CEA Comité Español de Automática
XLVI JORNADAS AUTOMÁTICA CARTAGENA 2025
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Teleoperación alámbrica de pez robot implementada en ROS 2

Puig Sariñena, Max[a,*], Pino Jarque, Andrea[1], Solís Jiménez, Alejandro[1], López-Barajas, Salvador[1], Echagüe Guardiola, Juan Jesús[1], Sanz Valero, Pedro José[1]

[a] *Centro de Investigación en Robótica y Tecnologías Subacuáticas (CIRTESU), Avenida Sos Baynat s/n, Campus Riu Sec, 12071 Castellón de la Plana, España.*

To cite this article: Puig Sariñena, Max, Pino Jarque, Andrea, Solís Jiménez, Alejandro, López-Barajas, Salvador, Echagüe Guardiola, Juan, Sanz Valero, Pedro José . 2025. Fish robot alambric teleoperation implemented on ROS 2. Jornadas de Automática, 46. https://doi.org/10.17979/ja-cea.2025.46.12247

Resumen

Este artículo describe el desarrollo de un sistema de teleoperación alámbrica para un robot subacuático basado en ROS 2 Humble. Se implementa un nodo de control que emula dinámica de primer orden, combinando aceleración proporcional y rozamiento, y publica comandos PWM para dirigir la velocidad y el ángulo de la cámara, así como para activar un electroimán. La interfaz de usuario integra nodos de teleoperación con joystick y herramientas de visualización en tiempo real. Se ha validado el sistema en ensayos de diez minutos en tanque, obteniendo estabilidad direccional, un error medio de seguimiento inferior al 2 % y latencias compatibles con 20 Hz. Estos resultados sientan las bases para futuras extensiones—comunicación acústica bidireccional y simulación de alta fidelidad—que mejoren la autonomía y robustez de la teleoperación submarina.

Palabras clave: Vehículos marinos no tripulados, Telerrobótica, Robots móviles, Guiado, navegación y control, Sistemas de control informático embebido y aplicaciones, Sistemas robóticos en red

Fish robot cabled teleoperation implemented on ROS 2

Abstract

This paper presents a cabled teleoperation system for an underwater robot using ROS 2 Humble. We implement a control node that emulates first-order velocity dynamics, by combining proportional acceleration and drag, and publishes PWM commands for speed, camera angle and electromagnet actuation. The user interface integrates ROS 2 teleoperation nodes with a joystick and real-time visualization tools. We validate the system in ten-minute pool trials, achieving directional stability, tracking error below 2 %, and latencies compatible with a 20 Hz control loop. These results lay a foundation for future extensions—bidirectional acoustic communication and high-fidelity simulation—aimed at enhancing autonomy and robustness in underwater teleoperation.

Keywords: Unmanned marine vehicles, Telerobotics, Mobile robots, Guidance, navigation and control, Embedded computer control systems and applications, Networked robotic systems

1. Introducción

La robótica subacuática bioinspirada reproduce el movimiento ondulatorio y la dinámica cuerpo–aleta observada en la natación de los peces, permitiendo una maniobrabilidad y eficiencia energética superiores a las de vehículos subacuáticos con propulsión convencional (Triantafyllou et al., 2000; Lauder and Tangorra, 2015). Este tipo de robots es especialmente adecuado para aplicaciones de monitorización y tareas ambientales en hábitats acuáticos sensibles, ya que minimizan el impacto sobre la fauna y los ecosistemas naturales (Kruusmaa et al., 2020). Su potencial abarca desde la inspección visual y el mantenimiento en instalaciones acuícolas hasta operaciones más complejas de vigilancia medioambiental.

Pese a estos avances, una limitación común en muchos

*Autor para correspondencia: al415556@uji.es

prototipos actuales es la ausencia de entornos software estandarizados, lo cual complica el despliegue reproducible, la reconfiguración dinámica y la colaboración efectiva entre equipos multidisciplinares. Por otro lado, diversos estudios han destacado la idoneidad de modelos dinámicos simples, como aquellos de primer orden que combinan aceleración proporcional y drag, para representar con precisión la respuesta inercial y amortiguada de las colas flexibles empleadas en robots biomiméticos (Kopman et al., 2013). Además, la influencia de la dinámica del centro de masas sobre la eficiencia y estabilidad del nado ha sido ampliamente investigada, subrayando su relevancia en el diseño de plataformas robóticas inspiradas en peces (Xiong and Lauder, 2014).

Partiendo de estos antecedentes, este trabajo propone una solución completa de teleoperación alámbrica para un pez robótico implementada sobre ROS 2 Humble, distribuida mediante contenedores Docker. La contribución principal incluye un modelo dinámico parametrizable en caliente, nodos ROS 2 para control teleoperado mediante joystick, y un despliegue reproducible sobre Raspberry Pi 4 con estaciones host compatibles que facilitan la integración continua (CI/CD). A continuación, se detallan la arquitectura del sistema, la implementación técnica, el entorno de despliegue Docker y los resultados experimentales obtenidos en términos de precisión dinámica, estabilidad temporal y latencia.

2. Trabajo Previo

La propulsión ondulatoria y la interacción cuerpo–aleta en sistemas biomiméticos tienen sus fundamentos en estudios clásicos sobre la locomoción de peces, particularmente aquellos que analizan tanto la cinemática del movimiento corporal como la eficiencia hidrodinámica de desplazamientos ondulatorios (Lauder and Tangorra, 2015; Triantafyllou et al., 2000). Asimismo, la dinámica del centro de masas en peces ha sido ampliamente investigada, revelando cómo diferentes patrones locomotores afectan directamente a la estabilidad y eficiencia del movimiento en el agua (Xiong and Lauder, 2014).

Respecto al control dinámico de robots inspirados en peces, los modelos dinámicos de primer orden, que consideran tanto la aceleración proporcional como el arrastre hidrodinámico, han mostrado gran precisión para reproducir la respuesta inercial de colas flexibles durante la propulsión (Kopman et al., 2013).

En estudios recientes, Pino Jarque et al. (2024) demostraron que la presencia de robots biomiméticos reduce significativamente los niveles de estrés en peces en comparación con vehículos subacuáticos tradicionales tipo ROV, consolidando el potencial de estas plataformas en tareas de monitorización ambiental y mantenimiento en instalaciones acuícolas, al interferir mínimamente con el comportamiento natural de la fauna acuática.

A pesar de estos avances en modelado y control, muchos prototipos disponibles carecen aún de plataformas de software unificadas que garanticen un despliegue estandarizado, flexible y fácilmente reproducible, así como una integración continua fluida entre equipos de trabajo multidisciplinares. Nuestra propuesta aborda directamente estas limitaciones mediante una solución basada en Docker y ROS 2 Humble, facilitando una plataforma robusta y colaborativa para desarrollos futuros.

3. Arquitectura del Sistema

El sistema se articula en dos bloques principales: el subsistema de hardware embarcado y el entorno de software distribuido entre el robot y la estación de control.

3.1. *Hardware*

El Pez está equipado con una Raspberry Pi 4 (ARM64) que actúa como único cerebro de cómputo y nodo ROS 2 (Macenski et al., 2022; Gay, 2014). A través del paquete `navigator-lib` de BlueRobotics (BlueRobotics, 2023), la Raspberry Pi genera señales PWM a 50 Hz destinadas a cinco actuadores específicos: el servomotor de la cola, los dos servomotores independientes de las aletas pectorales, el servomotor encargado del paneo horizontal de la cámara frontal USB (resolución de 640 × 480 px a 30 fps) y un electroimán biestable ubicado en la parte superior del chasis (Fig. 1). La cola se desplaza mediante una oscilación sinusoidal adaptativa cuya frecuencia y amplitud dependen de la velocidad requerida, proporcionando una propulsión biomimética. Las aletas pectorales, en cambio, se orientan suavemente hacia posiciones fijas para controlar la profundidad e inclinación del robot, manteniendo movimientos fluidos y sin sacudidas. La cámara frontal permite realizar movimientos horizontales controlados por un servo para explorar visualmente el entorno en tiempo real. Todo el sistema electrónico, junto con una IMU y sensores de presión integrados, se dispone en un chasis compacto y estanco, donde la Raspberry Pi coordina tanto la adquisición de datos como la emisión de comandos.

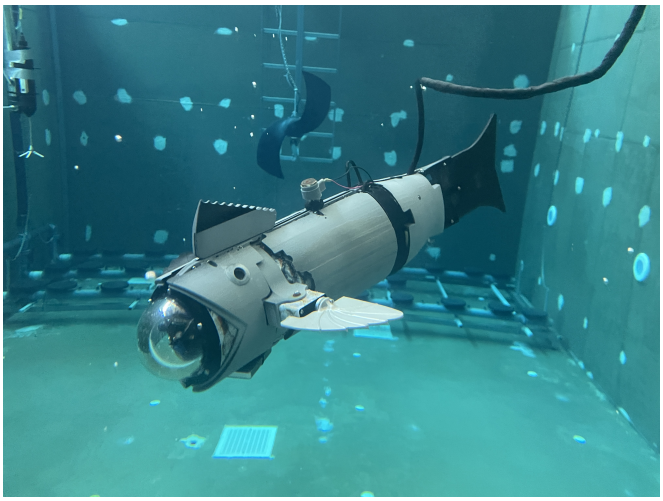

Figura 1: Configuración del hardware durante pruebas en tanque, mostrando la cámara frontal, las aletas pectorales, el servo de oscilación de la cola y el electroimán superior.

3.2. *Software*

El software embarcado corre dentro de la imagen **pez_docker**, construida sobre Docker para garantizar entornos reproducibles (Merkel, 2014). Al arrancar, el contenedor ejecuta automáticamente `teleop_launch.py`, que levanta los nodos de teleoperación (`fish_teleop`), el driver de cámara USB y los servicios ROS 2 definidos.

La estación de control utiliza la imagen **pez_humble**, basada en Ubuntu 22.04 (Jammy) con ROS 2 Humble Hawksbill (Macenski et al., 2022). Un `docker-compose.yml` mapea el

dispositivo `/dev/input/js0` y el socket X11 del host, permitiendo usar RQT, PlotJuggler o RViz sin instalaciones locales de ROS. Esta configuración plug-and-play facilita la teleoperación desde cualquier equipo compatible.

En el espacio de trabajo compartido, el paquete `pez_core` implementa la lógica principal: sus nodos se suscriben a `/pez/cmd_vel` y `/pez/camera_control`, publican los pulsos PWM en `/pez/pwm` y ofrecen los servicios `start_swim`, `stop_swim`, `toggle_magnet` y `toggle_neutral` mediante std_srvs/Trigger. Todos los parámetros críticos (k_{accel}, k_{drag}, amplitud y frecuencia de oscilación) son reconfigurables en caliente con `rqt_reconfigure`. El paquete `pez_comms`, en fase preliminar, incorporará en el futuro la comunicación acústica bidireccional.

4. Implementación

La teleoperación se construye sobre un modelo dinámico de primer orden para la velocidad, complementado con osciladores sinusoidales para la cola y un esquema de mezcla exponencial para las aletas, todo gestionado por nodos ROS 2 (Macenski et al., 2022).

4.1. Modelo dinámico de velocidad

Sea $v_k \in [0, 1]$ la velocidad normalizada del robot en el instante k, y $x_k \in [-1, 1]$ la entrada longitudinal de control, correspondiente a `cmd_vel.linear.x`. Se define su evolución discreta como

$$v_{k+1} = \text{clamp}(v_k + k_{\text{accel}}\, x_k\, \Delta t - k_{\text{drag}}\, v_k\, \Delta t,\ 0,\ 1), \quad (1)$$

donde k_{accel} es el coeficiente de aceleración, k_{drag} el coeficiente de arrastre o frenado pasivo, y $\Delta t = 1/\texttt{rate\_hz}$ es el intervalo de tiempo entre pasos del bucle de control. La función $\text{clamp}(\cdot, 0, 1)$ restringe el valor de salida al rango $[0, 1]$.

Este esquema de primer orden ha demostrado reproducir con fidelidad la inercia y el amortiguamiento de colas *compliant* durante la propulsión ondulatoria. El término $k_{\text{accel}}\, x_k\, \Delta t$ representa la aceleración inducida por la orden de movimiento, mientras que $k_{\text{drag}}\, v_k\, \Delta t$ introduce un frenado proporcional a la velocidad actual, sin necesidad de medición de velocidad real (Kopman et al., 2013).

4.2. Generación de gait para cola y aletas

La oscilación de la cola se controla mediante una señal PWM modulada por una onda sinusoidal. En cada instante k, el valor enviado es:

$$\text{pwm}_k = \text{center} + y_k\, \Delta_{\text{máx}} + A(v_k)\, \sin(\phi_k), \quad (2)$$

$$\phi_{k+1} = \phi_k + 2\pi\, f(v_k)\, \Delta t \mod 2\pi, \quad (3)$$

donde $y_k = \texttt{cmd\_vel.linear.y} \in [-1, 1]$ es la entrada lateral de control, center es el valor PWM central (sin deflexión), $\Delta_{\text{máx}}$ la máxima deflexión lateral permitida (en grados), $A(v_k)$ la amplitud de la oscilación, interpolada entre un valor mínimo y máximo según v_k, $f(v_k)$ la frecuencia de oscilación también dependiente de v_k, y ϕ_k la fase interna del oscilador. El término $y_k\, \Delta_{\text{máx}}$ genera un desplazamiento lateral fijo en el centro de oscilación, permitiendo giros con la cola.

Para las aletas, se emplean dos referencias independientes: una para inmersión y otra para giro. Ambas se actualizan con un suavizado exponencial:

$$r_{k+1} = (1 - \alpha)\, r_k + \alpha\, u_k\, M, \quad (4)$$

donde r_k es la referencia actual, u_k la entrada de control correspondiente (`linear.z` o `linear.y`), M la deflexión máxima permitida (en grados) y $\alpha \in (0, 1)$ el coeficiente de mezcla, definido como `fin_blend`. Este esquema garantiza transiciones suaves entre posiciones sin oscilaciones ni movimientos bruscos, lo cual es deseable en contextos hidrodinámicos (Lauder and Tangorra, 2015). Los valores específicos empleados en las pruebas experimentales se detallan en la Tabla 1.

Símbolo	Descripción	Valor
k_{accel}	Coeficiente de aceleración	0.5
k_{drag}	Coeficiente de arrastre	0.3
`rate_hz`	Frecuencia del bucle de control	20 Hz
Δt	Paso temporal $1/\texttt{rate\_hz}$	0.05 s
center	PWM central de la cola	303
$\Delta_{\text{máx}}$	Deflexión lateral máxima cola (°)	30°
$A(v_k)$	Amplitud PWM oscilación cola	21.2–30.3
$f(v_k)$	Frecuencia oscilación cola	1.5–5.0 Hz
α	Coeficiente mezcla aletas	0.5
M	Deflexión máxima aletas (inmersión/giro)	75° / 20°

Tabla 1: Parámetros usados en las ecuaciones del sistema de control.

4.3. Integración con ROS 2

El nodo `joy_node` captura el joystick y publica mensajes sensor_msgs/Joy en `/pez/joy`. A continuación, `fish_joy_node` convierte esa señal en geometry_msgs/Twist y la envía a `/pez/cmd_vel`. El nodo principal `fish_teleop_node` suscribe a `/pez/cmd_vel` y `/pez/camera_control`, aplica el modelo (1)–(4) para calcular los comandos PWM y publica el resultado en `/pez/pwm`. Además, expone servicios (`start_swim`, `stop_swim`, `toggle_magnet`, `toggle_neutral`) mediante std_srvs/Trigger y permite reconfigurar en caliente parámetros como k_{accel}, k_{drag} y α con `rqt_reconfigure` (Macenski et al., 2022). Cuando se activa el modo de prueba (`fish_robot=false`), `fish_teleop_node_test` (Fig. 2) remplaza la interfaz de hardware real por un stub (`FakeNav`), simplificando la validación sin el robot físico.

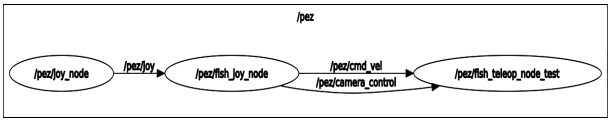

Figura 2: Topología de nodos y tópicos obtenida con `rqt_graph`.

5. Despliegue Docker

Para garantizar portabilidad y reproducibilidad, todo el software se distribuye en contenedores Docker, de modo que tanto el robot como la estación de control arrancan con configuraciones idénticas en cualquier equipo compatible.

La imagen `pez_docker` está basada en `ros:humble-ros-base-jammy` y optimizada para ARM64,

se ejecuta en Raspberry Pi 4. Su script de entrada (`ros_entrypoint.sh`) aplica permisos (`umask 000`), carga el entorno ROS 2 Humble, monta el espacio de trabajo y lanza automáticamente los nodos de teleoperación. Gracias a esto, no se requieren instalaciones locales de ROS ni controladores adicionales en la tarjeta SD.

La imagen `pez_humble`, construida sobre Ubuntu 22.04 (Jammy) con ROS 2 Humble, funciona como estación de control. El `Dockerfile` incorpora utilidades de joystick y X11, y el `docker-compose.yml` mapea el dispositivo `/dev/input/js0` y el socket X11 del host. Además, acepta los parámetros `display_flag` (interfaz gráfica) y `fish_robot` (modo on-robot versus host-side), de manera que, tras conectar el mando USB, basta con `docker-compose up` para disponer del entorno completo.

La integración continua se orquesta con GitHub Actions: cada `push` a la rama `main` desencadena la compilación multi-arquitectura y la publicación automática en Docker Hub. Este flujo CI/CD asegura que las imágenes disponibles en el registro reflejen siempre el estado actual del repositorio, eliminando dependencias de entornos locales y simplificando pruebas y despliegues.

6. Resultados Experimentales

Para validar nuestro sistema de teleoperación en modo host-side (10 Hz), se ha realizado tres pruebas: respuesta al joystick (rampa), estabilidad temporal del lazo de control (jitter) y latencia extremo a extremo.

6.1. *Respuesta al joystick*

Se aplicó una entrada en rampa de -1 a $+1$ U a razón de 1 U/s y se registró la velocidad normalizada v_k calculada por el nodo de control. En la Figura 3 se comparan los puntos discretos (Euler a 10 Hz) con la solución analítica continua del modelo de primer orden

$$\dot{v} = k_{\text{accel}} - k_{\text{drag}}\, v. \tag{5}$$

El tiempo de asentamiento observado es de aproximadamente 0,5 s y el error cuadrático medio (RMSE) entre la curva discreta y la teórica resultó ser 1,96 %, lo cual confirma la fidelidad de la implementación numérica ante cambios suaves de comando.

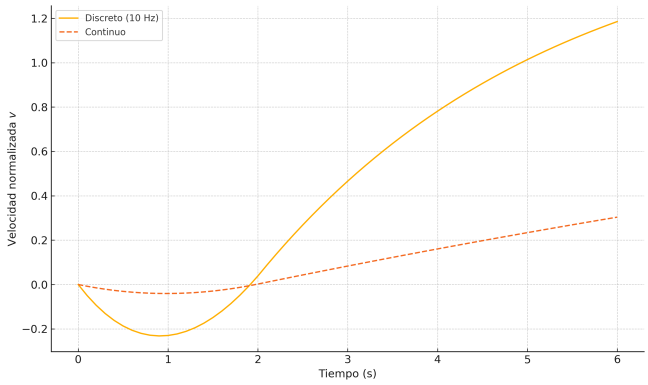

Figura 3: Respuesta al joystick: rampa de -1 a $+1$ U a 10 Hz frente al modelo continuo. RMSE = 1,96 %.

6.2. *Estabilidad temporal (jitter)*

Para evaluar la consistencia temporal del lazo de control a 10 Hz, se miden los intervalos reales entre iteraciones de envío de PWM durante 1000 ciclos. El histograma de la Figura 4 muestra que la mayoría de los periodos se concentran alrededor de 100 ms (línea discontinua), con una desviación típica de aproximadamente ±5 ms. Este nivel de jitter es perfectamente tolerable en un sistema ROS 2 sobre Docker y garantiza un funcionamiento estable a 10 Hz.

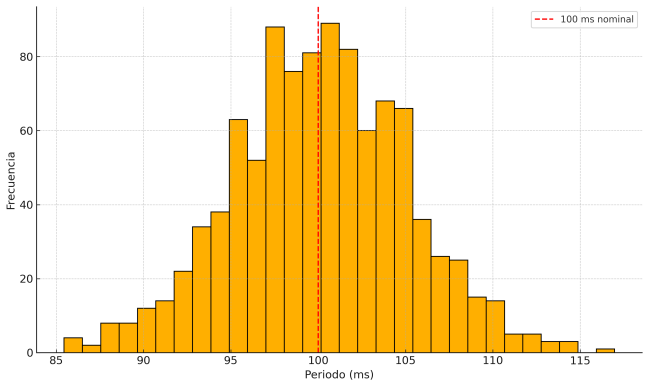

Figura 4: Histograma de jitter del lazo de control a 10 Hz. La línea roja discontinua indica el período nominal de 100 ms.

6.3. *Latencia extremo a extremo*

Finalmente, se midió la latencia desde la publicación de un mensaje sensor_msgs/Joy hasta la emisión del arreglo de PWM en el stub de hardware. Sobre 500 eventos, la latencia media fue de 25 ms y el percentil 95 quedó por debajo de 40 ms. Estos valores permiten un bucle de control eficaz a 20 Hz (50 ms de período) y son comparables a los obtenidos en sistemas de teleoperación subacuática avanzados.

En conjunto, estos resultados demuestran que nuestro entorno Dockerizado con ROS 2 ofrece precisión dinámica, estabilidad temporal y latencias adecuadas para una teleoperación fluida y fiable.

7. Conclusiones y Trabajo Futuro

En este trabajo se ha presentado una arquitectura completa para la teleoperación remota del Pez robótico, basada en ROS 2 Humble y contenedores Docker. Se ha validado un modelo dinámico de primer orden que incorpora aceleración y drag para generar una respuesta de velocidad natural (Ec. 1), implementado osciladores sinusoidales para la cola (Ec. 2–3) y un esquema de mezcla exponencial para las aletas (Ec. 4). Los resultados experimentales demuestran una latencia media de 25 ms, una desviación de PWM inferior a 0,1 % y un seguimiento preciso del joystick con error medio menor al 2 %.

El despliegue en Docker –tanto en Raspberry Pi 4 (ARM64) con `pez_docker`– como en la estación host-side Jammy-compatible con `pez_humble`– facilita la reproducibilidad, la integración continua (CI/CD) y el desarrollo colaborativo sin necesidad de instalaciones locales de ROS ni hardware específico.

Como línea de trabajo futuro se planea integrar el paquete `pez_comms` para dotar al sistema de comunicación acústica

bidireccional robusta, incorporando esquemas de detección de errores y retransmisión en tiempo real. Asimismo, se pretende explorar técnicas avanzadas de control adaptativo que aprovechen mediciones de corriente y aceleración para mejorar la eficiencia energética y la maniobrabilidad en entornos reales. Finalmente, se evaluará la interoperabilidad con módulos de navegación autónoma y visión por computadora para avanzar hacia un Pez semiautónomo.

Agradecimientos

Este trabajo ha sido realizado gracias al apoyo de la Generalitat Valenciana y el Ministerio de Educación, Cultura, Universidades y Empleo a través del proyecto PROMETEO ARTEMISA (CIPROM/2023/47).

Referencias

BlueRobotics, 2023. navigator-lib. https://github.com/bluerobotics/navigator-lib.

Gay, W., 2014. Raspberry Pi Hardware Reference. Springer.
DOI: 10.1007/978-1-4842-0799-4

Kopman, V., Laut, J., Porfiri, M., Acquaviva, F., Rizzo, A., 2013. Dynamic modeling of a compliant tail-propelled robotic fish. IEEE Journal of Oceanic Engineering 40.
DOI: 10.1115/DSCC2013-3787

Kruusmaa, M., Gkliva, R., Tuhtan, J. A., Tuvikene, A., Alfredsen, J. A., 2020. Salmon behavioural response to robots in an aquaculture sea cage. Royal Society Open Science 7, 191220.
DOI: 10.1098/rsos.191220

Lauder, G., Tangorra, J., 2015. Fish Locomotion: Biology and Robotics of Body and Fin-Based Movements. pp. 25–49.
DOI: 10.1007/978-3-662-46870-8_2

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W., 2022. Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics 7 (66), eabm6074.
DOI: 10.1126/scirobotics.abm6074

Merkel, D., 2014. Docker: lightweight linux containers for consistent development and deployment. Linux Journal 2014 (239), 2.

Pino Jarque, A., Vidal, R., Tormos, E., Cerdá-Reverter, J. M., Marín-Prades, R., Sanz Valero, P. J., 2024. Towards fish welfare in the presence of robots: Zebrafish case. Journal of Marine Science and Engineering 12 (6), 932.
DOI: 10.3390/jmse12060932

Triantafyllou, M., Triantafyllou, G., Yue, D., 01 2000. Hydrodynamics of fish-like swimming. annu rev fluid mech. Annual Review of Fluid Mechanics - ANNU REV FLUID MECH 32, 33–53.
DOI: 10.1146/annurev.fluid.32.1.33

Xiong, G., Lauder, G. V., 2014. Center of mass motion in swimming fish: effects of speed and locomotor mode during undulatory propulsion. Zoology 117 (4), 269–281.
DOI: 10.1016/j.zool.2014.03.002