DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

TEMA: IMPLEMENTAR UN ESQUEMA DE TELE-OPERACIÓN PARA UN ROBOT MANIPULADOR MÓVIL MANIOBRADO A TRAVÉS DE DISPOSITIVOS HÁPTICOS, PARA INCREMENTAR LA TRANSPARENCIA DEL SITIO REMOTO, A TRAVÉS DE ENTORNOS DE REALIDAD VIRTUAL Y REALIDAD AUMENTADA, EN EL SITIO LOCAL

AUTOR: WASHINGTON XAVIER QUEVEDO PÉREZ

DIRECTOR: ING. VÍCTOR HUGO ANDALUZ, PH.D.

LATACUNGA

2016

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Certifico que el proyecto de investigación**, "IMPLEMENTAR UN ESQUEMA DE TELE-OPERACIÓN PARA UN ROBOT MANIPULADOR MÓVIL MANIOBRADO A TRAVÉS DE DISPOSITIVOS HÁPTICOS, PARA INCREMENTAR LA TRANSPARENCIA DEL SITIO REMOTO, A TRAVÉS DE ENTORNOS DE REALIDAD VIRTUAL Y REALIDAD AUMENTADA, EN EL SITIO LOCAL"** realizado por **WASHINGTON XAVIER QUEVEDO PÉREZ**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar a **WASHINGTON XAVIER QUEVEDO PÉREZ** para que lo sustente públicamente.

Latacunga, agosto de 2016

Ing. Víctor Hugo Andaluz, Ph.D.

DIRECTOR

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORÍA DE RESPONSABILIDAD

Yo, **Washington Xavier Quevedo Pérez**, con cedula de identidad N°0503144172 declaro que el presente proyecto de investigación**, "IMPLEMENTAR UN ESQUEMA DE TELE-OPERACIÓN PARA UN ROBOT MANIPULADOR MÓVIL MANIOBRADO A TRAVÉS DE DISPOSITIVOS HÁPTICOS, PARA INCREMENTAR LA TRANSPARENCIA DEL SITIO REMOTO, A TRAVÉS DE ENTORNOS DE REALIDAD VIRTUAL Y REALIDAD AUMENTADA, EN EL SITIO LOCAL"** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, agosto de 2016

Washington Xavier Quevedo Pérez
C.C.: 0503144172

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA DE ELECTRÓNICA E INSTRUMENTACIÓN

AUTORIZACIÓN

Yo, **WASHINGTON XAVIER QUEVEDO PÉREZ**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la Biblioteca Virtual de la institución el presente trabajo de titulación **"IMPLEMENTAR UN ESQUEMA DE TELE-OPERACIÓN PARA UN ROBOT MANIPULADOR MÓVIL MANIOBRADO A TRAVÉS DE DISPOSITIVOS HÁPTICOS, PARA INCREMENTAR LA TRANSPARENCIA DEL SITIO REMOTO, A TRAVÉS DE ENTORNOS DE REALIDAD VIRTUAL Y REALIDAD AUMENTADA, EN EL SITIO LOCAL"** cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Latacunga, agosto de 2016

Washington Xavier Quevedo Pérez
C.C.: 0503144172

DEDICATORIA

Dedicado a mi madre, quien supo guiarme por el buen camino, alentarme a seguir adelante y no desmayar en los problemas que se presentaban, enseñándome a encarar las adversidades sin desfallecer en el intento.

AGRADECIMIENTO

A mi tutor, quien ha hecho lo posible por incluirme en el fantástico proceso de investigación con paciencia y dedicación.

A los integrantes del laboratorio de investigación, por su valiosa colaboración que hizo posible la experimentación del presente proyecto.

A la Universidad de las Fuerzas Armadas ESPE Extensión Latacunga, que con sus instalaciones y metodología ha hecho posible que pueda lograr una meta profesional en el interminable camino por alcanzar el conocimiento.

Al Consorcio Ecuatoriano para el Desarrollo de Internet Avanzado -CEDIA-, Universidad de las Fuerzas Armadas ESPE, Universidad Técnica de Ambato y a la Escuela Superior Politécnica de Chimborazo por el financiamiento del proyecto *Tele-operación Bilateral Cooperativo de Múltiples Manipuladores Móviles* CEPRAIX-2015-05 y el apoyo para el desarrollo del presente trabajo.

ÍNDICE DE CONTENIDO

CAPÍTULO III

CAPÍTULO IV

CAPÍTULO V

CAPÍTULO VI

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente proyecto de investigación tiene el propósito de implementar un esquema de tele-operación para un robot manipulador móvil maniobrado a través de dispositivos hápticos, para incrementar la transparencia del sitio remoto, a través de entornos de realidad virtual y aumentada, en el sitio local, permitiendo a un operador humano para llevar a cabo complejo tareas en entornos remotos. En el esquema de tele-operación se propone que el operador humano esté inmerso en un entorno de realidad aumentada para tener una mayor transparencia del sitio remoto. La transparencia de un tele-operación sistema indica una medida de cómo el humano siente el sistema remoto. En el sitio local es implementado un entorno de realidad aumentada desarrollada en Unity3D, que a través de dispositivos de entrada recrea las sensaciones que el operador humano sentiría si estuviera en el sitio remoto, por lo que se considera los sentidos de la vista, el tacto y el oído. Estos sentidos ayudan al operador humano para "transmitir" su habilidad y experiencia para realizar una tarea. Finalmente, los resultados experimentales se indican para verificar el rendimiento del esquema propuesto.

PALABRAS CLAVE:

- **TELEOPERACIONES**
- **REALIDAD VIRTUAL**
- **DISPOSITIVOS HÁPTICOS**

ABSTRACT

This work presents the implement a scheme tele-operation for a mobile manipulator robot maneuvered through haptic devices, to increase the transparency of the remote site through environments of virtual and augmented reality, in the local site allowing a human operator to perform complex tasks in remote environments. In the tele-operation scheme it is proposed that the human operator is immersed in an augmented reality environment for greater transparency in the remote site. The transparency of a tele-operation system indicates a measure of how the human operator feels the remote system. An augmented reality environment developed in Unity3D is implemented at the local site, which through input devices recreates the sensations that the human operator would feel at the remote site, which is considered the senses of sight, touch and hearing. These senses assist the human operator to "transmit" their skills and experience to perform a task. Finally, experimental results are given to verify the performance of the proposed scheme.

KEYWORDS:

- **TELEOPERATIONS**
- **VIRTUAL REALITY**
- **HAPTIC DEVICES**

CAPÍTULO I

1. INTRODUCCIÓN

1.1 Planteamiento del problema

En contraposición de las aplicaciones clásicas de la robótica en ambientes industriales estructurados, los desarrollos actuales de esta disciplina están orientados a aplicaciones en escenarios parcialmente estructurados o variantes en el tiempo. Durante mucho tiempo, varios prototipos de robots han sido desarrollados para reemplazar a los humanos en tareas peligrosas, rescate, misiones de guerra, exploración en lugares confinados; sin embargo, un robot 100% autónomo es prácticamente imposible que realice una tarea específica en un ambiente no estructurado. Esto debido a la gran cantidad de variables que se generan en entornos dinámicos, desconocidos y variantes en el tiempo, razón por la cual es eminente la utilización de robots tele-operados.

Para tareas complejas tele-operadas (ambientes explosivos o de riesgo, manipulación de materiales peligrosos o misiones de rescate en caso de desastres naturales), donde sea necesario que un robot tenga las habilidades de locomoción y manipulación se utilizan los robots manipuladores móviles. Un robot manipulador móvil está conformado por una plataforma móvil con patas o ruedas en la cual se monta un brazo robótico tipo antropomórfico. Para un control coordinado de los movimientos del manipulador móvil es necesario contar con información de realimentación proveniente de sensores que determinen la localización exacta del manipulador móvil en el espacio de trabajo, o a su vez cámaras que muestren el entorno en el cual se desenvuelve el robot.

Los sistemas de control manual y tele-operados usados actualmente cuentan con pantallas de visualización que muestran al robot manipulador móvil y su entorno en tiempo real desde varios ángulos con el objetivo de ofrecer información suficiente al operador humano para su manipulación. Estos sistemas mejoran la transparencia del esquema de tele-operación. Por lo mencionado este trabajo tiene como objetivo implementar algoritmos de control en lazo cerrado para un robot manipulador móvil en un esquema tele-operación, en el cual en el sitio local se implemente una interfaz en realidad virtual y aumentada, para lograr mayor trasparencia del sitio remoto.

1.2 Formulación del problema

¿Cómo influye el uso de realidad virtual y realidad aumentada en el incremento de transparencia del sitio remoto de un esquema de tele-operación para un robot manipulador móvil?

1.3 Antecedentes

La tele-operación de robots implica manejar un robot a distancia para realizar alguna tarea dada. De esta forma se permite que un operador humano pueda “transportar” su capacidad y destreza hacia ambientes de trabajos remotos y/o peligrosos, minimizando los posibles riesgos asociados al ambiente, o más aún, se puede alcanzar lugares inaccesibles para el hombre (Carlson & Murphy, 2005). Así, se puede extender la inteligencia y experiencia del operador hacia el lugar remoto donde se encuentra el robot (Brady & Tarn, 2000).

Los sistemas de tele-operación están constituidos por: una estación o sitio local, donde se encuentra el operador humano que manipula un robot o dispositivo maestro generando comandos de control y además recibe información táctil de fuerza y/o visual de imagen; una estación de trabajo o sitio remoto, donde un robot esclavo (manejado por el operador humano

desde el sitio local) realiza una tarea determinada en interacción con un medio o entorno, por ejemplo: ensamblar un conjunto de piezas mecánicas, navegar en pasillos con personas, entre otras aplicaciones; y un canal de comunicación bidireccional, que vincula ambos sitios como lo muestra la Figura 1.

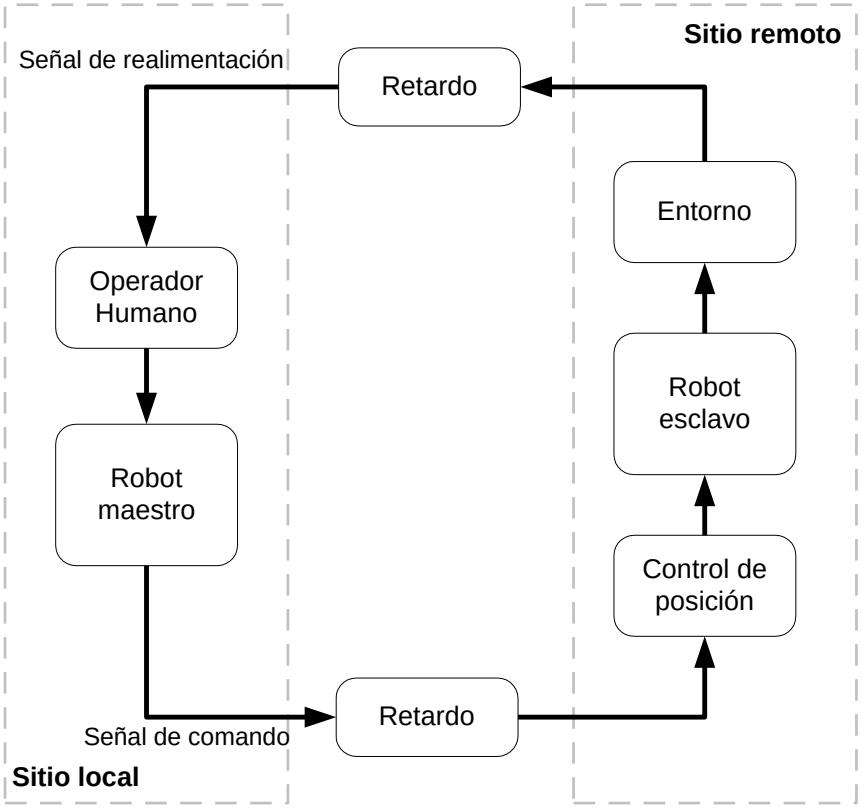


Figura 1. *Diagrama de bloques general de un sistema de tele-operación bilateral de un robot*

En tele-robótica, el operador humano aplica una fuerza intencional al sistema local (por ejemplo, un joystick) generando una serie de movimientos sobre el robot remoto. Además, existen tareas en las que, por su naturaleza, es muy importante, que el operador humano posea una sensación de presencia en el lugar remoto. Por ello, se incorpora en el lazo de control, tanto realimentación de fuerza como realimentación visual hacia el operador humano. El uso de realimentación de fuerza en robótica sirve para que el operador humano pueda sentir la fuerza de contacto entre el robot y algún objeto de su entorno, lo cual se manifiesta en un mayor sentido remoto del operador, como, por ejemplo: noción del peso a manipular.

La imagen adquirida a partir de una cámara se realimenta hacia el operador humano, de forma que éste pueda ver el movimiento del robot desplazándose en el ambiente de trabajo remoto y genere comandos de control para ser enviados hacia el robot. Un sistema de tele-operación en el cual el operador humano envía comandos a través de un sistema local hacia un sistema de operación remota, y recibe simultáneamente información proveniente desde el sitio remoto (por ejemplo, realimentación táctil de fuerza o realimentación visual), se denomina sistema de tele-operación bilateral debido a que la información fluye en forma bidireccional.

Los dispositivos hápticos son medios de entrada que permiten interactuar con variables remotas o virtuales (Hara, *et al*, 2004), con el fin de lograr mayor transparencia del sistema de control. Además, proporcionan la realimentación de fuerza, logrando así una sensación de sentido del tacto al sujeto que interactúa con el entorno.

La realidad virtual (RV) implica entorno de escenas u objetos de apariencia real generados por computadora, que crean en el usuario la sensación de estar dentro de dicho entorno. El operador humano puede experimentar la realidad virtual, así como la percepción de diferentes estímulos que intensifican la sensación de inmersión como: gravedad, velocidad, deformación, colisiones, texturas y demás; a través de un dispositivo conocido como HMD (Head Mounted Display) o casco de realidad virtual.

La aplicación de la tecnología de realidad virtual, aunque centrada inicialmente en el terreno del entretenimiento y de los videojuegos, se ha extendido a otros campos, como la medicina, arqueología, creación artística, entrenamiento militar, simuladores de vuelo, entre otros. La realidad aumentada (RA) define una visión a través de un dispositivo tecnológico de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real. La información sobre el mundo real alrededor del usuario se convierte en interactiva y digital como una forma de ampliar el mundo real.

Para la visualización de realidad aumentada es necesario el uso de pantallas, una cámara que capte el entorno y un controlador que superponga la información en los objetos reales captados por la cámara. La principal diferencia con la realidad aumentada es que, al añadir información virtual a la información física ya existente, es decir, añadir una parte sintética virtual a lo real no sustituye la realidad física, sino que sobreimprime los datos informáticos al mundo real.

El control gestual es un sistema de control basado en la interacción física con un periférico, su objetivo es ofrecer una experiencia más inmersiva que los mandos tradicionales a través de acciones motrices como agitar los brazos o inclinar las manos. Los sistemas de control gestual han estado presentes en gran medida en pantallas y paneles táctiles (touchpad) que han sido de gran utilidad al ofrecer funciones de atajos generalmente en dispositivos portátiles. El siguiente paso de estos sistemas de control gestual, es eliminar el contacto del órgano controlador con la superficie sensora, esta mejora se la puede obtener con el dispositivo Leap Motion provee un sistema de control gestual basado en una cámara infrarroja que detecta cuando un órgano controlador (mano, brazo) incide en su campo de visión, generando datos para la visualización del órgano controlador en realidad virtual y su posterior interacción con objetos virtuales.

El control gestual ofrece al usuario mayor inmersión en aplicaciones de realidad virtual y realidad aumentada al otorgar la capacidad de interactuar con objetos virtuales directamente con las manos. En conjunto con los dispositivos hápticos controladores gestuales y entornos en realidad virtual, se genera una sinergia de tecnologías para otorgar al operador un sistema de tele-operación transparente para el control de manipuladores móviles en sitios remotos o peligrosos.

1.4 Justificación e Importancia

En el esquema de tele-operación propuesto, el operador humano aplica una fuerza intencional al sistema local por medio de un dispositivo háptico, para generar una serie de movimientos sobre el robot manipulador móvil que se encuentra en el sitio remoto. Debido a que existen tareas en las que, por su naturaleza es muy importante, que el operador humano posea una sensación de presencia en el lugar remoto, se incorpora en el lazo de control, tanto la realimentación de los estados del robot como realimentación visual hacia el operador humano.

La realimentación de los estados del robot y la realimentación visual desde el sitio remoto a partir de una cámara, permite implementar un ambiente de realidad aumentada en el sitio local, de forma que el operador humano pueda ver el movimiento del robot desplazándose en el ambiente de trabajo. Mientras que el uso de un ambiente en realidad virtual permite simular el comportamiento del robot en un entorno predefinido como una prueba o demostración de algoritmos de control propuestos.

Para aumentar la transparencia del esquema de tele-operación se implementa una interfaz que interactúe con el operador humano a través del movimiento de las manos. La interacción a través de dispositivo LeapMotion permite al usuario navegar en el menú de opciones y elegir la información que se muestra en la pantalla en el modo de realidad virtual y realidad aumentada, otorgan así mayor inmersión y usabilidad al sistema propuesto.

Es importante indicar que este proyecto se encuentra enmarcado en el proyecto de investigación “Tele-Operación Bilateral Cooperativo de Múltiples Manipuladores Móviles”, ganador de la IX convocatoria de CEDIA-CEPRA 2015, en el cual la Universidad de las Fuerzas Armadas ESPE Extensión Latacunga lidera la coordinación del mismo. Los resultados obtenidos en el presente proyectos servirán como base para nuevas investigaciones

formativas y generativas, aplicadas a la robótica industrial o robótica de servicio y/o para otras áreas de ingeniería.

1.5 Objetivos

1.5.1 Objetivo General

- Implementar un esquema de tele-operación para un robot manipulador móvil maniobrado a través de dispositivos hápticos, para incrementar la transparencia del sitio remoto, a través de entornos de realidad virtual y realidad aumentada, en el sitio local.

1.5.2 Objetivos Específicos

- Investigar acerca de las técnicas de simulación de objetos en realidad virtual y realidad aumentada.
- Implementar en el sitio local una Interfaz de realidad virtual y realidad aumentada que permita maniobrar a través de un dispositivo háptico un robot manipulador móvil.
- Proponer un algoritmo de control en lazo cerrado que permita manipular el extremo operativo del manipulador móvil a través de del dispositivo háptico.
- Evaluar experimentalmente el esquema de tele-operación propuesto sobre el robot construido en el proyecto de investigación “Tele-Operación Bilateral Cooperativo de Múltiples Manipuladores Móviles”, ganador de la IX convocatoria de CEDIA-CEPRA 2015, en el cual la Universidad de las Fuerzas Armadas ESPE Extensión Latacunga lidera la coordinación del mismo.

1.6 Publicaciones

Los resultados del presente proyecto de investigación constan de tres artículos publicados en la serie de las Lecture Notes in Computer Science con ISSN: 03029743 y se presentan en el Anexo A. Los artículos publicados son los siguientes.

- Víctor H. Andaluz, **Washington X. Quevedo**, Fernando A. Chicaiza, José Varela, Cristian Gallardo, Jorge S. Sánchez and Oscar Arteaga, **"Transparency of a Bilateral Tele-Operation Scheme of a Mobile Manipulator Robot"**, Augmented Reality, Virtual Reality, and Computer Graphics - Lecture Notes in Computer Science, ISSN 0302-9743, Proceedings, Part I, pp 228-245, Switzerland, 2016.

- Víctor H. Andaluz, Jorge S. Sánchez, Jonnathan I. Chamba, Paúl P. Romero, Fernando A. Chicaiza, José Varela, **Washington X. Quevedo**, Cristian Gallardo and Luis F. Cepeda, **"Unity3D Virtual Animation of Robots with Coupled and Uncoupled Mechanism",** Augmented Reality, Virtual Reality, and Computer Graphics - Lecture Notes in Computer Science, ISSN 0302-9743, Proceedings, Part I, pp 89-101, Switzerland, 2016.

- Víctor H. Andaluz, Fernando A. Chicaiza, Cristian Gallardo, **Washington X. Quevedo**, José Varela, Jorge S. Sánchez and Oscar Arteaga, **"Unity3D-MatLab Simulator in Real Time for Robotics Applications"**, Augmented Reality, Virtual Reality, and Computer Graphics - Lecture Notes in Computer Science, ISSN 0302-9743, Proceedings, Part I, pp 246-263, Switzerland, 2016.

CAPÍTULO II

2. MARCO TEÓRICO

2.1 Antecedentes Investigativos

En el contexto del nivel de inmersión que ofrece la realidad virtual y realidad aumentada especialmente para aplicaciones en donde el usuario necesita retroalimentación de las variables que tiene en su entorno, y haciendo énfasis en aumentar el nivel de inmersión que tiene un sistema de tele-operación en el sitio remoto, surge la idea de acoplar los sistemas de tele-operación con realidad virtual y realidad aumentada.

Este nuevo sistema tendrá como punto fuerte el aumento drástico de inmersividad para el operador que se encuentra en el sitio local, con el manipulador móvil que se encuentra en el lugar remoto, se requieren realizar pruebas cuantitativas para comprobar el incremento de transparencia del sistema. Del análisis del estado del arte se destaca (Hamner, *et al*, 2010) el mismo que muestra el diseño y construcción de un manipulador móvil autónomo que supera efectivamente incertidumbres inherentes del sistema y excepciones mediante la utilización de estrategias de control que emplea un control coordinado, se realimenta de control visual y fuerza.

En (García, *et al*, 2014) se implementa un sistema de tele-operación multi-operador multi-robot, utilizando cuatro robots móviles reales y cuatro dispositivos maestros. Además, se presentan dos tipos de pruebas, para los que se llevaron a cabo experimentos con el fin de evaluar el rendimiento del método de prevención de colisiones y con relación al tema de dispositivos hápticos, en (Henao, *et al*, 2014) menciona el uso de dispositivos hápticos con realimentación de fuerza como método de entrada y control en prototipo del exoesqueleto. En (Andaluz, *et al*, 2011), menciona un sistema de tele-operación bilateral en el cual el operador humano envía comandos a través

de un sistema local hacia un sistema de operación remota, y recibe simultáneamente información proveniente desde el sitio remoto (realimentación táctil de fuerza y realimentación visual).

2.2 Fundamentación Teórica

El esquema de tele-operación que se propone requiere de múltiples dispositivos como: manipuladores móviles, dispositivos hápticos; conceptos teóricos de control, tele-operación; y tecnologías de inmersión como realidad virtual y realidad aumentada. Las especificaciones se describen a continuación.

2.2.1 Manipuladores Móviles

El término manipulador móvil es utilizado para referirse a los robots construidos por un brazo robótico (robot manipulador o brazo robótico) montado sobre una plataforma móvil (con ruedas o patas), los más populares son: manipulador móvil omni-direccional (KUKA), MM-500 SK (NEOBOTIX) y Pioneer 3AT con Pioneer Arm 5DOF (MOBILE ROBOTS) como se pueden observar en la Figura 2.

Figura 2. *Algunos ejemplos de manipuladores móviles*

Los manipuladores móviles, se caracterizan por tener un alto grado de redundancia, que combina la manipulación de un manipulador de base fija con la movilidad de una plataforma con ruedas o patas (Bayle, *et al*, 2003). Estos sistemas permiten realizar las misiones más habituales de los sistemas

robóticos que requieren tanto la capacidad de locomoción y manipulación (White & Bhatt, 2009; Andaluz, *et al*, 2012).

Entre los trabajos respecto a la tele-operación de un manipulador móvil se puede citar a (Farkhatdinov, *et al*, 2008) en el que presenta un estudio de viabilidad de un enfoque pasividad dominio del tiempo para tele-operación bilateral de un manipulador móvil; la propuesta método está validado por ejemplos de simulación. Una nueva tendencia en los sistemas de tele-operación es la conmutación de control de señales (Neo, *et al*, 2005) los esquemas de control propuestos basados en la computación de plataforma móvil con un manipulador han sido analizadas y validadas únicamente por simulación.

2.2.2 Tele-Operación

El termino tele-operación hace referencia a un conjunto de tecnologías que comprenden la operación o gobierno a distancia de un dispositivo por un ser humano. Por tanto, tele-operar es la acción que realiza un ser humano de operar o gobernar a distancia un dispositivo; mientras que un sistema de tele-operación será aquel que permita tele-operar un dispositivo, que se denominará dispositivo tele-operado (Alencastre, *et al*, 2003). Un sistema de tele-operación consta de los siguientes elementos, como se observa en la Figura 3:

- Operador o tele-operador: es un ser humano que realiza a distancia el control de la operación. Su acción puede ir desde un control continuo hasta una intervención intermitente, con la que únicamente se ocupa de monitorizar y de indicar objetivos y planes cada cierto tiempo.
- Dispositivo tele-operado: podrá ser un manipulador, un robot, un vehículo o dispositivo similar. Es la máquina que trabaja en la zona remota y que está siendo controlada por el operador.
- Interfaz: conjunto de dispositivos que permiten la interacción del operador con el sistema de tele-operación. Se considera al manipulador maestro como parte del interfaz, así como a los monitores

de vídeo, o cualquier otro dispositivo que permita al operador mandar información al sistema y recibir información del mismo.

- Control y canales comunicación: conjunto de dispositivos que modulan, transmiten y adaptan el conjunto de señales que se transmiten entre la zona remota y la local. Generalmente se contará con uno o varias unidades de procesamiento.
- Sensores: conjunto de dispositivos que recogen la información, tanto de la zona local como de la zona remota, para ser utilizada por el interfaz y el control.

Figura 3. *Elementos del esquema de tele-operación*

Los sistemas de tele-operación pueden ser multi-operador y/o multi-robot, permitiendo ejecutar tareas independientes y simultaneas, de manera cooperativa (Herbert, *et al*, 2003). Realizar tareas complementarias de forma simultanea es de suma importancia cuando se pretende reducir el tiempo de un objetivo común, que en diferentes situaciones puede ser extremadamente crítico, como, por ejemplo, el rescate de varias personas, el apagado de un incendio, entre otros. Por otra parte, desde el punto de vista de la seguridad, múltiples manipuladores móviles pequeños son más adecuados para la realización de varias tareas en el entorno humano que un manipulador móvil grande y pesado.

Actualmente, el estado del arte de los sistemas multi-operador multi-robot es diverso, en el cual son usados diferentes conceptos e ideas. Dentro de esta área se encuentran los trabajos de (Chong, *et al*, 2000) en los que propone un control asistido para colaboración remota, donde los operadores manipulan dos brazos robóticos con probabilidades de colisión; mientras (Lo, *et al*, 2014) propone un control remoto colaborativo a través de Internet con

reflexión de fuerza, donde un operador manipula un brazo robótico y otro controla un manipulador móvil; (Passenberg, *et al*, 2010) presenta una tele-operación colaborativa con realimentación de fuerza, donde un objeto es sujetado entre dos robots de un grado de libertad, los cuales son manipulados cada uno por un operador.

2.2.3 Realidad Virtual

La realidad virtual se podría definir como un sistema informático que genera en tiempo real representaciones de la realidad, que de hecho no son más que ilusiones ya que se trata de una realidad perceptiva sin ningún soporte físico y que únicamente se da en el interior de los ordenadores.

La simulación que hace la realidad virtual se puede referir a escenas virtuales, creando un mundo virtual que sólo existe en el ordenador de lugares u objetos que existen en la realidad. También permite capturar la voluntad implícita del usuario en sus movimientos naturales proyectándolos en el mundo virtual que estamos generando, proyectando en el mundo virtual movimientos reales. Además, permite experimentar completamente en un mundo virtual, desconectando los sentidos completamente de la realidad teniendo la sensación la persona que está dentro de que la realidad corresponde en el mundo virtual.

La realidad virtual fue ideada y desarrollada fallidamente en los años 90 para su uso específicamente en videojuegos como en la consola Virtual Boy de Nintendo; lamentablemente el concepto de inmersión no fue logrado por las propuestas debido a la poca capacidad de procesamiento de los dispositivos de aquella época y al alto costo al usuario final de los prototipos. Recientemente en 2012 mediante la empresa Oculus (Oculus, 2015), ha traído de vuelta al escenario tecnológico la realidad virtual con su HMD Oculus Rift, en gran medida debido al desarrollo tecnológico en pantallas portátiles de alta definición, procesadores y sensores de bajo coste de producción. La

versión usada en este desarrollo corresponde al Oculus Rift DK2 y se la puede ver en la Figura 4.

Figura 4. *Oculus Rift Development Kit Versión 2*

Las aplicaciones que en la actualidad encontramos de la realidad virtual a actividades de la vida cotidiana son muchas y diversas. Hay que destacar: la reconstrucción de patrimonios culturales, la medicina, la simulación de cualquier entorno y la sensación de presencia. La reconstrucción del patrimonio cultural consiste en la recuperación a través de la simulación de piezas únicas de la antigüedad que han sido destruidas o se encuentran degradadas. En algunas, a partir de unos pocos restos se pueden simular piezas enteras. Además, la realidad virtual permite mostrar la pieza en perfecto estado en diversos lugares del mundo a la vez, e incluso permite crear museos enteros con piezas virtuales. La aplicación en la medicina la encontramos en la simulación virtual del cuerpo humano. A partir de imágenes del cuerpo humano, se puede hacer la recreación en 3D del paciente, lo que facilita la elaboración de un diagnóstico, o la simulación de operaciones en caso que sea necesario.

2.2.4 Dispositivos Hápticos

Un dispositivo háptico permite a un usuario tocar, sentir, manipular, crear, y cambiar objetos tridimensionales simulados dentro de un ambiente virtual. Un dispositivo háptico añade el sentido del tacto a la experiencia virtual

y buscan aplicar el sentido del tacto a la interacción humana con sistemas informáticos.

Los dispositivos Hápticos proporcionan la realimentación de fuerza (producción mecánica de información sensorial por el sistema kinésico, es decir por la sensación del movimiento, sensaciones originadas en el músculo, tendones y uniones) al sujeto que interactúa con entornos virtuales o remotos. Tales dispositivos trasladan una sensación de presencia al operador. El usuario no sólo envía la información a la computadora, sino que también puede recibir la información de la computadora en forma de una sensación sobre alguna parte del cuerpo.

Estos dispositivos se pueden utilizar en varios paradigmas de interacción, pero con el que más encaja es con el de la realidad virtual. Estos dispositivos permiten un control más natural sobre el entorno, permitiendo al usuario tocar y sentir objetos virtuales. Con este dispositivo las aplicaciones de software que proporcionan simulaciones de realidad virtual han cobrado más vida en muchos entornos como es el caso del aprendizaje, permitiendo entrenamientos especializados (por ejemplo, cirujanos, astronautas, en cuanto al aprendizaje de la mecánica de la habilidad a entrenar), Aprendizaje de conceptos docentes (por ejemplo, "el sentir" de cómo las moléculas atraen o rechazan distintos átomos, de manera que una sensación táctil puede incrementar el nivel de comprensión), modelado de objetos tridimensionales sin un medio físico, entre otros.

Actualmente éstos dispositivos proporcionan varias desventajas o inconvenientes, de los cuales se puede destacar: su alto costo, su complejidad a la hora de diseñar o fabricar estos dispositivos, la ausencia de estándares que permitan un desarrollo eficaz de estos dispositivos, la poca familiarización de la de los usuarios con estos dispositivos. El dispositivo háptico a utilizarse en el presente desarrollo se denomina Falcon, desarrollado por la empresa Novint. Es un periférico destinado a sustituir el ratón en los videojuegos y otras aplicaciones de ordenador como lo muestra la Figura 5. El dispositivo Falcon tiene asas extraíbles, o agarres, que el usuario se sujeta para controlar el

Falcon. A medida que el usuario mueve el agarre en tres dimensiones (derecha-izquierda y hacia delante, hacia atrás, como un ratón, sino también arriba-abajo, a diferencia de un ratón), el software de Falcon mantiene un seguimiento de dónde se mueve la empuñadura y crea fuerzas de realimentación que el usuario puede sentir, mediante el envío de señales eléctricas a los motores que conforman el dispositivo.

Figura 5. *Novint Falcon*

Los sensores del dispositivo pueden realizar un seguimiento de la posición de la empuñadura a una resolución de 400dpi, y los motores están actualizados a una frecuencia de 1000 veces por segundo (1 kHz), dando una sensación realista y de solidez al contacto con superficies de objetos virtuales, y además pueden tener texturas detalladas. El peso y la dinámica de los objetos se pueden simular de manera que la inercia y el impulso de un objeto se pueden sentir de forma real. Las acciones e interacciones de un personaje en un juego pueden ser “sentidas”, como el retroceso de un arma de fuego al dispararla, el movimiento de un palo de golf, las aceleraciones de un coche, entre otras. El dispositivo se puede integrar perfectamente con la mayoría de los periféricos informáticos actuales, dado que estos dispositivos solo intentan introducir el sentido del tacto para interaccionar con la máquina y no intentan sustituir otros dispositivos actuales que normalmente utilizan otros sentidos diferentes al tacto para la interacción hombre-máquina.

2.2.5 Control Gestual

El "Reconocimiento de Gestos" es un tema en ciencias de la computación y la tecnología del lenguaje con el objetivo de interpretar gestos humanos a través de algoritmos matemáticos. Los gestos pueden ser cualquier movimiento corporal o estado, pero comúnmente se originan en el rostro y la mano. Enfoques actuales en el campo incluyen reconocimiento de las emociones faciales y de gestos manuales, haciendo uso de cámaras, procesamiento digital de imágenes y algoritmos para interpretar el lenguaje de señas. Sin embargo, la identificación y el reconocimiento de la postura, la marcha, la prosémica, y los comportamientos humanos es también el tema de técnicas de reconocimiento de gestos.

El reconocimiento de gestos puede ser visto como una manera para que las computadoras empiecen a entender el lenguaje corporal humano, construyendo así una relación más cercana entre máquinas y seres humanos, dejando atrás sistemas primitivos como las interfaces de usuario de texto o incluso GUIs (interfaces gráficas de usuario), que aún limitan la mayoría de las entradas informáticas al teclado y mouse. Además, permite a seres humanos comunicarse con la máquina e interactuar naturalmente sin dispositivos mecánicos. Utilizando el concepto de reconocimiento de gestos, es posible usar los dedos en un espacio de trabajo para relacionar los movimientos del usuario con el movimiento del cursor. Esto podría hacer que los dispositivos convencionales de entrada, tales como ratón, teclados e incluso pantallas táctiles sean innecesarios y en un futuro no muy lejano sean reemplazados en su totalidad.

El dispositivo Leap Motion (Leap Motion SDK Documentation, 2015), es un periférico de entrada, que está diseñado para ser colocado en un escritorio físico hacia arriba o instalado en un HMD de realidad virtual. Funciona mediante el uso de dos cámaras monocromáticas que trabajan en la longitud de onda de 850nm y tres LEDs infrarrojos, que proporcionan la luz infrarroja que será captada por las cámaras. El dispositivo observa un área

semiesférica de 61cm de radio, a una distancia de trabajo de 7-25 cm desde el dispositivo. La velocidad de captura es aproximadamente de 200fps, que luego son enviados a través de una interface USB al ordenador principal para ser renderizada por el software del controlador LeapMotion tal como se puede ver en la Figura 6.

Figura 6. *Dispositivo Leap Motion ejecutándose en una laptop*

El área de observación de alta y baja resolución del dispositivo diferencia el producto del conocido Kinect, el cual es más adecuado como seguimiento del cuerpo en una zona de trabajo mayor, como una sala de estar. El salto en la usabilidad de LeapMotion como método de entrada en ordenadores se demostró al realizar tareas como navegar por un sitio web, el uso de gestos de zoom en los mapas, dibujos de alta precisión, y la manipulación de visualizaciones de modelos en 3D, fue un éxito total y una experiencia innovadora para los nuevos usuarios.

Leap Motion desde el año 2013 ha distribuido miles de unidades para los desarrolladores que están interesados en crear aplicaciones para el dispositivo. Esta tecnología no sólo reduce el impacto del hardware en el sistema, también aumenta el rango de usos aplicables desde un objeto en el mundo físico a un objeto en el mundo digital. El uso de esta tecnología puede crear una nueva tendencia en la fabricación de un nuevo hardware con

LeapMotion integrado, ahorrando la necesidad de adquirir métodos de entrada tradicionales.

2.2.6 Motor Gráfico Unity

Unity es un motor de videojuegos multiplataforma creado por Unity Technologies, está disponible como plataforma de desarrollo para Windows y OS X, y permite crear juegos para las plataformas: Windows, OS X, Linux, Xbox 360, PlayStation 3, PlayStation Vita, Wii, Wii U, iPad, iPhone, Android y Windows Phone y WebGL. El motor gráfico utiliza Direct3D (en Windows), OpenGL (en Mac y Linux), OpenGL ES (en Android y iOS), e interfaces propietarias (en el caso de Wii). Tiene soporte para mapeado de relieve, reflexión de mapeado, mapeado por paralaje, pantalla de espacio oclusión ambiental (SSAO), sombras dinámicas utilizando mapas de sombras, render a textura y efectos de post-procesamiento de pantalla completa en el entorno de desarrollo como se puede ver en la Figura 7.

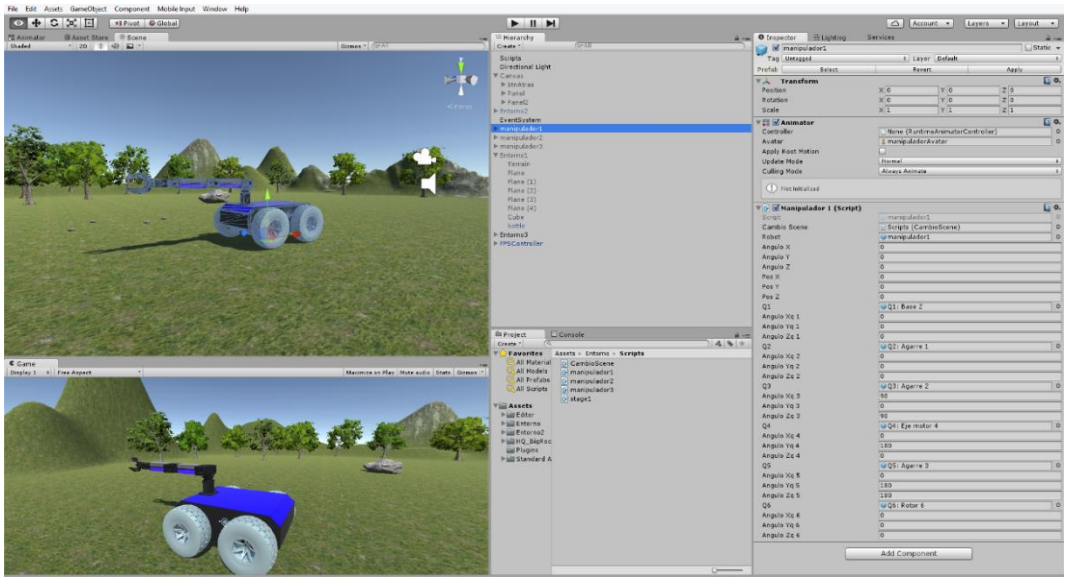


Figura 7. *Entorno de desarrollo de Unity*

Unity es la herramienta de creación orientada a videojuegos más popular del momento, debido a que ofrece integración nativa o por plugins con gran cantidad de dispositivos de entrada. Con el auge de la realidad virtual, realidad aumentada y tecnologías de control inalámbricas, y una gran

comunidad de desarrolladores, Unity se ha ido convirtiendo en la herramienta de desarrollo por excelencia al momento de crear experiencias visuales y videojuegos.

Entre las ventajas que ofrece Unity, también incluye el Unity Asset Server, como se puede ver en la Figura 8, que ofrece a los desarrolladores recursos como modelos 3D, scripting, proyectos, audio, creados por la comunidad de desarrolladores de Unity para su uso gratuito y también de pago. Con esta funcionalidad el desarrollador se concentra en crear su proyecto y deja a un lado la problemática del diseño, funciones especiales o incluso inteligencia artificial aplicable en su proyecto.

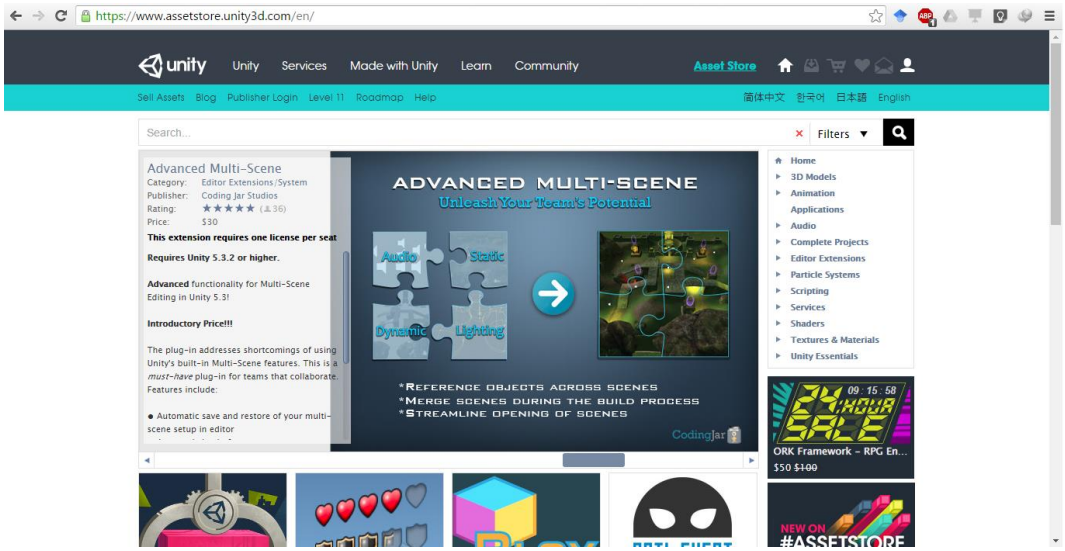


Figura 8. *Interfaz de Unity Assets Store*

CAPÍTULO III

3. DESARROLLO DE LA INTERFAZ

El desarrollo de la interfaz para el sitio local exige la convergencia de dispositivos, y tecnologías que en primera instancia no son compatibles entre sí, y más aún después de la investigación a fondo de las funciones específicas que benefician al presente trabajo se ha desarrollado de forma individual con cada dispositivo (Oculus Rift, LeapMotion, Novint Falcon), para luego converger el funcionamiento en tiempo real de los dispositivos, superando posibles errores al momento de empaquetar todo el desarrollo en una aplicación de escritorio. Así entonces se ha subdividido esta sección en dispositivos de control, interfaz de usuario y compatibilidad de tecnologías

3.1 Dispositivos de control

En esta sección se hará una explicación del hardware que interviene en el sitio local donde se encuentra el operador, estos dispositivos ofrecerán todas sus ventajas con la cometida de ofrecer mayor transparencia al operador humano. El punto neurálgico del desarrollo es innegablemente el motor gráfico Unity, debido a que es el entorno de desarrollo en el cual “suceden” los eventos virtuales (Realidad Virtual) y reales (Realidad Aumentada), por consecuencia se explicará el proceso de conexión de los dispositivos: Oculus Rift, LeapMotion y Novint Falcon al entorno de desarrollo de Unity. Es necesario mencionar que la versión de Unity que se utilizará en el presente trabajo de investigación es la 5.3.5f1, con el fin de evitar problemas de compatibilidad al usar versiones anteriores del entorno que para ciertos procedimientos no aplica lo que se explicará a continuación

3.1.1 Integración de Oculus Rift en Unity

Antes de entrar en la conexión entre Oculus Rift y Unity, es necesario instalar y configurar Oculus Rift en el ordenador, lo cual se detalla en el Anexo C. La integración de Oculus Rift con Unity se da de forma nativa desde la versión 5.1 de Unity, es decir, no se necesita de plugins externos o Unity Packages[1] para lograr su uso en el entorno. La versión del runtime para Oculus Rift es la V0.8.0.0 por ofrecer estabilidad en el ordenador utilizado.

Para comprobar la integración de Oculus Rift en el entorno de desarrollo de Unity es necesario ingresar en Edit>Project Settings>Player, y en la pestaña inspector aparece un menú de configuración del proyecto. En la sección “Other Settings” seleccionar la opción “Virtual Reality Supported”, tal como se muestra en la Figura 9.

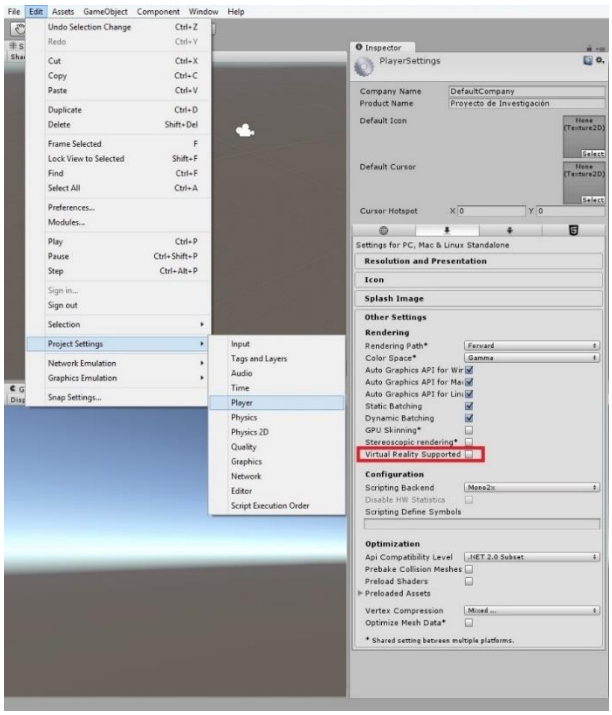


Figura 9. *Activación del soporte para Realidad Virtual en Unity.*

El último paso es ejecutar la escena y colocar el HMD en la cabeza del usuario para observar el ambiente creado en el entorno de desarrollo de Unity.

[1] Unity Packages, son paquetes comprimidos de recursos de un proyecto de Unity que se pueden importar fácilmente a cualquier escena de Unity.

Al mover la cabeza, el tracker[2] detecta el cambio de posición del HMD y lo representa modificando la posición de la cámara del proyecto.

3.1.2 Integración de LeapMotion a Unity

De la misma forma que sucedió en el apartado anterior, antes de configurar la integración, se necesita instalar el dispositivo LeapMotion en el ordenador, lo cual se detalla en el ANEXO C.

Para lograr la integración del dispositivo LeapMotion en el entorno de desarrollo de Unity es necesario descargar el Unity Core Assets[3], el cual es un Unity package que contiene los scripts[4], plugins[5] y prefabs[6] necesarios para ejecutar escenas que usen los servicios del dispositivo LeapMotion. Para lograrlo se debe importar el Unity package descargado de la página de LeapMotion, desde el menú: Assets>Import Package>Custom Package, tal como lo muestra la Figura 10.

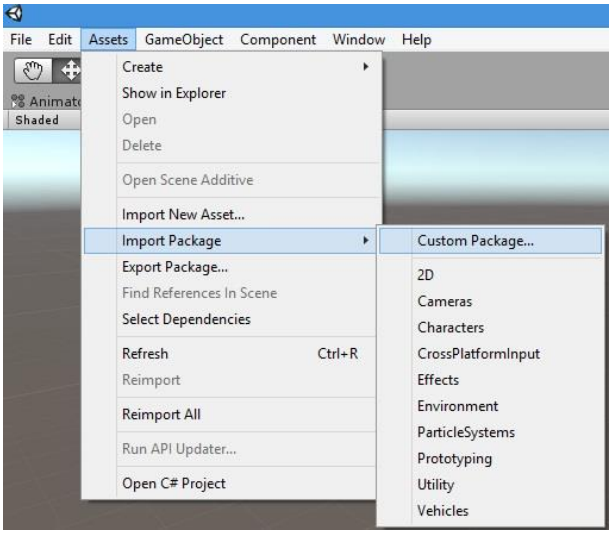


Figura 10. *Importación de un Unity Package.*

[2] Dispositivo que registra los movimientos en el espacio del HMD Oculus Rift
[3] Recurso descargado desde https://developer.leapmotion.com/unity
[4] Archivos con código de programación.
[5] Archivos o dll que contienen librerías que no existen en el API de Unity
[6] Objetos estándar de Unity que facilitan la creación de nuevos objetos igual o más complejos

Localizar el package descargado e importar. En el navegador de proyecto acceder a la dirección Assets>LeapMotion>Scenes y seleccionar Leap_Hands_Demo como se muestra en la Figura 11.

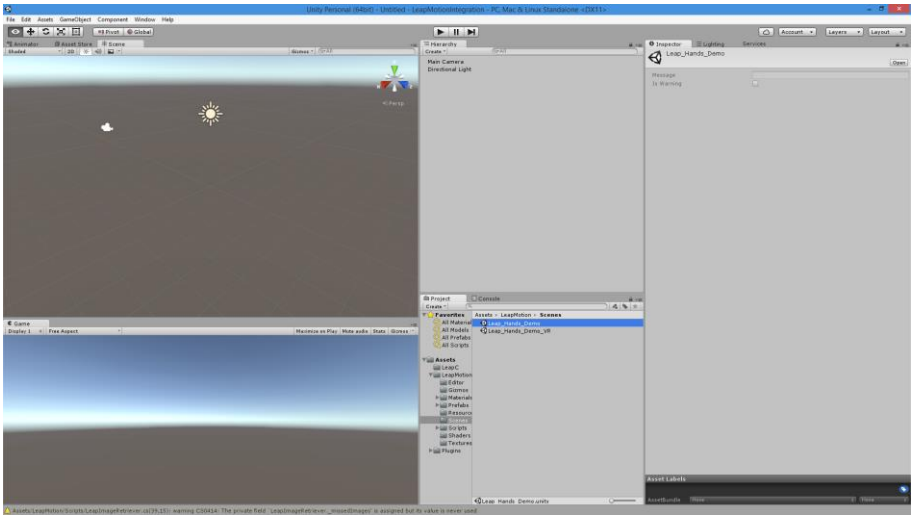


Figura 11. *Selección del demo de LeapMotion en Unity*

En la pestaña scene se observan objetos o GameObjects que simulan manos humanas, para ejecutar la escena se debe asegurar que el controlador de LeapMotion esté ejecutándose en backgroud, para comprobarlo, se debe hacer clic en el icono de LeapMotion y leer el mensaje "*Leap Motion El controlador está encendido"*, tal como se observa en la Figura 12.

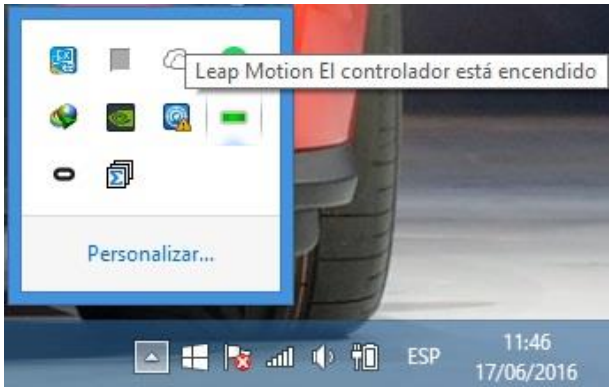


Figura 12. *Controlador encendido de LeapMotion.*

Clic en el botón play de Unity y colocar las manos del usuario sobre el sensor a una distancia aproximada de 10-20 cm, y mover las manos de tal forma que el modelo 3D de Unity se mueva de forma continua con las manos del usuario, tal como se observa en la Figura 13.

Figura 13. *Funcionamiento de LeapMotion en Unity*

3.1.3 Conexión de Falcon a Unity

Después de instalar el software oficial como se detalla en el Anexo C, se debe descargar el Unity Package de Falcon[7], e inmediatamente importarlo en un proyecto nuevo de Unity. En la carpeta scenes, abrir la escena “movimiento” y ejecutar el proyecto presionando el botón play tal como lo muestra la Figura 14.

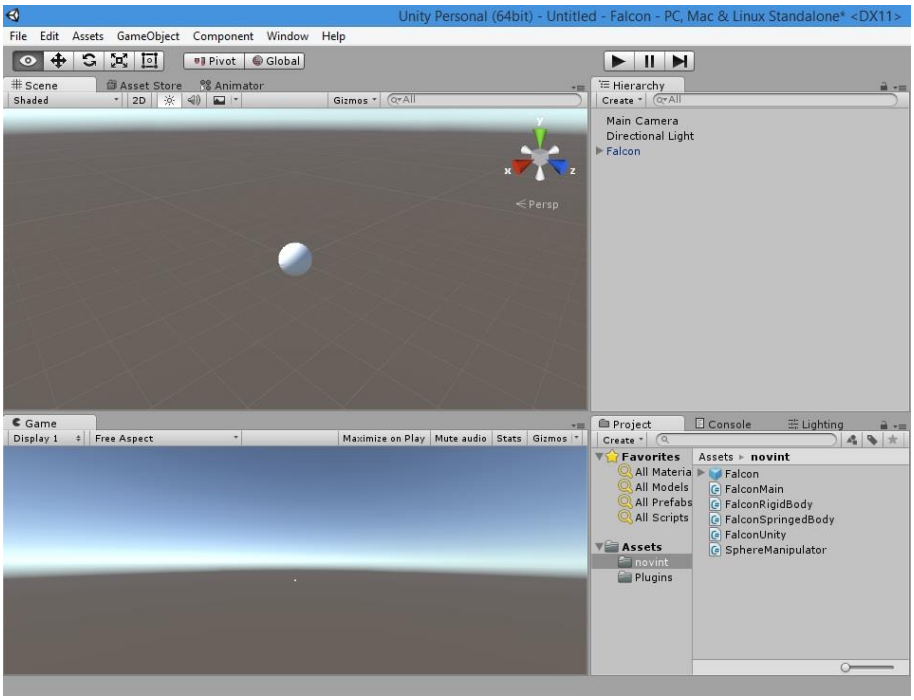


Figura 14. *Falcon integrado a Unity*

[7] El recurso fue descargado desde el siguiente enlace https://github.com/kbogert/falconunity

Al mover el extremo controlador de Falcon, la esfera que se aprecia en la scene de Unity debe moverse en los respectivos ejes que el usuario mueve el extremo operativo del dispositivo Falcon.

3.2 Interfaz de usuario

En este apartado se explicará el proceso que se necesita para crear y animar los modelos 3D, crear entornos virtuales, y desarrollar la interfaz de usuario.

3.2.1 Importación de modelos 3D

El modelo generalmente se obtiene a partir de un diseño CAD en SolidWorks, el cual asegura el funcionamiento cinemático sin problemas. Es necesario guardarlo como un archivo de ensamblaje (*.ASM). En la Figura 15 se puede observar la importación de la plataforma móvil en un formato no jerarquizado (formato por defecto de SolidWorks),

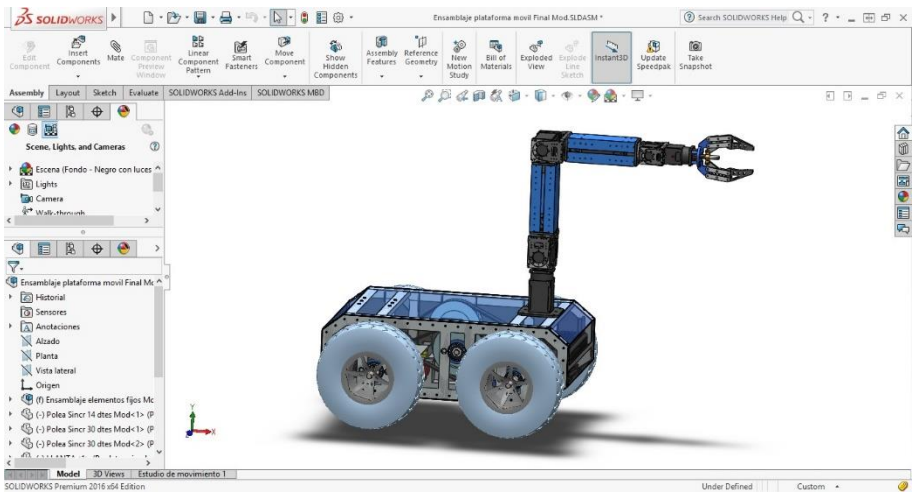


Figura 15. *Modelo 3D del manipulador móvil en SolidWorks*

A continuación, se importa el modelo en formato *.asm en el programa 3ds Max, con el objetivo de establecer jerarquías de cada uno de los elementos que conforman el modelo 3D. La organización de los componentes

mediante jerarquías asegura la correcta compatibilidad de movimiento cinemático que tendrá el modelo, dependiendo del funcionamiento del robot, este proceso se lo puede observar en la Figura 16.

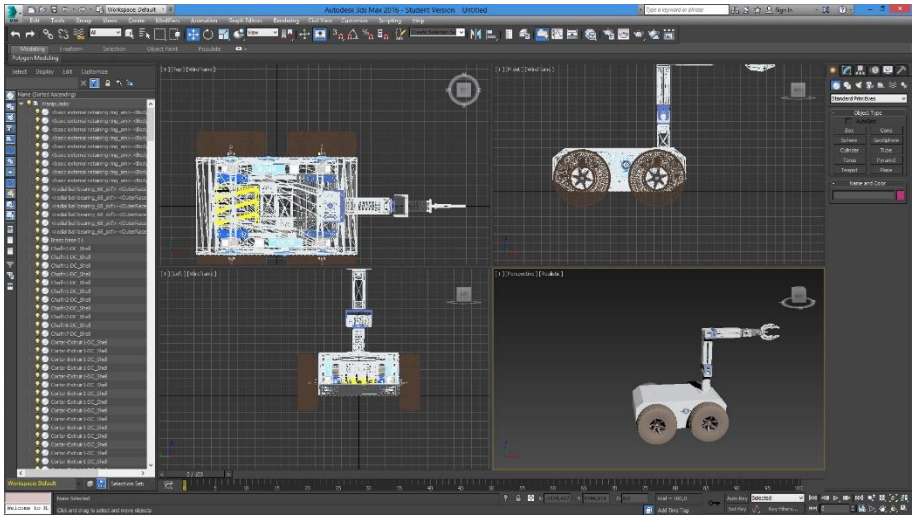


Figura 16. *Modelo 3D del manipulador móvil en 3DMax*

Después de este proceso, se debe exportar el modelo en formato *.fbx, para posteriormente importarlo en Unity, como se puede observar en la Figura 17. El proceso de agregar texturas al modelo se lo realiza dentro del entorno de Unity con el fin de observar en tiempo real el estado del modelo y que los recursos empleados sean compatibles con Unity.

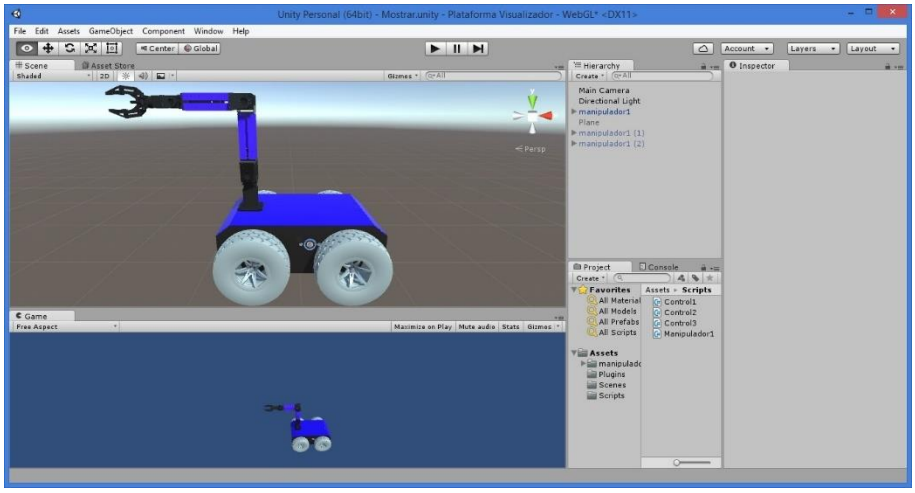


Figura 17. *Modelo 3D del manipulador móvil en Unity.*

3.2.2 Movimiento de modelos 3D

Para darle movimiento al modelo 3D, tal y como lo haría un modelo en la vida real, se analizará su modelo cinemático. En la mayoría de modelos mecánicos y robóticos existen dos tipos de movimiento predominantes (Andaluz, y otros, 2016b) el movimiento por punto de rotación y el movimiento por rigging.

3.2.2.1 Movimiento por Punto de Rotación

Este tipo de movimiento se caracteriza por mover independientemente cada articulación del modelo 3D, sin tener en cuenta el movimiento de otra articulación, aunque se tiene que tomar en cuenta que el script que controla la totalidad del modelo tomara en cuenta todos y cada uno de los puntos de rotación para lograr el objetivo del modelo 3D (locomoción, tarea de control, entre otros). Dentro de este tipo de movimiento se encuentran los robots que tienen motores en cada articulación como (humanoides, brazos robóticos, vehículos y similares).

En la Figura 18 se muestra el modelo 3D de un brazo robótico que forma parte del manipulador móvil en el cual se dará movimiento por puntos de rotación. Se debe configurar un script de Unity para determinar el movimiento que tendrá el brazo robótico. El script se presenta en el Anexo B, en el cual se observa los límites y sentido de movimiento que tiene cada articulación simbolizada como Q1, Q2, Q3, Q4, Q5 y Q6, debido a los 6 DOF que tiene el modelo del brazo robótico. El movimiento se lo realiza variando el valor del Q deseado, correspondiente a la “articulación” deseada.

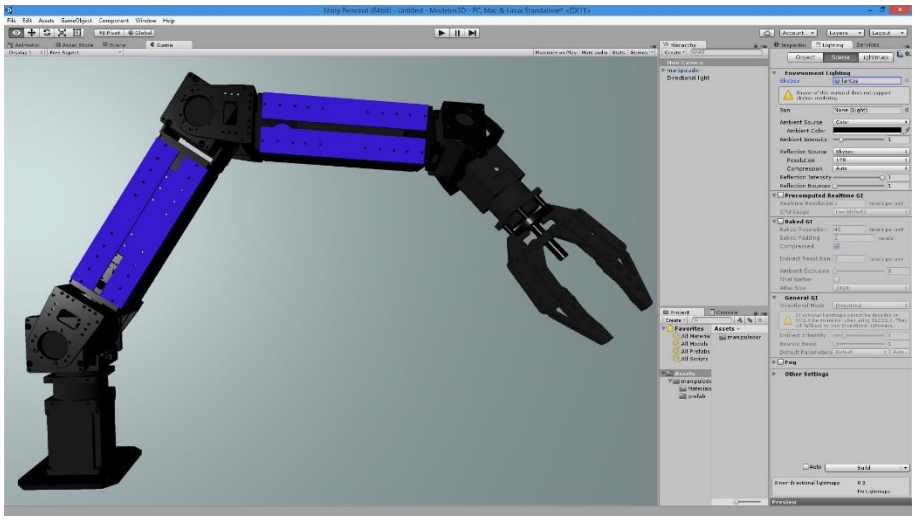


Figura 18. *Movimiento del brazo robótico en Unity*

En el apartado 3.3.1 se dará a conocer el método por el cual desde MatLab se puede controlar al brazo robótico con un script de control mucho más elaborado.

3.2.2.2 Movimiento por Rigging

Este tipo de movimiento se usa para lograr el modelo cinemático de robots que se mueven mediante transmisión de movimiento desde un motor pasando por articulaciones hasta llegar a sus extremos operativos. El caso más representativo para agregar este tipo de movimiento es el mecanismo Theo Jansen, el cual mediante un motor es capaz de mover tres pares de patas por lado, tal como se observa en la Figura 19.

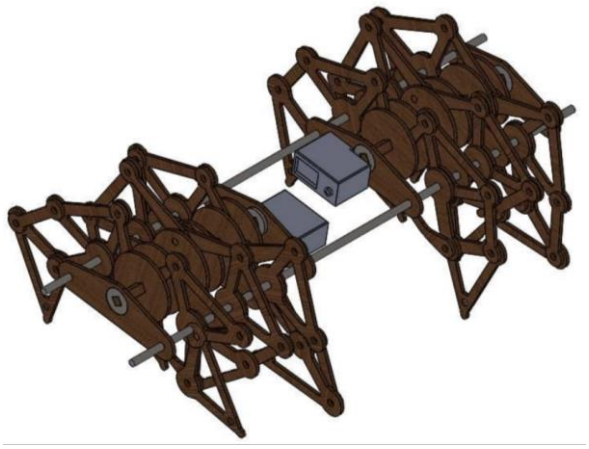


Figura 19. *Mecanismo Theo Jansen*

Para otorgarle movimiento es necesario que el programador entienda cómo se produce la locomoción del modelo, y utilizando la aplicación de escritorio de blender se procede a ubicar “huesos” en cada elemento que conforma la estructura que transmite movimiento desde el motor hasta el extremo operativo o pata del modelo 3D como se observa en la Figura 20.

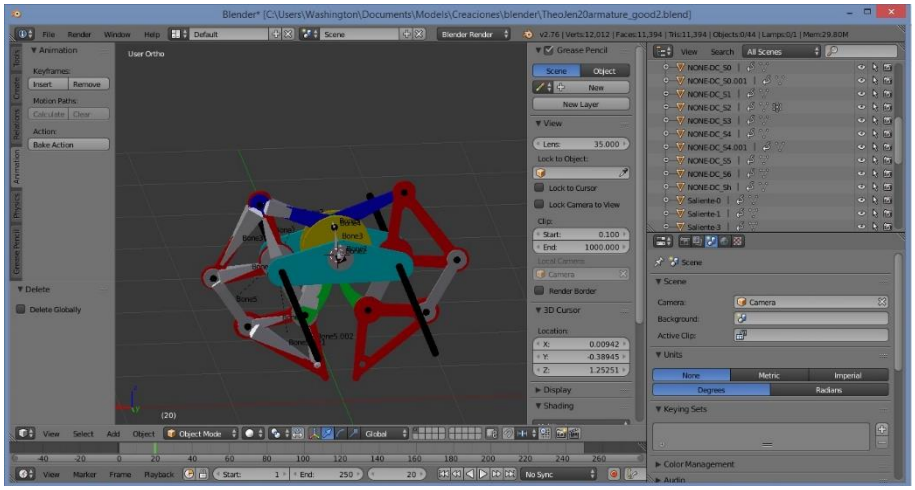


Figura 20. *Rigging del mecanismo Theo Jansen en Blender*

El movimiento del modelo 3D en Unity, necesita crear una secuencia de animación en blender. Esta animación dependerá de la cinemática del modelo, por ejemplo: si se necesita precisión gradual en el movimiento del modelo, se realizará la captura de posiciones cada grado sexagesimal de movimiento del motor, por otro lado, si se necesita que el modelo no se mueva con precisión gradual, sino que lo importante es la locomoción, se realizará la captura de ciertas posiciones por revolución, es decir cada 90 grados. La captura de posiciones en la construcción de la animación de movimiento se muestra en la Figura 21.

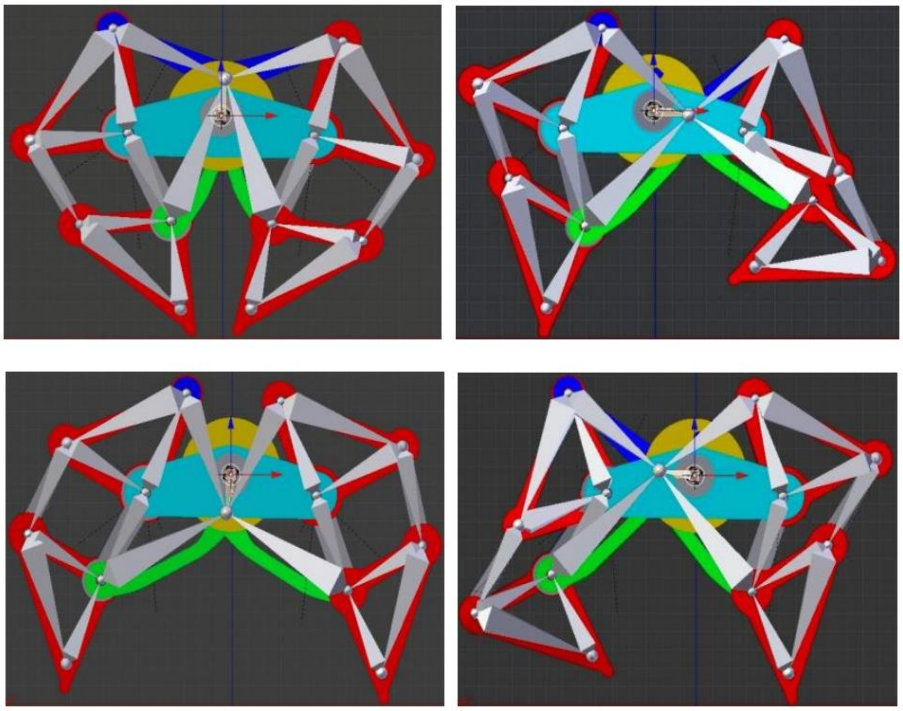


Figura 21. *Animación del mecanismo Theo Jansen*

El siguiente paso es importar a Unity el modelo con su animación incluida en formato *.fbx tal como lo muestra la Figura 22

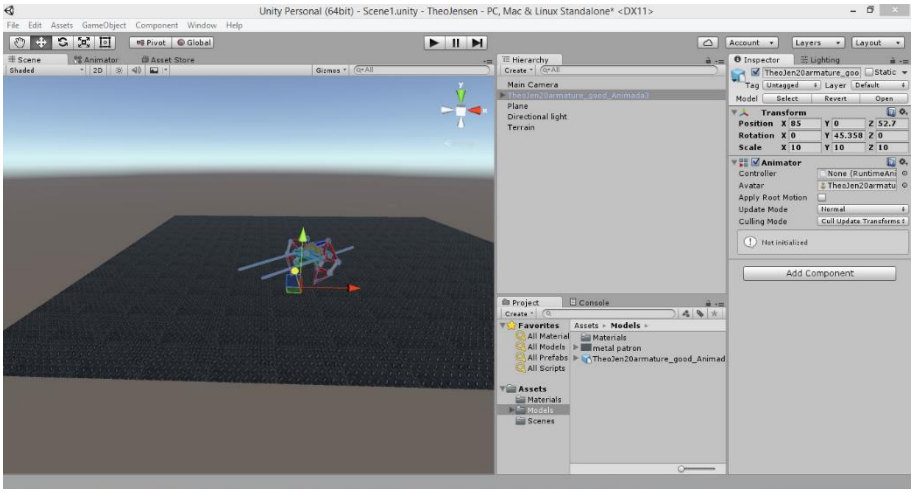


Figura 22. *Modelo 3D del mecanismo Theo Jansen en Unity*

3.2.3 Creación de entornos en Unity

Unity ofrece al usuario una herramienta para crear entornos en campo abierto, el GameObject de tipo Terrain cumple esta función. Se lo puede

encontrar en el menú GameObject>3DObject>Terrain. Tal como se muestra en la Figura 23.

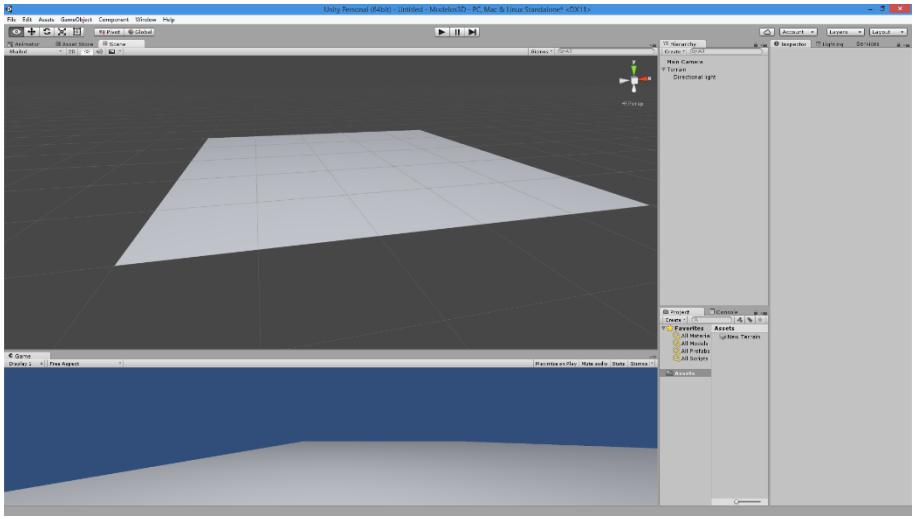


Figura 23. *Componente terreno en Unity*

Unity tiene packages que contiene materiales, texturas que darán más realismo al terreno que esta por crearse, para adquirir estos recursos es necesario importar el Package Environment que se encuentra disponible al ingresar en Assets>Import Package>Environment, tal como lo muestra la Figura 24.

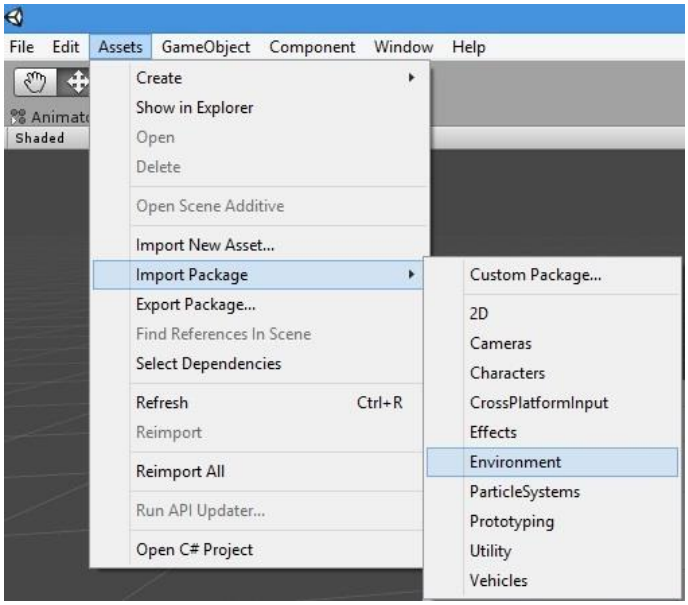


Figura 24. *Importación del Unity Package de Environment*

Para agregar detalles al terreno como deformaciones, texturas, árboles y tipo de suelo se utilizan las opciones del GameObject “Terrain” que se encuentra en la pestaña Inspector, como lo ilustra la Figura 25.

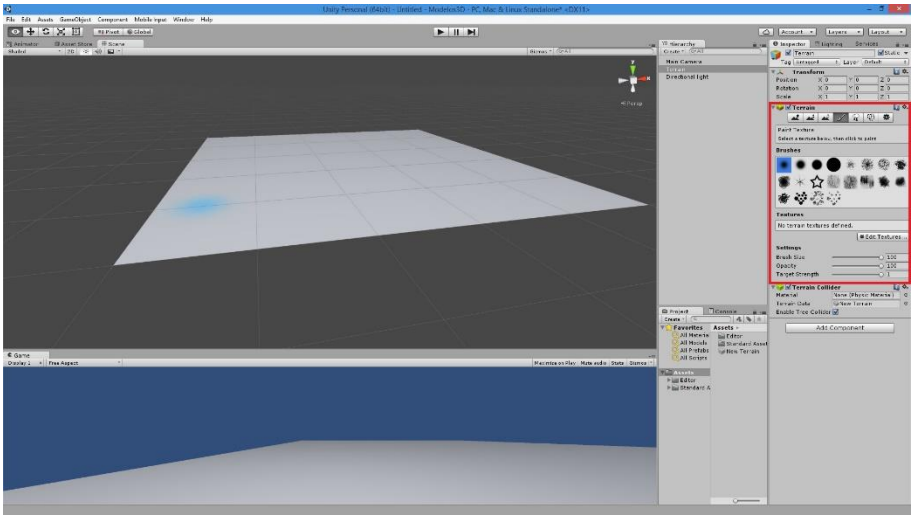


Figura 25. *Herramientas de edición del terreno*

El resultado es un terreno que contiene montañas, arboles, rocas, color y textura acorde a una planicie, elementos con los cuales el manipulador móvil puede interactuar y poner a prueba el esquema de teleoperación tal como se puede observar en la Figura 26.

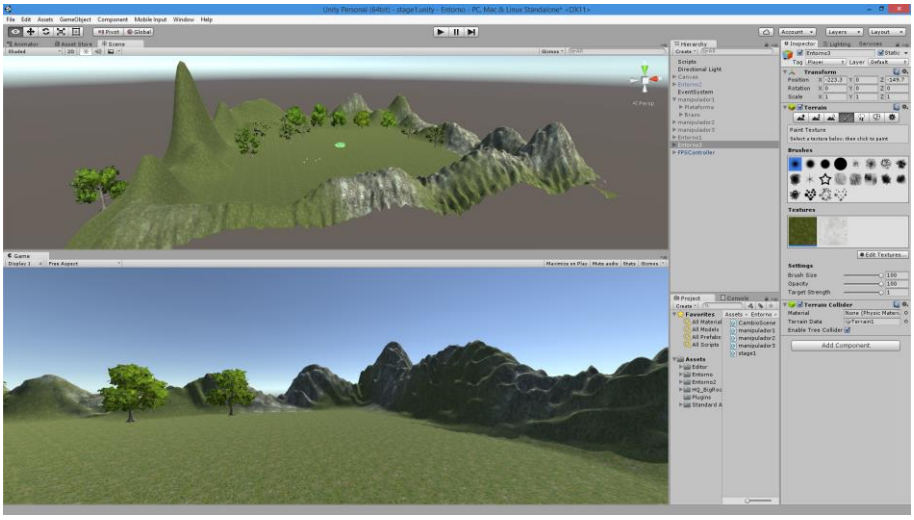


Figura 26. *Terreno modelado en Unity*

3.2.4 Heads Up Display

Se conoce como Heads Up Display (HUD) a la información que en todo momento se muestra en pantalla durante el funcionamiento del esquema de tele-operación, constituye la interfaz de usuario que le permite al operador monitorear datos del estado del manipulador móvil e información útil para realizar la tele-operación. Esta interfaz permite además al operador configurar las opciones disponibles de simulación.

La interfaz de usuario se construye en el apartado de UI (User Interface) de la escena principal de Unity. Para que las opciones disponibles en la pantalla puedan interactuar con las órdenes del operador y considerando que, cuando se está usando el casco Oculus Rift no es posible acceder a métodos de entrada tradicionales como teclado y/o mouse, (debido a la limitación visual del operador) se usa el dispositivo LeapMotion que, de forma que se explicó en el apartado 3.1.2, detecta las manos del usuario y las plasma en el entorno virtual con el objetivo de manipular e interactuar los objetos virtuales directamente.

El mapa de la interfaz de usuario ofrece la posibilidad de elegir los entornos en los que se desenvolverá el manipulador, además del número de manipuladores móviles que trabajaran simultáneamente en el control cooperativo. También ofrece la posibilidad de ubicar los manipuladores en la posición inicial deseada por el operador. El mapa de la interfaz de usuario se muestra en la Figura 27.

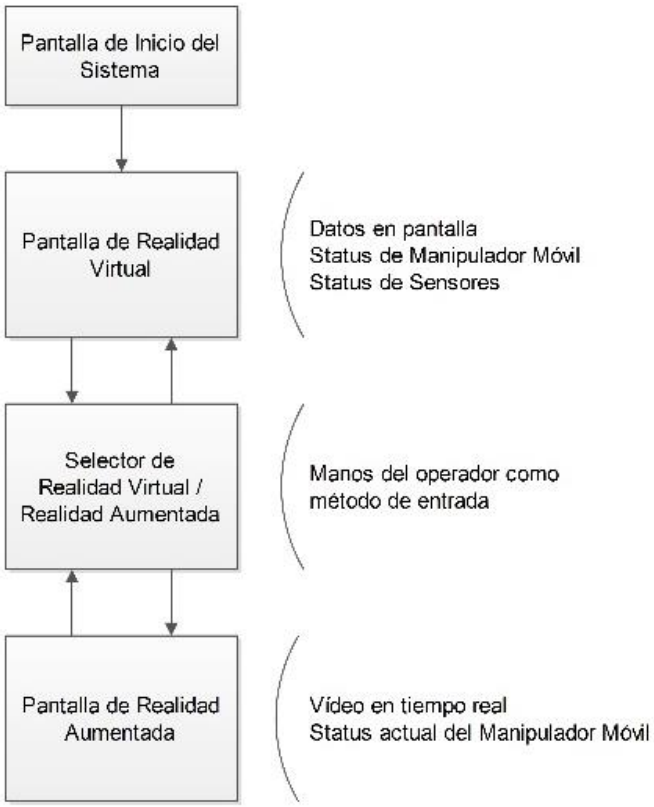


Figura 27. *Mapa de la interfaz de usuario*

La construcción de la interfaz de usuario se la realiza tomando los elementos del package de LeapMotion para Unity el cual ofrece los widgets necesarios para desarrollar la interfaz. Los widgets que se ocupan en esta interfaz son: botones, sliders y seleccionadores de opciones, cada uno de estos tienen la posibilidad de adaptarse al diseño propuesto de interfaz en términos de color, tamaño, posición, y calibración de funcionamiento con el fin de alcanzar el mayor grado de usabilidad de la interfaz. En la Figura 28 se puede observar la configuración inicial de cada widget.

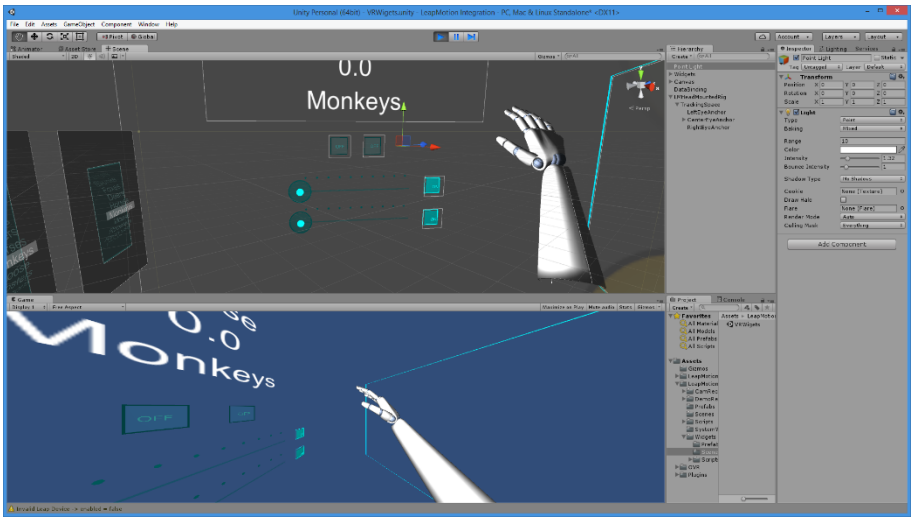


Figura 28. *Widgets de LeapMotion*

3.3 Compatibilidad de tecnologías

Hasta este punto se ha desarrollado el esquema de teleoperación, interfaz de usuario, entornos virtuales, movimientos del modelo 3D y la integración de los dispositivos Oculus Rift, LeapMotion, y Falcon de forma separada, es momento de unir todo el desarrollo para crear una aplicación de escritorio que será el programa final que utilizará el operador. Para lograrlo se necesita compatibilizar todas las tecnologías compuestas en cada dispositivo y hacer que los desarrollos convivan en una sola escena de Unity. En este apartado se explicará la conexión entre MatLab y Unity, entre Falcon y MatLab. También se añadirá la transmisión de video para las pruebas de realidad aumentada.

3.3.1 Interconexión entre Unity3D y MatLab

Dado que el esquema de control no se puede implementar directamente en Unity, el software de control a utilizarse es MatLab. Por consiguiente, se necesita que las variables de control sean transmitidas al entorno virtual de Unity y que se vean reflejadas en el modelo 3D del robot manipulador móvil.

La comunicación entre Unity y Matlab debe ser bidireccional, es decir que las dos aplicaciones puedan leer y escribir datos simultáneamente y por el mismo canal, con ello se hace imprescindible el uso de memoria compartida. Con la herramienta de la Figura 29, se puede crear n número de memorias compartidas, con la opción de determinar tamaño, y tipo de datos de cada memoria.

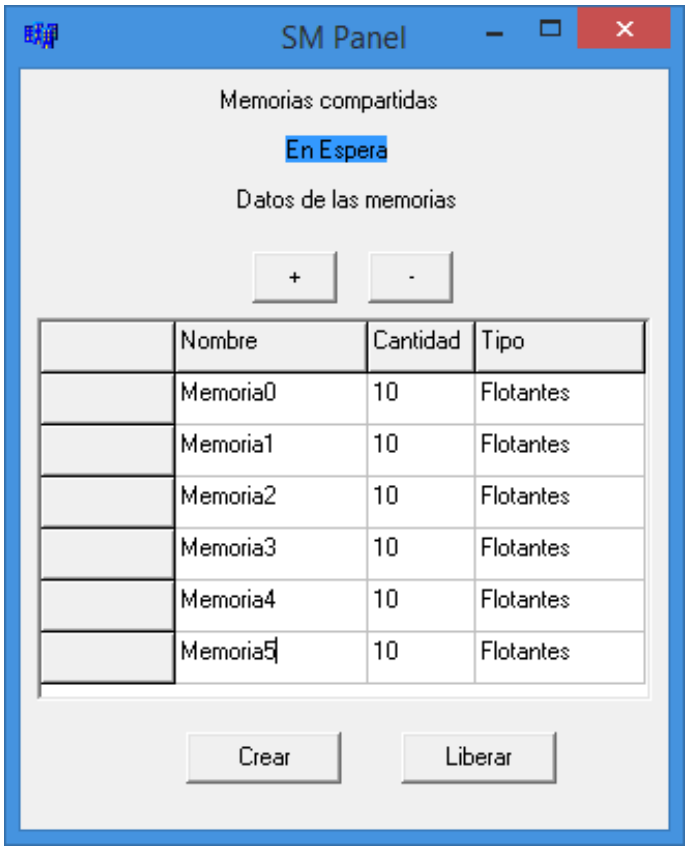


Figura 29. *Interfaz de la memoria compartida*

En (Andaluz, et al, 2016a) se detalla el procedimiento para ejecutar la interconexión entre Unity y Matlab utilizando memoria compartida. Después nada más queda añadir líneas de código en el script de MatLab y en el script de Unity, los scripts de control de ambas aplicaciones se encuentran disponibles en el Anexo B.

3.3.2 Conexión de Falcon a MatLab

Debido a que el script de control se realiza en MatLab, es necesario que el dispositivo háptico trabaje directamente con el software de control, para lograrlo, MatLab utiliza un archivo de biblioteca de vínculos dinámicos (DLL, Dynamic-Link Library) con el cual se conecta directamente al dispositivo Falcon, el mismo que entrega los datos correspondientes al movimiento que realiza el operador al script de control. Este recurso se encuentra disponible en el Anexo B como líneas de como líneas de código.

3.3.3 Transmisión de video en la web

Para la transmisión de video desde el sitio remoto hacia el sitio local donde se encuentra el entorno de realidad aumentada, se utiliza el protocolo WebRTC en Unity, el cual es capaz de consumir datos presentes en un servidor y visualizarlos a modo de textura de video sobrepuesta en un plano en Unity. El código de web RTC utilizado en el presente desarrollo se encuentra disponible en el Anexo B.

3.3.4 Conexión del master a la red

El esquema de tele-operación trabaja mediante una red local o WAN en la cual el operador y el manipulador móvil se conectan simultáneamente para lograr su comunicación. En el sitio local la conexión hacia la red se da desde MatLab. Para lograrlo se usa el estándar WebSocket hacia el servidor. El código necesario para la conexión se encuentra disponible en un el Anexo B.

3.3.5 Conexión del usuario consumidor a la red

Como complemento al esquema de teleoperación se hace imprescindible mostrar mediante la web un cliente de los datos que envía y recibe el sitio local. Este cliente está disponible como un enlace web hacia el servidor, el cual se mantiene actualizado por el servidor. Con ello cualquier persona en cualquier parte del mundo puede observar lo que ve el operador mientras realiza las tareas de teleoperación del manipulador móvil. El desarrollo de este cliente se lo realiza del mismo modo en que se concibió el programa para el operador en el sitio remoto salvo eliminando el HUD y ciertos elementos que no son compatibles con la perspectiva de visión de una pantalla plana. la versión final del cliente web se la puede observar en la Figura 30.

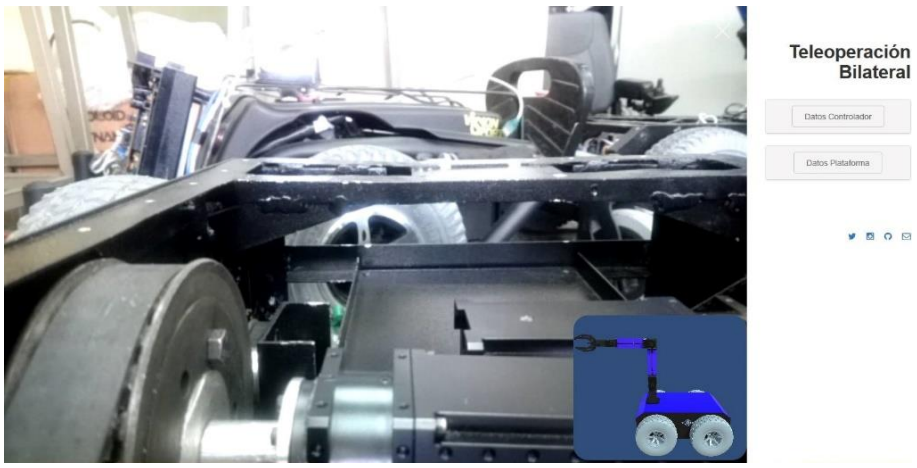


Figura 30. *Cliente web*

CAPÍTULO IV

4. TELE-OPERACIÓN BILATERAL

4.1 Esquema de tele-operación bilateral

En esta sección se muestra el esquema propuesto para la tele-operación bilateral de un manipulador móvil para tareas que requieren tanto habilidades de locomoción como de manipulación.

Un sistema de tele-operación, como se lo puede ver en la Figura 31, consiste en los siguientes elementos: *i) El operador humano* es quien realiza la operación de control de forma remota. Su acción puede variar de un control continuo a una intermitente intervención, que sólo se ocupa del monitoreo e indicar los objetivos y planes de tiempo al tiempo; *ii)* La interfaz es el conjunto de dispositivos que permiten la interacción entre el operador y el sistema de tele-operación. El manipulador móvil se considera como parte de la interfaz, así como monitores de vídeo, o cualquier otro dispositivo que permita al operador humano enviar y recibir información en el sistema; *iii) Canal de comunicación* es el conjunto de dispositivos que modulan, transmiten y adaptan el paquete de señales de transmisión entre el sitio local y el sitio remoto; *iv) Dispositivo tele-operado* puede ser un robot, un vehículo o un dispositivo similar, es decir, la máquina que trabaja en el sitio remoto y está siendo controlado por el operador; y por último, *v) Sensores* son el conjunto de dispositivos que recogen información tanto del sitio local y el sitio remoto para ser utilizado por la interfaz y control.

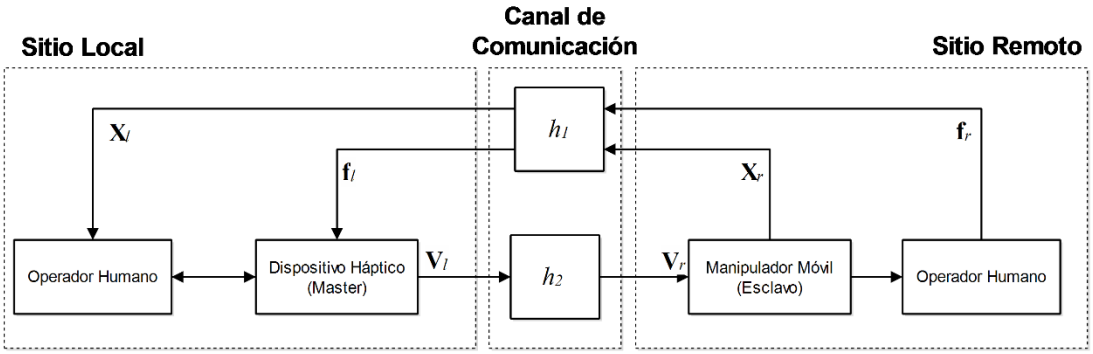


Figura 31. *Diagrama de tele-operación*

Una misión propuesta se compone generalmente de varios procesos de operación, y éstos comúnmente incluyen tres etapas: acercamiento, manipulación y retorno de procesos. Cuando se inicia la misión, el operador humano debe mover el sistema esclavo para acercarse al objeto de destino dentro del espacio de trabajo del brazo robótico; esta es la primera etapa de acercamiento. Después de que la base móvil llega a las proximidades del objeto de destino, el operador humano cambia el modo de locomoción a modo de manipulación. Cuando se haya completado la tarea de manipulación, el operador humano cambia de nuevo al modo de manipulación para la locomoción y el esclavo volvería a una zona segura, como se muestra en la Figura 32.

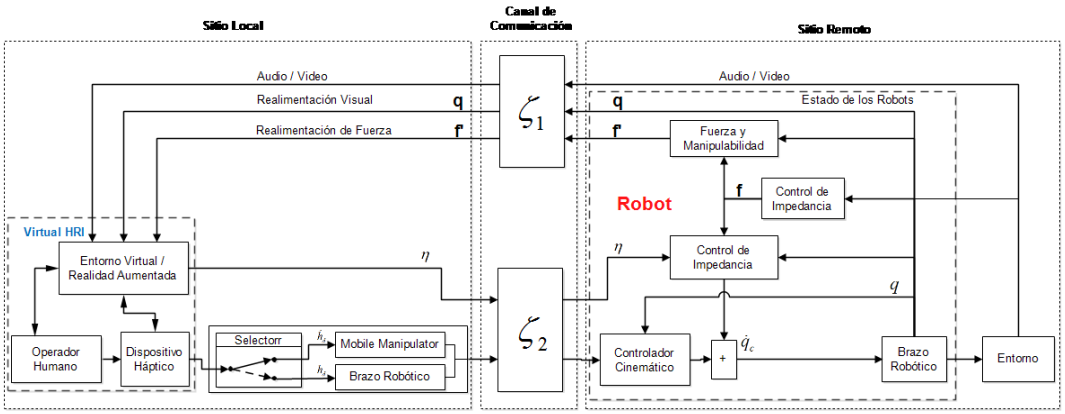


Figura 32. *Diagrama de bloques del sistema de tele-operación bilateral.*

Cuando se seleccionada el modo de locomoción, el operador humano controla el manipulador móvil enviando comandos de velocidad al extremo operativo del robot: h_l , h_m , y h_n , una por cada eje, usando un dispositivo háptico $\mathbf{h_d} = \begin{bmatrix} h_l & h_m & h_n \end{bmatrix}^T$; mientras tanto, cuando el modo manipulación es

seleccionado, el operador humano controla únicamente el brazo robótico enviando comandos de posición al extremo operativo: h_l , h_m y h_n una por cada eje, usando el mismo dispositivo háptico en modo locomoción $\mathbf{h_d} = \begin{bmatrix} h_l & h_m & h_n \end{bmatrix}^T$.

Los comandos del operador humano son generados con el uso del dispositivo háptico Falcon de la empresa Novint Technologies Inc.[8] como lo indica la Figura 33.

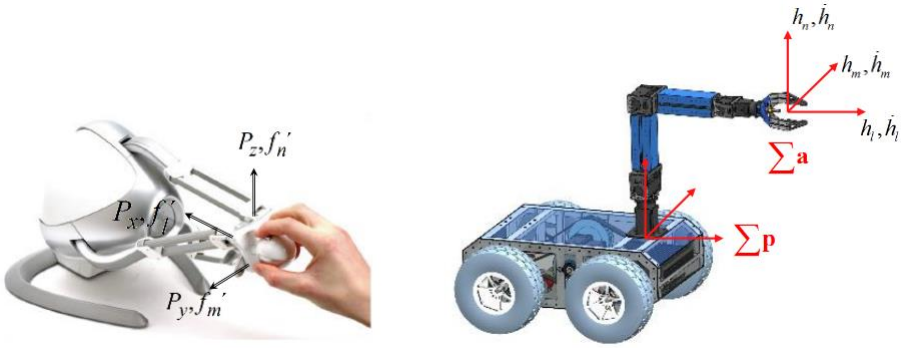


Figura 33. *Campo de trabajo del dispositivo Falcon*

Las posiciones de P_x , P_y y P_z son trasladadas a comandos de velocidad linear $\dot{h}_l$, $\dot{h}_m$ y $\dot{h}_n$ para el modo de locomoción; o trasladadas a comandos de posición h_l , h_m y h_n para el modo de manipulación a través de la siguiente matriz de rotación,

$$\begin{bmatrix} h_l, \dot{h}_l \\ h_m, \dot{h}_m \\ h_n, \dot{h}_n \end{bmatrix} = \begin{bmatrix} \cos(\psi + \theta_1) & -\sin(\psi + \theta_1) & 0 \\ \sin(\psi + \theta_1) & \cos(\psi + \theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \quad (1)$$

donde ψ define la orientación de la plataforma móvil y θ_1 representa la primera articulación del brazo robótico. Para más detalles acerca del esquema de teleoperación propuesto puede ver (Andaluz, *et al*, 2011).

[8] Más información en “Characterization of the Novint Falcon Haptic Device for Application as a Robot Manipulator”.

De acuerdo con lo descrito anteriormente, cabe mencionar que el espacio de trabajo del dispositivo háptico Novint Falcon permite el control del robot en dos maneras: a) Modo locomoción, el área de trabajo del Novint Falcon está estrechamente relacionada con las velocidades mínima y máxima en los ejes l,m,n del sistema de coordenadas $\sum a$; mientras que b) en el modo manipulación, el espacio de trabajo del Novint Falcon es linealmente proporcional al espacio de trabajo del brazo robótico, por lo tanto el espacio de trabajo del brazo depende de la longitud de las articulaciones

4.2 Modelación cinemática del manipulador móvil

El modelo cinemático de un manipulador móvil ofrece la ubicación del extremo operativo $\mathbf{h}$ en función del brazo robótico y de la ubicación de la plataforma (o sus coordenadas operacionales como funciones de coordenadas generalizadas del brazo robótico y las coordenadas operativas de la plataforma móvil). (Bayle, *et al*, 2003).

$$f : N_a \times M_p \rightarrow M \tag{2}$$

$$\left(\mathbf{q}_p, \mathbf{q}_a\right) \mapsto \mathbf{h} = f\left(\mathbf{q}_p, \mathbf{q}_a\right) \tag{3}$$

donde N_a es la configuración del espacio del brazo robótico, M_p es el espacio operacional de la plataforma.

El modelo cinemático instantáneo de un manipulador móvil brinda la posición su extremo operativo como una función de la configuración del brazo robótico y la localización de la plataforma móvil.

$$\dot{\mathbf{h}}(t) = \mathbf{J}(\mathbf{q})\mathbf{v}(t) \tag{4}$$

donde $\dot{\mathbf{h}} = \begin{bmatrix} \dot{h}_1 & \dot{h}_2 & \ldots & \dot{h}_m \end{bmatrix}^T$ es el vector velocidad del extremo operativo, $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & \ldots & v_{\delta_n} \end{bmatrix}^T = \begin{bmatrix} v_p^T & v_a^T \end{bmatrix}^T$ es el vector velocidad del manipulador móvil

en el cual v_p contiene las velocidades linear y angular de la plataforma móvil y v_a contiene las velocidades de las articulaciones del brazo robótico. La dimensión del vector $\mathbf{v}$ es $\delta_n = \delta_{np} + \delta_{na}$, donde δ_{np} y δ_{na} son las dimensiones del vector velocidad asociado a la plataforma móvil y al brazo robótico respectivamente. La matriz que se muestra a continuación

$$\mathbf{J(q)} = \frac{\partial f\,\mathbf{(q)}}{\partial \mathbf{q}}\mathbf{T(q)} \tag{5}$$

representa la matriz Jacobiana que define una correspondencia lineal entre el vector velocidad del manipulador móvil $\mathbf{v}(t)$ y el vector velocidad del extremo operativo $\mathbf{h}(t)$; y $\mathbf{T(q)}$ es la matriz transformación que relaciona el vector velocidad de las articulaciones $\mathbf{q}$ con el vector velocidad del manipulador móvil $\mathbf{v}(t)$ tal que $\mathbf{q} = \mathbf{T(q)v}(t)$. Tenga en cuenta que $\mathbf{T}$ incluye las limitaciones no holonómicas de la plataforma móvil. Esas configuraciones en donde $\mathbf{J(q)}$ es de rango incompleto se denominan configuraciones cinemáticas singulares. Encontrar las singularidades del manipulador es de gran interés debido a las siguientes razones:

1. Las singularidades representan configuraciones en la que la movilidad de la estructura se reduce, es decir, no es posible imponer un movimiento arbitrario al extremo operativo.
2. En el entorno de una singularidad, pequeñas velocidades en el espacio operativo pueden causar grandes velocidades en el espacio $\mathbf{q}$

Es de notar que, en general, la dimensión del espacio operativo m es menor que el grado de movilidad δ_n del manipulador móvil. Por lo tanto, el sistema es redundante y esta característica se debe tener en cuenta al diseñar el controlador para lograr algo del rendimiento deseado

4.3 Diseño de algoritmos de control en lazo cerrado

El diseño de los controladores cinemáticos, tanto para el sistema de manipuladores móviles y para el brazo robótico se basan en una solución de mínima norma, lo que significa que, en cualquier momento el manipulador o el brazo robótico alcanzaran su destino de navegación con el menor número posible de movimientos.

4.3.1 Control cinemático de un manipulador móvil

El diseño del controlador cinemático del manipulador móvil está basado en el modelo cinemático del manipulador móvil (4). Se propone la siguiente ley de control de velocidad para el manipulador móvil.

$$\mathbf{v}_c = \mathbf{J}^{\#}\dot{\mathbf{h}}_{\mathbf{d}} + (\mathbf{I} - \mathbf{J}^{\#}\mathbf{J})\mathbf{L}_{\mathbf{B}} \tanh(\mathbf{L}_{\mathbf{B}}^{-1}\mathbf{B}\Lambda) \tag{6}$$

donde $\mathbf{J}^{\#} = \mathbf{W}^{-1}\mathbf{J}^{T}(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^{T})^{-1}$, siendo $\mathbf{W}$ define una matriz positiva que controla las acciones del sistema, $\mathbf{h}_{\mathbf{d}}$ es el vector velocidad deseado del extremo operativo $\mathbf{h}$, $\mathbf{B}$ y $\mathbf{L}_{\mathbf{B}}$ son matrices diagonales definidas positivas que pondera el vector Λ

4.3.2 Control cinemático de un brazo robótico

Usando el mismo razonamiento que en la sección anterior, el diseño del control cinemático de posición para el brazo robótico está basado en el modelo cinemático del brazo robótico.

$$\mathbf{h}(t) = \mathbf{J}(\mathbf{q}_{\mathbf{a}})\mathbf{q}_{\mathbf{a}}(t) \tag{7}$$

Se propone la siguiente ley de control de posición.

$$\mathbf{v_c} = \mathbf{J_a^{\#}}\left(\mathbf{L_K}\tanh\left(\mathbf{L_k^{-1}K\tilde{h}}\right)\right) + \left(\mathbf{I} - \mathbf{J_a^{\#}J_a}\right)\mathbf{L_{Ba}}\tanh\left(\mathbf{L_{Ba}^{-1}B_a\Lambda_a}\right) \quad (8)$$

donde $\tilde{\mathbf{h}}$ es el vector de control de errores definido como, $\tilde{\mathbf{h}} = \mathbf{h_d} - \mathbf{h}$, $\mathbf{K}, \mathbf{B_a}, \mathbf{L_K}$ y $\mathbf{L_{Ba}}$ son matrices diagonales positivas que pesan el vector error $\tilde{\mathbf{h}}$ y del vector $\boldsymbol{\Lambda_a}$.

Con el fin de incluir una saturación de análisis de las velocidades de la plataforma móvil y en el brazo robótico, tanto para (6) y (8) se propone el uso de la función **tanh(.)** lo que limita el error en $\tilde{\mathbf{h}}$ y la magnitud de los vectores $\boldsymbol{\Lambda}$ y $\boldsymbol{\Lambda_a}$. El segundo termino de (6) y (8) representan la proyección del espacio nulo de $\mathbf{J}$ y $\mathbf{J_a}$, respectivamente; donde $\boldsymbol{\Lambda}$ y $\boldsymbol{\Lambda_a}$ son vectores arbitrarios que contienen las velocidades asociadas al manipulador móvil y al brazo robótico respectivamente. Por lo tanto, cualquier valor dado a $\boldsymbol{\Lambda}$ y $\boldsymbol{\Lambda_a}$ tendrá efectos solo en la estructura interna del manipulador móvil y del brazo, y no afectará el control final del extremo operativo en absoluto.

CAPÍTULO V

5. ANÁLISIS DE RESULTADOS

5.1 Consideraciones Generales

Las pruebas de tele-operación se realizan desde un sitio local, hacia el sitio remoto. En el sitio local se encuentra el operador y los instrumentos descritos en este proyecto que brindan transparencia al sistema tal como se observa en la Figura 34, mientras que en el sitio remoto se encuentra el robot manipulador móvil. La tarea específica de prueba consiste en tomar una botella que se encuentra alejada del robot manipulador móvil como lo muestra la Figura 35, ésta tarea la realizará el operador humano desde el sitio local utilizando el dispositivo háptico Novint Falcon en modalidad de Realidad Virtual y Realidad Aumentada.

Figura 34. *Sitio local*

Figura 35. *Sitio remoto*

Para las consideraciones técnicas de las pruebas a realizarse o para comprobar los resultados de las pruebas, se muestran las características de los equipos utilizados en las tablas 1, 2, 3 y 4.

Tabla 1.
Características técnicas del ordenador

Características	
Procesador	Intel® Core ™ i7-3610QM CPU @ 2,30GHz
Caché L2	32 KB
Memoria	8,092 MB
Tarjeta Gráfica Integrada	Intel® HD Graphics 4000
Tarjeta Gráfica Dedicada	NVIDIA GeForce GT 630M 1GB
Sistema Operativo	Windows 8.1 Professional 64 bits

Tabla 2.
Características técnicas del HMD de Realidad Virtual

Características	
Rastreo de Cabeza	6 grados de libertad de baja latencia
Campo de Visión	100° en diagonal
Tecnología de la Pantalla	OLED
Resolución	1920X1080 (960X1080 por ojo)
Entradas	DVI/HDMI y USB
Plataformas	PC
Peso	440g

Fuente: (Oculus.Inc, 2015)

Tabla 3.
Características técnicas del Dispositivo de Control Gestual

Características	
Dimensiones	75X25X11 (mm)
Cámaras	CMOS Monocromático L:850nm
Velocidad de Captura	200 fps
LEDs	Infrarrojos L:850nm
Zona de cobertura	Semiesfera de 61cm de radio
Campo de Visión	150°
Campo de Trabajo	110,55X110,55X69,43 (mm)
Distancia de Trabajo	7-25 cm

Fuente: (LeapMotion.Inc, 2015)

Tabla 4.
Características técnicas del Dispositivo Háptico

Características	
Espacio de Trabajo	10,16X10,16X10,16 (cm)
Realimentación de Fuerza	8,9 Newtons
Resolución de posición	400 dpi
Comunicación	USB
Velocidad de captura	1KHz

Fuente: (Novint.Inc, 2015)

5.2 Pruebas en Realidad Virtual

El entorno de realidad virtual empleado usa un HUD (Head Up Display), diseñado para mostrar información útil en la pantalla del usuario. En la Figura 36 se puede observar los elementos que conforman la interfaz de usuario implementada en las gafas Oculus Rift, los elementos son: un visor de estado que indica en que realidad se encuentra el operador (Realidad Virtual o realidad aumentada), cuenta también con dos paneles de información de la plataforma y del brazo robótico que conforma el robot manipulador móvil, en el segundo panel se encuentra la información del dispositivo háptico de posición y retroalimentación de fuerzas en cada eje del extremo operativo. Adicionalmente se cuenta con un indicador de hora y un botón para alternar el cambio entre realidad virtual y realidad aumentada.

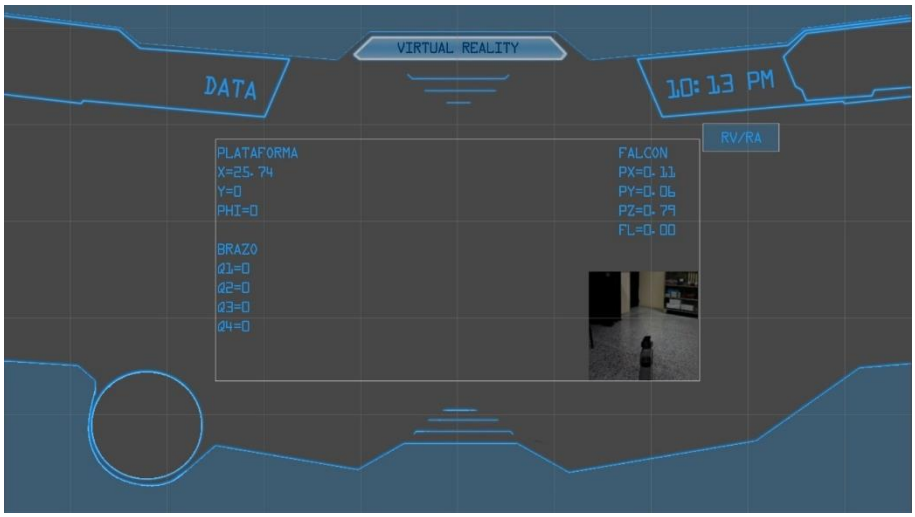


Figura 36. *HUD de interfaz de usuario*

En cuanto al escenario, como lo muestra la Figura 37, se cuenta con un espacio de trabajo relativamente grande el cual tiene cuatro columnas, en el cual está presente el robot manipulador móvil y una botella para realizar la tarea anteriormente descrita. El entorno se encuentra libre de obstáculos con el fin de no tener problemas hasta que el operador se acostumbre al nuevo sistema y su destreza se incremente.

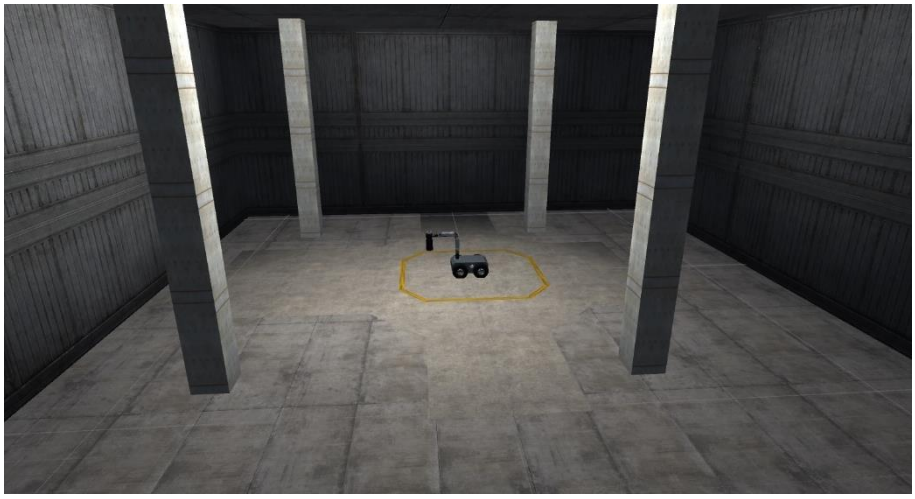


Figura 37. *Entorno de simulación*

Un vistazo del modo realidad virtual en funcionamiento se puede apreciar en la Figura 38 en la cual se nota la inclusión del entorno virtual, HUD y la interfaz gestual con las manos del operador para intercambiar de escena entre realidad virtual y realidad aumentada.

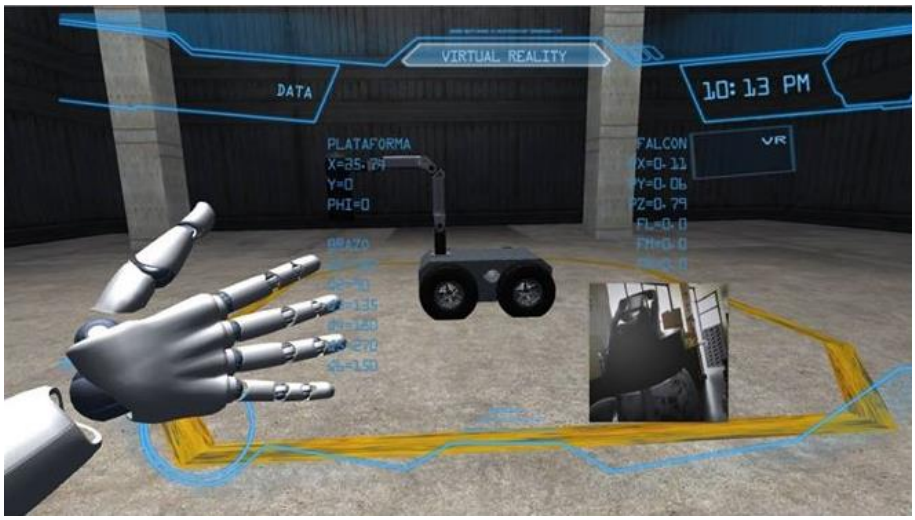


Figura 38. *Interfaz de usuario en el sitio local*

Al inicio de la tarea de alcanzar la botella, el robot manipulador móvil parte de una posición en la cual no tiene alcance a la botella, con esto se asegura usar el algoritmo de control que mueve el brazo y la plataforma en conjunto para llegar al objetivo. La Figura 39 muestra la posición inicial del robot manipulador móvil y el objetivo. Como realimentación de lo que está sucediendo en el mundo real, se tiene la imagen en tiempo real de la cámara instalada en el extremo operativo.

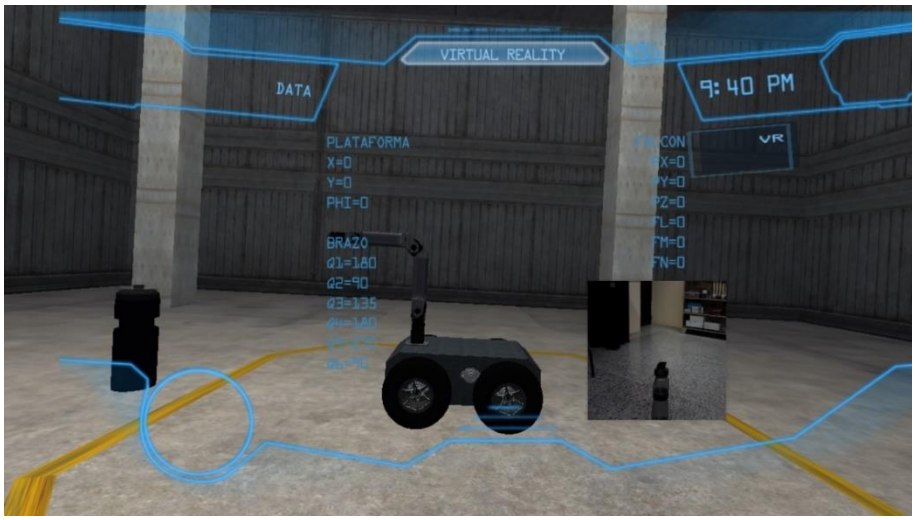
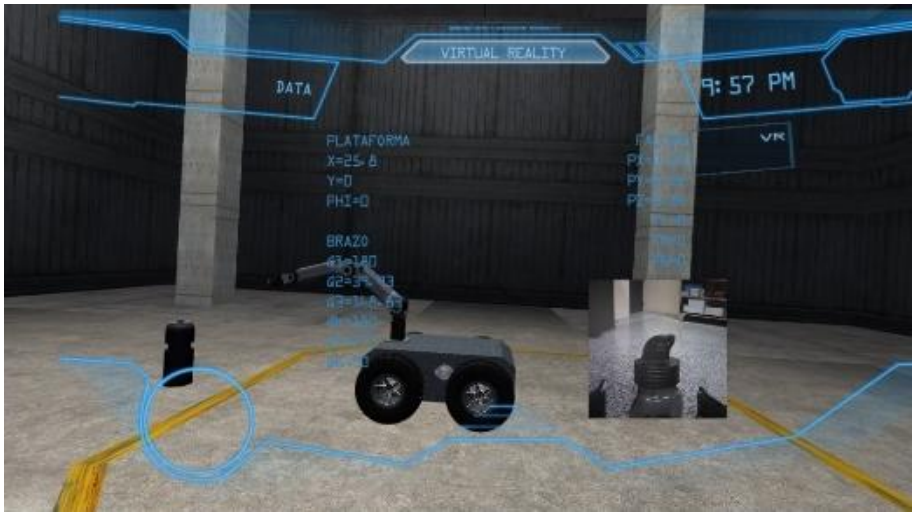


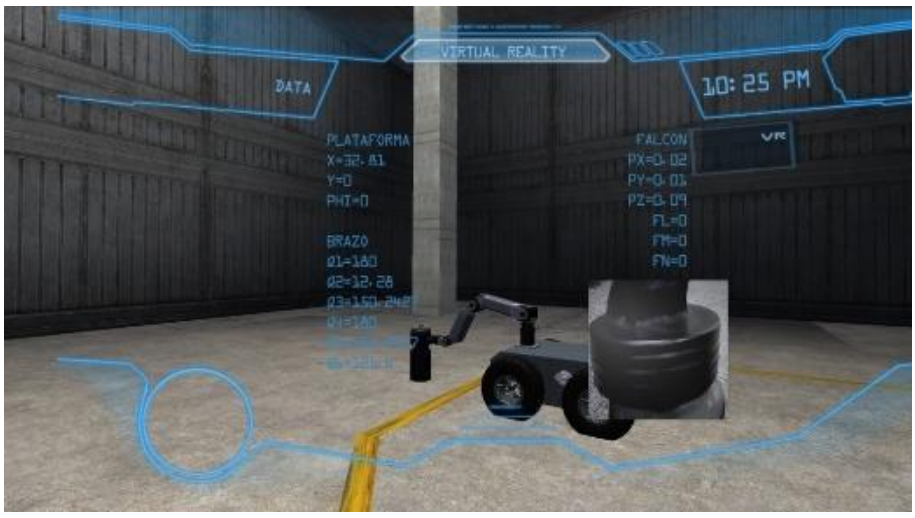
Figura 39. *Entorno de Realidad Virtual*

A continuación, en la sucesión de imágenes de la Figura 40 se puede apreciar el movimiento del manipulador hasta llegar al objetivo, tomarlo y volver al punto de partida. Al momento de agarrar el objetivo puede ser un

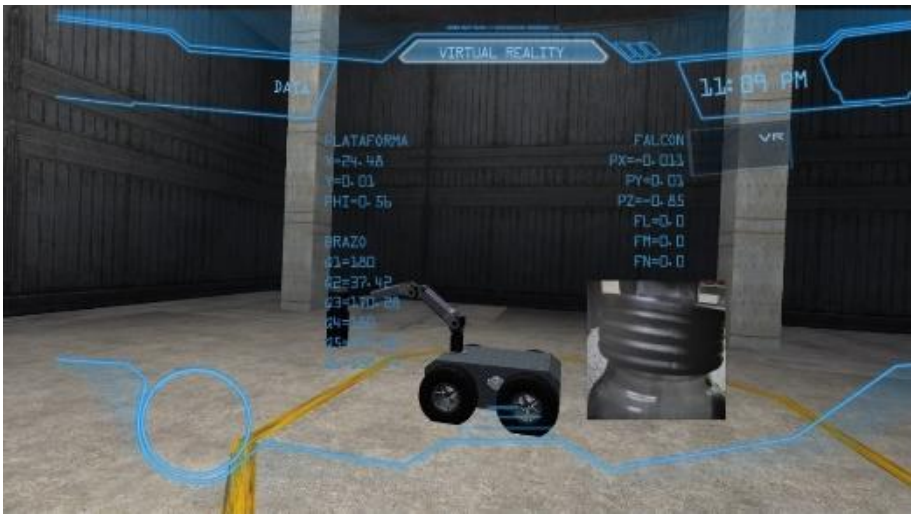
tanto confuso saber si la garra tomo ya el objetivo, es aquí donde radica la importancia de usar un recuadro en miniatura del video de la cámara ubicada en el extremo operativo del robot manipulador móvil.



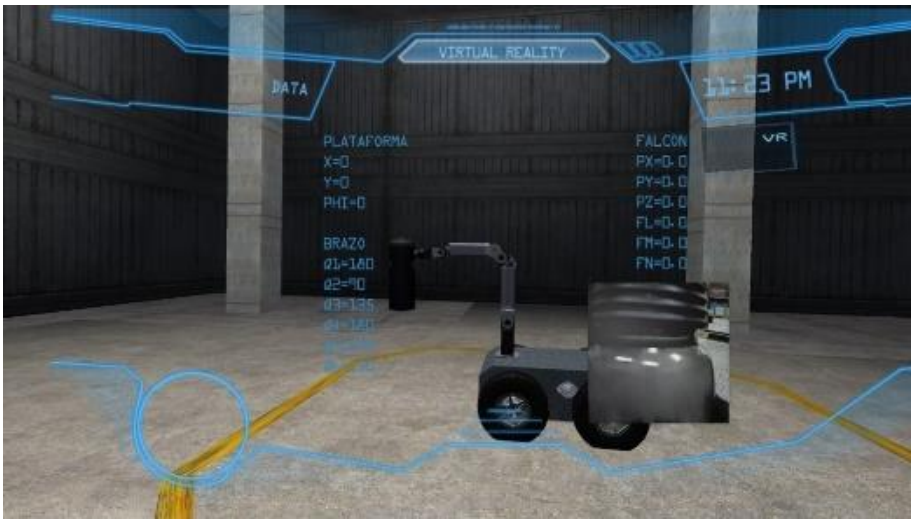
(a) Acercamiento del manipulador hacia el objetivo



(b) Movimiento del brazo robótico hacia el objetivo



(c) Agarre del extremo operativo del manipulador al objetivo



(d) Traslación del objetivo hacia la posición inicial del manipulador

Figura 40. *Secuencia de movimiento en Realidad Virtual*

Al final de la prueba se nota la experticia del operador al tener dos tipos de realimentación visual (realidad virtual y video en tiempo real), lo que ayuda a terminar la tarea de forma satisfactoria. La vista de realidad virtual hace que el operador “vea” al robot manipulador móvil, aunque realmente no lo “vea”, esto se alcanza, graficando los estados del robot en un modelo 3D, con este recurso tridimensional se logra ángulos de visión importantes para la teleoperar el vehículo de forma más inmersiva que usando únicamente la visión de la cámara. El video en tiempo real, realimenta al usuario información acerca de si agarro o no el extremo operativo al objetivo. El nivel de inmersión

se incrementa al proveer al dispositivo háptico de realimentación en fuerzas al mover el robot manipulador móvil, y al tomar el objetivo.

5.3 Pruebas en Realidad Aumentada

Para usar el modo Realidad Aumentada, el operador debe presionar el botón RA, y la interfaz de usuario se verá afectada, el entorno 3D desaparece y se ve la imagen que otorga la cámara instalada en el extremo operativo del robot manipulador móvil, y en la esquina inferior derecha se encuentra un modelo 3D que obedece al estado real del robot manipulador móvil que se encuentra en el sitio remoto, tal como lo muestra la Figura 41.

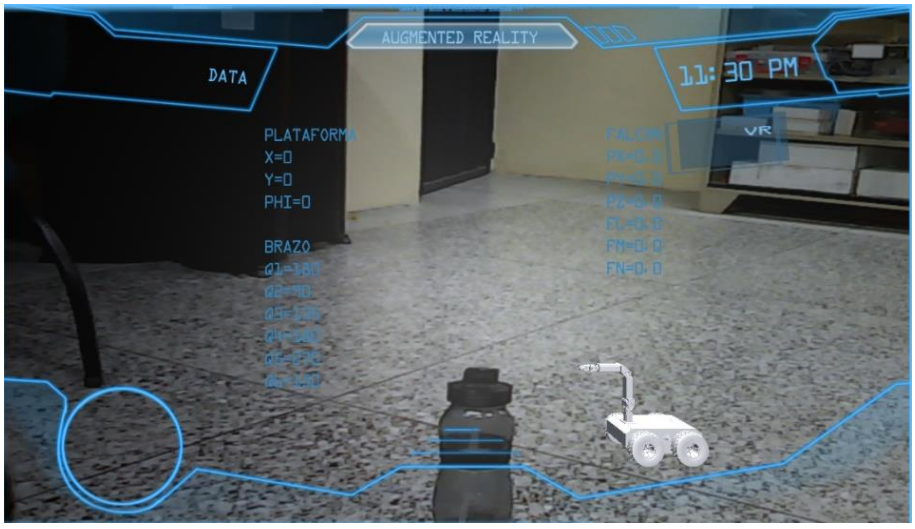
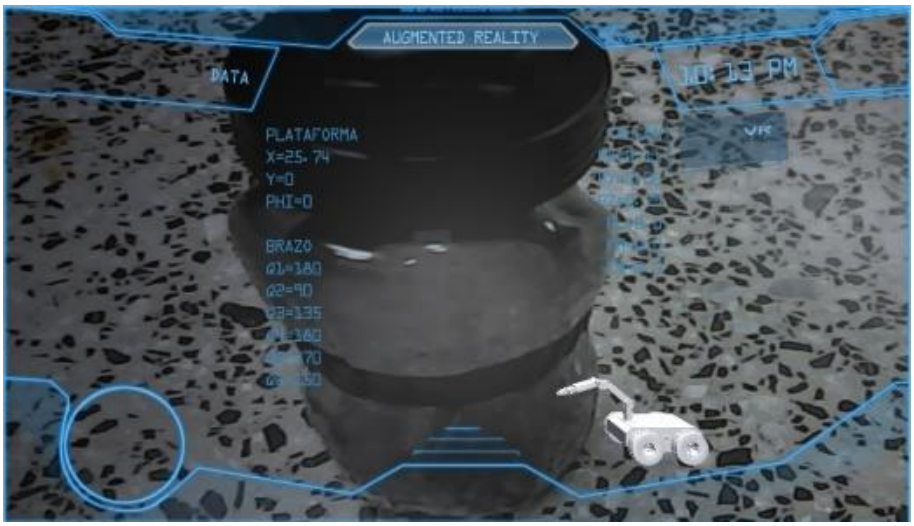
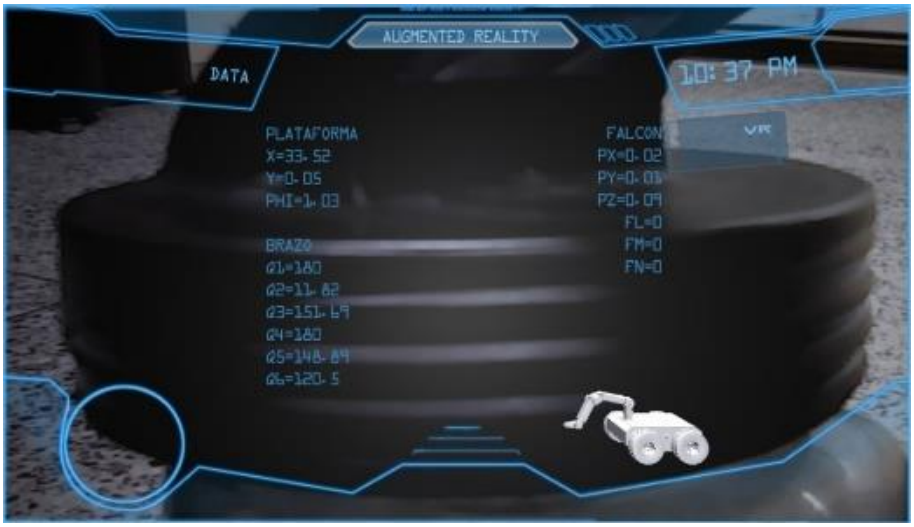


Figura 41. *Entorno de Realidad Aumentada*

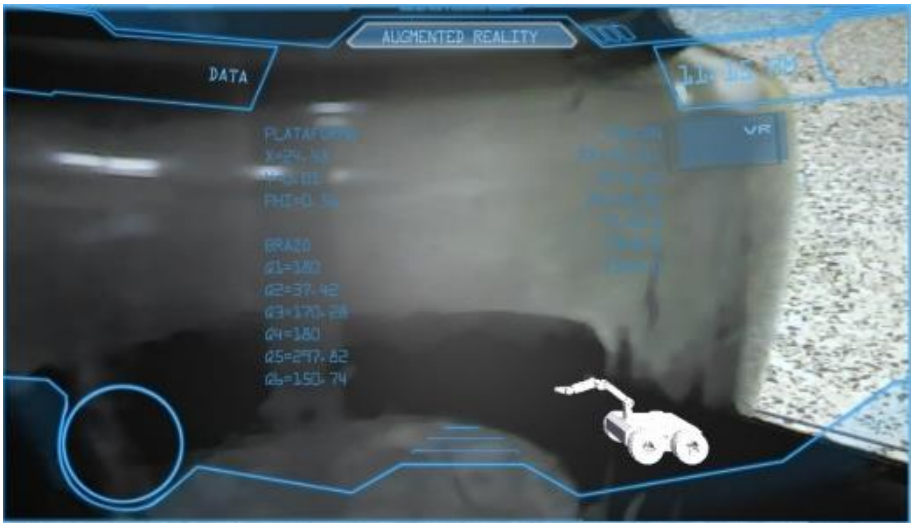
En la secuencia de imágenes de la Figura 42 se observa el proceso por el cual el extremo operativo logra llegar al objetivo y trasladarse a la posición de partida. En ciertos tramos de la tarea se observa que el objetivo se encuentra en la garra del manipulador, y por ende no se puede observar el estado del manipulador. Por ello la importancia de mostrar el estado del modelo 3D en miniatura del robot manipulador móvil.



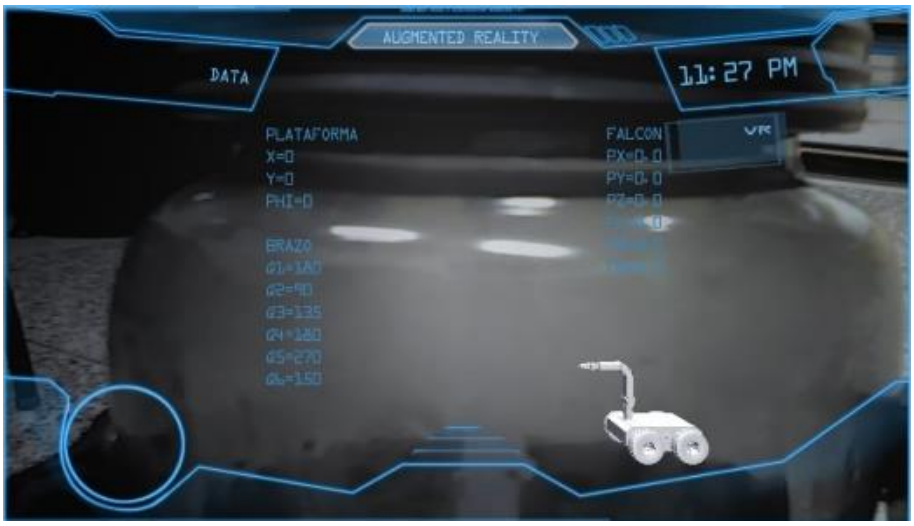
(a) Acercamiento del manipulador al objetivo



(b) Movimiento del brazo robótico para alcanzar el objetivo



(c) Agarre del extremo operativo del manipulador al objetivo



(d) Traslado del objetivo a la posición inicial del manipulador

Figura 42. *Secuencia de movimiento en Realidad Aumentada*

Al término de la prueba en Realidad Aumentada, se observa que el modelo 3D en miniatura que se ubica en la parte inferior izquierda de la interfaz de usuario, aporta a tener un mejor ángulo de visualización del robot manipulador móvil. Mientras en la pantalla principal se observa la transmisión en streaming de la cámara ubicada en el extremo operativo del vehículo en el sitio remoto, el modelo 3D informa al operador el estado del brazo robótico, esta información en pantalla guía al operador en su tarea de alcanzar el objetivo al saber si el brazo se encuentra extendido en su totalidad o si el objetivo se encuentra dentro del área de trabajo del brazo robótico. El nivel de transparencia del sistema aumenta dramáticamente al utilizar el dispositivo háptico Falcon para mover al robot manipulador móvil y al momento de tomar el objetivo y transportarlo de vuelta a la posición inicial.

CAPÍTULO VI

6. CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

- La implementación de un esquema de tele-operación para un robot manipulador móvil maniobrado a través de dispositivos hápticos, incrementa la transparencia del sitio remoto, a través de entornos de realidad virtual y realidad aumentada, en el sitio local.

- Para incrementar la transparencia del esquema de tele-operación propuesto se implementó un sistema de fuerzas háptico y una interfaz 3D humano robot; la cual muestra un entorno en realidad virtual que permite ver el modelo 3D del manipulador y un video streaming en miniatura del extremo operativo del robot, mientras que en realidad aumentada se muestra el video streaming en pantalla completa y en una esquina el modelo 3D del manipulador móvil.

- Los resultados experimentales muestran la factibilidad y el buen desempeño del esquema de tele-operación propuesto para realizar tareas complejas en un entorno remoto con un robot manipulador móvil utilizando dos modos de operación: de locomoción , en el cual el operador humano controla el manipulador móvil enviando comandos de velocidad al extremo operativo del robot; y de manipulación, en el cual el operador humano controla el brazo robótico enviando comandos de posición al extremo operativo.

- El trabajo en equipo con investigadores de varias instituciones y especialidades técnicas genera resultados más impactantes que al trabajar sin un marco colaborativo, como lo demuestran la ejecución de un macro proyecto multidisciplinario y la publicación de 3 artículos académicos.

6.2 Recomendaciones

- No se debe saturar el HUD con información irrelevante para el usuario, como botones persistentes en la interfaz, únicamente se debe mostrar indicadores que el operador pueda distinguir y leer fácilmente, tratando de evitar distracción y molestia al operador en entornos de realidad virtual y realidad aumentada.

- Al momento de ejecutar la aplicación en realidad virtual o realidad aumentada, la capacidad de procesamiento gráfico es primordial por sobre la capacidad de procesamiento lógico, por tanto, es imperativo que el ordenador tenga al menos una tarjeta de video dedicada (Nvidia GTX 970 o superior), que ofrezca 60fps en la ejecución de la aplicación.

REFERENCIAS BIBLIOGRÁFICAS

Alencastre, M., Muñoz, L., & Rudomon, I. (2003). Teleoperating Robots in Multiuser Virtual Environments. *Proceedings of the Fourth Mexican International Conference on Computer Science* (págs. pp. 8-12). ENC 2003.

Andaluz, V., Chicaiza, F., Gallardo, C., Quevedo, W., Varela, J., Sánchez, J., & Arteaga, O. (2016a). Unity3D-MatLab Simulator in Real Time for Robotics Applications. *Augmented Reality, Virtual Reality and Computer Graphics*, 246-263.

Andaluz, V., Roberti, F., Toibero, J., & Carelli, R. (2012). Adaptative Unified Motion Control of Mobile Manipulators. *In: Journal of Control Engineering Practice, Vol. 66*, pp. 1337-1352.

Andaluz, V., Salinas, L., Roberti, F., Toibero, J., & Carelli, R. (2011). Switching Control Signal for Bilateral Tele-Operation of a Mobile Manipulator. pp. 778-783.

Andaluz, V., Sánchez, J., Chamba, J., Romero, P., Chicaiza, F., Varela, J., . . . Cepeda, L. (2016b). Unity3D Virtual Animation of Robots with Coupled and Uncoupled Mechanism. *Augmented Reality, Virtual Reality, and Computer Graphics*, 89-101.

Bayle, B., Fourquet, J., & Renaud, M. (2003). Manipulability of wheeled mobile manipulators: Application to motion generation. *In: International Journal of Robotics Research*, pp. 565-581.

Brady, K., & Tarn, T. (2000). Internet-Based Teleoperation. *In: IEEE International Conference on Robotics & Automation*, (págs. 843-848).

Carlson, J., & Murphy, R. (2005). How UGVs physically fail in the field. *IEEE Trans. on Robotics, Vol. 21*, pp. 423-36.

Chong, N., Kotoku, T., Ohba, K., Komoriya, K., Matsuhira, N., & Taine, K. (2000). Remote coordinated controls in multiple telerobot cooperation. *In:*

IEEE-ICRA International Conference on Robotics and Automation, Vol. 4, pp. 3138-3143.

Farkhatdinov, I., Ryu, J., & Poduraev, J. (2008). A Feasibility Study of Time-Domain Passivity Approacho for Bilateral Teleoperation of Mobile Manipulator. *In: Int. Conf. on Control*, 353-358.

García, S., Slawinski, E., & Mut, V. (2014). Implementacion de sistema de tele-operacion, multi-operador, multi-robot. *In: Biennial Congress of Argentina (ARGENCON)*, PP. 827-832.

Hamner, B., Koterba, S., Shi, J., Simmons, R., & Singh, S. (2010). An Autonomous mobile manipulator for assembly tasks. *Autonomous Robots*, *28*, págs. 131-149.

Hara, M., Asada, C., Higuchi, T., & Yabua, T. (2004). Perceptual illusion in virtual reality using haptic interface. *Intelligent Robots and Systems. Vol. 4*, págs. 3901-3906. IEEE/RSJ Internacional Conference.

Henao, A., Montoya, C., & Moreno, R. (2014). Exoesqueleto para control de brazo robótico. *EN INGENIERIA ELECTRONICA*, 28-31.

Herbert, G., Savvas, G., & Kostas, J. (2003). Nonholonomic Navigation and Control of Cooperating Mobile manipulators. *In: IEEE Transactions on Robotics and Automation, Vol. 19*(1), 53-64.

Leap Motion SDK Documentation. (21 de Agosto de 2015). Obtenido de https://developer.leapmotion.com/documentation/index.html

Lo, W.-t., Liu, Y., Elhajj, I., Xi, N., Wang, Y., & Fukuda, T. (2014). Cooperative teleoperation of a multirobot system with force reflection via internet. *In: Mechatronics, Vol. 9*(4), pp. 661-670.

Neo, E., Yokoi, K., Kajita, S., Kanehiro, F., & Tanie, K. (2005). A Switching Command-Based Whole-Body Operation Method for Humanoid Robots. *In: IEEE/ASME Trans.*, pp. 546-559.

Oculus. (22 de Agosto de 2015). *Developer Center - Documentation and SDK*. Obtenido de https://developer.oculus.com/documentation/pcsdk/latest/

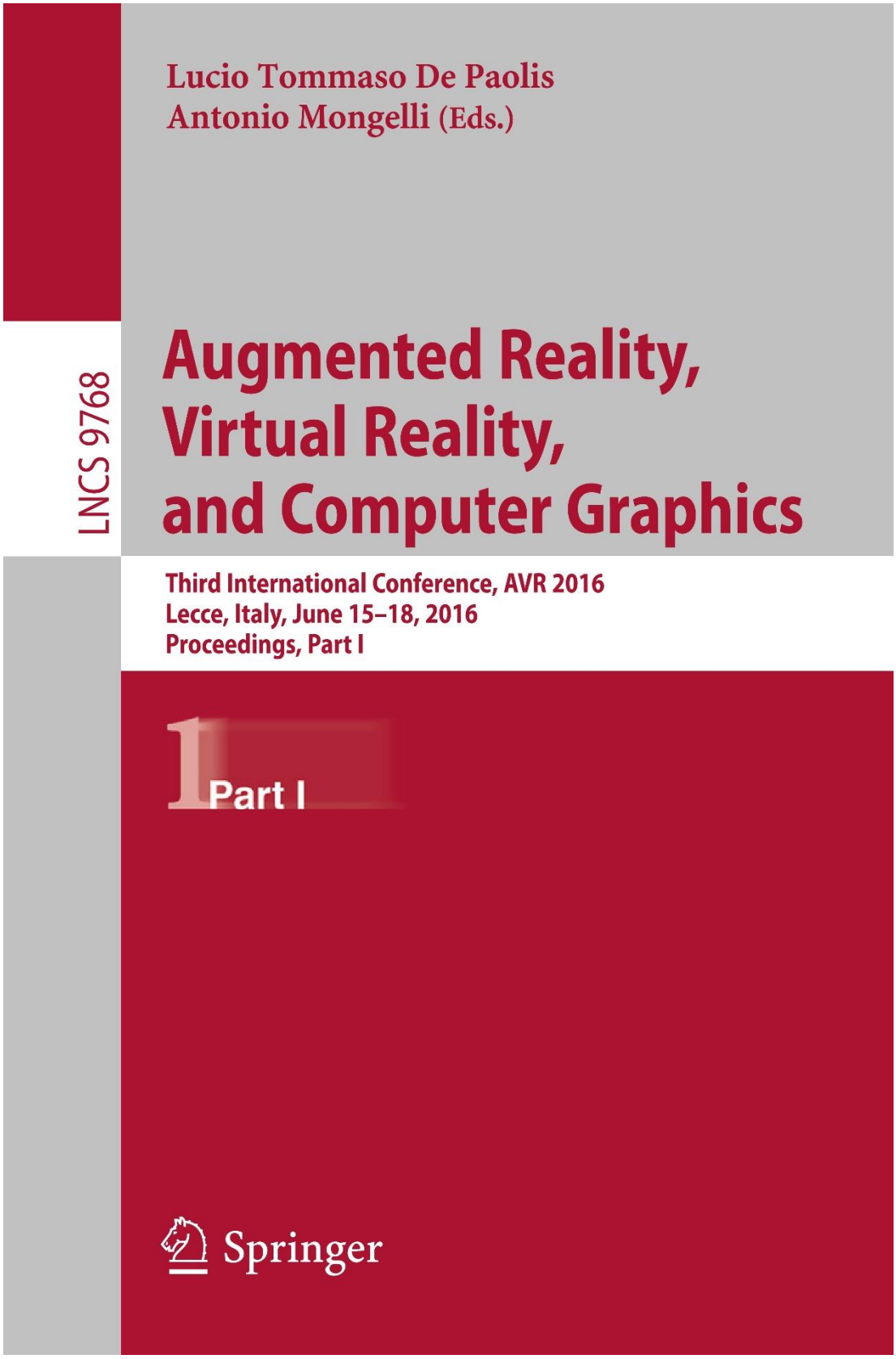
Passenberg, C., Peer, A., & Buss, M. (2010). Model-mediated teleoperation for multi-operator multi-robot systems. *In: IEEE-IROS Robot and Systems*, pp. 4263-4268.

White, G., & Bhatt, M. (2009). Experimental evaluation of dynamic redundancy resolution in a non holonomic wheeled mobile manipulator. *In: IEEE/ASME Transactions on Mechatronics*, pp. 349-357.

ANEXOS

Anexo A: Artículos Publicados

PORTADA


Lucio Tommaso De Paolis
Antonio Mongelli (Eds.)
LNCS 9768
Augmented Reality, Virtual Reality, and Computer Graphics
Third International Conference, AVR 2016
Lecce, Italy, June 15–18, 2016
Proceedings, Part I
1
Part I
Springer

CONTRAPORTADA

Editors
Lucio Tommaso De Paolis
University of Salento
Lecce
Italy

Antonio Mongelli
University of Salento
Lecce
Italy

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-40620-6 ISBN 978-3-319-40621-3 (eBook)
DOI 10.1007/978-3-319-40621-3

Library of Congress Control Number: 2016941288

LNCS Sublibrary: SL6 – Image Processing, Computer Vision, Pattern Recognition, and Graphics

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

Artículo Publicado N°1

Transparency of a Bilateral Tele-Operation Scheme of a Mobile Manipulator Robot

Víctor Hugo Andaluz[1,2(✉)], Washington X. Quevedo[1], Fernando A. Chicaiza[1], José Varela[2], Cristian Gallardo[2], Jorge S. Sánchez[1], and Oscar Arteaga[1]

[1] Universidad de las Fuerzas Armadas ESPE, Sangolquí, Ecuador
{vhandaluzl, wjquevedo, fachicaiza, jssanchez, obarteaga}@espe.edu.ec
[2] Universidad Técnica de Ambato, Ambato, Ecuador
jazjose@hotmail.es, cmgallardop@gmail.com

Abstract. This work presents the design of a bilateral tele-operation system for a mobile manipulator robot, allowing a human operator to perform complex tasks in remote environments. In the tele-operation system it is proposed that the human operator is immersed in an augmented reality environment to have greater transparency of the remote site. The transparency of a tele-operation system indicates a measure of how the human feels the remote system. In the local site an environment of augmented reality developed in Unity3D is implemented, which through input devices recreates the sensations that the human would feel if he were in the remote site, for which is considered the senses of sight, touch and hearing. These senses help the human operator to "transmit" their ability and experience to the robot to perform a task. Finally, experimental results are reported to verify the performance of the proposed system.

Keywords: Transparency · Bilateral tele-operation · Virtual reality · Augmented reality · Mobile manipulator

1 Introduction

Since remote times, man has been seeking tele-operated tools to enable it to increase the scope of its manipulations. At present there are devices that allow to reach and manipulate objects in places that are inaccessible or hazardous environments [1, 2]. The most significant application fields of tele-operation are in experimentation and planetary exploration, maintenance and operation of satellites; in the nuclear industry for the handling radioactive substances; exploration of dangerous environments with unmanned robots [3–9], among others. A tele-operation system consists of the following elements: *(i) The human operator* is the one who performs the operation control remotely. Their action can range from continuous control to an intermittent intervention, which only deals with monitoring and indicate objectives and plans from time to time; *(ii) Interface* is the set of devices that allow operator interaction with the system of tele-operation. The manipulator master is considered as part of the interface, as well

L.T. De Paolis and A. Mongelli (Eds.): AVR 2016, Part I, LNCS 9768, pp. 228–245, 2016.
DOI: 10.1007/978-3-319-40621-3_18

as video monitors, or any other device enabling the human operator to send and receive information to the system; *(iii) Communication channel* is the set of devices that modulate, transmit and adapt the set of signals transmitted between the remote and the local zone; *(iv) Tele-operated device* may be a robot, a vehicle or similar device, *i.e.,* the machine working at the remote site and is being controlled by the operator; and finally, *(v) Sensors* is the set of devices that collect information from both the local area and the remote area to be used by the interface and control.

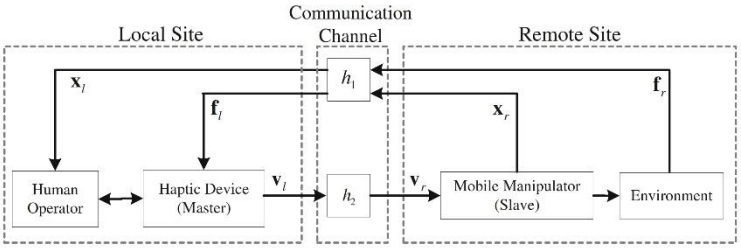


Fig. 1. General block diagram of a bilateral tele-operation system

In Fig. 1 it can be seen a general diagram of a bilateral teleoperation system. The human operator drives a robot through a haptic device generating velocity or position commands to be sent to the remote site, which will be executed by the robot [9]. The robot's state and its environment are visually back-fed to the human operator and a force is also back-fed to the human operator through the haptic device. The forces can be real or fictitious and are generated by the robot contact or non-contact interaction with the environment. The evolution of robotics and advancement of computers have helped increase the capabilities of tele-operated systems, especially as regards transparency in the local site, thus improving the performance of the human-machine interface [10–12].

Among recent works, presents a feasibility study of a time domain passivity approach for bilateral tele-operation of robots [13]. The Tele-operation in the beginning do not consider the feedback, currently applying tele-operated systems need greater sensitivity so the feedback becomes a very important element in the teleoperation especially when there are haptic devices [14, 15], for which the concept of transparency is introduced, so that the operator is able to perceive the forces occurring on the remote during teleoperation robot. In this sense, we often speak of tele-operated system with force reflection, as it is intended that the forces acting on it appear somehow reflected on the controller so that the operator is able to feel and take corrective action if appropriate. So, the infinite transparency would be a feature of a teleoperation system was able to make the operator feel exactly the same forces that feel if manipulate directly the environment [16]. There are different types of haptic interfaces, depending on the type of information that the operator is sent: visual, tactile numeric, among others [17]. These interfaces can be anything from a simple joystick to the most sophisticated virtual reality device. The use of haptic tools is increasing in industrial applications [18].

In this context, this article proposes a bilateral tele-operation scheme in which considers the human operator is immersed in an augmented reality environment to increase the transparency of the remote site. The remote site is the workplace where a mobile manipulator robot performs a task. Mobile manipulator robot is nowadays a widespread term that refers to robots built by a robotic arm mounted on a mobile platform. This kind of system, which is usually characterized by a high degree of redundancy, combines the manipulability of a fixed-base manipulator with the mobility of a wheeled platform. Such systems allow the most usual missions of robotic systems which require both *locomotion* and *manipulation* abilities. This way, they offer multiple applications in different industrial and productive areas as mining and construction or for people assistance [19, 20]. The proposed tele-operation system integrates switching of control signals generated by the master to control the whole mobile manipulator system, or to control the robotic arm only. When the mobile manipulator control is selected, the human operator sends velocity commands to the slave system; while when the arm control is selected, the human operator sends position commands. Furthermore, the local site has proposed a virtual environment and augmented reality implemented in Unity3D. The augmented reality provides feedback of different signals from the remote site in real time, so that through different haptic devices stimulate the senses of sight, touch and hearing of the human operator so that it can "transmit" their skill, experience and expertise to the robot to perform a task. To validate the proposed tele-operation system, experimental results are included and discussed.

This paper is divided into 5 Sections including the Introduction. In Sect. 2 the modeling of the mobile manipulator robot and the bilateral tele-operation scheme are formulated. The design of the local site of the human operator is presents in Sect. 3; while the experimental results are presented and discussed in Sect. 4. Finally, the conclusions are given in Sect. 5.

2 Modeling and Control Scheme

This section presents the kinematic modeling of the mobile manipulator robot. In addition, it shows the proposed scheme to solve the bilateral Tele-operation of a mobile manipulator for task that require both locomotion and manipulation abilities.

The kinematic model of a mobile manipulator gives the derivative of its end-effector location as a function of the derivatives of both the robotic arm configuration and the location of the mobile platform,

$$\dot{\mathbf{h}}(t) = \mathbf{J}(\mathbf{q})\mathbf{v}(t) \tag{1}$$

where, $\mathbf{J}(\mathbf{q})$ is the Jacobian matrix that defines a linear mapping between the vector of the mobile manipulator velocities $\mathbf{v}(t)$ and the vector of the end-effector velocity $\dot{\mathbf{h}}(t)$. The Jacobian matrix is, in general, a function of the configuration $\mathbf{q}$; those configurations at which $\mathbf{J}(\mathbf{q})$ is rank-deficient are termed *singular kinematic configurations*. It is fundamental to notice that, in general, the dimension of operational space m is less than the degree of mobility of the mobile manipulator. In this case we recall that the problem, mobile manipulator and task, is redundant [21].

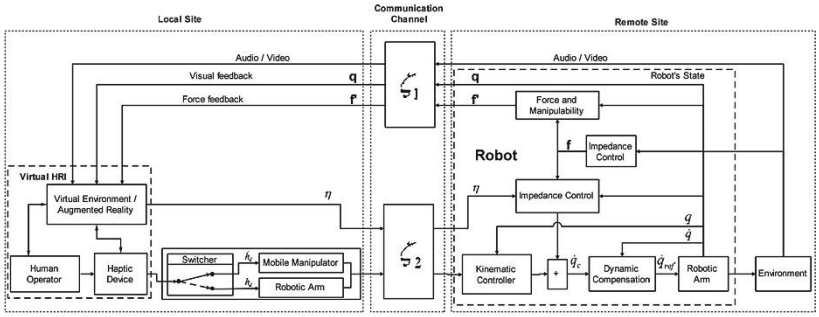


Fig. 2. Block diagram of the bilateral tele-operation system

In other hand, the proposed bilateral tele-operation system is shown in Fig. 2. In this system both the force back-fed to the human operator are considered.

A given mission is generally composed of several operation processes, and these commonly include three stages: approaching, manipulating and returning process. When the mission starts, the human operator should move the slave system closed to the target object within the r each of its robotic arm. This is the first stage: approaching. After the mobile base reaches the vicinity of the target object, the human operator changes the mode from locomotion to manipulation mode. When the manipulation mission is completed, the human operator changes again the mode from manipulation to locomotion and the slave would go back to a safe area (view Fig. 2).

When the | is selected, the human operator controls the mobile manipulator by sending velocity commands to the end-effector of the robot: h_l, h_m, and h_n, one for each axis, using a haptic device $\dot{\mathbf{h}}_\mathbf{d} = \begin{bmatrix} \dot{h}_l & \dot{h}_m & \dot{h}_n \end{bmatrix}^T$; whereas, when the *manipulation* mode is selected, the human operator controls only the robotic arm by sending position commands to the end-effector: h_l, h_m, and h_n, one for each axis, using the same haptic device that in locomotion mode $\mathbf{h}_\mathbf{d} = \begin{bmatrix} h_l & h_m & h_n \end{bmatrix}^T$.

The human operator commands are generated with the use of a Falcon™ haptic device from Novint Technologies Incorporated [22] as indicated in Fig. 3.

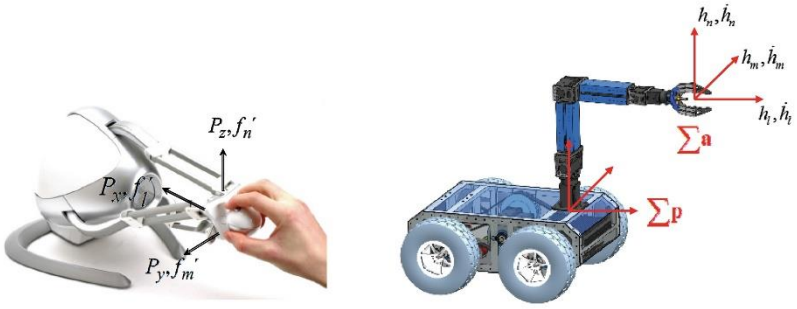


Fig. 3. Falcon ™ from Novint Technologies Incorporated

Its positions P_x, P_y, and P_z are translated into linear velocity commands $\dot{h}_l$, $\dot{h}_m$ and $\dot{h}_n$ for the locomotion mode; or into position commands h_l, h_m and h_n for the manipulation mode, through the following rotation matrix,

$$\begin{bmatrix} h_l, \dot{h}_l \\ h_m, \dot{h}_m \\ h_n, \dot{h}_n \end{bmatrix} = \begin{bmatrix} \cos(\psi + \theta_1) & -\sin(\psi + \theta_1) & 0 \\ \sin(\psi + \theta_1) & \cos(\psi + \theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

where ψ defined the mobile platform's orientation and θ_1 represents the first joint of the robotic arm. For more details about the proposed tele-operation scheme see [23].

Remark 1. According to that described above, the Novint Falcon™ 3D Touch Workspace allow the control of two ways of operation mode of robot for: *(a) locomotion mode,* the workspace of Novint Falcon™ is linearly related with the minimum and maximum velocities in the axis l, m, n of the reference system l, m, n Σa; while that *(b) manipulation mode* the workspace of Novint Falcon™ is linearly proportional to the workspace of the robotic arm, therefore the workspace of the arm depends of the length of the joints.

3 Transparency's Local Site

The transparency of a teleoperation system indicates a measure of how the human *feels* the remote system. In addition, the transparency gives an idea of how much the human controls the remote system; since the inclusion of delays, and control schemes makes that the human *takes smaller part* compared with a non-delayed direct teleoperation. In this section, we propose a definition of transparency in time and how it can be measured [24]. The transparency of a system define an equivalent system attached to the human such that it interacts with the human in the same way as the remote system. Figure 4 shows how the equivalent system is placed together with the human.

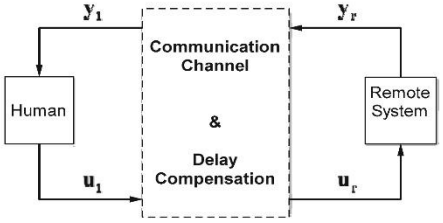


Fig. 4. Bilateral tele-operation system

The teleoperation systems search that the human operator is linked as close as it is possible to the remote task. Ideally, the tele-operation must be completely transparent in order that the human feels a direct interaction with the remote task [23, 24].

In order that the human operator have more transparency of the remote site, this work shows a Human Robot Interaction in 3D (HRI-3D) implemented on local site (view Fig. 1). For HRI-3D is considered 3D augmented reality through a graphical interface developed in Unity. The augmented reality 3D shows to the human operator the remote environment in which the robot is developing the task and the internal configuration of the robot, also it allows listening the environment's audio and feel through Falcon^{TM} haptic device the fictitious force feedback produced by the interaction of the robot with the working environment (Fig. 5).

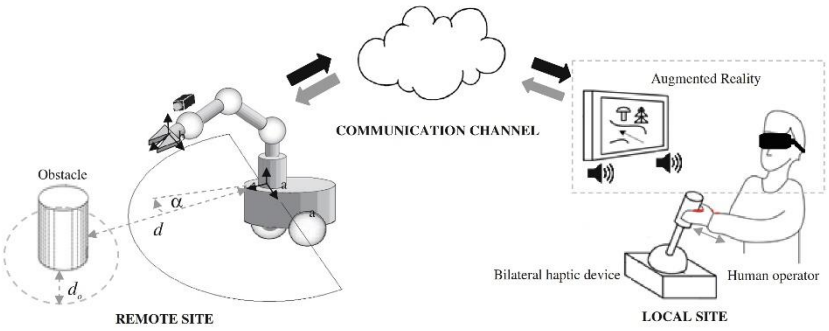


Fig. 5. Flow diagram of the system's transparency

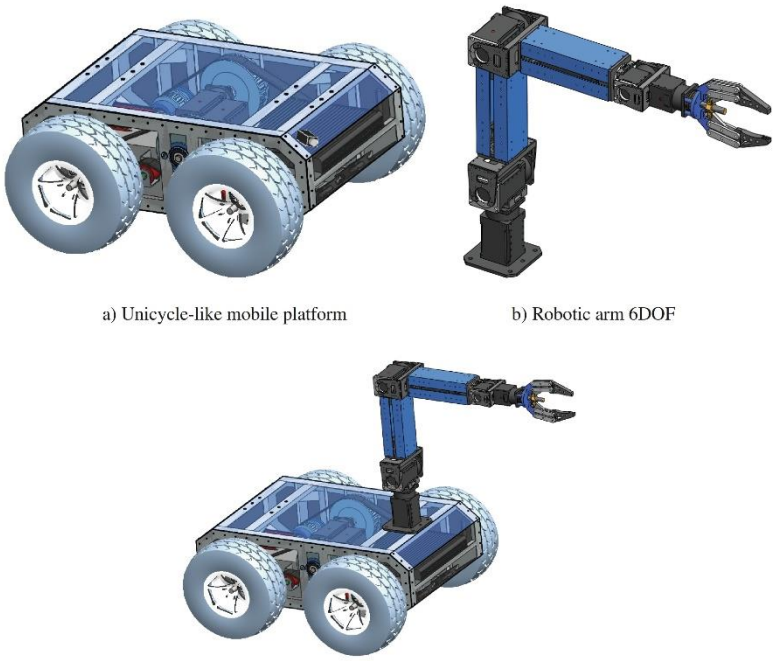
3.1 Assembly Robot

The first step is to assemble a mobile manipulator robot that has the ability of locomotion and manipulation, it should be formed of a unicycle type mobile platform and a robotic arm 6 DOF. The robot can be designed in any CAD (computer aided design) software for mechanical 3D modeling, *i.e.*, SolidWorks, Autodesk Inventor, etc. (Fig. 6)

3.2 Virtual Environment

The local site consists of a virtual reality and augmented to allow the human operator to have greater transparency in the remote site reality, as illustrated by the Fig. 7.

The Fig. 7 can be described in two stages, the first stage is *(A) Inputs,* These let read all the variables considered in the bilateral tele-operation of both the robot and the environment in which develops the task, entries that are processed in the script are: *(i) Output states of the robot,* represents the variables that describe the robot kinematics $\mathbf{q} = [q_1 \quad q_2 \quad \dots \quad q_n]^T = [\mathbf{q}_a^T \quad \mathbf{q}_p^T]^T$. In the script, the data acquired from the platform $\mathbf{q}_p$ are assigned to the *Position* and *Rotation* features of the Game Object corresponding to the mobile robot manipulator simulated in the virtual scene; while the data arm rotation $\mathbf{q}_a$ are assigned to the characteristic of *Rotation* for each joint of the simulated arm Game Object.



a) Unicycle-like mobile platform b) Robotic arm 6DOF

c) Robotic arm 9DOF

Fig. 6. Mobile manipulator, assembly on SolidWorks

Remark 2. The kinematics of the robot represents the position-orientation of the mobile platform and orientation of each joint of the robotic arm.

(ii) Velocity or desired positions through the Falcon^{TM} haptic device emit the desired location $\mathbf{h_d} = [h_l \quad h_m \quad h_n]^T$ for locomotion mode or desired velocity $\dot{\mathbf{h}}_\mathbf{d} = [\dot{h}_l \quad \dot{h}_m \quad \dot{h}_n]^T$ for manipulation mode. The operational coordinates of the mobile manipulator define the position or velocity of the end-effector in R - environment in which it develops the task. The script obtains the variation of position in world coordinates by a Game Object that simulates the movement of the end-effector of Falcon^{TM} haptic device in space, as shown in Fig. 8.

Figure 9 shows the use of the Falcon^{TM} Game Object in Script "Control 2" linked to the 3D model of mobile robot manipulator.

Then part of the script of "Control 2" is shown where the change in position or speed Falcon^{TM} on the axes l, m, n of the reference system Σa allows controlling the movement of the mobile manipulator or only the robotic arm, depending on the mode selected by the human operator.

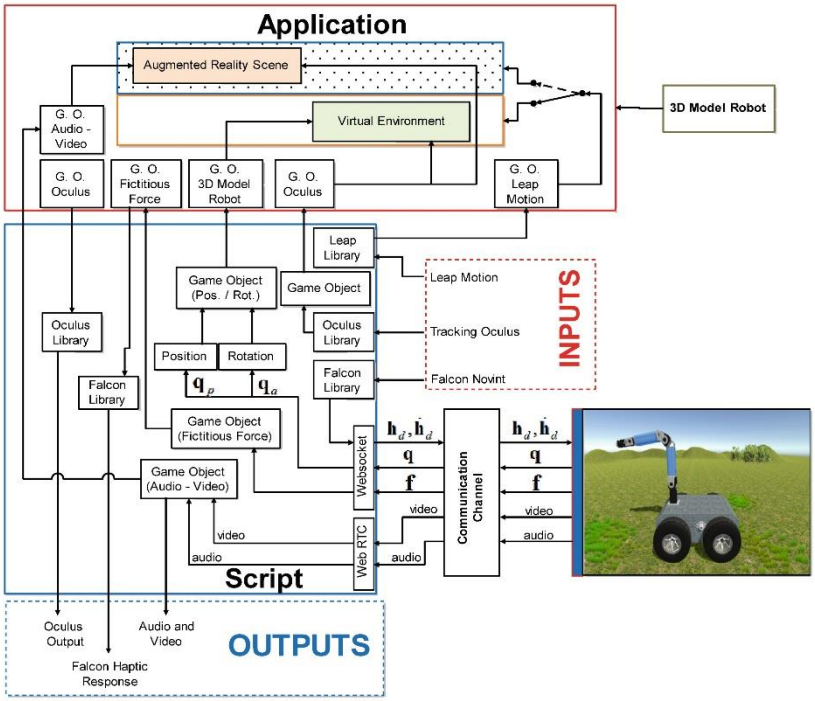


Fig. 7. Block diagram of the virtual environment

```
using UnityEngine;
...
public class Control2 : MonoBehaviour {
public GameObject GodFalcon;
...
void Update(){
...
fx = GodFalcon.transform.position.x;
fy = GodFalcon.transform.position.y;
fz = GodFalcon.transform.position.z;
... } }
```

(iii) Video and audio transmits the image and audio captured by the camera located at the end-effector of the robot. The image allows knowledge of the environment in which the task is performed, while the audio provides the sound of the medium and the sound generated by the robot motors; the audio information allows for greater transparency of the remote site because the human operator can know whether the robot is moving at high speeds based on the sound of the engines or can determine if the robot is entering a singularity configuration.; *(iv) Fictitious force* is produced by a sensor that

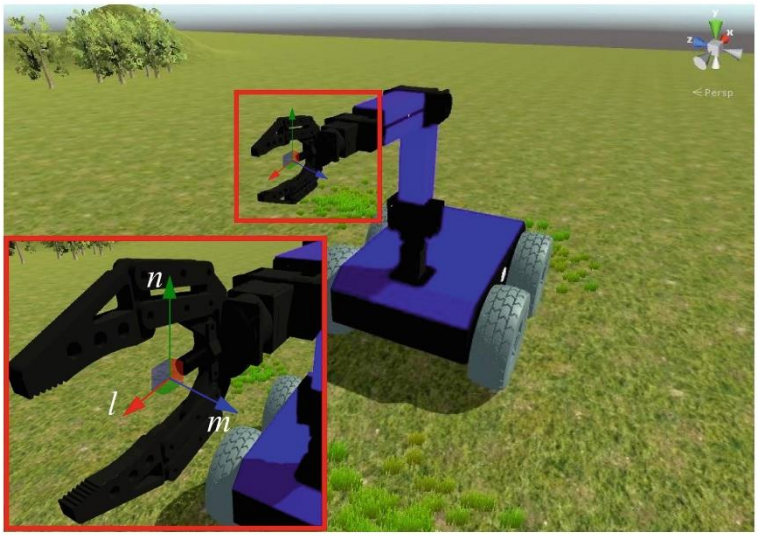


Fig. 8. Game Object: Simulates the FALCON^{TM} position or velocity.

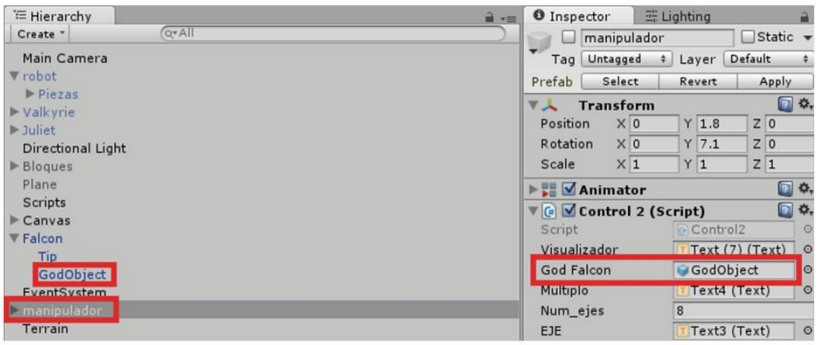


Fig. 9. Game Object: FALCON^{TM} linked to the 3D model of mobile robot manipulator.

maps the area in which the robot can be moved freely. The fictitious force is denoted by $\mathbf{f} = [f_l \quad f_m \quad f_n]^T$, review the previous section, *i.e.,* the robot will not collide with any fixed or mobile obstacle; *(v) Oculus Positional Tracking* modifies the position and rotation of the camera in the virtual and/or augmented environment, corresponding to the movement performed by the human operator when using the Head-mounted Display, HMD, in order to have a greater immersion in the experience of using the application; y finally *(vi) Leap Motion* allows a virtual model of the operator's hands in order to interact with the virtual environment and the user interface.

Remark 3. The Leap Motion device improves the usability of the user interface; while the joint use of Falcon^{TM}, Oculus Rift HMD and the Leap Motion device; allows the human operator to have greater transparency in the remote site and a more immersive experience in the virtual environment.

Furthermore, the second stage of Fig. 7 is *(B) Outputs Script* that recreates the sensations that the human operator should feel if he were at the remote site, this paper considers three ways that help the human operator to "transmit" their ability and experience to the robot to perform a task: *(i) Sense of vision,* to stimulate this sense is used the HMD Oculus Rift, to visualize an environment in virtual reality and augmented reality. The selection of these scenarios is performed through a user interface using the Leap motion; *(ii) Sense of touch,* this sense is created through the Falcon^{TM} device, it generates a force in the three axes according to the existing fictitious forces at the remote site between robot - workspace. This force allows the human operator feel obstacles that cannot be displayed with the image transmitted by the camera placed at the remote site, the obstacles are simulated in the virtual environment, while in the augmented reality, the human operator can observe the actual object that produces the fictitious force detected by the sensor on the mobile platform; and *(iii) Sense of hearing,* to excite this sense in the augmented reality environment, the actual sound of the remote site is reproduced; while in the virtual reality, the fusion sound of the robot's engines is simulated and varies according to each engine rotating speed of the mobile platform and the robotic arm.

Remark 4. The data packets corresponding to the kinematics of mobile manipulator $\mathbf{q}(t)$, the fictional force between the robot - workspace $\mathbf{f}(t)$, and the position or velocity of the end-effector of the robot $\mathbf{h_d}(t)$, $\dot{\mathbf{h}}_\mathbf{d}(t)$ generated by the Falcon^{TM} are transmitted through the WebSocket communication protocol, once the structure is standardized in JSON format.

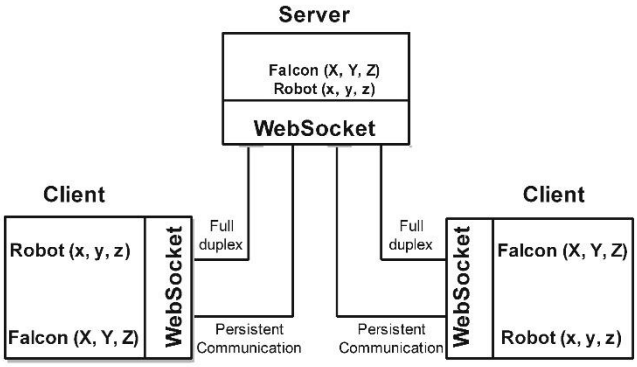


Fig. 10. Block diagram of the local site

Figure 10 illustrates the server-client communication via the WebSocket communication protocol, in this work it is considered like clients the Unity Script and Mobile

Platform. It is noted that the WebSocket protocol allows full duplex connection between the server and the client, occurs in real time and remains permanently open until closed explicitly, *i.e.*, when the client sends data to the server, the message is moved immediately, without the need to constantly initialize communication.

```
using WebSocketSharp;
...
public class Control2: MonoBehaviour {
public static int band_hilo;
public static int band_hilo2;
private Thread thread;
public static string mensaje;
static string botonstate;
public static JsonData itemData;
private static string jsonString;
public Text Visualizador;
void Start(){
   ...
   band_hilo2 = 1;
   jsonString = "{\"mot1\":\"0\",\"mot2\":\"0\",\"mot3\":\"0\"}";
   Thread.Sleep(20);
   thread = new Thread(new ThreadStart(socketInit));
   thread.Start();
}
void OnApplicationQuit(){
   band_hilo2 = 0;
   thread.Abort();
}
public static void socketInit(){
using (var ws = new WebSocket("ws://192.168.1.119:9300")){
ws.OnMessage += (sender, e) =>
jsonString = e.Data;
ws.Connect();
ws.Send("{\"cliente\":\"controlador\"}");
while (band_hilo2 == 1) {
       double PX= Convert.ToInt32(map(fx,-3,3,-0.5,0.5)*1000);
       double PY = Convert.ToInt32(map(fy,-3.2,3.2,-0.5,0.5) *1000);
      double PZ = Convert.ToInt32(map(fz,-1.8,3.1,-0.5,0.5) *1000);
      ws.Send("{\"origen\": \"controlador\", \"destino\": \"plataforma\", \"valor_x\": \""
      +PX+"\",  \"valor_y\": \"" +PY + "\", \"valor_z\": \"" + PZ + "\",\"band\": \"" + botonstate + "\"}");
      band_hilo = 1;
      Thread.Sleep(50);
        ...}}
}
```

Remark 5. The augmented reality and virtual reality for robot's motion simulation and their extension into video games, of unmanned systems are an emerging topic. There are at least three motivations for robot simulators. One is the role of simulators in adoption of new technology, another is their potential for low cost training, and finally their utility in research. The range of robot computer simulations is economically and technically diverse.

4 Simulation Experimental Results

The feasibility of the proposed structure of bilateral tele-operation has been tested through real experiments using a mobile manipulator composed by a mobile platform and a robotic arm 6 DOFs. The robot is also equipped with a HD PRO WEBCAM C920 Video Sensor. The locate station consist in a desktop computer Core I7 3610QM running at 2.3 GHz, 8 GB RAM and a NVIDIA GeForce GT 630 M 1 GB dedicated graphics card; also the local site has a haptic device Novint Falcon^{TM}, Oculus Rift HMD and a LeapMotion device. Communication between the robot and the remote station is performed through of INTRANET using WebSocket protocol. According to [25] the Virtual Reality PC-Ready can run apps at least 60 fps. The configuration described above can run the app at 10/15 fps. The refresh rate of the HMD is 60/75 Hz independly of fps and the refresh rate of Novint Falcon is 1 kHz. The actual configuration setup is depicted in Fig. 11.

(a) Remote Site

(b) Local Site

Fig. 11. Experimental setup

The developed software consists of two different interfaces. The first interface displays a virtual scenario while the second is an augmented reality; each interface is based on actual experimental data.

4.1 Virtual Reality 3D

The 3D Virtual Reality scene shows the movement of the mobile robot manipulator based on experimental data transmitted from the remote site. The Virtual environment

consists mainly in the lower right illustrated the online video captured by the camera located at the end-effector of the robotic arm, while in the upper right there is a menu that is controlled through the Leap Motion device with the hand movements of the human operator, view Fig. 12.

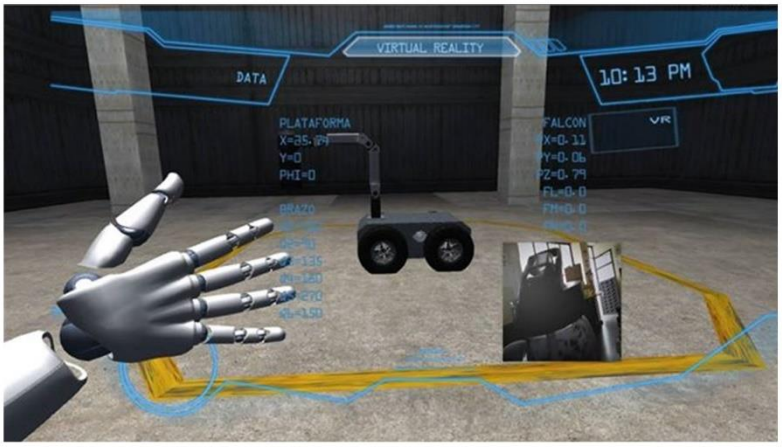


Fig. 12. Virtual Environment

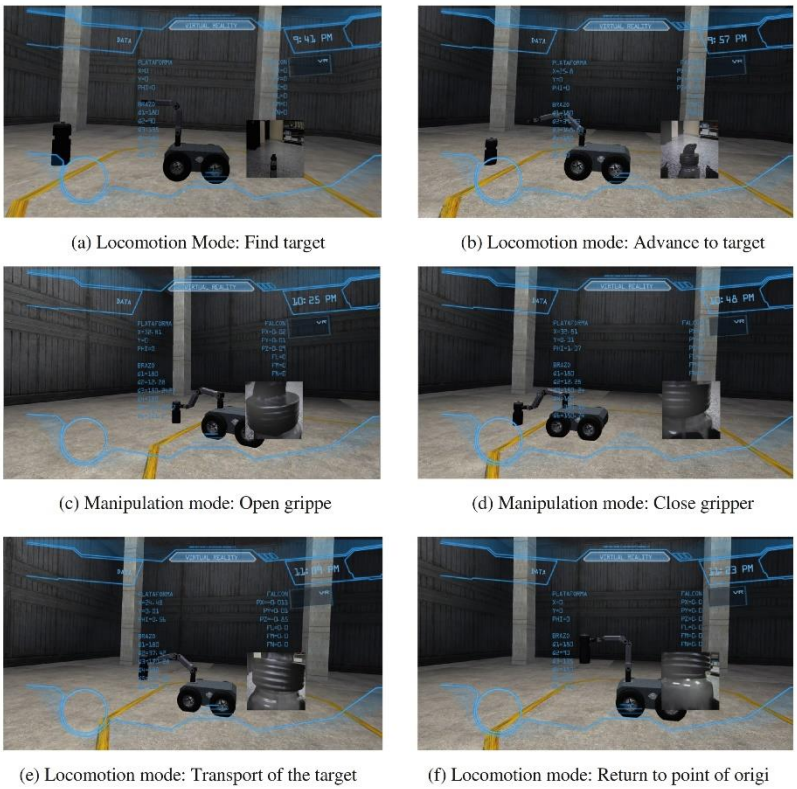
Menu virtual environment controls the various camera angles of the virtual scene, displays the communication variables and Falcon^{TM} data input; also it allowed to pass Augmented Reality interface. Figure 13 shows snapshots of the experiment detailed in previous paragraphs, where different scenes are observed based on the modification of the different menu parameters.

Remark 6. Virtual reality allows the human operator to have a vision of the whole scenario where the experimental test is performed, but is useful as long as the environment of the remote site where it is to perform the task is known.

4.2 Augmented Reality

The Augmented Reality interface consists of a visual feedback where the video is captured by the camera located at the end-effector of the robot and transmitted in real time to the local site. The video provides information of the robot interaction with the environment Fig. 14(a) but does not show the internal configuration of the robot, *i.e.*, it is not known the position of each joint of the robot arm Fig. 14(b). This is a critical problem in robots that are made up of several chain links, - redundant robots - because they can enter a singularity configuration, which would cause an unstable control algorithm.

To prevent the mobile manipulator entering in a singularity configuration, a virtual model 3D of the mobile manipulator that simulates the internal configuration of the real



(a) Locomotion Mode: Find target (b) Locomotion mode: Advance to target

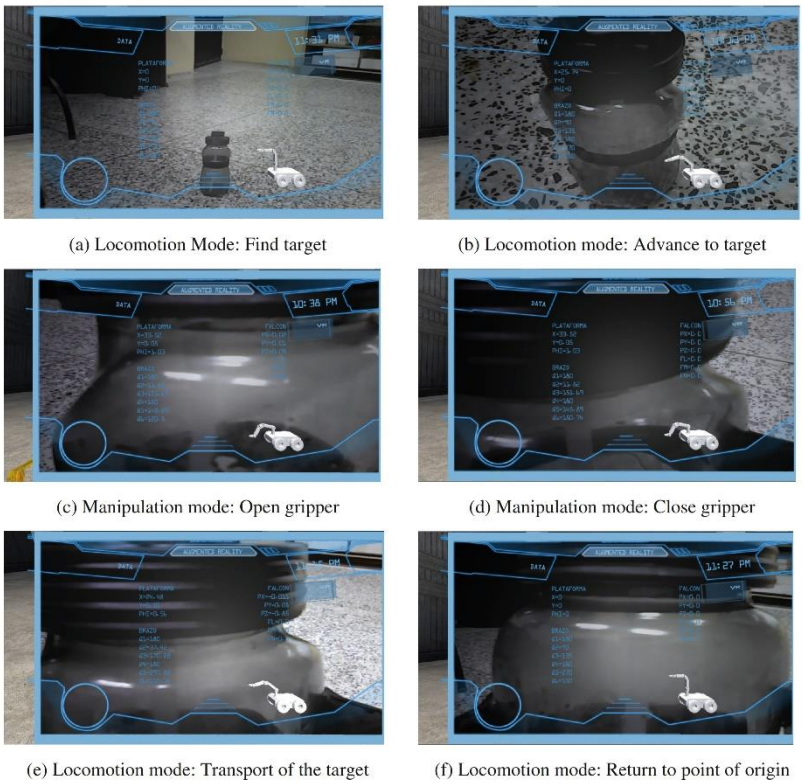
(c) Manipulation mode: Open grippe (d) Manipulation mode: Close gripper

(e) Locomotion mode: Transport of the target (f) Locomotion mode: Return to point of origi

Fig. 13. Snapshots of the experiment on virtual environment

(a) Video image feedback (b) Robot Internal configuration

Fig. 14. Visual feedback of the remote site



(a) Locomotion Mode: Find target

(b) Locomotion mode: Advance to target

(c) Manipulation mode: Open gripper

(d) Manipulation mode: Close gripper

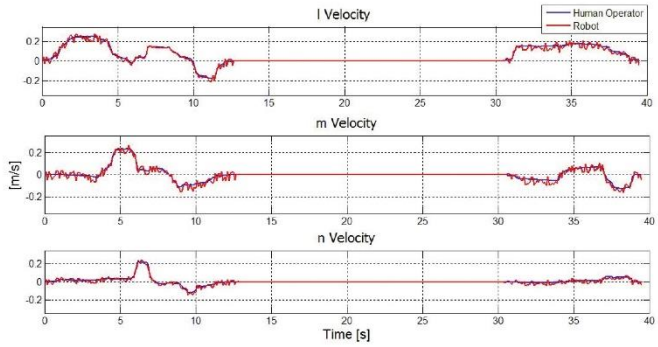
(e) Locomotion mode: Transport of the target

(f) Locomotion mode: Return to point of origin

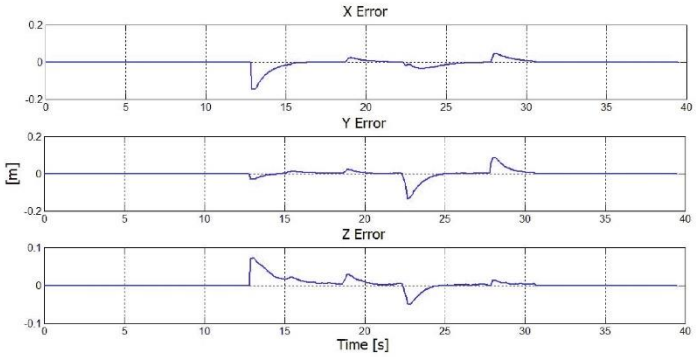
Fig. 15. Visual feedback of the remote site.

robot is plotted on the bottom right. The model simulates the movement based on the actual data transmitted from the local site. Additionally, the augmented reality allows the user to set the desired posture of the robotic arm and displays actual velocities of the robot and reference velocities sent by the user (Fig. 15).

Figure 16 shows a comparison between the reference generated by the human operator and the actual velocities of the end-effector.



(a) Commands of velocity of the end-effector



(b) Error position of the end-effector

Fig. 16. (a) Comparison between the reference generated by the human operator and the actual velocities and positions of the end-effector; while (b) depicts the time evolution of the control error of the robotic arm in manipulation mode

5 Conclusions

To increase the transparency of the proposed tele-operation system, two human-robot interface 3D were implemented, an interface shown a virtual environment while the second is an augmented reality; each interface is based on real experimental data. Experimental results were also presented showing the feasibility and the good performance of the proposed tele-operation structure. In this system is considers that human operator has greater transparency of the remote site. This system allows a human operator to perform complex tasks in a remote environment with a mobile manipulator robot. The teleoperation system integrates switching of control signals generated by the master to control the whole mobile manipulator system, or to control the robotic arm only.

Acknowledgment. The authors would like to thanks to the Consorcio Ecuatoriano para el Desarrollo de Internet Avanzado -CEDIA-, Universidad de las Fuerzas Armadas ESPE, Universidad Técnica de Ambato and the Escuela Superior Politécnica del Chimborazo for financing the project *Tele-operación bilateral cooperativo de múltiples manipuladores móviles – CEPRAIX-2015-05*, for the support to develop this paper.

References

1. Lawrence, D.A.: Stability and transparency in bilateral teleoperation. IEEE Trans. Robot. Autom. **9**(5), 624–637 (1993). 2015 24th IEEE International Symposium on Communication (RO-MAN), pp. 431–437. IEEE
2. Carlson, J., Murphy, R.R.: How UGVs physically fail in the field. IEEE Trans. Rob. **21**(3), 423–437 (2005)
3. Brunet, P., Vinacua, A.: Sistemas Gráficos Interactivos. Universidad Politécnica de Cataluña. Barcelona, España, Mayo de 2006. http://www.lsi.upc.edu/∼pere/SGI/guions/ArquitecturaRV.pdf
4. Desbats, P., Geffard, F., Piolain, G., Coudray, A.: Force-feedback teleoperation of an industrial robot in a nuclear spent fuel reprocessing plant. Ind. Robot Int. J. **33**(3), 178–186 (2006)
5. Mukherjee, J.K.: Fast visualisation technique for view constrained tele-operation in nuclear industry. In: 2014 International Conference on Information Science and Applications (ICISA), pp. 1–4. IEEE (2014)
6. Sanchez, J.G., Patrao, B., Almeida, L., Perez, J., Menezes, P., Dias, J., Sanz, P.: Design and evaluation of a natural interface for remote operation of underwater robots. IEEE Comput. Graphics Appl. **1**, 1 (2015)
7. Christ, R.D., Wernli Sr., R.L.: The ROV manual: a user guide for remotely operated vehicles. Butterworth-Heinemann (2013)
8. Al Mashagbeh, M., Khamesee, M.B.: Unilateral teleoperated master-slave system for medical applications. IFAC-PapersOnLine **48**(3), 784–787 (2015)
9. Livatino, S., De Paolis, L.T., D'Agostino, M., Zocco, A., Agrimi, A., De Santis, A., Lapresa, M.: Stereoscopic visualization and 3-D technologies in medical endoscopic teleoperation. IEEE Trans. Ind. Electronics **62**(1), 525–535 (2015)
10. Andaluz, V.H., Salinas, L., Roberti, F., Toibero, J.M., Carelli, R.: Switching control signal for bilateral tele-operation of a mobile manipulator. In: 2011 9th IEEE International Conference on Control and Automation (ICCA), pp. 778–783. IEEE (2011)
11. Freund, E., Rossmann, J.: Proyetive virtual reality: Bringing the gap between virtual reality and robotic. IEEE Trans. Robot. Autom. **15**(3), 411–422 (1999)
12. Najmaei, N., Asadian, A., Kermani, M., Patel, R.: Design and Performance Evaluation of a Prototype MRF-based Haptic Interface for Medical Applications (2015)
13. Farkhatdinov, I., Ryu, J.H.: Switching of control signals in teleoperation systems: Formalization and application. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2008, AIM 2008, pp. 353–358. IEEE (2008)
14. Pacchierotti, C., Tirmizi, A., Prattichizzo, D.: Improving transparency in teleoperation by means of cutaneous tactile force feedback. ACM Trans. Appl. Percept. (TAP) **11**(1), 4 (2014)
15. Song, G., Guo, S., Wang, Q.: A Tele-operation system based on haptic feedback. In: 2006 IEEE International Conference on Information Acquisition, pp. 1127–1131. IEEE (2006)

16. Willaert, B., Reynaerts, D., Van Brussel, H., Vander Poorten, E.B.: Bilateral teleoperation: quantifying the requirements for and restrictions of ideal transparency. IEEE Trans. Control Syst. Technol. **22**(1), 387–395 (2014)
17. Tanzini, M., Tripicchio, P., Ruffaldi, E., Galgani, G., Lutzemberger, G., Avizzano, C.A.: A novel human-machine interface for working machines operation. In: 2013 IEEE RO-MAN, pp. 744–750. IEEE (2013)
18. Okamura, A.M.: Methods for haptic feedback in teleoperated robot-assisted surgery. Ind. Robot Int. J. **31**(6), 499–508 (2004)
19. Khatib, O.: Mobile manipulation: The robotic assistant. Robot. Auton. Syst. **26**(2/3), 175–183 (1999)
20. Das, Y., Russell, K., Kircanski, N., Goldenberg, A.: An articulated robotic scanner for mine detection-a novel approach to vehicle mounted systems. In: Proceedings of the SPIE Conference, USA, pp. 5–9 (1999)
21. Andaluz, V., Roberti, F., Toibero, J., Carelli, R.: Adaptive unified motion control of mobile manipulators. Control Eng. Pract. **20**(12), 1337–1352 (2012)
22. Martin, S., Hillier, N.: Characterization of the novint falcon haptic device for application as a robot manipulator. In: Australasian Conference on Robotics and Automation (2009)
23. Andaluz, V., Salinas, L., Roberti, F., Toibero, J., Carelli, R.: Switching control signal for bilateral tele-operation of a mobile manipulator. In: 2011 9th IEEE International Conference on Control and Automation (ICCA), Santiago, Chile, pp. 778–783, 19–21 December 2011
24. Slawiñski, E., Mut, V.: Transparency in time for teleoperation systems. In: 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, pp. 200–205, 19–23 May 2008
25. Oculus Ready PCs: Full Rift Experience System Recommendations, Abril de 2016. https://www.oculus.com/en-us/oculus-ready-pcs

Artículo Publicado N°2

Unity3D Virtual Animation of Robots with Coupled and Uncoupled Mechanism

Víctor Hugo Andaluz[1,2(✉)], Jorge S. Sánchez[1], Jonnathan I. Chamba[3], Paúl P. Romero[3], Fernando A. Chicaiza[1], Jose Varela[2], Washington X. Quevedo[1], Cristian Gallardo[2], and Luis F. Cepeda[1]

[1] Universidad de las Fuerzas Armadas ESPE, Sangolquí, Ecuador
{vhandaluz1,jssanchez,fachicaiza,wjquevedo,lfcepeda}@espe.edu.ec
[2] Universidad Técnica de Ambato, Ambato, Ecuador
jazjose@hotmail.es, cmgallardop@gmail.com
[3] Escuela Superior Politécnica de Chimborazo, Riobamba, Ecuador
{jonnathanchamba,p_romero}@espoch.edu.ec

Abstract. This paper presents the development of the animation of robots in virtual reality environments, whose mechanisms can be coupled -the movement relies on mechanical principles-; and uncoupled mechanisms, *i.e.*, the degrees of freedom are controlled independently via a control unit. Additionally, the present phases to transfer the design of a robot developed in a CAD tool to a virtual simulation environment without being lost the physical characteristics of the original design are showed, for which it is considered the various types of motions that the robot can perform depending on the design. Finally, shows the results obtained from the simulation of motion of a robot hexapod 18DOF and Theo Jansen mechanism.

Keywords: Virtual reality · Robot simulation · Unity3D · Coupled and uncoupled mechanism

1 Introduction

In recent years, robotics research has experienced a significant change. Research interests are moving from the development of robots for structured industrial environments to the development of autonomous mobile robots operating in unstructured and natural environments [1–8]. The advancement of technology has allowed the development of computers that support increasingly real and complex simulations in different areas of research and industry. Based on this progress, tools are designed continuously to replicate reality in completely virtual environments. One such tool is the graphic engine developed by Unity Technologies, Unity3D, which allows to develop virtual environments for a wide range of platforms [9] and operating systems.

The simulation process begins with the creation of the 3D model of the robot, considering the aspects of modeling, translation of the format and application. In *modeling phase* is designed the three-dimensional structure of each of the parts and elements of the robot using a tool Computer Aided Design (CAD) [10], which pursues

L.T. De Paolis and A. Mongelli (Eds.): AVR 2016, Part I, LNCS 9768, pp. 89–101, 2016.
DOI: 10.1007/978-3-319-40621-3_6

the physiognomic credibility of the robot taking into account the considerations that facilitate mobility for future animation effects. While the *Translation CAD format phase* to a supported format for Unity3D animation is achieved by additional software, due to hierarchy requirements it must contain certain elements. Finally, *application phase* represents the controlled movement of the 3D model in a similar way to a real environment, taking into account that certain parts move and rotate with respect to each other, for which you should consider animation techniques.

For the simulation of the 3D model, the hierarchy of the elements of the model are consider [10]. This consideration represents a set of rigid parts that are attached to other (anterior and posterior) through joints, so that in the articulated hierarchy is considered: *(i) Control by fixed points,* taking into account that the position of the root segment in space can be allocated independently and allow to move the entire articulated figure for the space; however, sometimes the position of another segment of the hierarchy can be directly determine and hold it in place during movement; for example, when walking one foot remains in contact with the ground while the other foot is moving, and then the other foot support is changed; to reproduce this behavior can begin assigning the node representing the fixed foot as the root node, from which the position of the other segments is calculated; the root node can be change from one foot to another and rearrange the hierarchy, given that the changes made in each joint will be invested; *(ii) Additional restrictive ligatures*, in this case the articulated structures can be combined with parameter values of the joints, so the number of degrees of freedom is reduced and it is easier to get the movements in the different parts of the structure; for example, the two arms of a human figure can move together, linking their angles by transfer functions that match.

In this context, this paper proposes the animation of coupled and uncoupled robotic mechanisms. Additionally, the phases for moving the robot from a design CAD tool to a virtual environment simulation is presented, maintaining the physical characteristics of the original design. Finally, the simulation results are presented in the Unity3D platform, where the different types of motion for a hexapod robot 18DOF (uncoupled mechanism) and other robot based on the mechanism of Theo Jansen (coupled device) formed by three mechanisms system by side and independently controlled by two DC motors are showed.

This paper is divided into 6 Sections including the Introduction. In Sect. 2 the control problem is formulated. Next in Sect. 3 the modeling of the mobile manipulator robot and the controllers design for path following are presents. While the bilateral communication between MatLab-Unity3D is present in Sect. 4. In Sect. 5 the experimental results for of autonomous control and tele-operated for a robotic arm are presented and discussed. Finally, the conclusions are given in Sect. 6.

2 Problem Formulation

The designs of robotic prototypes developed in software Computer Aided Design CAD, SolidWorks, Autodesk Inventor, among others, do not provide all the necessary conditions for a graphic simulation of a realistic 3D model and its environment. The need for

simulation focused on the interaction between the human operator, the robot and a more realistic and immersive environment, has made the simulation function of the CAD software are not sufficient to achieve this objective, because the CAD software does not have a high quality graphics compared to professional digital animations and video game made in other simulation software.

Unity3D is one of the main commercial graphics engines oriented to video games that have graphics quality in the simulation and the processing in the physical environment variables (gravity, collisions and deformations among others) that needs a 3D model for a more realistic interaction between robot-environment, enabling integration of software and hardware, *e.g.*, Leap motion, Oculus, Novint Falcon^{MT}, ROS, among others.

Limiting of support of Unity files from CAD software is not direct, therefore, is necessary to carry out intermediate modifications to integrate operation model designed. In this context, this work proposes a simple methodology to generate 3D movement of robots depending on their mechanism of movement and/or degrees of freedom, for this is considered a robot based on the mechanism of Theo Jansen, and hexapod robot with 18 DOF. The latter robot hasn't degrees of freedom related to joints, but with rotations that obeying the mechanics of the 3D model, as shown in Fig. 1.

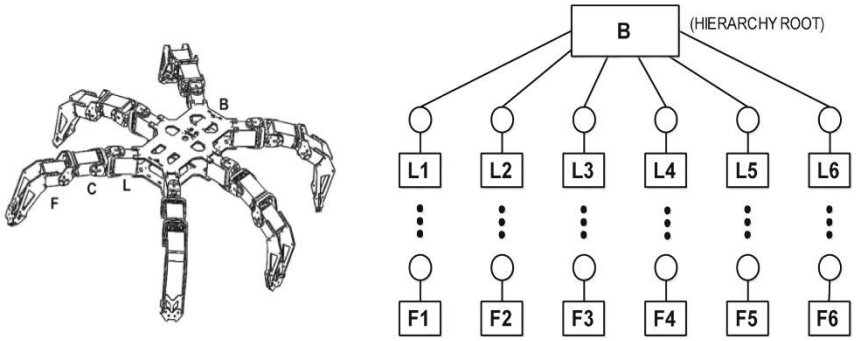


Fig. 1. Hierarchy scheme

The simulation in function of the hierarchy considers each rigid segment is defined with respect to a reference system centered on the origin of the joint. Therefore, each time is changed the position of a joint belonging to the hierarchy you should: *(i)* move the coordinate system from its previous position to the point where it will connect the next segment; *(ii)* once there, you have to rotate the coordinate system according to the state of the joint, that is, according to the target values.

3 Modeling 3D, CAD

The software Automation Mechanical Design SolidWorks is a design tool parametric solid modeling based on operations that leverages the ease of learning the graphical interface. This tool can create 3D solid models fully associative with or without

restrictions, while simultaneously using automatic reactions or user defined to capture design intent. A geometric solid model contains all surface geometry and wireframe necessary to describe in detail the edges and model faces [11].

For this work, the design of a hexapod robot and other based on the mechanism of Theo Jansen are considered. The hexapod robot consists of 18 servomotors Dynamixel AX-12A, which have the ability to feed back to the control unit: speed, temperature, position, torque, to be controlled independently according to control criteria implemented [12]. The structure of the hexapod robot has a central base in which is located the control system, feed, and six servo motors that serve for tips axis robot are fixed, as illustrated in Fig. 1.

The Fig. 2 shows one of the six extremities of hexapod robot, the same which has three degrees of freedom (including the servomotor located in the central base). This design is done with the objective that the robot moves on smooth or uneven surfaces, *i.e.,* wooded uneven surfaces, environments, rocks, sand environments, among others (Fig. 3).

By having the central base of the robot with fixed servomotors for each of the limbs, these provide independent movement there between therefore according to the application the robot can have different types of locomotion, e.g., non-holonomic or omnidirectional.

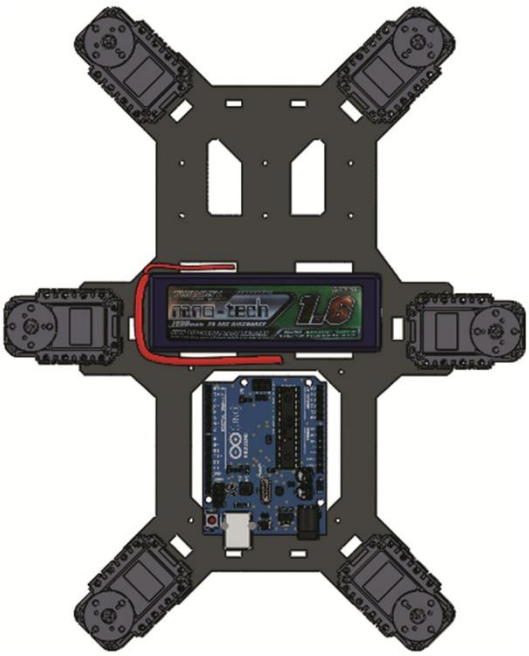
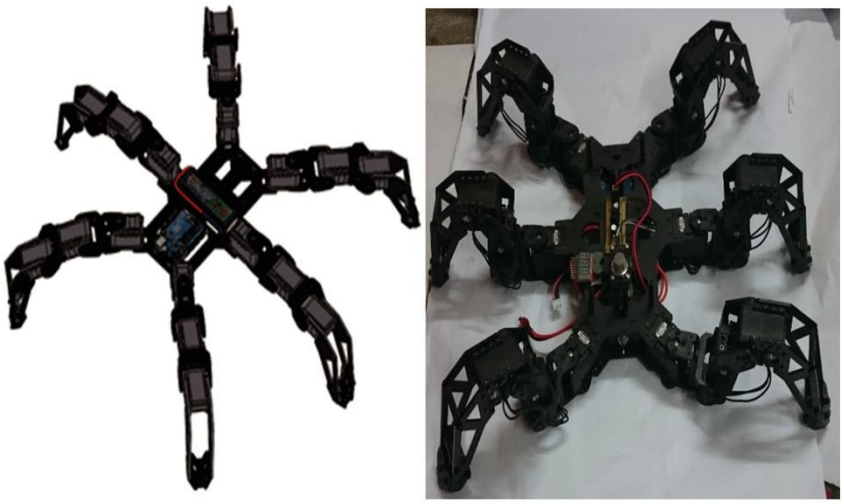


Fig. 2. Hexapod robot: central base

Fig. 3. Hexapod robot: assembly of an extremity

The final result of the assembly of robot shown in Fig. 4(a); while the physical construction of the robot shown in Fig. 4(b). The hexapod robot is based on a polymethylmethacrylate structure because it is lightweight and has high resistance to deformation.



(a) Hexapod robot assembly (b) Robot physical construction

Fig. 4. Hexapod robot 18DOF

The design of a robot based in the *Theo Jansen* mechanism consists of a group of linked bars strategically, such that the movement generated is similar to the limbs of a spider, as shown in Fig. 5.

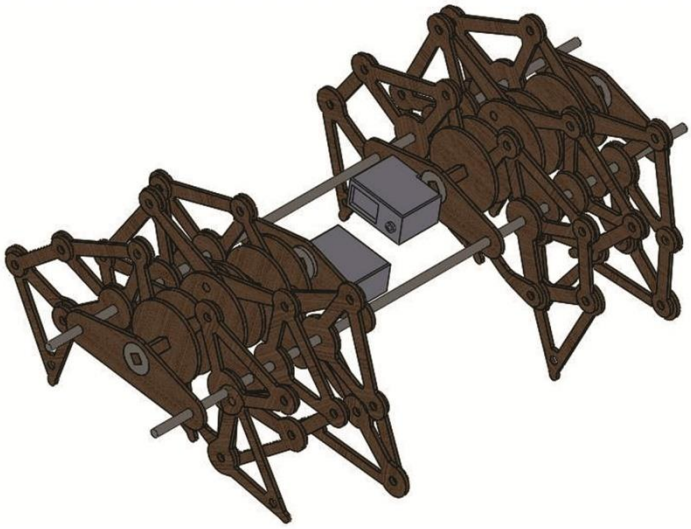


Fig. 5. Spider robot fully assembled

The design has performed will have two extremities, each leg has six joints, two servo motors which are responsible for generating movement; these are associated with each extremity allowing the movement generated is transmitted to all elements of each joint as shown in Fig. 6.

Fig. 6. Extremity assembled of spider robot

4 Export the 3D Model to Unity3D

The process to generate animation of 3D model of a robot in a virtual environment is divided into 6 stages: obtaining 3D model, addition of hierarchies, establish the kind of movement, add textures, generate the motion and finally, apply the animation, view Fig. 7:

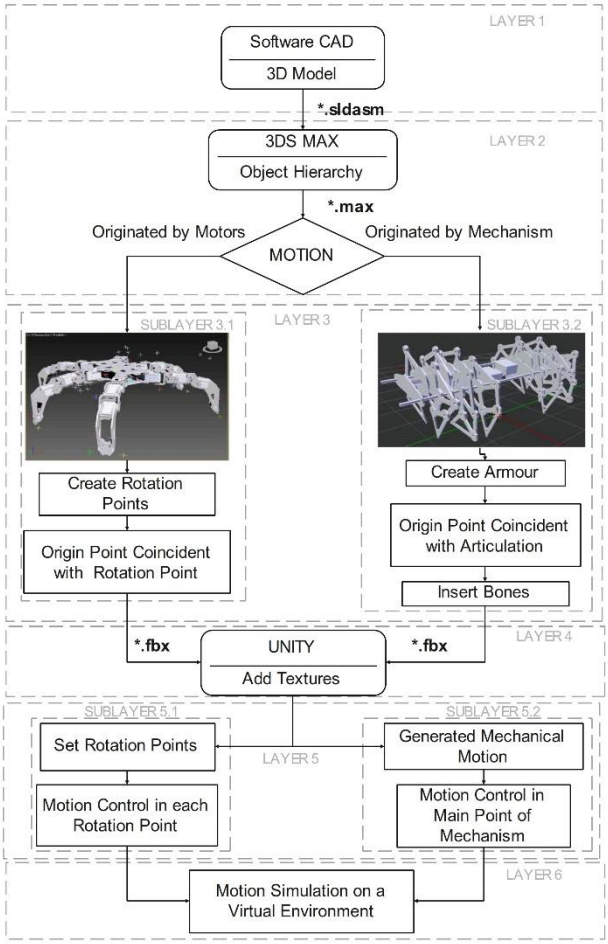


Fig. 7. Process diagram

The step of obtaining the 3D model presents the process of assembling the robot in SolidWorks, which exports the 3D model file with an extension *.sldasm. Given that Unity does not support files with the extension generated by SolidWorks, the step of addition of hierarchies makes use of Autodesk 3DS MAX software. 3DS MAX can import the file generated by CAD software, also create hierarchies, which allow you to have the relationship of positions of each one of them elements of the model. When importing, each component of the 3D model maintains its position in the assembly space of but their relations and constraints of placement are lost during the process, which is necessary to use hierarchies and get a file with extension *.max. At the stage of establishing the type of movement, considering the type of locomotion that has the 3D model, which can be coupled mechanical movement (locomotion system based on caterpillars);

or mechanical movement uncoupled (mechanism composed engines). The uncoupled mechanical motion requires the use of pivot points that are created by using 3DS MAX software. In all joints of the 3D model, each point of origin of movement must match the location of the points of rotation, and in turn, should preferably be located in the axes of each motor, as shown in Fig. 8.

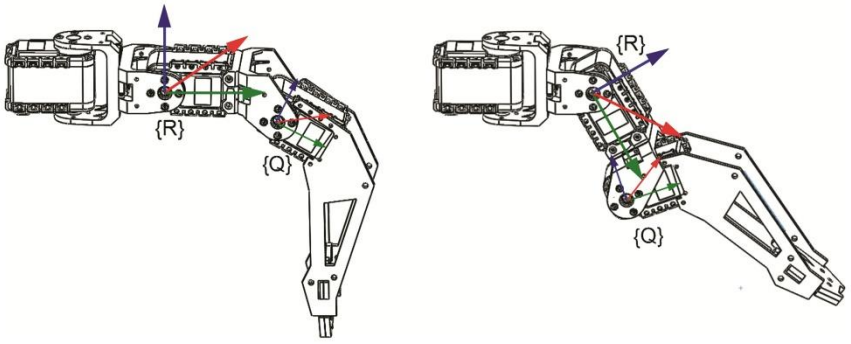


Fig. 8. Rotation points

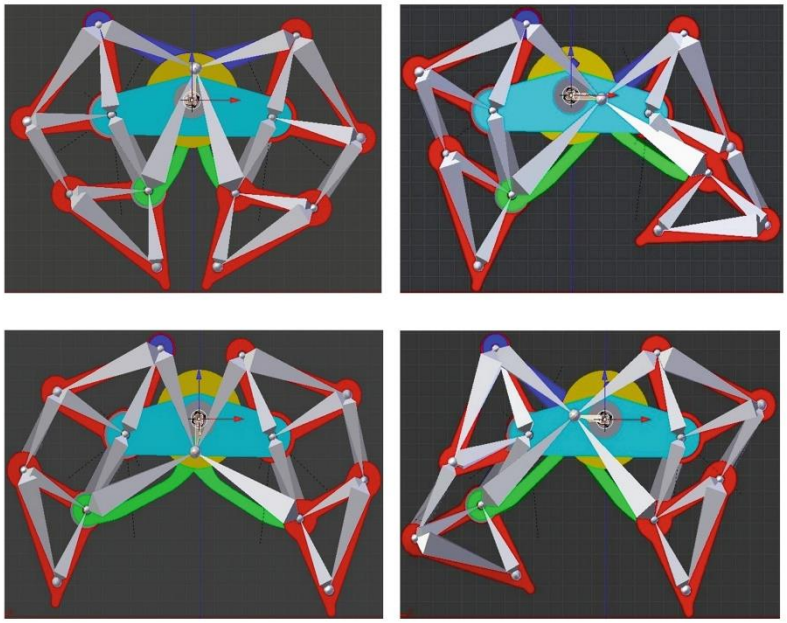


Fig. 9. Coupled motion

The coupled mechanical movement it was developed in Blender because it allows do this kind of armor. An armature acts as a skeleton which can move bones, these in turn, transmit the movement to each object that forms the 3D model. The location of the reference system should be the same as the point where the mechanical movement is generated -main articulation-, from this point, the movement will be transmitted through the bones to the various components of the 3D model as shown in the Fig. 9.

In the step of adding textures is imported the file generated in Blender or 3D MAX. The Fig. 10 shows a file that does not have textures, which have added by Unity to generate a similar actual appearance.

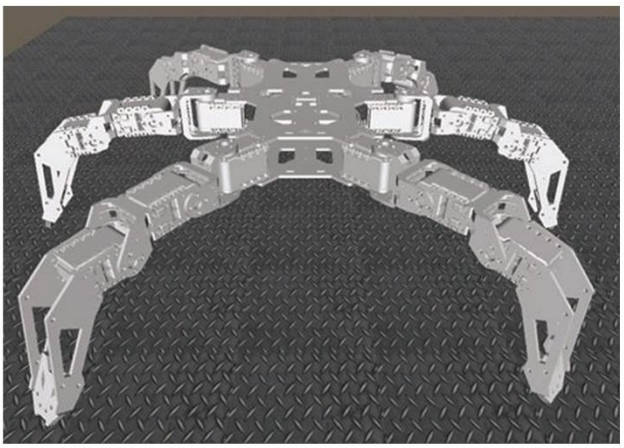


Fig. 10. Untextured hexapod robot

In the step of generating movement, it considering the use of pivot points, wherein the motion control in each of these points is established; and the use of armor where the motion control is set to the point where locomotion mechanism is generated. Finally, in the simulation stage, is added physical variables to the control set in the motion generation layer by integrating haptic devices, audio and video variables.

5 Simulation Experimental Results

The feasibility of the proposed structure for the simulation of designs for the developed robots, it has been proven through experiments in real form using a haptic device Novint Falcon^{MT}, to control movement of the hexapod robot and the other based on the mechanism of Theo Jansen. The simulation is performed in a desktop computer that has the following characteristic I7 2.3 GHz processor, 8 GB of RAM and a graphic card NVIDIA GeForce GT 630M.

5.1 Environment 3D Simulation

The platform on which the 3D simulation environment develops is Unity; on this simulation platform is built the robots and displacement thereof based on the control haptic device shown. Figure 11 shows the environment created for simulation, in this environment can generate the physical variables involved in a real environment.

In this environment is performed the movement of hexapod robot, which has 18 DOF. In the Fig. 12 is observed the principle of mechanical movement uncoupled, which is based on establish a point of origin of motion in each joint of the 3D model, these points are located on the axes of each motor.

To simulate is used a virtual environment unstructured, where it intends to make the inspection for which you can select between two types of displacement through the haptic device (Novint Falcon^{MT}). In the Fig. 12 is noted non-holonomic displacement type, where the robot has only perpendicular speeds to the axis of the motors, allowing a unidirectional movement, forward and backward.

The Fig. 13 shows the omni-directional movement of the hexapod, where the robot can be moved laterally, with this the displacement of the robot is guided by the two linear velocities, defined in a rotating right-handed spatial frame, and the angular velocity. The operator can drive the robot at will depending on the type of movement you want to do to move over the work area. The operation mode selection is performed by one of the buttons has the Novint Falcon^{MT}, also control in three-dimensional to drive the robot.

Similarly was made the robot simulation based on the mechanical system of Theo Jansen, the mechanism of each limb is formed by 7 solids, 5 joints and 2 fixed areas, the mechanism is driven by a pair of motors, the movement is based on the coupled mechanical movement, its principle is the armor, this acts as a skeleton can move joints mechanically, as shown in Fig. 14. This mechanism does not allow the lateral displacement of the joints, which is limited to a linear velocity perpendicular to the axis from the limbs and an angular velocity, like a mobile robot type unicycle. That can rotate freely around its axis. The term unicycle is often used in robotics to mean a generalized cart or car moving in a two-dimensional world; these are also often called unicycle-like or unicycle-type vehicles. On the other hand, the non-holonomic velocity constraint of the robot determines that it can only move perpendicular to the legs axis.

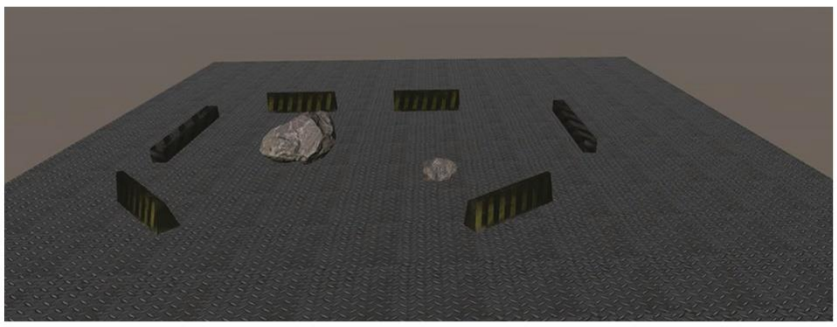


Fig. 11. Virtual environment

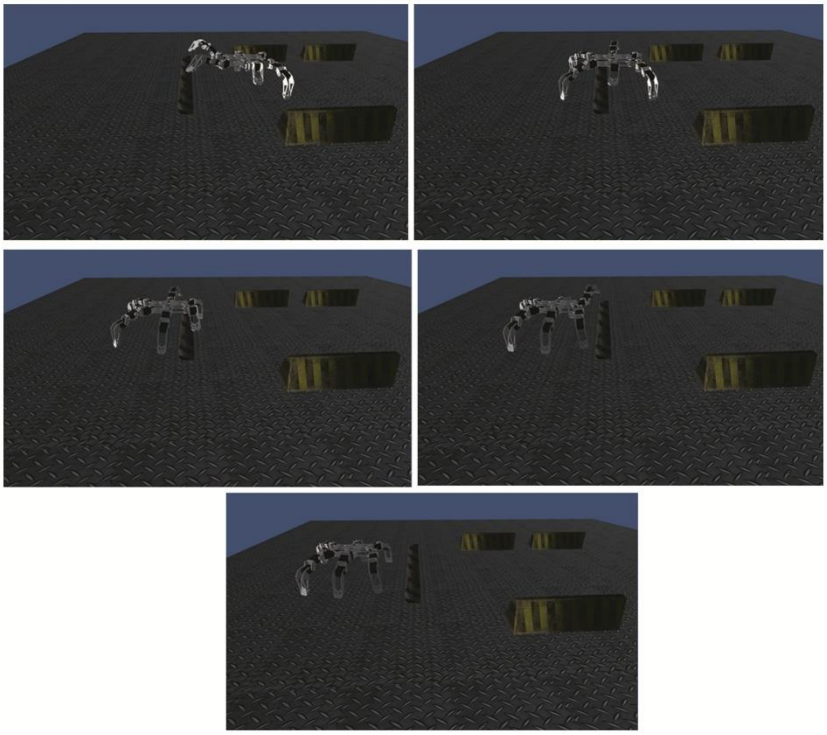


Fig. 12. Hexapod robot non-holonomic displacement

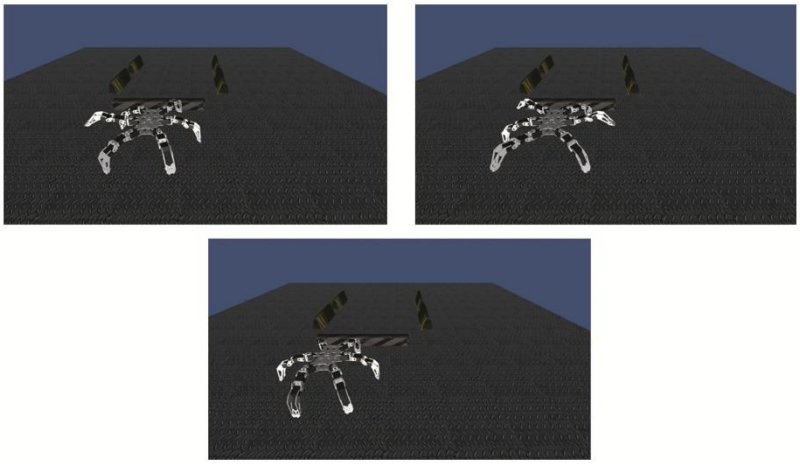


Fig. 13. Hexapod robot omni-directional displacement

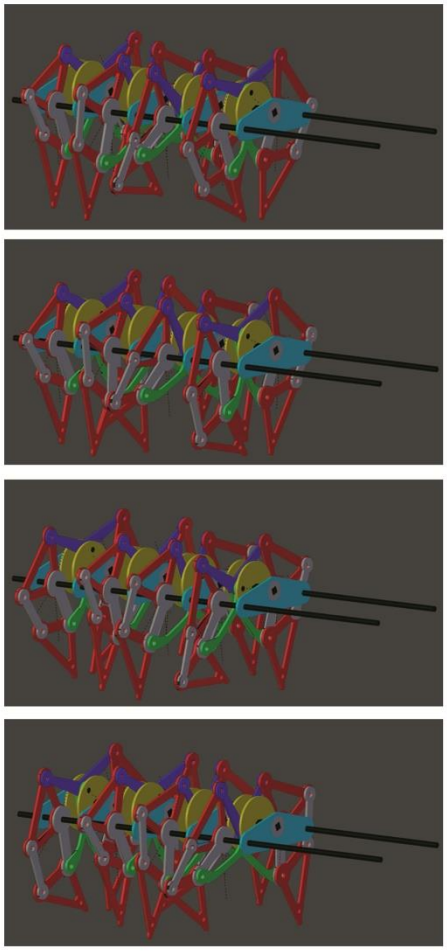


Fig. 14. Displacement of robot based on the of Theo Jansen mechanism

6 Conclusions

This work has focused on developing motion simulation of a hexapod robot and other based on the mechanism Theo Jansen in a virtual unstructured environment. The paper presents an innovative environment that facilitates the assays different control algorithms in various environments that can develop safely. Among the main contributions of the virtual environment is the ease you have to change the design of the work environment and to very simply increase the number and type of experiments to be performed.

Acknowledgment. The authors would like to thanks to the Consorcio Ecuatoriano para el Desarrollo de Internet Avanzado -CEDIA-, Universidad de las Fuerzas Armadas ESPE, Universidad Técnica de Ambato and the Escuela Superior Politécnica del Chimborazo for financing the project *Tele-operación bilateral cooperativo de múltiples manipuladores móviles – CEPRAIX-2015-05*, for the support to develop this paper.

References

1. Andaluz, V.H., Quevedo, W.X., Chicaiza, F.A., Varela, J., Gallardo, C., Sanchez, J.S.: Transparency of a bilateral tele-operation scheme of a mobile manipulator robot. In: SALENTO AVR International Conference on Augmented and Virtual Reality (2016)
2. Shekhar, H., Guha, R., Juliet, A.V., Kumar, J.: Mathematical modeling of neuro-controlled bionic arm. In: International Conference on Advances in Recent Technologies in Communication and Computing, pp. 576–578 (2009)
3. Andaluz, V.H., et al.: Nonlinear controller of quadcopters for agricultural monitoring. In: Bebis, G., et al. (eds.) ISVC 2015. LNCS, vol. 9474, pp. 1–12. Springer, Heidelberg (2015). doi:10.1007/978-3-319-27857-5_43. ISBN 978-3-642-25485-7
4. Ison, M., Vujaklija, I., Whitsell, B., Farina, D.: High-density electromyography and motor skill learning for robust long-term control of a 7-DoF robot arm. In: IEEE Transactions on Neural Systems and Rehabilitation Engineering (2015)
5. Andaluz, V.H., Ortiz, J.S., Sanchéz, J.S.: Bilateral control of a robotic arm through brain signals. In: De Paolis, L.T., Mongelli, A. (eds.) AVR 2015. LNCS, vol. 9254, pp. 1–14. Springer, Heidelberg (2015). ISBN 978-3-642-25485-7
6. Kiguchi, K., Hayashi, Y.: Estimation of joint torque for a myoelectric arm by genetic programming based on EMG signals. In: WAC World Automation Congress, pp. 1–4 (2012)
7. Ranky, G.N., Adamovich, S.: Analysis of a commercial EEG device for the control of a robot arm. In: IEEE Annual Northeast Bioengineering Conference (2010)
8. Gauthaam, M. Kumar, S.S.: EMG controlled bionic arm. In: NCOIET Innovations in Emerging Technology, pp. 111–114 (2011)
9. Indraprastha, A., Shinozaki, M.: The investigation on using Unity3D game engine in urban design study. J. ICT Res. Appl. **3**(1), 1–18 (2009)
10. Zwart, S.F.P., McMillan, S.L., van Elteren, A., Pelupessy, F.I., de Vries, N.: Multi-physics simulations using a hierarchical interchangeable software interface. Comput. Phys. Commun. **184**(3), 456–468 (2013)
11. SolidWorks Corporation. Conceptos básicos de SolidWorks Piezas y ensamblajes (2006)
12. http://support.robotis.com

Artículo Publicado N°3

Unity3D-MatLab Simulator in Real Time for Robotics Applications

Víctor Hugo Andaluz[1,2(✉)], Fernando A. Chicaiza[1], Cristian Gallardo[2], Washington X. Quevedo[1], José Varela[2], Jorge S. Sánchez[1], and Oscar Arteaga[1]

[1] Universidad de las Fuerzas Armadas ESPE, Sangolquí, Ecuador
{vhandaluzl, fachicaiza, wxquevedo, jssanchez, obarteaga}@espe.edu.ec
[2] Universidad Técnica de Ambato, Ambato, Ecuador
cmgallardop@gmail.com, jazjose@hotmail.es

Abstract. This paper presents the implementation of a new 3D simulator applied to the area of robotics. The simulator allows to analyze the performance of different schemes of autonomous and/or tele-operated control in structured environments, partially structured and unstructured. For robot-environment interaction is considered virtual reality software Unity3D, this software exchanges information with MATLAB to execute different control algorithms proposed through the use of shared memory. The exchange of information in real time between the two software is essential because the advanced control algorithms require a feedback from the robot-environment interaction to close the control loop, while the simulated robot updates its kinematic and dynamic parameters depending on controllability variables calculated by MATLAB. Finally, the 3D simulator is evaluated by implementing an autonomous control scheme to solve the problem of path following of a 6DOF robot arm, also the results obtained by implementing the tele-operation scheme for said robot are presented.

Keywords: Simulator 3D · Virtual reality simulator · Path following · Unity3d-MATLAB · Shared memory

1 Introduction

In recent years, robotics research has experienced a significant change. Research interests are moving from the development of robots for structured industrial environments to the development of autonomous mobile robots operating in unstructured and natural environments [1–5]. The robotic generally is classified according to their field of application, industrial robotics and service robotics [6–8]. In industrial and service applications it is necessary to avoid mistakes, they can cause economic and human losses; in this context it is necessary to have an environment in which to experience the performance of robots before they pass to perform any task in a real environment, for which it is considered a virtual simulation environment.

L.T. De Paolis and A. Mongelli (Eds.): AVR 2016, Part I, LNCS 9768, pp. 246–263, 2016.
DOI: 10.1007/978-3-319-40621-3_19

A virtual environment is an environment in which simulations activities found in everyday life are made, this is done with the purpose of bringing these activities to a controlled environment and analyze more deeply the stability and robustness of the systems designed, permitting in this virtual environment test, you may experience various system disturbances, and thus obtain a complete study of the operation of the system.

The advancement of technology has developed computers that let you simulations increasingly real and complex in different areas. A virtual environment would be divided into: *(i) interactive environment* it means that the user is "free" to navigate the virtual environment without having programmed the trajectory that you want to move, the system responds according to the user's wishes, this represents that the user can make decisions in "real time" in order to observe the scene from the selected viewpoint [9, 10]; *(ii) implicit interaction* this refers to the user must not learn commands or a procedure to perform some action in the virtual world, by contrast, the user performs movements that are natural to those used in the real world to move. It then searches the computer suits human nature and not the other, thereby ensuring that the experience in the virtual environment is as near as possible to the experience in the real world [9, 10]; and *(iii) sensory immersion* refers to disconnect sense the real world and the connection thereof to the virtual world [10].

The virtual environment was initially developed for application in computer games and consoles, recently the virtual environments are used to simulate different applications in the area of robotics. There are several commercial programs for the design and simulation of robots in virtual environments, between to simulate the behavior of any robot model are: Robcad, Robotstudio, Igrip, Workcell, Gazebo [11], etc.; specific for a robot in particular, *e.g.* V_CAT, V_TRAISIG y V_ISUAL of Staubli, not all programs are compatible with other CAD systems, do not support libraries all robots or other elements if any, and some are not sold under Windows, in this context, a software that is compatible with most CAD systems is sought, Unity3D for which the platform is analyzed.

Unity3D is a graphics engine developed by Unity Technologies in order to allow everyone to create attractive 3D environments, its creation was aimed at creating games. Unity3D possible to develop software for a wide range of platforms [12–14], so it is extremely attractive for a wide range of developers. For the simulation of a system is considered: *(i) 3D design*, this is done with special or general CAD programs; at this stage in addition to the three-dimensional drawing of the installation (environment modeling) the kinematic and dynamic characteristics of robots and other mobile elements of the system are defined; *(ii) trajectories following*, movements, velocities and sequences are determined; and *(iii) simulation of all movements,* the possibilities at this stage of the installation are checked, errors are corrected, the interference is detected and design are optimized.

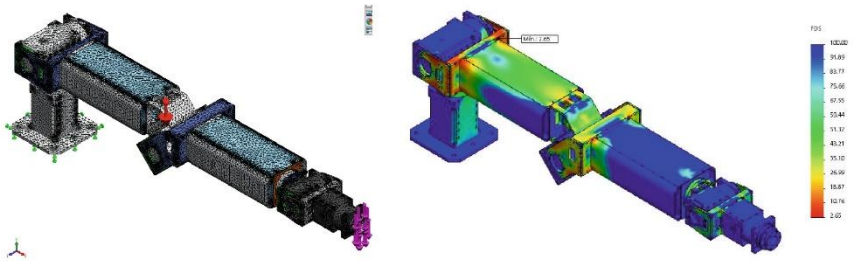
As mentioned above, this paper presents a new 3D virtual reality simulator for robotic applications. The proposed simulator allows real-time communication between Unity3D and MATLAB software. For bilateral communication it proposes to use shared memory between these two software; the method of shared memory is a technique easy to apply, with short delays and low use of computer resources by not calling functions third. In addition, the simulator allows to evaluate real-time

performance of different schemes of autonomous control and/or tele-operated in structured, partially structured environments, and unstructured; for tele-operation scheme 3D simulator accepts as input haptics devices that stimulate the senses of the human operator so that it can "transmit" their skill, experience and expertise to the robot to perform a task. Finally, to evaluate the performance of autonomous control simulator for monitoring roads proposed for a 6DOF robotic arm -system redundant-, as secondary objective is considered the maximum arm manipulability; also, the experimental test of a scheme bilateral tele-operation is performed.

This paper is divided into 6 Sections including the Introduction. In Sect. 2 the control problem is formulated. Next in Sect. 3 the modeling of the mobile manipulator robot and the controllers design for path following are presents. While the bilateral communication between MATLAB-Unity3D is present in Sect. 4. In Sect. 5 the experimental results for of autonomous control and tele-operated for a robotic arm are presented and discussed. Finally, the conclusions are given in Sect. 6.

2 Problem Formulation

The application development in the area of robotics requires accurately define the task to be performed to determine the needed characteristics of the robot. Determined these parameters, the execution of a task can be subdivided into the following steps: *(i) Modeling stage,* at this stage, it is essential to model the three-dimensional robot in a Computer Aided Design (CAD) software. The modeling lets to analyze the physical characteristics of the robot prior to the construction, in this context, there are tools such as SolidWorks, Autodesk Inventor, AutoCAD, among others, that allow to design mechanical elements and get results of mobility and strength of materials, among other mechanical characteristics, view Fig. 1;



(a) Gridding arm (b) Static analysis of the safety factor

Fig. 1. Ejemplo de un brazo robótico modelado en SolidWorks (Color figure online)

(ii) Construction stage, The main objective of this phase is to assemble each of the mechanical parts designed and incorporate the necessary electronics to move each joint, In addition to considering the signal conditioning for sensory perception of the external environment in which the robot will move, and internal sensors that must issue the

position, velocity, torque and strength of each link forming the robot. The information provided by the proprioceptive sensors and exteroceptive sensors will be used in the different advanced control algorithms proposed; *(iii) Controllers design stage,* to the design of advanced control algorithms it is essential to determine the mathematical model representing the robot kinematics and dynamics. The different mathematical models of robots are systems of multiple-input multiple-output, MIMO, so software tools that solve mathematical matrix operations to facilitate implementation of the proposed control scheme is required. MATLAB is a tool with its own programming language and development environment that offers the advantage of matrix manipulation and data processing. As a deployment scenario algorithms, MATLAB has libraries that can be extended according to programming needs [15]; *(iv) Simulation stage,* prior to the experimental implementation of the proposed control algorithms, it's necessary to check their performance in a three dimensional environment that emulates the actual conditions in which the robot operates, therefore, virtual development tools are required with the ability to support bilateral haptic devices, video output interfaces and audio, among others. Unity 3D is a tool for creating games, as well as development of virtual simulation and allows the incorporation of different haptic devices for manipulating its environment. Unity engine uses a script in C# language to manipulate the game objects with which you can modify the behavior of the simulated objects; and finally the *(v) Implementation stage,* it is the final phase in which the robot interacts with the environment where performs the task, this interaction is controlled via algorithms proposed control. The successful implementation of the planned task is based on compliance with each of the objectives of the above detailed steps; in this context, one can say that the design and simulation of control algorithms are the most critical stages for performing a task, so this work focuses on these two items.

In order to check the performance of control schemes in simulated/emulated environments, it is necessary to implement a communication channel between a bilateral graphics engine and mathematical software tool. The exchange of information in real time between the two applications is essential because the advanced control algorithms require the robot's feedback to close the control loop, while the emulated robot updates its kinematic and dynamic parameters depending on the robot-environment interaction, view Fig. 2.

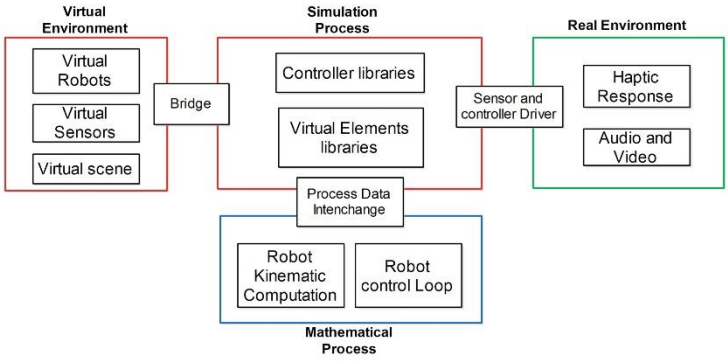


Fig. 2. Data interchange between Math software and 3D simulation software

In this context, the following sections show an emulator of advanced control algorithms implemented in MATLAB and displayed in real time in a virtual environment developed in Unity3D. It should be noted that the emulator allows bilateral interaction between MATLAB and Unity3D for any scheme or control technique implemented in MATLAB. As an example, autonomous control for tracking paths of a robotic arm 6 DOF is presented; and a tele-operated control from Unity's robotic arm.

Remark 1. A simulator represents reality in a similar way, while an emulator replica or improved conditions similar to the real ones.

3 Modeling and Control

The *instantaneous kinematic model of a robotic arm* sets the derivative of its location as a function of the derivative of its configuration (or its *operational velocities* as functions of its *generalized velocities*)

$$\dot{\mathbf{h}}(t)=\mathbf{J}(\mathbf{q})\dot{\mathbf{q}}(t). \tag{1}$$

It uses the Jacobian matrix $\mathbf{J}(\mathbf{q})$ of the function f: $\mathbf{J}(\mathbf{q}) = \frac{\partial f}{\partial \mathbf{q}}$. The configurations such that the rank of $\mathbf{J}(\mathbf{q})$ decreases are singular kinematic configurations and the problem, robotic arm and task, is redundant when $n > m$.

The mathematic model that represents the dynamics of a robotic arm can be obtained from Lagrange's dynamic equations, which are based on the difference between the kinetic and potential energy of each of the joints of the robot (energy balance). Most of the commercially available robots have low level PID controllers in order to follow the reference velocity inputs, thus not allowing controlling the voltages of the motors directly. Therefore, it becomes useful to express the dynamic model of the robotic arm in a more appropriate way, taking the rotational and longitudinal reference velocities as the control signals. To do so, the velocity controllers are included in the model. The dynamic model of the robotic arm, having as control signals the reference velocities of the system, can be represented as follows,

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \dot{\mathbf{q}}_{\mathbf{ref}} \tag{2}$$

where, $\mathbf{M}(\mathbf{q}) = \mathbf{H}^{-1}(\bar{\mathbf{M}} + \mathbf{D})$, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{H}^{-1}\left(\bar{\mathbf{C}} + \mathbf{P}\right)$, $\mathbf{g}(\mathbf{q}) = \mathbf{H}^{-1}\,\bar{\mathbf{g}}(\mathbf{q})$. Thus, $\bar{\mathbf{M}}(\mathbf{q}) \in \Re^{\delta_n \times \delta_n}$ is a positive definite matrix, $\bar{\mathbf{C}}(\mathbf{q},\mathbf{v})\mathbf{v} \in \Re^{\delta_n}$, $\bar{\mathbf{G}}(\mathbf{q}) \in \Re^{\delta_n}$ and $\dot{\mathbf{q}}_{\mathbf{ref}} \in \Re^{\delta_n}$ is the vector of velocity control signals, $\mathbf{H} \in \Re^{\delta_n \times \delta_n}$, $\mathbf{D} \in \Re^{\delta_n \times \delta_n}$ and $\mathbf{P} \in \Re^{\delta_n \times \delta_n}$ are constant symmetrical diagonal matrices, positive definite, that contain the physical parameters of the robotic arm, *e.g.,* motors, velocity controllers.

In the other hand, a trajectory will be automatically generated and a trajectory tracking control will guide the robotic arm to the desired target. As indicated, the fundamental problems of motion control of robots can be roughly classified in three groups: (1) *point stabilization*: the goal is to stabilize the robot at a given target point, with a desired orientation; (2) *trajectory tracking*: the robot is required to track a time parameterized reference; and (3) *path following*: the robot is required to converge to a

path and follow it, without any time specifications. For more details about the modeling and control see [16].

4 Bilateral Communication MATLAB-Unity3D

This section describes the methods of inter-process communication and exchange of information between MATLAB and Unity3D for control of an emulated manipulator with a haptic device.

4.1 Windows Inter-process Communication (IPC)

IPC is a feature enabled in the operating systems on which processes can exchange information through memory segments or through own communication tools, allowing resource sharing. Generally these processes are developed to low level – allowing to interact with the operating system resources – and according to the protocols for such communication.

Table 1. Windows Inter-process communication methods [17]

Method	Advantages	Disadvantages	Resources
Named Pipe	Easy to use and works across the network	The source code is platform dependent	Medium
WinSock	Works on the same computer as well as across networks. Moreover, it can be used across various platforms and protocols	Requires a knowledge of relatively advanced networking concepts	Low
Mailslots	Works across a network and supports broadcast	Provides one-way communication only	Medium
Shared Memory	**Linking processes using memory registers previously allocated, without functions of third party**	**Works on the same computer**	**Low**

The techniques to develop IPC vary depending on the application. This function can be used for the transmission of messages, synchronization, shared memory and remote procedures. The method used to communicate processes depends on the transfer rate required and the type of data to be treated. There are several ways to implement communication between processes, among which are: *(i) Named Pipe* is a method of channeling data by creating a memory space in the operating system explicitly declared before the execution of processes to communicate.; *(ii) WinSock* provides very high level networking capabilities, it supports TCP/IP (the most widely used protocol) along with many other protocols like AppleTalk, DECNet, IPX/SPX, etc.; *(iii) Mailslots* processes messages between applications via datagrams and allows to communicate unidirectionally, this method is useful for transmitting information to multiple clients;

and finally, *(iv) Shared memory* allows to create segments of memory to be accessed by multiple processes, access restrictions may be defined, *e.g.*, read only, read and write, execute, access over inheritance, among others. Table 1 presents the differences between the methods described for implementation of inter-processes communication.

Remark 2. Datagram is a data set of the communication protocol packet switched used to route information between nodes in a network.

In reference to illustrated in Table 1 and the proposed implementation guidance in this work, the method of *shared memory* is a technique easy to apply, with short delays and low use of computer resources by not calling third party functions.

4.2 MATLAB – Unity Communication

The bidirectional data communication between MATLAB - Unity3D is performed by a dynamic-link-library, dll, in which the Shared Memory method is implemented, SM, in RAM. The Fig. 3 illustrates the implementation of shared memory, where the dll manages the SM space, besides providing permits for the applications, label the memory space, provide functions to modify/obtain the stored information and liberate the space when the application is terminated.

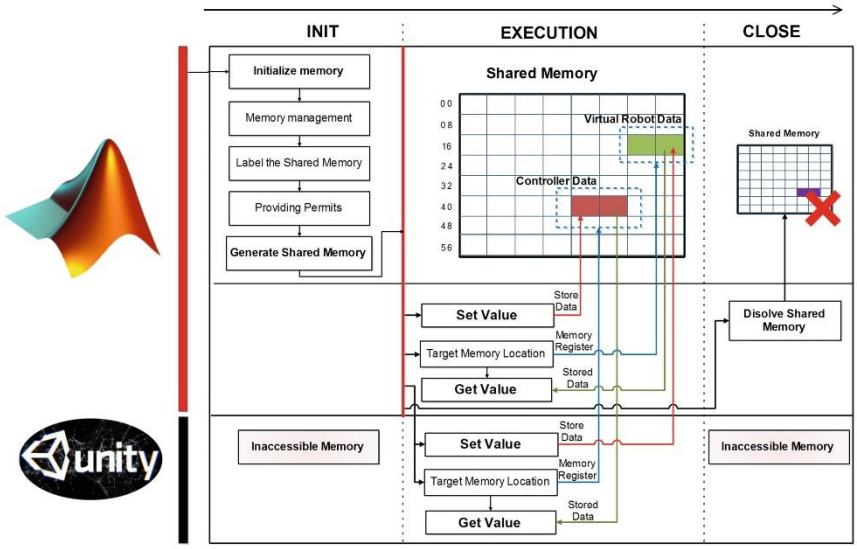


Fig. 3. Interprocess communication via shared memory

Using the dll between MATLAB and Unity3D is divided into three parts: ***(i) Init phase,*** the dll can be instantiated from an application through a handle, in which are set the security attributes and inheritance, permissions to read/write to memory registers reserved, RAM's space management and labeling. In this way, the client applications

can reference data wishing to modify or capture, provided it have access permissions, aware of their existence and location where it is staying.

Remark 3. The characteristics of the dll allows that the generated memory can be started from MATLAB or Unity3D. From this step, both applications must use functions to identify dedicated spaces of memory, modify data or get them.

```
void CreateSharedMemoryArea()
{
        hFile = CreateFileMapping(INVALID_HANDLE_VALUE,        //Handle to instantiate the dll
created
                NULL,                                          //Null Security attributes and heritage
                0x40,                                          //Read&Write permissions
                0,
                1024 * 4,                                      //Memory Space Managed
                _T("memoriaza"));                              //Label the shared memory
        if (hFile == NULL)                                     // Memory Validation
        {
                printf("Unable to create a file.");
                exit(1);
        }
}
```

(ii) Execution phase, at this stage MATLAB and Unity3D must invoke the function *OpenSharedMemory()* to find the handle through the label and create a memory view, defining the read/write permissions, the point where the view begins and the number of bytes to be mapped, view Fig. 4. The view points to handle, from which a casting to LPINT type variables is made to locate the index of each of the stored data. The handle is referenced by the view when the application desires to read or write in memory.

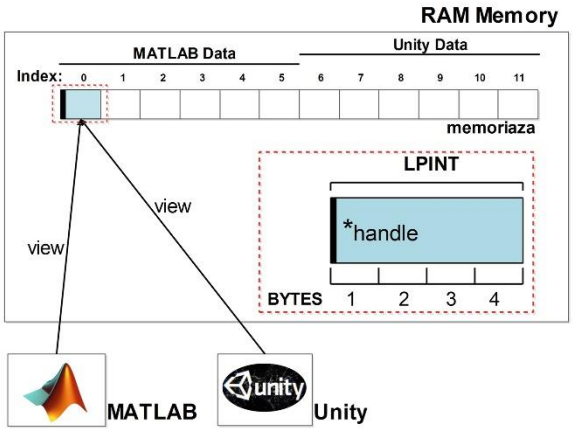


Fig. 4. Views of the Shared Memory

```
void OpenSharedMemory()
{
  hFile = OpenFileMapping(FILE_MAP_ALL_ACCESS,                    // Search tag memory
                          FALSE,
                          _T("memoriaza"));

        if (hFile == NULL)                                        // Validate the existence of
the memory
        {
                printf("Unable to open the shared area.\n");
                exit(1);
        }
        hView = (LPINT) MapViewOfFile(hFile,                      // Creating view type LPINT
                FILE_MAP_ALL_ACCESS,                              // Read / Write definition
                0,                                                    // Point where the view
begins
                0,
                0);                                               // All memory map

        if (hView == NULL)                                        // View Validation
        {
                printf("Unable to create a VIEW.\n");
                exit(1);
        }
        aux = hView;                                                  // Handle referenced to
Read or Write registers
}
```

The view allows update dedicated registers of each application. For writing data, the application is based on indexes that the dll provides in static manner for each of the variables.

```
void WriteOnSharedMemory(int data, int position)                  // data defines the value,
position defines the index
{
        aux[position] = data;
}
```

```
void ReadFromSharedMemory(int *data, int position)                // data defines the value,
position defines the index
{
        *data = aux[position];
}
```

Finally ***(iii) Close phase,*** SM is reserved while the process runs. When the application *Close*, memory must be released. By invoking the function DestroySharedMemoryArea(), ends with the reservation and labeling of RAM for that any other system process can use it.

```
    void DestroySharedMemoryArea ()                    // Free the shared memory
    {
            if (!UnmapViewOfFile(hView))
            {
        printf("Could not unmap view of file.");
            }
            CloseHandle(hFile);
            printf("The end.\n");
}
```

4.3 Interaction MATLAB-Unity3D

The interaction between MATLAB and the graphics engine is divided into three stages, described in the following paragraphs as: import of three-dimensional design, interaction human-robot and bilateral communication processes (Fig. 5).

SolidWorks is the CAD tool used for mechanical design, but has no export formats supported by the virtual tool development. In the first phase, 3DS Max is used to modify parameters SolidWorks 3D modeling and hierarchies are established in the pieces that make up the assembly and supported file by Unity3D is exported. Once the three-dimensional model imported to Unity 3D environment, texture for each piece that makes up the prototype are established. In addition, the degrees of freedom of the 3D model is specified by activating the points of rotation and/or translation for objects that guides each.

In the second stage, the Unity3D environment performs the animation of virtual objects using Game Objects, scripts and plugins. The behaviour of Objects Game is controlled by scripts, which allow you to modify its properties and respond to user input as scheduled. The plugins allow you to use native functionality (support Oculus Rift) or include external code (support Novint Falcon). The human-robot interaction is achieved by information from input devices (Falcon encoders, Tracking Oculus HMD), the mathematical tool uses this data to return control actions and generate output responses (Falcon motors, Oculus HMD and audio).

Finally, in the third stage, information virtual robot is linked to MATLAB through the dll file and the invocation of the SM. When MATLAB requires send or retrieve information from the SM sector, its programming should include the lines:

```
    loadlibrary('./dll64MATLAB.dll','./simple.h');          // Invoking the dll
    calllib ('dll64MATLAB','initMemory');                   // Initialize memory
    calllib ('dll64MATLAB','openMemory');                     // Create the view of the
shared memory
    calllib ('dll64MATLAB','setValue',v1,v2,v3,v4,v5,v6,v7,v8,v9,v10);          // Set values in the
memory
    val1 = calllib ('dll64MATLAB','getValue1');                  // Get values  from
memory
    calllib ('dll64MATLAB','destroy');                      // Free shared memory
```

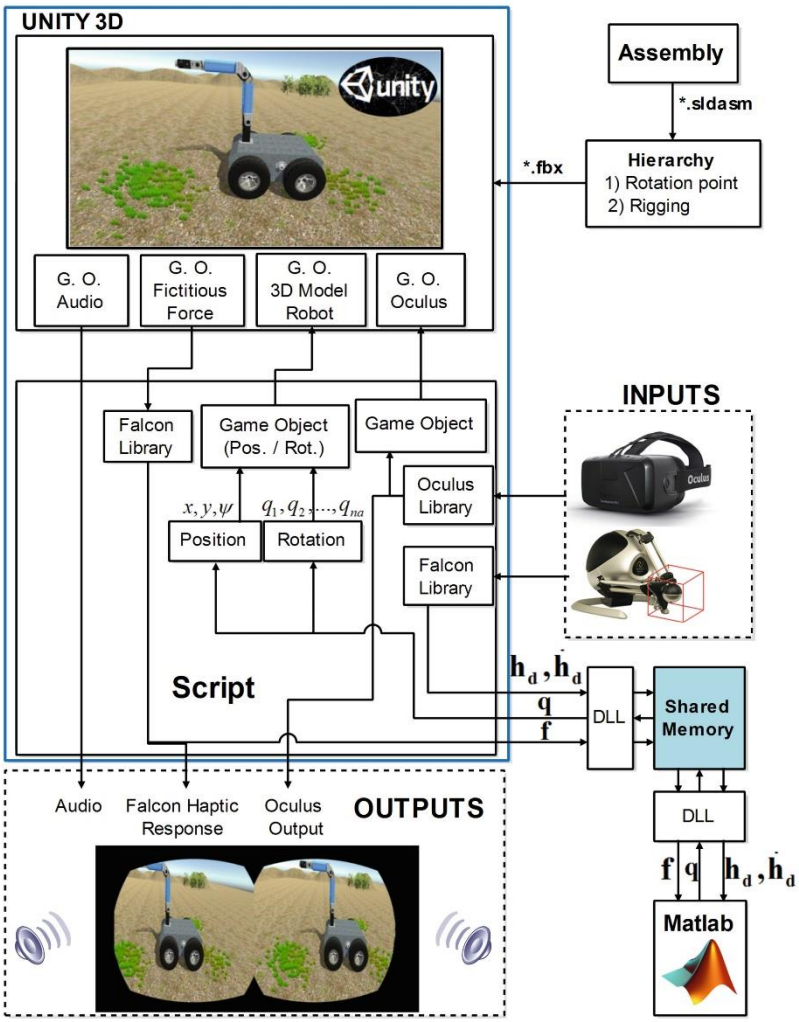


Fig. 5. Unity3D – MATLAB Interaction

While in each communication cycle, Unity receives velocities to control the rotation points and thus the operative end. Additionally, is sent the position of each actuator forming the modeled 3D robot and data human-robot, such as real interaction forces given by contact, effects of gravity, fictitious forces to avoid virtual obstacles, etc. Unity within the script, must contain the following lines of programming to invoke the read/write data in the SM.

```
    [DllImport(@"C:\Users\Public\Documents\Unity
Projects\Animación\Assets\Plugins\dll64MATLAB.dll")]
       private static extern void openMemory();                    // Create the view of the
shared memory
    [DllImport(@"C:\Users\Public\Documents\Unity
Projects\Animación\Assets\Plugins\dll64MATLAB.dll")]
       private static extern int getValue1();                      // Get values  from
memory
    PosX = -getValue1() / 100;                                     // Using the get value

    [DllImport(@"C:\Users\Public\Documents\Unity
Projects\Animación\Assets\Plugins\dll64MATLAB.dll")]
       private static extern int setValue(int v1, int v2,..., int v10);
    setValue(val1,val2,...,val10);                                 // Set values in the
memory
```

Remark 4: The libraries developed for information sharing allow interaction between Unity Game Objects with any software package of MATLAB like Script, Simulink, etc., once initialized the SM.

Remark 5: In the case of robotic applications, update time data is relatively low due at time of sampling used. This work do not try to raise synchronization methods of information in shared memory.

5 Simulation Experimental Results

In order to illustrate the performance of the proposed simulator 3D of an arm robotic 6DOF, several experiments were carried out for path following autonomous control and bilateral tele-operation of a robotic arm; the most representative results are presented in this Section. The experiments were carried with the kinematic and dynamic models of a robotic arm 6 DOF, view Fig. 6.

Fig. 6. Arm Robotic 6DOF developed in SolidWorks

On the other hand, the proposed simulator 3D consist in a desktop computer Core I7 3610QM running at 2.3 GHz, 8 GB RAM and a NVIDIA GeForce GT 630 M 1 GB dedicated graphics; also the local site has a haptic device Falcon^{MT} Novint. The evaluation of the latency in MATLAB takes into account the transmission, execution Unity and receiving data delay, which is within the desired sampling period of 100 [ms].

5.1 Autonomous Control

The performance of the control structure for path following is tested. The desired trajectory for the end-effector is described by $P(s) = (x_P(s), y_P(s), z_P(s))$, where $x_P = 0.35\sin(0.2s + \frac{\pi}{2})$; $y_P = 0.35\cos(0.2s + \frac{\pi}{2})$ and $z_P = 0.2 + 0.8\sin(0.1s)$. Note that for the path following problem, the desired velocity of the end-effector of the robotic arm will depend on the task, the control error, the joint velocity of the arm, among other design specifications. In this experiment, it is considered that the reference velocity module depends on the desired velocity of the end-effector on path P and the control errors. Then, reference velocity in this experiment is expressed as $|\mathbf{v_{hd}}| = \frac{v_P}{1+k\|\mathbf{h}\|}$, where k is a positive constant that weigh the control error module. Also, the desired location is defined as the closest point on the path to the end-effector of the experimental system.

Hence, Figs. 7, 8, 9 show the results of the experiment of autonomous control. Figure 7 shows the stroboscopic movement on the *X-Y-Z* space of Unity3D. It can be seen that the proposed controller works correctly; while the Fig. 8 shows the desired path and the current path of the end-effector of the robotic arm. It can be seen that the proposed controller presents a good performance;

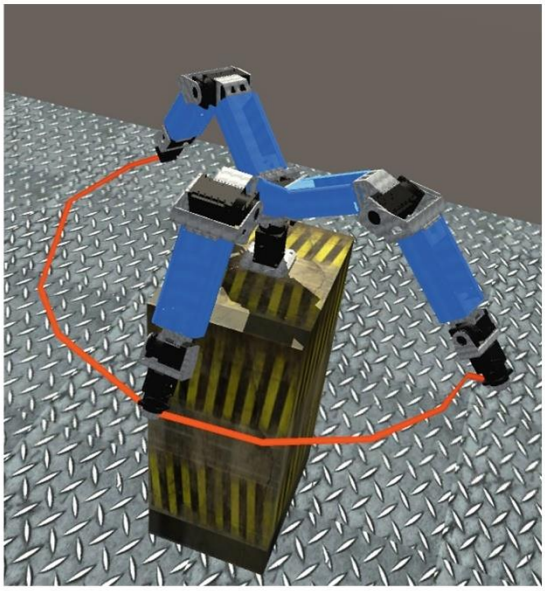


Fig. 7. Stroboscopic movement of the robotic arm in Unity 3D.

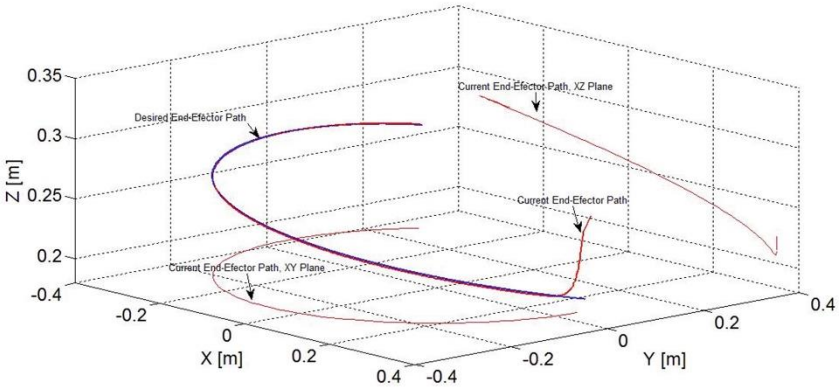


Fig. 8. Desired path and the current path of the end-effector

and finally Fig. 9 shows that the control errors of the robotic arm on the *X-Y-Z* space converge to values close to zero asymptotically.

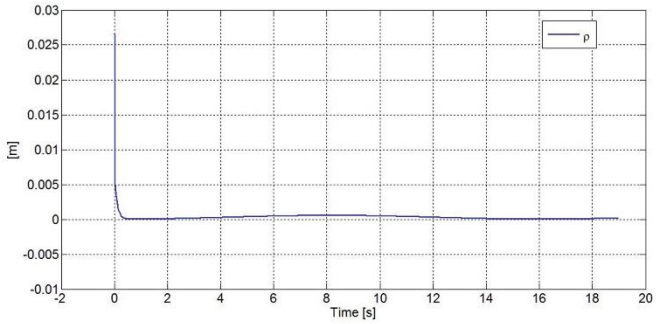


Fig. 9. Distance between the end-effector position and the closest point on the path

5.2 Tele-Operated Control

The feasibility of the proposed simulator 3D is tested through of bilateral tele-operation scheme using a robotic arm 6 DOFs. The local site has an Oculus and a haptic device Falcon^{MT} Novint. The human operator commands are generated with the use of a Falcon^{TM} haptic device from Novint Technologies Incorporated [18] as indicated in Fig. 10. Its positions are translated into desired velocities commands $P(s) = (x_P(s), y_P(s), z_P(s))$ of the end-effector of the robotic arm [19].

The simulation of a bilateral tele-operation scheme is presented, which consists on a grasping task. With this aim, the robot is guided near the object; then the user grasps the object opening the gripper; and finally the robot is guided to drop the object into a box. Obtained results are shown in Figs. 11, 12, 13. Figure 11 shows snapshots of the

Fig. 10. Local site of the tele-operation scheme

experiment on Unity 3D. Figure 12 shows a comparison between the reference generated by the human operator and the actual velocities of the end-effector. While Fig. 13 depicts the time evolution of the control error of the robotic arm.

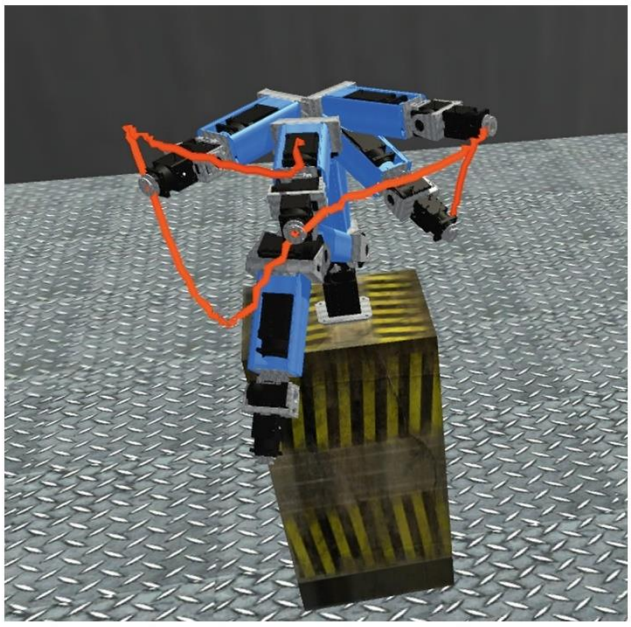


Fig. 11. Bilateral tele-operation: Grasping task on Unity 3D

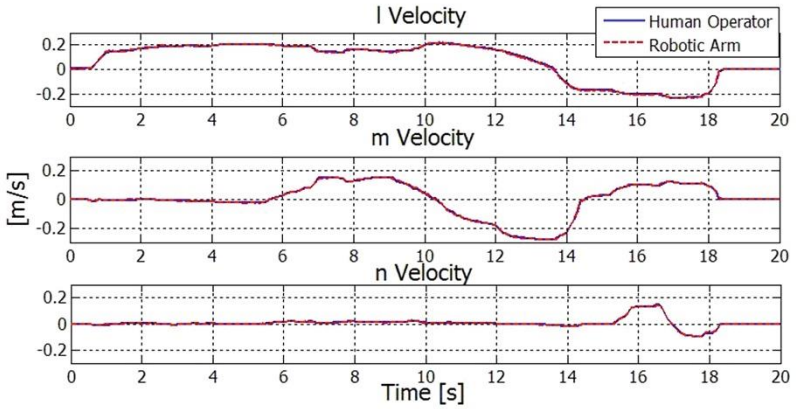


Fig. 12. Comparison between the reference generated by the human operator and the actual velocities of the end effector

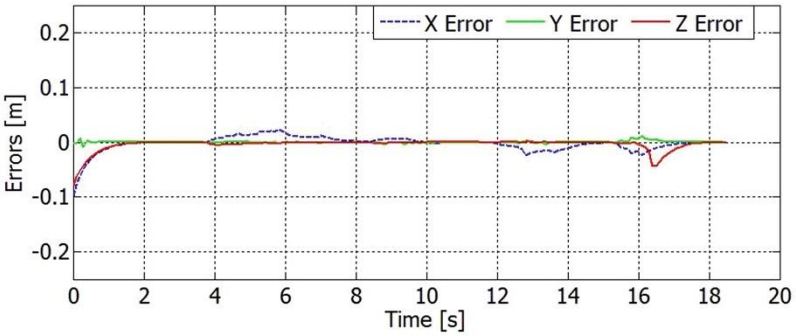


Fig. 13. Evolution of the control errors of the robotic arm. If $\lim_{t\to\infty} \tilde{x}(t) = 0$, $\lim_{t\to\infty} \tilde{y}(t) = 0$ and $\lim_{t\to\infty} \tilde{z}(t) = 0$ then $\lim_{t\to\infty} \rho(t) = 0$.

6 Conclusions

In this paper a 3D simulator in real time for robotics applications is proposed. This simulator considers the bilateral communication between MATLAB-Unity3D through of a dynamic-link-library, dll, in which the method of Shared Memory in RAM is implemented. The dll manages space SM, enable permissions to applications, puts the label to memory space, provide functions to modify/obtain the stored information and freeing the space when the application is terminated. Experimental results were also presented showing the feasibility and the good performance of the proposed simulator 3D; the experiments were carried out for path following autonomous control and bilateral tele-operation of a robotic arm 6DOF.

Acknowledgment. The authors would like to thanks to the Consorcio Ecuatoriano para el Desarrollo de Internet Avanzado -CEDIA-, Universidad de las Fuerzas Armadas ESPE, Universidad Técnica de Ambato and the Escuela Superior Politécnica del Chimborazo for financing the project *Tele-operación bilateral cooperativo de múltiples manipuladores móviles – CEPRAIX-2015-05*, for the support to develop this paper.

References

1. Andaluz, V.H., López, E., Manobanda, D., Guamushig, F., Chicaiza, F., Sánchez, J.S., Rivas, D., Pérez, F., Sánchez, C., Morales, V.: Nonlinear controller of quadcopters for agricultural monitoring. In: Bebis, G., et al. (eds.) ISVC 2015. LNCS, vol. 9474, pp. 476–487. Springer, Heidelberg (2015). doi:10.1007/978-3-319-27857-5_43
2. Andaluz, V.H., Chicaiza, F.A., Meythaler, A., Rivas, D.R., Chuchico, C.P.: Construction of a quadcopter for autonomous and tele-operated navigation. In: IEEE-DCIS Conference on Design of Circuits and Integrated Systems, Portugal (2015)
3. Andaluz, V.H., Canseco, P., Varela, J., Ortiz, J.S., Pérez, M.G., Roberti, F., Carelli, R.: Robust control with dynamic compensation for human-wheelchair system. In: Zhang, X., Liu, H., Chen, Z., Wang, N. (eds.) ICIRA 2014, Part I. LNCS, vol. 8917, pp. 376–389. Springer, Heidelberg (2014)
4. Andaluz, V.H., Ortiz, J.S., Roberti, F., Carelli, R.: Adaptive cooperative control of multi-mobile manipulators. In: IEEE–IECON Industrial Electronics Society, pp. 2669–2675, USA (2014)
5. Andaluz, V.H., Roberti, F., Marcos, T.J., Ricardo, C.: Adaptive unified motion control of mobile manipulators. J. Control Eng. Pract. 1337–1352 (2012). Elsevier Editorial System
6. Andersen, R.S.; Bogh, S.; Moeslund, T.B.; Madsen, O.: Intuitive task programming of stud welding robots for ship construction. In: 2015 IEEE International Conference on IEEE Industrial Technology (ICIT), pp. 3302–3307, March 2015
7. Andaluz, V.H., Ortiz, J.S., Sanchéz, J.S.: Bilateral control of a robotic arm through brain signals. In: De Paolis, L.T., Mongelli, A. (eds.) AVR 2015. LNCS, vol. 9254, pp. 355–368. Springer, Heidelberg (2015)
8. Ying, J.L., Peng, J.S., Qi, Z., Chang, C.L., Yong, H.: The review of workpiece loading and unloading robot in the catenary shot blasting. In: Research and Design of Machinery, Equipment and Technological Processes in Mechanical Engineering, Applied Mechanics and Materials, Vols. 496–500, pp. 578–581 (2014)
9. Freund, E., Rossmann, J.: Proyetive virtual reality: Bringing the gap between virtual reality and robotic. IEEE Trans. Robot. Autom. **15**(3), 411–422 (1999)
10. Brunet, P., Vinacua, A.: Sistemas Gráficos Interactivos. Universidad Politécnica de Cataluña, Barcelona, España, Mayo de 2006. http://www.lsi.upc.edu/∼pere/SGI/guions/ArquitecturaRV.pdf
11. Meyer, J., Sendobry, A., et. al.: Comprehensive simulation of quadrotor UAVs using ROS and Gazebo. In: SIMPAR 2012 Proceedings of the Third International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Berlin, Germany, vol. 12, pp. 400–411 (2015)
12. Oliveira, M., Pereira, N., Oliveira, E., Almeida, J.E., Rossetti, R.J.: A Multi-player Approach in Serious Games: Testing Pedestrian Fire Evacuation Scenarios. Oporto, DSIE15, January (2015)

13. Bartneck, C., Soucy, M., Fleuret, K., Sandoval, E.B.: The robot engine—Making the unity 3D game engine work for HRI. In: 2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 431–437. IEEE (2015)
14. Indraprastha, A., Shinozaki, M.: The investigation on using Unity3D game engine in urban design study. J. ICT Res. Appl. **3**(1), 1–18 (2009)
15. MATLAB and Simulink for Technical Computing. http://www.mathworks.com/
16. Andaluz, V., Salinas, L., Roberti, F., Toibero, J., Carelli, R.: Switching control signal for bilateral tele-operation of a mobile manipulator. In: 2011 9th IEEE International Conference on Control and Automation (ICCA), Santiago, Chile, 19–21 December 2011
17. Interprocess Communications. https://msdn.microsoft.com/en-us/library/windows/desktop/aa365574(v=vs.85).aspx
18. Andaluz, V., Roberti, F., Toibero, J., Carelli, R.: Adaptive unified motion control of mobile manipulators. Control Eng. Pract. **20**(12), 1337–1352 (2012)
19. Martin, S., Hillier, N.: Characterization of the novint falcon haptic device for application as a robot manipulator. In: Australasian Conference on Robotics and Automation (2009)

Anexo B: Scripts de Programación

Scripts de Unity

Control de Scene

```
using UnityEngine;
using System;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class CambioScene : MonoBehaviour {
    public GameObject luz;
    public GameObject panel;
    public GameObject panel2;

    public GameObject ContManipulador1;
    public Slider slX1;
    public Text lblX1;
    public Slider slY1;
    public Text lblY1;
    public Slider slPhi1;
    public Text lblPhi1;
    public GameObject ContManipulador2;
    public Slider slX2;
    public Text lblX2;
    public Slider slY2;
    public Text lblY2;
    public Slider slPhi2;
    public Text lblPhi2;
    public GameObject ContManipulador3;
    public Slider slX3;
    public Text lblX3;
    public Slider slY3;
    public Text lblY3;
    public Slider slPhi3;
    public Text lblPhi3;

    public GameObject atras;
    public Dropdown ddEntorno;
    public GameObject entorno1;
    public GameObject entorno2;
    public GameObject entorno3;
    public Button btnCantidad;
    public Text texto;
    public int cant;
    public GameObject Manipulador1;
    public GameObject Manipulador2;
    public GameObject Manipulador3;
    public int menu;

    // Use this for initialization
    void Start () {

        cant = 1;

    }

    // Update is called once per frame
    void Update () {
```

```
}
public void Movimiento_slx1()
{
    lblX1.text = " X: " + slX1.value;
}
public void Movimiento_slx2()
{
    lblX2.text = " X: " + slX2.value;
}
public void Movimiento_slx3()
{
    lblX3.text = " X: " + slX3.value;
}
public void Movimiento_slY1()
{
    lblY1.text = " Y: " + slY1.value;
}
public void Movimiento_slY2()
{
    lblY2.text = " Y: " + slY2.value;
}
public void Movimiento_slY3()
{
    lblY3.text = " Y: " + slY3.value;
}
public void Movimientoslphi1()
{
    lblPhi1.text = " Phi: " + slPhi1.value + "°";
}
public void Movimientoslphi2()
{
    lblPhi2.text = " Phi: " + slPhi2.value + "°";
}
public void Movimientoslphi3()
{
    lblPhi3.text = " Phi: " + slPhi3.value + "°";
}

public void cambio2()
{
    SceneManager.LoadScene("escena1");
    menu = 1;
}

public void cambio1()
{
    SceneManager.LoadScene("stage1");
    menu = 1;
}

public void cambio()
{
    SceneManager.LoadScene("home");
    menu = 0;
}

public void Atras()
{
```

```
    panel2.SetActive(true);
    atras.SetActive(false);
}
public void Atras2()
{
    panel2.SetActive(false);
    panel.SetActive(true);
}
public void siguiente()
{
    panel.SetActive(false);
    atras.SetActive(true);
    makeEntorno();
    makeRobots();
    panel2.SetActive(true);
}
public void siguiente2()
{
    panel2.SetActive(false);
}

public void Add()
{
    cant = Convert.ToInt32(texto.text)+1;
    texto.text = ( cant).ToString();
}
public void Subs()
{
    cant = Convert.ToInt32(texto.text);
    if (cant > 0)
        cant -= 1;
    else
        cant = 0;
    texto.text = (cant).ToString();
}
public void makeEntorno()
{

    switch (ddEntorno.value)
    {
        case 0:
            //Debug.Log("Entorno 1: Campo");
            entorno1.SetActive(true);
            entorno2.SetActive(false);
            entorno3.SetActive(false);
            luz.SetActive(false);
            break;
        case 1:
            //Debug.Log("Entorno 1: Laboratorio");
            entorno1.SetActive(false);
            entorno2.SetActive(true);
            entorno3.SetActive(false);
            luz.SetActive(false);
            break;
        case 2:
            entorno1.SetActive(false);
            entorno2.SetActive(false);
            entorno3.SetActive(true);
            luz.SetActive(true);
            //Debug.Log("Entorno 1: Ninguno");
            break;
```

```
            default:
                break;
        }
    }
    public void makeRobots()
    {
        switch (cant)
        {
            case 1:
                Manipulador1.SetActive(true);
                ContManipulador1.SetActive(true);
                Manipulador2.SetActive(false);
                ContManipulador2.SetActive(false);
                Manipulador3.SetActive(false);
                ContManipulador3.SetActive(false);
                break;
            case 2:
                Manipulador1.SetActive(true);
                ContManipulador1.SetActive(true);
                Manipulador2.SetActive(true);
                ContManipulador2.SetActive(true);
                Manipulador3.SetActive(false);
                ContManipulador3.SetActive(false);
                break;
            case 3:
                Manipulador1.SetActive(true);
                ContManipulador1.SetActive(true);
                Manipulador2.SetActive(true);
                ContManipulador2.SetActive(true);
                Manipulador3.SetActive(true);
                ContManipulador3.SetActive(true);
                break;
            default:
                break;

        }


    }

}
```

Datos de HUD

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
//using System.DateTime;

public class Datos : MonoBehaviour {
    public Text hora;
    public Text fecha;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        hora.text = System.DateTime.Now.ToLongTimeString();
        fecha.text = System.DateTime.Now.ToLongDateString();
    }
}
```

Manipulador

```
using UnityEngine;
using System.Runtime.InteropServices;
using System;
using System.Collections;

public class manipulador1 : MonoBehaviour {
    //*****VARIABLES DEL SCRIPT CAMBIOSCENE.CS*****//
    public CambioScene CambioScene;

    //*****VARIABLES DEL MANIPULADOR*****//
    public GameObject Robot;
    public float AnguloX = 0;
    public float AnguloY = 0;
    public float AnguloZ = 0;
    public float PosX = 0;
    public float PosY = 0;
    public float PosZ = 0;

    //*****VARIABLES DEL BRAZO*****//
    public GameObject Q1;
    public int AnguloXq1 = 0;
    public float AnguloYq1 = 0;
    public int AnguloZq1 = 0;
    public static int _AnguloYq1 = 180;

    public GameObject Q2;
    public float AnguloXq2 = 0;
    // public int AnguloYq2 = 90;
    public int AnguloYq2 = 0;
    public int AnguloZq2 = 0;
    public static int _AnguloYq2 = 90;

    public GameObject Q3;
    public float AnguloXq3 = 90;
    //public int AnguloYq3 = 130;
    public int AnguloYq3 = 0;
    public int AnguloZq3 = 90;
    public static int _AnguloYq3 = 130;

    public GameObject Q4;
    public int AnguloXq4 = 0;
    public int AnguloYq4 = 180;
    public int AnguloZq4 = 0;
    public static int _AnguloYq4 = 0;

    public GameObject Q5;
    public int AnguloXq5 = 0;
    public int AnguloYq5 = 180;
    public int AnguloZq5 = 180;
    public static int _AnguloYq5 = 0;

    public GameObject Q6;
    public int AnguloXq6 = 0;
    public int AnguloYq6 = 0;
    public int AnguloZq6 = 0;
    public static int _AnguloYq6 = 0;
```

```
///*****VARIABLES MEMORIA COMPARTIDA*****//

[DllImport(@"C:\Users\Washington\Documents\Entorno\Assets\Plugins\sm
Client.dll")]
    private static extern int abrirMemoria(string nombre, int tipo);
[DllImport(@"C:\Users\Washington\Documents\Entorno\Assets\Plugins\sm
Client.dll")]
    private static extern float readFloatValue(string nombre, int
posicion);
[DllImport(@"C:\Users\Washington\Documents\Entorno\Assets\Plugins\sm
Client.dll")]
    private static extern float writeFloatValue(string nombre, int
posicion, float valor);

      // Use this for initialization
    void Start()
    {
        abrirMemoria("Memoria0", 2);
        abrirMemoria("Memoria1", 2);
        abrirMemoria("Memoria2", 2);
        abrirMemoria("Memoria3", 2);
        abrirMemoria("Memoria4", 2);
        abrirMemoria("Memoria5", 2);
        CambioScene = FindObjectOfType<CambioScene>();
    }


    public void moverRobot(float angulograd os)
    {
        AnguloY = -angulograd os;
        Robot.transform.localEulerAngles = new Vector3(AnguloX,
AnguloY, AnguloZ);
    }
    public void TrasladarRobot(float ejeX, float ejeY, float ejeZ)
    {
        PosX = ejeX;
        PosY = ejeY;
        PosZ = ejeZ;
        Robot.transform.position = new Vector3(PosX, PosY, PosZ);
    }
    public void moverQ1(float angulograd os)
    {
        AnguloYq1 = (-angulograd os / 100) * Mathf.Rad2Deg;

        Q1.transform.localEulerAngles = new Vector3(AnguloXq1,
AnguloYq1, AnguloZq1);
    }
    public void moverQ2(float angulograd os)
    {
        AnguloXq2 = ((angulograd os / 100) * Mathf.Rad2Deg) - 90;
        Q2.transform.localEulerAngles = new Vector3(AnguloXq2,
AnguloYq2, AnguloZq2);
    }
    public void moverQ3(float angulograd os)
    {
        AnguloXq3 = ((angulograd os / 100) * Mathf.Rad2Deg) + 90;
        Q3.transform.localEulerAngles = new Vector3(AnguloXq3,
AnguloYq3, AnguloZq3);
    }
    public void moverQ4(int angulograd os)
    {
```

```
        AnguloYq4 = angulogrados;
        Q4.transform.localEulerAngles = new Vector3(AnguloXq4,
AnguloYq4, AnguloZq4);
    }
    public void moverQ5(int angulogrados)
    {
        AnguloYq5 = angulogrados;
        Q5.transform.localEulerAngles = new Vector3(AnguloXq5,
AnguloYq5, AnguloZq5);
    }
    public void moverQ6(int angulogrados)
    {
        AnguloYq6 = angulogrados;
        Q6.transform.localEulerAngles = new Vector3(AnguloXq6,
AnguloYq6, AnguloZq6);
    }

    // Update is called once per frame
    void Update()
    {

        PosX = CambioScene.slX1.value + readFloatValue("Memoria1",
0);
        PosZ = CambioScene.slY1.value + readFloatValue("Memoria1",
1);
        AnguloY = CambioScene.slPhi1.value +
readFloatValue("Memoria1", 2);
        TrasladarRobot(PosX, PosY, PosZ);
        moverRobot(AnguloY);
        moverQ1(readFloatValue("Memoria1", 3));
        moverQ2(readFloatValue("Memoria1", 4));
        moverQ3(readFloatValue("Memoria1", 5));

    }
}
```

Scripts de MATLAB

Trayectoria Programada

```
%*********************************************************************
%************************** MANIPULADOR MÓVIL*************************
%*********************************************************************
clear all;  close all;  clc;

loadlibrary('./smClient.dll','./main.h');
calllib('smClient','abrirMemoria','Memoria0',2);
calllib('smClient','abrirMemoria','Memoria1',2);
calllib('smClient','abrirMemoria','Memoria2',2);
calllib('smClient','abrirMemoria','Memoria3',2);
calllib('smClient','abrirMemoria','Memoria4',2);
calllib('smClient','abrirMemoria','Memoria5',2);

ts=.1;  tfin=60;
t=[0:ts:tfin];

%POSICION INICIAL DEL MANIPULADOR MÓVIL
    %a) Condiciones iniciales
        x(1) = 0;               %posición en el eje x de la
plataforma móvil EN METROS
        y(1) = -0.5;            %posición en el eje y de la
plataforma móvil EN METROS
        z(1) = 0;               %SIEMPRE EN CERO
        u(1) = 0;               %velocidad lineal del PIONEER 3DX
inicial en METROS/s
        w(1) = 0;               %velocidad angular del PIONEER 3DX
inicial en RADIANES/s
        th(1)= 0;               %dirección de la plataforma móvil EN
RADIANES

        rot1(1) = 0;            %articulación de la base del
manipulador EN RADIANES
        rot2(1) = 0*pi/180;     %articulación del primer brazo del
manipulador EN RADIANES
        rot3(1) = 0*pi/180;     %articulación del segundo brazo del
manipulador EN RADIANES

        a  = 0.19;              %distancia del centro de masa del
móvil al marco base del manipulador EN METROS
        h  = 0.37;              %altura de la plataforma móvil + la
altura de la base del manipulador EN METROS
        l1 = 0.143;             %longitud de la primera articulación
EN METROS
        l2 = 0.325;             %longitud de la segunda articulación
EN METROS

%TRAYECTORIA del EE PARAMETRIZADA EN TIEMPO: Obtengo pos y vel
% Trayectoria de una recta
%         EEx = 0.09*t+1;                             %Posición x
%         EEy = 0.5*ones(1,length(t));                %Posición y
%         EEz = 0.7*ones(1,length(t));                %Posición z
%
%         EEvx = 0.09*ones(1,length(t));              %Velocidad x
%         EEvy = 0*ones(1,length(t));                 %Velocidad y
%         EEvz = 0*ones(1,length(t));                 %Velocidad z
```

```
% Trayectoria de una senoidal
        EEx = 0.1*t;
%Posición x
        EEy = 0.3+0.5*sin(0.1*t);%0.25*ones(1,length(t));
%Posición y
        EEz = 0.45+0.2*sin(0.4*t);
%Posición z

        EEvx = 0.1*ones(1,length(t));
%Velocidad x
        EEvy = 0.5*0.1*cos(0.1*t);%0*ones(1,length(t));
%Velocidad y
        EEvz = 0.2*0.4*cos(0.4*t);
%Velocidad z

 %POSICION INICIAL DEL EE
        [xEEreal(1),yEEreal(1),zEEreal(1)] =
posEEmundo(x(1),y(1),th(1),rot1(1),rot2(1),rot3(1),l1,l2,a,h);

%*****************************************************************
%************************CONTROLADOR******************************
%*****************************************************************

 for k=1:length(t)
%1) CÁLCULO DE ERRORES (posición deseada - posición real)
        Xe(k) = EEx(k)-xEEreal(k);              %error en el eje x
        Ye(k) = EEy(k)-yEEreal(k);              %error en el eje y
        Ze(k) = EEz(k)-zEEreal(k);              %error en el eje z

%2) CÁLCULO DE LAS ACCIONES DE CONTROL
    % a) Jacobiano de todo el sistema (manipulador + móvil )
         J11 = cos(th(k));
         J12 = -2*a*sin(th(k))-
sin(rot1(k)+th(k))*(l1*cos(rot2(k))+l2*cos(rot2(k)+rot3(k)));
         J13 = -
sin(rot1(k)+th(k))*(l1*cos(rot2(k))+l2*cos(rot2(k)+rot3(k)));
         J14 = -
cos(rot1(k)+th(k))*(l1*sin(rot2(k))+l2*sin(rot2(k)+rot3(k)));
         J15 = -l2*cos(rot1(k)+th(k))*sin(rot2(k)+rot3(k));
         J21 = sin(th(k));
         J22 =
2*a*cos(th(k))+cos(rot1(k)+th(k))*(l1*cos(rot2(k))+l2*cos(rot2(k)+ro
t3(k)));
         J23 =
cos(rot1(k)+th(k))*(l1*cos(rot2(k))+l2*cos(rot2(k)+rot3(k)));
         J24 = -
sin(rot1(k)+th(k))*(l1*sin(rot2(k))+l2*sin(rot2(k)+rot3(k)));
         J25 = -l2*sin(rot1(k)+th(k))*sin(rot2(k)+rot3(k));

         J31 = 0;
         J32 = 0;
         J33 = 0;
         J34 = l1*cos(rot2(k))+l2*cos(rot2(k)+rot3(k));
         J35 = l2*cos(rot2(k)+rot3(k));

         J = [[J11 J12 J13 J14 J15];[J21 J22 J23 J24 J25];[J31 J32
J33 J34 J35]];

    % b) Matrices de peso para el cálculo de las acciones de
control, jugar con los valores.
         %Matriz de ganancia de los errores
```

```
          K = [[2    0     0];...
               [0    1     0];...
               [0    0     1]];

          %Matriz W pesa las acciones de control
          W = [[1    0     0    0     0];...
               [0    1     0    0     0];...
               [0    0     1    0     0];...
               [0    0     0    1     0];...
               [0    0     0    0     1]];

          D = [[1    0     0    0     0];...
               [0    2     0    0     0];...
               [0    0     0.7  0     0];...
               [0    0     0    2     0];...
               [0    0     0    0     2]];

    % c) Vector de ESPACIO NULO (Manipulabilidad)
          manirot1 =   -90.00*pi/180;
          manirot2 =  36.56*pi/180;
          manirot3 = -73.12*pi/180;

          Unulo = [0 0 (manirot1-rot1(k)) (manirot2-rot2(k))
(manirot3-rot3(k))]';

    % d) Acciones de control
          %Cálculo de la matriz pseudoinversa
          Jplus = inv(W)*J'*inv(J*inv(W)*J');

          %Cálculo del vector de espacio nulo
          EspNulo=(eye(5)-Jplus*J)*D*tanh(Unulo);

    % e) Ley de control completa,  solución = [u omega qpunto1
qpunto2]
          solucion = Jplus*([EEvx(k) EEvy(k) EEvz(k)]'+K*tanh([Xe(k)
Ye(k) Ze(k)]'))+EspNulo;

    % f) Separa las acciones de control, para el móvil necesito el
módulo de la velocidad lineal
          u(k) = solucion(1);               %En m/s
          w(k) = solucion(2);               %En rad/s
          qpunto1(k) = solucion(3);         %En rad/s
          qpunto2(k) = solucion(4);         %En rad/s
          qpunto3(k) = solucion(5);         %En rad/s

%3) APLICAR ACCIONES DE CONTROL AL MANIPULADOR MÓVIL
    % a) Modelo cinemático de la plataforma móvil
          x(k+1)=x(k)+u(k)*ts*cos(th(k));
          y(k+1)=y(k)+u(k)*ts*sin(th(k));
          th(k+1)=th(k)+w(k)*ts;


    % b) al modelo ideal del manipulador
          rot1(k+1)=rot1(k)+qpunto1(k)*ts;
          rot2(k+1)=rot2(k)+qpunto2(k)*ts;
          rot3(k+1)=rot3(k)+qpunto3(k)*ts;

         v1=x(k+1);
         v2=y(k+1);
         v3=th(k+1)*180/pi;
         v4=rot1(k+1)*180/pi;
```

```
        v5=rot2(k+1)*180/pi;
        v6=rot3(k+1)*180/pi;
%       calllib
('dll64Matlab','setValue',v1,v2,v3,v4,v5,v6,0,0,0,0);
 calllib('smClient','writeFloatValue','Memoria1',0,v1*10);
 calllib('smClient','writeFloatValue','Memoria1',1,v2*10);
 calllib('smClient','writeFloatValue','Memoria1',2,v3);
 calllib('smClient','writeFloatValue','Memoria1',3,v4);
 calllib('smClient','writeFloatValue','Memoria1',4,v5);
 calllib('smClient','writeFloatValue','Memoria1',5,v6);


 calllib('smClient','writeFloatValue','Memoria3',0,v1*10);
 calllib('smClient','writeFloatValue','Memoria3',1,v2*10);
 calllib('smClient','writeFloatValue','Memoria3',2,v3);
 calllib('smClient','writeFloatValue','Memoria3',3,v4);
 calllib('smClient','writeFloatValue','Memoria3',4,v5);
 calllib('smClient','writeFloatValue','Memoria3',5,v6);


 calllib('smClient','writeFloatValue','Memoria5',0,v1*10);
 calllib('smClient','writeFloatValue','Memoria5',1,v2*10);
 calllib('smClient','writeFloatValue','Memoria5',2,v3);
 calllib('smClient','writeFloatValue','Memoria5',3,v4);
 calllib('smClient','writeFloatValue','Memoria5',4,v5);
 calllib('smClient','writeFloatValue','Memoria5',5,v6);

%5) CALCULAR DONDE QUEDÓ EL EXTREMO DEL MANIPULADOR MÓVIL EE
        xEEreal(k+1) =
x(k+1)+a*cos(th(k+1))+cos(rot1(k+1)+th(k+1))*(l1*cos(rot2(k+1))+l2*c
os(rot2(k+1)+rot3(k+1)));
        yEEreal(k+1) =
y(k+1)+a*sin(th(k+1))+sin(rot1(k+1)+th(k+1))*(l1*cos(rot2(k+1))+l2*c
os(rot2(k+1)+rot3(k+1)));
        zEEreal(k+1) =
h+l1*sin(rot2(k+1))+l2*sin(rot2(k+1)+rot3(k+1));
             pause(0.01);
end
% calllib ('dll64Matlab','destroy')
load handel.mat;
sound(y)
```

Control Manual Con Falcon

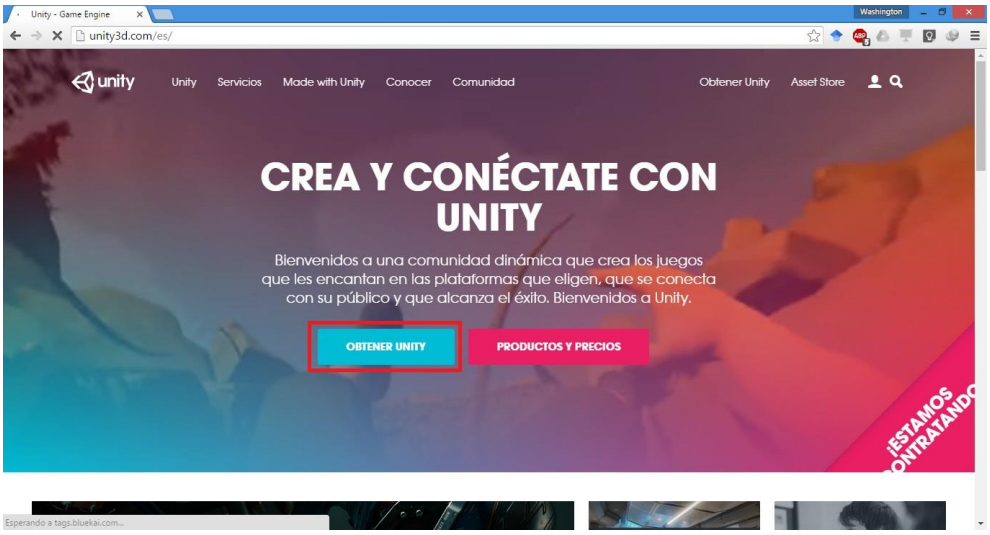
```
loadlibrary('./smClient.dll','./main.h');
calllib('smClient','abrirMemoria','Memoria0',2);
calllib('smClient','abrirMemoria','Memoria1',2);

        a  = 0.195; %distancia del centro de masa del móvil al marco
base del manipulador EN METROS
        h  = 0.375; %altura de la plataforma móvil + la altura de la
base del manipulador EN METROS
        l1 = 0.275; %longitud de la primera articulación EN METROS
        l2 = 0.25; %longitud de la segunda articulación EN METROS

        k=1;control=2;
%x(k)=0;y(k)=0;th(k)=0;q1(k)=0;q2(k)=0*pi/180;q3(k)=0;
[ok,x(k),y(k),th(k),q1(k),q2(k),q3(k)]=TcpLocal(obj1,com,0,0,0,0,0,0
,control);
[xr(1),yr(1),zr(1)] =
posEEmundo(x(k),y(k),th(k),q1(k),q2(k),q3(k),l1,l2,a,h);
Robot_Dimension;
%     fig=figure(2);
%     set(fig,'position',[10 60 980
600]);%set(fig,'DoubleBuffer','on');

% axes('position',[.05 .05 .65 .75]);
%         axis equal;         cameratoolbar
%         axis([-1 3.5 -2 2 0 0.9]);
%         H1=Robot_Plot_3D(x(1),y(1),th(1));
%
H2=Robot_Manipulador3D(x(1),y(1),h,th(1),q1(1),q2(1),q3(1),l1,l2);
hold on
%         H3=plot3(xr(1),yr(1),zr(1),'r');hold on

% control=TORQUE(1);
while control==2
    tic
    V = Falcon/6;
    V=[cos(th(k)+q1(k)) -sin(th(k)+q1(k)) 0; sin(th(k)+q1(k))
cos(th(k)+q1(k))  0;...
        0  0 1]*[-V(1);V(2);V(3)];
    if sum(abs(V)<.05)==3,V=V*0;end
    xd_p(k)=V(1);yd_p(k)=V(2);zd_p(k)=V(3);


[ok,x(k+1),y(k+1),th(k+1),q1(k+1),q2(k+1),q3(k+1)]=TcpLocal(obj1,com
,xd_p(k),yd_p(k),zd_p(k),0,...

0,0,control);

   if not(ok)
       ok

x(k+1)=x(k);y(k+1)=y(k);th(k+1)=th(k);q1(k+1)=q1(k);q2(k+1)=q2(k);q3
(k+1)=q3(k);

   end

 calllib('smClient','writeFloatValue','Memoria1',0,x(k+1)*10);
 calllib('smClient','writeFloatValue','Memoria1',1,y(k+1)*10);
 calllib('smClient','writeFloatValue','Memoria1',2,th(k+1)*10);
```

```
 calllib('smClient','writeFloatValue','Memoria1',3,q1(k+1)*100);
 calllib('smClient','writeFloatValue','Memoria1',4,q2(k+1)*100);
 calllib('smClient','writeFloatValue','Memoria1',5,q3(k+1)*100);
       %  3) APLICAR ACCIONES DE CONTROL AL MANIPULADOR MÓVIL
%     a) Modelo cinemático de la plataforma móvil
%          x(k+1)=x(k)+u(k)*ts*cos(th(k))-a*w(k)*ts*sin(th(k));
%          y(k+1)=y(k)+u(k)*ts*sin(th(k))+a*w(k)*ts*cos(th(k));
%          th(k+1)=th(k)+w(k)*ts;
% %
% %     b) al modelo ideal del manipulador
%          q1(k+1)=q1(k)+q1_p(k)*ts;
%          q2(k+1)=q2(k)+q2_p(k)*ts;
%          q3(k+1)=q3(k)+q3_p(k)*ts;

%5) CALCULAR DONDE QUEDÓ EL EXTREMO DEL MANIPULADOR MÓVIL EE
        xr(k+1) =
x(k+1)+a*cos(th(k+1))+cos(q1(k+1)+th(k+1))*(l1*cos(q2(k+1))+l2*cos(q
2(k+1)+q3(k+1)));
        yr(k+1) =
y(k+1)+a*sin(th(k+1))+sin(q1(k+1)+th(k+1))*(l1*cos(q2(k+1))+l2*cos(q
2(k+1)+q3(k+1)));
        zr(k+1) = h+l1*sin(q2(k+1))+l2*sin(q2(k+1)+q3(k+1));

%         drawnow
%         delete(H1);
%         delete(H2);

        %Gráfica del manipulador móvil
%          H1=Robot_Plot_3D(x(k+1),y(k+1),th(k+1));
%
H2=Robot_Manipulador3D(x(k+1),y(k+1),h,th(k+1),q1(k+1),q2(k+1),q3(k+
1),l1,l2);hold on;
%          H3=plot3(xr(1:k+1),yr(1:k+1),zr(1:k+1),'b');

    k=k+1;pause(.0001);
    toc
end
TcpLocal(obj1,com,0,0,0,0,0,0,4);
```

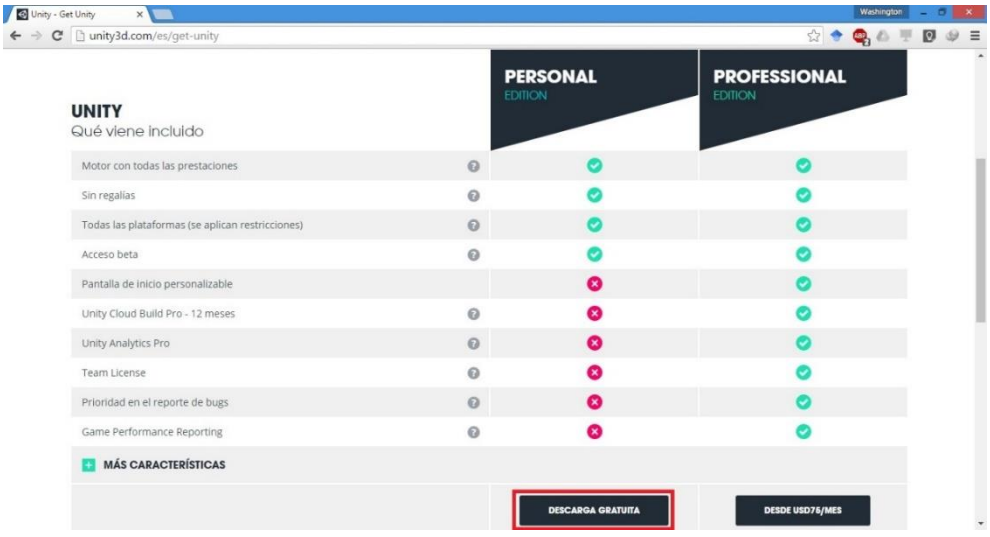
Anexo C: Manuales de Instalación

Instalador Unity3D

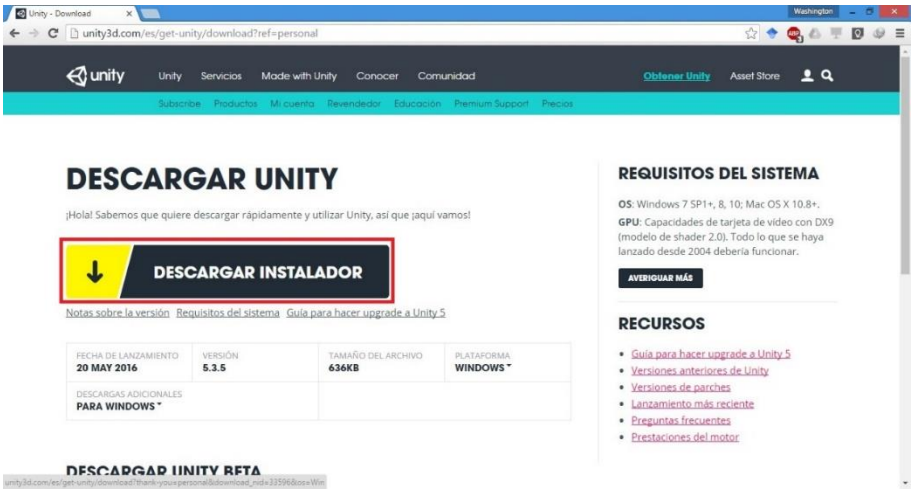
1. Ingresar a la dirección http://unity3d.com/es/ y hacer clic en Obtener Unity



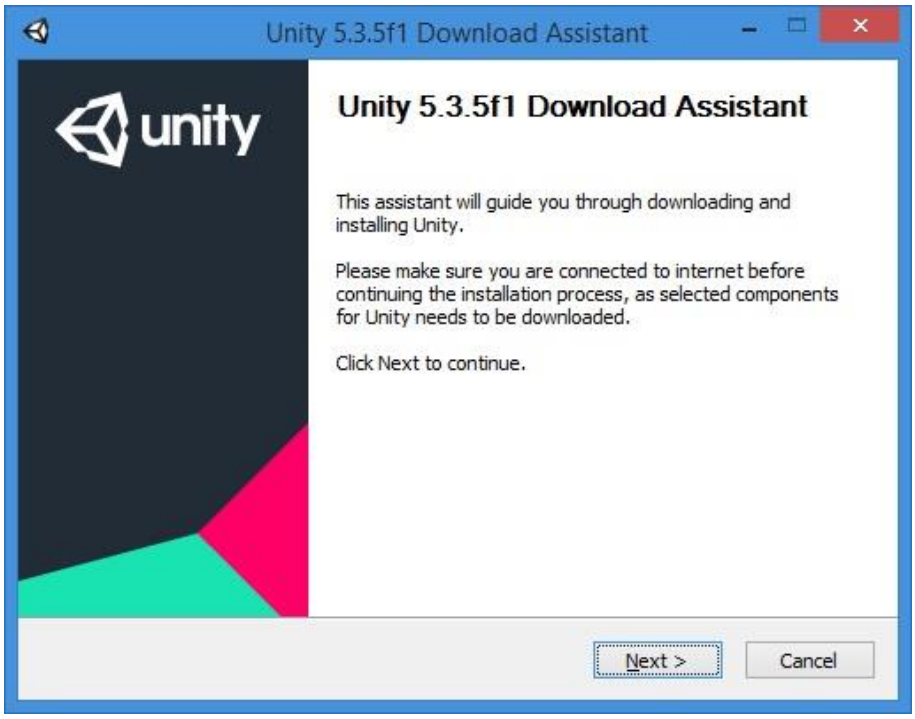
2. En la siguiente página, hacer clic en Descarga Gratuita



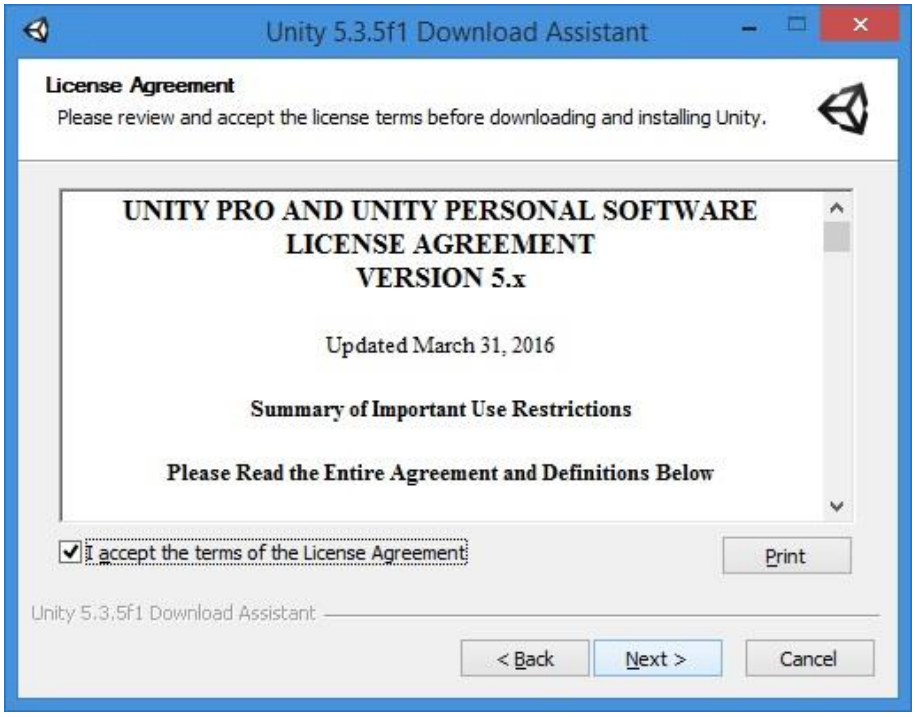
3. En la siguiente página, dar clic en Descargar Instalador, para descargar la última versión de Unity.



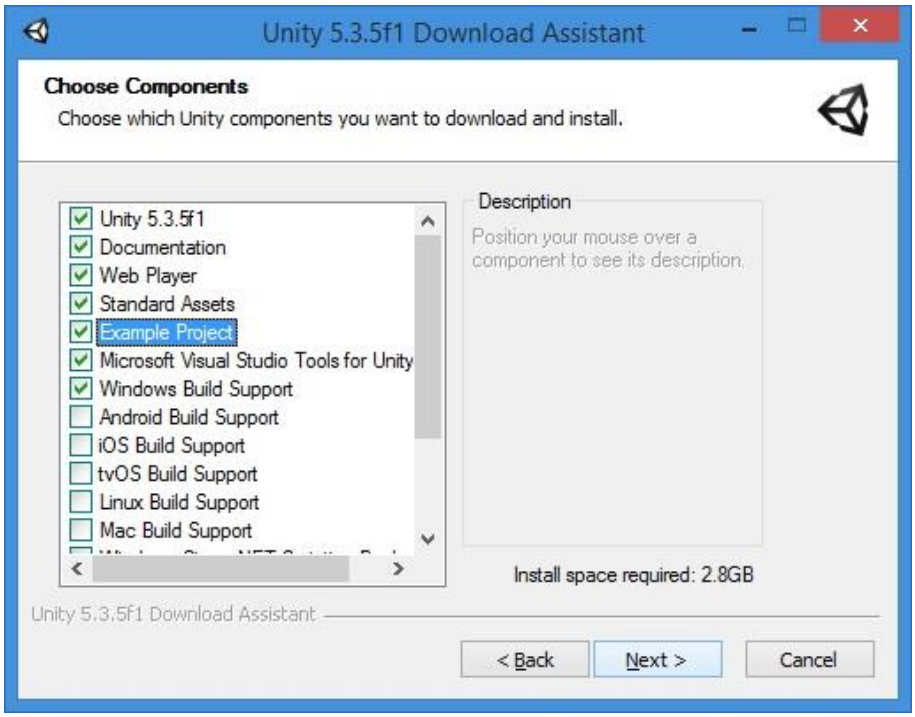
4. Abrir el instalador y dar clic en Next



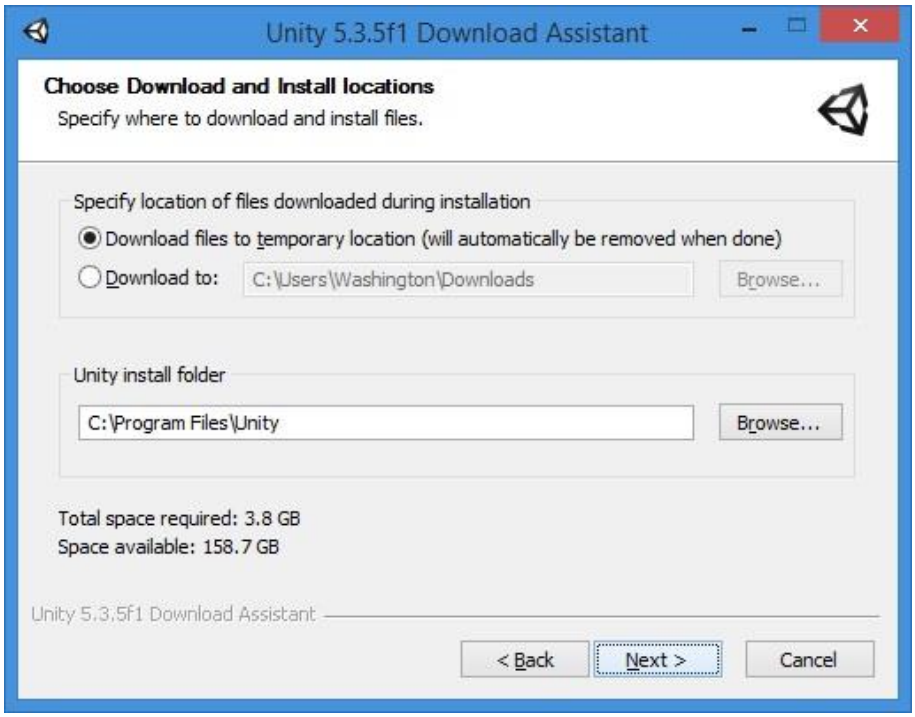
5. Aceptar los términos y el contrato de licencia y dar clic en Next



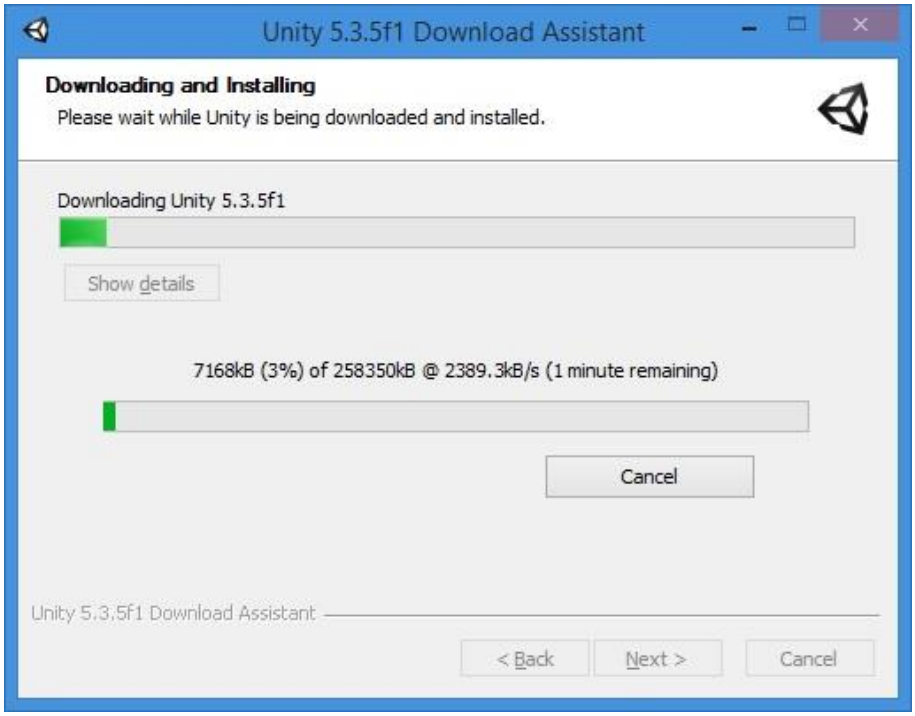
6. Seleccionar las opciones mostradas en la siguiente figura, y dar clic en Next.



7. Seleccionar la ruta de descarga y de instalación de Unity y dar clic en Next.



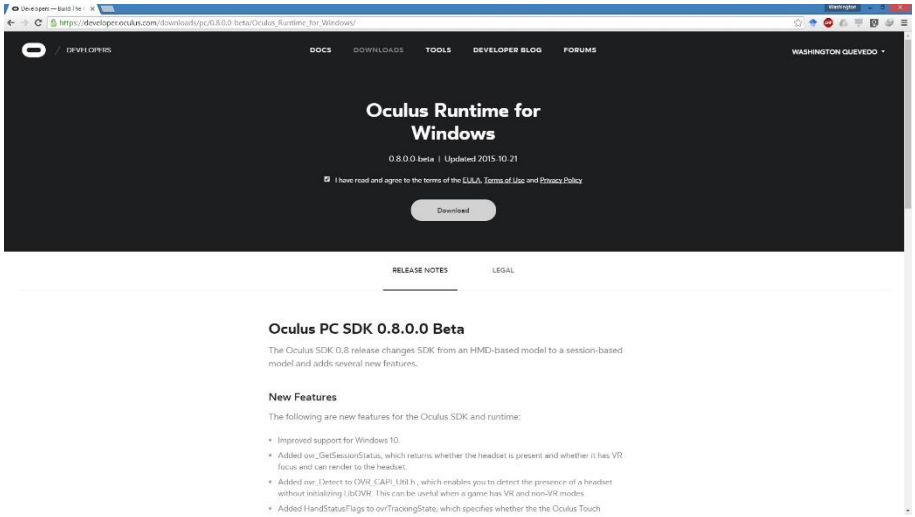
8. Esperar mientras se descargan los recursos necesarios para la instalación.



9. Clic en el botón Finish.

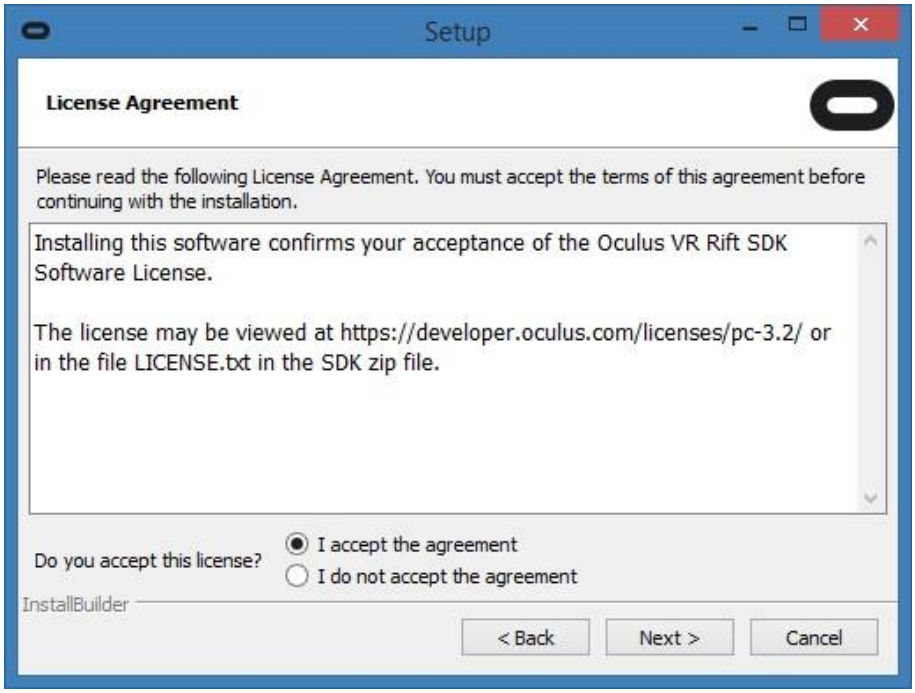
Instalador Oculus Rift Runtime

1. Ingresar a la página https://developer.oculus.com/downloads/ y descargar el Oculus Runtime for Windows V0.8.

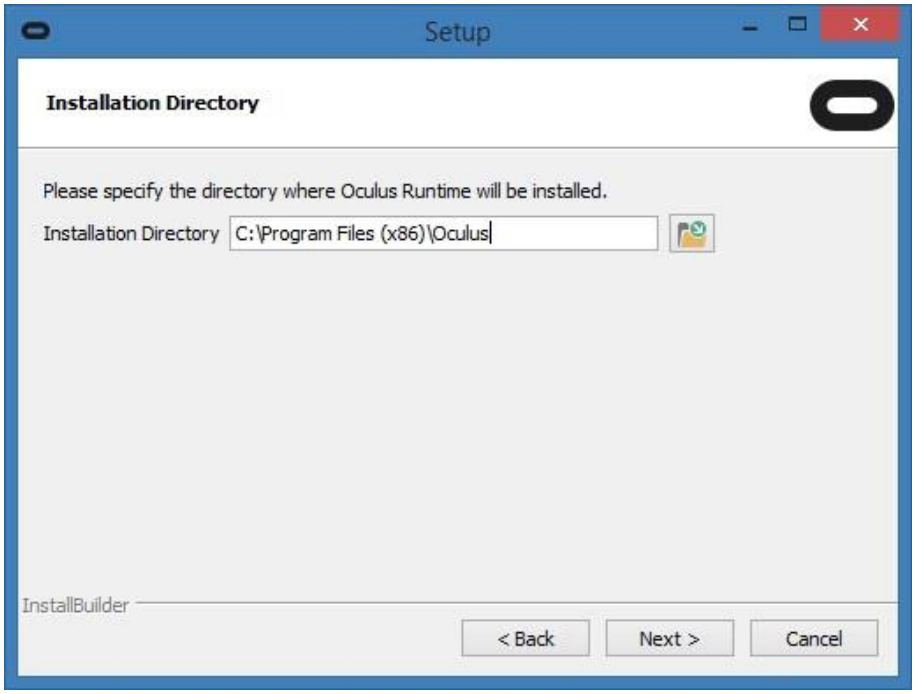


2. Instalar el runtime haciendo doble clic en el archivo descargado.

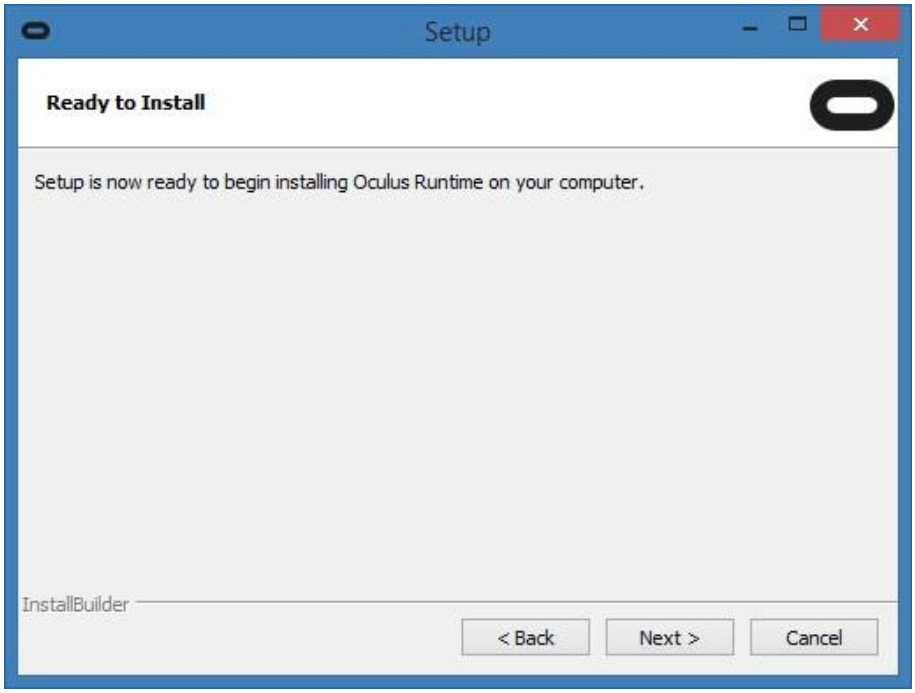
3. Aceptar los términos y condiciones de la licencia



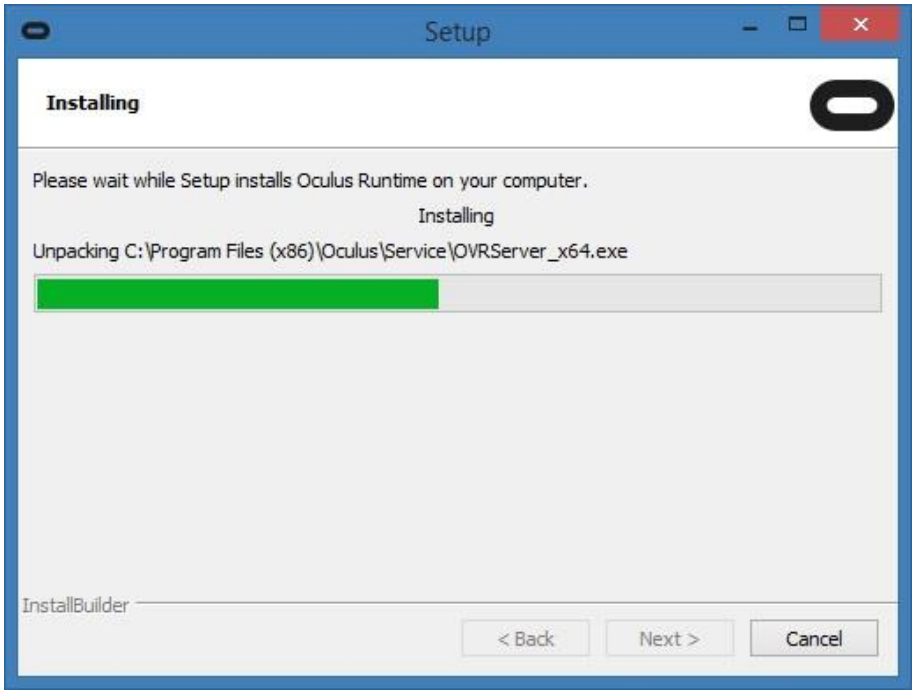
4. Seleccionar la ruta de instalación.



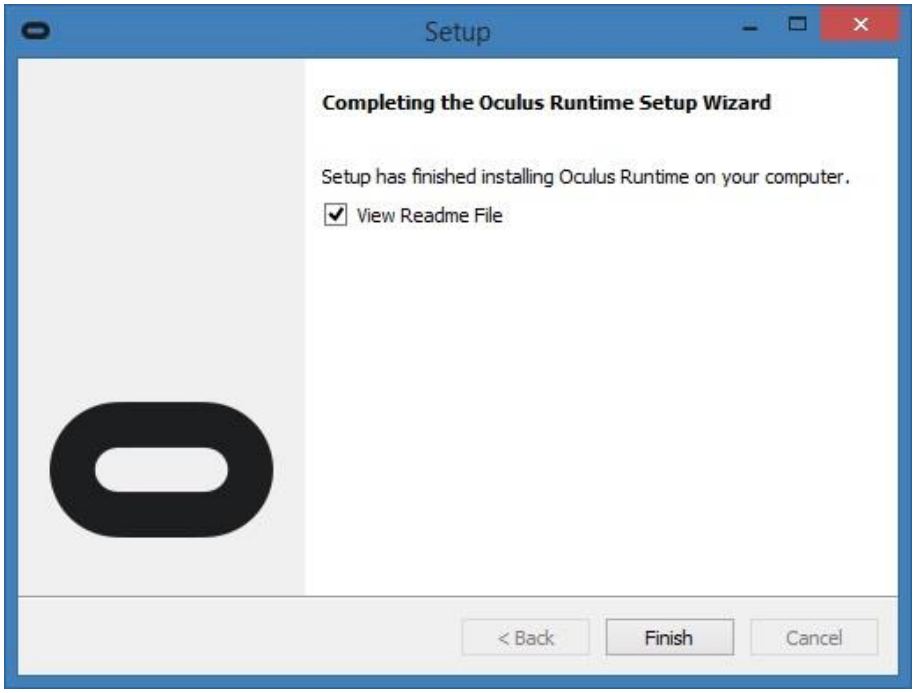
5. Clic en el botón siguiente para iniciar la transferencia de archivos.



6. Esperar hasta que se transfieran los archivos de instalación.

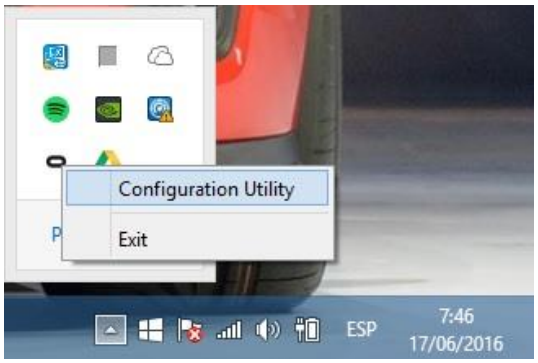


7. Clic en el botón siguiente para cerrar el instalador.



8. Conectar las gafas Oculus Rift y verificar su conexión en el área de notificaciones.

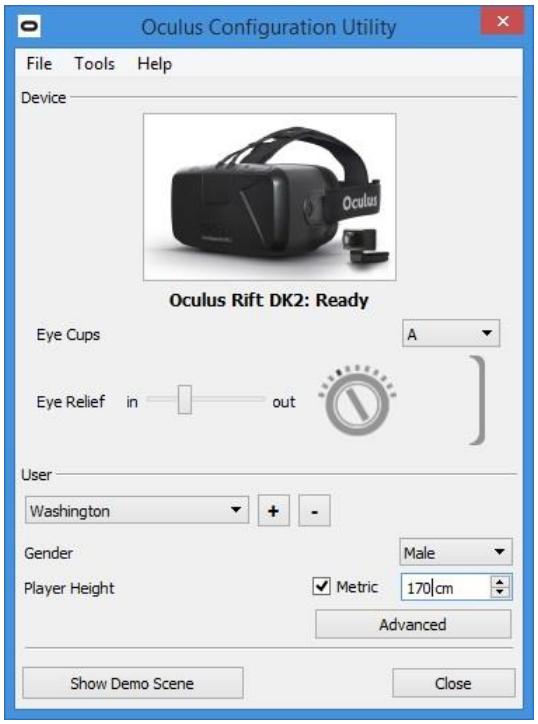
9. Para verificar el funcionamiento del runtime de Oculus, clic derecho en el icono de Oculus en el área de iconos ocultos y clic en la opción Configuration Utility.



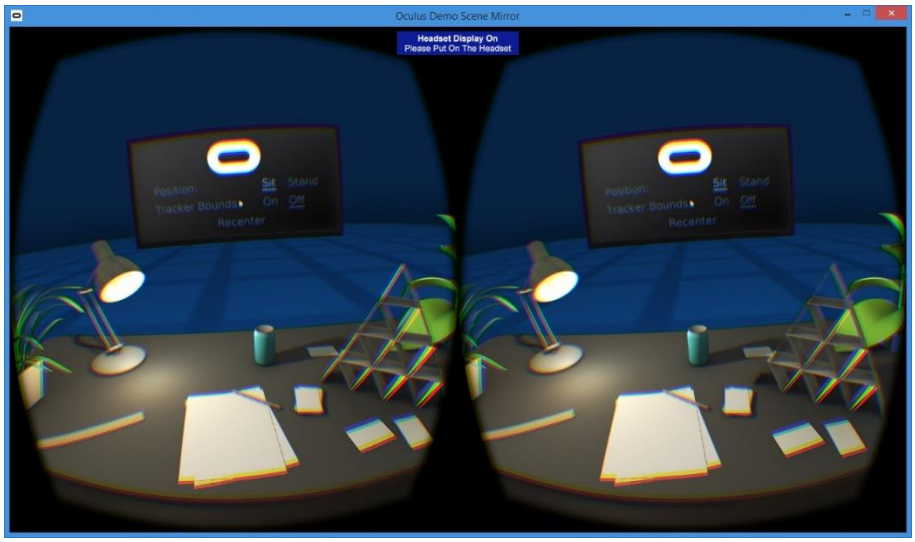
10. En la ventana siguiente, agregar un perfil de usuario

11. Colocar el nombre de perfil.

12. Seleccionar el género e ingresar la estatura en centímetros del usuario

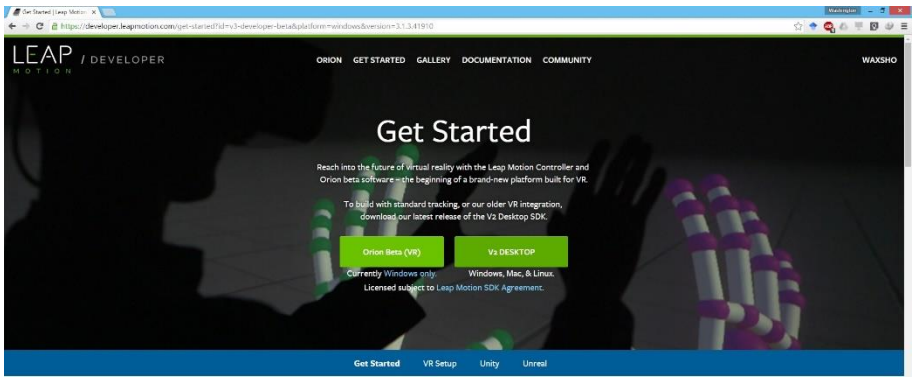


13. Clic en el botón Show Demo Scene para ejecutar el demo de Oculus

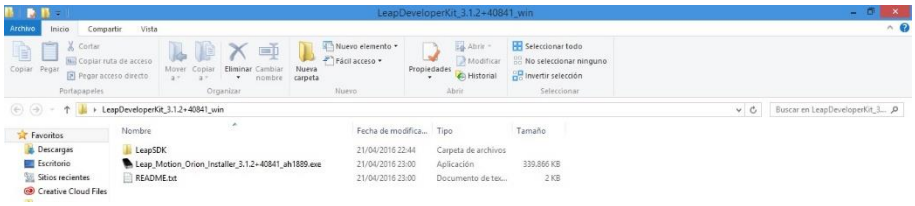


Instalador LeapMotion

1. Ingresar a la dirección https://developer.leapmotion.com/get-started y descargar el runtime Orion Beta.

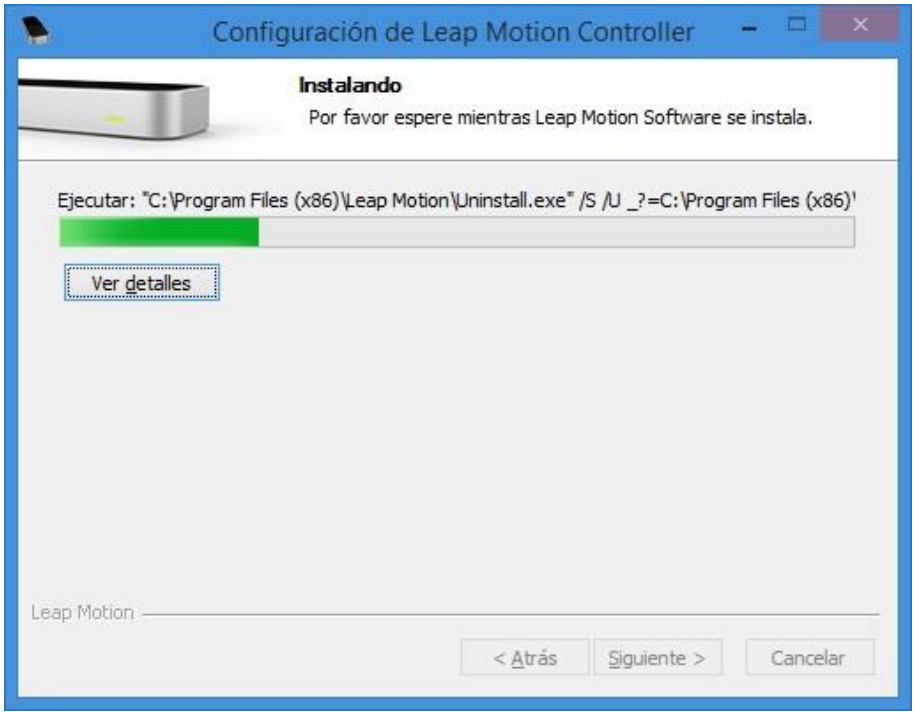


2. Doble clic en el ejecutable descargado.



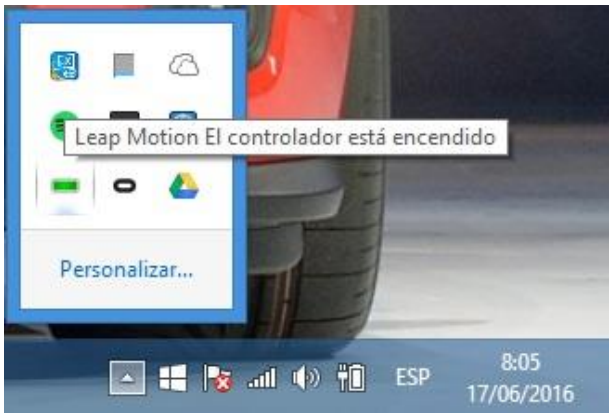
3. Clic en el botón siguiente en la ventana de bienvenida del instalador

4. Esperar hasta que se transfieran los archivos a la ruta de instalación

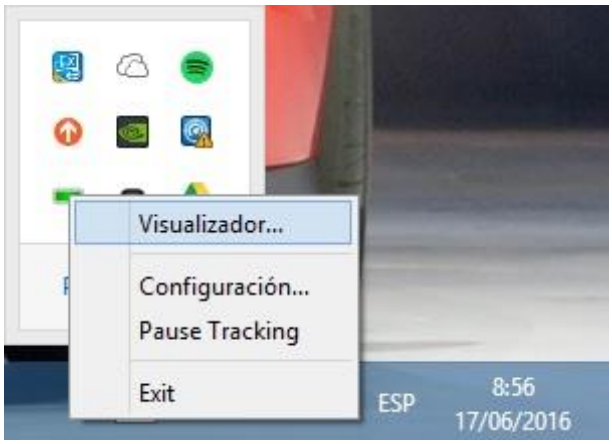


5. Clic en el botón terminar para cerrar el instalador

6. Conectar el dispositivo al ordenador y verificar que el controlador esté encendido.



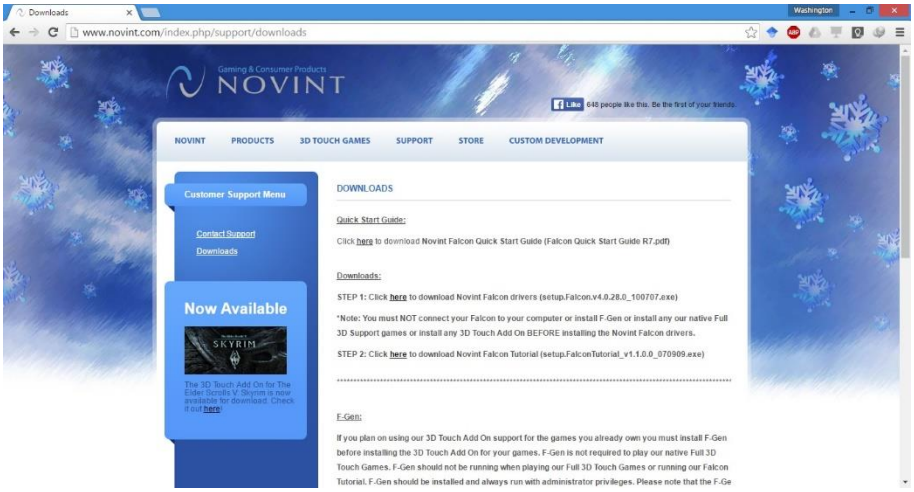
7. Clic derecho en el icono de LeapMotion y seleccionar la opción Visualizador.



8. Colocar las manos frente al dispositivo y verificar que sean detectadas por el software.

Instalador Novint Falcon

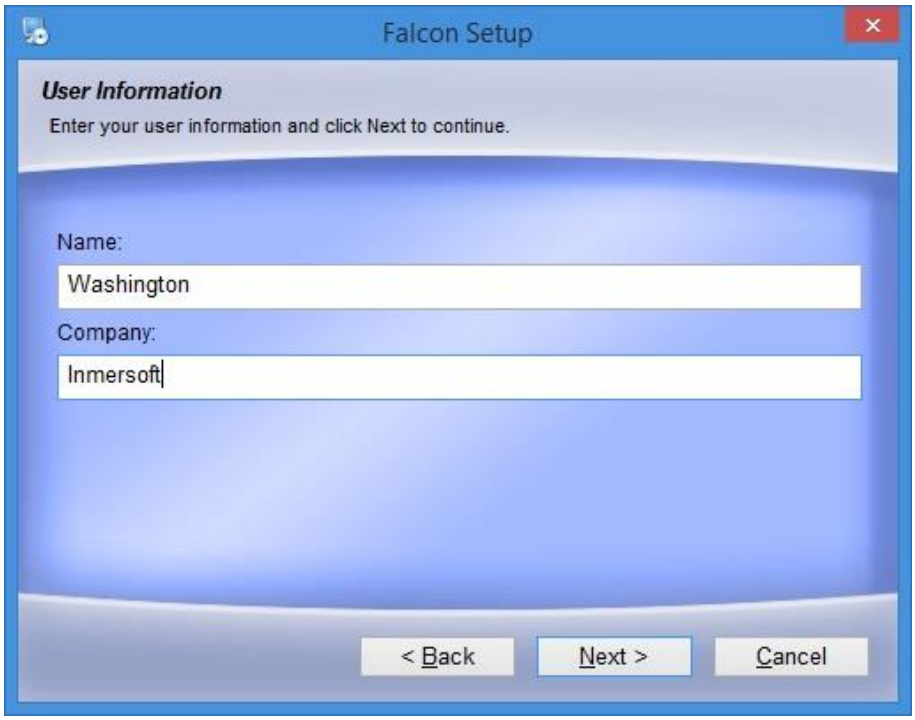
1. Ingresar en la página http://www.novint.com/index.php/support/downloads y descargar el instalador Falcon Setup.



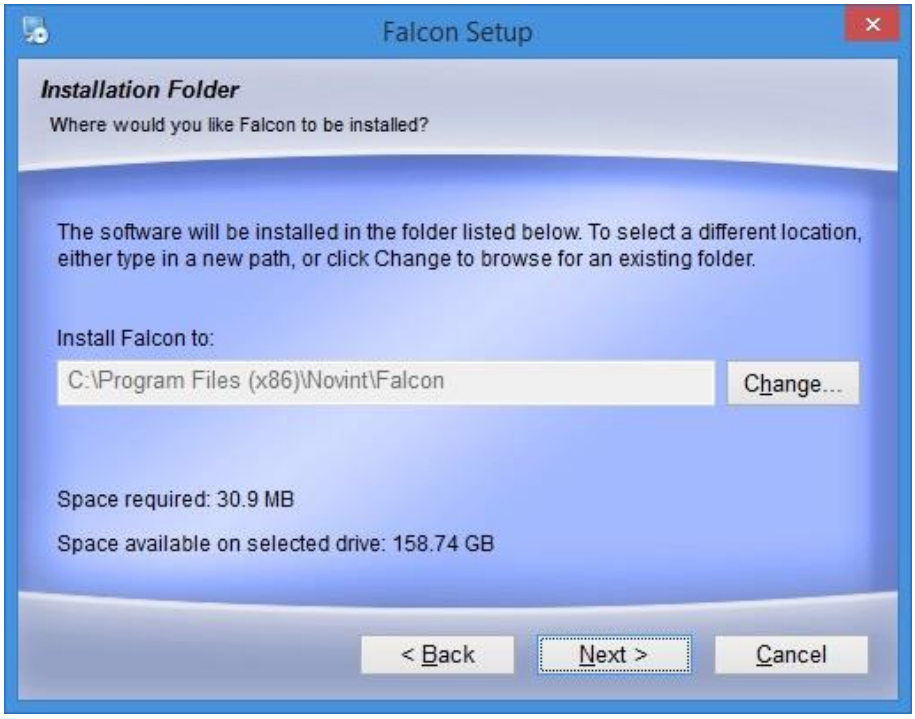
2. Doble clic en el archivo descargado para iniciar el instalador descargado.

3. Aceptar el acuerdo de licencia y clic en el botón siguiente

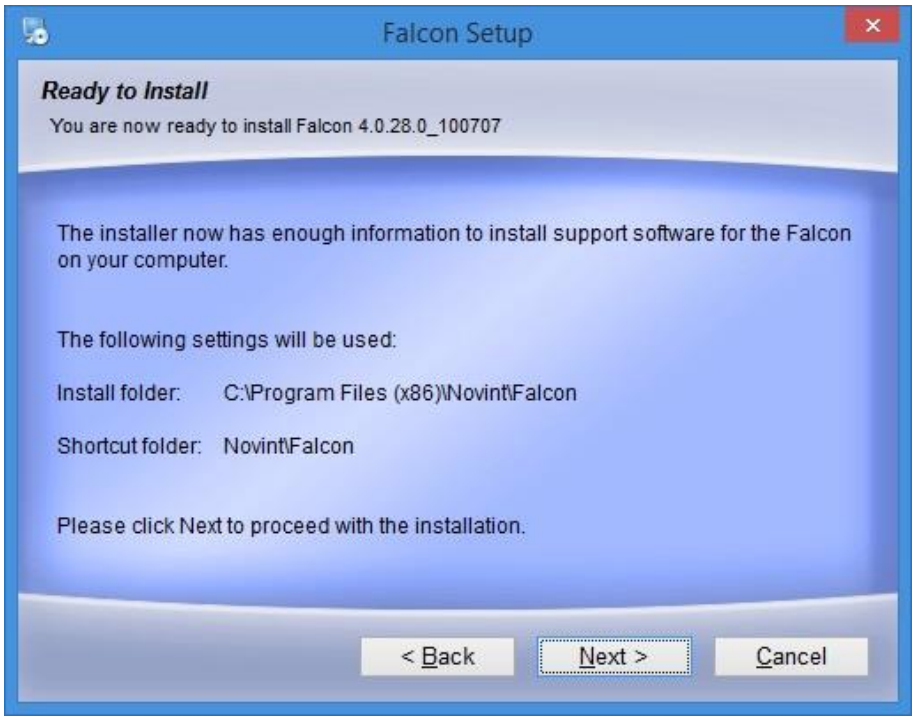
4. Agregar información en los campos requeridos.



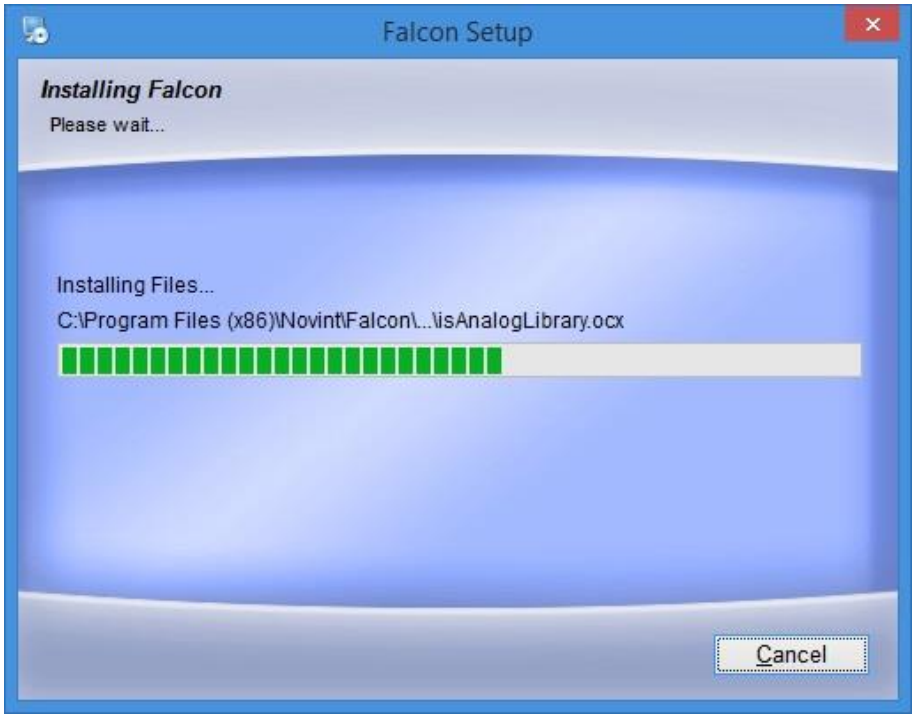
5. Seleccionar la ruta de instalación y clic en el botón siguiente.



6. Clic en el botón siguiente para iniciar la transferencia de archivos.



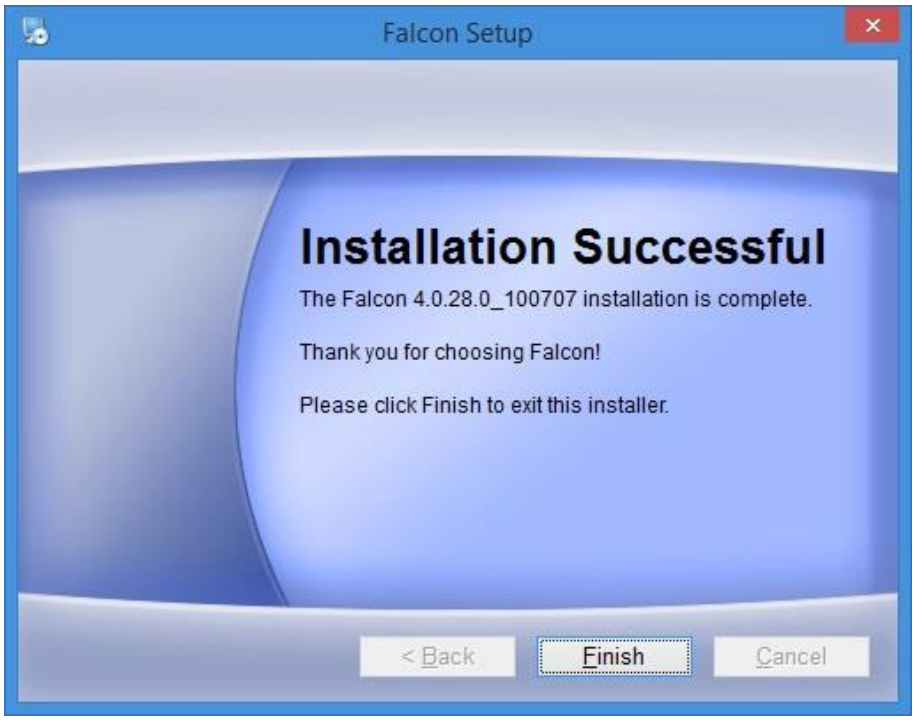
7. Esperar a que los archivos se instalen.



8. Clic en el botón siguiente para iniciar el instalador de los drivers.

9. Clic en finalizar para cerrar la ventana de instalación de los drivers

10. Clic en finalizar para cerrar el instalador inicial



UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por el señor:
Washington Xavier Quevedo Pérez

En la ciudad de Latacunga, a los 11 días del mes de agosto del 2016.

Ing. Víctor Hugo Andaluz Ortiz, Ph.D.
DIRECTOR DEL PROYECTO

Aprobado por:

Ing. Franklin Silva Monteros
DIRECTOR DE CARRERA

Dr. Rodrigo Vaca
SECRETARIO ACADÉMICO