Máster en Sistemas Inteligentes

Trabajo Final de Máster

Interfaz de realidad virtual para el manejo de robots submarinos

Autor:

Marcos de la Cruz Soler

Tutor:

Raúl Marín Prades

Co-Tutor:

Pedro José Sanz Valero

Curso académico 2019/2020

Contenido

Resumen

La necesidad de realizar intervenciones en entornos submarinos se ha incrementado en los últimos años, pero debemos tener en cuenta que, a día de hoy, la dificultad de algunas misiones impide que sean realizadas por *AUVs* (Autonomous Underwater Vehicles). Por ello, la solución actual es el uso de vehículos operados de forma remota (*ROV*, Remote Operated Vehicle). Estos robots están operados mediante interfaces realmente complejas que permiten la interacción entre el humano y el robot (HRI). En consecuencia, hay que reducir en la medida de lo posible la complejidad de estos sistemas para reducir el estrés que se genera en el operador. Este proyecto forma parte del *TWIN roBOTs* que es llevado por el *IRS Lab*, que consiste en hacer que dos robots actúen de forma cooperativa para realizar intervenciones submarinas. En este trabajo presentamos una interfaz de realidad virtual, que consiste en la mejora de la que fue presentada como trabajo de fin de grado el año pasado. Esta interfaz también se ha testeado ya con un robot real, el *BlueROV* el cual ya hemos operado en el CIRTESU. El valor añadido respecto al anterior trabajo realizado es la conexión con un robot real y funcional y también la mejora en la usabilidad.

En esta memoria se mencionará el estado actual de las investigaciones en este campo, se hablará del valor que proporciona el proyecto, el proceso de creación de experimentos, los resultados de esos experimentos y por último unas conclusiones junto con las referencias utilizadas en el proyecto.

Palabras clave: HRI, Realidad Virtual, Robótica Marina, Intervención submarina.

Agradecimientos.

El presente trabajo ha sido realizado gracias al apoyo de la Universidad Jaume I, concretamente al *IRS Lab* (*Interactive Robotic Systems*). Me gustaría dedicar un agradecimiento especial a Raúl Marín y Pedro José Sanz, por volcarse de manera activa a la hora de guiarme en este proyecto en el cual empecé a trabajar como parte de mi trabajo de fin de grado, y más sabiendo lo complicado que ha sido para ellos el tener que estar tan atentos a mí.

Gracias a su esfuerzo y al recibirme dentro del grupo con los brazos abiertos, como parte del proyecto TWINBOT, con este trabajo de realidad virtual, se han podido realizar tres publicaciones, dos de ellas en congresos y una tercera en la revista *Robotics* de la cual estoy especialmente orgulloso.

También agradecer directamente a todas las personas que han puesto algo de su esfuerzo en el proyecto, desde a mis compañeros del grupo, como Alejandro Solís que ha formado parte activa de la comunicación entre la interfaz y el robot, a la gente y compañeros de clase que se ofrecieron para probar el sistema en varias ocasiones y darnos sus recomendaciones de cara al mismo.

1. Introducción

Actualmente, las tareas en ambientes subacuáticos, como por ejemplo el mantenimiento de equipo submarino, o el despliegue y recogida de estaciones marinas para la extracción de recursos, se realizan mediante el uso de sumergibles o de *ROVs* equipados con brazos tele operados. En el estado actual del campo, la realidad virtual se utiliza principalmente como una simple forma de control y reconocimiento [1] y podemos encontrar algunos ejemplos de ello en el mercado actual. Algunas empresas ya han creado interfaces que utilizan la realidad virtual(RV) dentro de *ROVs* para la monitorización y la inspección del fondo marino, como por ejemplo: *Qysea* (https://www.qysea.com/products/fifish-v6/) o *Deep Trekker* (https://www.deeptrekker.com) mientras que otros incluyen la posibilidad de manejar los brazos del ROV como *ROV innovations* (https://www.rovinnovations.com/).

Mientras que se utilizan de forma regular vehículos autónomos(*AUVs*) para realizar las misiones de reconocimiento, hay otro tipo de tareas que exigen una serie de habilidades o capacidades que no pueden ser logradas en la actualidad con un equipo autónomo debido a su complejidad [2]. Algunas de las aplicaciones potenciales incluyen: el mantenimiento de observatorios submarinos permanentes, redes de cables y tuberías. Una clase especial de *AUVs* son los *AUVs* de intervención(I-AUV) que en la actualidad son principalmente prototipos cuyo tamaño y complejidad proporciona una serie de funcionalidades limitadas, a pesar de ser ya un avance importante en el campo. Algunas de sus aplicaciones serían: el ensamblaje, el uso de paneles submarinos y la recuperación de objetos sencillos.

En la actualidad, ya hay bastante trabajo realizado para intentar resolver estos problemas. Un ejemplo de ello es el proyecto SAUVIM(Vehículo submarino semiautónomo para las misiones de intervención) [3], que ha demostrado la posibilidad de utilizar un control de flotación autónoma aun cuando se controla el robot y ha abierto un posible nuevo enfoque. Algunos proyectos como el TRIDENT [4] han hecho avances en esa dirección al igual que otros proyectos como el Ocean One, que consiste en un robot humanoide como avatar del controlador humano con una serie de complejos manipuladores y sensores que cuenta incluso con *feedback* háptico [5] en un sistema semiautónomo.

Dentro del grupo de investigación. Consideramos que las tareas con mayor complejidad como podrían ser, la manipulación y el transporte de objetos de gran volumen, o el ensamblado de estructuras complejas debajo del agua podrían necesitar el uso de más de un I-AUV trabajando de manera conjunta. Es en esto en lo que se basa el proyecto TWINBOT. Un sistema con dos I-AUVs que serán capaces de realizar tareas destinadas a las intervenciones cooperativas en escenarios complejos. Una arquitectura de comunicación multimodal se utiliza para permitir que los vehículos puedan comunicarse entre ellos. Un escáner laser permite una búsqueda de nueve puntos 3D de los objetos de interés. Se utilizará por una arquitectura de reconocimiento para localizar los objetos, manipularlos y recrear el escenario en tres dimensiones de la operación. Esto también se puede utilizar a también en la tele operación de sistemas de forma que no haya que depender de las cámaras que tienen factores muy limitantes bajo el agua, como por ejemplo lo limpia que esté la misma o la potencia de las luces del propio robot, dado que se pierde intensidad de la luz solar a más se desciende en profundidad. En la *Figura 01* se puede observar un ejemplo de una tarea cooperativa del proyecto utilizando el simulador *UWSim* [6].

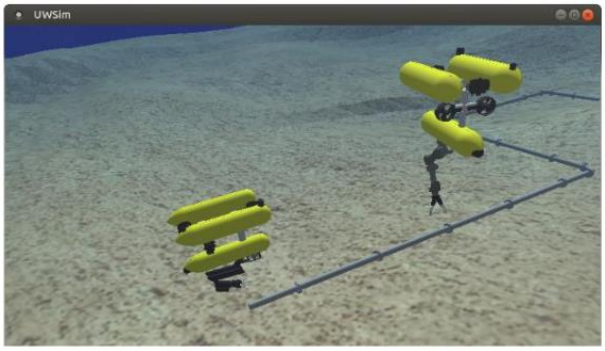


Figura 1. Intervención cooperativa.

Dado que el uso de varios robots para la intervención, hace que en el caso de ser necesaria la teleoperación, el piloto se tenga que encargar de "dos sistemas diferentes" (Dos sistemas de sensores a los cuales prestar atención, dos controladores para mover al robot…). Esto aumentaría su carga de estrés cognitivo al doble, cosa que en la actualidad y como ya se habló en una publicación anterior, es bastante alto impidiendo al usuario que las sesiones duren poco más de 20 minutos [7]. El uso de la realidad virtual puede ayudar a reducir esa fatiga cognitiva y permitir al operador realizar la intervención durante más tiempo, pues toda la información se encontraría dentro del casco de RV y se podría cambiar entre los diferentes sistemas a controlar solo utilizando un botón. Aparte de reducir la cantidad de diferentes emisores de información (generalmente pantallas) a las que se tiene que estar prestando atención, dado que los datos de los sensores están recogidos en el mismo dispositivo de salida.

En los siguientes apartados se hablará sobre el desarrollo de este trabajo. En la sección dos se mencionará la interacción humano-robot (HRI) y la realidad virtual tomando como partida los trabajos previos dentro del campo. Seguidamente, en la sección tres, se expondrán los detalles del equipo con el que llevaremos a cabo nuestra experimentación. En la cuarta sección, se verán reflejados los resultados que hemos obtenido de la experimentación y los métodos utilizados. Y, por último, en la quinta sección se mostrarán las conclusiones obtenidas de esos resultados y el plan de trabajo a futuro como parte de la creación de una interfaz para dos *I-AUVs*. También hay que mencionar que algunos resultados parciales ya se presentaron con anterioridad en un artículo [8].

2. Estado actual del HRI

En este apartado se expondrá todo lo relacionado con el estado actual del campo de investigación en el que se adentra este proyecto, pasando por algunos apartados generales hasta llegar a la motivación y desarrollo de este trabajo.

2.1 Tele operación.

El concepto de teleoperación podría decirse que se originó a finales del siglo XIX, como parte de la industrialización y desde entonces se ha ido desarrollando. Cuando una operación o trabajo tiene que realizarse en entornos con condiciones hostiles que podrían poner en peligro a las personas que están realizando esa acción, la solución es intentar mantener a la persona lejos del foco de peligro, pero permitiéndole controlar la situación, es aquí donde la teleoperación cobra especial fuerza. Uno de los métodos utilizados actualmente consiste en robots que tengan una serie de manipuladores y sensores, mientras que los sensores permiten al operador recibir la información necesaria para realizar el trabajo, el robot realiza las órdenes que el piloto le manda [9-11], aunque esto también genera sus propios problemas [12].

Hay tres problemas típicos producidos por los humanos cuando se realiza una operación a distancia los cuales son: el uso de un comando erróneo, tardar en mandar el comando o no mandar un comando en lo absoluto cuando es requerido. Las principales causas de estos errores generalmente son dos: el operador no tiene el entrenamiento o experiencia suficiente para actuar en dicha situación o el piloto no es capaz de interpretar la información recibida por el robot, algunas veces debido a la fatiga. En múltiples estudios, como en *Chen 2007* [12], se mencionan que algunos factores humanos, como el estrés, la carga de trabajo o la consciencia situacional pueden causar que el operador cometa errores durante la toma de decisiones.

Mientras que existen sistemas que intentan utilizar la inteligencia artificial para solucionar estos problemas, en los que el robot tiene una autonomía parcial, esto reduce el impacto de las decisiones del operador [13], o sistemas los cuales se anticipan las acciones del usuario [14]. A pesar de ello, tomando en consideración que estos problemas se pueden solventar mediante mejores interfaces, al igual que otros investigadores [15].

2.2 Realidad Virtual

Por su propia definición, la RV es la “realidad” más inmersiva dentro de la tecnología y generalmente conlleva el tener puesto un casco que permite crear una simulación de 360 grados permitiendo que el usuario sienta que está colocado dentro del entorno virtual.

Hace alrededor de medio siglo Sutherland presentó *Ultimate Display* [16], un sistema cuya intención era realizar una copia del mundo real en todos los sentidos, pero aún existe un largo recorrido para completar ese propósito. En 1989 Jaron Lanier acuñó el término “Realidad Virtual” [17] en un intento de homogenizar los diferentes conceptos y tecnologías que “representaban” la realidad virtual. Durante los siguientes años, la comunidad científica desarrolló nueva tecnología y algoritmos, lo que hizo que la RV evolucionara a un abanico de posibilidades más grande, no solo a la teleoperación. [1] [18][19][20][21][22][23]

En 2012 un proyecto *Kickstarter* llamado *Oculus Rift,* saldría a la luz. El objetivo de este proyecto era el crear un sistema de alta calidad con un casco como dispositivo de salida. Como ya muchos sabrán el proyecto fue un éxito y ha significado una nueva ola dentro de la RV, en la cual nos

encontramos actualmente. Si bien *Oculus* estaba orientado principalmente a los videojuegos, ha permitido la creación de muchas aplicaciones [24][25][26]. Gracias a la chispa encendida por el proyecto, han salido al mercado una gran cantidad de productos que implementan aspectos mencionados en *Ultimate Display* con precios accesibles para el público general, como por ejemplo el propio *Oculus Rift S*. Todo esto se suma a los esfuerzos realizados por compañías como *NVidia* y AMD que han mejorado mucho las capacidades gráficas de los sistemas proveyendo de soporte a los sistemas de HMDs(head-mounted displays) actuales y futuros.

Desde entretenimiento a medicina, la realidad virtual se ha convertido en una tecnología muy prometedora, que cuenta con mucho apoyo y esfuerzo dentro el campo. Con la que se han realizado muchas investigaciones durante los últimos años. La RV se ha utilizado también para realizar las pruebas de nuevos diseños, por ejemplo, en robótica submarina [15]. Ésta tecnología permite la creación de grandes y complejos entornos virtuales para el entrenamiento y prueba de sistemas que sería muy difícil, por no decir imposible, de conseguir en el mundo real [1].

Dentro de la medicina se han visto una variedad de aplicaciones de la RV, como por ejemplo en los problemas neurológicos [27] y se considera que está lista para aplicaciones generales [28]. En otros aspectos no se está tan avanzado pero las posibilidades son prometedoras. *Tan J.* presenta en su trabajo un complejo reflejo del estado actual de los sistemas de RV que pueden ser útiles para las industrias marítimas como una herramienta de aprendizaje [29]. En el campo de la respuesta en emergencias la RV también ha recibido especial atención [30][31]. También se ha utilizado obviamente en la robótica y en aplicaciones culturales [32][33].

Se considera la realidad aumentada como uno de los posibles siguientes pasos de la realidad vortual [34], en que se combina tanto el mundo real como virtual. En la robótica, mientras que la RV puede ser utilizada para el entrenamiento de los operadores, en la planificación de la misión y para dar instrucciones al robot, la realidad aumentada permite la implementación de sistemas de control en tiempo real con *feedback* inmediato.

2.3 Realidad virtual en HRI para sistemas de intervención submarinos.

La teleoperación es una labor desafiante debido a que el operador tiene que encargarse de realizar el trabajo a través de los sensores del robot, generalmente a través de imágenes, que suelen tener unas limitaciones en cuanto al campo de visión, la información que proporcionan de la profundidad y pueden tener incluso más limitaciones por el ancho de banda. En algunas ocasiones la comunicación puede perderse por completo. Como resultado de esto, la capacidad para conocer el entorno en el que se encuentra el ROV por parte del piloto se puede ver tremendamente afectada y comprometer la efectividad de la misión, incluso llegar a hacer imposible su realización. Sobre el papel, la RV permite la creación de un campo de visión completo e imágenes en 3D, lo que permite también proveer al operador de información de la profundidad, como se ha intentado demostrar en varios estudios [19]. En entornos submarinos, se incrementa la dificultad debido a que generalmente el usuario tiene que cambiar entre diferentes cámaras, tener en cuenta las limitaciones temporales para cada tarea o el retraso en las maniobras del robot debido a los anchos de banda aún más limitados respecto a los entornos terrestres, dado que muchas de las ondas que se usan como por ejemplo el *wifi* no pueden enviarse a través del medio acuático. También, el piloto generalmente se va a encontrar en un barco controlando el ROV desde la superficie del agua, lo que ya de por sí dificulta un poco más la tarea debido a entidades como las olas que se ven reflejadas de forma directa en el barco y aumenta las posibilidades de mareo.

Según ha avanzado la tecnología se han creado gran cantidad de interfaces. En la Tabla 1, podemos ver las características de algunas de las interfaces desarrolladas en algunos proyectos representativos, desde los más antiguos (parte superior) a los más modernos (parte inferior).

Tabla 1. Características principales de las interfaces en diferentes proyectos de ROV y AUV..

Proyecto	Nuevas Características
SAUVIM [3]	Múltiples dispositivos de salida, teclados, joysticks, varios usuarios expertos por robot.
TRIDENT [4]	GUI, un operador, GUIs contextuales
MERBOTS [24]	Cabina de RV con seguimiento y estimación de poses, un operador (no experto).
Venus [25]	Modelos 3D a partir de los datos de los sensores, interfaz de realidad aumentada
OCEAN [5,35]	Dispositivos hápticos bimanuales, visión estereoscópica, GUI, display del mundo, las restricciones cancelan acciones humanas.
DexROV [36]	Simulación en tiempo real del entorno, dispositivos hápticos (exoesqueletos del brazo y la mano), sistema cognitivo para traducir las instrucciones del usuario.

Si bien el objetivo de este trabajo es servir como interfaz de control para diferentes robots, está estrechamente relacionado con el proyecto TWINBOT en el cual participa el *IRS Lab*. En dicho proyecto los robots son autónomos y realizan su tarea simplemente con la supervisión de la persona más no con su control. El motivo por el cual se ha decidido desarrollar esta aplicación es principalmente para esa labor de supervisar el sistema mientras éste opera. Si bien se espera que el sistema de los dos robots sea casi completamente autónomo, siempre pueden surgir imprevistos que un ser humano sería capaz de manejar de manera más eficaz que los sistemas de seguridad del propio robot. Es por ello que existe la necesidad de mejorar las características de las interfaces para que el piloto tenga todas las facilidades posibles y ese es el enfoque principal de este trabajo: que el usuario tenga la experiencia más cómoda posible cuando tenga que actuar en situaciones inesperadas o de gran complejidad en las que el robot no pueda desempeñar su función por sí solo.

2.4 Usabilidad.

La usabilidad no tiene una definición que pueda considerarse homogénea, ni tampoco investigaciones o estandarizaciones sobre el término [37]. Hace referencia a múltiples conceptos como puede ser el rendimiento, el tiempo de ejecución, la satisfacción del usuario o su complejidad de aprendizaje. En la Tabla 2 podemos ver cómo se ha definido el concepto a través de tres diferente estándares. Los estándares relacionados con la usabilidad pueden agruparse en las siguientes categorías:

1. Eficiencia del producto.
2. Atributos del producto.
3. Procesos utilizados para el desarrollo del producto.
4. Capacidades de la organización.

ISO ha desarrollado diferentes estándares en la usabilidad y pueden dividirse en dos categorías diferenciadas:

1. Estándares orientados al producto (ISO 9126, 2001; ISO 14598, 2001).
2. Estándares orientados al proceso (ISO 9241, 1992/2001; ISO 13407, 1999).

Tabla 2. Definiciones de usabilidad según diferentes estádares.

Definiciones de usabilidad	Estandar
La capacidad que tiene el software para ser comprendido, aprendido, utilizado y lo atractivo que es para el usuario en condiciones específicas.	ISO/IEC 9126-1, 2000
El grado o nivel al que un producto puede ser utilizado por usuarios específicos para completar unas metas concretas con eficacia, eficiencia y satisfacción en un contexto cerrado de uso.	ISO 9241-11, 1998
La facilidad con la que un usuario puede aprender, operar, preparar acciones e interpretar los datos de salida del sistema o componente.	IEEE Std. 610.12-1990

Debido a las diferentes definiciones se han creado una gran cantidad de iniciativas para crear un sistema de usabilidad unificado, como por ejemplo el presentado por Seffa [38], pero su impacto ha sido reducido tan solo a campos específicos. Por ello se ha decidido utilizar los estándares de ISO 9241.

El ISO 9241 define la usabilidad de la siguiente manera: "el software es utilizable cuando permite al usuario realizar sus tareas de forma efectiva, eficiente y con un alto nivel de satisfacción en el contexto específico de su uso". De acuerdo a éste, la medida de usabilidad de un sistema consiste principalmente en tres variables:

1. Efectividad: Con qué precisión pueden los usuarios conseguir sus objetivos con el sistema.
2. Eficiencia: Qué recursos consume el sistema para completar el objetivo.
3. Satisfacción: Cómo se siente el usuario cuando utiliza el sistema.

Este estándar presenta unas guías para medir la usabilidad y se utiliza para evaluarla en consecuencia en el contexto en el que se utiliza el software. Se trata de una aproximación orientada al proceso de la usabilidad. Mediante este estándar se consigue un sistema con un proceso de creación orientado al usuario. Los productos útiles pueden diseñarse de acuerdo a las características y atributos requeridos en contextos muy particulares. Esto por sí solo no asegura la eficiencia, la efectividad y la satisfacción al usar el producto. Es por ello que, para verificar si el nivel de usabilidad es suficiente, es necesario comprobar el rendimiento y el nivel de satisfacción de usuarios trabajando con el producto. La medida de usabilidad es una interacción compleja entre los usuarios y el contexto de uso, es por ello que para un mismo sistema, puede tener distintos niveles de utilidad dependiendo del contexto en el que se utilice.

Otra posibilidad puede consistir en utilizar un sistema de escala de usabilidad (SUS), una escala de satisfacción sencilla que consiste en diez atributos de actitud y puede otorgar una visión global sobre la utilidad de un sistema. Fue desarrollado por John Brooke en 1996 [39]. También hay otros autores que han evaluado la usabilidad de sistemas de RV como Marsh [40] que nos da acceso a una perspectiva similar pero un tanto diferente a la mencionada.

3. Equipo y software empleados.

En este apartado se hablará del equipo que se ha utilizado junto con el software para la creación de esta interfaz y los serán comparados con otras posibilidades que se tuvieron en cuenta.

3.1 HTC Vive VS Oculus Rift.

Como ya se menciona anteriormente, *Oculus* fue el precursor de esta segunda ola de realidad virtual en la que nos encontramos, pero a la hora de preparar el proyecto se encontraron dos posibles sistemas como son *HTC Vive* y *Oculus Rift*. Finalmente se decidió que el sistema a utilizar sería el *HTC Vive* como se pude observar en la Figura 2. En la que se puede ver todo el equipamiento que se utiliza en la actualidad. *HTC Vive* es un sistema que crea una inmersión en tres dimensiones realista (https://www.htc.com/es/virtual-reality/). Sus principales características son las siguientes: Un campo de visión de 110 grados, una tasa de refresco de 90 imágenes por segundo, resolución de 2160 x 1200 pixeles, 32 sensores integrados en el casco para la localización espacial del mismo y 24 en cada controlador. Mientras que es recomendable utilizar gráficas de categoría superior a la 1060 de *NVidia*, en nuestro caso utilizamos una *NVidia* 960 GTX, 8 GB de RAM DDR3 y un procesador Intel core i7-4790 a 3.60 GHz.

Figura 2. Sistema de inmersión 3D con HTC Vive.

Con esto dicho aún hay que contestar la pregunta fundamental de este apartado. ¿Por qué utilizar *HTC Vive* y no *Oculus Rift*? Para contestarla se tendrá en cuenta un proyecto anterior [25] en el que ya se empleó el sistema de *Oculus* junto con *Leap Motion* como se puede observar en la Figura 3. El principal motivo es que la precisión recogida con el LM resultaba insuficiente. Esto junto a que LM reconoce gestos y actúa en consecuencia impedía el desarrollo que se tenía en mente. Por ello, había que definir una serie de gestos altamente diferenciados entre sí para realizar las diferentes acciones posibles, cosa que dificulta mucho su uso. Como ya se ha mencionado lo que se quería conseguir es un sistema que sea fácil de usar y aprender y por ello se decidió que se utilizarían joysticks en lugar de LM. Cuando se comenzó a estructurar el sistema la idea era utilizar tan solo un joystick para mantener el sistema lo más comprimido posible, pero finalmente y tras algunas pruebas, se decidió que incorporar el segundo joystick era algo necesario debido a que se pueden realizar muchas acciones con los robots, cosa que dificultaba en gran medida el aprendizaje del sistema. Finalmente se decidió que lo mejor era empezar a usar las gafas de HTC con sus controladores dedicados, pues estos eran más sencillos que un joystick y, aunque se siguen empleando dos, no resulta en un impedimento para el aprendizaje.

Figura 3. Sistema de inmersión 3D con *Oculus Rift* y *Leap motion*.

3.2 Unity VS UWSim.

El *IRS Lab*, como ya se ha recalcado, cuenta con experiencia en entornos de realidad virtual. Utilizando *UWSim* [6] un simulador HIL (Hardware in The Loop), se desarrolló un sistema HRI, adaptando el hardware y el software al escenario que la misión requería. En el pasado *UWSim* ha resultado ser una herramienta de simulación muy potente para sistemas robóticos basados en ROS, permitiendo la integración de las herramientas de simulación para robots en red, por ejemplo NS3. *UWSim* ofrece un sistema de código abierto que es muy utilizado y también tiene una gran aplicación cuando hablamos de robótica educacional. Esta faceta educacional se ha visto reflejada en el hecho de que permitía una simple interacción con el G500, un robot desarrollado en la Universidad de Girona con la que el laboratorio colabora en muchos de sus proyectos. El sistema también cuenta con la posibilidad de generar corrientes y añadir un grado de turbulencia al agua dentro de la escena. También incluye un módulo de *benchmark* que permite comparar diferentes algoritmos de control.

El problema con este sistema viene cuando se intenta implementar la capacidad inmersiva dentro del mismo. Realizar esta tarea suponía una gran carga de trabajo dado que *UWSim* no está preparado para ello, lo que hubiera retrasado los diferentes test y el proyecto en sí. La alternativa era utilizar *Unity* [41], que, cuando comenzó este trabajo, incluía como un *plug-in* toda la interacción con la realidad virtual, pero a día de hoy ya forma parte de las opciones del propio motor de juegos. También, *Unity* proporciona una capacidad más inmersiva que *UWSim*.

Un punto crítico en la decisión fue que el simulador *UWSim* es solo eso, un simulador. Por lo que, sin hacerle grandes modificaciones, no se iba a poder utilizar la interfaz creada en base al mismo con robots reales.

El último motivo por el que se decidió utilizar *Unity*, fue porque *UWSim*, tiene grandes problemas a la hora de que varios cuerpos interactúen. De hecho, el modelado de físicas para esta interacción era fundamental debido a las características del proyecto TWINBOT y dentro del simulador no estaba completamente solucionado este aspecto. Algunas soluciones han sido diseñadas para solventar esto como por ejemplo *ObjectPicker*, que es muy interesante de cara a la realización de prototipos de forma rápida, mientras que el motor de físicas de *Unity* 3D ((https://docs.unity3d.com/Manual/PhysicsSection.html) nos proporciona una respuesta más realista de cara al problema. Sin olvidar que UWSim presenta una mejor integración de cara a ROS que *Unity*, existe una integración de ROS Bridge para *Unity* que puede solventar esta pega de cara a la comunicación con el robot real.

En definitiva, ambas herramientas tienen focos u objetivos diferentes por lo que es difícil compararlas. Pero, *Unity* permite un diseño de físicas y de sensores más realistas y de forma más sencilla por lo que se recomendaría estudiar ambas herramientas según el problema antes de tomar la decisión.

3.3 La interfaz

En cuanto al desarrollo de la interfaz final, la planificación ha sido la siguiente: se realizó una primera versión mediante la cual se controlaba un robot G500 en la simulación de *Unity* en la cual se realizaron cuatro pruebas a diferentes usuarios para medir la usabilidad de la misma y posteriormente se les realizó una encuesta para ver cuán fácil de manejar y aprender era la interfaz dentro de la simulación.

La interfaz puede ser dividida en dos apartados: el primero consistiría en el propio control de los robots a través de los controladores y el segundo, en la información visual que se le presenta al usuario mediante paneles.

En la primera versión, los controles fueron divididos en dos grupos, unos que se encargaban del manejo del vehículo por un lado y los otros del manejo del brazo. Estos dos modos eran independientes y tenían sus propias cámaras. El usuario podía acceder a ellas mediante el botón lateral de los controles, el cual se puede apreciar en la Figura 4.

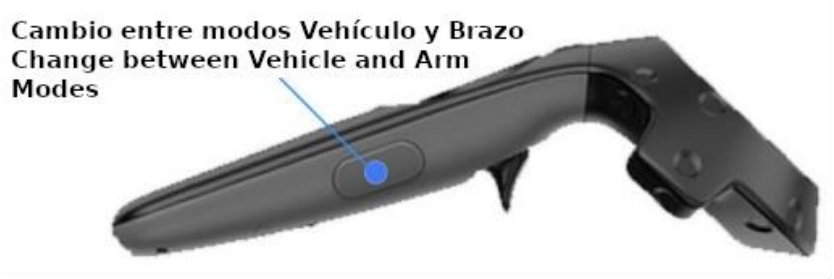
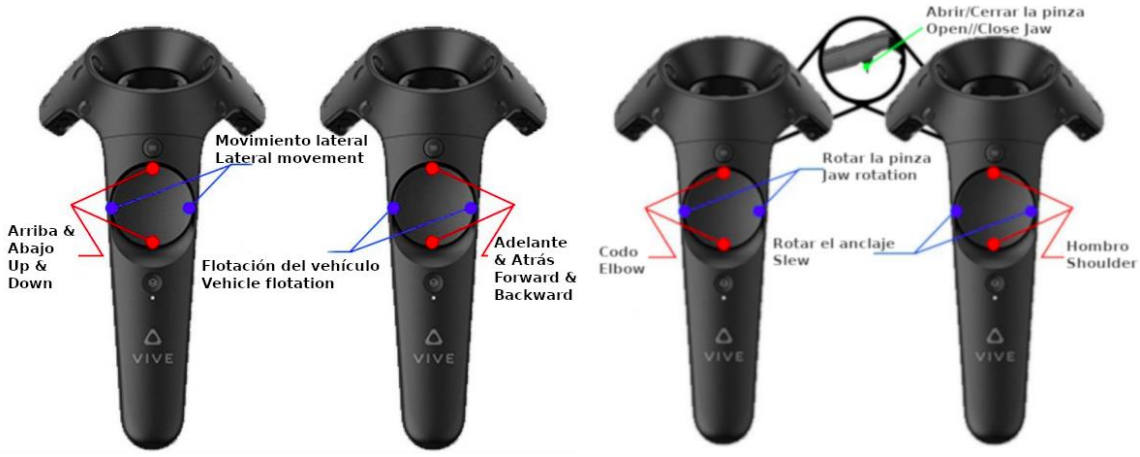


Figura 4. Cambio de modo.

En la segunda versión el usuario puede cambiar entre las cámaras sin necesidad de cambiar lo que está controlando. Por ejemplo: el usuario podría controlar el robot utilizando la cámara incorporada en el brazo, si necesitara cambiar de cámara lo realizaría con el botón lateral del controlador izquierdo y, si por el contrario necesitara cambiar el tipo de control al del brazo. Este cambio se realizaría con el botón lateral del mando de la mano derecha.

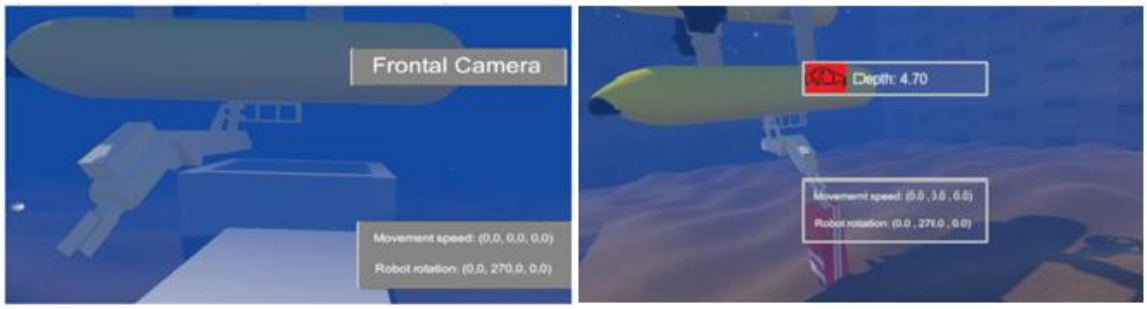
Para pilotar el robot, el control de la mano derecha es el encargado de controlar el desplazamiento adelante y atrás, también el giro del robot sobre el eje vertical, mientras que el mando de la mano izquierda se encarga de los desplazamientos laterales y verticales como se puede observar en la Figura 5 (A). Por otro lado, al controlar el brazo, la mano derecha está encargada de controlar la parte superior del brazo (los grados de libertad del “hombro”) mientras que la mano izquierda se encarga de controlar la rotación de la muñeca y el codo. En ambas manos el gatillo se encarga de abrir y cerrar la garra, como se pude apreciar en la Figura 5 (B).



(**A**) (**B**)

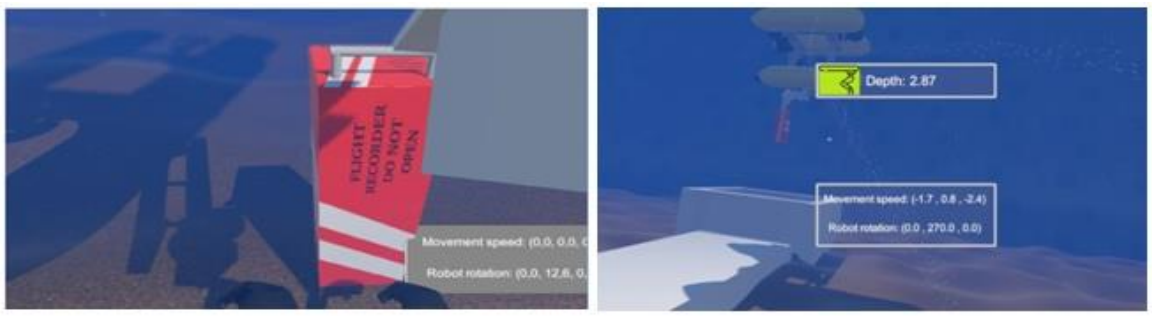
Figura 5. Controles del vehículo (**A**) y controles del brazo (**B**).

En la pantalla, mediante el uso de ventanas superpuestas (display-fixed [43]) se muestra la imagen captada por la cámara seleccionada, al igual que información extra para el operador, como puede ser la velocidad del vehículo y la orientación del mismo. En la segunda versión se agregó un *display* de la profundidad. En la primera versión, la información se mostraba encima de un panel sólido de color gris y en los límites de la visión del operador, como se observa en la Figura (6), A. En la segunda versión, para intentar facilitar la compresión y la posibilidad de ver la información por parte del usuario, se movieron a un punto más central dichos paneles y se les aplicó un grado de trasparencia, como se observa en la parte (B) de las Figuras 6, 7, 8 y 9.



(**A**) (**B**)

Figura 6. Primer test de usabilidad ((**A**) Primera versión y (**B**) Segunda versión).



(**A**) (**B**)

Figura 7. Segundo test de usabilidad ((**A**) Primera versión y (**B**) Segunda versión).

(**A**) (**B**)

Figura 8. Tercer test de usabilidad ((**A**) Primera versión y (**B**) Segunda versión).

(**A**) (**B**)

Figura 9. Cuarto test de usabilidad con el obstáculo pared vertical ((**A**) Primera versión y (**B**) Segunda versión).

Los paneles se dividen en dos secciones, los de la parte superior e inferior de la pantalla:

1. En la primera sección, dentro de la primera versión, se podía observar el nombre de identificación de la cámara activa, que coincidía con lo que se estaba manejando, el vehículo o el brazo. Por otra parte, en la segunda versión la identificación de la cámara se eliminó y se dejó un icono que representaba lo que se estaba controlando y el valor de profundidad actual del robot.
2. En la segunda sección tanto en la primera como en la segunda versión se muestran las velocidades del vehículo y la orientación del mismo.

En primera instancia iba a existir una tercera sección sobre la tasa de refresco de las imágenes que existía junto con el retraso a la hora de recibir y procesar los comandos dentro de la simulación, pero según se fue avanzando se desechó esta idea y la tasa de refresco no se muestra en el casco sino en la pantalla del ordenador al que está conectado.

Para finalizar, en la Tabla 3 se pueden observar las diferencias resumidas entre las versiones.

Tabla 3. Diferencias entre las versions de la interfaz.

Primera Versión	Segunda Versión
Información sobre la velocidad y la orientación del vehículo.	Se incluye también información sobre la profundidad actual del vehículo.
Paneles sólidos	Paneles transparentes.
Sin sonido.	Sonido del agua, el motor y aviso de proximidad.

Sin manual o demostración.	Para hacer más fácil la comprensión del sistema se realizó un manual de la interfaz y varios vídeos sobre la misma.
Dos cámaras.	También dos cámaras pero la del vehículo fue cambiada de posición .
Dos modos de trabajo: vehículo y brazo.	Tres modos: vehículo, brazo y combinación de cámaras de un modo con el otro.

La interfaz también cuenta con algunas funcionalidades para asistir al operador a la hora de realizar la misión. Para ello se añadió en la simulación un escenario realista que representa la recuperación de una caja negra de un avión. Tanto en la Figura 10 como en la Figura 11 podemos observar este sistema de asistencia en acción. Cuando la garra del robot se coloca en una posición en la que puede coger la caja, ésta comenzará a parpadear con un color purpura, en la Figura 10 se observa la caja cuando no se puede agarrar cerrando la garra y en la Figura 11 se observa el parpadeo cuando sí es posible el agarre.

Esta capacidad es aplicable a todos los objetos que el robot podría agarrar, dado que dentro de *Unity* reciben la etiqueta *Takeable* que delimita qué objetos dentro de la escena se pueden agarrar y cuáles no. En la Figura 12 podemos observar una secuencia de imágenes que muestran las distintas fases del agarre.

Figura 10. Caja negra antes de poder ser agarrada por la garra.

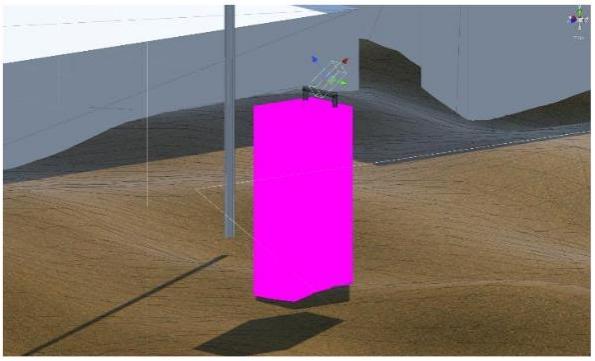
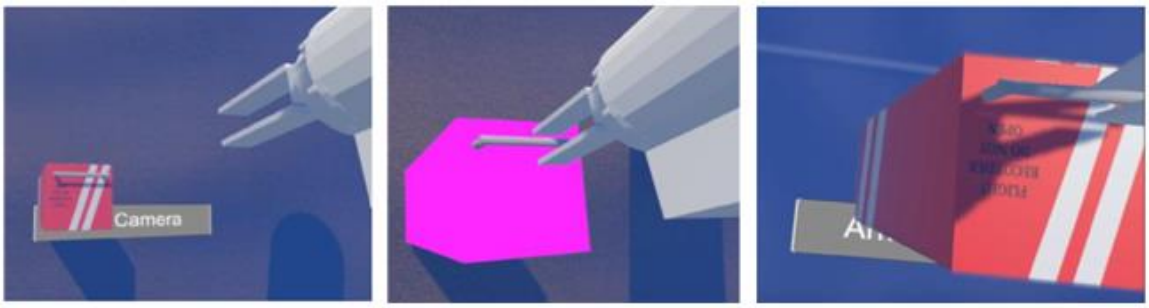


Figura 11. Caja negra cuando puede ser agarrada por la garra.

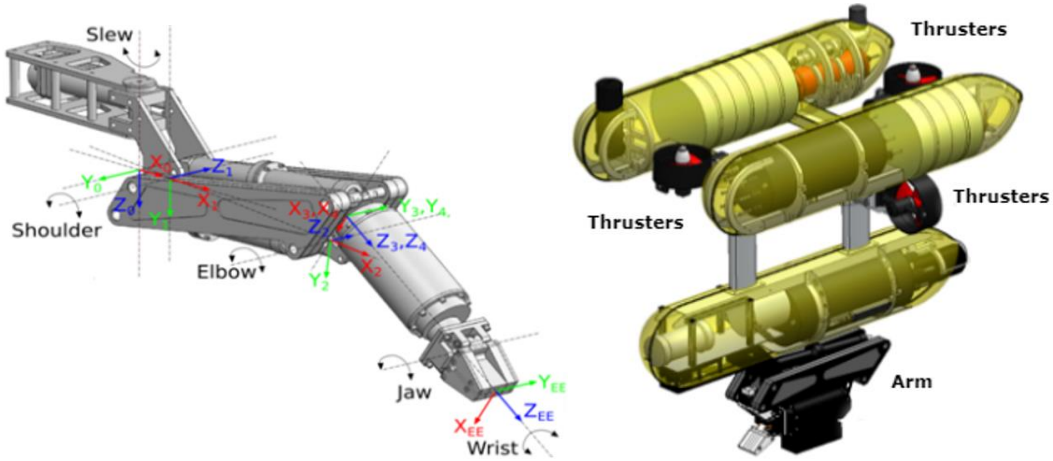


(**A**) (**B**) (**C**)

Figura 12. Fase de aproximación de la garra (**A**), fase de confimación del possible agarre (**B**), y agarre de la caja negra (**C**).

Esta cualidad de la simulación facilita al operador la misión del agarre, pues es una confirmación visual del correcto agarre del objeto.

El robot que aparece en la simulación es el Girona 500, que es el utilizado en el proyecto TWINBOT (https://cirs.udg.edu/auvs-technology/auvs/girona-500-auv/); el brazo es el Arm5e [42]. Ambos se pueden observar en la Figura 13 (A) el brazo y en la Figura 13 (B), el robot.



(**A**) (**B**)

Figura 13. A: Arm5e, brazo utilizado con el Girona 500. En la imagen se pueden apreciar los grados de libertad de las articulaciones. B: Robot Girona 500, un AUV reconfigurable diseñador para una profundidad máxima de 500 m. El vehículo está compuesto por un marco de aluminio, que soporta tres contenedores con forma de torpedo de 0.3 m de diámetro y 1.5 m de longitud, así como otros elementos como sensores y propulsores. Imagen A adaptada de [42]. Imagen B adaptada de imágenes en https://cirs.udg.edu/auvs-technology/auvs/girona-500-auv/.

4. Desarrollo de la interfaz

La interfaz, como ya se ha mencionado, está desarrollada mediante Unity. Para todo lo que es la parte de simulación se ha utilizado el motor de físicas con el que cuenta este entorno de desarrollo, teniendo en consideración algunas modificaciones necesarias para representar el entorno subacuático.

Esta categoría tiene dos partes, la primera que es la relacionada al software, en la que se hablará sobre los diferentes scripts que se han creado y cómo actúan los elementos dentro del entorno virtual. En cuanto a la segunda, se expondrán algunos aspectos del diseño que se centran principalmente en otorgar un alto grado de realismo a la simulación sobre la que se aplica la interfaz.

4.1 Software de la interfaz

Para la interfaz se utilizó en primera instancia el robot Girona 500 con el brazo Arm5e. Como ya se ha mencionado anteriormente, para el desarrollo de la misma se siguieron una serie de pasos. En primer lugar, se realizó el desarrollo del brazo, dado que era la parte más complicada, debido a que, dentro del entorno, es la parte del robot que más interacción tiene con el resto de los objetos que hay en la escena. En segundo lugar, se realizó la integración del vehículo a la simulación, haciendo que fuera capaz de moverse por la escena y que interactuara con algunas de las entidades presentes, como el fondo o las paredes. Después, se creó todo lo relacionado con la estética del entorno que otorgaba realismo a la simulación, como por ejemplo las burbujas de los motores al estar activos. Seguidamente, se programó todo lo relacionado con la interfaz que proporciona la información al usuario. Como ya se ha mencionado anteriormente, hay dos versiones de la misma con algunas diferencias que serán remarcadas en su apartado. Para finalizar, se creó la comunicación entre la interfaz y el robot real.

4.1.2 Integración del brazo en el entorno.

El brazo utilizado en la simulación es, como ya se ha mencionado el Arm5e, concretamente la modificación que se realizó en la UJI sobre el mismo, con un cambio en la garra para simplificar el agarre dentro de la simulación.

En un principio se realizó un modelo simplificado, compuesto únicamente de cubos en Unity para realizar la integración del movimiento, para más tarde aplicar esa programación a un modelo realista del brazo.

Para realizar los distintos movimientos del brazo se ha creado una serie de scripts, que consisten en una serie de condiciones que representan las posibles acciones del usuario, como la que se aprecia en la Figura 15, de cara a traducirlas al movimiento del brazo, como se observa en la Figura 14.

```
//Slew Control

if (slewLeft)
{
    if (slewMinR < 60 ) {
        slew.transform.Rotate(0, 0, rotatS);
        slewMaxR -= rotatS;
        slewMinR += rotatS;
    }

}
else if (slewRight)
{
    if(slewMaxR < 60)
    {
        slew.transform.Rotate(0, 0, -rotatS);
        slewMinR -= rotatS;
        slewMaxR += rotatS;
    }

}
```

Figura 14. Script del movimiento lateral del hombre del brazo.

```
SteamVR_Actions.default_TouchTouchpad.stateUp
```

Figura 15. Booleano que representa la orden recibida mediante el controlador.

Dentro de la jerarquía del *prefab* del brazo, que se puede apreciar en la Figura 16, se encuentran todas las diferentes articulaciones, lo que permite al script reconocer las diferentes partes del brazo para saber cuál tiene que mover según qué orden se reciba a través de los controles, esto es perceptible en la Figura 17.

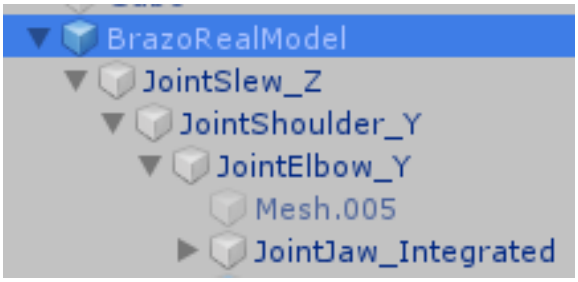


Figura 16. Jerarquía del modelo del brazo dentro del entorno de Unity.

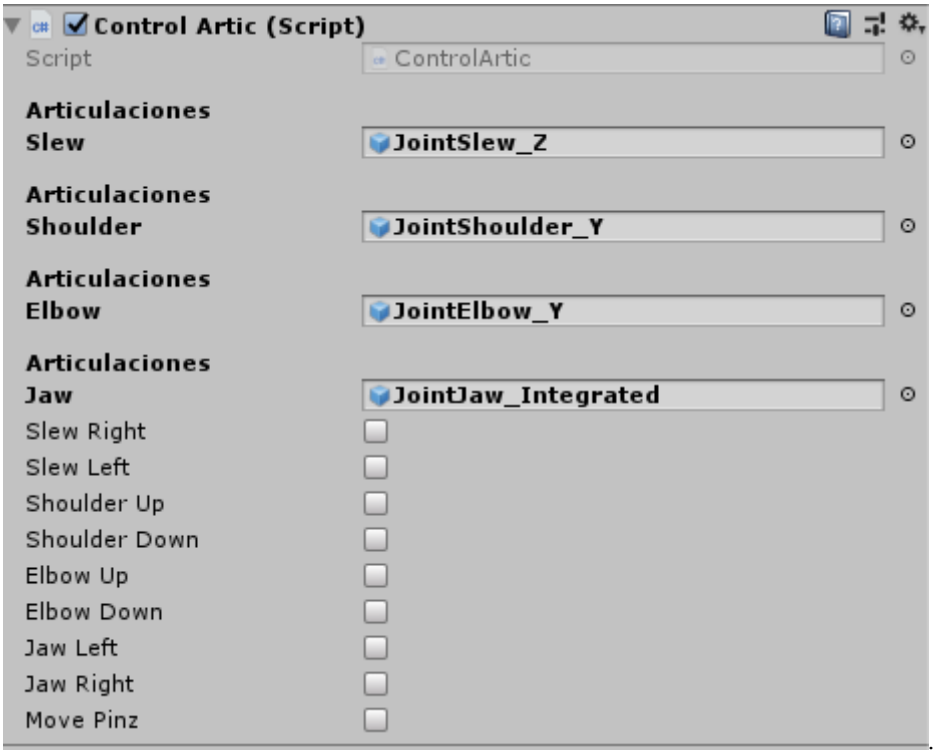


Figura 17. Enlaces entre las partes del brazo y el script de manejo que se encuentra en el padre del modelo.

La parte primordial del brazo es la garra, Figura 18, dado que es la encargada de agarrar los diferentes objetos dentro del entorno. Es por ello que le agregamos una caja de colisión a la zona de agarre de la misma para que se pueda saber en qué momento podemos agarrar algo y también realizar dicha interacción de agarre, Figura 19.

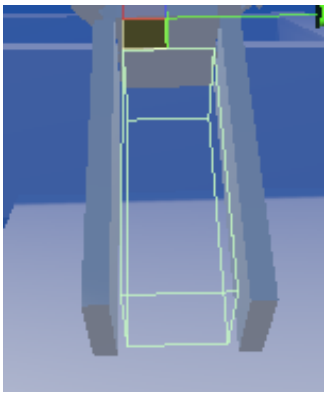


Figura 18. Caja de colisión de la garra.

```
if (Inter(other))
{
    objetoAgarrable = other.gameObject;
    triggerObj = objetoAgarrable.GetComponent<ParticleSystem>().emission;
    triggerObj.enabled = true;
}
```

Figura 19. Script que activa el sistema de partículas que indica si se puede agarrar un objeto.

4.1.2 Integración del vehículo submarino

En cuanto a la integración del G500 en la simulación, se hizo de forma contraria al brazo. Lo primero que se hizo fue poner el modelo dentro del entorno de *Unity*.

La forma en la que el vehículo recibe las órdenes es muy similar a la del brazo, razón por la cual no se realizó un modelo intermedio para el scripting. Lo único que cambia respecto al brazo es que, en lugar de ser un movimiento uniforme, el movimiento del vehículo tiene una aceleración que es directamente proporcional a la fuerza que generamos con los motores, Figura 20.

```
if (rb.velocity.magnitude < 10)
{
    AddForceForward(speedMove);
    ForwardBackwardEngineEnable(true);

}
```

Figura 20. Script que aplica fuerza sobre el modelo en respuesta al movimiento de los motores.

Aparte del script de movimiento, el vehículo solo tiene una interacción definida mediante las cajas de colisión del mismo, que en este caso son tres cilindros con las puntas redondeadas. Cuando se produce una colisión, ya sea con las paredes, el fondo de la piscina o la superficie del agua, la simulación se reiniciará a un punto inicial para comenzar la misión desde allí de nuevo, el estado inicial se puede apreciar en la Figura 21.

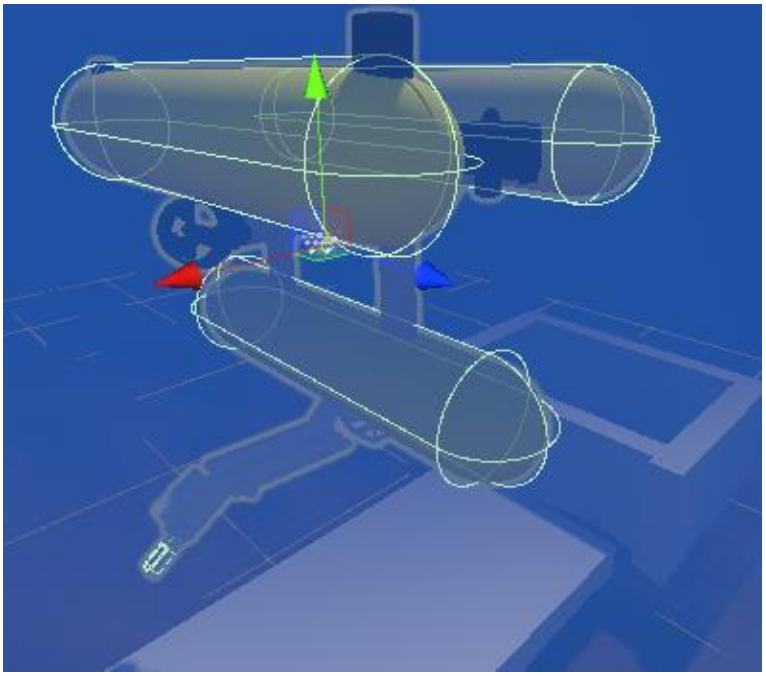


Figura 21. Estado original de la simulación.

4.1.3 Entorno subacuático.

En este apartado se hablará de las cosas que se han hecho para buscar realismo dentro de la simulación.

Lo primero de lo que se va a mencionar es el sistema de partículas que genera las burbujas en el agua con el movimiento de los motores, cuyo script podemos apreciar en la Figura 22. Este sistema tiene en cuenta la fuerza que ejercen los motores a la hora de decidir cada cuánto tiempo se generan las partículas.

```
private void RateControl(float speedMagnitude,ParticleSystem.EmissionModule bubblePart)
{
    bubblePart.rateOverTime = speedMagnitude/2;
}
```

Figura 22. Script del sistema de partículas de los motores.

Para representar la forma de actuar de las burbujas, la posición de generación de las mismas es aleatoria dentro de unos límites, al igual que el movimiento de las mismas a través del tiempo.

En cuanto al color del agua, se ha aplicado un filtro de color a la propia cámara para que se vea con los colores típicos del fondo submarino. Por otra parte, para representar el fondo del agua y la superficie, se ha simplificado el problema de tal forma que ambas partes utilizan el mismo principio. Se ha creado un *shader* que realiza unas modificaciones visuales sobre el fondo del agua y la superficie, haciendo que se creen unas ondulaciones dentro de unos límites. La única diferencia entre la superficie del agua y el fondo, es que en el *shader* encargado la superficie, que es esencialmente el mismo que el del fondo, se ha creado una variación con el tiempo que simula el movimiento del agua, que puede observarse en la Figura 23.

```
float3 v0 = v.vertex.xyz;
float3 bitangent = cross(v.normal, v.tangent.xyz);
float3 v1 = v0 + (v.tangent.xyz *0.01);
float3 v2 = v0 + (bitangent * 0.01);

float ns0 = _NoiseScale * snoise(float3(v0.x + _NoiseOffset.x, v0.y + _NoiseOffset.y, v0.z + _NoiseOffset.z));
v0.xyz += ((ns0 + 1) / 2) * v.normal;

float ns1 = _NoiseScale * snoise(float3(v1.x + _NoiseOffset.x, v1.y + _NoiseOffset.y, v1.z + _NoiseOffset.z));
v1.xyz += ((ns1 + 1) / 2) * v.normal;

float ns2 = _NoiseScale * snoise(float3(v2.x + _NoiseOffset.x, v2.y + _NoiseOffset.y, v2.z + _NoiseOffset.z));
v2.xyz += ((ns2 + 1) / 2) * v.normal;

float3 vn = cross(v2 - v0, v1 - v0);

v.normal = normalize(-vn);
v.vertex.xyz = v0;
```

Figura 23. Script modificador del *shader* de la superficie del agua.

4.1.4 Canvas e interfaz.

En este apartado se hablará de la información que el sistema muestra al usuario en la interfaz. Al igual que en la primera versión tenemos tres secciones, de las cuales, en este caso, el apartado en el que se muestra la tasa de refresco y el retraso al mandar órdenes, no se muestra al usuario. En la parte superior derecha se observa qué está controlando el usuario, si el vehículo o el brazo y un identificador de qué cámara está usando. En la parte inferior derecha el usuario puede ver diversa información como la velocidad de desplazamiento y la orientación del vehículo. Se puede observar esta parte en la Figura 25.

En cuanto a la segunda sección no tiene nada de scripting, simplemente al cambiar el control o al cambiar la cámara se muestra qué se está usando con un icono en el caso del control y un nombre en el caso de la cámara.

En cuanto a la tercera, que tiene la información de movimiento y orientación, el script tiene en cuenta la velocidad a la que se desplaza el cuerpo rígido atado al G500 y la orientación del mismo para mostrar por pantalla al usuario esos mismos datos, el *scripting* de esta parte se puede observar en la Figura 24.

```
private void UpdateTexts()
{
    for(int i=0; i < velocities.Length; i++)
    {
        if (i % 2 == 0)
        {
            velocities[i].text = "Movement speed: " + rb.velocity;
        }
        else if (i % 2 == 1) {
            velocities[i].text = "Robot rotation: " + rb.rotation.eulerAngles;
        }
    }
```

Figura 24. Script del display de texto.

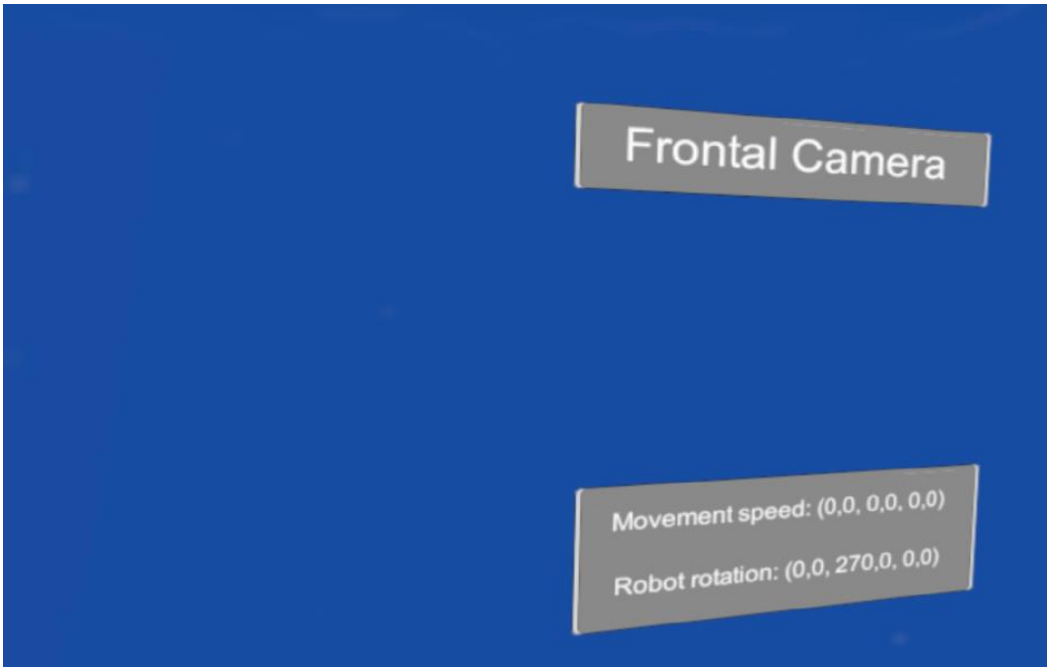


Figura 25. Primera versión del canvas.

4.1.5 Conexión robot, interfaz.

En cuanto a la conexión con el robot real, ésta se realiza a través de un script en Python que se encarga de realizar la comunicación, Existen dos versiones del mismo, la primera que utiliza simplemente MavLink para realizar la comunicación pasando información mediante un puerto, lo cual obliga a que el robot y el ordenador estén conectados a la misma red y, a ser posible, a través de un cable Ethernet, mientras que en la segunda versión el ordenador se conecta a un servidor que está conectado con el robot y le manda peticiones a través de mensajes, cuya construcción se puede observar en la Figura 27.

En cuanto a la primera versión, el script es bastante sencillo como se observa en la Figura 26. Simplemente tiene una serie de condiciones en cuanto a lo que se tiene que pulsar, como en la interfaz en sí, pero es “engañado” para que crea que se pulsan teclas a pesar de ser los botones del controlador.

```
#Right/Left
if(key.is_pressed('a') and y != -speed):
    y = -speed
    kpress=True
elif(key.is_pressed('d') and y != speed):
    y = speed
    kpress=True
else:
    y = 0
```

Figura 26. Script Python, Movimiento lateral.

```
#signal
if(kpress==True):
    kpress=False
    master.mav.manual_control_send(
        master.target_system,
        x, #x
        y, #y
        z, #z
        rotation, #rotation
        0) #active button
    print("signal on")
    #time.sleep(0.1)
    print("slept")
```

Figura 27. Mandar señal al robot.

En este caso, con Mavlink, la interacción se realiza de forma muy sencilla. Simplemente se utiliza la información que ya nos proporciona la librería para crear un vector y mandárselo al robot para que éste realice la acción.

En cuanto a la versión del servidor, que también tiene como parte interna la misma biblioteca, se realiza una comprobación de la conexión con éste y se van montando los mensajes que se le mandarán según las “interrupciones de teclado” que se generan cuando el usuario manda una orden. Todo esto se puede observar en la Figura 28 y la Figura 29.

```
if __name__ == '__main__':

    #master = mavutil.mavlink_connection('udpin:192.168.2.1:14550')
    master = mavutil.mavlink_connection('udpin:127.0.0.1:14551')
    master.wait_heartbeat()
    server_class = BaseHTTPServer.HTTPServer
    httpd = server_class((HOST_NAME, PORT_NUMBER), MyHandler)


    print time.asctime(), "Server Starts - %s:%s" % (HOST_NAME, PORT_NUMBER)
    try:
        httpd.serve_forever()
    except KeyboardInterrupt:
        pass
    httpd.server_close()
    print time.asctime(), "Server Stops - %s:%s" % (HOST_NAME, PORT_NUMBER)
```

Figura 28. Conectar con el servidor

```
#MOVE IN X (FRONT/BACK)
if(modes == "forward"):
    x=speed
    y=0
    z=500
    rotation=0
    kpress=True
    e=0
elif(modes == "backward"):
    x=-speed
    y=0
    z=500
    rotation=0
    kpress=True
    e=0
else:
    x=0
```

Figura 29. Variables para mandar el mensaje al servidor.

Cuando el usuario manda la orden y se actualiza el mensaje, éste se manda al servidor y éste último será el encargado de mandarlo al robot para que realice la acción.

Como se verá en la experimentación, esta segunda versión es algo más lenta incluso estando todo el sistema dentro de la misma red debido al paso intermedio de comunicación, pero es una solución válida al problema que se tenía entre manos.

4.2 Diseño de la interfaz.

Como ya se ha mencionado, en este apartado se comentarán los aspectos de la parte de diseño, sobre todo de aquellos elementos con los que se puede interactuar y aquellos que otorgan un mayor grado de realismo a la simulación.

4.2.1 Entorno submarino

Para comenzar con este apartado se explicarán los sistemas de las partículas.

Comenzando por las partículas generadas por los motores, que intentan representar las burbujas generadas por el movimiento de los mismos. Estas partículas se han creado con una textura de burbuja real que se ha aplicado sobre un partícula “sin forma”. El funcionamiento consiste en que la textura está orientada a la cámara, de forma que no hace falta que la misma sea una esfera y en consecuencia se reduce el consumo producido por estas texturas, Figura 30.

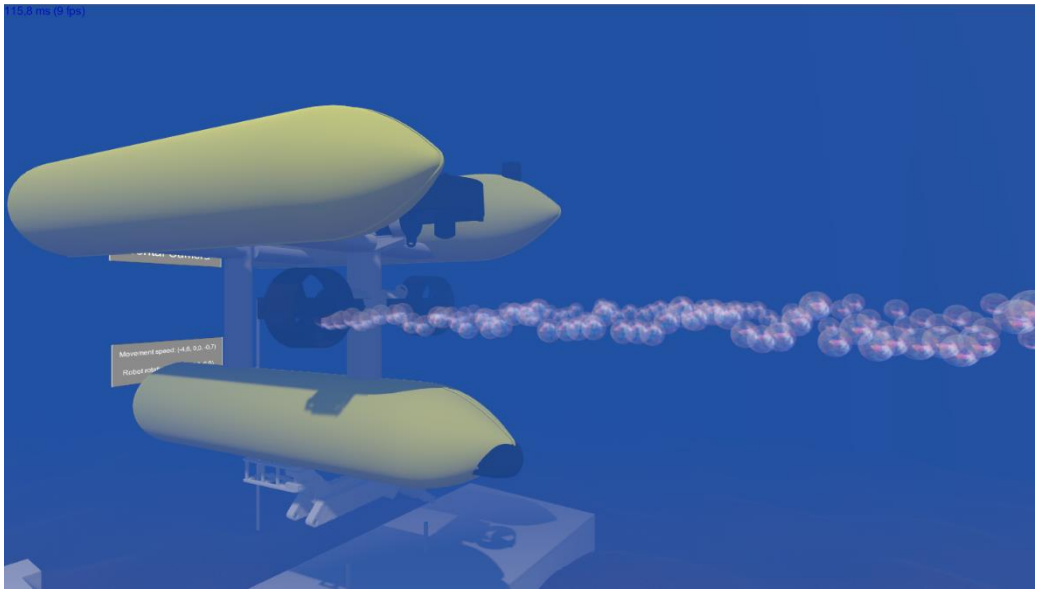


Figura 30. Partículas del motor.

Las partículas del agua son incluso más sencillas, pues solo consisten en puntos que tienen un movimiento aleatorio regido por unos límites que simulan las partículas movidas por las corrientes de agua, como se puede ver en la Figura 31.

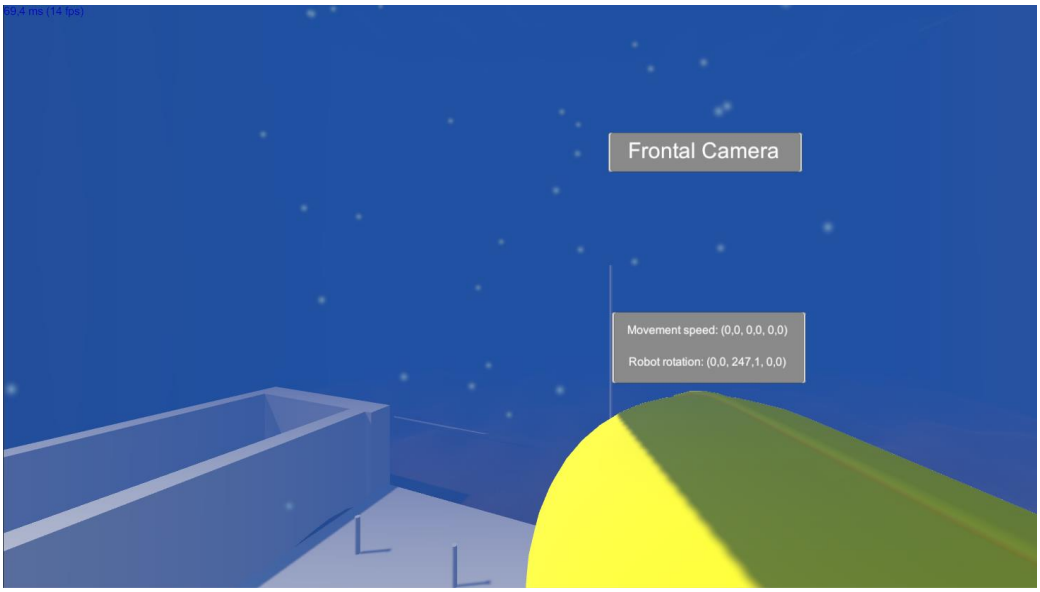


Figura 31. Partículas del agua.

Después de explicar los sistemas de partículas, se va a hablar sobre el fondo del agua y de su superficie. Ambas consisten en un *shader* de postprocesado que genera una cierta ondulación en la superficie del objeto sobre el que actúan, en este caso dos planos. Para la superficie del agua, el *shader* va cambiando los valores simulando el movimiento de la misma, mientras que el fondo se mantiene totalmente quieto, ambos, el fondo y la superficie se pueden observar en la Figura 32.

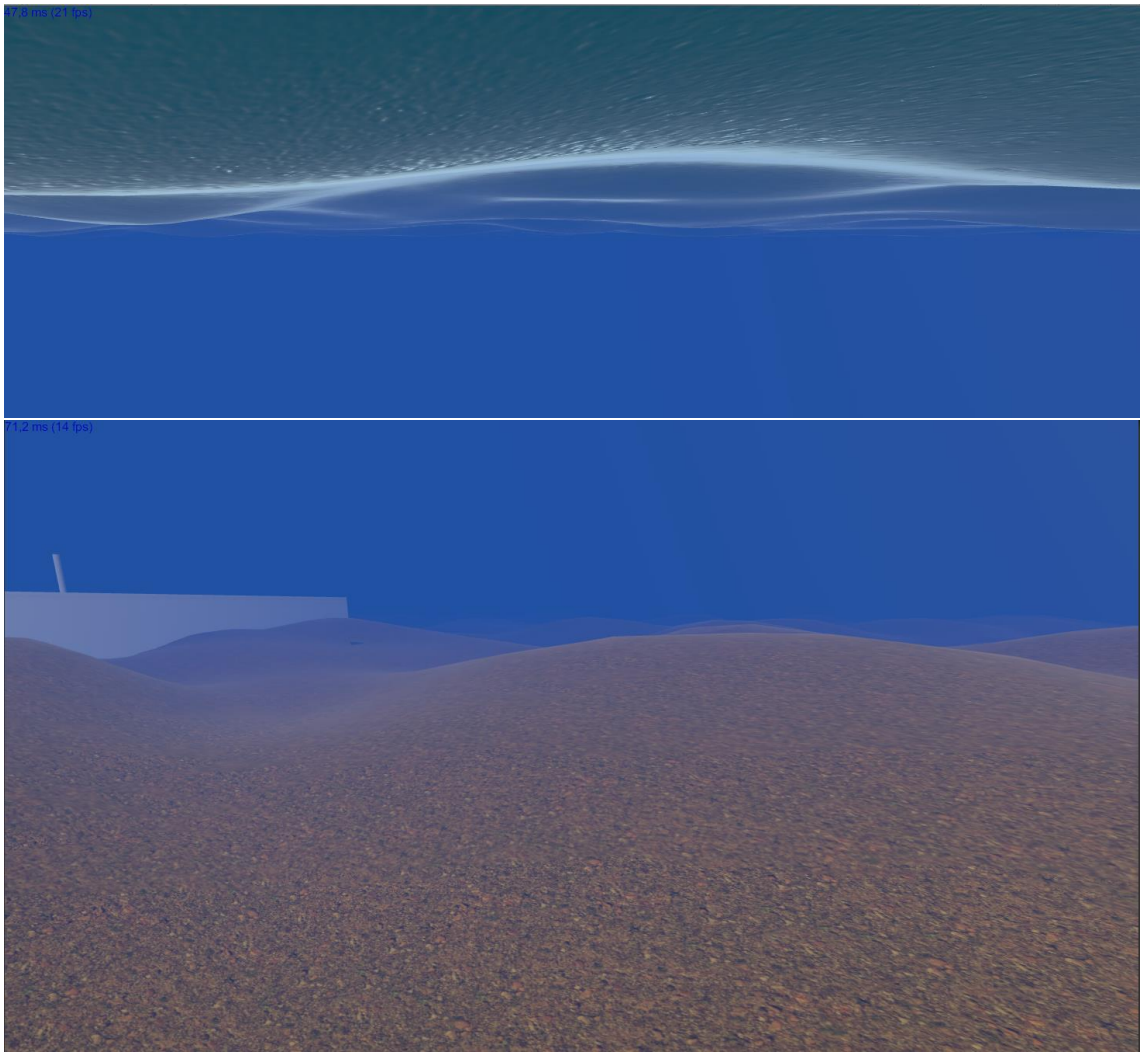


Figura 32. Superficie del agua(arriba) Fondo del agua(abajo).

Para continuar, se hablará sobre los efectos de la cámara que simulan que ésta se encuentra debajo del agua. Estos efectos se basan en un *plug-in* llamado *AQUAS* que simula el agua con mucha precisión, simplificando mucho su funcionamiento en cuanto a la cámara y sus colores. Este *plug-in* crea una "niebla" que dada cierta distancia difumina los colores a un azul oscuro, lo que simula el color del entorno submarino: eso es exactamente que lo que hemos hecho. Colocar una niebla de dicho color que dada una cierta distancia haga que la cámara solo capte un color azul, una comparativa del efecto con el modo normal se puede apreciar en la Figura 33.

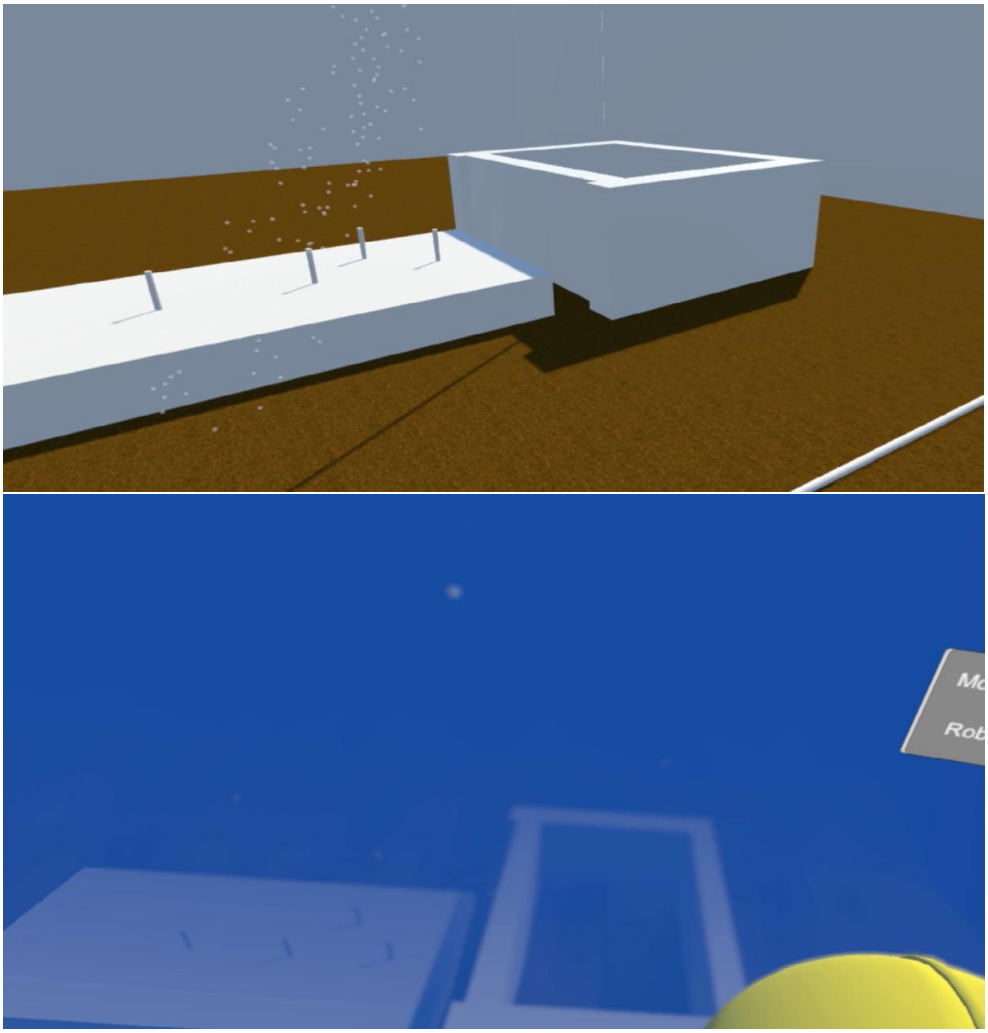


Figura 33. Simulación cuya cámara está sin la niebla(arriba) y con niebla(abajo).

Por último, se quiere mencionar que, si bien no otorga realismo a la simulación, simplifica mucho el trabajo del operador: un sistema de partículas para los objetos con los que se puede interactuar. Este sistema de partículas, sencillamente marca cuándo un objeto se puede agarrar con el *gripper* simplemente cerrándolo o no, esto se puede observar en la Figura 34.

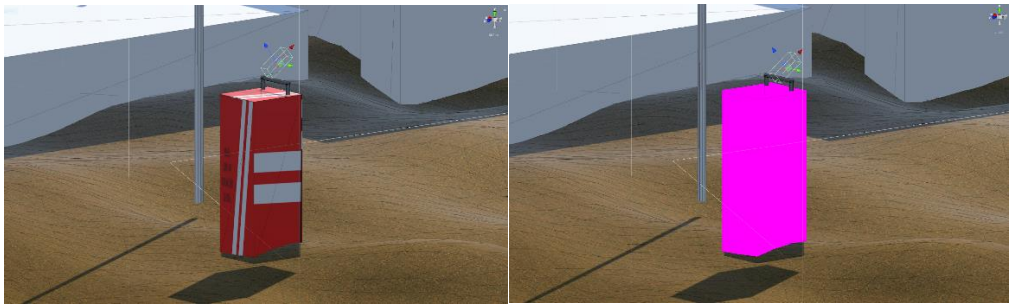


Figura 34. Sistema de partículas de agarre: no puede agarrarse cerrando(izquierda) y se puede realizar el agarre cerrando(derecha).

5. Experimentación.

En este apartado se mencionarán de las diferentes pruebas que se han realizado con la interfaz, comenzando con las pruebas de rendimiento para ver tanto la tasa de refresco y respuesta de los controles con la aplicación como los tiempos de control del robot real, el cual en lugar de ser el G500 fue el *BlueROV* debido a que el grupo aún no disponía del primero.

5.1 Pruebas de rendimiento

Como ya se mencionó en el apartado anterior, la interfaz ha sido utilizada a través de una simulación y posteriormente se ha conectado con el *BlueROV* para realizar las primeras pruebas de rendimiento con un sistema físico en lugar de simulado. La primera prueba que se realizó con la simulación consistía en medir la tasa de refresco de las imágenes de la misma, ya que la carga que los sistemas de realidad virtual ponen sobre el equipo puede derivar fácilmente en tasas por debajo de las 20 imágenes por segundo, cosa que podía suponer un problema. Tras algunas pruebas de optimización, principalmente de cara a la carga gráfica de la simulación, se consiguió realizar un test de rendimiento cuyos resultados se pueden observar en la Figura 35, en la cual se pueden apreciar los retardos a la hora de recibir las órdenes de los controladores a la simulación, para realizar la prueba se crearon tres tipos de cargas en los escenarios, la primera, tenía simplemente al robot, el agua, las paredes de la piscina y una base y los movimientos del robot eran lo más pequeños posibles. A este apartado se le denominó *Stand By* dentro de los registros de las pruebas. Por otro lado, se nombró *Average Overload* a un entorno en la que se añadieron varios objetos como cilindros a los cuales se les otorgó un material simple y su cantidad de polígonos no superaba los 100 por cilindro. Por último, se cargaron varias esferas y texturas junto con los anteriores objetos de la escena para realizar lo que denominamos *High Overload* que era un test en el que se intentaba sobrecargar todo el sistema para comprobar el tiempo de respuesta. Para las dos primeras pruebas la tasa de imágenes se mantenía en torno a 50 fps, mientras que en la última se llegó a alcanzar los 20 fps. Esto por si solo tampoco nos daba ningún dato interesante, por lo que mostramos en la Figura 35 son los resultados de cuánto tiempo tardaba la orden dada por el usuario en reflejarse dentro de la simulación con diferentes ordenadores.

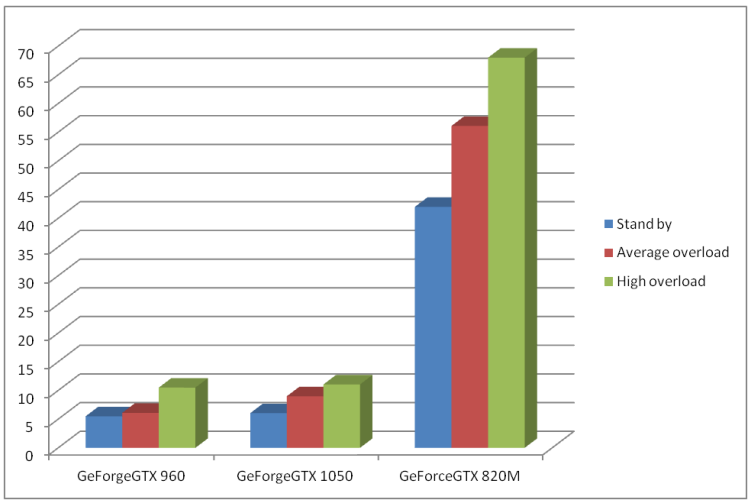


Figura 35. Tiempos de respuesta en milisegundos según las tarjetas gráficas.

Los resultados obtenidos en la Figura 35 se tomaron con ordenadores de las siguientes características: todos ellos contaban con un procesador Intel® Core™ i5-7300HQ 2.5 GHz y 8 Gb de RAM DDR3, por lo que el elemento determinante que cambiaba era la tarjeta gráfica que a su vez, es la que más trabajo tiene en cuanto al procesado de RV se refiere. Como era de esperar la gráfica de la gama para portátiles de *NVidia*, la *GeForce* 820M, consiguió tiempos que de cara a la simulación resultaron nefastos, con un tiempo mínimo de respuesta de casi medio segundo. Por otro lado, los mejores resultados se consiguieron con la gráfica que más tarde pasaría a ser la principal con la que se trabajaría en el sistema final que ya ha sido descrito, la *GeForce* GTX 960, cuyo tiempo de respuesta no superó los 10 milisegundos, ni siquiera en la prueba de alta sobrecarga.

Teniendo en cuenta los resultados de este test, y sumando el tiempo medio de respuesta de 5 milisegundos que se obtuvo para la gráfica seleccionada cuando se estableció la conexión con el robot real, el *BlueROV*, a través de dos medios diferentes *MavLink* y mediante un servidor, se adquirieron dos resultados para tiempo de respuesta con el robot real. Para ambos casos se tomaron 10.000 muestras de tiempo de respuesta para tomar la media y los resultados fueron los siguientes:

- Con *MavLink,* un sistema que está pensado para pilotar drones, pero que tiene una fácil implementación de cara al *BlueROV,* los tiempos de respuesta desde que el usuario mandaba la orden hasta que era mandada al robot pasando por la simulación fueron de 45,3571 milisegundos. Unos tiempos esperados si tomamos en cuenta que durante todo el proceso se utilizó una conexión con el robot a través de un cable Ethernet y teniendo en cuenta que la compañía encargada de desarrollar esta comunicación lleva varios años trabajando en ello (https://mavlink.io/en/).
- Por otro lado, utilizando el modelo de cliente-servidor, también se pudo realizar la misma cantidad de mediciones y sacar una media de 65,2143 milisegundos. Si bien el tiempo de respuesta es de alrededor de dos décimas más alto, la ventaja que tiene este modelo es que ha sido desarrollado por el grupo y es fácilmente escalable a otros robots, como por ejemplo el G500 que era el objetivo principal. También queda mucho que implementar, ya que se trata de una primera versión de prueba con la que se tiene la intención de seguir trabajando y mejorarla en el futuro.

La comparativa entre ambas pruebas se puede observar en la Figura 36.

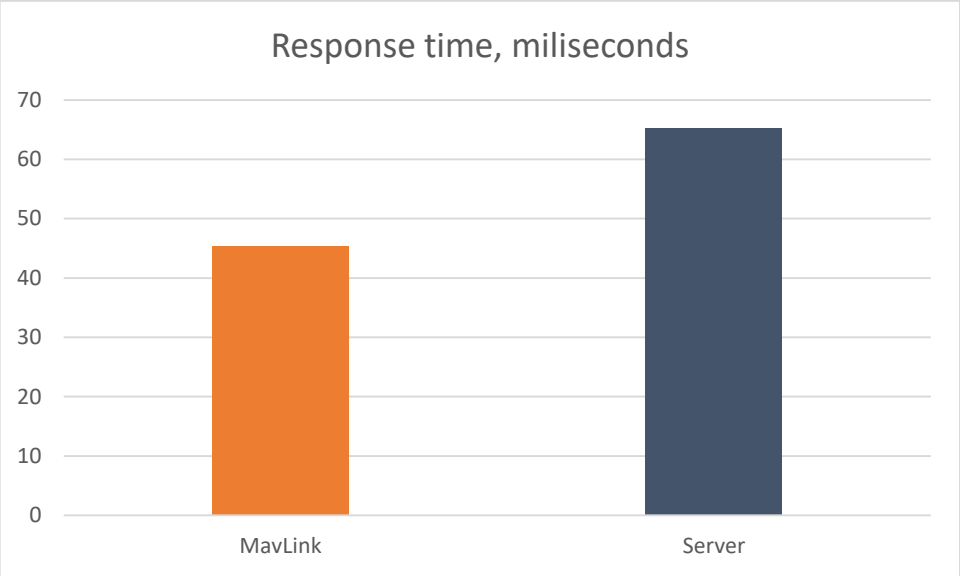


Figura 36. Tiempos de respuesta en milisegundos de las comunicaciones con el robot real.

4.2 Test de usabilidad

El siguiente set de experimentos que se realizó corresponde con los test de usabilidad. Se empezó describiendo los diferentes escenarios que se utilizaron para las pruebas. Todos ellos consistían en realizar el agarre de la caja negra y trasladarla de un punto A al punto B:

1. En el primer escenario o test, las dos cámaras disponibles se encontraban fuera del robot, por lo que el usuario debía realizar la operación viendo en tercera persona al robot. Solo se les decía cuál era el botón para cambiar la cámara y el objetivo de la prueba. El resto lo tenían que descubrir por su cuenta.
2. En el siguiente test, la cámara del brazo ya se colocaba en su posición encima del brazo, para realizar la misma operación.
3. En el tercero, ambas cámaras estaban ya dentro del robot por lo que el usuario tenía que manejar la operación totalmente en primera persona.
4. Por último, en el cuarto test, las cámaras estaban igual que en el test anterior, pero se añadía un obstáculo aleatorio seleccionado entre un set de tres: uno era una pared horizontal, otro un cilindro y el tercero una pared vertical.

Como uno de los principales objetivos con la prueba consistía en observar cuán fácil era adaptarse y aprender a manejar la interfaz, los usuarios recibían una cantidad mínima de información sobre el robot y el brazo de forma verbal. El primer test, gracias a la colocación de las cámaras intentaba permitir a los usuarios crear un modelo mental tanto del escenario como de las dimensiones del robot para que lo utilizaran en las siguientes pruebas.

La primera versión de la interfaz se probó con un grupo total de 25 personas, que consistía en una agrupación heterogénea de estudiantes, investigadores y profesores tanto de ciencia de la computación como de videojuegos. Antes de realizar las pruebas, se les realizaban preguntas relacionadas con su experiencia con entornos de realidad virtual y videojuegos y se generaba un valor entre 0 y 1 en base a las respuestas. Este valor se define como afinidad previa con el sistema y servía para observar, cómo la experiencia previa afectaba al aprendizaje del uso de la interfaz.

Las características de los usuarios, el número de intentos y el tiempo que les costó resolver cada uno de los test, se pueden observar en la Tabla 4.

Table 4. Usuarios de la primera interfaz: Edad, Tiempo de cada test(T1, T2, T3, T4), numero de intentos (Tr) hasta la finalización del test, afinidad (Af) y satisfacción (Pl), la última línea corresponde con el experto humano.

Edad	Género	T1	Tr	T2	Tr	T3	Tr	T4	Tr	Af	Pl
23	Male	352	1	199	1	1182	7	183	1	0.5	7
43	Male	265	1	385	1	395	2	296	2	0.2	8
27	Male	430	1	650	3	201	1	318	1	0.7	9
22	Male	543	3	138	1	156	1	236	1	1	9
28	Male	463	1	286	1	274	1	421	1	0.4	8.5
45	Male	647	3	360	1	558	2	352	1	0.1	7.5
28	Male	250	2	89	1	432	3	367	2	0.5	8
30	Female	965	3	337	1	451	2	724	3	0	8
24	Male	125	1	122	1	139	1	393	4	0.5	9
29	Male	381	4	491	2	676	2	919	4	0.1	8
38	Male	115	1	103	1	325	2	193	1	1	8.5
24	Male	313	2	97	1	256	1	453	4	1	9
54	Female	343	3	167	1	780	3	344	1	0.2	7.5
46	Male	352	1	196	1	744	4	240	1	0.4	8.25
22	Male	192	2	75	1	163	1	141	2	1	9.5
44	Male	430	2	198	1	711	3	811	3	0.2	9
46	Male	197	3	115	1	250	1	167	1	0.4	7
22	Male	144	1	155	1	167	1	153	1	1	9
22	Male	197	3	80	1	167	1	249	2	1	7.75
23	Male	163	1	135	1	150	1	176	1	1	8
24	Male	158	1	161	1	190	1	135	1	0.8	8
44	Female	238	1	174	1	745	5	311	1	0.2	7.75
33	Female	207	1	298	2	300	1	335	1	0.4	7
31	Male	207	3	80	2	324	3	159	2	1	7
48	Male	186	1	238	1	462	1	711	4	0.4	8.5
22	Male	51	1	46	1	46	1	55	1	-	-

El tiempo medio para cada uno de los test se puede observar en la Figura 37, y como se esperaba, la relación entre el tiempo que llevaba completar y el número de intentos está estrechamente relacionado: a más intentos más tiempo se tardaba.

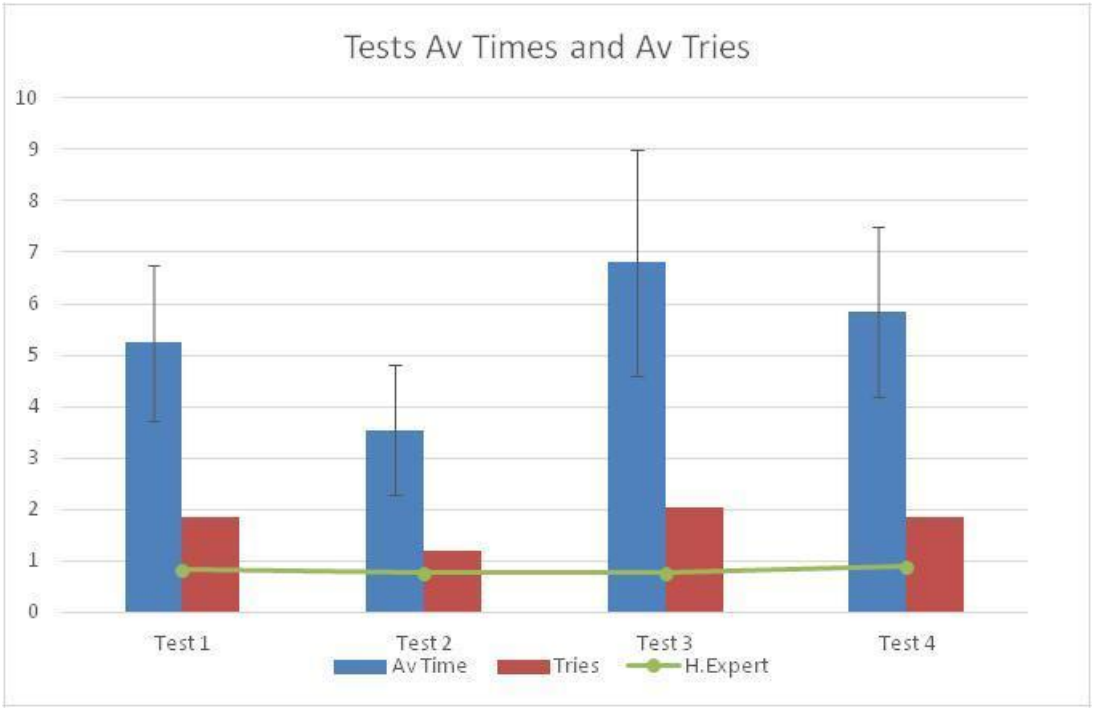


Figura 37. Tiempo medio en minutos(Azul), desviación típica, intentos por cada test(Rojo) y tiempo del usuario experto(linea verde) para la primera version.

Al final de los test se les realizaba a los usuarios un cuestionario inspirado por SUS que tenían que contestar con un valor de 0 a 10:

- ¿Los controles son sencillos de aprender?
- ¿Es el entorno realista?
- ¿Piensas que la interfaz puede resultar útil?
- ¿Qué añadirías a la interfaz?

Las respuestas otorgaron diversa información sobre el grado de satisfacción de los usuarios respecto a la interfaz y la simulación. De cara a la segunda versión las respuestas a la última pregunta tuvieron un gran peso en las decisiones que se realizaron. Por ejemplo, se añadió un tutorial y un medidor de la profundidad. En la figura 38, se puede observar un resumen en forma de media de las respuestas dadas por los usuarios.

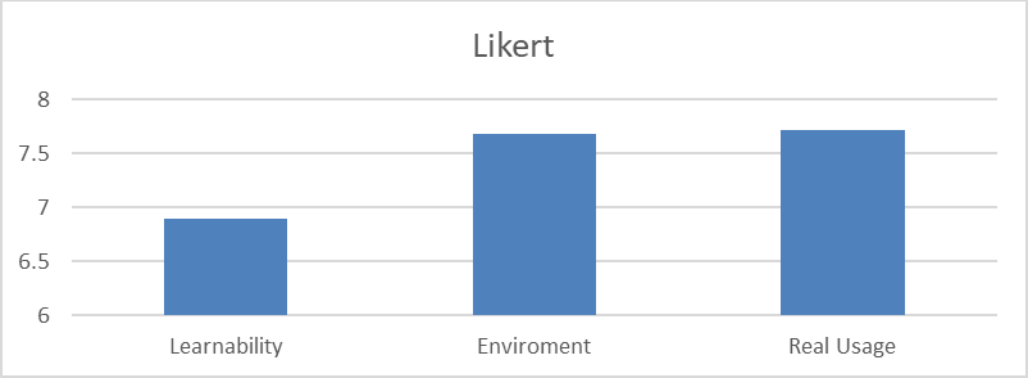


Figura 38. Nivel de statisfacción de los usuarios tras el uso de la primera version de la interfaz.

La figura 38, muestra cómo de sencillo pensaban los usuarios que era aprender los controles de la simulación y utilizarlos de forma eficaz, cuán realista era el entorno de la simulación e inmersiva era la interfaz para ellos y, por último, cómo de útil pensaban que era la simulación.

Para la parte de aprendizaje, la nota otorgada por los usuarios era de 6,9 sobre 10. Teniendo en cuenta que la información previa al uso que se le proporcionaba al usuario era mínima y que tenían que descubrir casi todo por sí mismos, la marca obtenida era superior a la esperada previamente.

En cuanto al entorno, la nota fue de 7,86 sobre 10. Los comentarios recibidos relacionados a esta faceta eran diversos, pero el más interesante fue el de intentar añadir corrientes que aumentaran la dificultad del manejo del robot desplazándolo, que resutló interesante de cara al trabajo futuro. Pero en general, las notas volvían a ser superiores a lo esperado.

Por último, se le preguntaba al usuario cómo de útil pensaban que podía ser la interfaz de cara a un uso real, tanto para entrenamiento de pilotos como para misiones reales. La nota en este aspecto fue de 7,72. Un comentario interesante fue el ajustar la tasa de refresco de la imagen a 1 Hz por segundo, cosa que debido al ancho de banda permitido en las operaciones puede pasar, que el piloto solo reciba una imagen por segundo o incluso una imagen cada pocos segundos por parte de la cámara del robot.

Para algunos de los usuarios, resultaba difícil imaginar el tamaño o la situación espacial del robot dependiendo del momento, sobre todo cuando estaban utilizando el brazo, lo que dificultaba su control. Nos comunicaron que, para ellos, no era suficiente con realizar los dos primeros test para adquirir una imagen mental clara sobre las dimensiones del robot. Este detalle estaba respaldado por las colisiones que se registraron durante el último test, el cual tenía obstáculos, Para el cilindro, que de los tres era el más fácil de esquivar en un principio, los usuarios era con el que más colisionaban como se puede observar en la figura 39. Para solucionar este problema, se realizaron algunas explicaciones sobre el robot y el brazo en el pequeño manual que fueron incluidos de cara a la segunda versión del sistema, para complementar las explicaciones orales que se le daban al usuario. También se crearon varios vídeos para observar de forma clara las dimensiones del robot (https://youtu.be/sLfUisdYlzM, https://youtu.be/LwgQM54GhM0, https://youtu.be/Fzp7_ud9NXA y https://youtu.be/heeDfSPIOzg).

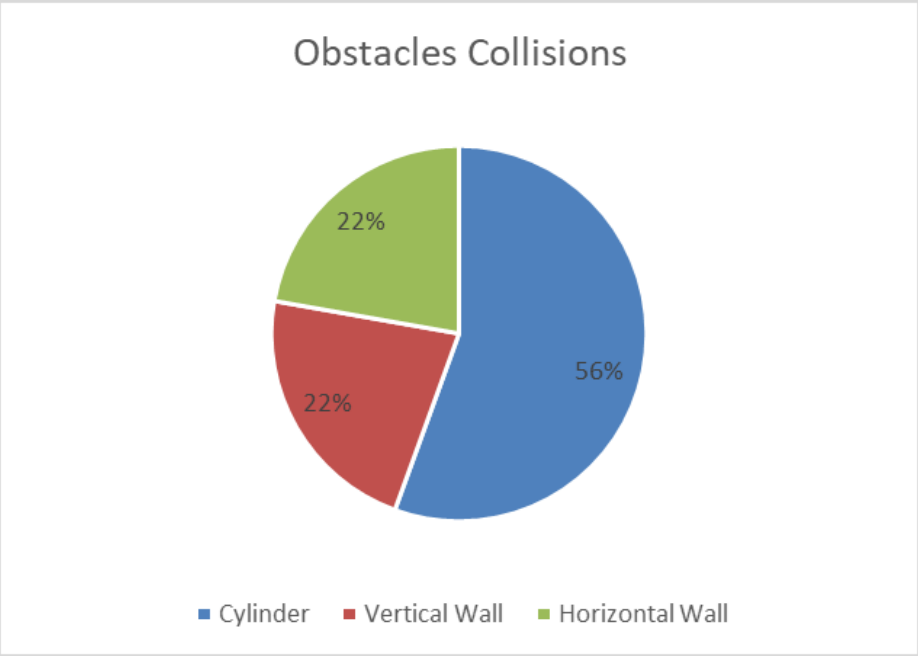


Figura 39. Ratios de collision en el test 4 con los diferentes obstáculos (Cilindro, pared horizontal y pared vertical).

De cara a la segunda versión de la interfaz, se siguió el mismo criterio de selección de los participantes, también se intentó que la mayor cantidad posible de usuarios que habían participado en el primer test volvieran a realizar la prueba con la segunda versión, los resultados se pueden observar en la Tabla 5.

Table 5. Segundo test con la interfaz: Edad, género, tiempo para cada test, número de intent, afinidad previa (Af), satisfacción (Pl) y si formaba parte del primer grupo (R), la última línea se corresponde de Nuevo, con el experto humano.

Edad	Género	T1	Tr	T2	Tr	T3	Tr	T4	Tr	Af	Pl	R
22	Female	207	2	190	1	266	1	180	2	0.1	8.5	No
23	Female	126	1	197	1	446	3	336	1	0.1	8	No
27	Male	135	1	98	1	149	1	263	2	0.7	9.5	Yes
21	Male	75	1	73	1	98	1	226	3	0.8	7	No
25	Male	138	2	68	1	98	1	113	1	0.8	8.85	No
20	Male	82	1	198	2	103	1	336	3	1	7.5	No
28	Male	71	1	83	1	112	1	101	1	0.5	9	Yes
25	Female	133	1	162	1	100	1	173	1	1	6.75	No
22	Male	75	1	101	1	102	1	109	1	0.8	7	No
24	Male	83	1	86	1	89	1	134	1	0.8	9	Yes
46	Male	285	2	241	1	275	1	257	1	0.4	9	Yes
38	Male	165	2	226	1	201	1	278	1	1	9.5	Yes
45	Male	250	1	193	1	233	1	260	1	0.1	8.5	Yes

43	Male	91	1	169	1	290	1	275	1	0.2	9	Yes
24	Male	111	1	76	1	112	1	152	1	0.8	9	Yes
48	Male	186	1	238	1	462	1	711	4	0	8.5	No
30	Female	159	1	145	1	149	1	292	1	0.1	9	Yes
22	Male	25	1	31	1	43	1	47	1			

En la figura 40 podemos observar la media de tiempos para cada uno de los test, de nuevo se puede ver una clara relación entre el número de intentos y el tiempo empleado para completar el test.

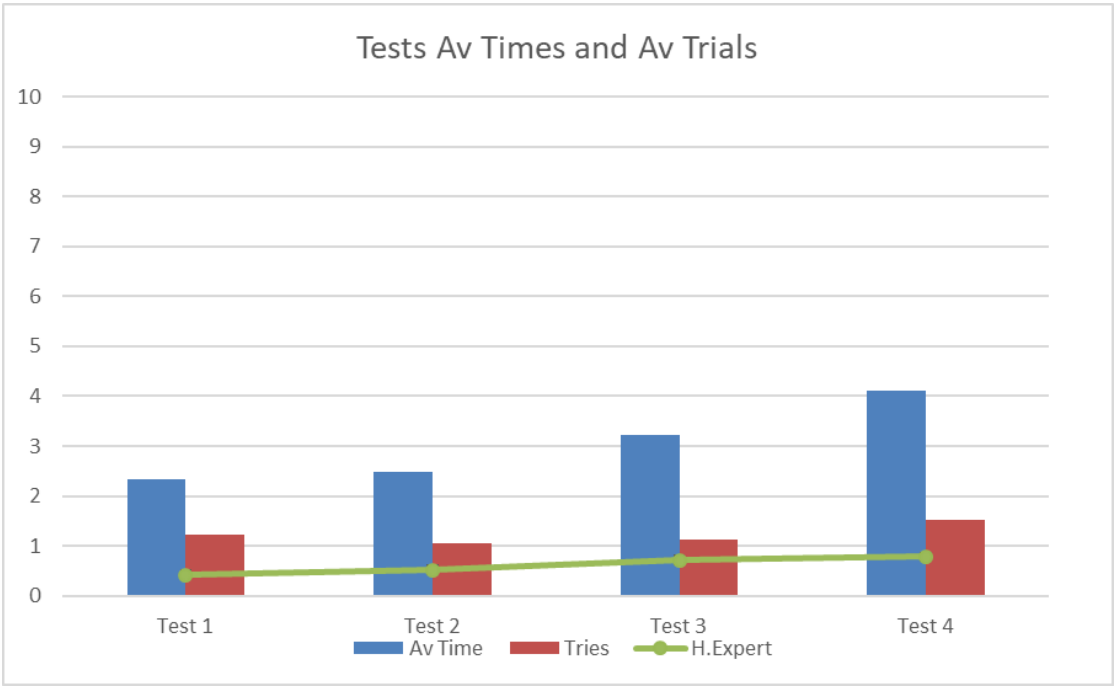


Figura 40. ATiempo medio en minutos(Azul), número de intentos por test(rojo) y datsos del usuario experto(línea verde).

Se pude ver una clara mejora entre ambas versiones si comparamos la Figura 37 y la Figura 40, como resultado de esta comparación se puede observar la Figura 41, la cual recoge el porcentaje de mejora tanto para los tiempos como para el número de intentos entre ambas versiones.

Figura 41. Porcentaje de mejora en los tiempos e intentos respect a la primera versión.

La mejora también puede verse si comparamos los tiempos de los que repitieron los test en ambas versiones, esto queda reflejado en la Figura 42.

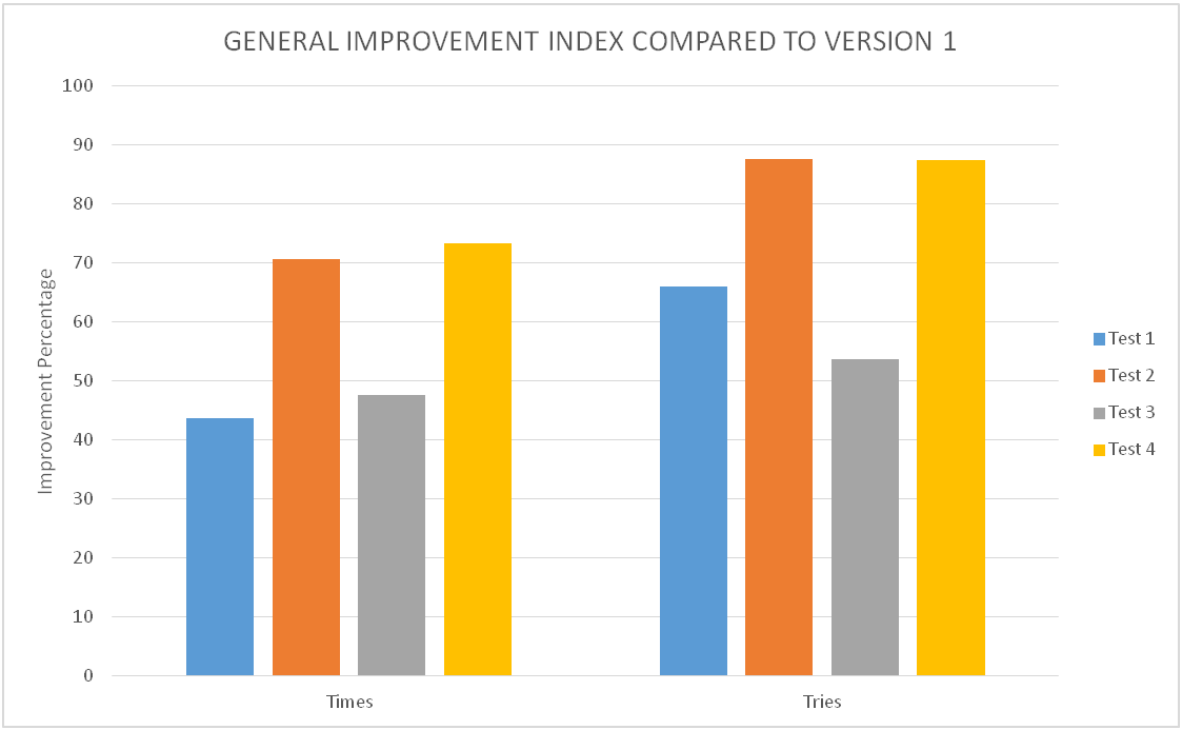


Figura 42. Porcentaje de mejora en tiempo y número de intentos de los usuarios que realizaron los tests en ambas versiones.

Se les realizó las mismas preguntas que en la primera versión a los usuarios, estos se pueden observar en la Figura 43, se consiguieron mejores resultados para todos los aspectos, en cuanto aprendizaje la nota media fue de 8,12, de cara al realismo del entorno la nota fue de 8,78 y por último, de cara al posible uso real de la aplicación la nota fue de 8,45. Algunos comentarios

resultaban contradictorios, por ejemplo, respecto a los sonidos de aviso, para algunos usuarios resultó una herramienta útil mientras que para otros resultaba una molestia. También se nos sugirió el colocar un avatar virtual del usuario dentro del entorno en una cabina virtual que estuviera atada al vehículo.

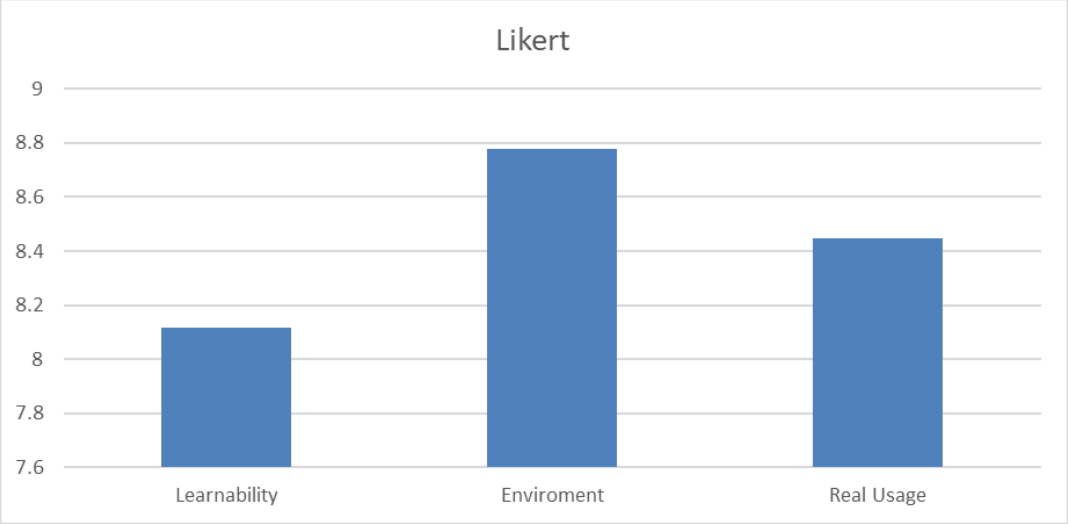


Figura 43. Resultados de satisfacción en la segunda versión.

También, la distribución de las colisiones cambió de forma drástica, mientras que en la primera versión la gente colisionaba en algunas ocasiones con la pared horizontal, en la segunda versión esto ya no pasaba como se puede observar en la Figura 44, aunque el cilindro seguía dando la mayor cantidad de problemas a los usuarios, la mayoría encontraron los sonidos de aproximación muy útiles para evitar las colisiones.

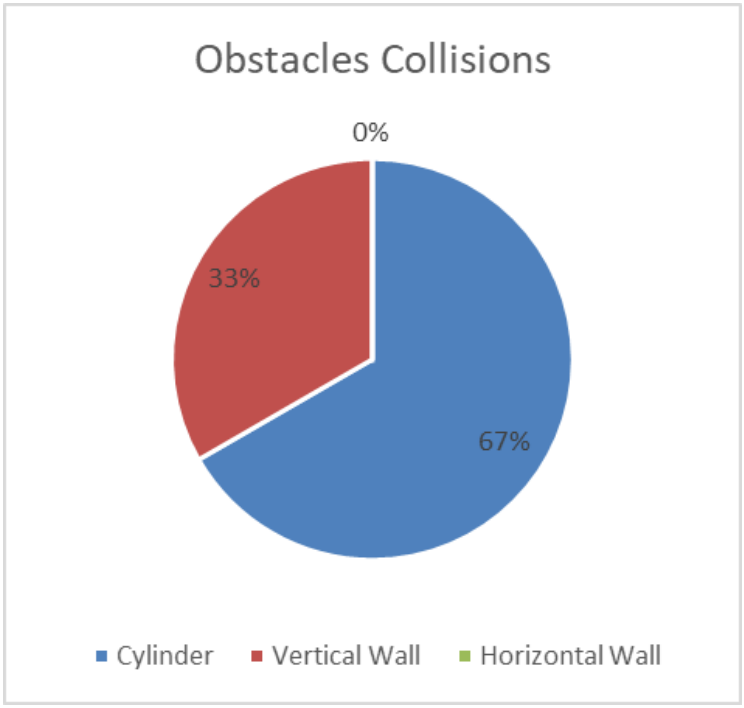


Figura 44. Distribución de las colisiones en la segunda versión.

Dado que tuvimos la suerte de contar con algunos de los usuarios que realizaron el primer conjunto de pruebas [25], también en la segunda versión, nos tomamos la libertad de entrevistarlos en mayor profundidad con algunos resultados interesantes.

El primer sujeto, nos comunicó que prefería con creces el sistema HTC Vive, dado que con los sistemas Oculus que previamente había utilizado sentía mareos. Tuvo también que parar la prueba debido a esos mareos, lo cual es un problema bastante común a la hora de utilizar sistemas de RV [44][45]. Durante el uso de la segunda versión el sujeto no sintió los mareos en ningún momento.

El segundo sujeto afirma que si bien le parece mejor el sistema HTC Vive, el sistema de Oculus es más fácil de utilizar debido a que otorga más información. También nos dijo que prefería utilizar un joystick complejo en lugar de los dos controladores de HTC. Si bien el usuario terminó también la segunda versión pensando que prefería el uso de un joystick, nos dejó claro que los controles le parecían mejor distribuidos en la segunda versión respecto a la primera.

El tercer sujeto que entrevistamos nos informó que prefería usar UWSim y un joystick mediante un sistema Oculus. Consideraba que el tener dos controladores para las tareas que realizaba el robot que aparentemente eran sencillas, era excesivo, aunque podría llegar a ser útil para tareas más complejas.

En todos los casos, los sujetos que repitieron el experimento preferían la segunda versión a la primera.

5.3 Eficiencia

De acuerdo a ISO-9241 (https://www.iso.org/standard/77520.html) *Ergonomics of human-system interaction*, la eficiencia de un producto se puede definir como *"los recursos utilizados por el usuario para completar de forma precisa una tarea"*.

Independientemente del tipo de software o sistema de información, tomán como medida principal el tiempo que el usuario emplea para conseguir un objetivo.

Por lo que, la eficiencia puede calcularse como la efectividad dividida por el tiempo empleado por el usuario.

Siendo N el número de escenarios (4 en nuestro caso).

R el número de usuarios (25 para la primera versión, 17 para la segunda)

n_{ij} es el resultado de pasar el escenario i, por parte de un usuario j, el valor será 1 si se completa la tarea y 0 si no.

t_{ij} se corresponde con el tiempo que tarda el usuario j en completar el escenario i, en caso de que no se completara el escenario, el tiempo hasta que el usuario abandona dicho escenario.

Entonces, la media de eficiencia basada en el tiempo de un producto P_t se podría calcular de acuerdo a la fórmula (1) en la que una eficiencia perfecta (todos los test terminados de forma completa en un segundo, que es el tiempo unidad usado), debería ser 1 y 0 en el caso de no ser capaz de finalizar la tarea:

$$\bar{P}_t = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{n_{ij}}{t_{ij}}}{RN} \quad (1)$$

La eficiencia para la primera versión del sistema resultaría ser de 0,004131 y de 0,007697 para la segunda versión. Si utilizamos el tiempo utilizado por algunos usuarios en los primeros y fallidos intentos los resultados serían peores, de 0,003642 para la primera versión y 0,006470 para la segunda.

Podemos calcular la media de la eficiencia relativa al tiempo utilizando la fórmula (2):

$$\overline{\mathrm{P}} = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N} n_{ij}t_{ij}}{\sum_{j=1}^{R}\sum_{i=1}^{N} t_{ij}} * 100\% \quad (2)$$

Teniendo en cuenta que no tiene sentido el utilizar solamente los intentos en los que el usuario es capaz de completar la tarea (porque daría 100% de eficiencia), los resultados que hemos obtenido son los siguientes, para la primera versión una eficiencia del 69,74% y para la segunda una eficiencia de 91,82%.

El creador de la simulación y la interfaz fue tomado como el usuario experto para las pruebas, teniendo en cuenta sus resultados como los mejores valores obtenibles en cuanto a porcentaje de superación de escenarios y velocidad del mismo:

Tomamos t_{0i} como el tiempo ideal que le llevaría al experto completar el escenario i.

Entonces, utilizando la fórmula (3) obtendríamos que la eficiencia del experto con la simulación, la cual sería la máxima obtenible en las tareas sería de 0,0203 para la primera versión y de 0,0292 para la segunda.

$$\overline{P_{te}} = \frac{\sum_{i=1}^{N}\frac{1}{t_{0i}}}{N} \quad (3)$$

Con todos estos datos concluimos que la eficiencia de la segunda versión es superior en todos los aspectos a la primera.

6. Conclusiones

Durante todo este documento hemos podido observar que la capacidad inmersiva de la realidad virtual está a un nivel muy diferente del resto de tecnologías actuales y hemos sido capaces de desarrollar una interfaz para controlar un robot submarino que ha sido testeada y ya se encuentra en un estado funcional. Comparada con el tipo de interfaces que encontramos en el mercado actualmente hemos podido observar que para los usuarios resulta mucho más sencilla de manejar y aprender, pues para nuestra interfaz con tan solo diez minutos de pilotaje ya han sido capaces de realizar una tarea relativamente compleja como es la aproximación y el agarre de una caja negra, mientras que para el uso de las interfaces actuales el tiempo de aprendizaje es mucho mayor debido a la cantidad de información a la que tienen que prestar atención y la distribución de la misma.

Gracias a los test de usabilidad que realizamos, fuimos capaces de tomar en consideración diferentes aspectos que no se nos habían pasado por la cabeza a la hora de desarrollar la primera versión de la interfaz como el hecho de crear un modo con una tasa de refresco de imágenes inferior a una cada segundo, dado que en la actualidad es bastante improbable conseguir esa tasa de refresco con la calidad a la que el entorno de la primera versión permitía y mucho menos la tasa de refresco de casi 60 Hz que teníamos.

En la actualidad el sistema no cuenta con ningún modelo de incertidumbre que pueda simular el movimiento propio de las corrientes marinas por ejemplo, en las pruebas con el *BlueROV* nos hemos dado cuenta que por la forma en que el agua actúa, aún en una piscina como es la del CIRTESU existe un pequeño movimiento del agua que puede hacer que el robot se desplace sin necesidad de que el usuario active los propulsores. Esta parte es algo que estamos planeando añadir de cara al desarrollo final del sistema.

Otro factor a tener en cuenta es, que al trabajar con personas para los test, siempre existe un cierto grado de subjetividad debido en gran medida a experiencias pasadas, hemos intentado reducir este factor subjetivo realizando el estudio con la mayor cantidad de gente que nos ha sido posible, un total de 39 sujetos si tenemos en cuenta los tres que participaron en las dos versiones. Por otro lado el tiempo y número de intentos que eran necesarios para realizar los test son un buen indicativo de la utilidad real de la interfaz.

En el futuro hay una serie de factores que nos gustaría añadir a la interfaz:

- Por un lado está el realizar la implementación del G500 en el sistema, cosa que podremos comenzar a implementar en las próximas semanas cuando el robot esté en el CIRTESU.
- Otro detalle es implementar un segundo robot a la simulación y un sistema cooperativo dentro de la misma de cara a que uno de los robots tenga el rol de líder mientras el otro tiene el rol de seguidor, y que estos roles se inviertan cuando el usuario quiera. Paralelamente a este factor, podría ser interesante que dos usuarios pudieran realizar un control simultaneo dentro de la aplicación y que cada uno se encargara de uno de los robots de la misión.
- Conseguir integrar por completo todos los sensores que posee el robot dentro de la interfaz para que ésta pasea a ser una interfaz de realidad mixta, y no solo exclusivamente de realidad virtual como es actualmente.

- También queremos explorar aún más conceptos para reducir la fatiga de los usuarios pues la propia RV ya genera por si sola un cierto grado de fatiga por diferentes razones [46] y el objetivo principal de la interfaz es precisamente reducir al mínimo la fatiga.

Por supuesto, estas no son las únicas posibilidades de futuro para el proyecto. También podría ser interesante el añadir una inteligencia artificial para reducir el grado de intervención del usuario como ya se ha hecho en algunos sistemas híbridos [12]-

Otra línea interesante podría ser integrar otro tipo de herramientas para intentar ayudar al usuario a la hora de realizar la tele operación [25]. Por ejemplo, podríamos utilizar el propio micrófono incorporado en las *HTC Vive* para permitir que el usuario realice algunas acciones a través de la voz, como cambiar el modo de control.

Por último mencionar que existe una carpeta de drive en la cual se pueden acceder a varios vídeos de la simulación, entre ellos uno de las pruebas que realizamos de tiempo de respuesta en el CIRTESU[48].

7. Referencias

1. Lin, Q.; Kuo, C. On Applying Virtual Reality to Underwater Robot Tele-Operation and Pilot Training. *Int. J. Virtual Real.* **2001**, *5*, 71–91, doi:10.20870/IJVR.2001.5.1.2670.
2. Ridao, P.; Carreras, M.; Ribas, D.; Sanz, P.J.; Oliver, G. Intervention AUVs: The next challenge. *Annu. Rev. Control.* **2015**, *40*, 227–241.
3. Yuh, J.; Choi, S.K.; Ikehara, C.; Kim, G.H.; Me Murty, G.; Ghasemi-Nejhad, M.; Sarlear, N.; Sugihara, K. Design of a semi-autonomous underwater vehicle for intervention missions (SAUVIM). In Proceedings of the 1998 International Symposium on Underwater Technology, Tokyo, Japan, 15–17 April 1998; IEEE Press: Hoboken, NJ, USA, 1998; pp. 63–68.
4. Sanz, P.J.; Ridao, P.; Oliver, G.; Melchiorri, C.; Casalino, G.; Silvestre, C.; Petillot, Y.; Turetta, A. TRIDENT: A Framework for Autonomous Underwater Intervention Missions with Dexterous Manipulation Capabilities. *IFAC Proc. Vol.* **2010**, *43*, 187–192.
5. Khatib, O.; Yeh, X.; Brantner, G.; Soe, B.; Kim, B.; Ganguly, S.; Stuart, H.; Wang, S.; Cutkosky, M.; Edsinger, A.; et al. Ocean One: A Robotic Avatar for Oceanic Discovery. *IEEE Robot. Autom. Mag.* **2016**, *23*, 20–29, doi:10.1109/MRA.2016.2613281.
6. Prats, M.; Perez, J.; Fernandez, J.J.; Sanz, P.J. An open source tool for simulation and supervision of underwater intervention missions. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2012, Algarve, Portugal, 7–12 October 2012; Curran Associates, Inc.: Red Hook, NY, USA, 2012; pp. 2577–2582.
7. Miller, G. The magical number seven, plus or minus two: Some limits on our capabilities for processing information. *Psychol. Rev.* **1956**, *63*, 81–97.
8. Sanz, P.J.; de la Cruz, M.; Lunghi, G.; Veiga, C.; Marín, R.; Di Castro, M. The Role of HRI within COMOMUIS Research Project. In Proceedings of the Jornadas Nacionales de Robótica. Alicante, Spain, 13–14 June 2019; Fernando, T.M., Óscar, R.G., Eds.; Universidad de Alicante: Alicante, Spain, 2019; pp. 141–147.
9. Preece, J.; Rogers, Y.; Sharp, H.; Benyon, D.; Holland, S.; Carey, T. *Human-Computer Interaction*; Addison-Wesley Longman Ltd.: Essex, UK, 1994; ISBN:0201627698.
10. Sheridan, T.B.; Verplank, W.L. *Human and Computer Control of Undersea Teleoperators*; Technical Report; Massachusetts Inst of Tech Man-Machine Systems Lab: Cambridge, MA, USA, 1978. Available online: https://apps.dtic.mil/dtic/tr/fulltext/u2/a057655.pdf (accessed on 3 May 2019).
11. Peshkova, E.; Hitz, M.; Kaufmann, B. Survey on Natural Interaction Techniques for an Unmanned Aerial Vehicle System. *IEEE Pervasive Comput.* 2017, *16*, 34–42, doi:10.1109/MPRV.2017.3.
12. Chen, J.Y.C.; Haas, E.C.; Barnes, M.J. Human Performance Issues and User Interface Design for Teleoperated Robots. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* 2007, *37*, 1231–1245.
13. Dicianno, B.E.; Sibenaller, S.; Kimmich, C.; Cooper, R.A.; Pyo, J. Joystick Use for Virtual Power Wheelchair Driving in Individuals with Tremor: Pilot Study. *J. Rehabil. Res. Dev.* **2009**, *46*, 269–75.
14. Huang, C.M.; Mutlu, B. Anticipatory robot control for efficient human–robot collaboration. In Proceedings of the 2016 11th ACM/IEEE International Conference on Human–robot Interaction (HRI), Christchurch, New Zealand, 7–10 March 2016; IEEE Press: Hoboken, NJ, USA, 2016; pp. 7–10.
15. Shim, H.; Jun, B.; Lee, P.; Baek, H.; Lee, J. Workspace control system of underwater tele-operated manipulators on an ROV. *Ocean Eng.* **2010**, *37*, 1036–1047, doi:10.1016/j.oceaneng.2010.03.017.
16. Sutherland, I.E. The ultimate display. In Proceedings of the International Federation of Information Processing IFIPS Congress, New York, NY, USA, 24-29 May 1965; Volume 2, pp. 506–508.
17. Rheingold, H. *Virtual Reality*; Summit Books: New York, USA 1991.
18. Chin, C.; Lin, W.; Lin, J. Experimental validation of open-frame ROV model for virtual reality simulation and control. *J. Mar. Sci. Technol.* **2018**, *23*, 267–287, doi:10.1007/s00773-017-0469-3.
19. Anthes, C.; García Hernandez, R.; Wiedemann, M.; Kranzlmüller, D. State of the Art of Virtual Reality Technologies. In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 5–12 March 2016, doi:10.1109/AERO.2016.7500674.
20. Burdea, G.; Coiffet, P. Virtual Reality Technology. *Presence* **2003**, *12*, 663–664, doi:10.1162/105474603322955950.

21. Bowman, D.; North, C.; Chen, J.; Polys, N.; Pyla, P.; Yilmaz, U. Information–rich virtual environments: Theory, tools, and research agenda. In Proceedings of the VRST '03 ACM Symposium on Virtual Reality Software and Technology, Osaka, Japan, 1-3 October 2003; pp. 81–90, doi:10.1145/1008653.1008669.
22. Sherman, W.; Craig, A. *Understanding Virtual Reality—Interface, Application, and Design*; Elsewier Science: New York, NY, USA, 2003.
23. Lin, M.C.; Otaduy, M.A.; Boulic, R. Virtual reality software and technology. *IEEE Comput. Graph. Appl.* **2008**, *28*, 18–19.
24. Kot, T.; Novak, P. Utilization of the Oculus Rift HMD in Mobile Robot Teleoperation. *Appl. Mech. Mater.* **2014**, *555*, 199–208, doi:10.4028/www.scientific.net/AMM.555.199.
25. García, J.C.; Patrão, B.; Almeida, L.; Pérez, J.; Menezes, P.; Dias, J.; Sanz, P.J. A Natural Interface for Remote Operation of Underwater Robots. *IEEE Comput. Graph.* **2015**, *37*, 34–43, doi:10.1109/MCG.2015.118.
26. Kot, T.; Novak, P. Application of virtual reality in teleoperation of the military mobile robotic system TA-ROS. *Int. J. Adv. Robot. Syst.* **2018**, *15*, 1729881417751545, doi:10.1177/1729881417751545.
27. Schiza, E.; Matsangidou, M.; Neokleous, K.; Pattichis, C. Virtual Reality Applications for Neurological Disease: A Review. *Front. Robot. AI* **2019**, *6*, 100, doi:10.3389/frobt.2019.00100.
28. Rizzo, A.S.; Koenig, S.T. Is clinical virtual reality ready for primetime? *Neuropsychology* **2017**, *31*, 877–899, doi:10.1037/neu0000405.
29. Tan, Y.; Niu, C.; Zhang, J. Head-Mounted Display-Based Immersive Virtual Reality Marine-Engine Training System. *IEEE Syst. Man Cybern. Mag.* **2020**, *6*, 46–51, doi:10.1109/MSMC.2019.2948654.
30. Hsu, E.B.; Li, Y.; Bayram, J.D.; Levinson, D.; Yang, S.; Monahan, C. State of virtual reality based disaster preparedness and response training. *PLoS Curr.* **2013**, *5*, doi:10.1371/currents.dis.1ea2b2e71237d5337fa53982a38b2aff.
31. Engelbrecht, H.; Lindeman, R.; Hoermann, S. A SWOT Analysis of the Field of Virtual Reality for Firefighter Training. *Front. Robot. AI* **2019**, *6*, 101, doi:10.3389/frobt.2019.00101.
32. Haydar, M.; Maidi, M.; Roussel, D.; Mallem, M.; Drap, P.; Bale, K.; Chapman, P. Virtual Exploration of Underwater Archaeological Sites: Visualization and Interaction in Mixed Reality Environments. In Proceedings of the 9th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage, Braga, Portugal, 2–5 December 2008; Eurographics Ass: Geneva, Switzerland, 2008, doi:10.2312/VAST/VAST08/141-148.
33. Bekele, M.; Champion, E. A Comparison of Immersive Realities and Interaction Methods: Cultural Learning in Virtual Heritage. *Front. Robot. AI* **2019**, *6*, 91, doi:10.3389/frobt.2019.00091.
34. Azuma, R.T. A Survey of Augmented Reality. *Presence Teleoper. Virtual Environ.* **1997**, *6*, 355–385, doi:10.1162/pres.1997.6.4.355.
35. Brantner, G.; Khatib, O. Controlling Ocean One. In *Field and Service Robotics*; Springer International Publishing: Cham, Switzerland, 2018; pp. 3–17, doi:10.1007/978-3-319-67361-5_1.
36. Gancet, J.; Weiss, P.; Antonelli, G.; Folkert Pfingsthorn, M.; Calinon, S.; Turetta, A.; Walen, C.; Urbina, D.; Govindaraj, S.; Letier, P.; et al. Dexterous Undersea Interventions with Far Distance Onshore Supervision: The DexROV Project. *IFAC-PapersOnLine* **2016**, *49*, 414–419.
37. Abran, A.; Khelifi, A.; Suryn, W.; Seffah, A. Usability Meanings and Interpretations in ISO Standards. *Softw. Qual. J.* **2003**, *11*, 325–338.
38. Seffah, A.; Donyaee, M.; Kline, R.B.; Padda, H.K. Usability measurement and metrics: A consolidated model. *Softw. Qual. J.* **2006**, *14*, 159–178, doi:10.1007/s11219-006-7600-8.
39. Brooke, J. SUS-A quick and dirty usability scale. In *Usability Evaluation in Industry*; Jordan, P.W., Thomas, B., McClelland, I.L., Weerdmeester, B., Eds.; Taylor & Francis, Milton Park: Abingdon-on-Thames, Oxfordshire, UK, 1996; pp. 189–194.
40. Marsh, T. *Evaluation of Virtual Reality Systems for Usability*; Proceedings CHI 99, 15-20 May 1999; ACM: New York, USA 1999; pp. 61–62, doi:10.1145/632716.632756.
41. Unity User Manual. Available online: https://docs.unity3d.com/Manual/index.html (accessed on 3 March 2019).

42. Fernández, J.J.; Prats, M.; Sanz, P.J.; García, J.C.; Marín, R.; Robinson, M.; Ribas, D.; Ridao, P. Grasping for the Seabed: Developing a New Underwater Robot Arm for Shallow-Water Intervention. *IEEE Robot. Autom. Mag.* **2013**, *20*, 121–130.
43. Feiner, S.; MacIntyre, B.; Haupt, M.; Solomon, E. Windows on the world: 2D windows for 3D augmented reality. In Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology, Atlanta, GA, USA, 3-8 December 1993; pp. 145–155; doi:10.1145/168642.168657.
44. McCauley, M.; Sharkey, T. Cybersickness: Perception of Self-motion in Virtual Environments. Teleoperators and Virtual Environments. *Presence* 1989, *3*, 311–318.
45. LaViola, J.J., Jr. A discussion of cybersickness in virtual environments. *ACM SIGCHI Bull.* **2000**, *32*, 47–56, doi:10.1145/333329.333344.
46. Lambooij, M.; Ijsselsteijn, W.; Fortuin, M.; Heynderickx, I. Visual Discomfort and Visual Fatigue of Stereoscopic Displays: A Review. *J. Imaging Sci. Technol.* **2009**, *53*, 30201-1, doi:10.2352/J.ImagingSci.Technol.2009.53.3.030201.
47. de la Cruz, M; Casañ, G, A; Sanz, P, J; Marín, R; Preliminary Work on a Virtual Reality Interface for the Guidance of Underwater Robots. *Robotics* **2020,** 9(4), 81.
48. De la Cruz, M; Shared drive folder with interface videos and demos. https://drive.google.com/drive/folders/12tvV_IQXk6vMWv98-LTkgOCrltYk1M0-?usp=sharing