

Evaluation of Control System Responsiveness by Different Linkage Methods between ROS2 and Unity: Application to a teleoperated fishing robot

Shuto Nomura†, Yoshiyuki Matsumura†* (Member) and Koji Kinoshita†

Abstract: In a remote-controlled fishing avatar robot, we build a simulation environment that utilizes 3D stereoscopic media technology by integrating ROS2 and Unity, assuming control of a real machine, and investigate the communication overhead of the robot control system. In order to enjoy outdoor activities such as fishing in a remote location, technology is required that instantly reflects the user's movements on the robot. Therefore, we aim to achieve natural operability by equipping the robot with a control system with higher responsiveness in terms of the communication method. In particular, we focus on the cooperation method between ROS2 and Unity, and compare and examine three systems with different characteristics such as the tools used, communication method, and servo motor control method. In the verification experiment, we evaluate the system's response time, CPU usage, and memory usage, evaluate the performance of a remote-controlled fishing robot using the system, and evaluate the psychological aspects of operating the fishing robot, and consider the practicality of the fishing robot.

Keywords: Avatar Fishing Robot, Remote Robot Control, Linkage Methods between ROS2 and Unity, Communication Delay

1. Introduction

In recent years, in the field of robot simulation, there has been a demand for the creation of a digital twin environment that uses 3D media technology to reproduce real-world objects such as buildings and products and their movements in real time in a digital virtual space.¹⁾ Recently, game engines are often used to create such environments. Game engines are a set of software for creating video games and interactive content, and are intended to provide the tools and functions necessary for game production, such as graphics, physics engines, audio, animation, and artificial intelligence, and to increase the efficiency and flexibility of the development process. They are capable of high-quality 3D rendering and physical calculations, and can realistically reproduce the surrounding environment and physical effects, making it possible to perform simulations and evaluations that are closer to reality. For these reasons,

Since then, it has been used not only for games and video content, but also for mechatronics robot simulations. Furthermore, game engineers are also using it for the Metaverse, which realizes communication in a 3D virtual space on the Internet.

Development examples by AI have also been attracting attention, and the places where users spend their time are changing from the real world to the Internet and then to three-dimensional virtual space.

However, in a digital space that does not involve physical movement, although it is efficient, it is difficult to create spontaneity, and there are concerns that it is not possible to enjoy the empathy and spatial value that comes from human-robot interaction.

An extension of this topic is avatars from 3D digital space.

The concept of the avatar robot is a type of remote-controlled robot that combines cutting-edge technologies such as robotics, AI, VR, communication, and haptic technology. The system allows users to control the robot via the Internet.

It is possible to transmit consciousness, skills, and presence to move, communicate, and perform tasks. By controlling an avatar robot, it is expected that a service will be developed that allows users to experience "telepresence" (a remote presence) as if they were actually there. In this case, high responsiveness is required for remote control of the robot. Therefore, technology is required to instantly reflect control command values generated on a game engine on an actual robot.

In this paper, we use the concept of avatar robots as the subject of a remote-controlled fishing robot. By integrating the robot development software ROS2 (Robot Operating System 2) with the game engine Unity, we build a 3D robot simulation environment that assumes control of the actual robot, and consider a Sim2Real robot control system that is highly responsive to the output. Specifically, in order to enjoy outdoor activities such as fishing in remote locations, we focus on communication technology that instantly reflects the user's movements on the avatar robot, and develop a highly responsive control system.

2024年3月29日受付, 2024年6月6日再受付, 2024年7月12日採録
† 富山大学

(〒930-8555 富山市五福3190, TEL 076-445-6011)

* Correspondence

[Copyrights to Machine Translated Content]

The copyright of the original papers published on this website belongs to the Institute of Image Information and Television Japanese. Unauthorized use of original papers or translated papers is prohibited. Please be sure to cite the original publication when referencing. For details, please refer to the copyright regulations of the Institute of Image Information and Television Engineers.

ROS2-Unity間の連携方法の違いによる制御システムの応答性評価

～遠隔操作型釣りロボットへの適用～

Evaluation of Control System Responsiveness by Different Linkage Methods between ROS2 and Unity: Application to a teleoperated fishing robot

野村 柊斗[†], 正会員 松村 嘉之^{†*}, 木下 功士[†]
Shuto Nomura[†], Yoshiyuki Matsumura^{†*} and Koji Kinoshita^{†1}

あらまし 遠隔操作型釣りアバターロボットにおいて、ROS2とUnityを統合することで実機制御までを想定した3次元立体メディア技術を活用したシミュレーション環境を構築し、ロボット制御システムの通信オーバーヘッドについて調査する。釣りのようなアウトドアアクティビティを遠隔地で楽しむために、ユーザの動きをロボットに即座に反映する技術が求められる。そのため、通信方式の観点でより応答性の高い制御システムを搭載することで、自然な操作性の実現を目指す。特に、ROS2-Unity間の連携方式に着目し、使用ツール、通信手法、サーボモータの制御方法等の異なる特性を持つ三つのシステムを比較検討する。検証実験では、①システムの応答時間、CPU使用率、メモリー使用率の評価、②システムを適用した遠隔操作型釣りロボットの性能評価、③釣りロボット操作時の心理的評価を行い、釣りロボットの有用性を検討している。

キーワード：アバター釣りロボット、遠隔操作、Unity-ROS2連携方式、通信遅延

1. ま え が き

近年ロボットシミュレーションの分野では、建物や製品といった現実世界のモノやその動きを、3次元立体メディア技術を活用して、デジタルな仮想空間上にリアルタイムに再現するデジタルツイン環境の構築が求められている¹⁾。このような環境構築において、最近では、ゲームエンジンが活用される場合が多くなっている。このゲームエンジンとは、ビデオゲームやインタラクティブなコンテンツを作るための一連のソフトウェア群であり、グラフィックス、物理エンジン、音声、アニメーション、人工知能などのゲーム制作に必要なツールや機能を提供し、開発プロセスの効率化や柔軟性を高めることを目的としている。高品質な3次元描画や物理演算が可能で、周囲環境や物理的な影響までリアルに再現できるため、より現実に近いシミュレーションや評価が可能になっている。これらの理由から、ゲームや映像コンテンツのみならず、メカトロニクス系のロボットシミュレーションへの活用が進んでいる。さらに、インターネット上の3次元化された仮想空間でコミュニケーションを実現するメタバースでもゲームエンジンによる開発例も注目され、ユーザが過ごす場は実世界からインターネット、そして、3次元の仮想空間へと変化し

ている。しかし、物理的な移動を伴わないデジタル空間では、効率的ではあるが偶発性を生みづらく、人とロボットが触れ合うことの共感や空間の価値を享受できないといった話題も見えている。

この話題の延長に3次元デジタル空間からのアバターロボットのコンセプトがある。アバターロボットとは、ロボティクス、AI、VR、通信、触覚技術などの先端技術を合わせた一種の遠隔操作ロボットである。ユーザがインターネットを介してそのロボットを操作できる仕組みになっており、意識・技能・存在感を伝送させて、移動やコミュニケーション・作業を行うことができる。アバターロボットを操作することで、利用者があたかもそこにいるかのような「テレプレゼンス（遠隔存在感）」を体験するサービスが期待できる。この際にロボットの遠隔操作には高い応答性が求められる。そのため、ゲームエンジン上で生成した制御指令値を実機ロボットに即時的に反映させる技術が求められる。

本稿では、アバターロボットの考え方にに基づき、遠隔操作型釣りロボットを題材とする。ロボット開発ソフトウェアのROS2 (Robot Operating System 2) とゲームエンジンのUnityを統合することで、実機制御までを想定した3次元ロボットシミュレーション環境を構築し、その出力に対する応答性の高いSim2Realなロボット制御システムを検討する。具体的には、釣りのようなアウトドアアクティビティを遠隔地で楽しむために、即時にユーザの動きをアバターロボットに反映する通信技術に着目し、応答性の高い制御

2024年3月29日受付, 2024年6月6日再受付, 2024年7月12日採録

† 富山大学

(〒930-8555 富山市五福3190, TEL 076-445-6011)

* Correspondence

By constructing this system, we aim to achieve more natural operability. In particular, we focus on the method of cooperation between ROS2 and Unity, and construct and compare three systems with different characteristics by changing the packages used, communication methods, servo motor control methods, etc.

In the following, in this paper, Chapter 2 explains the control of a remote-controlled fishing robot, and in the verification experiment in Chapter 3, we evaluate the response time, the CPU usage and memory usage of the entire PC, and the CPU usage and memory usage of each process.

In addition, we measured the time it took to complete a series of fishing actions and conducted a psychological evaluation of the operability to evaluate the operability of the avatar fishing robot. Finally, we conclude our paper in Chapter 4.

2. Remote fishing system

Figure 1 shows a conceptual diagram of the whole operation of the remote-controlled fishing robot. The administrator in Figure 1 will not be directly involved in the fishing robot system after the initial startup of the whole system, but will be on standby to monitor and perform maintenance so that he can respond appropriately in the event of any malfunctions. In particular, in the evaluation experiments of responsiveness and operational performance in this paper, the user will act as the administrator, shutting down the system once after each experiment and collecting the experimental data. It should be noted that in the psychological evaluation experiments, a third party who was not a subject acted as the administrator and collected the data after the experiment, so as not to have a psychological effect on the subjects.

Table 1 shows the desktop PC environment used to build the system. The fishing robot arm has a two-axis configuration and can cast a line using a fishing rod attached to the tip. Each joint uses a Robotis Dynamixel servo motor (XM430-W350-T) that can provide real-time feedback on status such as position, speed, and current (torque/load). The servo motor and fishing rod are attached using an aluminum frame and connecting parts created with a 3D printer.

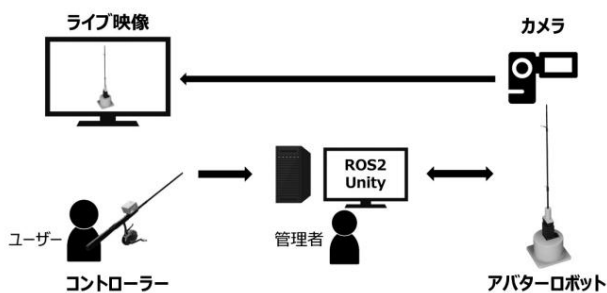


Figure 1. System concept of remote-controlled fishing robot

Table 1 Control PC environment

OS	Ubuntu 20.04.4 LTS 64{bit}
ROS	ROS2 Foxy Fitzroy
Unity Editor	2021.3.25f1
CPU	AMD Ryzen 5 5600X 6-Core Processor
Memory	16 [GB]
GPU	NVIDIA GeForce GTX 1660

The controller used by the user is a fishing rod with a microcomputer (Sony's Spresense) attached to the handle, which detects the user's movements with a gyro sensor connected to the microcomputer and transmits the measurement data to a PC as command values for controlling the robot. This gyro data corresponds to the angles of each joint in the robot arm, with the first joint corresponding to left-right data and the second joint corresponding to up-down data, and is reflected in the robot arm model on Unity. The motion trajectory of the Unity model is then transmitted to the servos of the actual robot via the ROS2 ecosystem.

is sent to the motor.

Figure 2 shows the state during operation. The current joint angles are fed back from the servo motors of the actual robot, so the movement of the actual robot can also be confirmed on Unity. Figure 3 shows an explanation of the Unity screen during robot operation. The robot installed in a remote location is photographed by a web camera, and the user operates it while watching the video sent in real time. This system enables remote fishing movements. Note that this paper focuses on the real-time control of the actual robot, and video communication technology will be discussed in the next paper.

2.1 Control system using ROS2/Unity

In this paper, we focus on a robot simulation environment that combines ROS2 and Unity from the perspective of integrating 3D media technology with real robot control in real time.

By integrating the development framework ROS2 and the game engine Unity, instant data sharing and visual representation is possible.

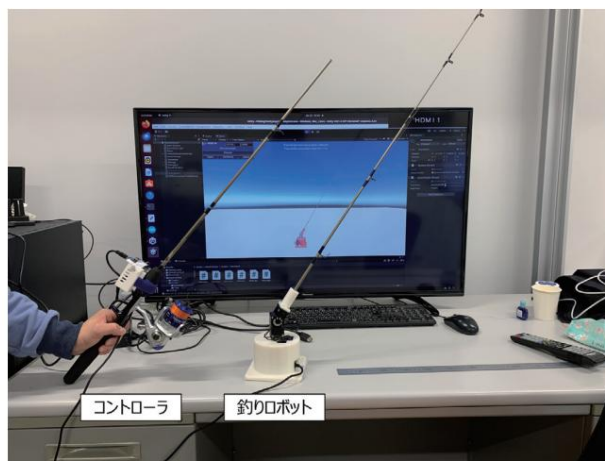


Figure 2. Controlling the fishing robot with a controller



Figure 3 Explanation of the Unity screen while operating the robot

[Copyrights to Machine Translated Content]

The copyright of the original papers published on this website belongs to the Institute of Image Information and Television Japanese. Unauthorized use of original papers or translated papers is prohibited. Please be sure to cite the original publication when referencing. For details, please refer to the copyright regulations of the Institute of Image Information and Television Engineers.

システムを構築することで、より自然な操作性の実現を目指す。特に、ROS2-Unity間の連携方法に着目し、使用パッケージ、通信手法、サーボモータの制御方法等を変化させた異なる特性を持つ三つのシステムを構築し比較する。

以下、本稿では、第2章において、遠隔操作型釣りロボット制御について説明し、第3章の検証実験では応答時間、PC全体のCPU使用率とメモリー使用率、プロセスごとのCPU使用率とメモリー使用率について評価を行う。さらに、一連の釣り動作にかかる時間を計測し、操作性に関する心理評価を行うことでアバター釣りロボットの操作感を評価する。最後に、第4章においてまとめる。

2. 遠隔釣りシステム

遠隔操作型釣りロボットを動かす全体の概念図を図1に示す。図1に配置された管理者は、最初の全体システム起動後は、基本的に釣りロボットシステムに直接関与しないが、何らかの不具合時には適宜対応できるように見守りメンテナンス待機を実施する。特に、本稿の応答性と動作性能の評価実験では、ユーザーが管理者となって各実験終了後にシステムを一旦終了し、実験データを回収する役割を担う。なお、心理的評価実験では、心理的な影響を被験者に及ぼさないように、被験者でない第三者が管理者になり、実験終了後にデータを回収していることを付記する。

表1にシステム構築に使用したデスクトップPCの環境を示す。釣りロボットアームは2軸構成とし、先端部に取り付けられた釣竿によって糸を垂らすことができる。各関節には位置、速度、電流(トルク/負荷)などの状態をリアルタイムにフィードバックできるRobotis社のDynamixelサーボモータ(XM430-W350-T)を使用する。アルミフレームと3Dプリンタで作成した接続部品によってサーボモータや釣竿を取り付けた。

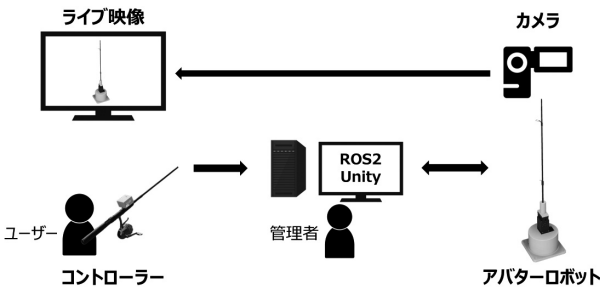


図1 遠隔操作型釣りロボットのシステム概念図

表1 制御用PCの環境

OS	Ubuntu 20.04.4 LTS 64[bit]
ROS	ROS2 Foxy Fitzroy
Unity Editor	2021.3.25f1
CPU	AMD Ryzen 5 5600X 6-Core Processor
Memory	16 [GB]
GPU	NVIDIA GeForce GTX 1660

ユーザーが使用するコントローラは、釣竿の持ち手にマイコン(Sony製SpreSense)を取り付けたもので、マイコンに接続したジャイロセンサによってユーザーの動きを検知し、計測データをロボット制御の指令値としてPCに通信する仕様としている。このジャイロデータは、ロボットアームの各関節角度に対応しており、第1関節は左右方向、第2関節は上下方向のデータに対応し、Unity上のロボットアームモデルに反映される。その後、Unityモデルの動作軌道がROS2エコシステムを介して実機ロボットのサーボモータに送信される。

操作中の様子を図2に示す。実機ロボットのサーボモータから、現在の関節角度がフィードバックされるため、実機ロボットの動きもUnity上で確認できる。ロボット操作中のUnity画面の説明を図3に示す。遠隔地に設置されたロボットは、Webカメラによって撮影され、ユーザーはリアルタイムに送られる映像を見ながら操作を行う。このシステムにより、遠隔での釣り動作を実現する。なお、本稿では、実機ロボットのリアルタイム制御に注目し、映像の通信技術は、次稿に譲る。

2.1 ROS2/Unityによる制御システム

本稿では、3次元立体メディア技術と即時的な実機ロボット制御との融合の観点で、ROS2とUnityを組み合わせたロボットシミュレーション環境に着目する。ロボット開発フレームワークのROS2とゲームエンジンのUnityを統合することで、即時のデータ共有や視覚的な表現が可能

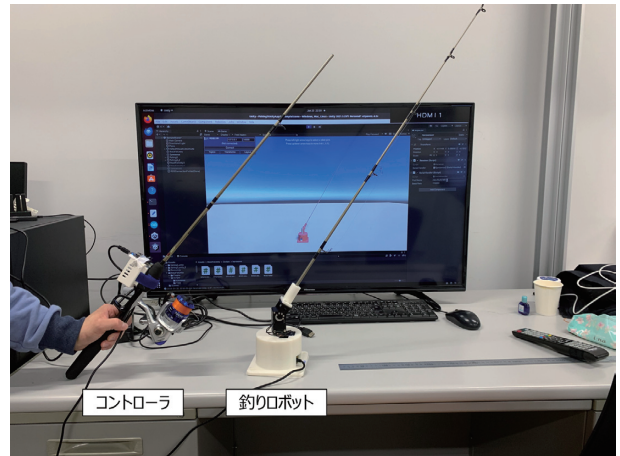


図2 釣りロボットをコントローラで操作する様子

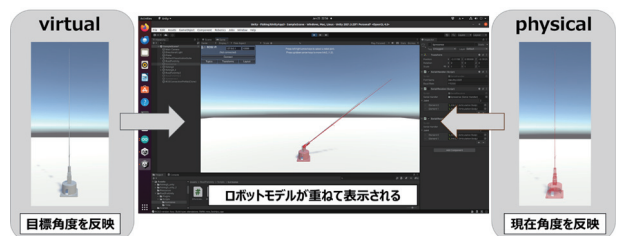


図3 ロボット操作中のUnity画面の説明

In order to maintain compatibility between the control programs of the real robot and the virtual robot, a bridge function is required to communicate control commands and sensor information between ROS2 and Unity.

The most important factor is the functionality. In this paper, we use Unity Robotics Hub and ROS2 for Unity as communication methods for integrating ROS2 and Unity, and compare their processing capabilities. Unity Robotics Hub is one of the methods for realizing communication between Unity and ROS2, and in reference 2), it was evaluated as having the smallest latency compared to other methods (ROS#, ROS.NET). On the other hand, ROS2 for Unity is a package that allows you to use the native functions of ROS2 within Unity. Both take different approaches, and each has its own characteristics and advantages. The method of controlling the servo motor is also important. We use two tools, Dynamixel SDK and ros2-control, to compare them. Three systems with different characteristics were built to make the comparison. The details of each are summarized in the next section.

2.2 Tools used for ROS2/Unity control

2.2.1 ROS2-Unity integration method (1) Unity

Robotics Hub³⁾ A project that compiles various samples and tutorials for performing robot simulation in Unity.

It provides tools to easily integrate Unity with existing ROS-based workflows.

shows the communication compatibility between Unity and ROS2 achieved by this technology.

Unity cannot communicate directly with ROS2, so ROS TCP Connector and ROS TCP

A ROS TCP Endpoint is provided. The ROS TCP Endpoint runs as a ROS application and sends ROS messages to ROS.

Converts to a format that can be communicated with the TCP Connector. ROS TCP

The Connector runs as a Unity plugin and is a ROS TCP

It is responsible for passing communication data received from the endpoint to the script in the Unity model.

Data exchange between ROS2 and Unity can be implemented via TCP communication.

(2) ROS2 for Unity⁴⁾

Unity and ROS2 ecosystem developed by Robotec.ai

It is a high-performance communication solution that connects ROS2 and Unity systems. It uses the ROS2 middleware stack and can communicate without a bridge.

Data exchange between ROS2 and Unity can be done using DDS, the standard communication protocol for ROS2, so there is no need for a normal bridge.

Compared to other ROS2 solutions, it has superior communication latency and information processing capabilities. Objects in Unity simulations can be treated as ROS2 nodes, publishers, and subscribers.

2.2.2 Servo motor control method (1)

Dynamixel SDK⁵⁾ This

SDK provides various source codes and tools for using Dynamixel servo motors. It is a software for controlling servo motors using packet communication.

It is a software development kit that can be used on desktop and laptop PCs, tablets, Raspberry Pi, etc.

It can be used with SBCs such as UpBoard and embedded boards that support the Arduino IDE. It can also be used as a ROS library using C++ and Python modules.

(2) ros2-control⁶⁾

It is a framework for controlling robots in real time using ROS2. It simplifies the integration of new hardware. It provides a controller based on control theory, enabling high-speed and accurate robot motion control.

It is closely linked to the ROS2 ecosystem, and can easily be linked to other ROS2 packages by utilizing ROS2 messages and services. It can be applied simply by adding the ros2_control tag to the URDF of the robot to be used, and the controls commonly used in the ros2-control framework are

Various control modes are available, such as position (angle), speed, acceleration, force and torque, corresponding to the roller.

Compared to the Dynamixel SDK, it is easier to implement a unique controller as a plug-in, and by selecting and applying the Controller Manager appropriately and managing access to the hardware interface

more appropriately, it is possible to easily reduce the redundancy of the Dynamixel SDK and achieve improvements that are suited to the intended use.

It is possible.

2.3 Details of the three control systems to be

compared 2.3.1 System 1

A conceptual diagram of System 1 is shown in Figure 5. This diagram shows that messages generated in Unity are transmitted to the Dynamixel servo motor via ROS2 to control the motor. System 1 uses the Unity Robotics Hub package to link ROS2 and Unity. When using Unity Robotics Hub, the robot control command values generated on Unity are sent via the ROS TCP Connector and ROS TCP

The endpoint converts the data into a ROS message and sends it to ROS2 via TCP.

Compared to using DDS communication, the latency may be slightly larger.

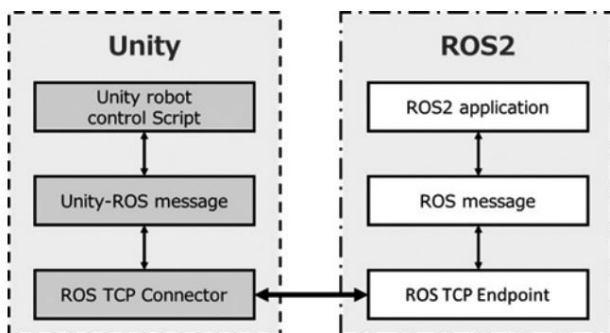


Figure 4. Data communication mechanism between ROS2 and Unity provided by Unity-Robotics-Hub

[Copyrights to Machine Translated Content]

The copyright of the original papers published on this website belongs to the Institute of Image Information and Television Japanese. Unauthorized use of original papers or translated papers is prohibited. Please be sure to cite the original publication when referencing. For details, please refer to the copyright regulations of the Institute of Image Information and Television Engineers.

となる。この時、実機ロボットと仮想ロボットの制御プログラムにおける互換性を保つためには、ROS2とUnity間で制御コマンドやセンサ情報を通信するためのブリッジ機能が最も重要な要素となる。本稿では、ROS2とUnityを統合するための通信手法として、Unity Robotics HubとROS2 for Unityの手法を採用し、処理能力の比較を行う。Unity Robotics Hubは、UnityとROS2間の通信を実現するための手法の一つであり、文献²⁾では、他の手法(ROS#, ROS.NET)と比較した際に、最も遅延が小さかったことが評価されている。一方、ROS2 for Unityは、Unity内でROS2のネイティブな機能を利用できるようにするためのパッケージである。両者は異なるアプローチを取っており、それぞれの特性や利点がある。また、サーボモータの制御方法も重要になる。Dynamixel SDKとros2-controlの二つのツールを使用し比較する。比較を行うために異なる特性を持つ三つのシステムを構築した。それぞれの詳細を次節にてまとめる。

2.2 ROS2/Unity制御で使用するツール群

2.2.1 ROS2-Unityの連携方式

(1) Unity Robotics Hub³⁾

Unityでロボットシミュレーションを行うためのさまざまなサンプルやチュートリアルがまとめられたプロジェクトとなっている。Unityと既存のROSベースのワークフローを簡単に連携するためのツールを提供している。図4に本技術で実現されるUnityとROS2の通信対応を示す。UnityからROS2への通信は直接行えないため、データを仲介する役割を担うROS TCP ConnectorおよびROS TCP Endpointが提供されている。ROS TCP EndpointはROSアプリケーションとして動作し、ROSメッセージをROS TCP Connectorと通信できる形式に変換する。ROS TCP ConnectorはUnityプラグインとして動作し、ROS TCP Endpointから受け取った通信データをUnityのモデル内のスクリプトに受け渡す役割を担う。この機能により、ROS2-Unity間のデータ交換をTCP通信により実装できる。

(2) ROS2 for Unity⁴⁾

Robotec.aiによって開発されたUnityとROS2エコシステ

ムを接続する高性能な通信ソリューションである。ROS2のミドルウェアスタックを利用し、ブリッジなしで通信することができる。ROS2-Unity間のデータ交換をROS2の標準通信規格であるDDSで通信できるため、通常のブリッジングソリューションと比較し、通信レイテンシと情報処理能力に優れている。Unityシミュレーション内のオブジェクトをROS2のノードやパブリッシャー、サブスクリバとして扱うことができる。

2.2.2 サーボモータの制御方式

(1) Dynamixel SDK⁵⁾

このSDKはDynamixelサーボモータを使うためのさまざまなソースコードやツールを提供している。これは、パケット通信を使用してサーボモータを制御するためのソフトウェア開発キットとなっている。デスクトップやラップトップなどのPCやタブレットの他、Raspberry PiやUpBoard等のSBC、Arduino IDEをサポートする組み込みボードでも使用できる。C++やPythonのモジュールを使用してROSライブラリとして使用できる。

(2) ros2-control⁶⁾

ROS2を使ってロボットをリアルタイムで制御するためのフレームワークとなっている。新しいハードウェアの統合を簡単にする。制御理論に基づいたコントローラを提供し、高速かつ正確なロボットの動作制御を実現する。ROS2のエコシステムと密接に連携しており、ROS2のメッセージやサービスを活用することで、他のROS2パッケージと容易に連携できる。使用するロボットのURDFにros2_controlタグを追加するだけで適用することができ、ros2-controlフレームワークで一般的に使用されるコントローラに対応した位置(角度)、速度、加速度、力やトルク等のさまざまな制御モードを利用できる。特に、Dynamixel SDKに比して、独自コントローラの実装をプラグイン化しやすく、Controller Managerをうまく選定して適用し、ハードウェアインタフェースへのアクセスをより適切に管理することによって、Dynamixel SDKの冗長性を容易に減らすことで用途目的にあった改善を図ることができる。

2.3 比較する三つの制御システムの詳細

2.3.1 システム1

システム1の概念図を図5に示す。この図は、Unityで生成されたメッセージがROS2を介してDynamixelサーボモータに伝達され、モータを制御することを意味する。システム1ではROS2とUnityの連携にUnity Robotics Hubパッケージを利用する。Unity Robotics Hubを用いる場合はUnity上で生成されたロボット制御の指令値をデータ仲介の役割を担うROS TCP ConnectorおよびROS TCP EndpointによってROSメッセージに変換し、TCP通信によってROS2に送信する。ROS2の標準通信規格であるDDS通信を使用する場合と比較すると、ややレイテンシが大きくなる可能性が考えられる。また、Dynamixelサーボ

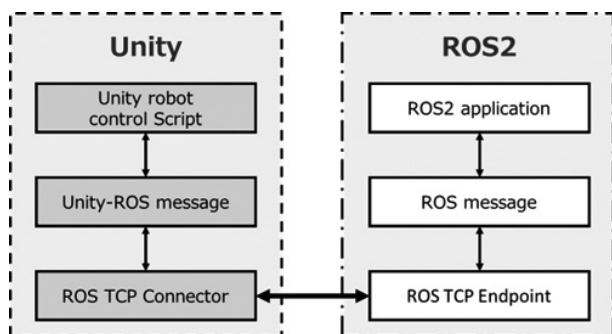


図4 Unity-Robotics-Hubが提供するROS2-Unity間のデータ通信の仕組み

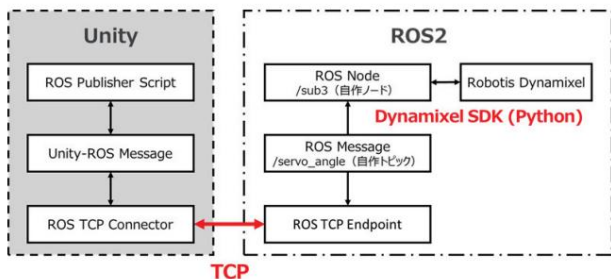


Figure 5. Conceptual diagram of System 1

The motors are controlled using the Dynamixel SDK, which transmits the target angle for each motor and receives the current angle of the motor, thereby controlling the position (angle).

2.3.2 System 2

A conceptual diagram of System 2 is shown in Figure 6. System 2 differs from System 1 in that it uses ros2-control in its servo motor control method. This system uses forward_command_controller7) as the control mode to control position (angle). The angle command value for the robot generated on Unity is sent to ROS2 via the forward_position_controller/commands topic of ROS2. ROS2 and Unity can be linked in the same way as System 1.

In order to use the Unity Robotics Hub package, the output data is sent via ROS TCP, which acts as a data intermediary between Unity and ROS2.

The data is collected once by the Connector and ROS TCP Endpoint, and then communicated. TCP communication is used as the data communication standard between ROS2 and Unity.

2.3.3 System 3

A conceptual diagram of System 3 is shown in Figure 7. System 3 uses the ROS2 for Unity package to link ROS2 and Unity.

In the ROS2 for Unity link, the control data generated on Unity is directly transmitted using the ROS2 message format, and the bridge is established.

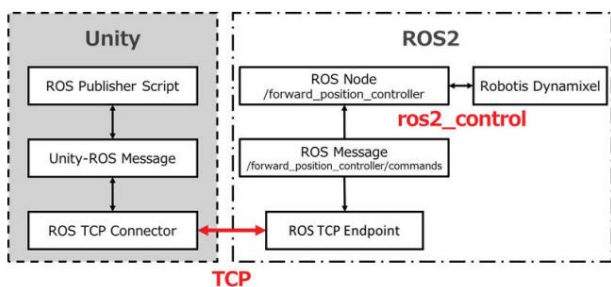


Figure 6. Conceptual diagram of System 2

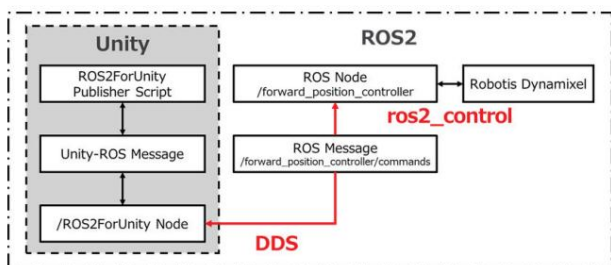


Figure 7. Conceptual diagram of System 3

This allows data transfer without requiring any modification.

ROS required when using the Robotics Hub package

This eliminates the need for intermediate processing such as TCP Connector.

This allows Unity applications to easily operate frameworks that rely on ROS message formats, such as ros2-control. Communication between ROS2 and Unity is via DDS, the standard communication protocol for ROS2. Servo motor control is done by a system.

As with Stem 2, position (angle) control is performed using ros2-control.

3. Evaluation experiment of three control systems

3.1 Response evaluation experiment

3.1.1 Comparison method for system responsiveness

In the responsiveness evaluation experiment, we will investigate the effects that differences in the ROS2 and Unity collaboration methods and the servo motor control methods have on system responsiveness, and consider a system that is suitable for robot control. Figure 8 shows the experimental method for comparing system responsiveness. This figure shows a series of processes in which the target angle of the Unity model is sent to the servo motor via ROS2 to perform control, and the current angle of the servo motor (encoder value) is fed back to Unity via ROS2. This series of processes will be performed in each of the three systems, and the differences between the systems will be investigated.

In the system comparison experiment of responsiveness, (i) response time and angle error, We quantitatively evaluate (ii) CPU usage and (iii) memory usage of the entire PC, and (iv) CPU usage and (v) memory usage of each process. The response time in this experiment was measured using the timestamp (Figure 8) obtained when sending the target angle from Unity.

The angle is defined as the difference between the time stamp (T1 in Figure 8) obtained when Unity receives the encoder angle information and the time stamp (T2 in Figure 8) obtained when Unity receives the encoder angle information. It is measured by outputting a log in which the time stamp (hour, minute, second, millisecond) and angle information are recorded using Unity script. The target angle sent from Unity to the servo motor was set to increase or decrease every 1.4 seconds. This angle information is generated by a microcontroller (Spresense) connected to a PC, sent to Unity via serial communication, and then sent to the servo motor every 10 milliseconds as the target

angle. The experimental procedure is shown below in (1) to (7).

- (1) Start the Unity Editor and wait for the simulation to start.

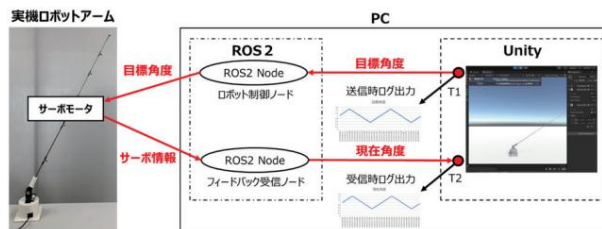


Fig. 8 Experimental method for comparing systems in terms of responsiveness

[Copyrights to Machine Translated Content]

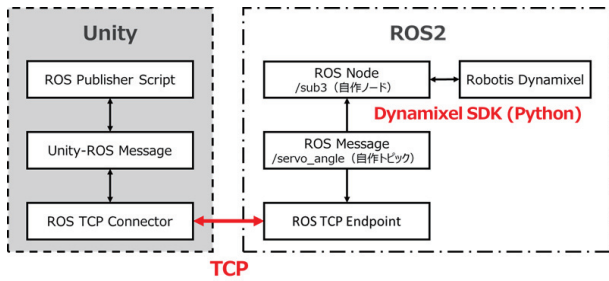


図5 システム1の概念図

モータの制御は、Dynamixel SDKを利用し、各モータに対する目標角度の送信やモータの現在角度を受信する設定等を行うことで、位置(角度)制御を行う。

2.3.2 システム2

システム2の概念図を図6に示す。システム2はシステム1に対し、サーボモータの制御方法を変更し、ros2-controlを用いる。本システムでは制御モードにforward_command_controller⁷⁾を利用し、位置(角度)制御を行う。Unity上で生成したロボットの角度指令値をROS2のforward_position_controller/commandsトピックを介し、ROS2に送信する。ROS2とUnityの連携にはシステム1と同様にUnity Robotics Hubパッケージを利用するため、出力データはUnityとROS2のデータ仲介を担うROS TCP ConnectorおよびROS TCP Endpointによって一度集約された後に通信される。ROS2-Unity間のデータ通信規格ではTCP通信を使用する。

2.3.3 システム3

システム3の概念図を図7に示す。システム3ではROS2とUnityの連携にROS2 for Unityパッケージを利用する。ROS2 for Unityによる連携では、Unity上で生成した制御データをROS2のメッセージ形式を直接利用し、ブリッジ

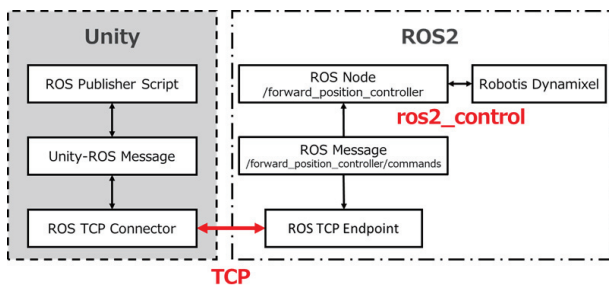


図6 システム2の概念図

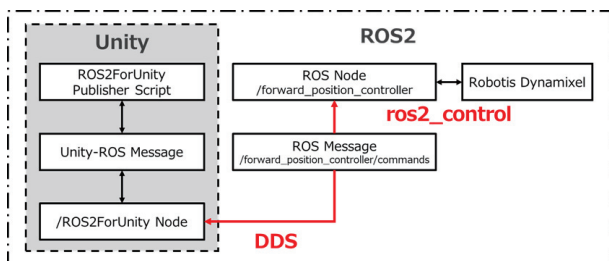


図7 システム3の概念図

なしでのデータ移動が可能になる。そのため、Unity Robotics Hubパッケージを利用した場合に必須になるROS TCP Connector等による中間処理が省略できる。これにより、ros2-control等のROSのメッセージ形式に依存したフレームワークをUnityアプリケーションから容易に操作できる。また、ROS2-Unity間の通信はROS2の標準通信規格であるDDSによって通信される。サーボモータの制御はシステム2と同様にros2-controlを利用し、位置(角度)制御を行う。

3. 三つの制御システムの評価実験

3.1 応答性評価実験

3.1.1 システムの応答性に関する比較方法

応答性の評価実験では、ROS2とUnityの連携手法とサーボモータの制御方法の違いがシステムの応答性に与える影響を調査し、ロボット制御に適したシステムを検討する。システムの応答性を比較する実験方法を図8に示す。この図は、Unityモデルの目標角度についてROS2を介してサーボモータに送信することで制御を行い、サーボモータの現在角度(エンコーダ値)についてROS2を介してUnityにフィードバックする一連の処理を示す。この一連の処理を三つのシステムそれぞれで実行し、システム間の差異を調査する。

応答性のシステム比較実験では、(i)応答時間と角度誤差、PC全体の(ii) CPU使用率と(iii) メモリー使用率、プロセスごとの(iv) CPU使用率と(v) メモリー使用率について定量的に評価する。本実験における応答時間は、Unityから目標角度を送信するときに取得したタイムスタンプ(図8中のT1)と、Unityがエンコーダの角度情報を受信したときに取得したタイムスタンプ(図8中のT2)の差として定義する。Unityスクリプトでタイムスタンプ(時、分、秒、ミリ秒)と角度情報が記録されたログを出力することで計測する。Unityからサーボモータ送信される目標角度は、1.4秒ごとに角度の増減が変わるように設定した。この角度情報は、PCに接続したマイコン(Sprensense)で生成し、シリアル通信でUnityに送信した後に、目標角度としてサーボモータに10ミリ秒毎送信される。

実験手順を以下(1)～(7)に示す。

- (1) Unity Editorを起動し、シミュレーション開始状態で待機させる。

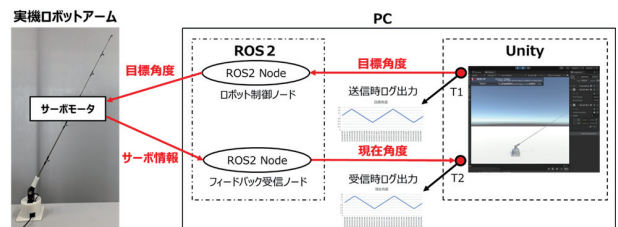


図8 応答性に関するシステム比較の実験方法

- (2) Start the Linux commands vmstat, free, and top in the background.
- (3) Wait three seconds for various processes to stabilize.

- (4) Program to control the robot after the process has stabilized run the above and measure for 17 seconds.

- (5) Measurement procedures (1) to (4) were performed five times for each system, and the response time, CPU usage and memory usage of the entire PC, and CPU usage and memory usage of each process were recorded.

- (6) The average values of the measurement results are tabulated for each system.

- (7) The differences between the systems are compared and investigated.

3.1.2 Evaluation Results

Table 2 shows the measured response time (the communication time from controlling the servo motor based on the target angle from Unity to feeding back the encoder value to Unity). It also shows the error between the fed back encoder value and the target angle. Figure 9 shows the change in the overall CPU usage of the PC over time.

Figure 10 shows the change in memory usage across the entire PC.

Tables 3 and 4 show the results of CPU usage and memory usage for each process at 15 seconds into the experiment. The specific usage for each process is shown below.

The connection process between ROS2 and Unity is broken down into items. CPU usage is up to 600% due to 6 cores, and memory usage is 16 GB.

The response time results in Table 2 show that System 1 has the shortest response time, followed by System 2 and System 3, and a significant difference was observed between all systems in the results of a multiple comparison of a two-group test ($p < 0.05$). From these results, it can be said that System 3 is suitable in situations where response time is important. Additionally, no significant difference was observed between the response times of Joints 1 and 2 in all systems ($p < 0.05$). This is attributed to the use of a two-axis robot arm. A more detailed investigation is required into the effect on response time when the number of joints is increased.

Although the three systems showed a response time of about 50 to 80 ms, it is necessary to investigate whether they meet the operational requirements for fishing. This will be evaluated through the experiments in Sections 3.2 and 3.3.

(i) Consideration of angle

The results of angle error in Table 2 all showed values between 0.71° and 0.78° , and no significant differences were found in any of the results of multiple comparisons of the two-group test ($p < 0.05$). Considering this, it cannot be said that there is a statistically significant difference between the systems.

(ii) CPU usage rate of the entire PC

Figure 9 shows that after Unity starts, the CPU usage rate is highest in System 1, followed by System 2 and System 3. According to literature 8), it is important to limit CPU usage in order to optimize ROS2 communication for real-time applications.

This is also a valid point in this experiment.

Table 2 Response time and angular error results

	joint 1		joint 2	
	応答時間(ms)	角度誤差(°)	応答時間(ms)	角度誤差(°)
システム 1	75.8	0.75	77.5	0.72
システム 2	64.5	0.75	64.7	0.71
システム 3	50.3	0.77	49.7	0.74

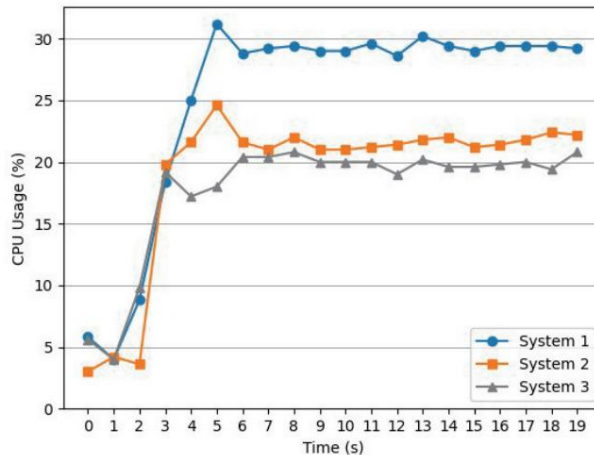


Figure 9. Overall PC CPU usage

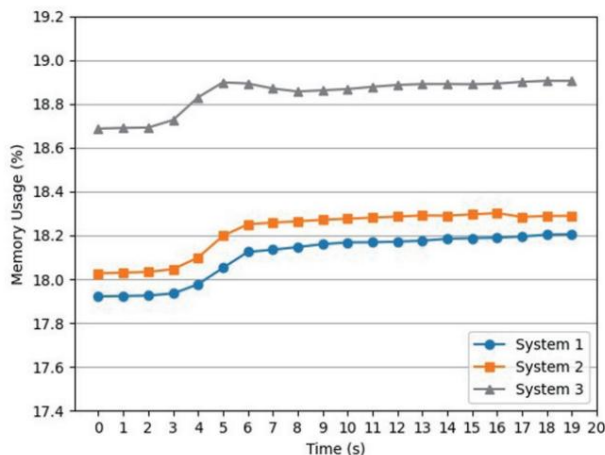


Figure 10. Overall PC memory usage

Table 3 CPU usage by process

システム名	Unityのプロセス		ROS2のプロセス		ROS2とUnityの接続
	プロセス名	Unity	python3	default_server_endpoint	default_server_endpoint
システム 1	使用率[%]	115.7	80.9	6.5	6.5
	使用率の合計[%]	115.7	80.9	6.5	6.5
システム 2	使用率[%]	116.7	3.8	1.0	8.3
	使用率の合計[%]	116.7	4.8	8.3	8.3
システム 3	使用率[%]	115.0	5.4	1.0	
	使用率の合計[%]	115.0	6.4		

Table 4. Memory usage by process

システム名	Unityのプロセス		ROS2のプロセス		ROS2とUnityの接続
	プロセス名	Unity	python3	default_server_endpoint	default_server_endpoint
システム 1	使用率[%]	4.7	0.2	0.4	0.4
	使用率の合計[%]	4.7	0.2	0.4	0.4
システム 2	使用率[%]	4.7	0.2	0.2	0.4
	使用率の合計[%]	4.7	0.4	0.4	0.4
システム 3	使用率[%]	5.5	0.2	0.1	
	使用率の合計[%]	5.5	0.3		

[Copyrights to Machine Translated Content]

- (2) Linuxのvmstat, free, topコマンドをバックグラウンドで起動する。
- (3) 各種プロセスの安定を3秒待つ。
- (4) プロセスの安定後、ロボットを制御するプログラムを実行して17秒間計測する。
- (5) (1)～(4)の計測手順を各システム5回ずつ行い、応答時間、PC全体のCPU使用率とメモリー使用率、プロセスごとのCPU使用率とメモリー使用率を記録する。
- (6) 計測結果の平均値をシステムごとに集計する。
- (7) システム間の差異を比較調査する。

3.1.2 評価結果

表2に計測した応答時間 (Unityからの目標角度によってサーボモータを制御してから、エンコーダ値をUnityにフィードバックするまでの通信時間)の結果を示す。また、フィードバックされたエンコーダ値と目標角度との誤差も示す。

時間経過によるPC全体のCPU使用率の変化を図9に、PC全体のメモリー使用率の変化を図10に示す。

表3、表4に実験15秒時点でのプロセスごとのCPU使用率とメモリー使用率の結果を示す。具体的なプロセスごとの使用率を示し、ROS2のプロセス、Unityのプロセス、ROS2-Unity間の接続プロセスに項目分けする。CPU使用率は6コアのため最大600%、メモリー使用率は16GB中の使用%である。

(i) 応答時間の検証

表2の応答時間の結果がシステム1、システム2、システム3の順に応答時間が小さくなり、二群検定の多重比較の結果においてすべてのシステム間で有意差が認められた ($p < 0.05$)。この結果から、応答時間が重要な状況ではシステム3が適していると言える。また、すべてのシステムにおいてジョイント1と2の応答時間に有意差は見られなかった ($p < 0.05$)。これは、2軸のロボットアームを使用したことが原因として挙げられる。関節数が増えた際の応答時間への影響については、より詳細な調査が必要になる。

三つのシステムで約50～80 msの応答時間を示しているが、釣り動作における操作要件を満たしているかを調査する必要がある。3.2節と3.3節の実験で評価する。

(i') 角度誤差の考察

表2の角度誤差の結果が、すべて 0.71° から 0.78° の間の数値を示し、二群検定の多重比較のすべての結果において有意差が認められなかった ($p < 0.05$) ことを考慮するとシステム間の統計的な有意差があるとは言えない。

(ii) PC全体のCPU使用率

図9より、Unityの開始後に注目すると、システム1、システム2、システム3の順にCPU使用率が大きいことがわかる。文献⁸⁾によると、ROS2の通信をリアルタイムアプリケーションに最適化するためには、CPU使用量の制限が重要であることが示されている。本実験においても妥当な

表2 応答時間と角度誤差の結果

システム	joint 1		joint 2	
	応答時間(ms)	角度誤差(°)	応答時間(ms)	角度誤差(°)
システム1	75.8	0.75	77.5	0.72
システム2	64.5	0.75	64.7	0.71
システム3	50.3	0.77	49.7	0.74

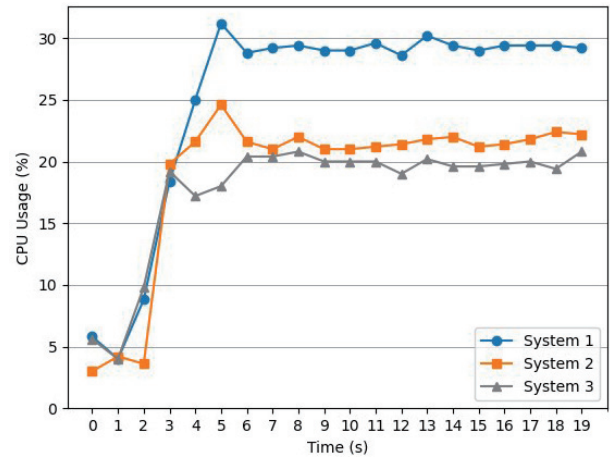


図9 PC全体のCPU使用率

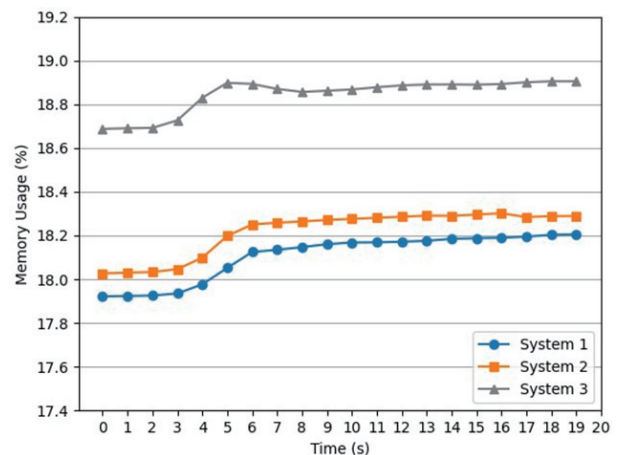


図10 PC全体のメモリー使用率

表3 プロセスごとのCPU使用率

システム名	Unityのプロセス	ROS2のプロセス		ROS2とUnityの接続	
		python3		default_server_endpoint	
システム1	プロセス名	Unity	python3	default_server_endpoint	
	使用率[%]	115.7	80.9	6.5	
	使用率の合計[%]	115.7	80.9	6.5	
システム2	プロセス名	Unity	ros2_control_node	robot_state_publisher	default_server_endpoint
	使用率[%]	116.7	3.8	1.0	8.3
	使用率の合計[%]	116.7	4.8		8.3
システム3	プロセス名	Unity	ros2_control_node	robot_state_publisher	
	使用率[%]	115.0	5.4	1.0	
	使用率の合計[%]	115.0	6.4		

表4 プロセスごとのメモリー使用率

システム名	Unityのプロセス	ROS2のプロセス		ROS2とUnityの接続	
		python3		default_server_endpoint	
システム1	プロセス名	Unity	python3	default_server_endpoint	
	使用率[%]	4.7	0.2	0.4	
	使用率の合計[%]	4.7	0.2	0.4	
システム2	プロセス名	Unity	ros2_control_node	robot_state_publisher	default_server_endpoint
	使用率[%]	4.7	0.2	0.2	0.4
	使用率の合計[%]	4.7	0.4		0.4
システム3	プロセス名	Unity	ros2_control_node	robot_state_publisher	
	使用率[%]	5.5	0.2	0.1	
	使用率の合計[%]	5.5	0.3		

It can be said that results were obtained.

In addition, the time transition of the graph shows that when Unity is started, a large load is applied to all systems.

In particular, it can be seen that in Systems 1 and 2, the memory usage increased rapidly, peaked once, and then stabilized.

(iii) Memory usage rate of the entire PC

Figure 10 shows that the memory usage rate throughout the entire measurement period was greatest in System 1, followed by System 2 and then System 3.

Since the difference is the same from before Unity starts the ROS2 node to the end, the amount of memory used after Unity starts is about the same on all systems, suggesting that there may be a difference in the amount of memory used before Unity starts.

(iv) CPU usage by process According to

literature 8), in robot simulation using ROS2 and Unity, Unity processing accounts for a large proportion of CPU usage, and evaluation by process also showed that Unity and ROS TCP Endpoint consumed a large portion of the computational resources. Similar results were obtained from the results of this experiment, so this is a reasonable result. (v) Memory usage by process As shown in Table 4, although there are differences in the

processes executed, as with (iii), the amount of memory used after Unity starts up is about the same on all systems, and does not have a significant impact on memory usage.

It is clear that

3.2 Evaluation of performance during remote

control 3.2.1 Method for comparing operation time during

remote control In the performance evaluation, a control system is applied to a remote-controlled fishing robot, and its performance is evaluated by measuring the time required for a series of fishing actions. Specifically, the robot is controlled by each of the three systems, and the responsiveness during actual operation is investigated. Figure 11 shows the operator during the experiment. The fishing robot and web camera were installed 5 m away from the control PC, and a situation was set up in which the operator could not see the robot directly. The timing of the measurements

is shown as T1 to T5 in Figure 12. The numbers in [] indicate the stage at which the measurement function is installed and the measurement method. T1 to T5

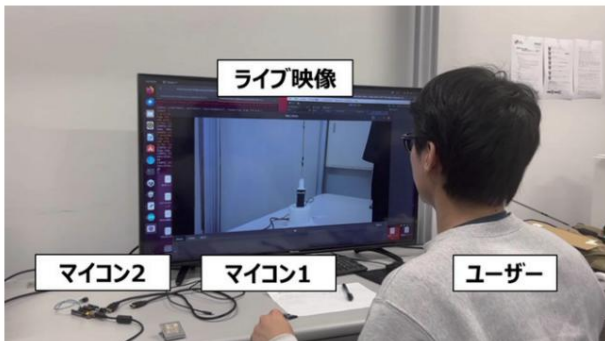


Figure 11. Operator's appearance

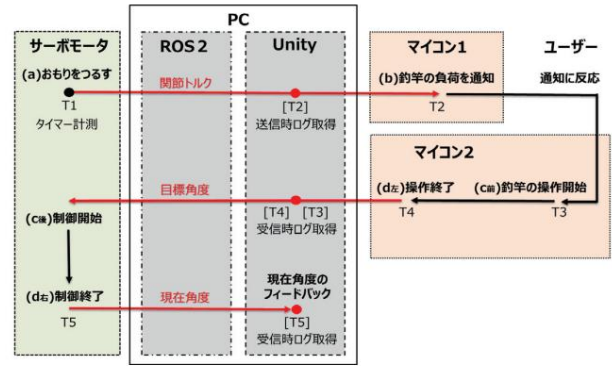


Figure 12. Flow of a series of fishing actions

The measurement timing is shown below.

T1: The measurer hangs a weight on the tip of the robot's pole (Figure a)

T2: Transmits load information to the microcontroller and notifies the user (see Figure 1).
11. Microcomputer 1)

T3: The user reacts and moves the controller (the command value sent from the controller to the robot is 10 degrees or more).
(Fig. c)

T4: The user lifts the controller all the way up (the command value sent from the controller to the robot is 70° or more) (Figure d).

T5: The current angle (encoder value) of the servo motor controlled based on the command value is fed back via ROS2 and Unity (the current angle of the servo motor is used to confirm that the rod angle is 70 degrees or more).

A weight (40 g) was attached to the end of the pole. The person measuring the robot prepared the weight by lifting it up, and measured the time when the weight was released. This applied a load to the robot's joints.

Before starting the operation, the rod angle is kept below 10 degrees, and after confirming the load notification, the fishing motion is performed. The reaction time is measured when the rod angle exceeds 10 degrees, and the operation end time is measured when the angle exceeds 70 degrees. Measurements are obtained from Unity's time-stamped logs.

In addition, when evaluating the statistical significance of the system's performance, we focus only on the intervals that vary due to differences in the systems. Therefore, in this paper, we focus on the measurement results of the time it takes for the operator's movement to be reflected in the movement of the actual machine (T4-T5). increase.

3.2.2 Evaluation results

Table 5 shows the measurement times. A one-way analysis of variance showed a significant difference ($p < 0.05$), suggesting that the average value of at least one system is statistically significantly different from the other systems.

In addition, the results of the t-test for the cases without a corresponding relationship were compared.

Table 5 Time required for operator's movements to be reflected in the machine's operation

区間		計測範囲	システム1	システム2	システム3
4	操作者の動きが実機ロボットに反映される時間 [ms]	T4-T5	323.8	121.2	91.0

[Copyrights to Machine Translated Content]

結果が得られたと言える。

また、グラフの時間推移から、Unityを起動した際にはすべてのシステムで共通して負荷が大きくなるのがわかる。特に、システム1と2においては、急激に増加し、一度ピークに達してから安定した状態に移行したことが読み取れる。

(iii) PC全体のメモリー使用率

図10より、計測時間全体を通してシステム1、システム2、システム3の順にメモリー使用率が大きいことがわかる。UnityがROS2ノードを起動する以前から最後まで同等の差であるため、Unityの起動後に用いるメモリー量はすべてのシステムで同等程度であり、Unity起動以前に用いるメモリー量に差がある可能性を示唆している。

(iv) プロセスごとのCPU使用率

文献⁸⁾によると、ROS2とUnityによるロボットシミュレーションにおいて、CPU使用率はUnityの処理が大きな割合を占め、プロセスごとの評価においても、UnityおよびROS TCP Endpointが計算資源の多くを消費したことを示した。本実験の結果からも同様の結果が得られているため、妥当な結果と言える。

(v) プロセスごとのメモリー使用率

表4より実行されるプロセスの違いはあるものの、(iii)同様、Unityの起動後に用いるメモリー量はすべてのシステムで同等程度であり、メモリー使用率に大きく影響しないことがわかる。

3.2 遠隔操作時の動作性能評価

3.2.1 遠隔操作時の動作時間の比較方法

動作性能評価では、遠隔操作型釣りロボットに制御システムを適用し、一連の釣り動作にかかる時間を計測することで、その動作性能を評価する。具体的には、三つのシステムそれぞれでロボットを制御し、実際に操作する際の応答性を調査する。実験中の操作者の様子を図11に示す。釣りロボットとWebカメラは、制御用PCから5m離れた場所に設置し、操作者が直接ロボットを見られない状況を設定した。

計測のタイミングは図12中のT1～T5で示す。[]内の数字は計測機能を備える段階と計測方法を示す。T1～T5

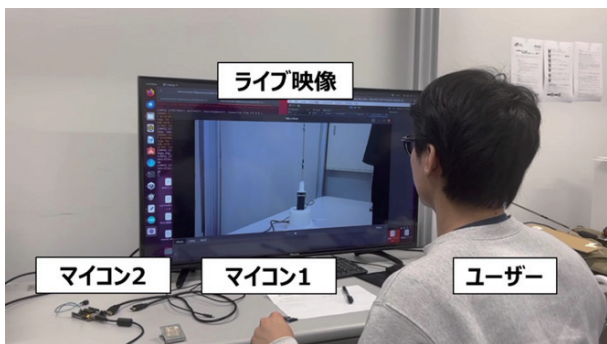


図11 操作者の様子

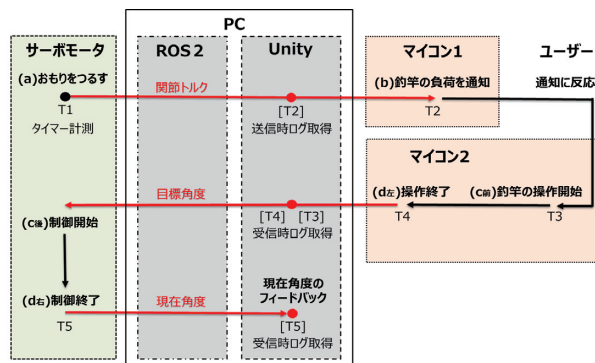


図12 一連の釣り動作の流れ

の計測タイミングを以下に示す。

- T1：計測者がロボットの竿先におもりをつるす (図a)
- T2：マイコンに負荷情報を通信し、ユーザに通知 (図11, マイコン1)
- T3：ユーザが反応してコントローラを動かす (コントローラからロボットに送信する指令値が 10° 以上になる) (図c)
- T4：ユーザがコントローラを上げきる (コントローラからロボットに送信する指令値が 70° 以上になる) (図d)
- T5：指令値に基づいて制御されたサーボモータの現在角度 (エンコーダの値) がROS2とUnityを経由してフィードバックされる (竿の角度が 70° 以上になったことをサーボモータの現在角度で確認)

なお、竿の先にはおもり (40 g) を取り付けた。ロボット側の計測者がおもりを持ち上げた状態で準備し、おもりを離れたタイミングの時間を計測する。これにより、ロボットの関節に負荷がかかる。

操作開始前は竿の角度を 10° 以下で待機し、負荷の通知を確認後、釣り上げ動作を行う。竿の角度が 10° を超えた時間を反応したタイミング、角度が 70° より大きくなった時間を操作終了時間として計測する。計測はUnityのタイムスタンプ付きログにより取得する。

なお、システムの動作性能について統計的な有意差を評価する際は、システムの違いに起因して変動する区間のみ注目する。そのため、本稿では操作者の動きが実機の動作に反映されるのにかかる時間 (T4-T5) の計測結果を取り上げる。

3.2.2 評価結果

表5に計測時間を示す。一元配置の分散分析の結果、有意差が認められ ($p < 0.05$)、少なくとも一つのシステムの平均値が他のシステムと統計的に有意に異なることが示唆された。また、対応関係がない場合のT検定による条件間

表5 操作者の動きが実機の動作に反映される時間

区間	計測範囲	システム1	システム2	システム3	
4	操作者の動きが実機ロボットに反映される時間 [ms]	T4-T5	323.8	121.2	91.0

Multiple comparisons showed that there were significant differences in all combinations ($p < 0.05$). These results show that Systems 2 and 3 showed high overall performance, with System 3 being optimal in situations where responsiveness is most important.

According to reference 9), a study that focused on the critical delay time at which delayed self-body images and visual field images are no longer perceived as the self found that the sense of the self-body and the sense of the self-field of vision decreased as the delay in visual feedback increased, and the critical delay time at which the sense of the self changed from the self to the other or non-self was 300 to 350 ms. It has been shown that there is a possibility of providing a natural feeling of operation. Compared to the results of this paper, the "time required for the operation of the actual device (T4-T5)" is 121 ms for System 2 and 91 ms for System 3, which is less than 300 ms, indicating the possibility of providing a natural feeling of operation. The result of System 1 is 324 ms, which is less than 350 ms, but some users may feel that the feeling of operation needs improvement. This will be investigated in the psychological evaluation in Section 3.3.

3.3 Evaluation of operability during remote

operation 3.3.1 Psychological evaluation experiment when operating a fishing robot

Remote-controlled fishing robots have the potential to provide a new form of entertainment that allows not only fishing enthusiasts but also inexperienced people to share a realistic fishing experience remotely. It is expected that the task will be easier to perform and operability will be improved by controlling the robot as if it were a body, even when in a distant location. Therefore, in addition to the investigation of the time required for fishing motion as described in Section 3.2, we will evaluate how changes in responsiveness due to differences in the system contribute to the psychological effects. This will allow us to consider whether the robot and control system meet the conditions for experiencing realistic fishing motion. In this experiment, we hypothesize that the sense of agency, sense of body ownership, cognitive load, operability, reliability of information provided, and preference during fishing motion will improve by improving the responsiveness of the system, and aim to evaluate these.

In this experiment, 10 participants (9 men, 1 woman, aged 22 to 25) signed the consent form and participated in the experiment after providing informed consent. Of the 10 participants, 4 had fishing experience and 6 had no experience. Figure 13 shows the experimental setup. Participants first experienced operating a real fishing rod to get a feel for real fishing movements. After that, they used the controller.



Figure 13 Robot arm operation during psychological evaluation experiment

The participants were asked to operate the robot arm for about 30 seconds. The participants were instructed to try various movements, such as moving slowly and quickly. In order to compare the operating sensation of the three systems by operating each system once, the participants answered a questionnaire after each trial. In order to eliminate participants' expectations and preconceptions and to collect more objective data, the participants were not informed of the system they were using. In addition, the three systems were presented in different orders, and the number of subjects was equalized to ensure balance. Three of the 10 participants were given the order of systems 1→2→3, three were given the order of systems 2→3→1, and the remaining four were given the order of systems 3→1→2. In addition, in order to avoid the effect of video delay in this experiment, the participants operated the robot while directly looking at it. In the psychological evaluation experiment, the participants' physical cognition and psychological evaluation were measured using the following six questionnaire items.

Q1. Sense of agency:

"The movement of the robot arm is caused by my own movement."

Participants responded to the question, "I strongly disagree" and "I strongly agree" on a 7-point scale (1: "I don't agree at all" and 7: "I agree very strongly").

Q2. Body ownership:

The question "The robot arm was a part of my body."

Q3, Cognitive load: "How much psychological load (stress, anxiety, etc.) did you feel when operating the robot arm?" Participants answered on a 7-point scale from 1 to 7 (1: "Very strong", 7: "Not at all").

Q4, Operability: "Was it easy to operate the robot arm?" Participants answered on a 7-point scale from 1 to 7 (1: "Very difficult to operate", 7: "Very easy to operate").

Q5, Reliability of information provided: "Was the feedback and information provided by the robot arm appropriate?" Participants answered on a 7-point scale from 1 to 7 (1: "Insufficient and difficult to understand", 7: "Sufficient and easy to understand").

Q6, Preference

Participants answered the question "How much did you want to use this robot?" on a 7-point scale from 1 to 7 (1: "I didn't want to use it at all" and 7: "I really wanted to use it"). The psychological evaluation results were examined for significant differences between the systems. All data was tested for normality using the Shapiro-Wilk test before analysis. In this experiment, none of the three systems compared in all questions followed a normal distribution.

(Shapiro-Wilk test, $p < 0.05$). Therefore, the Friedman test (significant difference) is suitable for comparison when there are three or more groups and a corresponding relationship.

[Copyrights to Machine Translated Content]

The copyright of the original papers published on this website belongs to the Institute of Image Information and Television Japanese. Unauthorized use of original papers or translated papers is prohibited. Please be sure to cite the original publication when referencing. For details, please refer to the copyright regulations of the Institute of Image Information and Television Engineers.

の多重比較では、すべての組み合わせで有意な差があることを示した ($p < 0.05$)。これらの結果から、システム2と3が全体的に高いパフォーマンスを示し、最も応答性が求められる状況では、システム3が最適となる。

文献⁹⁾によると、遅延した自己身体像や視野映像が自己のものとして感じられなくなる臨界遅れ時間に注目した研究において、視覚的フィードバックの遅延が増加すると自己身体感や自己視野感が低下し、特に300~350msが自己のものから他者や非自己のものへ変化する臨界遅れ時間であることが示されている。本稿の結果と比較すると、「実機の動作にかかる時間 (T4-T5)」がシステム2で121ms、システム3で91msと、300ms以下に抑えられていることから、自然な操作感を提供する可能性を示している。システム1の結果は324msと、350ms以下には抑えられているが、ユーザによっては操作感に改善が必要と感じる可能性がある。これについては3.3節の心理評価にて調査する。

3.3 遠隔操作時の操作性評価

3.3.1 釣りロボット操作時の心理的評価実験

遠隔操作型釣りロボットは、釣り愛好者だけでなく未経験者も遠隔でリアルな釣り体験を共有できる新しいエンタテインメントを提供する可能性がある。遠く離れた場所にいる場合でも、ロボットを身体のように操ることでタスクが行いやすくなり、操作性が向上すると期待される。そこで、3.2節で行った釣り動作にかかる時間の調査に加え、システムの違いによる応答性の変化が心理的な影響にどのように寄与するかを評価する。これにより、ロボットや制御システムがリアルな釣り動作を体験する条件を満たしているかを検討できる。本実験では、釣り動作時の行為主体感、身体所有感、認知負荷、操作性、情報提供への信頼度、選好がシステムの応答性改善によって向上するという仮説を立て、これらを評価することを目的とする。

本実験では10名の参加者(男性9名、女性1名、22歳から25歳)が、インフォームドコンセントのもと、実験同意書に署名を行い、実験に参加した。10名の参加者のうち釣り経験者は4名、非経験者は6名だった。

実験の様子を図13に示す。参加者には最初に本物の釣竿を動かす体験をしてもらうことで、実際の釣り動作に対する感覚をつかんでもらった。その後、コントローラを使用



図13 心理評価実験中のロボットアーム操作の様子

し、ロボットアームを30秒程度操作してもらった。この時参加者は、ゆっくり動かす場合や素早く動かす場合など、さまざまな動きを試すように教示された。三つのシステムの操作感を各システム1試行ずつの操作で比較するため、参加者は1試行終了するごとにアンケートに答えた。ここでは、参加者の期待や先入観を排除し、より客観的なデータを収集するため、参加者には使用中のシステムを明示しなかった。また、三つのシステムを異なる順序で提示し、被験者数を均等にすることでバランスをとった。10名の参加者のうち3名にはシステム1→2→3、3名にはシステム2→3→1、残りの4名にはシステム3→1→2といった順序で心理評価を行った。さらに、本実験では映像遅延の影響を考えないようにするために、実機ロボットを直接見ながら操作を行った。

心理評価実験では、以下の6問のアンケート項目を用いて、参加者の身体認知や心理評価を測定した。

Q1, 行為主体感

「ロボットアームの動きは自分の動きによって引き起こされたものであった」という質問に対し、1から7の7段階で回答した(1:「まったくそう思わない」、7:「非常に強くそう思う」)。

Q2, 身体所有感

「ロボットアームは自分の身体の一部であった」という質問に対し、1から7の7段階で回答した(1:「まったく感じない」、7:「非常に強く感じる」)。

Q3, 認知負荷

「ロボットアーム操作時にどの程度心理的負荷(ストレス、不安など)を感じましたか?」という質問に対し、1から7の7段階で回答した(1:「非常に強く感じる」、7:「まったく感じない」)。

Q4, 操作性

「ロボットアームの操作は容易であった」という質問に対し、1から7の7段階で回答した(1:「非常に操作しにくかった」、7:「非常に操作しやすかった」)。

Q5, 情報提供への信頼度

「ロボットアームが提供するフィードバックや情報は適切でしたか?」という質問に対し、1から7の7段階で回答した(1:「不十分で理解が難しい」、7:「充分で理解しやすい」)。

Q6, 選好

「このロボットをどのくらい使いたかったですか?」という質問に対し、1から7の7段階で回答した(1:「まったく使いたくないと思った」、7:「とても使いたかった」)。

心理評価結果について、各システム間の有意差を検討する。すべてのデータは、解析前にShapiro-Wilk検定を用いて正規性の検定を行った。本実験では全質問で比較する三つのシステムすべてが正規分布に従うものはなかった(Shapiro-Wilk検定, $p < 0.05$)。そのため、比較対象数が3群以上で対応関係がある場合に適したFriedman検定(有意

A multiple comparison between the conditions was then performed.

A Wilcoxon signed rank test (significance level 5%) was performed.

For the resulting box plots, the line inside the box indicates the median, and the upper and lower edges indicate the third and first quartiles. The upper whiskers of the box indicate the maximum value within 1.5 times the interquartile range (IQR = Q3-Q1) from Q3 (Q3 + 1.5xIQR). Similarly, the lower whiskers indicate the minimum value within the range of Q1-1.5xIQR10).

3.3.2 Evaluation Results

Figures 14 to 19 show box plots of the psychological evaluation results obtained from the experiment regarding sense of agency, sense of body ownership, cognitive load, operability, reliability of information provided, and preference. Table 6 shows the results of tests to show whether there were significant differences. Items marked with a circle are based on the Friedman

The p-values of the mean mean variance test (significance level 5%) and the Wilcoxon signed rank test (significance level 5%) were less than 5%, indicating that a significant difference exists.

The psychological evaluation results in Figures 14 to 19 suggest that Systems 2 and 3 are superior to System 1 in all evaluation items. Although the variance of responses for each evaluation item differs for Systems 2 and 3, the median score for all evaluation items was 5 or higher, indicating that they received high evaluations.

Table 6 shows that the psychological evaluation results showed that there were significant differences in all evaluation items in the Friedman test (significance level 5%) between the three groups, Systems 1 to 3 ($p < 0.05$). In addition, multiple comparisons between conditions using the Wilcoxon signed rank test showed significant differences between Systems 1 and 2, and between Systems 1 and 3, in all evaluation items ($p < 0.05$). No significant differences were found between Systems 2 and 3 ($p > 0.05$).

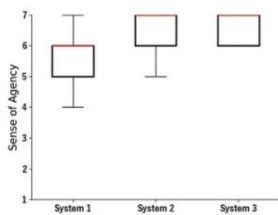


Figure 14 Q1 Sense of agency

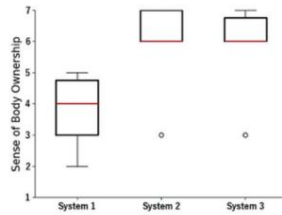


Figure 15 Q2 Body ownership

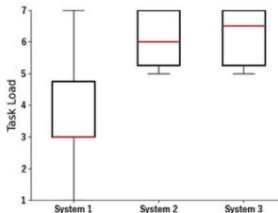


Figure 16 Q3 Cognitive load

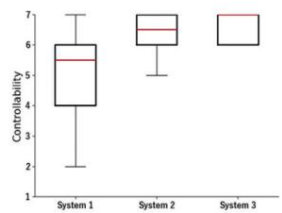


Figure 17 Q4 operability

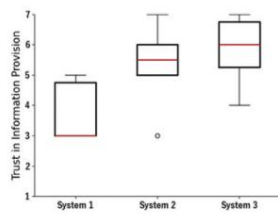


Fig. 18. Q5 reliability

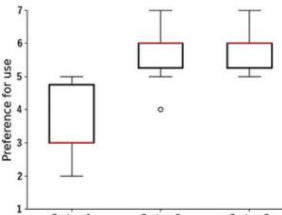


Figure 19 Q6 Preferences

Table 6. Psychological evaluation test results

番号	評価項目	Friedman検定		Wilcoxon符号順位検定			
		System	p	p<0.05	System	p	p<0.05
1	行為主体感 Sense of Agency	(1)-(2)-(3)	0.003	○	(1)-(2)	0.023	○
					(1)-(3)	0.024	○
					(2)-(3)	0.317	
2	身体所有感 Sense of Body Ownership	(1)-(2)-(3)	0.000	○	(1)-(2)	0.002	○
					(1)-(3)	0.002	○
					(2)-(3)	0.317	
3	認知負荷 Task Load	(1)-(2)-(3)	0.000	○	(1)-(2)	0.011	○
					(1)-(3)	0.011	○
					(2)-(3)	0.317	
4	操作性 Controllability	(1)-(2)-(3)	0.006	○	(1)-(2)	0.041	○
					(1)-(3)	0.017	○
					(2)-(3)	0.180	
5	情報提供への信頼度 Trust in Information Provision	(1)-(2)-(3)	0.001	○	(1)-(2)	0.011	○
					(1)-(3)	0.011	○
					(2)-(3)	0.414	
6	嗜好 Preference for use	(1)-(2)-(3)	0.000	○	(1)-(2)	0.002	○
					(1)-(3)	0.002	○
					(2)-(3)	0.655	

[Copyrights to Machine Translated Content]

However, in previous studies, the space occupied by the master robot on the user side was not enough.

While the problem of the large space required for remote control fishing robots is that the controller in this paper requires less space, the control method used in this paper can be operated in a small space. The psychological evaluation results also showed that the requirements for enjoying fishing were met overall, making this an effective control method and further suggesting the practicality of remote-controlled fishing robots.

4. Conclusion

In this paper, we constructed an avatar fishing robot in a digital twin environment by integrating ROS and Unity. In particular, we focused on the control of a real robot using a game engine and its response.

水準5%)を実施し、その後条件間の多重比較としてWilcoxon符号順位和検定(有意水準5%)を行った。

結果の箱ひげ図について、箱の中の線は中央値、上下の縁は第3四分位と第1四分位を示している。箱の上ひげはQ3から四分位範囲(IQR = Q3 - Q1)の1.5倍以内の最大値を示す(Q3 + 1.5 × IQR)。同様に、下ヒゲはQ1 - 1.5 × IQRの範囲内の最小値を示している¹⁰⁾。

3.3.2 評価結果

実験より得られた行為主体感、身体所有感、認知負荷、操作性、情報提供への信頼度、選好に関する心理評価結果の箱ひげ図を図14～図19に示す。また、有意差の有無を示す検定結果を表6に示す。丸印を示した項目はFriedman

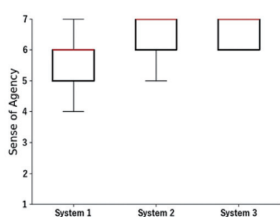


図14 Q1 行為主体感

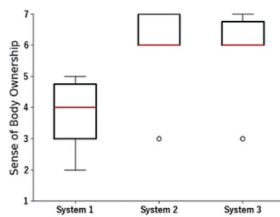


図15 Q2 身体所有感

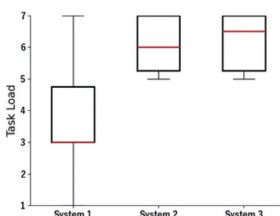


図16 Q3 認知負荷

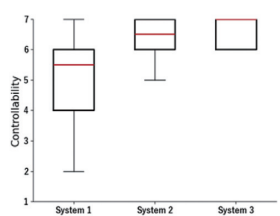


図17 Q4 操作性

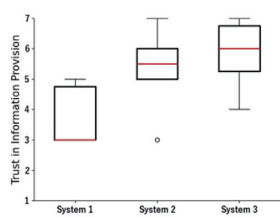


図18 Q5 信頼度

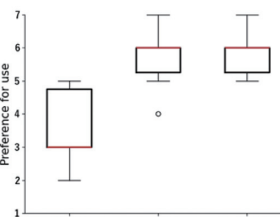


図19 Q6 選好

表6 心理評価の検定結果

番号	評価項目	Friedman検定			Wilcoxon符号順位和検定		
		System	p	p<0.05	System	p	p<0.05
1	行為主体感 Sense of Agency	(1)-(2)-(3)	0.003	○	(1)-(2)	0.023	○
					(1)-(3)	0.024	○
					(2)-(3)	0.317	
2	身体所有感 Sense of Body Ownership	(1)-(2)-(3)	0.000	○	(1)-(2)	0.002	○
					(1)-(3)	0.002	○
					(2)-(3)	0.317	
3	認知負荷 Task Load	(1)-(2)-(3)	0.000	○	(1)-(2)	0.011	○
					(1)-(3)	0.011	○
					(2)-(3)	0.317	
4	操作性 Controllability	(1)-(2)-(3)	0.006	○	(1)-(2)	0.041	○
					(1)-(3)	0.017	○
					(2)-(3)	0.180	
5	情報提供への信頼度 Trust in Information Provision	(1)-(2)-(3)	0.001	○	(1)-(2)	0.011	○
					(1)-(3)	0.011	○
					(2)-(3)	0.414	
6	選好 Preference for use	(1)-(2)-(3)	0.000	○	(1)-(2)	0.002	○
					(1)-(3)	0.002	○
					(2)-(3)	0.655	

検定(有意水準5%)とWilcoxon符号順位和検定(有意水準5%)のp値が5%以下のものであり、有意差が存在することを示す。

図14～図19の心理評価結果より、すべての評価項目においてシステム1に対してシステム2と3が優位と示唆された。システム2とシステム3は、評価項目ごとに回答のばらつきが異なるが、すべての評価項目で中央値が5以上を記録しており、高評価を得ている。

表6より、心理評価の検定結果では、すべての評価項目においてシステム1～3の3群のFriedman検定(有意水準5%)で有意差があることが示された(p < 0.05)。また、Wilcoxon符号順位和検定による条件間の多重比較では、すべての評価項目においてシステム1とシステム2、システム1とシステム3の間に有意差が認められた(p < 0.05)。なお、システム2とシステム3の間では有意差が認められなかった(p > 0.05)。

これら心理評価の結果より、システム2と3は、評価項目ごとに回答のばらつきが異なるが、すべての評価項目で中央値が5以上を記録しており、ロボット操作において高評価を得ていることがわかる。このことから、釣りロボットを動かすための応答条件を満たすシステムと判断できる。

また、検定結果においてすべて項目でシステム1と2、システム1と3の間に有意差が認められたのは、3.2節より「実機の動作にかかる時間(T4-T5)」がシステム1とシステム2で202.6ms、システム1とシステム3で232.8msの差があるため、ユーザの評価に差異が生まれた。システム2と3で類似した評価を示し、システム間で有意差が認められなかったのは、システム2とシステム3で30.2msの差しかないため、操作感に違いを感じ取れなかったことが原因と考えられる。

遠隔操作型釣りロボットの改善点に関する意見では、釣り経験者から魚がかかった際の負荷を感じたいという意見があり、負荷の通知方法に改良を加える必要性が読み取られた。遠隔操作型釣りロボットの先行事例にはマスタスレーブ方式で遠隔地のロボットの負荷が操作者側のロボットにフィードバックされるものがある¹¹⁾。本稿で採用したコントローラによる制御では精度よく負荷をフィードバックできないことが弱点であることを再確認した。しかし、先行研究ではユーザ側のマスタロボットが占領するスペースが広がる課題があることに対し、本稿のコントローラによる制御は省スペースでの運用ができる。心理評価結果からも全体として釣り動作を楽しむ要件を満たしているといえるため、有効な操作方法の一つではあり、遠隔操作型釣りロボットの実用性をより示唆できた。

4. むすび

本稿ではROSとUnityの統合によるデジタルツイン環境でのアバター釣りロボットを構築した。特に、ゲームエンジンを用いた実機ロボットの制御に焦点を当て、その応

The response evaluation experiment in Section 3.1 showed that System 3 was the most responsive, followed by System 2 and System 1, and the remote operation performance evaluation experiment in Section 3.2 showed that Systems 2 and 3 were the most responsive. The results showed that the requirements for operating a robot naturally were met. Although the results of the operability evaluation experiment in Section 3.3 did not show significant differences in all items, the overall evaluation showed that System 3 was suitable for situations where responsiveness was required. In addition, in terms of the comparison of the tools used, the responsiveness of the system was improved when ROS2 for Unity was used for the ROS2-Unity integration method and ros2-control was used for the servo motor control method. However, this result can be said in the experimental environment assuming fishing as a specific example in this paper, and the constructed system is only one example. Furthermore, since this system is a combination of existing assets and the comparison of psychological evaluation is only a comparison of the created system, it is necessary to proceed with the construction of a new system that takes a step forward to practical use. Furthermore, in order to put a remote-controlled fishing robot into practical use, it is necessary to develop a more natural robot control method, provide more realistic feedback, and conduct demonstration experiments in remote locations. As a specific example, we are considering adding a reel mechanism for winding up the fishing line to the controller. We would like to improve the system so that the user can feel the load when a fish is hooked by matching the tension on the fishing robot's line with the force of winding the reel mounted on the controller.

In this research, we have been working on the creation of an avatar robot, using a fishing robot as a subject. In an era where the fusion of the digital and real worlds is progressing, the fusion of the virtual world, including the metaverse, and the real world will also progress.

The avatar, which has both elements, can be used in both the real world and the augmented reality world. It could become a unique means of transportation that allows users to seamlessly move between the three worlds: the augmented reality (AR) world, the virtual world, and the virtual reality (VR) world.¹²⁾ Users will be able to freely control their avatars and act according to their own goals and interests, transcending the boundaries between the real and the digital worlds.

In this research project, we are conducting a system comparison and psychological evaluation based on a basic framework as the first report on some of the elemental technology issues of the basic concept of the "SPRESENSE-connected fishing information platform" announced at the SONY-SICE Sensing Solution Ideathon/Hackathon¹³⁾. Based on the results of this paper, we are working in parallel to develop a mechanism built using new system technology and new implementation methods for specific examples of fishing gear for practical application necessary to realize the concept, by reconciling some of the elemental issues of the basic concept. We will summarize these in the next paper.

Supported by Grant-in-Aid for Scientific Research (17H00302) and SONY-SICE Sensing From the Solution Ideathon/Hackathon to SPRESENSE

We would like to express our sincere gratitude to the editorial board for their support.

We would like to express our sincere gratitude to the reviewers for their insightful comments, detailed guidance, high expectations, and advice that has contributed to our vision for the future.

[literature]

- 1) 白瀬敬一：“デジタルツインの捉え方—シミュレーションからデジタルツインへ— (<特集>デジタルツインでかわるものづくりのこれから)”, 日本機械学誌, 124, pp.6-9 (2021)
- 2) J. Allspaw, G. LeMasurier, H. Yanco: "Comparing Performance Between Different Implementations of ROS for Unity". Virtual Augmented and Mixed Reality for HRI, VAM-HRI (2023)
- 3) "Unity-Technologies/Unity-Robotics-Hub: Central repository for tools, tutorials, resources and documentation for robotics simulation in Unity", <https://github.com/Unity-Technologies/Unity-Robotics-Hub> (2024/1/26閲覧)
- 4) "RobotecAI/ros2-for-unity | High-performance ROS2 solution for Unity3D", <https://github.com/RobotecAI/ros2-for-unity> (2024/1/26閲覧)
- 5) "ROBOTIS e-Manual | DYNAMIXEL SDK Overview", https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/ (2024/1/26閲覧)
- 6) "Real-time control in ROS and ROS 2.0, ROSCon 2015", <https://roscon.ros.org/2015/presentations/RealtimeROS2.pdf> (2024/1/26閲覧)
- 7) "ros-controls/ros2_controllers. Generic robotic controllers to accompany ros2_control", https://github.com/ros-controls/ros2_controllers (2024/1/26閲覧)
- 8) 高瀬英希, 細合晋太郎, 福田竜也, 高田光隆, 久保秋真, 森崇：“hakoniwa-ros2sim：仮想環境を活用したROS2アプリケーションのシミュレーション手法”, 日本ロボット学誌, 41, 4, pp.399-402 (2023)
- 9) 神谷聖耶, 葭田貴子：“遅延した自己身体像や視野映像が自己のものとして感じられなくなる臨界遅れ時間”. Vision, 26, 3, pp.122-126 (2014)
- 10) 萩原隆義, 湯川光, 西村匠生, 棚田亮平, 田中由浩, 南澤孝太：“ロボットアバターを通じた身体融合に基づく身体的協調”, 日本バーチャルリアリティ学論, 27, 4, pp.435-446 (2022)
- 11) “ANA アバター—あらゆる制限を超えて人々を繋ぎ、より良い世界を—”, <https://about.avatarin.com/ana-avatar/> (2024/1/26閲覧)
- 12) 深堀昂：“アバターロボットを活用した新たな移動サービス (<特集> with コロナにおける新たな生活様式を支える技術)”, 日本機械学誌, 125, 1244, pp.30-33 (2022)
- 13) “Sensing Solution アイデアソン・ハッカソン2022：SPRESENSEでつながる釣り情報プラットフォーム”, <https://www.sony-semicon.com/ja/info/2023/2023012701.html> (2024/1/26閲覧)



野村 稔斗 (のむら しゅうと) 2022年, 富山大学工学部卒業. 現在, 同大学院理工学研究科に在学中. アバターロボットによる身体融合を実現するために, 制御システムの応答性に関する研究に従事.



松村 嘉之 (まつむら よしゆき) 1998年, 神戸大学工学部卒業. 2002年, 同大学自然科学研究科博士後期課程修了. 2000年, 日本学術振興会特別研究員(DC1). 2002年同特別研究員(PD), 東京大学客員研究員, 英国パーミンガム大学名誉研究員. 2009年, 信州大学繊維学部准教授, 中国蘇州大学特別客座教授. 2021年より, 富山大学学術研究部工学系教授. EC, RL, EANN, ER, Grid, GPU-CUDA, Cloud, 複雑ネットワーク等の研究に従事. 博士(工学). 正会員.



木下 功士 (きのした こうじ) 1999年, 富山大学工学部機械知能システム工学科卒業. 現在, 同大学工学部に技術専門職員として在職中.

[Copyrights to Machine Translated Content]

The copyright of the original papers published on this website belongs to the Institute of Image Information and Television Japanese. Unauthorized use of original papers or translated papers is prohibited. Please be sure to cite the original publication when referencing. For details, please refer to the copyright regulations of the Institute of Image Information and Television Engineers.

答性や精度の向上に注力した。3.1節の応答性評価実験よりシステム3, システム2, システム1の順に応答性が高く, 3.2節の遠隔時動作性能評価実験よりシステム2と3が釣りロボットを自然に操作するための要件を満たしている結果を得た。3.3節の操作性評価実験ではすべての項目で有意差があると言える結果ではないが, 全体的な評価を鑑みると, システム3が応答性の求められる状況下に適していることが判明した。また, 使用したツールの比較観点では, ROS2-Unity間の連携手法ではROS2 for Unity, サーボモータの制御方法ではros2-controlを採用したほうがシステムの応答性が向上した。しかし, この結果は本稿の具体事例としての釣りを想定した実験環境において言えることであり, 構築したシステムは一例である。そして, このシステムは既存アセットの組み合わせであり, 心理評価の比較も作成したシステムの比較のみであるため, 一歩実使用へ踏み出した新しいシステム構築を進める必要もある。さらに, 遠隔操作型釣りロボットの実用化には, より自然なロボット制御方法の開発, よりリアルなフィードバックの提供, 遠隔地での実証実験などが挙げられる。具体例としては, 釣り糸を巻き取るためのリール機構をコントローラに追加することを考えている。魚がかかったことで釣りロボットの糸にかかる張力と, コントローラに搭載したリールを巻く力を対応させることで, 魚を掛けた際の負荷を感じられるように改良したい。

本研究では釣りロボットを題材とし, アバターロボットの制作に取り組んだ。デジタルとリアルの融合が進む時代では, メタバースなどをはじめとしたバーチャルな世界とリアルな世界の融合も進んでいく。デジタルとリアルの両方の要素を持つアバターは, 現実世界と拡張現実世界 (ARワールド), 仮想世界の三つの世界をシームレスに行き来できる唯一無二の移動手段となりうる¹²⁾。ユーザは自在にアバターを操り, リアルとデジタルの垣根を越えながら, 自らの目的や興味に応じて行動することが可能になるだろう。

本研究課題ではSONY-SICE Sensing Solution アイディアソン・ハッカソン¹³⁾にて発表した「SPRESENSEでつながる釣り情報プラットフォーム」の基本構想の一部の要素技術課題の第一報として, 本稿では基本的なフレームワークに基づくシステム比較と心理評価を行っている。本稿の結果に基づいて, 基本構想のいくつかの要素課題との擦り合わせから, 新しいシステム技術を用いて構築した仕組みと, 構想の具現化に必要な実用化のための釣り具に関する具体事例の新しい実装方法などを並行して進めている。今後, これらについては次稿にて取りまとめていく。

科研費 (17H00302) の助成と SONY-SICE Sensing Solution アイディアソン・ハッカソンから SPRESENSE の支援を受けた。ここに謝意を示す。さらに, 編集委員会の

査読者からの鋭い御指摘, 精緻な御教示, 高い期待と未来への展望に資するアドバイスを数多く頂いたことに, 重ねての感謝の意を表す。

〔文 献〕

- 1) 白瀬敬一: “デジタルツインの捉え方—シミュレーションからデジタルツインへ— (特集) デジタルツインでかわるものづくりのこれから”, 日本機械学誌, 124, pp.6-9 (2021)
- 2) J. Allspaw, G. LeMasurier, H. Yanco: "Comparing Performance Between Different Implementations of ROS for Unity". Virtual Augmented and Mixed Reality for HRI, VAM-HRI (2023)
- 3) "Unity-Technologies/Unity-Robotics-Hub: Central repository for tools, tutorials, resources and documentation for robotics simulation in Unity", <https://github.com/Unity-Technologies/Unity-Robotics-Hub> (2024/1/26閲覧)
- 4) "RobotecAI/ros2-for-unity | High-performance ROS2 solution for Unity3D", <https://github.com/RobotecAI/ros2-for-unity> (2024/1/26閲覧)
- 5) "ROBOTIS e-Manual | DYNAMIXEL SDK Overview", https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/ (2024/1/26閲覧)
- 6) "Real-time control in ROS and ROS 2.0, ROSCon 2015", <https://roscon.ros.org/2015/presentations/RealtimeROS2.pdf> (2024/1/26閲覧)
- 7) "ros-controls/ros2_controllers. Generic robotic controllers to accompany ros2_control", https://github.com/ros-controls/ros2_controllers (2024/1/26閲覧)
- 8) 高瀬英希, 細合晋太郎, 福田竜也, 高田光隆, 久保秋真, 森崇: “hakoniwa-ros2sim: 仮想環境を活用したROS2アプリケーションのシミュレーション手法”, 日本ロボット学誌, 41, 4, pp.399-402 (2023)
- 9) 神谷聖耶, 霞田貴子: “遅延した自己身体像や視野映像が自己のものとして感じられなくなる臨界遅れ時間”, Vision, 26, 3, pp.122-126 (2014)
- 10) 萩原隆義, 湯川光, 西村匠生, 棚田亮平, 田中由浩, 南澤孝太: “ロボットアバターを通じた身体融合に基づく身体的協調”, 日本バーチャルリアリティ学論, 27, 4, pp.435-446 (2022)
- 11) “ANA アバター—あらゆる制限を超えて人々を繋ぎ, より良い世界を—”, <https://about.avatarin.com/ana-avatar/> (2024/1/26閲覧)
- 12) 深堀昂: “アバターロボットを活用した新たな移動サービス (特集) with コロナにおける新たな生活様式を支える技術”, 日本機械学誌, 125, 1244, pp.30-33 (2022)
- 13) “Sensing Solution アイディアソン・ハッカソン2022: SPRESENSEでつながる釣り情報プラットフォーム”, <https://www.sony-semicon.com/ja/info/2023/2023012701.html> (2024/1/26閲覧)



野村 稔斗 2022年, 富山大学工学部卒業。現在, 同大学院理工学研究所に在学中。アバターロボットによる身体融合を実現するために, 制御システムの応答性に関する研究に従事。



松村 嘉之 1998年, 神戸大学工学部卒業。2002年, 同大学自然科学研究科博士後期課程修了。2000年, 日本学術振興会特別研究員(DC1)。2002年同特別研究員(PD), 東京大学客員研究員, 英国パーミンガム大学名誉研究員。2009年, 信州大学繊維学部准教授, 中国蘇州大学特別客座教授。2021年より, 富山大学学術研究部工学系教授。EC, RL, EANN, ER, Grid, GPU-CUDA, Cloud, 複雑ネットワーク等の研究に従事。博士(工学)。正会員。



木下 功士 1999年, 富山大学工学部機械知能システム工学科卒業。現在, 同大学工学部に技術専門職員として在職中。