DESARROLLO DE UN SISTEMA DE TELEPRESENCIA ROBÓTICA CON OCULUS RIFT

Universidad Carlos III de Madrid

TRABAJO DE FIN DE GRADO

Grado en Ingeniería en Tecnologías Industriales

Autor: Enrique Ruiz-Medrano García

Tutor: Raúl Pérula Martínez

Leganés, Septiembre 2014

Agradecimientos

Personalmente pienso que este apartado es un gran acierto dentro de un trabajo de esta magnitud, que deja atrás el simple informe de prácticas e introduce de lleno al autor en un proyecto más complejo y que marca el final de una etapa. Pienso que es un acierto ya que sería mezquino pensar que un trabajo que requiere tanto estudio y tiempo puede ser realizado por una única persona sin recibir ayuda de nadie. Pienso que es un gran acierto porque, aunque actuando de forma altruista, a todos nos gusta que nos reconozcan nuestros aportes. Y finalmente pienso que es un acierto porque desde pequeño me enseñaron que dar las gracias es lo que se debe hacer, así que allá voy.

Como no podía ser de otra forma, quisiera comenzar los agradecimientos por mi familia, que me ha apoyado constantemente a lo largo de todos mis estudios. Sin ellos estoy seguro que no habría logrado llegar hasta aquí.

Por otra parte también quiero agradecer a todos mis amigos, tanto de la infancia como los que hice en la universidad, el apoyo y la ayuda recibida por su parte a lo largo de toda la carrera, y especialmente en este último año con el proyecto por el que han mostrado mucho interés, aportando ideas y echando una mano en lo necesario.

Dentro del ámbito universitario, quiero agradecer a mi tutor, Raúl Pérula Martínez, el interés mostrado por el proyecto desde el primer día que le comenté la posibilidad de realizarlo como trabajo de fin de grado, haciendo todo lo posible para que saliese adelante y aportando los conocimientos y experiencia que a mí me faltaban. Así mismo también quiero agradecer al grupo de RPC de la asociación de robótica de la Universidad Carlos III de Madrid toda la ayuda y apoyo que me han prestado, especialmente a Pablo que me ha enseñado muchas cosas sin las cuales habría sido mucho más difícil terminar el proyecto.

Para terminar me gustaría lanzar un agradecimiento generalizado a todas esas personas que comparten sus conocimientos por internet de forma desinteresada, sin ellos habría sido prácticamente imposible sacar adelante este trabajo.

Resumen

Después de varias décadas de vida, la realidad virtual vuelve a estar de actualidad. Esto se debe a que, al contrario que los dispositivos que han existido hasta ahora que pecaban de ser bastante aparatosos y ofrecían poca calidad de imagen, en los últimos años y gracias a los nuevos avances tecnológicos ha surgido una nueva hornada de dispositivos asequibles para el consumidor medio y que ofrecen una buena calidad de visualización.

Estos nuevos dispositivos, encabezados por el Oculus Rift, tienen como objetivo la inmersión del usuario en mundos virtuales generados por ordenador, estando orientados por tanto al ocio electrónico. Sin embargo, en este proyecto se ha querido buscar un uso práctico de esta tecnología en el campo de la ingeniería, planteando pues un sistema de telepresencia robótica.

Los robots no industriales se suelen utilizar en la actualidad para realizar tareas en zonas de difícil acceso o peligrosas, siendo en la mayoría de los casos controlados por operadores humanos situados a distancia. Una de los hándicaps con los que se encuentra el operador a la hora de manejar el robot es la dificultad de apreciar correctamente el entorno, por lo que este proyecto pretende proporcionar un sistema de visión lo más parecido al ojo humano posible con la tecnología actual.

Abstract

After several decades of life, the virtual reality has become popular again. This is because, unlike the devices which have existed until now which where bulky and offered bad image quality, in the last years and thanks to the last technological advances has rise a new group of devices which are affordable for the average consumer and offer a good image quality.

This new devices, headed by Oculus Rift, has as the main objetive the user inmersión in computer generated virtual worlds, focused on the electronic entertainment. However, in this project we have tried to find a practical use of this technology in the field of the engineering, setting out a system of robotic telepresence.

The no industrial robots are usually used for make tasks in dangerous or inaccesible áreas, being in the most cases remote controlled by humans. One of the handicaps which have the human operator when is controlling a robot is that can't see environment clearly, so this project's objetive it's to provide a visión system as close as posible to the human vision with the current technology.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

En este capítulo se van a explicar los motivos por los que se decidió realizar este proyecto, así como la definición del problema real, el problema técnico y los objetivos marcados. Posteriormente se expondrá la forma en la que se va a estructurar la memoria.

1.1 Motivación y contexto

En los últimos años, y enfocados sobre todo al ocio electrónico, se han estado desarrollando diferentes dispositivos de realidad virtual. El mayor exponente de estos dispositivos son las Oculus Rift, un HMD (Head Mounted Display) capaz de mostrar imágenes estereoscópicas, realizando además un seguimiento de la posición de la cabeza del usuario para así variar el punto de vista del mismo en el mundo virtual, logrando así una mayor inmersión.

Por otra parte, en la actualidad los robots son cada vez más avanzados, incorporando nuevas tecnologías que los permite moverse con una mayor agilidad y acceder a lugares que hace unos años era impensable. Sin embargo, y aunque se está trabajando mucho para lograr una mayor autonomía de los robots actuales, lo cierto es que se está lejos de lograr una inteligencia artificial lo suficientemente compleja como para realizar ciertas tareas.

Siendo esto así, en la actualidad se suele recurrir a operadores humanos cuando es necesaria la actuación de un robot. Por poner algunos ejemplos, el cuerpo de policía utiliza robots para la desactivación de artefactos explosivos, hay determinados centros hospitalarios que cuentan con robots con la capacidad de operar siendo manejados a distancia por un cirujano, o en el escape radioactivo de Fukushima fueron utilizados robots para el sellado de grietas de la central nuclear. Todos estos robots se valen de cámaras y sensores para enviar información al operador, pero a menudo esta es insuficiente.

Uniendo estas dos premisas, la creación de nuevos dispositivos que permiten la inmersión en mundos virtuales y la necesidad de mejorar la información visual proveída por los robots teleoperados, nace la idea de la realización de un sistema que sea capaz de transmitir imágenes estereoscópicas a un operador, dotándole así de mayor información del entorno y permitiéndole variar el punto de vista de forma natural.

1.2 Definición del problema

1.2.1 Problema real

Como se ha explicado en la motivación y contexto del proyecto, el problema de los sistemas teleoperados actuales reside en las limitaciones de información que tiene el operador, que a menudo tiene que controlar el robot valiéndose únicamente de cámaras convencionales o de diversos sensores, que si bien a menudo ofrecen información valiosísima, se aleja de la forma natural que tenemos los seres humanos de interactuar con el entorno.

1.2.2 Problema técnico

Técnicamente y como se explicará más adelante, hasta ahora no existían sistemas de realidad virtual que ofreciesen resultados de calidad y que fuesen asequibles al mismo tiempo. Con la llegada de los nuevos dispositivos como las Oculus Rift esto ha cambiado y es posible tener sistemas baratos y de calidad. Estos dispositivos pueden ser usados tanto para el ocio electrónico como para visión estereoscópica de entornos reales, que es lo que se plantea en este proyecto para el sistema de telepresencia robótica.

1.3 Objetivos

El objetivo principal del proyecto será la realización de un sistema de telepresencia robótica totalmente funcional, demostrando que en base a la tecnología actual se puede lograr mejorar los sistemas de visión utilizados en los robots teleoperados a día de hoy.

Puesto que para la realización del proyecto sería necesario un robot preparado para ser teleoperado y con las cámaras adecuadas para proveer las imágenes en 3D, del que no se disponía en la universidad, se ha incluido en el objetivo la fabricación de un prototipo que haría las veces de cabeza del robot.

Siendo esto así, el objetivo inicial del proyecto se podría dividir en las siguientes tres tareas de menor envergadura:

- Desarrollo de una aplicación encargada de la comunicación entre el HMD y el robot teleoperado.

- Diseño de un soporte para la cámara encargada de proveer imágenes estereoscópicas. Este soporte sería lo equivalente a la cabeza del robot.
- Control de los servos encargados del movimiento de la cabeza del robot.

A lo largo de la realización del proyecto estos objetivos se han ido adaptando y han surgido otros nuevos, como se explicará detalladamente más adelante.

1.3 Estructura de la memoria

A continuación se detalla la forma en la que se ha estructurado la memoria:

- Capítulo 1: Introducción al proyecto, presentando los motivos que llevaron a la realización del mismo así como los objetivos marcados.
- Capítulo 2: Se hará un breve repaso de la historia de la realidad virtual, así como de las cámaras capaces de tomar imágenes estereoscópicas. También se hará un breve repaso sobre la telepresencia robótica.
- Capítulo 3: Se detallarán los factores dato y factores estratégicos que han afectado al proyecto.
- Capítulo 4: Descripción de los recursos utilizados para la realización del proyecto, desde recursos humanos hasta hardware y software.
- Capítulo 5: presupuesto del proyecto completo, incluyendo el precio de todos los componente sutilizados así como una estimación del coste de los recursos humanos.
- Capítulo 6: Explicación detallada de la metodología usada para la realización del proyecto así como las decisiones tomadas
- Capítulo 7: Descripción del funcionamiento del prototipo y exposición del resultado obtenido.
- Capítulo 8: Conclusiones y trabajos futuros que se pueden realizar en el entorno del proyecto.
- Bibliografía: Se incluyen las fuentes utilizadas para la realización del proyecto y la memoria.
- Apéndices: Manual de usuario del sistema desarrollado, detallando los pasos para su instalación y uso.
- Acrónimos: Lista de acrónimos utilizados a lo largo de la memoria.

Capítulo 2. Estado del Arte

En este capítulo se va a tratar el estado del arte de la realidad virtual, de la cámara Kinect y de la telepresencia robótica. Para ello se hará un repaso de cómo han ido avanzando estas tecnologías hasta nuestros días, detallando finalmente el estado actual de todas ellas y el uso que se las está dando.

2.1 Realidad virtual

2.1.1 Introducción

La realidad virtual se define como la generación por parte de un sistema informático de una realidad alternativa a al mundo real en algunos de sus aspectos o incluso en su totalidad. Dentro de la realidad virtual hay dos campos bien diferenciados:

- Realidad virtual inmersiva: es normalmente aquella que se basa en un mundo tridimensional que se puede manipular mediante periféricos como cascos o gafas (Head Mounted Display) además de guantes u otros dispositivos de captura de la posición corporal del usuario.
- Realidad virtual no inmersiva: es aquella que se puede manipular gracias a hardware común como puede ser un monitor, un teclado o un ratón. El ejemplo más claro de este tipo de realidad es Internet, donde el usuario puede interactuar con otros usuarios y ambientes que en realidad no existen.

En este proyecto el tipo de realidad virtual que nos interesa es el de realidad virtual inmersiva, y más en concreto aquellos periféricos denominados HMD que nos permiten visualizar entornos tridimensionales. A continuación se describe cronológicamente la historia de este tipo de realidad virtual así como de sus dispositivos más relevantes desde su invención hasta la actualidad.

2.1.2 Antecedentes

Invención de estereoscopio

La visión estereoscópica es la que posee el ser humano y se basa en la percepción de un entorno tridimensional a partir de dos imágenes bidimensionales ligeramente

diferentes entre ellas (cada correspondiente con la posición de un ojo). Siguiendo esta misma premisa, Sir Charles Wheatstone inventó en 1838 el estereoscopio, un artilugio que permitía ver imágenes estereoscópicas gracias a la reflexión de estas en unos espejos situados en el medio (ver figura 1).

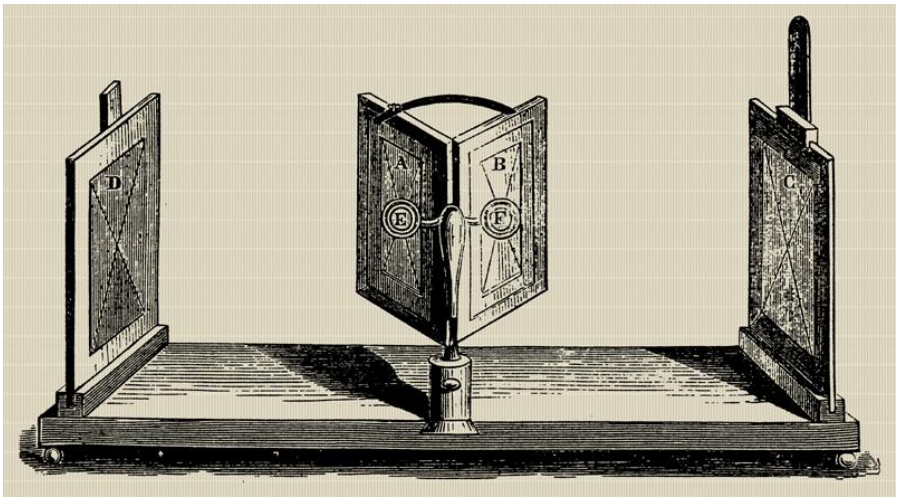

Figura 1: Estereoscópio de Sir Charles Wheatstone.

Posteriormente, en 1845, Sir. David Brewster presentó un estereoscopio de menores dimensiones que, al contrario que su que su predecesor que sólo servía para observar grandes láminas, permitía ver pequeñas imágenes. La mayor novedad en el artilugio de Brewster fue la inclusión de unas lentes correctivas que permitían al usuario enfocar imágenes cercanas. Gracias a este invento, en 1859 el poeta y médico Olver Wendell Holmes diseñó un estereoscopio más ligero con forma de gafas, que se podría decir que estableció las bases para el desarrollo de los HMDs actuales. [1] [2]

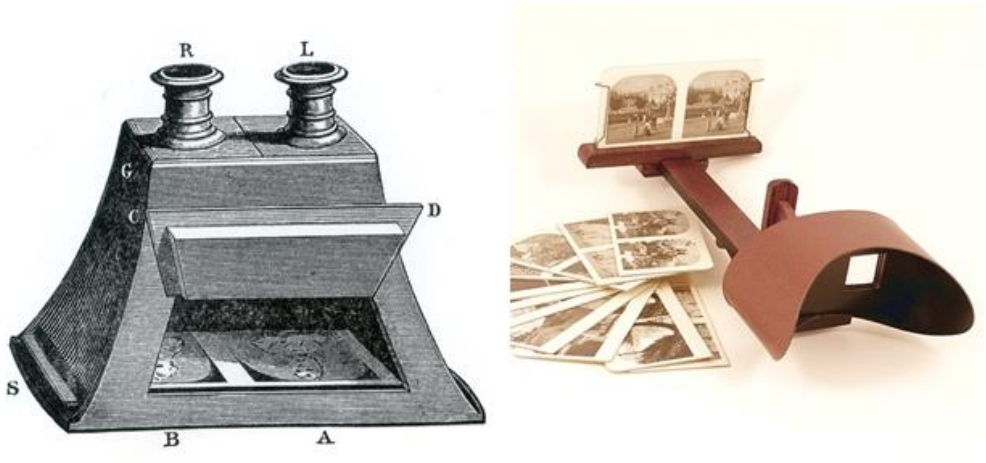

Figura 2: Izquierda - estereoscópio de Brewster. Derecha - estereoscópio de Olver Wendell Holmes.

Primer dispositivo de inmersión: Sensorama

En 1962 Morton Heilig desarrolló el primer dispositivo analógico de inmersión multisensorial, al que llamó Sensorama. Este dispositivo permitía al usuario visualizar una grabación estereoscópica en 3D, además de percibir otros estímulos por el resto de los sentidos, como el tacto gracias a vibraciones o el olfato gracias a olores liberados por la máquina. Entre las películas que se podían visualizar estaban un paseo en motocicleta o un viaje en todoterreno por el desierto. [2] [3]

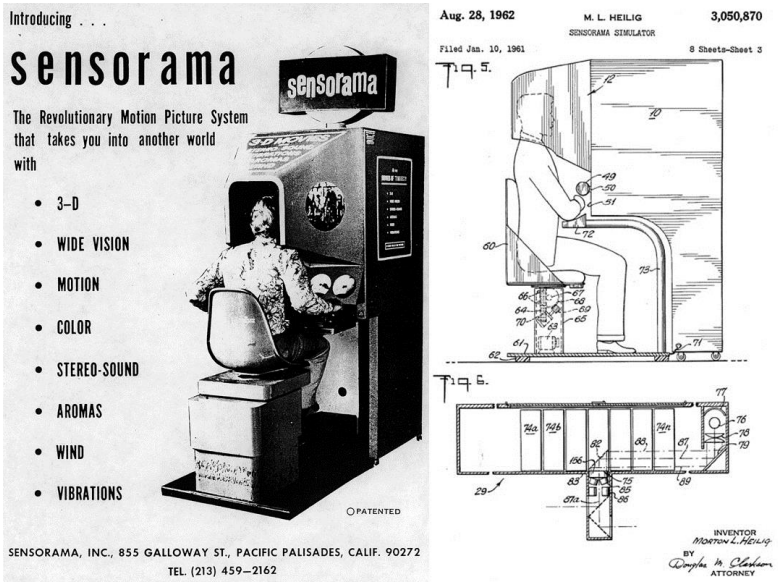

Figura 3: Anuncio de Sensorama y planos.

Nacimiento del concepto de Realidad Virtual y primer Head Mounted Display

En 1965 Ivan Sutherland publica un artículo en el que se habla por primera de realidad virtual, aunque no llega a referirse a ella por este término:

"La pantalla es una ventana a través de la cual uno ve un mundo virtual. El desafío es hacer que ese mundo se vea real, actúe real, suene real, se sienta real".

Posteriormente, en 1966, sería el propio Ivan Sutherland el que crearía el primer casco de realidad virtual, que contaba con un tubo de rayos catódicos para cada ojo, mostrando cada uno imágenes diferentes permitiendo al usuario visualizar imágenes estereoscópicas. Para el renderizado de las imágenes tridimensionales es necesario conocer la posición de la cabeza, por lo que el prototipo contaba con un sistema mecánico bastante aparatoso que colgaba del techo y que dio nombre al sistema, conocido como La Espada de Damocles (Figura 4). [3] [2] [4]

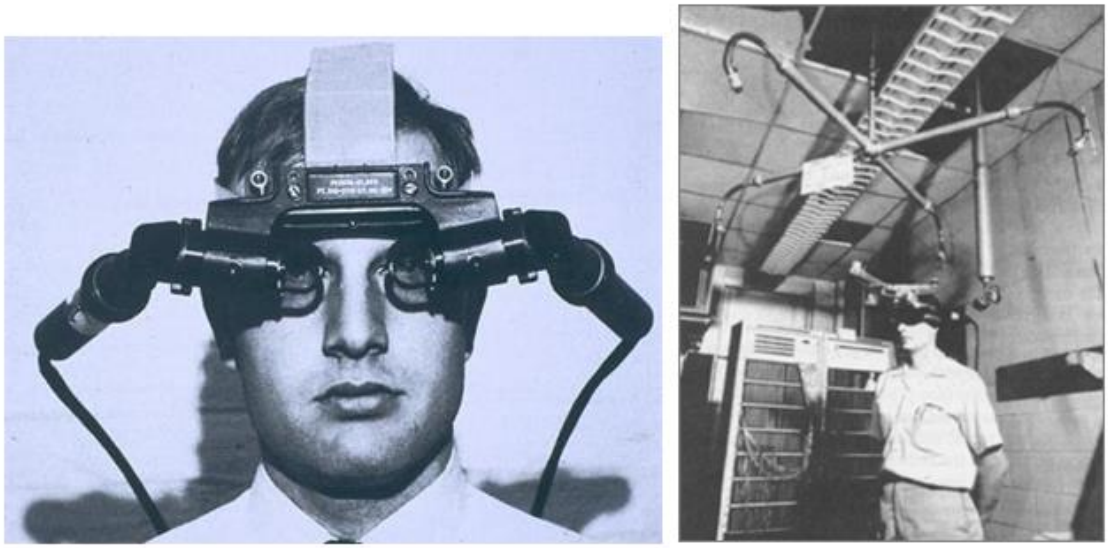

Figura 4: La Espada de Damocles.

El desarrollo de los HMD

A principios de la década de los 70, tanto el ejército del Reino Unido como la armada estadounidense empiezan a fabricar sus propios simuladores basados en displays gráficos, siendo el 1972 cuando General Electric desarrolla el primer simulador que actuaba en tiempo real. Dentro del ámbito militar, fue en 1979 cuando se empezó a utilizar cascos de realidad virtual para entrenar, además de otros periféricos como guantes capaces de capturar el movimiento de la mano del usuario, como el Sayre Glove (Figura 5), desarrollado por Tom Defanti y Daniel Sandin y que calculaba la posición de los dedos gracias a emisores y receptores de luz que llevaba en unos tubos de fibra óptica montados sobre cada dedo.

Figura 5: Sayre Glove.

En 1982 Thomas Furness desarrolló un simulador de vuelo que se valía de una cabina real de avión junto con un casco de realidad virtual (Figura 6). El casco, provisto de dos pantallas, mostraba representaciones simbólicas del mundo exterior, ya que por aquel entonces los gráficos por ordenador no estaban suficientemente desarrollados. Este simulador es considerado como el primer simulador de vuelo moderno, y a partir de él se desarrollaron el resto hasta llegar a los que hay hoy en día.

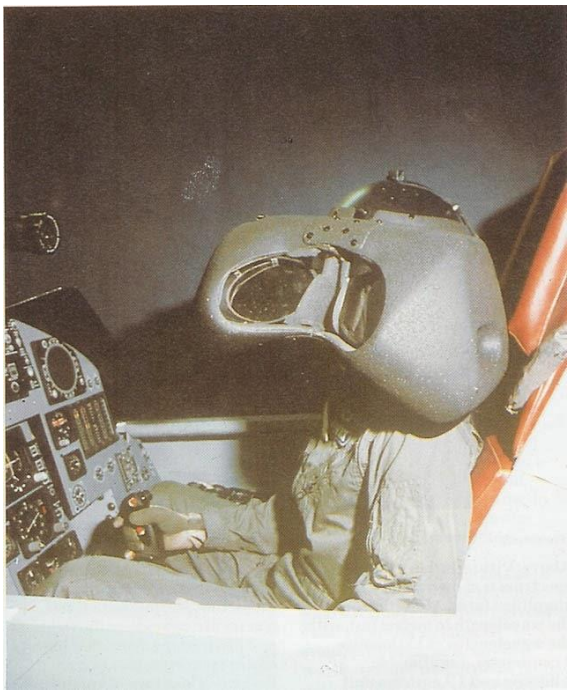

Figura 6: Simulador de vuelo de Thomas Furness.

En 1985 la NASA lanza el prototipo VIVED (Figura 7), un casco de realidad virtual equipado con diversos sensores que era capaz de monitorizar la posición de la cabeza. Así mismo, el sistema contaba también con sonido envolvente, reconocimiento de voz y un guante de reconocimiento de gestos. Entre las características técnicas a destacar de este sistema están dos pantallas LCD de 2,7 pulgadas que, junto a las lentes gran angular que poseía, ofrecían un ángulo de visión de 120 grados. Además, el sistema era relativamente barato en comparación con los HMDs anteriores, ya que la construcción de uno costaba aproximadamente 2.000 $ frente a los más de 15.000 $ de sus antecesores. [5]

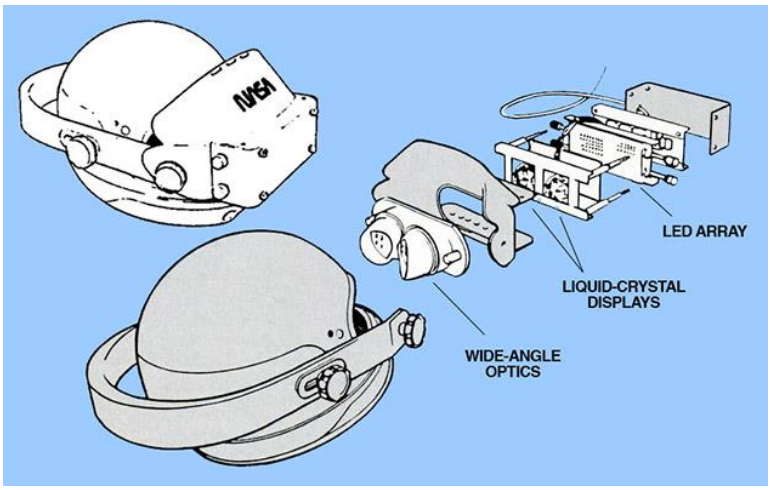

Figura 7: Dispositivo VIVED de la NASA.

Llegada de la realidad virtual al público general

Hasta la década de los 90 la realidad virtual había sido usada principalmente para el entrenamiento de tropas o personal especializado como pilotos de avión o astronautas, siendo esto así debido al elevado coste de producción de los dispositivos, lejos de las posibilidades del usuario medio. Sin embargo, a causa del avance de la tecnología y por tanto del abaratamiento de costes de los componentes, se empezó a pensar en la realidad virtual como una posibilidad de negocio.

En 1991 la empresa W Industries lanza el primer dispositivo de realidad virtual enfocado a salones recreativos, llamado Virtuality (Figura 8), que permitía a los usuarios jugar a diversos juegos en con visión estereoscópica gracias a un HMD con dos pantallas de 276x372 píxeles cada una, que además tenía una respuesta al movimiento de la cabeza del jugador de 50 ms. [6]

Figura 8: Virtuality de W Industries.

Debido al relativo éxito de la realidad virtual en las salas recreativas, Nintendo se decidió a lanzar el que sería el primer acercamiento de los HMDs de realidad virtual a la tecnología doméstica, la Virtual Boy. Esta consola, lanzada al mercado en 1995, ofrecía gráficos 3D gracias a imágenes estereoscópicas en rojo y negro (ver Figura 9), y tuvo un precio de salida de 180 $, lo que la posicionaba al alcance del gran público. Sin embargo, esta consola fue un fracaso comercial por diversos factores, como que no disponía de seguimiento del movimiento de la cabeza, era difícil de publicitar por los medios de comunicación convencionales, era incómoda de usar y producía dolor de cabeza después de jugar un periodo corto de tiempo. Siendo esto así y unido a la aparición de nuevas consolas capaces de renderizar gráficos en tres dimensiones, fue retirada del mercado al año siguiente de su aparición, no llegando a ser lanzada en Europa. [7] [8] [9]

Figura 9: Izquierda - Imagen de un juego de Virtual Boy. Derecha - Aspecto exterior de Virtual Boy.

Debido al escaso éxito de la Virtual Boy de Nintendo, el desarrollo de dispositivos de realidad virtual con objetivo el consumidor fue abandonado por la mayoría de las grandes empresas, cabiendo destacar el lanzamiento del HMD Philips Scuba en 1998, diseñado por los creadores del dispositivo Virtuality anteriormente mencionado, pero que de nuevo fue un fracaso comercial vendiendo únicamente 50.000 unidades.

Mejora de la tecnología

El mayor problema que presentaba la realidad virtual y los HMDs en la década de los 90 es que pecaban de tener poca resolución de imagen, además de que los gráficos generados por ordenador eran bastante básicos. Sin embargo, con la llegada del nuevo la

tecnología fue mejorando exponencialmente, ofreciendo pantallas de mejor calidad y ordenadores más potentes.

En 2006, la empresa japonesa Toshiba desarrolló el prototipo Toshiba Head Dome, que como se puede ver en la Figura 10 era un casco de 80 centímetro de diámetro y 3 kilogramos de peso. Este HMD ofrecía un campo de visión horizontal de 120 grados, en el que se mostraba una imagen gracias a un proyector de alta calidad iluminado por luces LED. Nunca llegó a salir a la venta y quedó simplemente como prototipo.

Figura 10: Toshiba Head Dome.

Pese a que la tecnología avanzaba las pantallas todavía eran demasiado grandes, como se puede ver en el prototipo de Toshiba. Con la llegada de los teléfonos móviles, se empezaron a desarrollar pantallas cada vez más pequeñas y ligeras, además de contar con una mayor resolución, lo que dio pie a la aparición de una nueva era de HMDs de realidad virtual.

En Agosto de 2012, Oculus VR, empresa fundada por Palmer Luckey, publica en Kickstarter la idea de desarrollar un HMD de realidad virtual de nueva generación, conocido como Oculus Rift. El objetivo de financiación marcado por Oculus VR para el desarrollo de los primeros kits de desarrollo fue de 250.000 $, ya que pese a que habían calculado que necesitarían 500.000 $ redujeron esa cantidad para fijar un objetivo más realista y que atrajese a más gente. Finalmente, tras terminar los 30 días de financiación que fija Kickstarter, Oculus VR había conseguido un apoyo económico de más de 2 millones de dólares, asegurando el futuro de la compañía. [9]

Figura 11: Oculus Rift DK1.

Las primeras unidades para desarrolladores, conocidas como DK1 (Developer Kit 1), empezaron a ser enviadas en Marzo de 2013. Estas unidades, que se vendían a un precio de 300 $, contaban con una pantalla de 7 pulgadas de 1280×800 píxeles (640×800 por ojo) de resolución y una frecuencia de refresco de 60 Hz. La latencia de muestreo del movimiento de la cabeza era de unos 50 ms gracias a un IMU (unidad de medición inercial) situado en el HMD, y ofrecía un ángulo de visión de 110 grados. [10] [11]

Esta unidad dejó de venderse en Marzo de 2014, siendo sustituido por el DK2, que mejora algunas de sus características como la pantalla que ofrece una resolución Full HD, además de incluir una cámara capaz de determinar la posición en el espacio de la cabeza del usuario. Esta segunda versión se está vendiendo actualmente por un precio de 350 $, y se espera que la versión final no difiera mucho de esta, aunque es posible que se aumente aún más la resolución de la pantalla hasta llegar a 4 K. La versión comercial tiene prevista su salida para 2015.

A raíz del éxito cosechado por la campaña de Kickstarter de Oculus Rift, acompañado además por unas buenas críticas por parte de los que habían probado los prototipos, otras empresas del sector tecnológico se interesaron por los HMDs de realidad virtual, como Samsung que ha anunciado que está trabajando en su propio sistema orientado a smartphones, o Sony, que en Marzo de 2014 presentó el que sería su casco de realidad virtual para la videoconsola PlayStation 4, Project Morpheus (Figura 12). Este

dispositivo es similar en características al DK2 de Oculus VR, contando también con una pantalla de resolución Full HD.

Figura 12: Project Morpheus de Sony.

2.2 Kinect

2.2.1 Descripción

El dispositivo Kinect fue presentado por Microsoft en 2009 como un nuevo periférico para su consola Xbox 360, que permitía al usuario controlar esta con gestos además de poder jugar a algunos videojuegos utilizando el cuerpo. Esta nueva corriente de videojuegos controlados por los movimientos del usuario fue iniciada con Nintendo y su consola Wii, y debido a su tremendo éxito Microsoft sacó al mercado Kinect para aprovechar este nuevo mercado.

Figura 13: Imagen promocional de Kinect como sistema de juego.

Kinect se basa en la detección de movimiento gracias a una cámara RGB y un sensor de profundidad, que acompañados de un software adecuado es capaz de identificar figuras humanas y seguir sus movimientos. Pese a que Kinect fue ideada para jugar lo cierto es que en este campo no ha gozado de demasiado éxito, teniendo un catálogo de videojuegos limitado. Sin embargo, a raíz de la liberación del SDK y al apoyo de diferentes desarrolladores, Kinect ha encontrado su nicho de mercado en investigadores que han encontrado en este dispositivo una alternativa barata a otros sensores que había en el mercado.

En Mayo de 2013, Microsoft presentó la segunda iteración de este dispositivo, que presentaba como principales mejoras respecto a su predecesor una mayor resolución por parte de las cámaras, mayor precisión del sensor de profundidad y un mayor ángulo de visión, así como mejoras en el software de reconocimiento que en esta nueva versión permite detectar hasta a 6 personas diferentes.

2.2.2 Alternativas

Asus Xtion

Figura 14: Asus Xtion pro live.

A mediados de 2012 la empresa Asus llegó a un acuerdo con los creadores de los sensores de la tecnología en la que se basa Kinect, Prime Sense, que les daba derecho a sacar al mercado su propio dispositivo. Así nació la gama de cámaras Xtion, que ofrecen una alternativa a la Kinect, y cuyas ventajas respecto a esta son unas dimensiones más contenidas, un peso significativamente menor, mejor calidad de la imagen RGB y la no necesidad de suministro adicional de energía. Por el contrario, no cuenta con motor para ajustar la inclinación y dado que goza de menos popularidad, hay menos información y los drivers no son tan completos.

2.3 Teleoperación y telepresencia

2.3.1 Introducción

La teleoperación se define como el conjunto de tecnologías que engloban el control de dispositivos a distancia por parte de un ser humano. Dentro de la teleoperación cabe destacar la rama de la telerobótica, que consiste en la operación a distancia de un robot, y que es uno de los pilares de este proyecto. [12]

Para la correcta teleoperación de un robot por parte de un ser humano, es necesario que este tenga información suficiente del entorno en el que está situado el robot. Una de las formas más complejas y completas de realizar esto es haciendo uso de la telepresencia, que es aquella situación en la que el operador tiene la sensación de encontrarse físicamente en el lugar en el que está trabajando el robot. Para el funcionamiento de un sistema de telepresencia es necesario que exista realimentación de información entre el entorno remoto y el operador.

Un sistema teleoperado cuenta con los siguientes elementos:

- Dispositivo teleoperado: es la máquina que trabaja en la zona remota y que está siendo controlada por un operador.
- Operador: es el encargado de controlar el dispositivo remoto. La intervención del operador puede ser continua, generalmente en sistemas teleoperados maestro-esclavo, o discontinua, en la que le operador da órdenes de alto nivel.
- Interfaz: dispositivos que permiten la interacción entre el sistema teleoperado y el operador. Dentro de la interfaz se incluyen los monitores de vídeo, los manipuladores maestro o cualquier dispositivo que permita al operador mandar o recibir información del sistema.
- Control y comunicación: conjunto de dispositivos encargados de procesar y transmitir la información entre el dispositivo remoto y el local.
- Sensores: se encargan de recoger información para ser utilizada por el interfaz y el control.

2.3.2 Antecedentes

La teleoperación se lleva usando desde la antigüedad, ya sea mediante el uso de útiles para alcanzar objetos lejanos o las simples pinzas de un herrero para poder manejar una pieza de metan al rojo. Sin embargo, no es hasta la irrupción de la industria nuclear cuando se empezó a desarrollar, ya que era necesario manipular material altamente radioactivo y nocivo para los seres humanos.

Uno de los pioneros en esta área fue Raymond Goertz, que alrededor de 1945 desarrolló el primer sistema maestro/esclavo (ver Figura 15) con el objetivo de manipular material radioactivo. Este primer dispositivo de teleoperación era completamente mecánico, y el operador veía la zona remota a través de varios cristales que le protegían de la radiación.

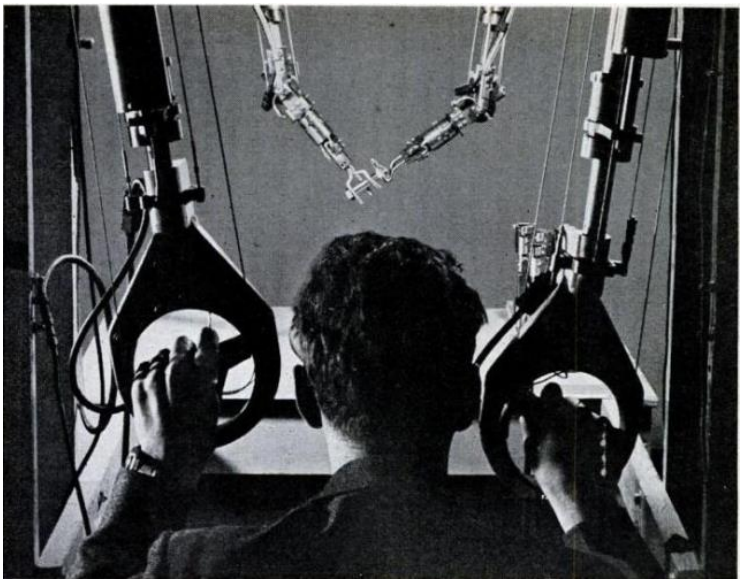

Figura 15: Sistema maestro/esclavo ideado por Raymond Goertz.

Estos sistemas mecánicos fueron reemplazados en 1954 por otros eléctricos, que funcionaban gracias a servo motores, además de que se incluyó un circuito cerrado de televisión, de forma que el operador podía situarse a una distancia mayor. A partir de este momento los sistemas de teleoperación se volvieron más complejos, cabiendo destacar el dispositivo General Electric Handyman que se puede ver en la Figura 16, desarrollado por Ralph Mosher, que era un dispositivo maestro/esclavo compuesto por dos brazos robóticos hidráulicos con diez grados libertad. Además se empezó gestar la idea de la telepresencia y el uso de HMD que dotasen a los operarios de mayor información, aunque

como se ha explicado en el apartado de realidad virtual habría que esperar hasta mediados de la década de los sesenta para que apareciese esta tecnología.

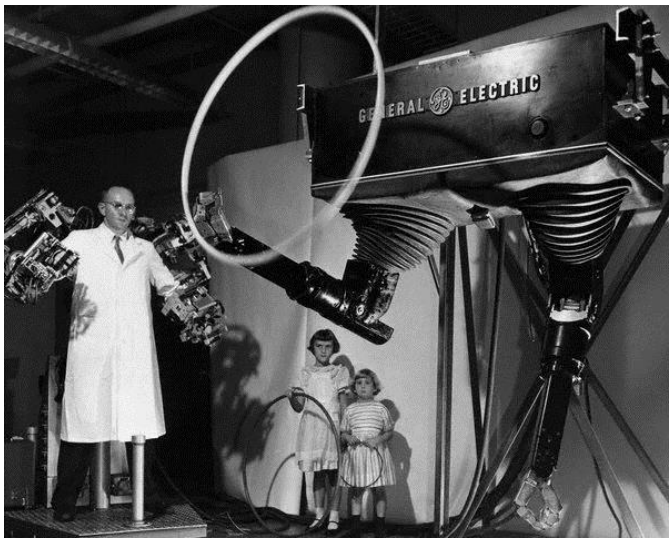

Figura 16: General Electric Handyman.

A comienzos de los 60 se empezaron a incluir dispositivos de teleoperación y cámaras en los submarinos de forma experimental, y por ejemplo el vehículo submarino CURV-I de la armada estadounidense fue el encargado de recuperar la bomba de hidrógeno caída accidentalmente cerca de Palomares en 1966, siendo este submarino totalmente teleoperado desde un barco. [13]

Figura 17: Ilustración de submarino CURV-I recuperando un torpedeo del lecho marino.

Con la llegada de la carrera espacial el desarrollo de la teleoperación cobró mucha importancia. Uno de los principales problemas que se presentó a la hora de llevar los sistemas utilizados hasta ese momento al espacio fue el retraso temporal que sufrían por las

largas distancias. Este retraso impedía que se usasen sistemas de lazo cerrado, ya que se volvían inestables al tener que esperar tanto tiempo a una respuesta. Tras realizar numerosos experimentos, se llegó a la conclusión de que la única forma de conseguir un sistema teleoperado estable era que este fuese de lazo abierto, teniendo que esperar la confirmación del operador después de cada acción. De esta forma fue teleoperado el Lunokhod 1, enviado a la Luna por Rusia a principios de los setenta y que recorrió una distancia de 10 kilómetros en 11 días. Este vehículo sufría un retraso en sus telecomunicaciones de unos tres segundos.

En las próximas décadas, la teleoperación siguió avanzando muy ligada a la exploración espacial, así como también se empezó a investigar en materia de cirugía a distancia. La NASA se propuso crear un robot que fuese capaz de ser teleoperado desde la Tierra para poder realizar cirugía en astronautas que se encontrasen en misiones espaciales de larga duración. En base a esta idea, en la década de los ochenta se llevaron a cabo varias operaciones quirúrgicas a distancia gracias a los robots Arthrobot y PUMA 560. En base a estas primeras operaciones, los robots se fueron mejorando dotándolos de mayor precisión, cámaras estereoscópicas para ofrecer visión 3D al cirujano, así como en algunos casos interfaces hápticas que permiten al usuario recibir estímulos mediante el sentido del tacto. Uno de los hitos más en este campo se llevó a cabo en 2001, cuando se realizó una operación de un paciente en Estrasburgo (Francia) desde Nueva York (Estados Unidos), siendo esta la primera operación transatlántica que se realizaba. [14]

2.3.3 Usos actuales

A continuación se detallan brevemente los usos que se le da a la teleoperación en la actualidad, que gracias al desarrollo de la informática y de las nuevas tecnologías está presente en muchos campos:

- Industria nuclear: como se ha explicado, desde el principio ha sido uno de los pilares de la teleoperación ya que se trabaja con materiales altamente radioactivos. Generalmente se usa la teleoperación para el mantenimiento de las centrales nucleares pero también en desastres nucleares ha sido de utilidad, como en el accidente de Fukushima de 2011, donde se usaron varios robots para inspeccionar la zona afectada.
- Industria militar: en los últimos años el uso de aviones no tripulados en zonas conflictivas se ha extendido enormemente, debido a que pueden realizar misiones

de reconocimiento o incluso de ataque sin poner en peligro vidas humanas. Estos aviones no tripulados suelen estar controlados por operadores que a través de numerosos sensores y cámaras tienen una visión clara de la zona que está sobrevolando la aeronave. Se puede ver unos de estos aviones en la Figura 18, donde además se pueden apreciar una serie de sensores así como armamento. Así mismo, también es común el uso de robots teleoperados para la desactivación de artefactos explosivos.

Figura 18: UAV Predator.

- Exploración submarina: el uso de robots en ámbitos submarinos se hace crucial, ya que a medida que aumenta la profundidad la presión se vuelve insoportable para el ser humano. Su uso es muy frecuente en la exploración de naufragios y en la recuperación de objetos ene l lecho marino, pero también es frecuente su uso en mantenimiento y construcción de estructuras submarinas, como oleoductos o plataformas petrolíferas.
- Medicina: como se ha explicado anteriormente, las operaciones quirúrgicas a distancia fue una de las primeras aplicaciones que se idearon para la teleoperación. Actualmente cabe destacar el robot Da Vinci (Figura 19), que permite la realización de operaciones mínimamente invasivas, estando capacitado para realizar cualquier tipo de operación que se pueda realizar por laparoscopia. En este sistema el cirujano está sentado en una silla teleoperando el robot que es el encargado de realizar la cirugía, dotando al cirujano de visión estereoscópica 3D y ofreciéndole mayor comodidad y precisión durante la operación. [14]

Figura 19: Sistema de cirugía Da Vinci.

- Exploración espacial: con el objetivo de no poner en peligro la vida de ningún ser humano es frecuente el uso de sistemas teleoperados en las misiones espaciales. El mayor problema dentro de este campo es la latencia a la hora de controlar los sistemas, ya que las distancias entre el dispositivo remoto y el local son enormes. Por ejemplo, las misiones enviadas a Marte cuentan con un retraso en sus comunicaciones de unos 6 minutos, por lo que se suele dotar a estos sistemas de cierta autonomía a la hora de realizar las tareas.

Capítulo 3. Restricciones

En este capítulo se van a tratar los factores limitativos existentes en el ámbito del diseño que han influido en la toma de decisiones. Estos factores se pueden clasificar en dos grupos: factores dato y factores estratégicos.

3.1 Factores dato

Los factores dato son aquellos que no pueden ser modificados a lo largo del trascurso del proyecto, como pueden ser el límite presupuestario, el tiempo asignado, o las limitaciones de recursos por parte de la universidad.

3.1.1 Presupuesto

Una de las limitaciones que presenta el proyecto es el presupuesto asignado para la realización del mismo. En la etapa de análisis del proyecto se estuvo estudiando la posibilidad de utilizar un HMD de fabricación casera debido a estas restricciones, o incluso a la adquisición de un Oculus Rift DK1 de forma externa a la universidad. Finalmente la universidad decidió adquirir un kit de desarrollo de Oculus Rift debido a las posibilidades que ofrece en el mundo de la investigación, como la propuesta en este proyecto.

Por otra parte, a lo largo del proyecto han ido surgiendo diversas alternativas para la realización del mismo que en ocasiones requerían desembolso económico por parte de la institución, pero finalmente hubo que usar otras alternativas que pasaban por usar los recursos disponibles.

3.1.2 Tiempo

La fecha límite establecida para la finalización del proyecto se fijó a finales de Septiembre de 2014, fecha en la que se debe entregar la memoria y posteriormente realizar la defensa del mismo. Pese a que existía la posibilidad de aplazar esta fecha a posteriores convocatorias, se decidió no hacerlo debido al deseo del autor de realizar un Máster académico en el curso 2014-2015.

3.1.3 Recursos

Como se detallaba anteriormente la limitación de recursos está estrechamente relacionada con la presupuestaria, ya que debido a esta hubo que plantear la realización del proyecto con los recursos ya disponibles en la universidad.

Esta limitación de recursos fue una de las razones por las que se usó Kinect en el proyecto, así como el esqueleto del robot Mini Maggie como soporte para la cámara.

3.2 Factores estratégicos

Los factores estratégicos son aquellos relacionados con las variables de diseño del proyecto. En función de las elecciones tomadas el desarrollo del proyecto puede verse alterado, así como el resultado final obtenido. Dentro de estos factores entrarían el hardware, el entorno de desarrollo o la interfaz de usuario elegida.

3.2.1 Hardware

Para la realización del proyecto el único dispositivo de hardware que podríamos clasificar de factor estratégico son las Oculus Rift, ya que el resto de hardware se vio condicionado por los factores dato anteriormente expuestos.

La elección de las Oculus Rift como HMD supuso que no hubo que preocuparse por la fabricación de un dispositivo de similares características, por lo que se pudo empezar directamente con el proyecto propiamente dicho. Además, el desarrollo del proyecto fue más sencillo al existir paquetes compatibles con este HMD, y el resultado del proyecto ha sido probablemente de mayor calidad.

3.2.2 Entorno de desarrollo

Para la edición de paquetes de ROS y para la creación de la interfaz de usuario se escogió el IDE Qt Creator. Esto fue así debido a que se tenían conocimientos previos de su uso y librerías.

Las consecuencias de esta elección fueron que no fue necesario aprender el uso de un nuevo IDE, que habría llevado más tiempo y seguramente el resultado final habría sido menos productivo.

3.2.3 Interfaz de usuario

Para la realización del proyecto no era estrictamente necesaria la creación de una interfaz gráfica de usuario, ya que se podían ejecutar todos los servicios de ROS por consola, sin embargo se decidió que era conveniente crear una para que orientar el sistema a un usuario final sin conocimientos técnicos específicos.

Esta elección supuso un incremento en el tiempo de desarrollo del proyecto, pero por el contrario el resultado final del mismo es más más completo.

Capítulo 4. Recursos

En este apartado se va a definir tanto el hardware como el software utilizado para la realización del proyecto, así como los recursos humanos disponibles para el mismo. El planteamiento inicial del proyecto pasaba por el uso de otros dispositivos y programas, pero en este apartado se detallan únicamente los usados en el desarrollo final del proyecto. Los cambios realizados se explicarán detalladamente en el capítulo de diseño.

4.1 Recursos Humanos

Como en todos los proyectos de esta índole, el único recurso humano disponible para la realización del mismo es el autor. Aun así, se ha contado con el asesoramiento continuo por parte del tutor del proyecto, Raúl Pérula Martínez. Adicionalmente, cabe destacar la ayuda obtenida por parte de fuentes externas al proyecto, como en el caso de algunos integrantes de ASROB, la Asociación de Robótica de la Universidad Carlos III de Madrid.

Si se trasladase el trabajo hecho por el autor del proyecto al mundo laboral actual este equivaldría al realizado por un analista, un diseñador y un programador, dependiendo de la fase del proyecto en la que se encontrase, como se explica a continuación:

- Analista: antes de la realización del proyecto y a comienzos del mismo hubo que hacer un estudio de las diferentes alternativas que había para el desarrollo del sistema, teniendo en cuenta las restricciones con las que se contaban.

- Diseñador: una vez planteado el proyecto, hubo que realizar el diseño de los sistemas que se iban a implementar y sus interconexiones, así como de la interfaz gráfica de usuario del programa final.

- Programador: finalmente hubo que desarrollar el sistema previamente diseñado, lo que consistió fundamentalmente en la programación de los diversos paquetes de ROS así como de la interfaz gráfica gracias a Qt.

4.2 Hardware

4.2.1 Oculus Rift DK1

Las Oculus Rift DK1 fueron el primer prototipo de gafas de realidad virtual lanzado por la empresa Oculus VR. Como se ha explicado en el capítulo de estado del arte, fueron desarrolladas gracias a una exitosa campaña de Kickstarter, y esta primera iteración de las mismas (más tarde saldría el DK2) era una de las recompensas que se daba a los que contribuyeron en la financiación con 300 $ o más. Así mismo, posteriormente se pusieron a la venta en su página web, teniendo como público objetivo los desarrolladores ya que de momento es un producto inacabado.

Figura 20: Izquierda - Oculus Rift DK1. Derecha - Oculus Rift DK2.

A grandes rasgos, el DK1 ofrece al usuario un HMD de realidad virtual con la capacidad de mostrar imágenes estereoscópicas además de llevar un seguimiento de la posición de la cabeza. El modo de funcionamiento de este sistema, pensado inicialmente para el mundo del ocio electrónico, se basa en la renderización de dos puntos de vista dentro del mundo virtual, uno para cada ojo, y en la variación de este dependiendo de la posición de la cabeza del usuario.

En la siguiente imagen se pueden apreciar los componentes del DK1 desmontados, y a continuación se explicará detalladamente la función de cada uno de ellos [15]:

Figura 21: Componentes de Oculus Rift DK1.

1.- Carcasa

Su función es la de mantener todos los componentes en su lugar, así como de proporcionar una sujeción cómoda para el usuario. En los laterales lleva dos tornillos ajustables con la ayuda de una moneda como se aprecia en la Figura 22, cuya función es la de acercar o alejar las lentes de los ojos. La sujeción de las lentes está diseñada de tal manera que cada ojo sólo pueda ver la mitad de la pantalla en la que se mostrará su imagen.

Figura 22: Ajuste de distancia de las lentes.

2.- Lentes

El DK1 viene con tres juegos de lentes, denominadas como A, B y C (ver Figura 23). Las lentes A son las que vienen montadas por defecto y son las que ofrecen un mayor ángulo de visión. Las lentes B y C están diseñadas para usuarios con problemas de miopía, aunque existe la posibilidad de utilizar gafas no demasiado grandes si se aleja la pantalla con el ajuste lateral de la carcasa. La utilización de lentes es necesaria ya que la pantalla se encuentra muy cerca de los ojos y sin ellas es imposible para el ojo humano enfocar la imagen correctamente. Las características físicas de estas lentes hacen que el punto de enfoque se sitúe en el infinito, por lo que para el usuario la sensación es como si mirase por una ventana.

Figura 23: Juego de lentes incluidas en el Oculus Rift DK1.

3.- Pantalla

Es una pantalla LCD de 7 pulgadas Innolux HJ070IA-02D. Ofrece una resolución de 1280x720 píxeles, aunque esta resolución hay que dividirla entre dos ya que cada ojo ve la mitad de la pantalla, por lo que la resolución real es de 640x720 píxeles. La forma en la que se muestran las imágenes es deformada con forma de cojín, como se puede apreciar en la Figura 24. Esto es así ya que al tener que incluir lentes de aumento para que sea posible enfocar la imagen, es necesario aplicar posteriormente una deformación a la imagen para que al visualizarla a través de las lentes se vea correctamente. La frecuencia de refresco de esta pantalla es de 60 Hz.

Figura 24: Captura de pantalla de una aplicación para Oculus Rift.

4.- Unidad de medición inercial o IMU

Es el encargado de calcular la posición de la cabeza del usuario. Para ello se vale del método conocido como fusión sensorial, que gracias a la información obtenida por tres tipos de sensores diferentes (giroscopio, magnetómetro y acelerómetro) es capaz de determinar la dirección exacta a la que está mirando el usuario, facilitando esta información al desarrollador en cuaternios. La tecnología utilizada en este chip ha sido desarrollada específicamente por el equipo de Oculus VR para sus gafas, y es que se estableció como una de las prioridades en la fabricación de las Oculus Rift que estas tuviesen la mínima latencia posible. Los IMUs que había en el mercado y que no encarecían demasiado el producto contaban con una tasa de muestreo de 120 Hz, mientras que el IMU desarrollado por Oculus VR cuenta con una frecuencia de 1000 Hz, que disminuye significativamente la latencia.

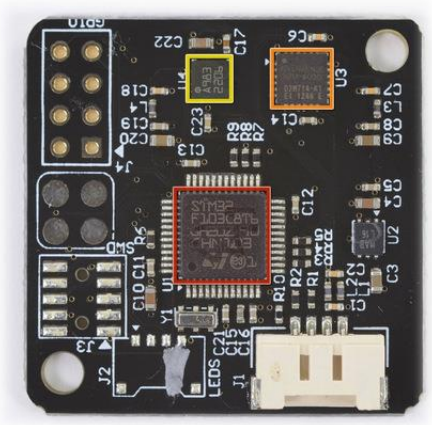

Figura 25: IMU de Oculus Rift DK1.

5.- Placa controladora de vídeo.

Se encarga de la comunicación entre el PC y el HMD, del control de la pantalla LCD y de alimentar todos los sistemas del mismo. Esta placa cuenta con botones de encendido, ajuste del brillo y del contraste de la pantalla y proporciona al usuario distintos conectores de entrada de video, siendo estos DVI y HDMI. También cuenta con un puerto mini USB por el que se conecta con el PC para enviarle los datos medidos por el IMU. Por último, tiene una entrada de corriente continua para alimentar a todos los dispositivos electrónicos con lo que cuenta.

Figura 26: Conexiones de Oculus Rift DK1.

En la siguiente tabla se muestran las características técnicas tanto del DK1 como del DK2, aunque para el proyecto se ha utilizado el DK1 ya que la nueva y más actual versión de las Oculus no estaba disponible cuando se comenzó la realización del mismo. Después de la tabla se explica la importancia de cada uno de estos apartados a la hora de mostrar mundos virtuales y favorecer la inmersión. [11]

	DK1	**DK2**
Resolución de la pantalla	1280 x 720 píxeles	1920 x 1080 píxeles
Diseño de los píxeles	RGB	Pentile
OLED	NO	SI
Tamaño de pantalla	7”	5.7”
Fabricante de la pantalla y modelo	Innolux HJ070IA-02D 7″ LCD	Samsung Galaxy Note 3
Latencia	50ms – 60ms	20ms – 40ms
Pantalla de baja persistencia	NO	SI
Frecuencia de refresco	60Hz	75Hz
Muestreo de orientación	SI	SI
Muestreo de posición	NO	SI
IMU	SI	SI
FOV	110 grados	100 grados
3D	Estereoscópico	Estereoscópico

Tabla 1: Comparación Oculus Rift DK1 con Oculus Rift DK2.

Resolución de pantalla

La resolución de pantalla del DK1 es de 720p mientras que la del DK2 es de 1080p. Pese a que estas resoluciones puedan parecer suficientes para una pantalla de esas dimensiones, lo cierto es que al estar situadas a apenas unos centímetros de los ojos y vistas a través de unas lentes de aumento se produce un efecto denominado backdoor. Este efecto consiste en que se aprecia el espacio que hay entre píxeles, como se puede ver en la siguiente imagen. Al aumentar la resolución de la pantalla, la densidad de píxeles aumenta y el efecto desaparece, por lo que se espera que la resolución del dispositivo comercial de Oculus tenga una resolución de 2K o 4K. En la figura 27 se puede ver este efecto captado en un Oculus Rift DK1.

Figura 27: Efecto backdoor de la pantalla del Oculus Rift DK1.

Diseño de los píxeles

El diseño de los píxeles es importante de cara a reducir el efecto backdoor. Mientras que en la disposición RGB convencional de los píxeles hace que el efecto backdoor sea fácilmente apreciable, la trama pentile utilizada en el DK2 contribuye a disminuir este efecto. La diferencia en la disposición de los píxeles se puede ver en la siguiente imagen:

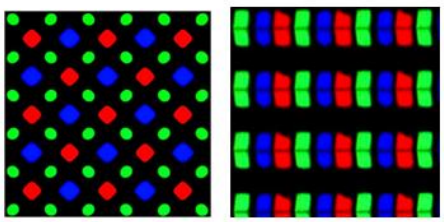

Figura 28: Izquierda - Trama Pentile del DK2. Derecha - Trama RGB del DK1.

OLED

La tecnología OLED permite que la pantalla del DK2 sea de baja persistencia, eliminando cualquier efecto de ghosting. La persistencia es cuando una imagen residual de la pantalla sigue siendo aún visible incluso cuando ya no se está mostrando esa imagen [16]. Este problema ocurre generalmente cuando se muestra una imagen fija durante largos periodos de tiempo, pero a menor escala también ocurre en las pantallas LCD con imágenes en movimiento, haciendo que los objetos mostrados parezcan desenfocados por la aparición de una estela de imágenes residuales. Debido a esta estela se suele denominar a este efecto ghosting, y se puede apreciar un ejemplo del mismo en la siguiente imagen que ha sido capturada en un simulador de Oculus Rift. [17]

Figura 29: Izquierda - Pantalla del DK1. Derecha - Pantalla de baja persistencia del DK2.

Tamaño de pantalla

El tamaño de la pantalla del DK2 es menor al del DK1, por lo que usa unas lentes más potentes para cubrir aproximadamente el mismo ángulo de visión. La reducción de la pantalla se ha llevado a cabo para reducir el tamaño del HMD, siendo así más cómodo para el consumidor final.

Latencia

Es una de las cosas más importantes en un HMD de realidad virtual, ya que si se quiere que la inmersión del usuario sea completa este debe de ver correspondidos sus movimientos en la vida real con los movimientos de su avatar en el mundo virtual. La latencia se define como la suma de los retardos temporales dentro de una red, por lo que en

nuestro caso estos retrasos vienen ocasionados por una parte por la lectura de datos del IMU, y por otra por la renderización y transmisión de las imágenes mostradas en la pantalla. Para disminuir la latencia Oculus Rift cuenta con funciones en su librería para predecir los movimientos del usuario, ayudando así a que la latencia final sea de unos 20 ms.

Frecuencia de refresco

Al igual que la latencia, la frecuencia de refresco de la pantalla también es muy importante de cara a que la inmersión sea completa, ya que el usuario debe tener la sensación de que no hay pantalla. La frecuencia de refresco recomendada para HMDs de realidad virtual es de 90 Hz, ya que por debajo de esta el cerebro puede notar la falta de imágenes, eliminando completamente la sensación de inmersión y pudiendo llegar a causar mareos.

Muestreo de orientación y posición

Una de las cosas que caracteriza a los HMDs de realidad virtual es la posibilidad del usuario de mirar a su alrededor. Esto se consigue gracias a un IMU que poseen las gafas que como se ha explicado en el apartado de descripción de componentes tiene una frecuencia de muestreo de 1000 Hz. El DK2 cuenta con la novedad de incluir un sistema de seguimiento de posición, cosa que consigue gracias a unos LEDs infrarrojos situados en la carcasa de las gafas y a una cámara infrarroja que detecta la posición de estos. El seguimiento de posición es importante a la hora de conseguir una mejor inmersión, ya que cuando se gira la cabeza es fácil realizar pequeños desplazamientos, aunque en el caso de este proyecto no sería muy relevante ya que lo que se busca es mover únicamente la cabeza del robot teleoperado.

FOV (campo de visión)

El ser humano posee un ángulo de visión medio de 180 grados en la horizontal, por lo que cuanto mayor sea el FOV de un HMD de realidad virtual mayor será la inmersión experimentada por el usuario. Oculus Rift posee un ángulo de visión de 110 grados, lo que le sitúa por encima de otros HMDs que había hasta la fecha, y siendo esto suficiente para lograr una inmersión muy buena ya que pese a que el ser humano cuenta con un ángulo mayor, la visión periférica que posee es bastante pobre y para mirar a las cosas suele hacerlo de frente.

4.2.2 Kinect

Como se ha explicado en el capítulo anterior, la cámara Kinect fue ideada por Microsoft para que se usase como controlador de videojuegos en su consola Xbox 360, pero debido a sus características ha gozado un éxito relativo en el campo de la investigación y la visión artificial, donde se ha usado en numerosos sistemas.

Figura 30: Microsoft Kinect para Xbox 360.

Técnicamente hablando, Kinect funciona gracias a un Software creado expresamente por Microsoft unido a una cámara ideada por la empresa israelí Prime Sense que permite detectar la distancia de los objetos. Además cuenta con una matriz de micrófonos que permiten identificar la fuente de sonido, así como de un pequeño servo para modificar la inclinación del dispositivo. La disposición de estos sensores y actuadores se puede apreciar en la figura 31. A continuación se describen cada uno de estos componentes [18]:

- Cámara RGB: se trata de una cámara VGA a color capaz de captar imágenes de una resolución de 640 x 480 píxeles a una frecuencia de 30 fps. Se utiliza para reconocimiento facial y para una técnica conocida como segmentación, que consiste en separar a las personas del fondo gracias a la información de profundidad y los datos obtenidos por la cámara RGB. Para que este método funcione correctamente es necesario que las cámaras estén bien calibradas, cosa que viene hecha de fábrica, aunque con los paquetes de ROS encargados de controlar la Kinect nos permiten realizar pequeños cambios en esta calibración.
- Sensor de profundidad: está formado por un proyector de láser infrarrojos, un sensor CMOS monocromo de resolución 320 x 240 píxeles y un módulo encargado de procesar los datos obtenidos. El método de funcionamiento de este sensor consiste en la proyección de una matriz de puntos de luz infrarroja, que al chocar

con un objeto refleja esta luz y es capturada por el sensor CMOS. En función de la cantidad de luz reflejada y capturada por la cámara, el módulo de procesamiento es capaz de determinar la distancia a la que se encuentran los objetos, ya que cuanto mayor es la distancia menor es la intensidad de la luz recibida. Este método tiene como mayor ventaja que al usar luz infrarroja puede funcionar bajo prácticamente cualquier condición de luz, pero sin embargo no se comporta bien con objetos negros. [19]

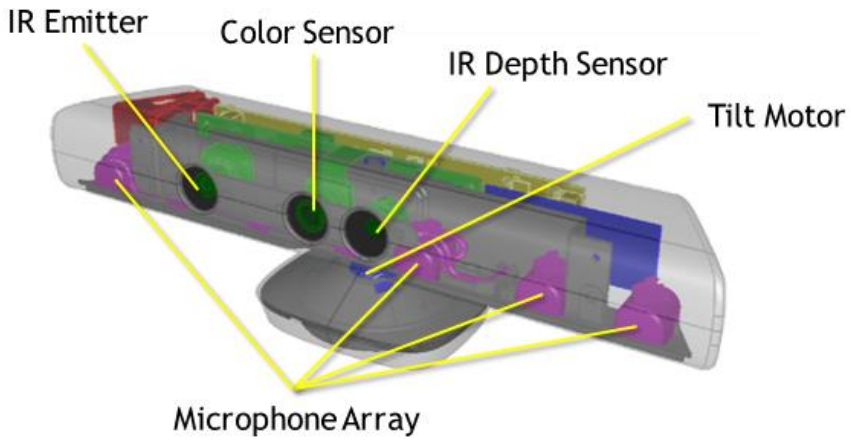

Figura 31: Disposición de los componentes de Kinect.

- Matriz de micrófonos: estos micrófonos son los encargados de determinar la posición de las fuentes acústicas que reciben. Estos es posible ya que cada uno recibe los sonidos de manera independiente, y en función de la intensidad con la que recibe un mismo sonido cada micrófono son capaces de determinar la posición de la fuente. Además, al contar con varios micrófonos se puede aplicar una cancelación de ruido notable y también cuentan con filtros para que sólo sean capturadas las frecuencias de la voz humana, entre 80 y 1100 Hz. En este proyecto no se ha incluido ningún tipo de reconocimiento de voz ni de transmisión de sonido, por lo que no se hará uso de ellos.
- Servo motor: sirve para controlar la inclinación de las cámaras, permitiendo ángulos máximos de ±27 grados sobre la horizontal. En nuestro caso disponemos de servos externos encargados de realizar esta función, por lo que no se ha usado y se ha mantenido en un ángulo fijo de 0 grados.

Dentro de estas características, cabe destacar que el ángulo de visión que ofrecen las cámaras de la Kinect es de sólo 57 grados en la horizontal, siendo esto un problema para la inmersión buscada ya que el ser humano tiene un ángulo de visión de 180 grados y las

Oculus Rift DK1 tienen un FOV de 110 grados que apenas es aprovechado. Además, otro problema encontrado es que la resolución de la nube de puntos proporcionada por los sensores tiene una resolución bastante baja, lo que hace que al visualizar los entornos sea difícil identificar pequeños objetos.

4.2.3 Plataforma de Servos Dynamixel

La plataforma de servos utilizada para poner a prueba el proyecto ha sido el esqueleto del robot asistencial Mini Maggie de la UC3M (Figura 32). Este esqueleto cuenta con cinco servos Dynamixel AX-12A cuyas funciones son:

- Dos para el movimiento de los brazos
- Uno para el movimiento del torso.
- Dos para los movimientos de la cabeza (pitch y yaw).

Para el proyecto que nos ocupa sólo ha sido necesaria la utilización de los dos servos que controlan la cabeza del robot, a los que se ha acoplado la cámara Kinect con la ayuda de unas bridas.

Figura 32: Estructura de servos Dymanixel con Kinect montada.

El control de los servos se realiza mediante una placa controladora USB2Dynamixel, que una vez conectada a una fuente de alimentación a un voltaje entre 9 V y 12 V alimenta a los servos con el amperaje necesario para mantener la posición deseada. Esta placa se conecta al PC mediante USB, y en este proyecto se va a controlar mediante el paquete de ROS dynamixel_controllers.

Los Servos Dynamixel AX-12A cuentan con las siguientes características:

Peso	54,6 gr
Dimensión	32 x50 x 40 mm
Resolución	0.29°
Ratio de reducción	254 : 1
Par motor	1,52 N·m (a 12.0 V , 1.5 A)
Velocidad sin carga	59 rpm (a 12 V)
Grados de giro	0° ~ 300°
Rotación continua	Sí
Temperatura de trabajo	-5ºC ~ +70ºC
Tensión de operación	9 ~ 12 V (tensión de operación recomendada 11.1 V)
Señal de comandos	Paquete digital
Tipo de protocolo	Comunicación serie asíncrona half duplex (8 bit,1 stop, no parity)
Conexión física	TTL Level Multi Drop (conector tipo daisy chain)
ID	254 ID (0~253)
Velocidad de comunicación	7343 bps ~ 1 Mbps
Feedback	Posición, temperatura, carga, tensión de entrada, etc.
Material	Plástico

Tabla 2: Características de servo Dynamixel AX-12A

Con objetivo de no dañar los servos y debido al elevado peso de la Kinect, se han limitado algunas de estas características.

- Velocidad: se ha limitado para que las fuerzas inerciales no sean muy grandes. Aun así se ha mantenido una velocidad lo suficientemente elevada como para seguir el movimiento normal de la cabeza.
- Grados de giro: para el motor encargado del movimiento Yaw se han establecido unos valores naturales para el movimiento del cuello humano, estableciendo estos en +45º y -45º. Así mismo, para el servo en cargado del movimiento Pitch (cabeceo) se han establecido unos valores de +33º y -33º, en este caso también con el objetivo de minimizar el momento ocasionado por la Kinect al inclinarse.

A continuación se adjuntan una serie de imágenes en la que se puede apreciar la disposición de los motores así como su alimentación y control:

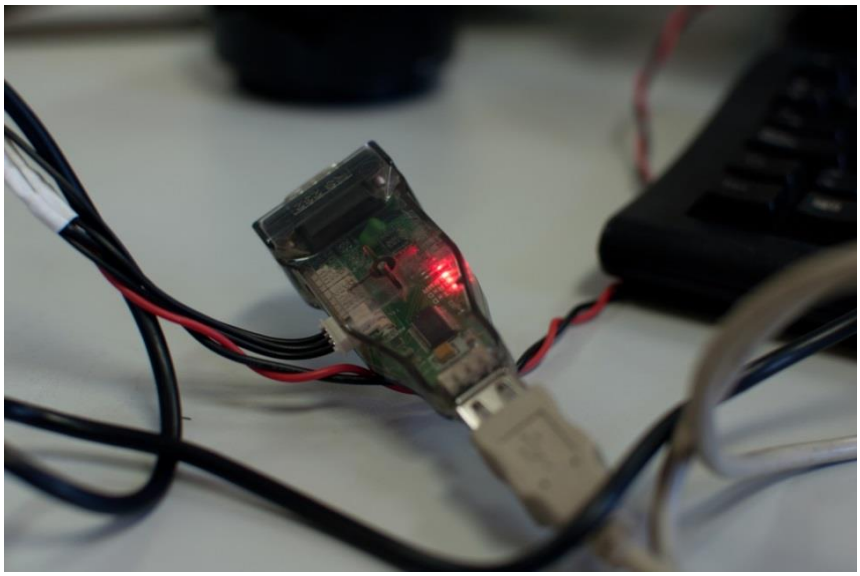

Figura 33: Placa controladora USB2Dynamixel.

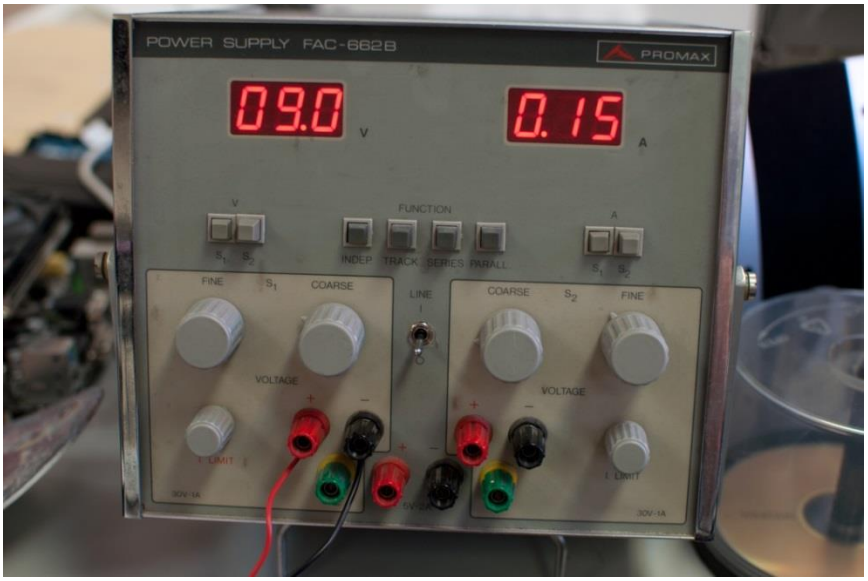

Figura 34: fuente de alimentación alimentando a los servos Dynamixel.

Figura 35: Detalle de servo Dynamixel AX-12A.

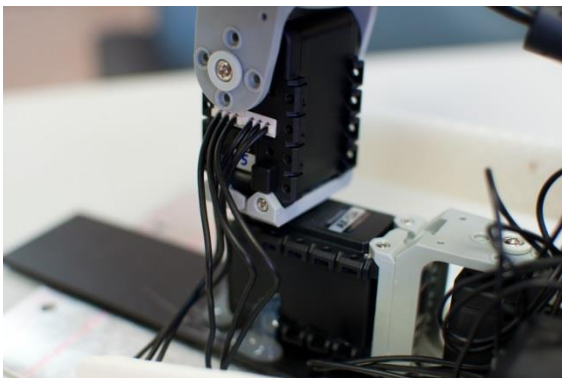

Figura 36: Montaje de los servos Dynamixel utilizados.

4.3 Software

4.3.1 ROS

Figura 37: Logotipo de ROS.

Introducción

El acrónimo ROS hace referencia a Robot Operating System, sin embargo no se trata de un Sistema Operativo al uso, si no que se define como una plataforma de desarrollo de software robótico o RSF (Robotics Software Framework) que ofrece diversas librerías y herramientas para este fin. ROS ofrece una infraestructura de desarrollo, despliegue y ejecución de sistemas robóticos, además de contar con un amplio repositorio compuesto por paquetes compatibles con todo tipo de robots. Entre las características que posicionan a ROS como el RSF de referencia se encuentran las siguientes:

- Abstracción de Hardware (HAL): hace que el programador no tenga que preocuparse por la implementación específica del Hardware y sus drivers, si no que las diferencias entre distintos dispositivos de la misma índole se reserva a archivos de configuración, permitiendo la aplicación de un mismo paquete a cualquier periférico compatible con ROS.
- Algoritmos de robótica con bajo acoplamiento: permite la modificación de los programas sin que la repercusión sea muy grande, evitando así tener que reescribir todo el código y, por lo tanto, favoreciendo la reutilización. Esto se aplica desde los algoritmos de bajo nivel a los de más alto nivel.
- Sistema de comunicaciones (middleware) basado en nodos. Un nodo es cualquier unidad de software del sistema, y se divide así con el objetivo de facilitar la reutilización del código y de dotar de independencia a la situación del nodo en el sistema. La comunicación entre nodos se hace mediante publishers y subscribers, que permiten el paso de mensajes. Gracias a esta organización, ROS permite crear redes P2P de componentes robóticos distribuidos.

- Simulación de sistemas robóticos en mundos virtuales con dinámicas de sólidos rígidos gracias a RViz (paquete de ROS).

Pese a que estas características ya estaban presentes en otros RSFs, ROS ha logrado unificar todas en un sistema completo y uniforme. Además cuenta con otras ventajas como ser multilenguaje (C++ y Python), permitir comunicaciones p2p (peer2peer), está orientado a herramientas, es ligero y Open Source. [20]

ROS comenzó a desarrollarse en 2007 en la Universidad de Stanford para dar soporte al proyecto STAIR, un robot con inteligencia artificial desarrollado por la propia universidad. Actualmente ROS está siendo desarrollado de la empresa Willow Garage, una SpinOff de Stanford, que además cuenta con el apoyo con diversos grupos de investigación como el ya citado STAIR.

Conceptos

Antes de entrar a explicar el desarrollo del proyecto en el siguiente capítulo, conviene familiarizarse con una serie de conceptos del ecosistema de ROS, que basándonos en la información proporcionada por su web se dividen en tres niveles: sistema de archivos, computación gráfica y comunidad. [21]

Sistema de archivos

Entre los recursos que utiliza ROS se puede encontrar los siguientes:

- Paquetes: son la unidad principal de organización. Un paquete puede contener nodos, archivos de configuración, bibliotecas ROS-dependientes, o cualquier otro archivo que se organice y relacione eficazmente con los demás.
- Metapaquetes: son paquetes que sirven para representar otros paquetes relacionados.
- Manifiestos: proporcionan metadatos de un paquete en un fichero denominado packege.xml. Entre estos metadatos se incluyen nombre del paquete, versión, descripción, información sobre la licencia y dependencias de otros paquetes.
- Repositorios: colección de paquetes que comparten un sistema VCS (sistema de control de versiones) común.
- Tipos de mensaje (msg): descripción de los tipos de mensajes que define la estructura de los datos intercambiados en ROS.

Computación gráfica

Se denomina nivel de computación gráfica a la red p2p de los procesos de ROS que se encargan de procesar la información. Dentro de la computación gráfica se definen los siguientes conceptos, implementados todos ellos dentro del repositorio ros_comm:

- Nodos: son los procesos encargados de la computación o cálculo de la información recibida. Es una de las unidades básicas dentro de ROS, y el control de un solo robot se suele componer de varios nodos. La creación de nodos se hace usando las librerías que proporciona ROS, como roscpp (para C++) o rospy (para Python).
- Master: ROS Master proporciona un nombre de registro así como un índice al resto de elementos encargados de la computación gráfica. Sin ROS Master los nodos no podrían encontrarse unos a otros ni intercambiar mensajes.
- Servidor de parámetros: forma parte del Master y se proporciona un lugar donde almacenar la información.
- Messages: es la forma que utilizan los nodos de comunicarse entre ellos. Los mensajes son una estructura de datos y pueden contener varios campos en su interior (de tipo integer, boolean, char, etc), permitiendo además la anidación de mensajes dentro de otros mensajes.
- Topics: un topic es un nombre que se le da al lugar donde un nodo envía o lee un mensaje determinado. El envío o lectura de mensajes se realiza mediante los elementos publisher y subscriber, que especifican el nombre del topic al que pertenecen. Para un mismo topic puede haber varios publishers o subscribers, permitiendo así separar la información de su origen o destino, aumentando la reutilización de códigos y su versatilidad a la hora de aplicarlo en otros proyectos.

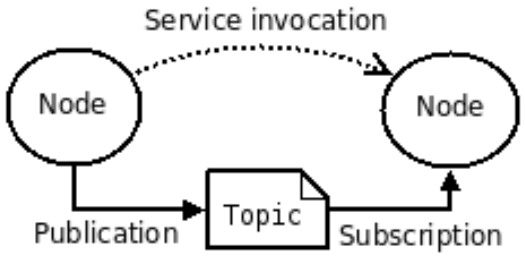

Figura 38: Esquema de comunicación entre nodos vía publishers y subscribers.

- Services: es el sistema que utiliza ROS para procesar las operaciones de tipo petición / respuesta, o lo que es lo mismo, callbacks. Los services están definidos por dos estructuras de messages, uno para la petición y otro para la respuesta.
- Bags: sirven para guardar la información generada por un nodo para luego poder reproducirla en futuras pruebas.

En la siguiente imagen se puede apreciar la estructura de una plataforma de ROS, con varios nodos que intercambian información entre sí y estando todos ellos listados bajo el ROS Master.

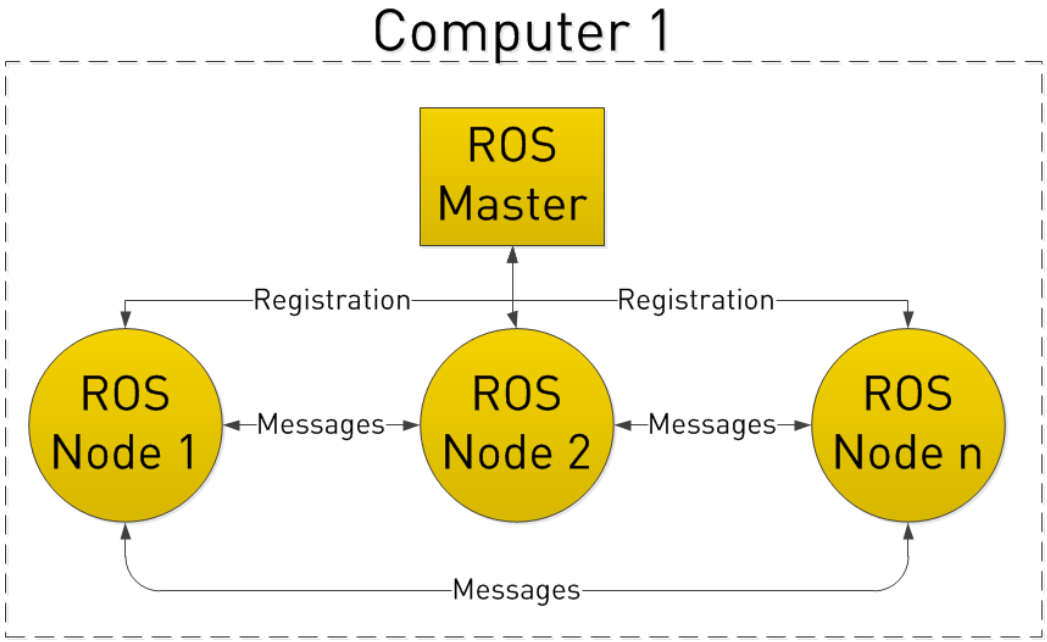

Figura 39: Esquema general de un sistema controlado por ROS.

Comunidad

Por último existe lo que ROS define como el nivel de la comunidad. Uno de los puntos fuertes de ROS respecto a otros RSF es el alto grado de implicación de la comunidad, ofreciendo una gran cantidad de documentación que favorece la expansión del ecosistema. Dentro de este ecosistema conviene destacar los siguientes términos:

- Distribuciones: se podría definir una distribución como una versión del framework, que incluye todas las librerías necesarias para ponerlo en funcionamiento. En este proyecto se ha usado ROS Hydro Medusa, ya que era la última versión disponible al comienzo del proyecto, sin embargo el 22 de Julio de 2014 se lanzó una nueva versión bajo el nombre de ROS Indigo Igloo.

- Repositorios: ROS ofrece un repositorio donde la comunidad puede lanzar sus propios paquetes de ROS listos para ser utilizados por otras personas en diversos proyectos.
- ROS Wiki: proporciona documentación sobres los paquetes disponibles. El registro es libre, por lo que cualquiera puede aportar la información que crea que puede ser de utilidad.
- ROS Answers: lugar donde la comunidad puede publicar sus dudas o preguntas de forma abierta.

Papel en el proyecto

ROS es el núcleo del proyecto que nos ocupa, haciendo todas las labores de computación de datos y de comunicación de los diversos periféricos. El sistema ROS se ha implementado en Ubuntu 12.04 LTS, y para la realización del proyecto se han necesitado los siguientes paquetes disponibles en el repositorio, además de un paquete de creación propia para el objetivo del proyecto:

- OpenNI: drivers de la Kinect.
- Dynamixel_controller: drivers controladores de los servos Dynamixel.
- Oculus_rviz_plugin: plugin para RViz que permite la visualización de su entorno gráfico desde las Oculus Rift.
- Rqt_gui: dota a ROS de una interfaz gráfica de usuario. Para este paquete es para el que se ha diseñado el plugin.
- Rqt_servo: plugin diseñado para rqt_gui. Sirve para establecer la posición de los servos o bien mediante las Oculus Rift de forma automática, o bien de forma manual desde la interfaz.

En el capítulo de diseño se explicarán más a fondo cada uno de estos paquetes utilizados, especialmente el de creación propia.

4.3.2 Qt Creator

Figura 40: Logotipo de Qt.

Introducción

Qt Creator es un entorno de desarrollo integrado (IDE) de aplicaciones multiplataforma que admite los lenguajes C++, Java o QML, y forma parte del SDK Qt GUI.

En este proyecto se ha usado Qt Creator para facilitar la edición de paquetes de ROS, así como para la creación de una interfaz gráfica para el plugin desarrollado para el control de los servomotores mediante las Oculus Rift, así como la implementación de un control manual desde la misma interfaz.

Características

Las principales características con las que cuenta Qt Creator son:

- Diseño de aplicaciones de interfaz de usuario basadas en widgets de Qt de forma rápida, visual y sencilla gracias al editor Qt Designer, incluido en el IDE.
- Creación rápida de aplicaciones gracias al asistente de proyectos.
- Desarrollar aplicaciones con el editor de código en C++, que ofrece un gran control y poder sobre la aplicación creada.
- Desarrollo de proyectos multiplataforma, pudiendo exportar las aplicaciones a diferentes sistemas operativos como Windows, Mac OS, Linux, Symbian, MeeGo, Maemo o Android.
- Depuración de código mediante la interfaz gráfica.
- Usar herramientas de análisis de código para verificar la administración de memoria de las aplicaciones creadas.
- Acceder a información fácilmente gracias a la ayuda contextual que ofrece Qt Creator.

Además de estas características principales, cabe destacar que la elección de Qt para la realización del proyecto se ha hecho ya que proporcionaba compatibilidad con los proyectos basados en CMake utilizados en los paquetes de ROS, lo que permitía su fácil edición. Así mismo, se tenían conocimientos previos del uso del programa gracias a la asignatura de Informática Industrial cursada durante la carrera, lo que terminó de decantar la balanza hacia el uso de Qt Creator y no de otros IDEs compatibles con ROS.

Capítulo 5. Presupuesto

En este capítulo se va a detallar el presupuesto del proyecto realizado, incluyendo el coste del hardware y el software, así como datos sobre el coste humano en el caso de que existiese un sueldo por el desarrollo del sistema.

5.1 Costes Hardware

En la siguiente tabla se va a detallar el precio de los componentes utilizados, pero sin embargo es conveniente señalar que la mayoría de ellos ya se tenían en la universidad y que además de este proyecto han tenido otros usos, al igual que los recién adquiridos que serán usados en el futuro para otros proyectos. Aun así se ha creído conveniente incluir el precio completo de cada uno, sin ningún tipo de porcentaje de uso o depreciaciones, ya que el objetivo del proyecto es establecer un sistema de telepresencia permanente.

Componente	Cantidad	Precio unitario	Precio total
Oculus Rift DK1	1	300 €	300 €
Microsoft Kinect	1	99 €	99 €
Servo Dynamixel AX-12A	2	43 €	86 €
USB2Dynamixel	1	45 €	45 €
Ordenador gama media	1	400 €	400 €
		TOTAL	**930 €**

Tabla 3: Costes hardware.

5.2 Costes Software

Todo el software empleado en la realización del proyecto final es software libre no comercial y por tanto no incurren en ningún gasto. Aun así, en el planteamiento inicial del proyecto se definió que se iba a diseñar un soporte gracias a Catia, un programa comercial de diseño 3D, cosa que se hizo. Sin embargo no se ha creído conveniente incluir en este presupuesto el coste de dicho programa ya que para la versión final del proyecto no se han usado dichos diseños.

Software	Precio
Ubuntu 12.04 LS	0 €
ROS Hydro	0 €
Qt Creator	0 €

TOTAL	0 €

Tabla 4: Costes Software.

5.3 Coste humano

El coste humano real de este proyecto se puede decir que es nulo, ya que se trata de un trabajo de fin de grado. Sin embargo, en este apartado se adjunta un presupuesto de lo que costaría realizar este proyecto fuera del ámbito universitario, estableciendo unos sueldos de acorde a lo que se ofrece actualmente en el mercado para un graduado en ingeniería. Las horas de trabajo que se especifican en este apartado son aproximadas, ya que no se ha llevado un registro exhaustivo de las mismas.

Tarea	Horas	€/hora	Total
Análisis	10	15	150
Planteamiento y diseño	30	15	450
Desarrollo	200	15	3000
Documentación	100	15	1500
		TOTAL	**5.100 €**

Tabla 5: Coste humano.

5.4 Coste total

Con este presupuesto el coste total del proyecto es de 6.030 €. A este coste total se le podría añadir una tasa de costes indirectos del 20 %, lo que supondría que el coste total fuese de 7.236 € o siete mil doscientos treinta y seis euros.

Capítulo 6. Diseño e Implementación del Sistema

En este apartado se va a detallar el modo en el que se ha abordado el proyecto y cómo se ha ido desarrollando. Al final del capítulo se detallarán los cambios que se han incluido en el proyecto en relación con el planteamiento inicial, comentando las consecuencias de cada uno de ellos.

6.1 Planteamiento del proyecto

El objetivo del proyecto es la realización de un sistema de telepresencia de calidad y de bajo coste en comparación con lo que existía hasta ahora. Esta idea surgió debido a la creciente popularidad que están teniendo los dispositivos de realidad virtual, con Oculus Rift a la cabeza, que en un futuro temprano ofrecerán HMDs con una alta calidad y a un precio accesible para el consumidor medio.

Para la realización del proyecto inicialmente se estudió la posibilidad de la construcción de un HMD casero, que contando con todos los componentes se situaría en un precio de unos 150 € aproximadamente. Sin embargo finalmente se decidió adquirir un kit de desarrollo de Oculus Rift por las siguientes razones:

- Calidad: como se ha detallado en el capítulo de Hardware, Oculus Rift cuenta con unas características técnicas que lo sitúan muy por encima de los dispositivos que existían hasta ahora, y por supuesto también por encima de un HMD de construcción casera.
- Inmediatez: no necesita ninguna clase de montaje, que puede llevar a toda clase de problemas técnicos (malas soldaduras, incompatibilidad de componentes, etc).
- SDK: Oculus VR provee a los desarrolladores de un completo SDK para la fácil implementación de sus proyectos basados en Oculus Rift. En el caso del HMD casero habría que realizar las librerías necesarias para el funcionamiento del proyecto.

- Comunidad y popularidad: Oculus Rift cuenta con una gran comunidad a la que acudir en caso de encontrarse con problemas. Además, debido a la popularidad del dispositivo el proyecto cuenta con el aliciente de poder ser usado por otras personas que dispongan de un Oculus Rift.
- Precio: el precio del kit de desarrollo utilizado es de 300 $, por lo que se creyó que la diferencia no era significativa teniendo en cuenta todas las ventajas que proporcionaba para este u otros futuros proyectos de la universidad.

Siguiendo con la parte de hardware del proyecto, se necesitaba dotar al usuario de visión estereoscópica. Para ello, se pensó que lo ideal sería usar una cámara de visión estéreo o como alternativa utilizar dos webcam calibradas para captar imágenes estéreo. En este punto se consideró importante que las cámaras usadas tuviesen un ángulo de visión lo más amplio posible para favorecer la inmersión, dado que Oculus Rift proporciona un FOV de 110 grados. Pese a estas consideraciones, finalmente se utilizó una cámara Kinect por motivos que se especificarán en el apartado de cambios.

Por último, se necesitaba un soporte para las cámaras y se pensó que lo mejor es que fuese uno que proporcionase movimiento en los tres ejes (pitch, roll y yaw, que se puede ver su disposición en la Figura 41), ya que las Oculus proporcionan información de cada uno de ellos y así la inmersión obtenida por el usuario sería mayor. Se intentó buscar un soporte adecuado, pero debido a que el tercer eje (roll) no es necesario para tener una visión completa del entorno, no se encontró ninguno ya construido, por lo que se decidió diseñar uno gracias al programa de diseño 3D Catia para posteriormente imprimirlo en las impresoras 3D de la universidad. Para el control de los servos se pensó que lo más adecuado y sencillo era la utilización de una placa de desarrollo Arduino, que se comunicase con el PC por puerto serie para recibir las posiciones de cada motor. Al igual que con el caso de Kinect, en el proyecto final se cambió el sistema inicialmente planteado y se usó como soporte un esqueleto del robot asistencial Mini Maggie, que cuenta con dos servos Dynamixel para los ejes pitch y yaw, controlados por una placa proporcionada por el fabricante.

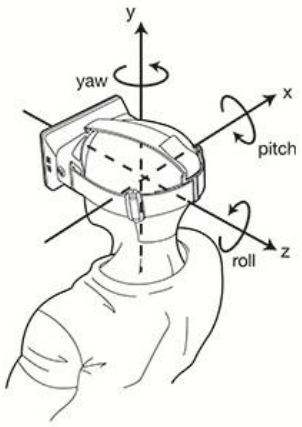

Figura 41: Disposición de los ejes pitch, roll y yaw en Oculus Rift.

En lo que a Software se refiere, se necesitaba una interfaz que mostrase cierta información de todo el sistema y que comunicase todas las partes del mismo. Para ello se decidió que lo ideal sería usar Qt Creator, del que ya se tenían conocimientos previos, para la creación de la interfaz y del framework ROS para la comunicación entre los diversos componentes. Así mismo era necesaria la creación de un programa para Arduino que controlase los tres servomotores con la información recibida por el puerto serie. Con los cambios introducidos, ROS adquirió más relevancia y se usó en prácticamente todos los puntos del proyecto, desde el procesamiento y transporte de la información hasta el control de los servos Dynamixel, aunque por comodidad y facilidad se usó Qt para la creación de la GUI así como para la edición de los paquetes de ROS.

Una vez definidos el hardware y el software que se quería usar, el desarrollo del proyecto se planteó de forma que se siguiese una hoja de ruta preestablecida, sin embargo y como se ha comentado anteriormente, a medida que se realizaba el proyecto se fueron incluyendo modificaciones en la misma, unas veces con objetivo de facilitar el trabajo y otras veces por causa de limitaciones surgidas a la hora de implementar las ideas que se habían tenido. A continuación se detalla la hoja de ruta establecida inicialmente, y posteriormente la hoja de ruta seguida en el proyecto final con los cambios introducidos.

Hoja de ruta inicial.

- Búsqueda de información sobre proyectos similares.

- Familiarización con las Oculus Rift, calibración y comprensión del SDK.

- Obtención de los valores del IMU de las Oculus Rift mediante un programa de Qt.

- Realización de un programa para Arduino que fuese capaz de controlar 3 servos mediante valores que le llegasen por puerto serie.

- Realización de un programa en Qt capaz de procesar y enviar los datos del IMU de las Oculus a Arduino.

- Diseño de un soporte en Catia e impresión del mismo.

- Prueba de movimiento del soporte con carga mediante el programa de Qt y Arduino.

- Captura y deformación de imágenes para mostrarlas en las Oculus.

- Creación de una GUI con toda la información y funciones necesarias.

Hoja de ruta definitiva

- Búsqueda de información de proyectos similares.

- Familiarización con las Oculus Rift, calibración y comprensión del SDK.

- Familiarización con ROS.

- Funcionamiento del plugin de Oculus para RViz.

- Captura de imágenes de Kinect y representación en RViz.

- Configuración del paquete encargado de mover los servos Dynamixel.

- Creación de un paquete encargado de procesar la información obtenida de las Oculus y enviarla a los servos.

- Diseño e implementación de una GUI como plugin para rqt con las funciones del paquete creado.

- Creación de un archivo launch encargado de lanzar todos los servicios necesarios.

6.2 Desarrollo del proyecto

6.2.1 Configuración y calibración del Oculus Rift DK1

Una vez recibido el DK1 de Oculus Rift, lo primero que hubo que hacer fue familiarizarse con el mismo y sus características técnicas y físicas. Del mismo modo, se descargó el SDK de su página web y se estudió su documentación a fondo.

La conexión del dispositivo al PC se hizo mediante un cable HDMI y un cable USB proporcionado dentro del DK1. Una vez conectado y siendo reconocido por el sistema operativo, dentro de la configuración de pantallas de Ubuntu se estableció el Display de Oculus Rift como segundo monitor y con una resolución de 1280 x 800 píxeles. En el Apéndice A se puede encontrar una guía que explica cómo realizar esta configuración.

Por otra parte, dentro del funcionamiento de las Oculus Rift una parte importante es la correcta calibración de las mismas para el usuario que vaya a usarlas. Esta calibración consiste en establecer el IPD (ver figura 42) del usuario, que se puede hacer o bien manualmente si se conoce esta distancia o bien utilizando una herramienta incluida en el SDK, que mide el IPD siguiendo unos sencillas pasos con el HMD puesto. Al igual que con la configuración del display, en el Apéndice A se puede encontrar una guía que explica cómo realizar esta calibración.

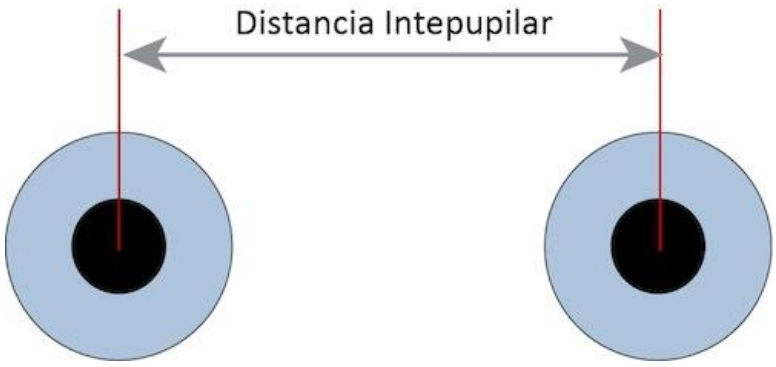

Figura 42: Distancia interpupilar.

6.2.2 Plugin de Oculus Rift para RViz

El plugin de Oculus Rift para RViz, desarrollado por David Gossow, añade la opción de utilizar el HMD como un nuevo Display de RViz, mostrando en él las imágenes del simulador de forma estereoscópica y con la deformación adecuada para su correcta visualización.

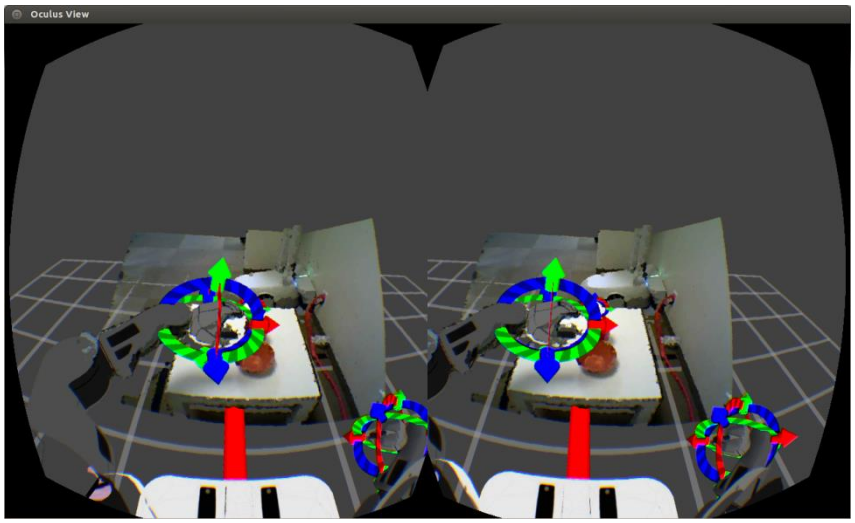

Figura 43: Visión en Oculus Rift de RViz gracias al plugin.

Este plugin ofrece la posibilidad de mostrar las imágenes tanto en una nueva ventana como directamente en las Oculus Rift si se marca la opción en su menú de **Render to Oculus**. Para que se pueda utilizar la opción **Render to Oculus** es necesario que las Oculus Rift estén bien configuradas como segundo monitor en Ubuntu y con una resolución de 1280 x 800 píxeles, como se detalla en el Apéndice A.

Dentro de las características del plugin se encuentra la opción de publicar la posición de las Oculus Rift en un tf frame (sistema de coordenadas) denominado **/Oculus**, que en este proyecto se ha cambiado el nombre a **/tf_oculus** para facilitar su identificación. Esta opción es importante para el funcionamiento del proyecto ya que se utiliza posteriormente para determinar el ángulo que deben adoptar los servomotores.

6.2.3 Captura de imágenes de Kinect y representación en RViz

La única forma de obtener una imagen 3D a partir de la cámara Kinect es mediante la obtención de una nube de puntos y su posterior representación de la misma en RViz. Para la obtención de la nube de puntos se ha usado el paquete de ROS **OpenNI**, que proporciona los drivers necesarios para la integración de Kinect en ROS.

Existe la posibilidad de que la cámara Kinect no se encuentre correctamente calibrada, para ello en el Apéndice A se puede encontrar una guía de calibración.

6.2.4 Control de los servos Dynamixel

Para el control de los servos Dynamixel se ha utilizado el paquete disponible para ROS denominado **dynamixel_controllers**, que permite controlar varios servos del fabricante al mismo tiempo.

Una vez instalado, se ha creado un paquete siguiendo el tutorial de la documentación del paquete **dynamixel_controllers** disponible en la página web de ROS. Dentro de la creación de este paquete se han editado ciertos parámetros de determinados archivos para ajustar el control de los servos a lo que se buscaba en el proyecto. La configuración de este paquete se puede encontrar en el Apéndice A.

Con el paquete ya creado y configurado, solo falta ejecutar los archivos launch correspondientes para poner en funcionamiento los servos.

Con el primer launch que de los que se deben iniciar, se reconoce el número de servos conectados y guarda su número identificador (ID). En nuestro caso se reconocen cinco servos, ya que el esqueleto de Mini Maggie usado contiene servos para mover otras partes del cuerpo además de la cabeza, pero de estos cinco sólo nos interesan aquellos que tienen el ID 4 y 5, que son los correspondientes a los movimientos de los ejes pitch y yaw de la cabeza. La configuración de estos dos servos se realiza con los otros dos archivos launch que han sido llamados, que leen la información de los ficheros yaml correspondientes que han sido editados como se ha explicado en el Apéndice A.

Una vez iniciados todos los servicios correctamente, basta con publicar la posición deseada en radianes de cada servo en un topic especificado. En nuestro caso la posición de los servos correrá a cargo de un paquete creado específicamente con esta función, que leerá la información obtenida de las Oculus y las publicará en los siguientes topics:

```
/yaw_controller/command

/pitch_controller/command
```

6.2.5 Creación de un paquete de ROS

Uno de los pilares del proyecto pasa por el movimiento coordinado de los servo motores en función de la información obtenida por las Oculus Rift. Para conseguir esto fue necesaria la creación de un paquete específico de ROS que se encargase de recabar la

información obtenida de los sensores, y acto seguido la publicase en los topics correspondientes de cada servo.

Para la obtención de la información de los sensores de las Oculus Rift se ha hecho uso del tf frame publicado por el plugin de Oculus para RViz, denominado **/tf_oculus**, que comprándolo con el tf frame de referencia llamado **/camera_link** se pueden obtener los ángulos de inclinación de los ejes pitch, yaw y roll del HMD. A continuación se adjunta la parte del código encargada de esto:

```
int main(int argc, char** argv){
  …
  tf::TransformListener listener;
  double deg_pitch,deg_roll,deg_yaw;
  …
  while (node.ok()){
    …
    tf::StampedTransform t;
    try{
      listener.lookupTransform("/camera_link", "/tf_oculus",ros::Time(0), t);
    }
    catch (tf::TransformException ex){
      ROS_ERROR("%s",ex.what());
      ros::Duration(1.0).sleep();
    }
    tf::Matrix3x3(t.getRotation()).getRPY(deg_roll,deg_pitch,deg_yaw);
    …
  }
  return 0;
};
```

Dentro de este código conviene destacar las siguientes líneas que son las encargas de realizar todo el procesamiento de la información:

```
listener.lookupTransform("/camera_link", "/tf_oculus",ros::Time(0), t)
```

Con el método lookupTransform de la clase TransformListener se obtiene la transformada del frame **/camera_link** al frame **/tf_oculus**, almacenando esta información en la variable **t**, definida anteriormente como StampedTransform.

```
tf::Matrix3x3(t.getRotation()).getRPY(deg_roll,deg_pitch,deg_yaw);
```

De la transformada almacenada anteriormente en **t** se obtiene la rotación en forma de ángulos roll, pitch y yaw, y se almacena en las variables **deg_roll**, **deg_pitch** y **deg_yaw**. Pese a que en nuestro proyecto se ha descartado el uso del eje roll, se incluye dentro del código para facilitar su implementación en futuros proyectos.

Una vez obtenidos los datos de los sensores, bastan con publicarlos en sus correspondientes topics para que los sensores adquieran la posición deseada. Estos e consigue con el siguiente código:

```
int main(int argc, char** argv){

  …

  ros::NodeHandle node;

  ros::Publisher pub_yaw = node.advertise<std_msgs::Float64>("/yaw_controller
/command", 100);

  ros::Publisher pub_pitch = node.advertise<std_msgs::Float64>("/pitch_contro
ller/command", 100);

  double deg_pitch,deg_roll,deg_yaw;

  …

  while (node.ok()){

    std_msgs::Float64 yaw,pitch;

    pitch.data = -deg_pitch;

    yaw.data = deg_yaw;


    pub_yaw.publish(yaw);

    pub_pitch.publish(pitch);

    …
```

```
  }
  return 0;
};
```

En este fragmento de código se crean dos publishers, uno para cada uno de los servos controlados, y se enlazan con el topic donde queremos que se publiquen los ángulos obtenidos. Posteriormente se crean mensajes de tipo Float64, en los cuales se almacena la información obtenida anteriormente de la transformada **t** y el método getRotation, y se publican los mensajes en el topic especificado.

6.2.6 Creación de un plugin para rqt_gui

Una vez creado el paquete de ROS encargado del movimiento de los servos, ya se tenía todo lo necesario para poner en funcionamiento el sistema de telepresencia. Sin embargo, con objeto de facilitar el trabajo al usuario final y de añadir algunas funcionalidades al proyecto se decidió crear una interfaz gráfica de usuario.

El uso del plugin de Oculus para RViz obliga a que la interfaz de RViz esté abierta para que se muestren las imágenes en las Oculus Rift, por lo que las alternativas disponibles para hacer esto son limitadas. Para que el modo de uso fuese lo más sencillo posible se optó por usar el paquete de ROS **rqt_gui**, que dota a ROS de una interfaz de usuario para múltiples de sus funciones, incluyendo entre ellas a RViz, y las muestra en una única ventana pudiendo organizarlas modularmente. Para el caso que nos ocupa, fue necesaria la creación de un plugin para **rqt_gui** que incluyese las características del paquete creado en el apartado anterior, es decir, que se encargase del procesamiento de la información proporcionada por las Oculus y se la enviase a los servos.

En cuanto a lo que la GUI se refiere, se diseñó una interfaz que muestra la información obtenida por las Oculus, y por tanto, la posición de los servos, en un sistema de barras acompañado del valor numérico en grados. Además se incluyó un modo de control manual de los servos en el que se selecciona el valor deseado deslizando un indicador sobre las barras donde se muestra la información en el modo automático. Por último, se incluyó un botón de Reset que se encarga de establecer el origen del eje Yaw en la posición en la que se encuentre en el momento de la pulsación. El funcionamiento de esta interfaz se explica a fondo en el manual de usuario del Apéndice A.

6.2.7 Creación de un fichero launch

Por último, se planteó la realización de un único launch encargado de lanzar todos los nodos necesarios para el funcionamiento del proyecto. Con este launch se inicia el paquete controlador de Kinect **OpenNi**, así como el encargado del control de los servos **dynamixel_controller** junto con sus ficheros de configuración de cada servo. De esta forma, el trabajo del operador se reduce considerablemente, teniendo únicamente que lanzar este fichero launch en una terminal y **rqt_gui** en otra.

6.3 Modificaciones técnicas

En este apartado se va a detallar por qué se decidió cambiar el planteamiento inicial del proyecto en determinados puntos de la realización del mismo, y las consecuencias que han tenido estos cambios en el resultado final del proyecto.

6.3.1 Uso de Kinect en lugar de una cámara estereoscópica:

Pese a que inicialmente se había definido el uso de una cámara estéreo, finalmente para la realización del proyecto se ha usado Kinect. Esto ha sido así principalmente por la no disponibilidad en la universidad de una cámara estéreo libre, pero ante la posibilidad de comprar una o la obtención de imágenes estéreo mediante dos webcams independientes, se prefirió usar Kinect ya que esta está presente en muchos robots que incluyen un sistema de visión, por lo que el proyecto podría ser aplicado a un mayor número de robots.

Entre las consecuencias de esta decisión se pueden contar los siguientes pros y contras:

Pros:

- Compatibilidad con un mayor número de robots debido al gran éxito de la Kinect.
- Mayor información disponible.
- Posibilidad de usar el plugin de Oculus para RViz.

Contras:

- Imposibilidad de obtener una imagen estereoscópica pura, teniendo que obtener las dos imágenes variando el punto de vista sobre una nube de puntos obtenida gracias al sensor de profundidad.

- Calidad de imagen mucho menor, empeorando la sensación de telepresencia. Esta pérdida de calidad es debida a la baja resolución que ofrece el sensor de profundidad de Kinect.
- Mayor peso, haciendo que los servos tengan que ser más potentes.

6.3.2 Cambio de soporte y servos

Como se ha detallado en el apartado de contras del uso de la Kinect, el peso de esta es mucho mayor que la de una cámara estéreo convencional. Con esta carga (500 gramos) se hicieron nuevas pruebas de estrés sobre el soporte de tres ejes diseñado con Catia (figura 44), dando como resultado que los servos utilizados hasta el momento no daban la suficiente potencia como para soportar la Kinect. Al estudiar las posibles causa se concluyó que los momentos de inercia creados por la carga eran demasiado grandes para los servos, y se estudió la posibilidad de resdiseñar el soporte con una serie de rodamientos de tal forma que el peso de la cámara no fuese soportado completamente por los servos, si no que se apoyase sobre el propio soporte. Finalmente, se pensó que aún con esa alternativa el par de los servos no sería el suficiente para soportar el peso de la Kinect, y se decidió cambiarlos por unos más potentes.

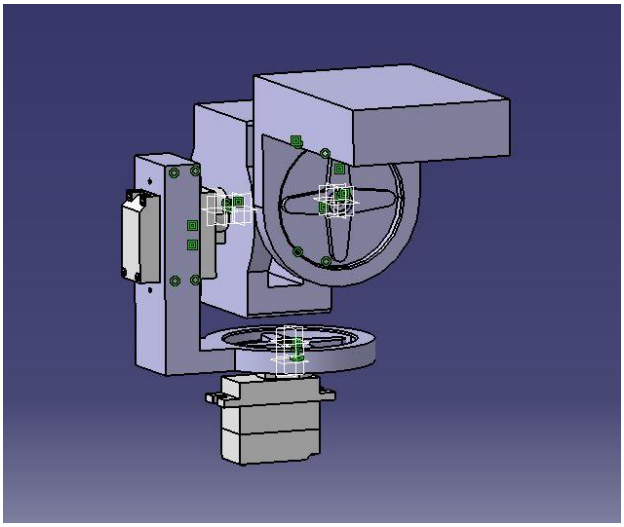

Figura 44: Primer diseño en Catia del soporte de Kinect.

En ese momento, en el departamento de robótica se contaba con un esqueleto del robot asistencial Mini-Maggie, y se me ofreció la posibilidad de usarlo para realizar el proyecto, a lo que acepté teniendo en cuenta los siguientes factores:

Pros:

- Soporte ya construido y listo para ser usado.
- Facilidad de uso de los servos Dynamixel gracias a paquetes de ROS ya existentes.
- Facilidad de portar el trabajo realizado. Hasta ahora se estaba trabajando con servos controlados por Arduino, y la forma de enviar la información era publicando la misma en un topic determinado. En el caso de los servos Dynamixel la forma de control era enviar la posición deseada a un topic especificado, por lo que solo hubo que cambiar la dirección donde se enviaba antes la información para que funcionase el paquete de ROS que se había desarrollado.
- Uso del esqueleto de un robot real, lo que demuestra la alta compatibilidad del proyecto con robots controlados por ROS.
- Mejores servos, con un control mucho más preciso.

Contras:

- Pérdida del eje "Roll". En un principio se decidió incluir este eje para que la inmersión del usuario fuese mayor, pero se tenía en cuenta que no era necesario para tener una visión completa del entorno por lo que no se consideró especialmente importante la pérdida del mismo.

6.3.3 Uso de ROS

Inicialmente se planteó realizar el proyecto en Qt, desarrollando un programa que se encargase de realizar todas las operaciones necesarias para hacer funcionar el sistema de telepresencia. Esto era así ya que se poseían conocimientos de Qt de haberlo utilizado a lo largo de la carrera, cosa que no ocurría con ROS. Sin embargo, puesto que el objetivo era realizar un sistema que fuese aplicable al mayor número de robots posibles, se decidió utilizar ROS ya que es el sistema más extendido dentro de este campo.

Entre las ventajas y desventajas de este cambio se encuentran las siguientes:

Pros:

- Compatibilidad con un mayor número de robots.
- Mayor información y recursos disponibles. Como se ha detallado en el capítulo de descripción de software, uno de los puntos fuertes de ROS es la comunidad y la cantidad de aportes que realizan. En el caso de este proyecto, el uso de ROS ha

posibilitado aprovechar los paquetes disponibles para el uso de Oculus Rift en RViz.

- Facilidad a la hora de realizar las comunicaciones. El sistema que utiliza ROS de nodos facilita mucho la tarea de comunicar unas partes del sistema con otras, además de que en caso de que algo requiera modificaciones no es necesario reescribir otras partes del código.
- Enfoque del proyecto hacia el mundo de la investigación y el desarrollo.

Contras:

- Necesidad de aprendizaje de ROS. Inicialmente no se tenían conocimientos sobre ROS, por lo que antes de realizar el proyecto fue necesario familiarizarse con el entorno y su funcionamiento.
- Dificultad de uso. Se requieren unos mínimos conocimientos de ROS para poner en funcionamiento el sistema desarrollado, mientras que con una aplicación de Qt se habría tenido un programa con todo lo necesario y fácilmente accesible.
- Enfoque del proyecto a un público más avanzado. Un sistema de telepresencia puede ser interesante también para el gran público y no sólo para los profesionales de la robótica, pero esta posibilidad se descarta al incluir ROS.

6.3.4 Uso de RViz

Como se ha explicado, inicialmente el proyecto se pretendía desarrollar en Qt, y para la captura y procesamiento de imágenes se tenía pensado utilizar OpenCV. Con el cambio de plataforma a ROS se buscaron alternativas ya existentes, encontrando un paquete que se encargaba de renderizar las imágenes obtenidas por una cámara estéreo y mostrándolas directamente en las Oculus Rift. Sin embargo, frente a la no disponibilidad de una cámara estéreo en la universidad y ante la decisión de utilizar Kinect para obtener las imágenes, hubo que buscar otras alternativas.

El uso de Kinect, como se ha especificado en el apartado donde se hablaba del uso de la misma en vez de una cámara estéreo, consiste en la obtención de dos imágenes a partir de una nube de puntos, generando dos puntos de vista diferentes de esta. Para realizar esto primeramente se requería la obtención de la información captada por la Kinect, de lo que se encargaba el paquete de ROS denominado OpenNI. Posteriormente,

se requería que esta información fuese mostrada correctamente por las Oculus Rift como ya se ha explicado. Para esto la única forma factible que se encontró fue la utilización del plugin de Oculus Rift para RViz desarrollado por David Gossow de Willow Garage, que mostraba el visor de RViz en las Oculus Rift, habilitando además el seguimiento de la cabeza del usuario.

Entre las consecuencias de esta elección se encuentran las siguientes:

Pros:

- Facilidad de implementación. Con el uso del plugin para RViz de Oculus Rift la carga de trabajo en este aspecto del proyecto se vio considerablemente reducida.

Contras:

- Falta de alternativas. Al usar RViz las alternativas disponibles pasaban por el uso del plugin disponible o por la creación de uno de similares características.
- Limitación de la interfaz de usuario a la proporcionada por RViz.

6.3.5 Uso de rqt_gui

Con el uso de RViz una de las cosas más importantes que se perdió fue la posibilidad de utilización de un interfaz de usuario personalizada. Por suerte, la existencia del paquete de ROS rqt_gui permite mostrar la interfaz de RViz dentro de otra interfaz, pudiendo añadir a esta otros módulos con información relevante sobre el sistema. Para el caso de este proyecto, hubo que crear un plugin para rqt_gui que mostrase la información de las Oculus Rift y permitiese un control de las funciones básicas como el reseteo del origen o el control manual.

Las consecuencias de la inclusión de rqt_gui en el proyecto fueron las siguientes:

Pros:

- Implementación de una GUI fácil de usar. Con el uso de rqt_gui se permitió el uso de RViz a la vez que se incluía una GUI personalizada con la información que se quería mostrar, de forma fácil y accesible.
- Diseño modular de la interfaz, siendo sencillo añadir futuras mejoras o nuevos módulos con más información.

Contras:

- Necesidad de aprendizaje extra. No se tenían conocimientos previos sobre el desarrollo de plugins para rqt_gui.

Capítulo 7. Pruebas y Resultados

En este apartado se van a detallar las pruebas realizadas sobre el prototipo desarrollado, haciendo un comentario crítico sobre los resultados obtenidos.

7.1 Pruebas realizadas

Pese a que este apartado está orientado a exponer las pruebas sobre el prototipo final, es conveniente antes señalar las pruebas que se fueron realizando a lo largo del desarrollo del proyecto.

7.1.1 Pruebas y resultados durante el desarrollo

Oculus Rift

Antes de comenzar con el desarrollo del proyecto se hicieron varias pruebas con el Oculus Rift DK1. Estas pruebas pasaban por la reproducción de demostraciones técnicas que permiten al usuario explorar mundos virtuales y comprobar el correcto funcionamiento del dispositivo. Una demostración que conviene destacar es la de Oculus World, que sitúa al usuario en una casa de la Toscana y le permite explorar libremente el terreno. La parte que hace a esta demo interesante es que ofrece la posibilidad de modificar en tiempo real valores de configuración de las Oculus Rift, como la distancia interpupilar o la predicción de movimiento de los sensores, y permite ver instantáneamente el efecto que tienen sobre la visualización.

Soporte y servomotores

Como se ha explicado en la parte de cambios en el proyecto del capítulo de desarrollo, una vez impreso el soporte inicial se realizaron pruebas sobre el mismo. Las primeras pruebas, antes de cambiar las cámaras estéreo por Kinect, las pasó satisfactoriamente, moviéndose correctamente según se movían las Oculus. Sin embargo, al introducir Kinect, con un peso muy superior, los servos que se estaban usando no tenían suficiente par y se decidió cambiarlos junto al soporte usado.

Con los servos Dynamixel y el esqueleto del robot Mini-Maggie también se realizaron las pruebas de estrés pertinentes, y a raíz de estas se decidió limitar el ángulo de inclinación del eje pitch. Esto se decidió así ya que Kinect al contar con una base que la eleva unos centímetros, al inclinarla hacia delante o hacia atrás el momento causado es mayor que si no estuviese elevada, ya que aumenta la distancia del eje de giro con el centro de masa de la cámara. Este momento producido hace que el trabajo que tengan que realizar los servos para moverla sea mayor en los puntos de máxima inclinación, por lo que se decidió limitar estos así como la velocidad para evitar altas demandas de corriente por parte de los servos, que podrían causar daños en los mismos. Con las limitaciones impuestas, el amperaje máximo que se ha llegado a registrar está comprendido entre 0,3 y 0,4 Amperios, mientras que los servos pueden llegar a trabajar hasta a 1,5 Amperios, por lo que la integridad de los mismos está asegurada.

ROS

En ROS se realizaron pruebas continuamente de todos los paquetes que se iban añadiendo al sistema. Para estas pruebas algunos de los comandos de ROS que se usaron son los siguientes:

```
rostopic list
```

Muestra una lista de todos los topics que están siendo publicados. Se usó para comprobar el correcto funcionamiento de los paquetes.

```
rostopic echo /topic_name
```

Se muestran los mensajes publicados en el topic especificado. Se usó para comprobar que los datos publicados de la posición de los servos eran correctos.

```
rostopic pub /topic_name topic_type data
```

Publica un mensaje del tipo especificado en un topic también especificado. Este comando se usó sobre todo para publicar las posiciones de los servos y poder realizar las pruebas de funcionamiento sin necesidad de las Oculus Rift.

```
rqt_graph
```

Muestra una GUI con un esquema de comunicaciones del sistema ROS, mostrando la relación entre nodos y topics. Es muy útil para ver de una forma gráfica si todo está conectado como se desea.

```
rosrun tf tf_echo tf1 tf1
```

Muestra la transformada entre dos tf frames. Antes de implementar el sistema de transformadas en el paquete desarrollado se probó que funcionaba correctamente gracias a este comando.

```
rosrun tf view_frames
```

Muestra un diagrama con todos los tf frames, además de información sobre estos como frecuencia de publicación o tamaño del buffer. Estos datos se usaron para determinar la velocidad de procesamiento exigido a la obtención de la transformada entre **/tf_oculus** y **/camera_link**.

Gracias a todos estos comandos se pudieron realizar pruebas sobre partes del sistema específicas, sin necesidad de tener todos los componentes conectados o todos los paquetes necesarios para el prototipo final ejecutados.

7.1.2 Pruebas sobre el prototipo final

Una vez desarrollado completamente el prototipo de telepresencia se procedió a realizar las pruebas sobre el mismo. Para estas pruebas se inició todo el sistema como viene descrito en el Manual de Usuario del Apéndice A y se vistió el Oculus Rift DK1. Se puede ver el dispositivo en funcionamiento en el vídeo que se ha publicado en YouTube llamado Desarrollo de un Sistema de Telepresencia Robótica con Oculus Rift. [22]

En el vídeo se puede ver en grande la imagen del usuario con las Oculus Rift puestas, así como el soporte de servos con Kinect montada. Por otra parte, en la esquina inferior derecha se puede ver lo mismo que se está mostrando en las Oculus Rift, y que por tanto está viendo el usuario. Encima de esta imagen se encuentra la captura de pantalla del ordenador en el que se está ejecutando el sistema, con la GUI proporcionada por rqt_gui con el plugin de RViz así como el de creación propia en pantalla.

En estas pruebas se comprobó el correcto funcionamiento de todos los dispositivos, así como de la interfaz de usuario creada. Dentro de la interfaz de usuario, al

comienzo del vídeo se puede comprobar el sistema de control automático, que recoge los datos de las Oculus Rift y los envía a los servos. Durante la ejecución de este modo es posible pulsar el botón de Reset, que reinicia el origen del eje yaw, aunque esto no se muestra en el vídeo. Por otra parte, al final del vídeo se puede ver como el usuario se quita el HMD y activa el modo manual, que permite controlar los servos deslizando unos indicadores de la interfaz.

7.2 Análisis de los resultados

Los resultados obtenidos en la prueba del prototipo final se pueden considerar generalmente satisfactorios. El sistema de seguimiento de la cabeza funciona correctamente y el desfase entre el movimiento de la cabeza del usuario y el de los servomotores no es significativo, siendo prácticamente inapreciable para el usuario su existencia. Así mismo, el funcionamiento de la GUI fue el adecuado a las especificaciones, enviando la información de forma continua en el modo automático y mostrando esa información en tiempo real en las barras inferiores para cada uno de los ejes. El botón de Reset se comportó como era esperado, estableciendo como referencia del eje yaw la posición del HMD en el momento de la pulsación, siendo esto importante debido a que tras usos prolongados el Oculus Rift DK1 tiende a descalibrarse. Del mismo modo, el modo de control manual funcionó también correctamente, enviando la información que se establecía o bien con los indicadores de las barras o bien introduciendo el ángulo a mano.

Por otro lado, no todos los resultados fueron buenos y también hay que destacar algunos fallos que se encontraron en el sistema. Dos de estos fallos ya se conocían y tienen que ver con Kinect, y es que debido a la baja resolución que ofrece la nube de puntos la imagen que se visualiza en las Oculus Rift no tiene una buena calidad. Del mismo modo, al no tratarse de una imagen estereoscópica pura, sino que se está utilizando una nube de puntos visualizada desde dos puntos de vista distintos, se puede apreciar falta de información en los bordes de los objetos. Por otro lado también se detectó un fallo con el que no se contaba, y fue la falta de potencia del ordenador portátil usado para las pruebas. En las pruebas previas realizadas durante el desarrollo del proyecto no se detectó ningún problema con ninguna parte del mismo, sin embargo, al iniciar todo el sistema se apreció que el movimiento de los servomotores perdía fluidez, del mismo modo que el frame rate de la nube de puntos obtenida en RViz se veía reducido. La única solución que se ha encontrado para esto sin tener que cambiar de ordenador ha sido reducir la resolución de la

imagen RGB obtenida por la Kinect desde el menú de calibración, cosa que en el resultado final no es apreciable debido a la baja resolución de la nube de puntos así como de las Oculus Rift. Otra forma de solucionarlo pasaba por limitar el frame rate de la Kinect desde la calibración, reduciendo la carga del procesador significativamente. Aunque estas soluciones han funcionado relativamente bien, lo aconsejable sería el uso de un ordenador más potente que se encargase de todos los procesos sin tener que sacrificar la calidad del sistema.

Por último, cabe destacar un problema que surgió con el uso del plugin de Oculus para RViz. Este plugin ofrece la posibilidad de visualizar en las Oculus el simulador 3D de RViz, donde está situada la nube de puntos obtenida por Kinect, pudiendo variar el punto de vista moviendo la cabeza en la dirección deseada. El problema es que la nube de puntos de la Kinect se encuentra fijada en el espacio, y al mover la cabeza la visión de RViz cambia y la nube de puntos deja de cubrir el centro del campo de visión del usuario. Este problema se ha intentado solucionar sin éxito, por lo que se ha publicado una pregunta abierta en la comunidad de ROS así como se ha escrito un correo electrónico al creador del plugin para preguntarle por una solución, pero en ambos casos se continúa esperando respuesta. En cualquier caso, con las Oculus Rift puestas se puede apreciar el error aunque no por ello el sistema pierde utilidad, ya que no se pierde la visión del entorno en ningún momento, lo único que obliga a que el usuario redirigir la mirada a diferentes puntos de la pantalla del HMD.

Capítulo 8. Conclusiones y Futuras Mejoras

En este apartado se explicarán las conclusiones obtenidas a raíz del desarrollo del proyecto y de las pruebas realizadas. Además se hará un análisis de lo que ha supuesto la realización de este proyecto a nivel personal.

Por otra parte, se hablará sobre las futuras mejoras que se proponen para el sistema de telepresencia desarrollado.

8.1 Conclusiones

Tras la realización del proyecto se puede concluir que se ha sido capaz de desarrollar un sistema de telepresencia robótica eficiente y abierto a su implementación en cualquier robot que cuente con los requisitos, cumpliendo de esta forma todos los objetivos planteados al comienzo del proyecto.

El sistema desarrollado tiene como puntos fuertes el precio, que es mucho más asequible que los sistemas de telepresencia basados en HMDs que existían hasta ahora, y la alta compatibilidad, ya que puede ser aplicado a una gran base de robots que cuentan con Kinect como sistema de visión. Por el contrario, sus puntos débiles son la mala calidad de imagen, debido por una parte a que las Oculus Rift son de momento un prototipo orientado al desarrollo que está en constante mejora y por otra a la baja resolución que ofrece la nube de puntos de Kinect, y el fallo que no permite mantener centrada la nube de puntos.

A nivel personal la realización del proyecto ha sido muy gratificante en varios aspectos, aunque no por ello sencillo. En primer lugar el trabajo con un dispositivo en fase de desarrollo y con tecnología puntera dentro de su campo como Oculus Rift ha sido toda una suerte, y el estudio de su documentación y SDK me ha acercado a lo que en un futuro me puede deparar la profesión de ingeniero. El mayor reto que se me ha presentado ha sido el desconocimiento de cada uno de los componentes utilizados, desde las ya citadas Oculus Rift, que al tratarse de un kit de desarrollo que aún no está a la venta para el público

general no cuenta con demasiada información, hasta ROS, que se ha tenido que aprender su uso desde cero. Dentro de las cosas positivas que me llevo está este aprendizaje de ROS, que si bien no ha resultado sencillo será sin duda de utilidad de cara al futuro, ya que opino que es un sistema potentísimo que ofrece una gran plataforma para el desarrollo de robots.

8.2 Futuras mejoras

El sistema de telepresencia desarrollado a lo largo de este proyecto no deja de ser un prototipo, cuyo objetivo de crear un sistema basado en los nuevos dispositivos de realidad virtual está ampliamente logrado. Sin embargo, como todo prototipo está sujeto a mejoras, y a nivel personal pienso que un buen punto de partida para continuar con el desarrollo sería empezar por los siguientes puntos:

- Solución de los problemas con el plugin de Oculus para RViz.
- Mejora de calidad del HMD. El Oculus Rift DK1 como prototipo que es ofrece una calidad de imagen muy baja, por lo que de cara al futuro lo ideal sería adquirir e implementar el sistema desarrollado en los nuevos HMDs, como el Oculus Rift DK2 o la futura versión comercial de Oculus VR.
- Mejora de calidad de la imagen representada. Como se detallaba en los puntos débiles del proyecto, la calidad de la nube de puntos que ofrece Kinect es muy baja. Para mejorar esto las posibilidades pasan por cambiar el dispositivo capturador o bien por una nueva Kinect, que ha sido lanzada al mercado este mismo año y ofrece una calidad mucho mayor, o por el uso de cámaras estéreo de alta definición, aunque en este segundo caso el proyecto requeriría una profunda reestructuración.
- Mejora de la interfaz de usuario. La interfaz de usuario desarrollada en este proyecto cumple básicamente con su función, pero podría ser mejorada con la inclusión de información extra del resto de sensores del robot en el que está funcionando el sistema de telepresencia. Así mismo, se podría añadir compatibilidad del control manual con periféricos externos como joysticks, ya que permitiría la realización de dicho control sin necesidad de quitarse el HMD.
- Comunicación inalámbrica con el robot. El sistema desarrollado actualmente funciona completamente sobre un mismo ordenador, pero lo normal es que el

sistema robótico se encuentre a una distancia considerable del usuario que lo está teleoperando, por lo que sería necesario la implementación de un sistema de comunicación inalámbrica.

- Manejo de otras funciones del robot. En el proyecto se ha desarrollado un sistema de telepresencia, pero lo ideal sería que este sistema se usase conjuntamente a otro de teleoperación, aunque esto ya sería algo más específico del tipo de robot utilizado.

Bibliografía

[1] S. Epelbaum, «Historia de la Estereoscopía y sus Aplicaciones,» Diciembre 2010. [En línea]. Available: http://www.sao.org.ar/index.php/archivos-de-oftalmologia/ediciones-anteriores/59-archivos-de-oftalmologia/ediciones-anteriores/volumen-81-numero-02/212-historia-de-la-estereoscopia-y-sus-aplicaciones. [Último acceso: Julio 2014].

[2] G. Baranda, «Historia cronológica de la Realidad Virtual,» 25 Junio 2012. [En línea]. Available: http://gabrielbaranda.blogspot.com.es/2012/06/historia-cronologica-de-la-realidad.html. [Último acceso: Julio 2014].

[3] J. «La realidad virtual,» 11 Junio 2009. [En línea]. Available: http://www.javi.it/r_virtual.html. [Último acceso: Julio 2014].

[4] C. Ortigueira España, M. Reigosa García, M. Rodríguez Fernández, C. Santamaría González y J. Veiga Fachal, «Realidad Virtual,» [En línea]. Available: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/Realidad%20Virtual/web/introduccion.html. [Último acceso: Julio 2014].

[5] T. Mazuryk y M. Gervautz, «Virtual Reality - History, Applications, Technology and Future,» [En línea]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7849. [Último acceso: Julio 2014].

[6] Wikipedia, «Virtuality (gaming),» [En línea]. Available: http://en.wikipedia.org/wiki/Virtuality_(gaming). [Último acceso: Julio 2014].

[7] G. Brunner, "Virtually perfect: The past, present, and future of VR," 2 Mayo 2014. [Online]. Available: http://www.extremetech.com/gaming/181454-virtually-perfect-the-promise-of-virtual-reality. [Accessed Julio 2014].

[8] S. Burke, «The History of Virtual Reality & The Future: Rift, Omni, STEM, castAR,» 20 Octubre 2013. [En línea]. Available: http://www.gamersnexus.net/guides/1208-history-of-virtual-reality-and-future. [Último acceso: Julio 2014].

[9] F. Álvarez, «5 curiosos gadgets antiguos de realidad virtual,» 2 Marzo 2014. [En línea]. Available: http://gizmologia.com/2014/03/gadgets-antiguos-realidad-virtual. [Último acceso: Julio 2014].

[10] M. Scgnipper, A. Robertson, M. Zelenko, E. Hamburguer, C. Mazza, S. Thonis, C. Newton y S. O'Kane, «The Rise and Fall and Rise of Virtual Reality,» [En línea]. Available: http://www.theverge.com/a/virtual-reality. [Último acceso: Julio 2014].

[11] J. Popa, «Oculus Rift: DK1 vs DK2,» 10 Agosto 2014. [En línea]. Available: http://in2gpu.com/2014/08/10/oculus-rift-dk1-vs-dk2/. [Último acceso: Agosto 2014].

[12] E. Nuño ortega y L. Basañez Villaluenga, «Teleoperación: técnicas, aplicaciones, entorno sensorial y teleoperación inteligente,» Abril 2004. [En línea]. Available: http://www.ual.es/~rgs927/documents/teleoperaci%C3%B3n.pdf. [Último acceso: Agosto 2014].

[13] T. B. Sheridan, Telerobotics, Automation, and Human Supervisory Control, MIT, 1992.

[14] J. Marescaux, «Estado actual de la cirugía. Cirugía robótica y telecirugía,» Julio-Agosto 2013. [En línea]. Available: http://www.redalyc.org/pdf/662/66228318001.pdf. [Último acceso: Agosto 2014].

[15] ifixit, «Oculus Rift Teardown,» [En línea]. Available: https://es.ifixit.com/Teardown/Oculus+Rift+Teardown/13682. [Último acceso: Agosto 2014].

[16] Lacie, «Persistencia de la Imagen,» [En línea]. Available: https://www.lacie.com/es/support/faq/faq.htm?faqid=10508. [Último acceso: Agosto 2014].

[17] M. e. blix, «Oculus Rift Simulator,» 12 Febrero 2014. [En línea]. Available: http://vr.mkeblx.net/oculus-sim/. [Último acceso: Agosto 2014].

[18] ifixit, «Microsoft Kinect Teardown,» [En línea]. Available: https://www.ifixit.com/Teardown/Microsoft+Kinect+Teardown/4066. [Último acceso: Agosto 2014].

[19] Pensamientos Computables, «Kinect: Como funciona su 3D body tracking,» 23 Noviembre 2010. [En línea]. Available: http://www.pensamientoscomputables.com/entrada/kinect/como-funciona/caracteristicas/microsoft/xbox-360/3d-body-tracking. [Último acceso: Agosto 2014].

[20] P. Iñigo Blasco, «ROS: ¿El nuevo estándar de facto en robótica?,» 17 Abril 2010. [En línea]. Available: http://geus.wordpress.com/2010/04/17/ros-%C2%BFel-nuevo-estandar-de-facto-en-robotica/. [Último acceso: Agosto 2014].

[21] E. Berger , K. Conley, J. Faust, T. Foote, B. Gerkey, J. Leibs, M. Quigley y R. Wheeler, «ROS wiki,» [En línea]. Available: http://wiki.ros.org/ROS. [Último acceso: Agosto 2014].

[22] E. Ruiz-Medrano García , «Desarrollo de un Sistema de telepresencia Robótica con Oculus Rift,» 16 Septiembre 2014. [En línea]. Available: http://youtu.be/hQpHhEK1-4g. [Último acceso: Septiembre 2014].

Apéndice A. Manual de Usuario

En este apartado se van a detallar los pasos que se deben seguir para la correcta puesta en marcha del sistema desarrollado, siendo estos instalación, configuración, calibración e inicialización. Además se explicarán las funciones de la interfaz gráfica o GUI implementada.

A.1 Requisitos

Los componentes con los que se debe contar si se va a instalar el sistema de telepresencia son los siguientes:

- Ordenador de gama media con sistema operativo Ubuntu 12.04.
- ROS Hydro instalado.
- Oculus Rift DK1.
- Kinect o similar.
- Soporte de servos Dynamixel de dos ejes. Si los servos son de otro fabricante se deberá estudiar la forma en la que funcionan estos y enviarles la información de las Oculus Rift como sea requerido.

Alternativamente a estos requisitos, si se cuenta con un robot que cuente con un sistema de visión basado en Kinect o similares, dos ejes de movimiento de la cabeza y esté controlado por ROS Hydro, se podría adaptar el funcionamiento fácilmente para el sistema de telepresencia.

A.2 Instalación y configuración

En este apartado se van a detallar los paquetes de ROS que es necesarios instalar para la ejecución del sistema. Del mismo modo se explicará cómo deben ser configurados

cuando sea necesario. Como se ha especificado en el apartado de requisitos, se parte de la idea de que el usuario posee un ordenador con Ubuntu 12.04 y ROS Hydro.

A.2.1 Oculus Rift DK1

Instalación

La instalación del Oculus Rift DK1 es muy simple y solo requiere que los conectores se enchufen correctamente, es decir, HDMI y USB al ordenador que se vaya a usar y cable de alimentación a la red eléctrica. Alternativamente al cable HMDI, si el ordenador donde se vaya a instalar el sistema no cuenta con este tipo de entrada también se puede usar un cable DVI. El Oculus Rift DK1 no requiere la instalación de ningún driver para funcionar, por lo que con configurarlo correctamente estará listo para ser usado.

Configuración

Para que Oculus Rift funcione correctamente con nuestro sistema se requiere que la configuración del display sea como segundo monitor y con una resolución de 1280 x 800 píxeles. Para conseguir esto en Ubuntu hay que dirigirse al administrador de monitores, al que se puede acceder desde el menú de opciones, que está situado por defecto arriba a la derecha de la pantalla e indicado con un símbolo de engranaje. Una vez desplegada la ventana del administrador de monitores (Figura 45) lo primero que hay que comprobar es que la opción de espejar monitores esté desactivada. Una vez hecho esto, veremos que aparecen en el gráfico superior los displays disponibles, siendo uno el principal de nuestro ordenador y otro el de las Oculus Rift, denominado como OVR. Con el display OVR seleccionado, elegimos dentro del campo de resolución una de 1280 x 800 píxeles. Adicionalmente, conviene asegurarse de que la posición del lanzador está configurada para aparecer únicamente en el monitor principal de nuestro ordenador.

Una vez hecho esto ya se tiene configurado correctamente el Oculus Rift DK1 para ser usado, solo faltaría realizar las calibraciones pertinentes para mejorar la experiencia del usuario.

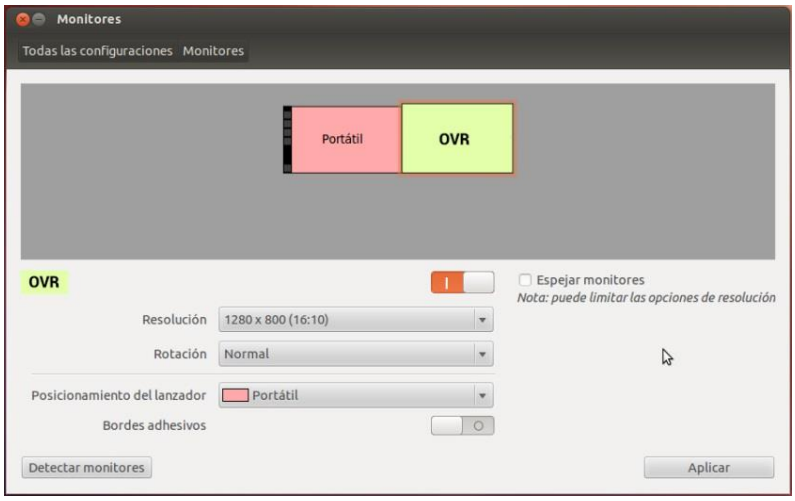

Figura 45: Configuración de Oculus Rift como segundo monitor.

Calibración

La calibración del Oculus Rift DK1 pasa por dos fases: primero comprobar la correcta calibración de los sensores, y en segundo lugar el ajuste del IPD (distancia interpupilar) del usuario. Para ambos casos el SDK cuenta con aplicaciones que ayudan a la correcta puesta a punto del dispositivo., y que se encuentra dentro de la carpeta Tools del mismo bajo el nombre de **OculusConfigUtility**. Una vez ejecutada esta aplicación aparecerá una nueva ventana como la siguiente:

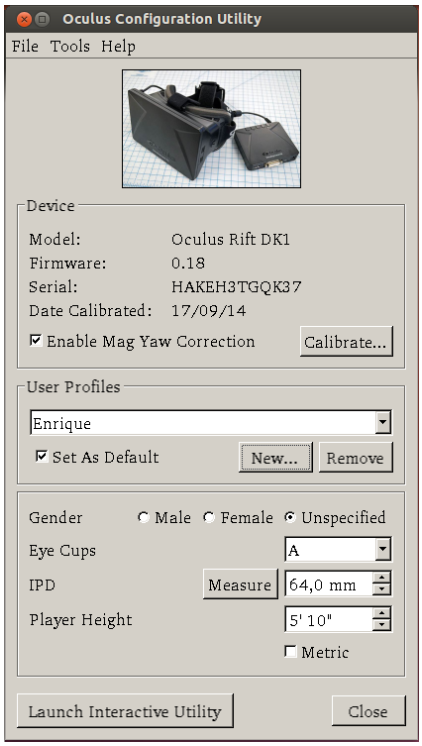

Figura 46: Herramienta de configuración de Oculus.

La primera opción a la que debemos mostrar atención es la de calibrar los sensores, a la que se accede pulsando el botón **Calibrate**. Una vez pulsado se nos desplegará una nueva ventada (Figura 47) en la que se nos da una serie de instrucciones sobre cómo calibrar correctamente el dispositivo, siendo estas:

- Colocar el dispositivo cerca de nuestra cabeza, como si fuesemos a usarlo.
- Pulsar el botón Start.
- Girar el dispositivo en todas direcciones hasta que se llene la barra de progreso.
- Guardar la calibración pulsando el botón Save.

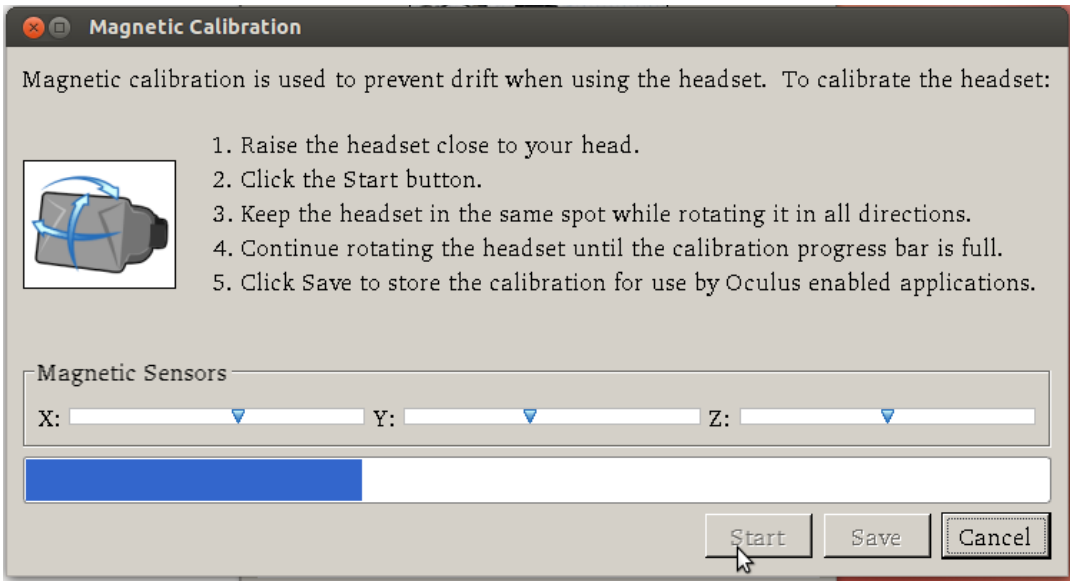

Figura 47: Herramienta de calibración de los sensores.

La siguiente opción que nos ofrece la herramienta de configuración es la creación de perfiles de usuario personalizados. Esto es así porque cada usuario suele tener unas características físicas distintas, por lo que el programa nos deja ajustar una serie de opciones para cada uno. Estas opciones son género, lentes usadas en el HMD, IPD y altura. En cuanto a estas opciones, el género no tiene relevancia, las lentes usadas pueden diferir si un usuario tiene problemas de miopía y utiliza el HMD sin gafas o lentillas puestas, el IPD se utiliza para mostrar correctamente las imágenes dependiendo del usuario, y por último la altura se usa para la colocación de la cámara en los mundos virtuales y que el usuario no tenga la sensación de ser demasiado grande o demasiado pequeño, pero en nuestro caso esto no nos influye ya que nuestro sistema está limitado a la altura que tenga el robot a teleoperar.

Entre todas estas opciones la más importante es el ajuste del IPD, ya que la distancia interpupilar entre usuarios suele cambiar y una mala configuración puede conducir a náuseas y mareos. Para medir esta distancia Oculus Rift proporciona una herramienta con este fin, a la que se accede pulsando el botón Measure. Una vez pulsado el botón nos tendremos que poner el HMD y seguir una serie de instrucciones que van apareciendo en pantalla. Las pruebas realizadas por el programa consisten en determinar el límite del campo de visión de cada uno de nuestros ojos, para lo que se muestra una línea verde que debemos ir moviendo hasta perder de vista.

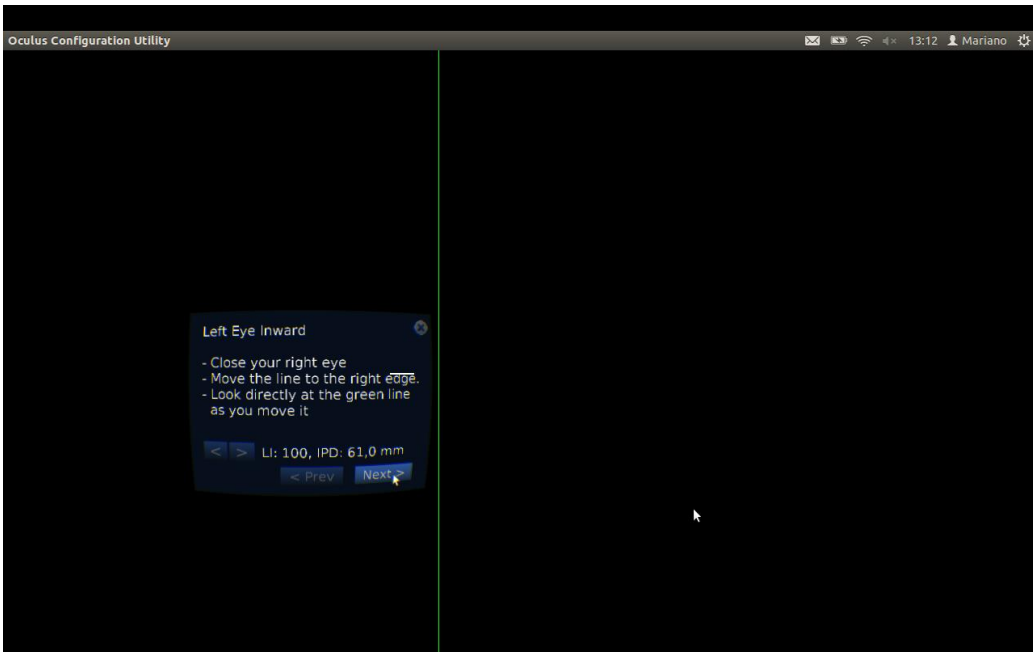

Figura 48: Herramienta de medición del IPD.

Una vez terminada la calibración y ajuste de IPD, se puede decir que ya tenemos el Oculus Rift DK1 listo para funcionar en el sistema de telepresencia.

A.2.2 Kinect y librería OpenNI

Instalación

Para conectar Kinect al ordenador esta cuenta con un cable USB, pero requiere alimentación externa para funcionar. Una vez conectada correctamente, necesitamos instalar los drivers para ROS. En este proyecto se van a utilizar los drivers proporcionados por OpenNi, y para su instalación hay que ejecutar los siguientes comandos:

```
sudo apt-get install ros-hydro-openni-camera ros-hydro-openni-launch
```

Una vez instalados los drivers correctamente, para poder utilizar la Kinect solo queda utilizar el siguiente comando para iniciar los controladores:

```
roslaunch openni_launch openni.launch
```

Calibración

Es posible que al usar la Kinect nos demos cuenta de que no funciona correctamente, y esto puede ser así porque no esté correctamente calibrada. En nuestro caso para calibrarla se ha usado la herramienta dynamic_reconfigure, que nos permite calibrar Kinect a la vez que se están ejecutando los drivers, y que se puede abrir con el siguiente comando:

```
rosrun dynamic_reconfigure reconfigure_gui
```

Una vez ejecutado el comando anterior, se abrirá una ventana donde deberemos elegir en el menú desplegable la opción **/camera/driver**, los que nos ofrecerá varias opciones de calibración como se muestra en la siguiente imagen:

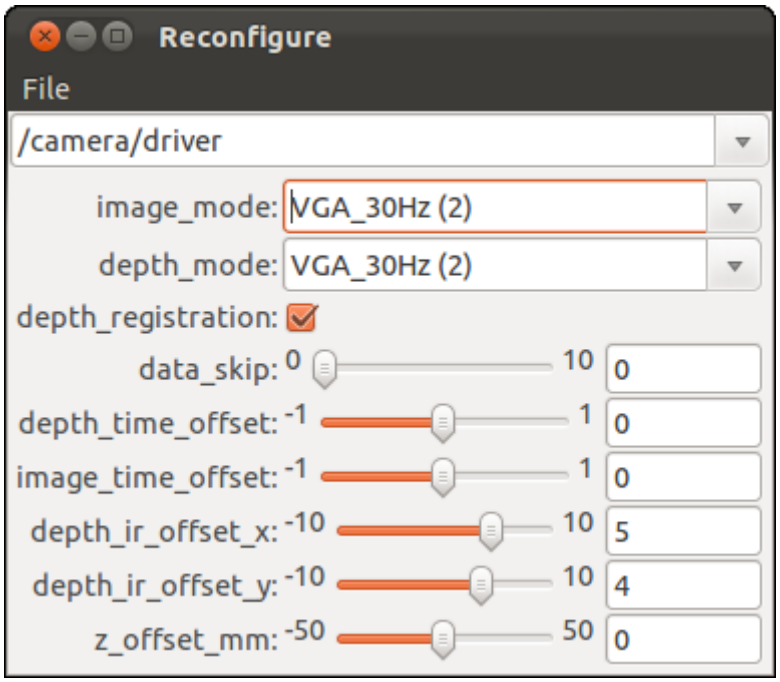

Figura 49: Herramienta de calibración de Kinect.

Para calibrar la Kinect basta con ir modificando los indicadores de offset de las diferentes opciones hasta lograr una imagen correcta, lo que por ejemplo podemos comprobar mostrando la imagen de disparidad a través del siguiente comando:

```
rosrun image_view disparity_view image:=/camera/depth_registered/disparity
```

Además de calibrar los offsets, la herramienta de dynamic_reconfigure nos ofrece otras opciones interesantes, como son **image_mode**, **depth_mode** y **data_skip**. Las dos primeras nos permiten cambiar el modo en el que se procesan las imágenes de profundidad y RGB, pudiendo modificar el formato y la resolución. Por otra parte, la opción data_skip nos permite desechar información de tal modo que se limite el frame rate de las imágenes obtenidas, y por tanto la cantidad de información que debe procesar el ordenador. Estas opciones conviene modificarlas si se está implementando el sistema de telepresencia en un ordenador que no tenga suficiente potencia.

A.2.3 RViz y plugin de Oculus Rift

Instalación

En el caso de que la distribución de ROS Hydro instalada no incluya RViz, habrá que instalarlo de forma manual. Para ello hay que ejecutar el siguiente comando en la consola:

```
sudo apt-get install ros-hydro-RViz
```

Una vez instalado correctamente RViz hay que añadir el plugin de Oculus Rift. Este plugin tiene como requisito que el SDK de Oculus Rift esté instalado en ROS, cosa que se puede hacer a través de este comando:

```
sudo apt-get install ros-hydro-oculus-sdk
```

Una vez hecho esto se puede proceder a la instalación del plugin con el siguiente comando:

```
sudo apt-get install ros-hydro-RViz-plugins
```

Con esto ya tendremos RViz y el plugin de Oculus instalados correctamente, solo tendríamos que configurarlo todo correctamente para que funcionase.

Configuración

Para configurar RViz y que funcione con nuestro sistema de telepresencia primero debemos haber configurado correctamente las Oculus Rift así como la Kinect y sus drivers, que deben estar en funcionamiento. Una vez hecho esto se puede iniciar RViz con el siguiente comando:

```
rosrun rviz rviz
```

Al iniciar RViz nos encontraremos con la interfaz que viene establecida por defecto. En la barra lateral izquierda tendremos una lista de los displays que están activos, y aquí es donde tenemos que añadir tanto la nube de puntos obtenida por Kinect como el display de Oculus Rift. Para ello pulsamos el botón **Add** que hay debajo de la lista, y seleccionamos **PointClod2** para el caso de Kinect. Una vez hecho esto desplegamos las opciones de **PointCloud2** haciendo click encima, y seleccionamos en **Topic** la opción **/camera/deph_registerd/points**, lo que nos dará como resultado una imagen en profundidad en blanco y negro. Para aplicar la imagen de color obtenida por la cámara RGB basta con seleccionar la opción **RGB8** en **Color Transformer**.

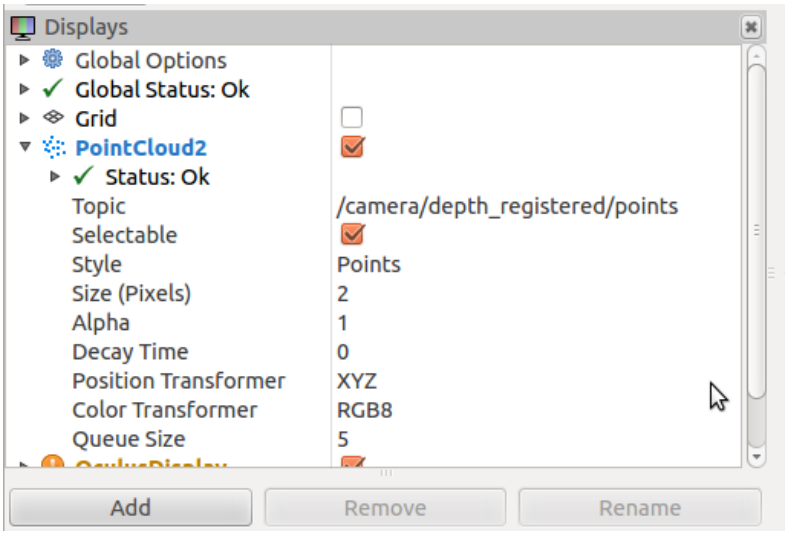

Figura 50: Opciones de PointCloud2.

Por otra parte, para añadir el display de Oculus Rift seguimos el mismo procedimiento y pulsamos en el botón **Add**, y en la nueva ventana elegimos **OculusDisplay**. Una vez añadido hacemos click encima para desplegar sus opciones. Si hemos configurado correctamente las Oculus Rift como segundo monitor y con la resolución especificada, la opción **Render to Oculus** estará activa y la pulsaremos. La otra

opción importante que debemos modificar es la del **tf frame**, que debemos especificar que queremos que se publique marcando la opción **Publish tf** y debemos modificar el nombre del mismo a **tf_oculus**. Además de estas opciones, el plugin de Oculus para RViz ofrece otras opciones que pueden resultar interesantes. Una de ellas es la posibilidad modificar el tiempo de las predicciones de los sensores, que conviene hacerlo si se aprecia desfase entre el movimiento de la cabeza y el de la cámara de las Oculus Rift. Otra opción es la de modificar el **Offset** de la imagen mostrada en las Oculus, que habrá que cambiarlo si la nube de puntos se ve demasiado grande o demasiado pequeña, o simplemente descentrada.

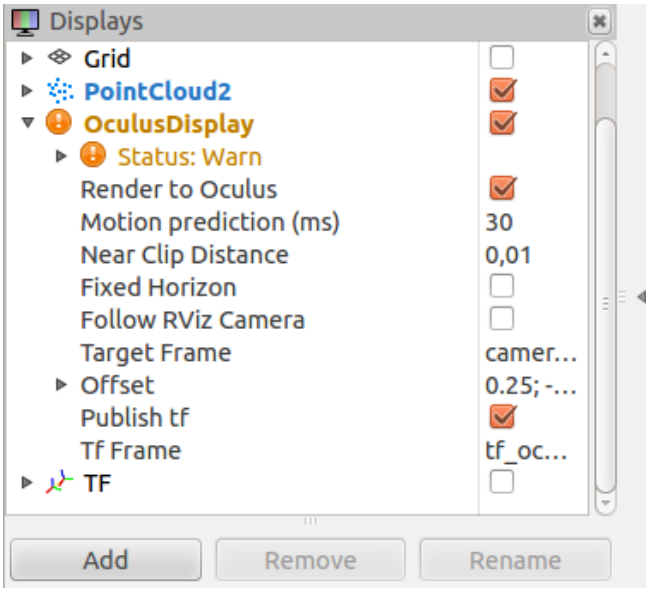

Figura 51: Opciones de OculusDisplay.

Como se puede apreciar en la Figura 49, al añadir un display de tipo **OculusDisplay** a RViz en el estado nos aparece un aviso. Esto es debido a que el plugin ofrece la posibilidad de usar un magnetómetro compatible con Oculus Rift para mejorar la medida de los sensores, pero no es necesario para el funcionamiento del sistema.

Una vez seguidos estos pasos, se puede guardar la configuración de RViz establecida y cerrar el programa, que más tarde incluiremos en la interfaz del sistema gracias a **rqt_gui**.

A.2.4 Servos Dynamixel

Para este manual se va a considerar que los servos usados son los mismos que en la realización del proyecto, es decir, servos Dynamixel AX-12A. Para el uso de cualquier otro

tipo de servos se deberá estudiar la forma en la que estos son controlados y ajustar el paquete desarrollado en este proyecto a cada caso.

Instalación

La disposición física de los servos debe ser de tal forma que uno se encargue de realizar el movimiento del eje pitch y otro del eje yaw, con la cámara encima de ellos. Además, los servos se deberán alimentar con un voltaje de 9 Voltios, y serán conectados al ordenador vía la placa controladora **USB2Dynamixel**.

Una vez conectados al ordenador, es necesario instalar el paquete de ROS encargado del control de los servos denominado dynamixel_controllers, que se hace con el siguiente comando:

```
sudo apt-get install ros-hydro-dynamixel-controllers
```

Con este paquete instalado el siguiente paso es descargar el paquete creado en este proyecto para el control de los dos servos.

Configuración

Este paquete viene configurado con unos valores que han servido para el soporte en el que se han llevado a cabo las pruebas, pero conviene modificarlos para que se ajusten mejor a las características del sistema utilizado. Para ello basta con editar los archivos .yaml que contiene el paquete, por ejemplo a continuación se muestra el archivo yaw.yaml:

```
yaw_controller:
    controller:
        package: dynamixel_controllers
        module: joint_position_controller
        type: JointPositionController
    joint_name: yaw_joint
    joint_speed: 0.65
    motor:
        id: 4
```

```
        init: 512

        min: 350

        max: 660
```

Entre estas opciones las que se pueden modificar con objetivo de ajustarlas mejor al sistema en las que se está aplicando el paquete son:

- Joint_speed: velocidad a la que se moverá el motor.
- Id: número identificador del motor. En nuestro caso se usaban los servo motores 4 y 5 del soporte, pero si nuestro motor tiene otra ID basta con modificar este número. Por el contrario, si queremos modificar la ID de uno de nuestros motores se puede usar el siguiente comando:

```
rosrun dynamixel_driver change_id.py 1 2
```

Siendo 1 la ID del motor que se quiere cambiar y 2 la nueva ID que se quiere asignar. Las IDs de los servos Dynamixel vienen de fábrica definidas como 1.

- Init: posición inicial del servo. Las posiciones posibles va desde 0 a 1023.
- Min: posición mínima del servo. En nuestro caso se ha ajustado a unos -45 º desde la posición inicial para el eje yaw y a -33º para el pitch.
- Max: posición máxima del servo. En nuestro caso se ha ajustado a unos 45 º desde la posición inicial para el eje yaw y a 33º para el pitch.

Con estas modificaciones ya se tendría listo el sistema para ser ejecutado, a lo que se procede con los siguientes comandos, siendo "my_dynamixel_tutorial" el nombre del paquete:

```
roslaunch my_dynamixel_tutorial controller_manager.launch

roslaunch my_dynamixel_tutorial pitch_controller.launch

roslaunch my_dynamixel_tutorial yaw_controller.launch
```

De esta forma los servos estarán listos para recibir los datos de posición en el topic correspondiente, siendo estos:

```
/yaw_controller/command

/pitch_controller/command
```

A.2.5 Rqt GUI

Instalación

La instalación de rqt se realiza con la ejecución del siguiente comando de consola:

```
sudo apt-get install ros-hydro-rqt ros-hydro-rqt-common-plugins
```

Una vez instalado correctamente, hay que descargar el paquete que contiene el plugin desarrollado en este proyecto, denominado rqt_servo_oculus. Para iniciar

Configuración

Para acceder a la interfaz de rqt_gui basta con utilizar el siguiente comando:

```
rqt
```

Esto abrirá una ventana vacía a la que tendremos que añadir los plugins que queramos utilizar. Para nuestro sistema de telepresencia necesitamos añadir dos plugins: RViz y el plugin de creación propia Oculus Servo Plugin. Para añadirlos hay que dirigirse al menú de Plugins, donde encontraremos a RViz dentro del submenú Visualization, mientras que Oculus Servo Plugin no se encuentra enmarcado en ningún submenú. La forma en la que se dispongan los plugins en la ventana puede ser la preferida por el usuario, pero se recomienda que se sitúen como en la Figura 52.

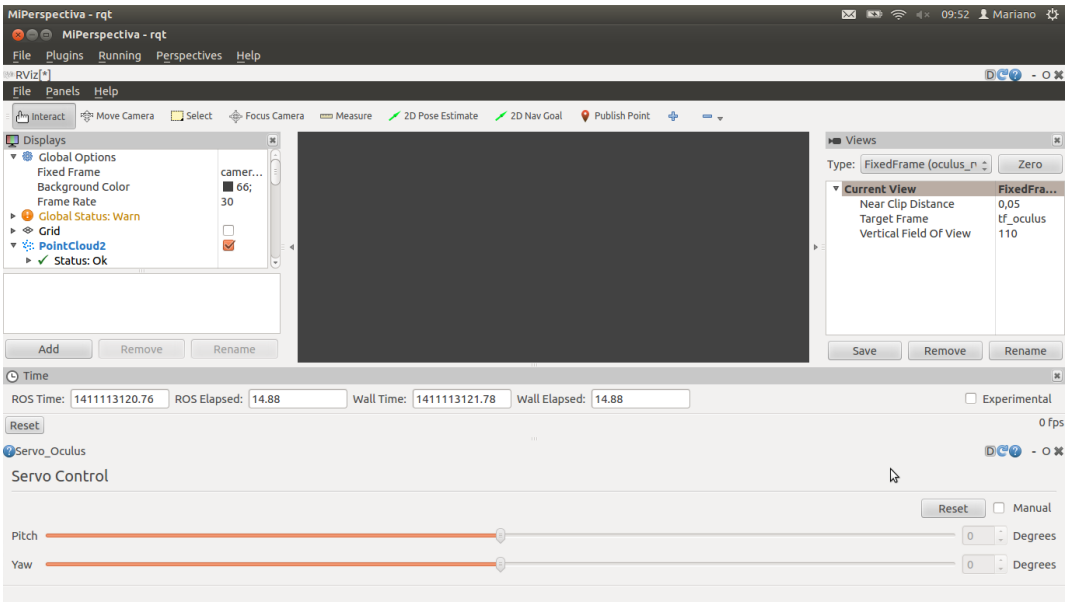

Figura 52: Interfaz de rqt con los plugins de RViz y Oculus Servo.

Por último, dentro de la interfaz de RViz deberemos cargar la perspectiva que hemos guardado anteriormente y que contiene los displays de las Oculus y Kinect. Una vez hecho esto ya se tiene el sistema instalado y configurado para ser usado, lo que se explica a continuación en el apartado Modo de Empleo.

A.2.6 Archivo launch

Configuración

Este archivo técnicamente no requiere ninguna configuración especial, sino que únicamente hay que descargarlo y situarlo en un lugar conocido. En nuestro caso se ha situado este archivo dentro de catkin_ws/launch y se le ha dado el nombre de TeleOculus.launch. Debido a la simplicidad de este archivo, en vez de descargarlo también se puede crear uno propio utilizando un documento de texto vacío y pegando el siguiente código:

```
<launch>

   <include file="$(find openni_launch)/launch/openni.launch" />

   <include file="$(find my_dynamixel_tutorial)/controller_manager.launch" />

   <include file="$(find my_dynamixel_tutorial)/yaw_controller.launch" />

   <include file="$(find my_dynamixel_tutorial)/pitch_controller.launch" />

</launch>
```

A.3 Modo de Ejecución

En este apartado se va a explicar el modo de funcionamiento del sistema de telepresencia creado, explicando cada una de sus funciones así como los pasos que hay que realizar para ponerlo en funcionamiento.

Para poner en marcha el sistema creado lo primero que es necesario es inicializar los controladores de todos los dispositivos usados. Para ello lo primero que hay que hacer es ejecutar el núcleo de ROS con el siguiente comando:

```
roscore
```

Acto seguido debemos navegar en la consola hasta la carpeta donde hayamos almacenado el archivo launch encargado de iniciar todos los controladores, que en nuestro caso es **catkin_ws/launch**, e iniciar el archivo con este comando:

```
roslaunch TeleOculus.launch
```

Una vez se terminen de ejecutar todos los nodos, hay que abrir la interfaz gráfica de usuario de rqt, lo que se consigue con la siguiente acción:

```
rqt
```

Al abrirse la ventana de rqt, el sistema ya está listo para funcionar y se inicia con el sistema de telepresencia en modo automático. Antes de ponerse las Oculus Rift conviene establecer el centro de nuestra vista, para ello sólo hay que colocar el HMD en la misma dirección que nuestra cabeza mirando al frente y pulsar en el botón **Reset**. De esta forma se establecerá como origen este punto y los movimientos del eje yaw registrados ser efectuarán respecto a este.

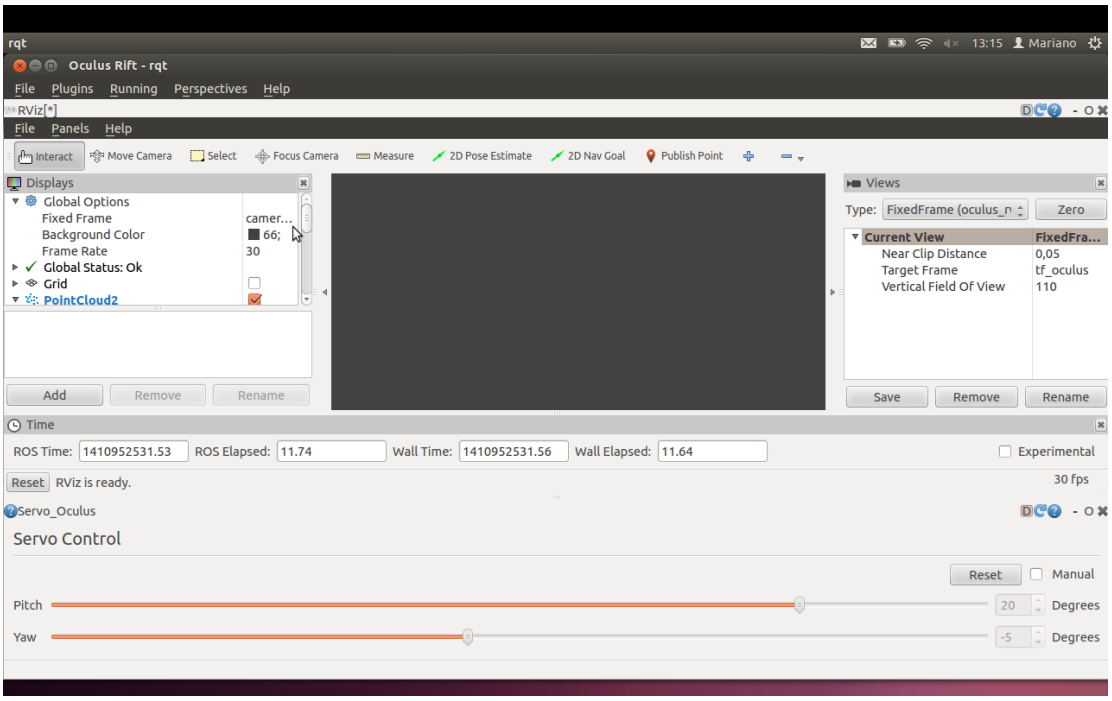

Figura 53: Sistema funcionando en modo automático.

Mientras que el sistema está funcionando en el modo automático, la información de los sensores de las Oculus Rift se envía directamente al soporte de servos. La posición del HMD y por tanto de los servos se puede consultar de forma gráfica en las barras de la parte inferior de la interfaz, así como el valor numérico en grados respecto al origen.

Por otra parte la interfaz ofrece la posibilidad de controlar los servomotores de forma manual. Para ello únicamente hay que marcar la casilla que indica **Manual**, lo que desactiva el movimiento de los servos respecto a la posición de las Oculus, y pasa a realizarse mediante el deslizamiento de las barras de cada eje. Adicionalmente se puede establecer la posición de los servos con las casillas donde se muestra el valor numérico, que permite su modificación de grado en grado o con la introducción de un valor deseado. Para volver al modo automático sólo hay que volver a desmarcar la casilla de **Manual**, lo que desactivará el control de los servos mediante la interfaz.

Acrónimos

HMD: Head Mounted Display, casco de realidad virtual.

FOV: Field Of View, campo de visión.

LCD: Liquid Crystal Display

3D: tres dimensiones

LED: Light Emiting Diode

Full HD: resolución de 1980 x 1090 píxeles

4K: resolución de 4096 × 2160 píxeles

DK: Developer Kit

SDK: Software Developer Kit

IMU: Inertial Media Unit

ROS: Robot Operating System

CMOS: Complementary metal-oxide-semiconductor