Configuración y ejecución de benchmarks de intervención robótica submarina en UWSim mediante herramientas Web

Javier Pérez, Jorge Sales, Raúl Marín, Enric Cervera and Pedro J. Sanz
Departamento de Ingeniería y Ciencia de Computadores
Av. Sos Baynat, s/n, 12071 Castellón de la Plana (España)
Universitat Jaume I
{japerez, salesj, rmarin, ecervera, sanzp}@uji.es

Resumen

La realización de benchmarks es hoy en día un problema a abordar en plataformas robóticas de investigación, debido al hecho de que no es fácil reproducir experimentos previos y conocer en detalle en qué condiciones reales se han ejecutado otros algoritmos. Contar con una herramienta basada en web para configurar y ejecutar benchmarks abre la puerta a nuevas oportunidades como el diseño de tele-laboratorios virtuales que permitan el desarrollo de nuevos algoritmos utilizando restricciones específicas y detalladas. En el contexto de las intervenciones submarinas con robots semi-autónomos, la situación resulta todavía más interesante. Para este escenario en particular, se propone el uso de una herramienta de simulación y benchmarking de código abierto, con la posibilidad de ser configurada y utilizada desde una interfaz web, permitiendo la descarga de los resultados para su posterior análisis.

Palabras clave: Benchmarking, Intervención Submarina, Datasets, Simulador Open Source, Simulador on-line.

1. Introducción

La manipulación submarina mediante I-AUVs (Autonomous Underwater Vehicles for Intervention) permite el diseño de nuevas aplicaciones como la estudiada en el proyecto FP7 TRIDENT [1], donde se recuperó una caja negra del fondo del mar de forma autónoma. Para lograr esto, el uso de UWSim (Underwater Simulator [2]) ha sido fundamental para poder realizar pruebas, integración y también benchmarking (véase Fig. 1).

Existen otros simuladores previos para aplicaciones submarinas, que o bien han quedado obsoletos o están siendo utilizados para fines muy específicos [3] [4]. Por otra parte, la mayoría de los simuladores disponibles no han sido diseñados como código abierto, lo que hace difícil mejorar y potenciar las capacidades del simulador. Además, hay otros simuladores comerciales como ROVSim [5], DeepWorks [6] o ROVolution [7]. Sin embargo, éstos han sido diseñados para entrenar a pilotos de ROV, que no es el objetivo de nuestra línea de investigación.

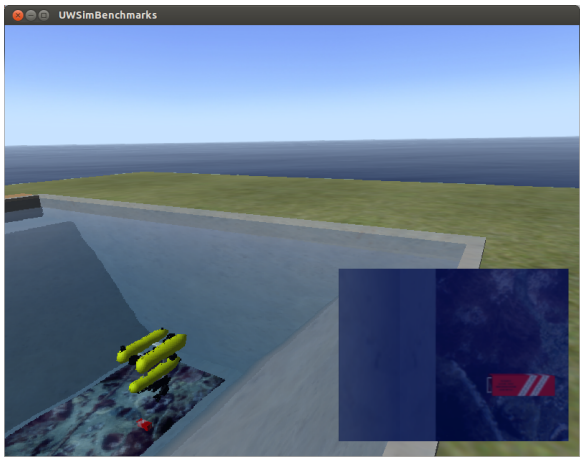

Figura 1: *Benchmark* de *tracker* visual ante cambios en las condiciones de visibilidad de la escena.

La definición de *benchmarks* específicos para intervenciones submarinas en simuladores, permite tener una plataforma perfecta para comparar en igualdad de condiciones el rendimiento de dos algoritmos diferentes. Se han propuesto varias definiciones de *benchmarks*, pero en este artículo vamos a utilizar la propuesta en [8], en el sentido de que "incluye evaluación numérica de los resultados (métricas de rendimiento) como un elemento clave", y también que "los aspectos principales son la repetibilidad, la independencia, y la no ambigüedad".

Algunos trabajos previos sobre *benchmarking* se han realizado utilizando otros simuladores, como Stage [9], USARSim [10] o Webots [11]. Sin embargo, este trabajo describe tanto el desarrollo de *benchmarks* específicos, como una plataforma *online* para la configuración y ejecución de *benchmarks* de forma remota. Por otra parte, y hasta donde los autores tenemos conocimiento, en el contexto de la robótica submarina no se ha propuesto ningun trabajo de este tipo hasta la fecha.

En este artículo se presenta una plataforma para el diseño de *benchmarks* configurables haciendo

ISBN-13: 978-84-697-0589-6

uso del simulador UWSim y se aporta un ejemplo de comparación de *trackers* visuales. En particular, se describe la infraestructura desarrollada que permite el uso de esta herramienta tanto de manera local [12], como *on-line*, utilizando una herramienta de configuración web [13].

2. Revisión de *suites* de *benchmarking* y *toolkits* relacionados

En los últimos años, se han desarrollado varias suites de *benchmarking* en el campo de la robótica. Muchas de ellas se centran exclusivamente en un subcampo específico de investigación robótica, pero, hasta donde los autores tenemos conocimiento, ninguna de ellas se centra en el campo de los vehículos submarinos autónomos.

En el campo de la manipulación robótica, se han presentado varias suites tales como el *OpenGrasp Benchmarking Suite* [14]. Esta suite es un entorno de software para la evaluación comparativa de agarre y manipulación diestra usando el toolkit OpenGrasp. También proporciona un servicio web que administra los escenarios, *benchmarks* disponibles, modelos y resultados de benchmarks. La desventaja es que el software a evaluar ha de ser adaptado.

Otra suite interesante de *benchmarking* en el campo de la manipulación es VisGrab [15]. VisGrab es un *benchmark* para la manipulación basada en visión. Esta suite ofrece un software capaz de evaluar métodos de generación de agarres basados en visión. En este caso, el sitio web sólo proporciona la publicación de los resultados.

Planificadores de movimiento, seguimiento y planificación de trayectorias han sido también campos de investigación muy activos en lo que respecta a métricas y suites de *benchmarking*. En [16], los autores describen una infraestructura genérica para hacer *benchmarking* de planificadores de movimiento. El punto clave de la contribución es la facilidad para comparar diferentes diseños debido al uso integrado de ROS (Robot Operating System) [17] y MoveIt!.

3. UWSim: una herramienta de simulación 3D para *benchmarking*

UWSim[1] [2] es una herramienta de software de código abierto para la visualización y la simulación de misiones robóticas submarinas y que permite conectarse con un módulo de *benchmarking* (ver Figs. 2 y 1). El software es capaz de visualizar escenarios virtuales bajo el agua que se pueden configurar utilizando software de modelado estándar y se puede conectar a programas de control externo mediante el uso de interfaces de ROS.

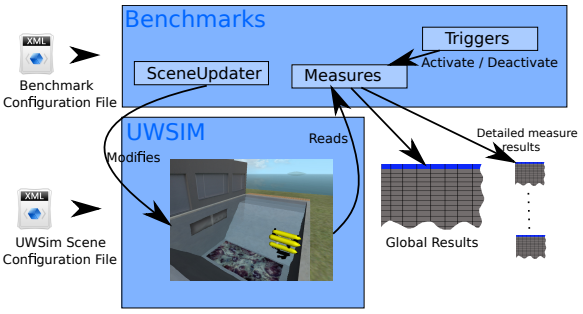

Figura 2: Diagrama de flujo del módulo de *benchmarking*: la configuración del *benchmark* se carga en el módulo y la escena en el simulador UWSim. A continuación, el sistema produce resultados que se guardan para un posterior análisis.

UWSim se está utilizando actualmente en diferentes proyectos financiados por la Comisión Europea (MORPH [18] y PANDORA [19]) con el fin de realizar *HIL (Hardware In the Loop)* y experimentos para reproducir misiones reales a partir de los registros o *logs* capturados durante la ejecución de las mismas.

Recientemente, se ha desarrollado un módulo de *benchmarking* para UWSim [20]. Al igual que UWSim, este módulo utiliza ROS para interactuar con otro software externo. La interfaz de ROS permite que se pueda evaluar un programa externo y que pueda comunicarse tanto con el simulador (que puede enviar comandos para llevar a cabo una tarea) como con el módulo de *benchmarking* (que puede enviar los resultados o los datos necesarios para ser evaluado). La información detallada sobre cómo configurar y ejecutar un *benchmark* en UWSim se puede encontrar en [12].

4. La ejecución de benchmarks online

4.1. El servidor de configuración y visualización web

El sistema de ejecución de benchmarks es el que se muestra en la Fig. 3. El servidor online *(robotprogramming.uji.es)* atiende las peticiones web y sirve las páginas. Además, hace de intermediario con el servidor de simulación *(arkadia.act.uji.es)*, el cual es inaccesible desde el exterior (ya que pertenece a la red de servidores internos del dominio *uji.es* y está protegido por un *firewall*), redirigien-

[1]Disponible online: http://www.irs.uji.es/uwsim

ISBN-13: 978-84-697-0589-6

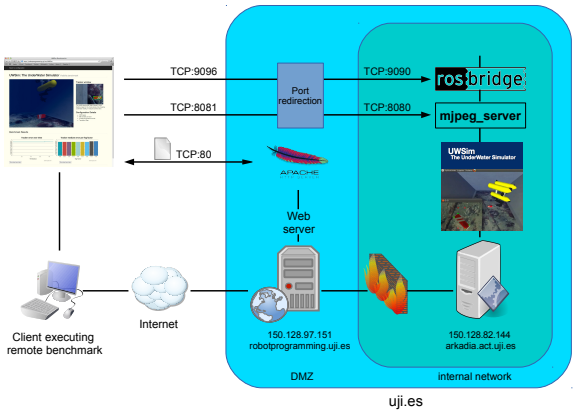

Figura 3: Diagrama de interacción del servicio web de configuración y ejecución de benchmarks.

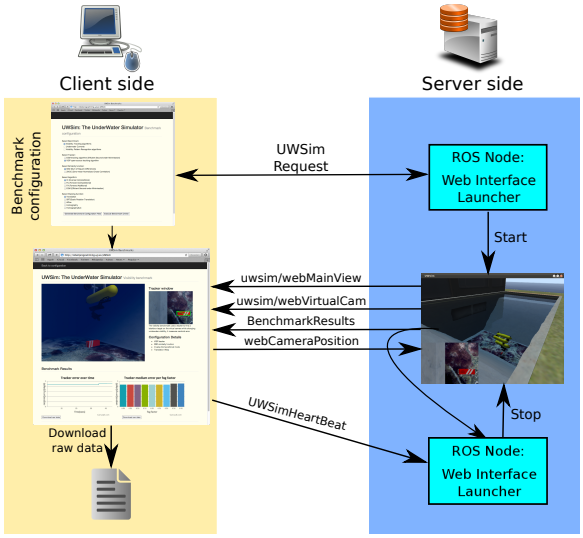

Figura 4: Diagrama de red del servicio web de configuración y ejecución de benchmarks.

do las llamadas de `rosbridge` y `mjpeg_server` a él. De forma que la interacción con el servidor de simulación es transparente para el cliente, el cual únicamente hace llamadas a través de la interfaz de JavaScript para `rosbridge` y recibe streaming de video mediante `mjpeg_server`.

4.2. El servicio de ejecución de benchmarks

El funcionamiento del servicio de ejecución web es el que se puede ver en la Fig. 4. En primer lugar, el usuario selecciona las opciones de configuración deseadas a través de un formulario que permite escoger entre diferentes benchmarks, y algoritmos a evaluar. Aunque de momento solo está activo el benchmark de visibilidad, ya está preparado para ejecutar benchmarks de corrientes y reconocimientos de patrones y en un futuro se podrá incluso proporcionar el código fuente a evaluar y publicar y compararse con aquellos que lo deseen.

Una vez el usuario ha configurado las opciones deseadas, se pasa a ejecutar el benchmark online. Para ello se llama al servicio `/uwsimRequest` utilizando `rosbridge`, el cual es un mensaje que contiene un diccionario de claves y valores en forma de dos vectores. En ellos se transmiten las opciones de configuración seleccionadas por el usuario al nodo de ROS `webInterfaceLauncher` situado en el servidor de simulación que se encarga de ejecutar la petición.

Para mantener igualdad de condiciones a la hora de comparar los resultados de distintos *benchmarks*, es necesario asegurar que los benchmarks se ejecuten de manera individual, ya que es posible que los resultados se vean afectados por los recursos disponibles en el momento de la simulación. Es por esto que el nodo de ROS que recibe la llamada al servicio `uwsimRequest` solo permite la ejecución de una instancia del simulador de manera simultánea. Esto se consigue utilizando un *Id* de cliente generado aleatoriamente en el servidor en el momento de la petición, y que servirá para identificar al cliente a lo largo de la simulación. En el caso de accesos simultaneos se muestra un mensaje para indicar que el servidor está ocupado, en el caso de que el servicio web se extienda se podría crear una cola de clientes con un tiempo de espera estimado. Además de este identificador, que permite conocer cuando llega un nuevo cliente, se necesita conocer cuando un cliente ha abandonado el servicio y cuando ha terminado la ejecución de un cliente.

Para ello se emplean dos métodos: un *heartbeat* asociado al cliente y monitorización de la ejecución. Para el *heartbeat* se utiliza un topic `webUWSimHeartBeat` que envía el cliente JavaScript cada segundo con su *Id* de cliente. De esta forma, el servidor de simulación es capaz de detectar una desconexión o una caída del cliente y abortar la simulación para permitir el acceso a otro cliente. En el caso de que la comunicación sea muy lenta y no sea capaz de enviar al menos 1 *heartbeat* cada 10 segundos el proceso de simulación se abortará igualmente. Para estos casos se están estudiando alternativas que permitan la simulación sin presencia del cliente y el envio de resultados mediante otro medio, como puede ser correo electrónico.

La monitorización de la ejecución se hace a través del envío de resultados, en el momento que se detecta que no se están enviando resultados al cliente durante un tiempo establecido se decide que el experimento ha finalizado, independientemente del

ISBN-13: 978-84-697-0589-6

estado en el que haya terminado.

Una vez el *webInterfaceLauncher* ha establecido que el experimento ha terminado, manda una señal de terminación *(SIGINT)* al proceso de simulación y deja libre el *slot* para futuros clientes. Además de esto queda como trabajo futuro añadir la funcionalidad de acelerar las simulaciones para aumentar la utilidad del servicio dando la posibilidad de realizar benchmarks en lotes.

Durante la ejecución de la simulación, se puede observar una pantalla principal con la vista principal del simulador en 3D recibida a través de un topic de imagen mediante `mjpeg_server` (ver Fig. 6). En esta vista, el cliente puede moverse por la escena utilizando el ratón a modo de `trackball`, el botón derecho para desplazarse y la rueda del ratón para hacer zoom. Estas órdenes se codifican en JavaScript para enviar en el topic `webCameraPosition` la nueva posición de la cámara al servidor, el cuál actualizará la imagen.

Para la visualización de resultados por parte del cliente se utiliza el topic `BenchmarkResults` publicado por el módulo de benchmarks [20], el cual envía los resultados conforme se van generando. Este topic es capturado a través de `rosbridge` por el JavaScript del cliente y los resultados se representan de forma gráfica y en tiempo real mediante la librería `canvasJS`. Además, existe la posibilidad de descargar los datos en formato de texto para posterior análisis. Para ello, existe un botón en la interfaz de usuario con la leyenda *"Download raw data"*. Aunque la simulación haya terminado el cliente aún será capaz de consultar y descargar los resultados obtenidos ya que están almacenados temporalmente en su navegador mediante JavaScript.

5. La interfaz web de benchmarking

La interfaz web de benchmarking consta de dos pantallas, la primera (ver Fig. 5), permite al cliente configurar las opciones de ejecución. Una vez configurado se pueden generar los ficheros de configuración para ejecutar el benchmark de forma local o ejecutar el benchmark online. Esta última opción lleva a la página de ejecución del benchmark donde se puede visualizar como se ejecuta el benchmark y consultar o descargar los resultados.

5.1. Pantalla de configuración

Además de ejecutar los benchmarks de forma totalmente local modificando los ficheros de configuración, ver [12], también se puede utilizar la herramienta de configuración web [13] (ver Fig. 5).

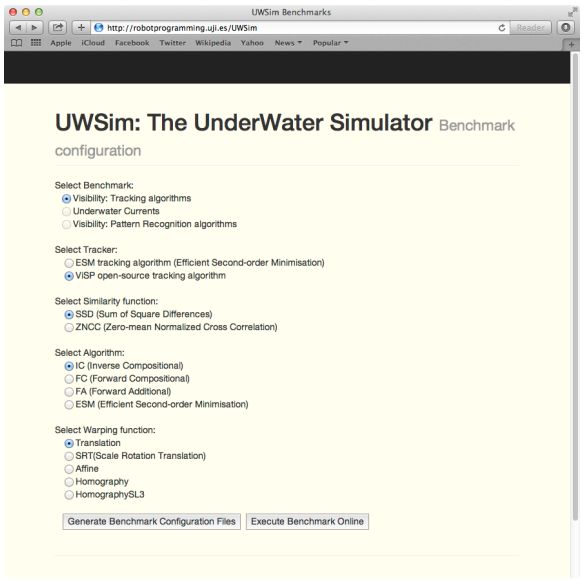

Figura 5: Herramienta de configuración de *benchmarks* para UWSim basada en web: ejemplo de configuración de un *benchmark* de visibilidad.

En el ejemplo de la figura se está configurando un benchmark de visibilidad. Las opciones a configurar incluyen, entre otras, la elección de dos tipos diferentes de tracking y sus opciones de funcionamiento. Una vez el usuario ha seleccionado los parametros y el benchmark esta configurado, se pueden generar los ficheros de configuración para ejecutar la simulación localmente o utilizar la herramienta de ejecución online en el servidor. Además, en el futuro esta herramienta será capaz de ejecutar código subido por los propios clientes para evaluarlo utilizando diferentes lenguajes como C, C++, Python, MATLAB, Simulink... etc, aunque esta opción está todavía en desarrollo.

5.2. Pantalla de ejecución del benchmark

Mientras se está ejecutando un benchmark, el usuario puede monitorizar su desempeño mediante la pantalla principal, el *feedback* secundario y los resultados. En la pantalla principal, el usuario puede moverse por el entorno mientras el benchmark se ejecuta de la misma forma que lo haría con la ventana del simulador (ver Fig. 6).

El feedback secundario es la zona situada a la derecha de la pantalla principal en la cual se muestra información relevante al benchmark en particular. Por ejemplo, en el caso de un benchmark de visibilidad, se muestra la cámara virtual sobre la que actúa el tracker con un recuadro sobre la detección del mismo. Esta zona, sin embargo, es una zona reservada para las características del benchmark y por tanto en otro tipo de benchmark puede mostrarse algo diferente.

ISBN-13: 978-84-697-0589-6

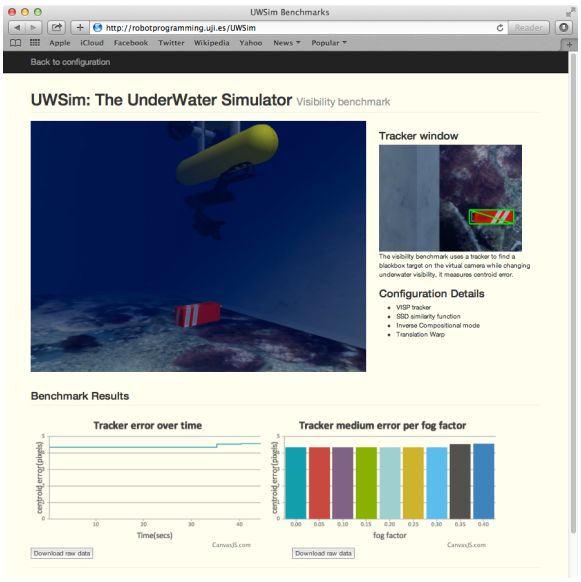

Figura 6: Herramienta de configuración de *benchmarks* para UWSim basada en web: el *benchmark* se está ejecutando en tiempo real. Es usuario puede también descargar los datos generados.

Por último, la zona de resultados muestra gráficas en tiempo real creadas mediante la librería `canvasJS`, una herramienta que permite mostrar datos de manera gráfica haciendo uso de HTML5, asegurando compatibilidad en distintos dispositivos como tablets, teléfonos...etc. En el caso de benchmarking de visibilidad, se muestra el error del tracker a lo largo del tiempo y el error medio del tracker según la visibilidad. Dependiendo del benchmark elegido pueden mostrarse diferentes resultados e incluso en un futuro podrá ser configurable por el usuario. Además, en cualquier momento pueden descargarse los datos de los gráficos en formato de texto para un posterior análisis utilizando el botón de la interfaz de usuario con la leyenda *"Download raw data"*.

6. Evaluación de trackers visuales en condiciones de visibilidad variable.

Utilizando las herramientas presentadas anteriormente es posible realizar *benchmarking*, llevando los algoritmos a su limite para conocer en qué condiciones pueden funcionar y qué resultados pueden obtener. De esta forma se pueden usar los mejores recursos disponibles en cada momento dependiendo de la situación.

El *benchmark* puede ser llevado a cabo modificando los ficheros de configuración y ejecutando localmente, utilizando la interfaz web para crear los ficheros de configuración [12] para ejecutarlo en la

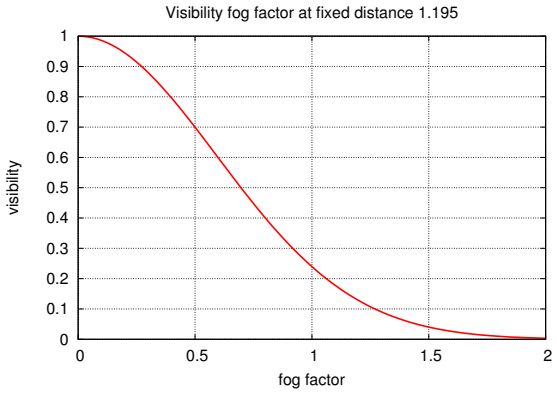

Figura 7: Relación de la visibilidad con el *fog factor*.

máquina local o utilizando la ejecución online.

A continuación, se presenta un ejemplo de *benchmarking* realizado con UWSim para demostrar las capacidades del sistema presentado. El objetivo del experimento es evaluar cómo afectan las variaciones de turbidez del agua a un algoritmo de tracking visual. El escenario virtual es el que se presenta en [20], en el que el vehículo *Girona500* con un brazo *ARM5E* está situado en la piscina del CIRS (Centro de Investigación en Robótica Submarina) sobre una caja negra (Fig. 1). En él se comparan dos algoritmos de tracking ESM de Inria [21] y el template tracker de ViSP (Visual Servoing Platform, INRIA Lagadic) con diferentes configuraciones. Ambos algoritmos se inicializan automáticamente para seguir la posición de la caja negra en la cámara virtual evaluando su precisión en las esquinas y centroide.

Para variar la turbidez del agua en UWSim se utiliza la niebla submarina (*fog factor*) que es una medida independiente de la distancia, del estado del agua. Según la definición de visibilidad de openGL la visibilidad es un valor adimensional de 0 a 1 que nos indica con que nitidez podemos ver el objeto. Esta niebla submarina se relaciona con la visibilidad de manera exponencial en función de la distancia siguiendo la siguiente formula:

$$visibility = e^{-(fogfactor*distance)^2} \qquad (1)$$

De esta forma se deduce que la visibilidad en la escena presentada, donde el *target* se encuentra a 1.195 metros, es la que se puede observar en la figura 7.

De igual forma, si fijamos una visibilidad de 0.01, se puede calcular la visibilidad en metros en función del *fog factor*. Esto lo podemos observar en la figura 8.

ISBN-13: 978-84-697-0589-6

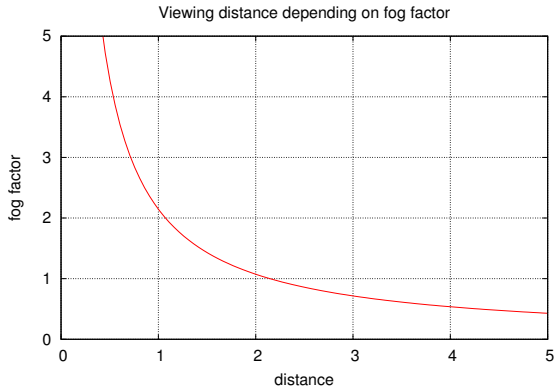

Figura 8: Distancia a la que los objetos son visibles en función del *fog factor*.

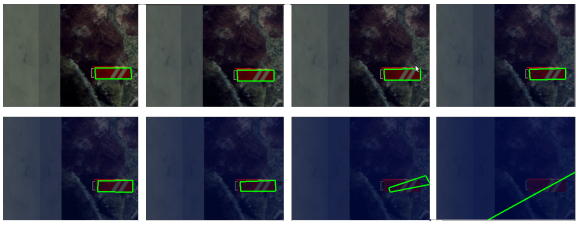

Figura 9: Capturas de pantalla del *tracker* visual ESM a lo largo del *benchmark*.

6.1. Tracking visual ESM

El primer algoritmo de *tracking* visual que se ha probado es el *ESM visual tracker* de Inria. En la Fig. 9 podemos observar como el tracker es capaz de encontrar con precisión la caja negra mientras la niebla incrementa en el *benchmark*. Finalmente cuando el nivel de niebla es elevado, y por tanto la visibilidad es mala, el tracker pierde el objetivo.

El resultado obtenido en este experimento es el que muestra la Fig. 10. Como se puede observar, el error es muy pequeño a lo largo de toda la simulación, por debajo de los 5 pixeles entre esquinas y centroide. Cuando la niebla alcanza un nivel de 1.25, el error incrementa drásticamente y finalmente el *tracker* no es capaz de seguir el objetivo. Según estos resultados el algoritmo de visión es fiable para niveles de niebla menores de 1.4.

6.2. ViSP visual tracker: métodos y *warps*

El tracker visual de ViSP (Visual Servoing Platform, INRIA Lagadic) utiliza algoritmos de registro de imágenes en lugar de las habituales aproximaciones basadas en características. Las funciones de similitud disponibles en ViSP para registro de imágenes son:

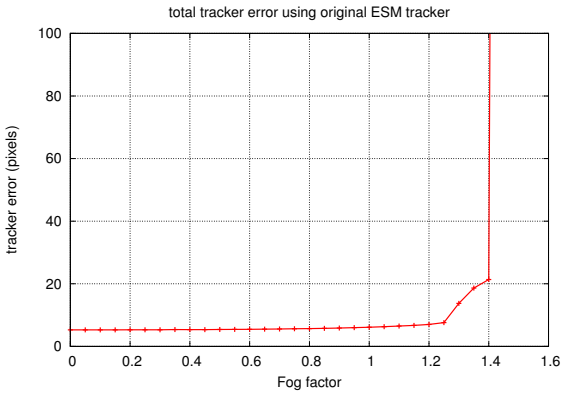

Figura 10: *Benchmark* de variación de la visibilidad: resultados del tracker ESM.

- SSD (Sum of Square Differences): Este método se puede utilizar de diferentes maneras: Inverse Compositional, Forward Compositional, Forward Additional, or ESM.
- ZNCC (Zero-mean Normalized Cross Correlation): Este método puede utilizarse de diferentes maneras: Inverse Compositional, Forward Additional.

Además de la función de similitud, ViSP permite el uso de diferentes *warps*. Los *warps* describen la transformación de la imagen buscada entre una imagen y la siguiente. El mejor *warp* depende de las restricciones de movimiento de la imagen que sigue, y por tanto, del problema en concreto. La librería ViSP ofrece 5 *warps* distintos:

- Translation: Esta es la transformación más sencilla de las que ofrece ViSP. Solo considera traslación en dos ejes (x e y).
- SRT (Scale Rotation Translation): La transformación SRT considera un escalado, una rotación en el eje z y una traslación 3D análoga al *translationwarp*.
- Affine: Este *warp* trata de conservar puntos, líneas rectas y planos.
- Homography: Estima los ocho parámetros de la matriz de homografía H.
- HomographySL3: Es exactamente igual a *Homography warp* pero los parámetros se estiman en el marco de referencia SL3.

Se han probado todas las combinaciones de *warp* y función de similitud en el *benchmark* de variación de visibilidad. Como el objeto a seguir no se mueve respecto a la cámara, el *warp* más restrictivo, en este caso el de traslación, debería ser el que

ISBN-13: 978-84-697-0589-6

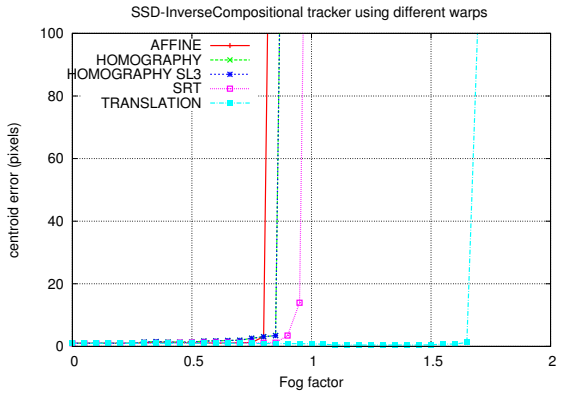

Figura 11: Benchmark de variación de la visibilidad: resultados para ViSP SSD-Inverse Compositional tracker.

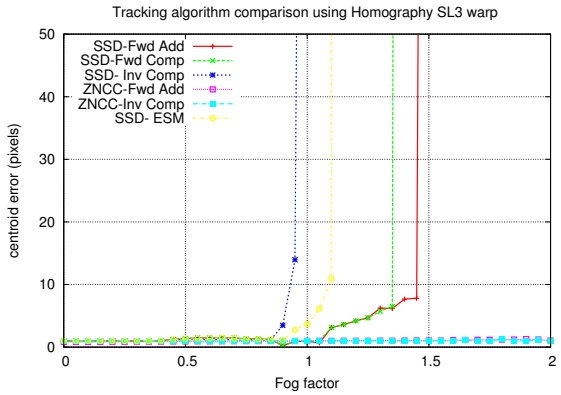

Figura 12: Benchmark de variación de la visibilidad: resultados para Homography SL3 warp.

ofrezca mejores resultados siendo capaz de seguir al objetivo con mayor cambio de visibilidad.

En la Fig. 11 podemos observar una comparación de los diferentes *warps* para una misma función de similitud. Como era de esperar, el *warp* menos restrictivo, traslación, consigue significativamente mejores resultados que los demás.

Como puede verse en la Fig. 12, en la cual se comparan las diferentes funciones de similitud para un mismo *warp*, las funciones que utilizan ZNCC ofrecen resultados mucho mejores, llegando al límite del experimento sin fallar. El resto de funciones de similitud consiguen diferentes resultados entre 0.9 y 1.45 de *fog factor*. Otro resultado interesante es que las funciones *inverse compositional* tienen un comportamiento asintótico, es decir, encuentran el objetivo con precisión hasta el momento de fallar drásticamente, lo cual es deseable para el caso de manipulación robótica.

Como conclusión, se puede decir que el tracker ViSP ZNCC-Inverse Compositional es el más robusto para el caso estudiado. Ofrece los mejores resultados para cambios de visibilidad en entorno submarino. El tracker ESM original obtiene buenos resultados comparados con los de ViSP, sin embargo, los trackers ZNCC y el *warp* de traslación consiguen mejores resultados.

Estos resultados obtenidos son un buen punto de partida para la elección de un software de *tracking* visual robusto para el sistema real. Además, es capaz de establecer los límites operacionales del sistema ayudando a elegir dinámicamente el mejor algoritmo en cada situación.

7. Conclusiones

En este trabajo se presenta una herramienta para ejecución online de *benchmarks* de intervención robótica mediante herramientas web para el simulador UWSim (UnderWater Simulator). Esta herramienta permite el diseño de experimentos de validación en un espacio web colaborativo de una forma sencilla y práctica para el usuario ya que no requiere instalar software.

Los benchmarks presentados se centran en el efecto de la visibilidad limitada para el algoritmo de *tracking*. Se han diseñado detallados *benchmarks* para tener en cuenta estas condiciones y extraer en qué condiciones son capaces de funcionar los algoritmos de visión submarina. Como trabajo futuro, la interfaz online está siendo mejorada de forma que será posible que el usuario envíe su propio código a comparar utilizando diferentes lenguajes como pueden ser C, C++, Python, Matlab, Simulink y además proporcionar "esqueletos" de funciones para que compararse sea aún más sencillo y rápido. Sin necesidad de descargar y ejecutar software de simulación.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y Competitividad, código de proyecto DPI2011-27977-C03 (proyecto TRITON), por la Universitat Jaume I y Fundación Caixa Castelló-Bancaixa, proyecto PI·1B2011-17, y por la beca predoctoral de la Universitat Jaume I con código PREDOC/2012/47.

Referencias

[1] 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.

ISBN-13: 978-84-697-0589-6

[2] M. Prats, J. Pérez, J. Fernández, and P. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 2577–2582.

[3] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, "A survey of commercial open source unmanned vehicle simulators," in *Robotics and Automation, 2007 IEEE International Conference on*, april 2007, pp. 852 – 857.

[4] O. Matsebe, C. Kumile, and N. Tlale, "A review of virtual simulators for autonomous underwater vehicles (AUVs)," *IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV'2008)*, 2008.

[5] Marine Simulation, "Marine Simulation ROVsim," Available online: http://marinesimulation.com.

[6] Fugro General Robotics Ltd., "Deepworks," Available online: http://www.fugrogrl.com/software/.

[7] GRL, "ROVolution, GRL (General Robotics Limited). ROV pilot training and mission planning simulator."

[8] R. Dillmann, "KA 1.10 Benchmarks for Robotics Research," University of Karlsruhe, Tech. Rep., 2004.

[9] D. Calisi, L. Iocchi, and D. Nardi, "A unified benchmark framework for autonomous Mobile robots and Vehicles Motion Algorithms (MoVeMA benchmarks)," in *Workshop on experimental methodology and benchmarking in robotics research (RSS 2008)*, 2008.

[10] B. Taylor, S. Balakirsky, E. Messina, and R. Quinn, "Analysis and benchmarking of a Whegs robot in USARSim," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, sept. 2008, pp. 3896–3901.

[11] O. Michel and F. Rohrer, "The Rat's Life benchmark: competing cognitive robots," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, ser. PerMIS'08. New York, NY, USA: ACM, 2008, pp. 43–49. [Online]. Available: http://doi.acm.org/10.1145/1774674.1774682

[12] UWSim-Benchmarks. (2014) The UWSim Benchmarks Workspace. Available online: http://sites.google.com/a/uji.es/uwsim-benchmarks.

[13] ——. (2014) UWSim: The UnderWater Simulator online. Available online: http://robotprogramming.uji.es/UWSim/config.html.

[14] S. Ulbrich, D. Kappler, T. Asfour, N. Vahrenkamp, A. Bierbaum, M. Przybylski, and R. Dillmann, "The opengrasp benchmarking suite: An environment for the comparative analysis of grasping and dexterous manipulation," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, Sept 2011, pp. 1761–1767.

[15] G. Kootstra, M. Popović, J. Jørgensen, D. Kragic, H. Petersen, and N. Krüger, "Visgrab: A benchmark for vision-based grasping," *Paladyn*, vol. 3, no. 2, pp. 54–62, 2012. [Online]. Available: http://dx.doi.org/10.2478/s13230-012-0020-5

[16] B. Cohen, I. Sucan, and S. Chitta, "A generic infrastructure for benchmarking motion planners," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 589–595.

[17] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

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

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

[20] J. Pérez, J. Sales, M. Prats, J. V. Martí, D. Fornas, R. Marín, and P. J. Sanz, "The underwater simulator UWSim: Benchmarking capabilities on autonomous grasping," in *11th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2013.

[21] E. Malis, "Improving vision-based control using efficient second-order minimization techniques," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 2, 26-may 1, 2004, pp. 1843 – 1848 Vol.2.

ISBN-13: 978-84-697-0589-6