https://doi.org/10.17979/spudc.9788497498081.1095

MEJORAS EN LA ARQUITECTURA DE CONTROL E INTERFAZ DE USUARIO PARA EL CONTROL DE UN HROV

J.C. García[1], J. Pérez[1], J. J. Fernández[1], P. Menezes[2], P.J. Sanz[1]
[1]Dep. de Ingeniería y Ciencia de los Computadores, Universitat Jaume I, Castellón de la Plana, España
{garciaju, japerez, fernandj, sanzp}@uji.es
[2]Institute of Systems and Robotics, Dep. of Electrical and Computer Engineering, University of Coimbra, Portugal
paulo@isr.uc.pt

Resumen

En la actualidad, la mayor parte de las misiones de intervención llevadas a cabo en ambientes submarinos, son realizadas por robots teleoperados conocidos como Remote Operated Vehicles (ROV). Sin embargo, debido a sus elevados costes económicos y logísticos, recientemente se están empezando a usar un nuevo sistema autónomo conocido como Intervention Autonomous Underwater Vehicles (I-AUV). No obstante, estos sistemas omiten al usuario dentro del bucle de control, que unido a ciertas restricciones operacionales, limitan su uso en algunas intervenciones. Con el fin de combinar las ventajas y minimizar los problemas de ambos sistemas, existe una nueva propuesta basada en un enfoque híbrido, en la que el robot puede permitir el cambio de modo de uso (i.e. autónomo-teleoperado). No obstante, los problemas de este nuevo enfoque se hallan en la arquitectura de control y la interacción hombre-robot (HRI).

Palabras clave: robótica submarina, teleoperación, ROV, AUV, HROV, interfaz de usuario, arquitectura de control, control híbrido ROV

1. INTRODUCCIÓN

Existen múltiples aplicaciones en las que un robot teleoperado es muy útil para la ejecución de tareas, pero esta utilidad es aún más importante cuando se trata de un entorno hostil para los seres humanos. Podemos encontrar ejemplos de entornos hostiles en los programas espaciales (e.g. exploración de otros planetas o en el trabajo en los satélites), militar / defensa (e.g. el control de un vehículo no tripulado, ya sea aéreo o terrestre), en problemas de seguridad (e.g. robots para la desactivación de bombas o robots de telepresencia [1]), en operaciones bajo el agua (e.g. topografía del fondo marino, batimetrías o la inspección de una presa) o en el campo médico (e.g. telecirugía). Por otro lado, los robots totalmente autónomos se encuentran todavía en sus primeras fases de desarrollo [2], principalmente debido a las dificultades para adaptarse, en tiempo real, a los cambios que se producen en su entorno.

En la actualidad, la mayoría de las actividades en entornos submarinos se llevan a cabo con robots teleoperados, dadas las ventajas sobre los robots autónomos, como la posibilidad de alcanzar mayores profundidades o su mayor versatilidad a la hora de realizar una intervención (e.g. manipular un objeto). No obstante, la principal ventaja de los robots autónomos frente a los teleoperados es la descarga de estrés [3] por parte del usuario durante la teleoperación [4]. Así, la innovación más reciente en robots submarinos son los llamados ROV-híbrido (HROV), ya que pueden funcionar tanto en modo autónomo como teleoperado, aunando las principales ventajes de ambos modos de control. No obstante, una de las principales limitaciones de este tipo de robots, es la arquitectura en la que se basan, ya que no suelen permitir un cambio de modo sencillo.

Las ventajas de usar un HROV son diversas:

- La capacidad del robot para hacer partes de la misión de manera autónoma reduce la fatiga cognitiva y el estrés del usuario, así como los fallos humanos [5].
- Las operaciones complejas que son difíciles de realizar de forma autónoma por el robot, pueden ser ejecutadas por el usuario de forma teleoperada
- El usuario puede supervisar el funcionamiento autónomo del robot, tomando el control del mismo en cualquier momento para prevenir y evitar cualquier problema.

No obstante, los principales problemas a la hora de aplicar dicho enfoque híbrido se concentran en el HRI y en la arquitectura de control.

Este artículo presenta un nuevo enfoque para controlar un HROV. Este enfoque incluye una arquitectura centrada en la reducción de la responsabilidad del usuario durante la misión, garantizando que el robot alcance con éxito el objetivo de la misión, y mejorando la transición entre los modos de trabajo autónomo y teleoperado.

El resto de este artículo se organiza de la siguiente manera: La sección 2 describe los detalles de implementación de la arquitectura propuesta. En la sección 3 se presenta la interfaz inmersiva que facilita el HRI. Por último se muestra el trabajo en curso y se ofrecen algunas conclusiones del trabajo expuesto.

2. DESCRIPCIÓN DE LA ARQUITECTURA

Tal y como se ha mencionado anteriormente, una de las ventajas del uso de un HROV es la posibilidad de operar de forma autónoma o teleoperada. A fin de mejorar la experiencia de usuario y ampliar las funcionalidades del HROV, es interesante que el usuario pueda controlar el robot una vez se ha iniciado la misión. Así, tras especificar qué debe realizar el robot, el usuario supervisa la misión hasta que decide tomar el control del robot. De este modo, cuando el robot opera de forma autónoma, se mantienen las ventajas de tener al usuario dentro del bucle de control (e.g. interpretar información compleja proveniente de diferentes fuentes), mientras que a la vez se reduce la fatiga cognitiva típica de un piloto de ROV, que es uno de los problemas más importantes a la hora de teleoperar.

Este nuevo método de control supone nuevos retos desde el punto de vista del HRI, pues se vuelve imprescindible una arquitectura y mecanismos de control entre las diferentes partes de la arquitectura. Usando este enfoque, se distinguen tres posibles situaciones:

- Mejor caso: el robot completa la misión satisfactoriamente sin la intervención del usuario.
- Caso intermedio: el robot realiza la misión, pero el usuario detecta un problema o decide modificar la misión.
- Peor caso: el robot no puede completar la misión satisfactoriamente, cancela la misión y es el usuario quien debe finalizar la misma de forma totalmente teleoperada.

En las dos últimas opciones, el usuario debe tener la posibilidad de comunicarse con el robot y poder teleoperarlo. La diferencia entre ambos casos es que en el caso intermedio, el usuario devuelve el control al robot una vez finaliza su intervención (e.g. evitar un obstáculo o modificar la ruta), mientras que en el último caso, el robot aborta la misión y es el usuario quien debe finalizar la misión.

A fin de lograr este tipo de control compartido (autónomo - teleoperado) y que el usuario pueda cambiar sobre la marcha el nivel de la autonomía del robot, es necesario un nuevo enfoque y arquitectura que lo soporte. En esta sección, se detalla una propuesta de arquitectura (ver Fig. 1) que permite dicho control de la autonomía del robot, moviendo la responsabilidad de la tarea del robot al usuario (o viceversa) dependiendo de la situación, o en el caso de que el usuario solicite explícitamente el control del robot.

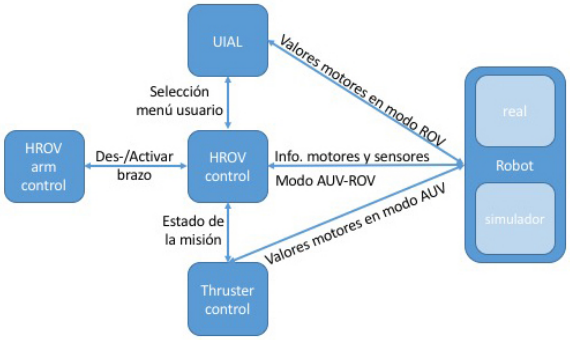

Figura 1: Estructura de la nueva arquitectura propuesta.

Debido a las dificultades para poner a prueba el sistema en entornos reales, la Universitat Jaume I (UJI) comenzó a desarrollar en 2011 una nueva herramienta de software de código abierto para la planificación, visualización y simulación de misiones robóticas submarinas: UWSim [6]. Este simulador fue creado en el proyecto financiado por la comisión europea [7], además ha sido utilizado en otros proyectos europeos como [8] o [9]. A fin de mejorar la integración del simulador y el robot en la misma arquitectura, es casi imprescindible el uso de un middleware que facilite dicha integración. Debido a la experiencia previa en nuestro laboratorio (IRSLab, UJI), su estabilidad, y su amplia aceptación en la comunidad de investigadores, se seleccionó el *middleware* Robot Operative System (ROS) [10]. El uso de ROS no sólo facilita la integración de los diferentes componentes y robot, sino que también proporciona una abstracción de la capa de comunicación. Una ventaja de esta abstracción es que UWSim se puede utilizar para la visualización, en tiempo real, de todos los datos del robot relacionados con la misión.

2.1. UIAL

La capa de abstracción de usuario (User Interface Abstraction Layer), ya fue presentada anteriormente en [11]. En la actualidad, este módulo ha sufrido diversas modificaciones para ampliar su funcionalidad, pero manteniendo la idea original:

- El usuario puede controlar diversos dispositivos de entrada, aunque no todos a la vez.
- Permite el control del vehículo con o sin dinámica, principalmente para su uso junto al simulador UWSim.
- Controla y publica información relativa a cuando el usuario requiere el control teleoperado del vehículo o del brazo.
- Recibe los datos de los sensores del robot, tanto reales como simulados.

2.2. AUV CONTROL

Esta parte del sistema es la encargada de controlar el vehículo de manera autónoma. Por tanto recibe órdenes de HROV CONTROL, el cual decide cuándo el vehículo debe actuar de manera autónoma. Recibe por tanto un objetivo en forma de posición y trata de alcanzarlo de la forma más eficiente posible monitorizando señales que indiquen que el usuario ha tomado el mando y debe en consecuencia dejar el control del vehículo.

La complejidad de éste módulo de la arquitectura dependerá por tanto de la necesidad y/o objetivo de la tarea a realizar. Ciertas misiones requieren un control muy preciso como la arqueología submarina mientras que otras pueden necesitar un resultado más reactivo como biología submarina. En este trabajo se presenta una implementación sencilla orientada a un objetivo general, no demasiado exigente, como puede ser una reconstrucción del terreno o inspeccionar un objeto de interés.

El sistema consta de dos controladores en cascada. El primero obtiene una referencia de velocidad dependiendo de la posición del vehículo mientras que el segundo intenta alcanzar esta velocidad mandando la fuerza a los motores teniendo en cuenta la velocidad actual del vehículo.

El controlador de posición ha sido implementado como un P utilizando la posición estimada por la odometría del vehículo y la posición objetivo. Hay que tener en cuenta que la odometría produce deriva y por tanto una medida absoluta es recomendable en misiones largas donde la posición real del vehículo puede ser muy diferente a la obtenida por odometría.

El controlador de velocidad es similar, pero en lugar de usar la estimación de odometría utiliza un sensor DVL. Además es necesario establecer la relación entre la configuración de los motores y los ejes de coordenadas del vehículo, lo cual es dependiente del vehículo en sí mismo. En este caso solo se utilizan desplazamientos y giros únicamente respecto al eje vertical, siendo suficiente para llegar a cualquier posición y poder manipular distintos objetos.

Además este control de velocidad permanece activo independientemente de si el usuario controla el robot o no. De esta forma la persona que teleopera el vehículo no necesita conocer la configuración exacta de motores, ya que ésta podría cambiar según las necesidades. Además, el usuario dirige el vehículo usando velocidades cartesianas, lo cual facilita en gran medida el control del mismo.

2.3. HROV CONTROL

Uno de los puntos clave de la arquitectura propuesta, es el control de la autonomía del robot y el estado de la misión. Entendemos como *control de la autonomía del robot*, el control del funcionamiento en modo autónomo o teleoperado; mientras que el *control del estado de la misión* es el control de cada una de las fases en las que se encuentra la misión. Para pasar de modo automático a teleoperado, el usuario debe pulsar el botón *User Control Request* del joystick. Una vez pulsado, este controlador se encargará de mandar la señal correspondiente para que se deshabilite el control autónomo de los motores, hasta que el usuario vuelva a presionar dicho botón.

Cabe indicar que cada misión está dividida en fases, como por ejemplo en una misión de tipo *Search & Recovery*: reconocimiento, reconstrucción de la escena, detección de objetos de interés, identificación del objetivo, recuperación del objetivo, subir a superficie y volver al punto de partida. Gracias a esta división en fases, el controlador supervisa la misión y sólo permite la interacción del usuario al respecto de la fase en marcha. Por ejemplo, durante la fase de reconocimiento, cualquier intervención del usuario se limita al control de la navegación del vehículo, imposibilitando el control y operación del brazo. A medida que una fase finaliza (satisfactoriamente o no), el algoritmo correspondiente envía un comando de control a este controlador. Una vez recibido, se procederá a la ejecución de la siguiente fase.

Así pues, a fin de conseguir ambos objetivos, el módulo HROV CONTROL se encarga de monitorizar el estado del robot y los procesos del resto de módulos para conocer el estado de la misión:

- Del módulo UIAL obtiene la información relativa a la interacción del usuario con los distintos dispositivos de entrada para el control del vehículo y del brazo, o cuando el usuario requiere el control de la misión.
- Del robot, real o simulado, obtiene la información relativa a la odometría y al resto de sensores disponibles.
- Al módulo THRUSTER CONTROL le envía la posición a la que el robot debe llegar.
- Al módulo HROV ARM CONTROL le envía el comando para activar el control del brazo.

Dado que este módulo podría considerarse como el centro neurálgico de información, una de las funciones más importantes es la de velar por la seguridad del robot (Robot Safety Measures, RSM). Tal y como se ha mencionado anteriormente, los pilotos de ROV sufren de la denominada *fatiga cognitiva* debido, principalmente, al estrés de prestar atención a múltiples entradas de información. A fin de mitigar este problema, las RSM se encargan de evitar que el robot pueda dañarse como consecuencia de un fallo de control humano o bien porque se puede comprometer la seguridad del robot (e.g. llegando a la profundidad máxima). Así, evitará que el robot pueda descender más allá de un límite preestablecido, aún cuando el usuario lo intente, evitará que el brazo se active mientras el robot esté navegando, o pueda navegar hacia un obstáculo previamente detectado. Otro ejemplo de su utilidad es en el caso de pérdida de comunicación con el usuario, ya que el robot emergerá a superficie para que pueda ser rescatado.

Resumiendo, las funciones más relevantes de este módulo son:

- Recibir la información de los sensores del robot real o simulado.
- Controlar las diferentes fases de cada tipo de misión.
- Controlar cuándo debe activarse el controlador del brazo o del vehículo.
- Establecer cuando el robot debe funcionar en modo automático (AUV) o teleoperado (ROV).
- Implementar funciones específicas para garantizar la seguridad del robot.

2.4. HROV ARM CONTROL

Tal y como se ha explicado anteriormente, el usuario puede controlar tanto el brazo como el vehículo. En primer lugar, el usuario debe pulsar el botón *User Control Request* del joystick, y posteriormente el botón *Arm Control Request*.

Para ello, el usuario mueve el joystick indicando cuál es la posición deseada del efector final, en este caso la pinza robótica. Estos movimientos son recogidos por la capa de abstracción (UIAL), que se encargará de filtrarlos. Con el resultado de dichos movimientos, se genera un vector con las posiciones x, y, z, roll, pitch, yaw. Usando la cinemática inversa de librería de KDL [12] para el modelo del brazo, obtenemos los valores finales de cada articulación para obtener dicha posición del efector final.

Si el módulo HROV CONTROL envía una señal de alarma de profundidad máxima/distancia al suelo mínima a través de RSM, este controlador bloqueará el uso del brazo en modo manual. Este bloqueo evitará que el usuario daña de forma accidental el brazo.

3. INTERFAZ INMERSIVA

Una vez presentada la arquitectura, el siguiente aspecto clave dentro de la arquitectura propuesta es la interfaz de usuario. Siguiendo con la propuesta presentada en [13], la evolución que se presenta aquí hace hincapié en la experiencia de usuario, sensación de inmersión, comunicación visual gráfica (usando iconos) y la reducción de la cantidad de información a mostrar a lo estrictamente necesario (e.g. durante la navegación, no se muestran datos relativos al brazo).

La sensación de inmersión se consigue gracias a las gafas de realidad virtual Oculus Rift. Los beneficios de usar un *Head-Mounted Display* (HMD) como este son: abstracción del entorno para evitar distracciones, sensación de realismo, inmunidad a la luz solar (en comparación a usar una pantalla al aire libre)... Del mismo modo, algunos de estos beneficios suponen, en cierta medida, una desventaja: la pérdida de referencias espaciales del entorno. Dicho de otro modo, una vez se está usando el HMD es complicado poder usar dispositivos de entrada como ratón-teclado o Leap Motion, ya que se pierde la referencia de su ubicación.

La interfaz de usuario que presentamos funciona como una nueva capa visual sobre el simulador UWSim y se compone de dos partes: el menú de usuario y un Virtual Cockpit (VC).

Dado que el módulo UIAL es capaz de controlar diversos dispositivos de entrada, podemos usar uno de dichos dispositivos (en nuestro caso un SpaceNavigator) para mostrar u ocultar el menú y para la selección de la opción deseada del menú. En la Fig. 2 podemos observar un ejemplo del

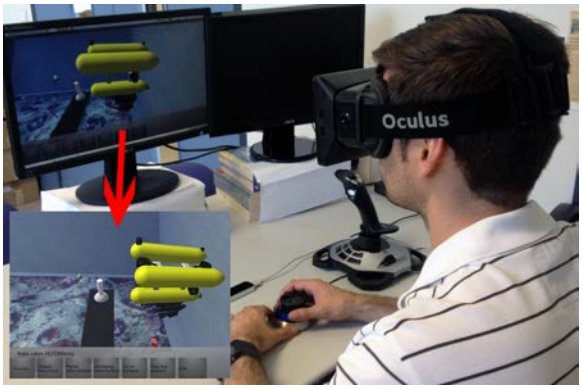

Figura 2: Ejemplo del menú de usuario dentro de una escena del UWSim.

menú principal que se muestra tras la activación del menú por parte del usuario. Las ventajas de usar este enfoque son múltiples, como por ejemplo que el menú es compatible con UWSim, teniendo acceso a todas las funciones del mismo; o que el usuario no necesita quitarse el HMD, pues el uso de un menú contextual es difícil de visualizar en 3D. Las opciones de este menú hacen referencia a instrucciones de alto nivel (e.g. Survey specification, Panel intervention, Object recovery). Una vez el usuario ha seleccionado la opción deseada, aparecerá un nuevo menú con las opciones disponibles para dicha selección.

Respecto al VC, la idea principal es reducir la cantidad de información visual a la que el usuario debe prestar atención. Así, nuestro enfoque inmersivo usa una visión egocéntrica natural, en contra de las interfaces de teleoperación tradicionales que utilizan el control manual de la cámara y la visualización de información relacionada con la misión a través de un conjunto de monitores. Para ello, el VC está compuesto por:

- Una mesa y joystick virtual, consiguiendo el efecto de visión egocéntrica natural. Además, el movimiento del joystick virtual, que es independiente del dispositivo de control, añade realimentación visual respecto al movimiento del vehículo.

- Una señal para el control del usuario, que se activa cuando el HROV CONTROL solicita la intervención del usuario o bien cuando el propio usuario solicita el control del robot.

- Una señal de alarma, que se activa en el momento que hay una alarma procedente de RSM solicitando la interacción del usuario.

3.1. Interactive markers

Además para esta interfaz se han adaptado los *interactive markers* de ROS para trabajar en el motor gráfico de UWSim: OpenSceneGraph (OSG) [14]. Los *interactive markers* permiten al usuario crear e interactuar con marcadores como cajas, flechas, puntos, objetos... etc. de una manera sencilla utilizando las interfaces de ROS. De esta forma el programa cliente de los marcadores, en este caso el simulador, se abstrae del servidor, la interfaz, que es la que toma el control de ellos.

Esto ofrece una gran versatilidad para el desarrollo de nuevas aplicaciones sobre el simulador ya que únicamente mandando información de alto nivel como puede ser: crea una esfera aquí o mueve el objeto 3 metros, puede controlar la información que finalmente se muestra. En el caso de la interfaz inmersiva es aún más interesante ya que permite mostrar información sobre el mismo simulador integrando todo en una única vista.

Para ello ha sido necesario crear un paquete de ROS, disponible para la comunidad, que traduce los mensajes al motor gráfico creando y modificando las diferentes geometrías disponibles o incluso cargando modelos 3D. Además ha sido necesario adaptar el simulador UWSim con nuevos servicios ROS que utilizan el paquete anteriormente mencionado para crear los modelos en la escena del simulador. Como ejemplo se muestra la figura 3 y el video disponible en goo.gl/ok6r7s donde un vehículo sigue la posición de un marcador guíado por el ratón.

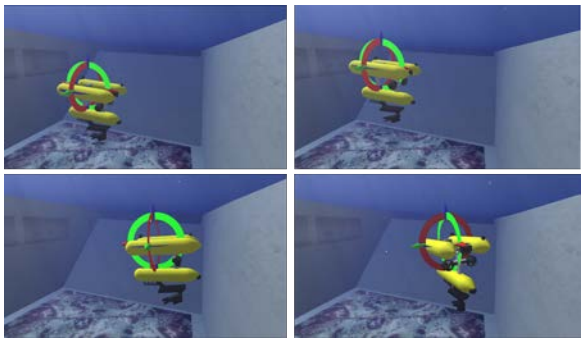

Figura 3: Capturas del control de la posición de un vehículo utilizando un *interactive marker* controlado por el ratón.

En el caso de la interfaz inmersiva existen gran cantidad de posibilidades de aplicación. En primer lugar puede añadirse un segundo brazo robótico para planear el movimiento del brazo real, y moverlo mediante cualquier dispositivo disponible (e.g. LeapMotion) hasta la posición deseada antes de ejecutar la acción. También puede hacerse algo similar para los movimientos del robot, permitien-

do ver el destino final y la trayectoria a seguir.

4. CONCLUSIONES

En este trabajo se ha presentado una propuesta para el control de un HROV, compuesta por una arquitectura y una interfaz de usuario. Gracias al uso de la arquitectura propuesta, el usuario podrá especificar la misión a realizar por el robot y quedará como supervisor del mismo hasta que decida tomar el control (e.g. en caso de detectar un fallo) o hasta que finalice la misión. De este modo, se consiguen mantener las ventajas de tener al usuario dentro del bucle de control, mientras que a la vez se reduce la fatiga cognitiva típica de un piloto de ROV. Por otra parte, la interfaz de usuario se centra en la reducción de la cantidad de información dada al usuario, haciéndolo de manera más visual e intuitiva. Gracias al uso de un HMD, el usuario obtiene una sensación de inmersión dentro de la escena, consiguiendo reducir la fatiga cognitiva.

Actualmente, este trabajo tiene su continuación en la integración de nuevos dispositivos que mejoren la interacción hombre-robot. En concreto, se está trabajando en un algoritmo que permita al usuario interactuar con un objeto representado a través de una nube de puntos. Dicho algoritmo permitirá modificar el punto de vista del mismo e indicar cuál y cómo debe ser el mejor punto de agarre

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y Competitividad, código de proyecto DPI2011-27977-C03 (TRITON project), DPI2014-57746-C3 (proyecto MERBOTS)and the University Jaume I predoctoral grant PRE-DOC/2012/47.

Referencias

[1] L. Almeida, B. Patrão, P. Menezes, J. Dias, "Be the Robot: Human Embodiment in Tele-Operation Driving Tasks", Ro-Man 2014: The 23rd IEEE International Symposium on Robot and Human Interactive Communication, Edinburgh, UK, 2014

[2] J. Yuh, "Design and Control of Autonomous Underwater Robots: A Survey", Autonomous Robots 8(1), pp.7-24, 2000.

[3] T. B. Sheridan, Telerobotics, .Automation and Human Supervisory Control", Massachusetts: MIT Press. 1992.

[4] W. Li, D. Sadigh, S. S. Sastry and S. A. Seshia, "Synthesis for Human-in-the-Loop Control Systems", Electrical Engineering and Computer Sciences University of California at Berkeley, 2013.

[5] Health and S. E. (HSE), Eds., Reducing Error and Influencing Behaviour (Guidance Booklets). HSE Books, 1999.

[6] M. Prats, J. Pérez, J. Fernández, and P. Sanz, "An open source tool for simulation and supervision of underwater intervention missions", in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, Vilamoura, Algarve, Portugal, pp. 2577–2582. 2012.

[7] P. J. Sanz, P. Ridao, G. Oliver, G. Casalino, Y. Petillot, C. Silvestre, C. Melchiorri, and A. Turetta, "TRIDENT: An european project targeted to increase the autonomy levels for underwater intervention missions," in OCEANS'13 MTS/IEEE conference. Proceedings, San Diego, CA. 2013.

[8] FP7-PANDORA "Persistent Autonomy through learNing, aDaptation, Observation and Re-plAnning (PANDORA)" Available: http://persistentautonomy.com

[9] FP7-MORPH "Marine Robotic System of Self-Organizing, Logically Linked Physical Nodes (MORPH)" Available: http://morph-project.eu

[10] ROS: Robot Operative System Available: www.ros.org

[11] J. C. García, B. Patrão, J. Pérez, J. Seabra, P. Menezes, J. Dias and P. J. Sanz, "Towards an immersive and natural gesture controlled interface for intervention underwater robots", in MTS/IEEE OCEANS'15, Genova (Italy), 2015.

[12] http://www.orocos.org/kdl

[13] J. C. García, B. Patrão, L. Almeida, J. Pérez P. Menezes, J. Dias and P. J. Sanz, "Design and Evaluation of a Natural Interface for Remote Operation of Underwater Robots", IEEE Computer Graphics and Applications, no. 99, 2015.

[14] OSG: OpenSceneGraph library Available: http://www.openscenegraph.org.