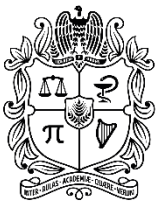


UNIVERSIDAD
NACIONAL
DE COLOMBIA

Diseño de un método de captación de movimientos para el registro, análisis y transmisión de datos a plataformas de rehabilitación robóticas, en un entorno virtual

Carolina Pimentel Gutiérrez

Universidad Nacional de Colombia – Sede Bogotá

Facultad de Ingeniería, Departamento de Ingeniería Mecánica y Mecatrónica

Bogotá D.C., Colombia

2023

Diseño de un método de captación de movimientos para el registro, análisis y transmisión de datos a plataformas de rehabilitación robóticas, en un entorno virtual

Carolina Pimentel Gutiérrez

Tesis de investigación presentado como requisito parcial para optar al título de:

Magíster en Ingeniería - Ingeniería Mecánica

Director (a):

Doctor Luis Miguel Méndez Moreno

Codirector (a):

Doctor Diego Alexander Garzón Alvarado

Línea de Investigación:

Ingeniería de Diseño y Biomecánica

Universidad Nacional de Colombia – Sede Bogotá

Facultad de Ingeniería, Departamento de Ingeniería Mecánica y Mecatrónica

Bogotá D.C., Colombia

2023

(Dedicatoria)

A mis amados padres.

Declaración de obra original

Yo declaro lo siguiente:

He leído el Acuerdo 035 de 2003 del Consejo Académico de la Universidad Nacional. «Reglamento sobre propiedad intelectual» y la Normatividad Nacional relacionada al respeto de los derechos de autor. Esta disertación representa mi trabajo original, excepto donde he reconocido las ideas, las palabras, o materiales de otros autores.

Cuando se han presentado ideas o palabras de otros autores en esta disertación, he realizado su respectivo reconocimiento aplicando correctamente los esquemas de citas y referencias bibliográficas en el estilo requerido.

He obtenido el permiso del autor o editor para incluir cualquier material con derechos de autor (por ejemplo, tablas, figuras, instrumentos de encuesta o grandes porciones de texto).

Por último, he sometido esta disertación a la herramienta de integridad académica, definida por la universidad.

Carolina Pimentel Gutiérrez

Fecha 01/08/2023

Agradecimientos

Son muchas las personas que han contribuido al proceso y conclusión de este trabajo. Para empezar, quiero agradecer al profesor Luis Miguel Méndez Moreno y al profesor Diego Alexander Garzón Alvarado de la Universidad Nacional de Colombia, supervisores de esta tesis, por su paciente orientación, su apoyo personal e institucional, y su entusiasmo para que concluyera esta investigación y pudiera llegar a esta instancia tan anhelada. También, me gustaría agradecer a mis colaboradores, especialmente al ingeniero José Manuel Fajardo, por su contribución en el entendimiento de este campo en el que me he visto inmersa, y al doctor Björn Lütjens quien sentó las bases para el desarrollo de este proyecto.

También, quiero agradecer a Dios, a mis padres, a mi pareja, a mi familia y amigos por estar a mi lado en esta etapa de mi vida, la más retadora de todas, su apoyo moral y aliento han sido fundamentales para seguir adelante en mis propósitos.

Resumen

Diseño de un método de captación de movimientos para el registro, análisis y transmisión de datos a plataformas de rehabilitación robóticas, en un entorno virtual

Esta investigación muestra el desarrollo de un sistema de teleoperación en tiempo real de un robot NAO (visualizado en un entorno virtual) a través de un sistema de captura de movimiento inercial conocido comercialmente como Perception Neuron. Para alcanzar este objetivo, los datos de movimiento capturados se transmiten desde Axis Neuron (software propio de Perception Neuron) instalado en Windows, hasta el Sistema Operativo Robótico (ROS), utilizando el protocolo de comunicación TCP/IP. Se evidenció una latencia poco significativa (<100ms) así como la transmisión de datos continua a una distancia de máx. 10 m entre el hardware y el software del sistema MoCap. En ROS, se reciben los datos capturados, y una vez tratados, se transfieren al modelo virtual del robot NAO visualizado en 3D con la herramienta Rviz de ROS. Para determinar la posición y orientación del efector final en los brazos se utiliza cinemática inversa numérica por el algoritmo de LMS, mientras que para la imitación de los movimientos de la cadena cinemática de la cabeza y de las piernas se realizó una relación *uno-a-uno* entre las articulaciones del operador y del robot. El robot NAO se puede clasificar como un Robot de Asistencia Social (En inglés, Socially Assistive Robot [SAR]), un campo de estudio reciente que se interesa por el uso de esta y otras plataformas robóticas en terapias de rehabilitación. Los resultados son prometedores para avanzar en la implementación y fortalecimiento de este sistema con un propósito terapéutico.

Palabras clave: Captura de movimiento, Perception Neuron; robot de asistencia social, robot humanoide, robot NAO; teleoperación, ROS.

Abstract

Design of a movement capture method for the recording, analysis and transmission of data to robotic rehabilitation platforms, in a virtual environment

This research shows the development of a real-time teleoperation system for a NAO robot (visualized in a virtual environment) through an inertial motion capture system known commercially as Perception Neuron. To achieve this goal, the captured movement data is transmitted from Axis Neuron (Perception Neuron's own software) installed on Windows, to the Robotic Operating System (ROS), using the TCP/IP communication protocol. Insignificant latency (<100ms) was evidenced as well as continuous data transmission at a distance of max. 10 m between the hardware and the software of the MoCap System. In ROS, the captured data is received, and once processed, it is transferred to the virtual model of the NAO robot visualized in 3D with the RViz tool of ROS. To determine the position and orientation of the end effector in the arms, inverse kinematics by LMS algorithm is used, while for the imitation of the movements of the kinematic chain of the head and legs, a one-to-one coupling between the operator's joints was performed. and the robot. The NAO robot can be classified as a Socially Assistive Robot (SAR), a recent field of study that is interested in the use of this and other robotic platforms in rehabilitation therapies. The results are promising to advance in the implementation and strengthening of this system with a therapeutic purpose.

Keywords: Motion capture, Perception Neuron; socially assistive robotics, humanoid robot, NAO robot; teleoperation, ROS.

Contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

Lista de Símbolos y abreviaturas

Abreviaturas

Abreviatura	Término
HRI	Human-Robot Interaction
AR	Assistive Robot
SIR	Socially Interactive Robot
SAR	Socially Assistive Robot
NNcD	Niños y Niñas con Discapacidad
MoCap	Motion Capture
ROS	Robot Operating System
IMU	Inertial Measurement Unit
DOF	Degree of Freedom
LSM	Levenberg-Marquadt with Sugihara's tweak

Introducción

“Los primeros sistemas robóticos que se diseñaron se centraron en aplicaciones industriales que no implicaban interacciones entre usuarios humanos y robots”, dichos sistemas fueron diseñados para obtener un desempeño eficiente y óptimo de una tarea específica [1]. Sin embargo, surgió un nuevo campo investigación que se enfoca en el diseño, implementación y evaluación de sistemas robóticos que interactúan con humanos. Este *“diseño centrado en el ser humano”*, según comenta (Burke et al. 2010) como se citó en [1], hace que la *(*Human-Robot Interaction [HRI]) que en español significa interacción Humano-Robot, sea un campo multidisciplinario único que requiere de experiencia en informática, ingeniería, psicología, ciencias sociales y del comportamiento, antropología, filosofía y ética [1].

Una de las primeras aplicaciones de la HRI, surge en la década de 1990, cuándo se evaluó a través de encuestas y entrevistas la utilidad de un robot manipulador diseñado para asistir a personas con discapacidad física severa, el robot recogía y colocaba objetos usando una interfaz de comando y control con entradas de voz. Y aunque hasta hoy el avance ha sido ampliamente significativo, la robótica, en rehabilitación, siempre ha tenido dos objetivos principales: (1) Diseñar robots que brinden una interacción física segura; y (2) Diseñar robots que participen en interacciones sociales comprensibles y aceptadas por el usuario [1].

Este tipo de avances en el campo de la rehabilitación tiene ciertas ventajas con respecto a la rehabilitación convencional. En todas las ocasiones, la rehabilitación de un paciente supone la repetición constante de uno o varios movimientos [2]. Lo anterior, puede significar inconvenientes para una adecuada terapia de rehabilitación, por ejemplo: que el profesional fuerce los movimientos en el paciente, dependiendo del ejercicio a realizar, pudiendo requerir el acompañamiento de otro profesional de la salud, esto en el ámbito físico; en el ámbito sicológico existe una carga generada por la repetición continuada del ejercicio [3], incluso, si el paciente no nota una mejoría en su condición, la repetición

continuada de los ejercicios puede llevar a su desmotivación [4]. De modo tal que, algunos estudios han evaluado el impacto positivo del uso de sistemas robóticos en la motivación del paciente [5]. Por lo tanto, las plataformas robóticas utilizadas en el contexto de la rehabilitación pueden ayudar a mitigar diferentes problemas. El robot, y específicamente un robot en un ambiente terapéutico, se encarga de incentivar los movimientos en el paciente, desde movimientos simples de una sola articulación hasta los más complejos. De esta manera, sólo se necesita de un profesional de la salud para vigilar la correcta ejecución del ejercicio y constatar la evolución del paciente [6].

Otra ventaja que se puede mencionar, tiene que ver con que el robot puede repetir el ejercicio tanto como sea necesario, sin que esto signifique fatiga para quien dirige la terapia, de hecho, esta faceta repetitiva de la terapia puede ser utilizada como algoritmo de aprendizaje del robot [6].

Los (Assistive Robots [ARs]) que en español significa Robots de Asistencia ayudan, asisten y monitorizan humanos, especialmente personas en condición de discapacidad. Pero, para eso el robot debe tener ciertas características como: la capacidad de percibir su ambiente a partir de sus sensores y actuar en consecuencia, y, la de interactuar con las personas de manera multimodal tomando decisiones de forma autónoma. No obstante, *"esta complejidad exige que se realicen algoritmos computacionales en tiempo real"* [7].

Hoy en día, los (Socially Assistive Robots [SAR]) que puede traducirse como Robots de Asistencia Social, término definido por la intersección entre los ARs y los (Socially Interactive Robots [SIR]) que en español significa Robots Socialmente Interactivos, representan lo mejor de la fusión entre la ingeniería y la medicina de rehabilitación para la intervención de las necesidades educativas, socio-emocionales, cognitivas, sensoriales y motrices en los niños y niñas con discapacidad (NNcD), parafraseando a Shic & Goodwin, 2015, como se citó en [8]. Pues, al igual que los juguetes, objetos por los cuales un niño o niña se siente naturalmente atraído, los SAR pueden diseñarse para que sean atractivos y divertidos de modo que puedan facilitar el descubrimiento y mejorar las oportunidades de juego, aprendizaje y desarrollo [8].

De otro lado, el método más popular para evaluar la tecnología robótica en rehabilitación (con el fin de saber si estos sistemas han sido utilizados de forma efectiva en las

intervenciones de rehabilitación en la población prevista) han sido las entrevistas y cuestionarios, como una forma rápida de recopilar información sobre la adopción y aceptación de la tecnología por parte de los usuarios [1]. Es aquí cuando el estudio de la cinemática del cuerpo humano, se convierte en un área de interés en el plano de la ingeniería biomédica, biomecánica, ciencias del deporte y especialmente en el campo de rehabilitación, con aplicaciones como el estudio del efecto de la edad sobre el cuerpo humano [9]. La obtención de patrones de movimientos de una persona sana a través de estos sistemas permite analizar el movimiento de una persona con algún tipo de patología con afectación motriz, comparaciones que se llevan a cabo por medio de diferentes métodos numéricos [10]. Conocer con precisión los movimientos de un deportista, por ejemplo, también resulta de gran interés, puesto que permite a los entrenadores mejorar el rendimiento y ayuda a evitar lesiones. De hecho, una vez producida la lesión, permite llevar a cabo un mejor seguimiento de la recuperación [11].

Los sistemas más usados para la captura de movimientos en la actualidad son los sistemas ópticos con marcadores; otros como: los sistemas ópticos sin marcadores, los sistemas basados en sensores inerciales y los goniómetros, también presentan diversas aplicaciones; la utilización de alguno de estos sistemas dependerá de las condiciones del estudio, puesto que cada uno presenta ventajas y desventajas dependiendo de su aplicación [11]. En general, estos sistemas de (Motion Capture [MoCap]) que se traduce como Captura de Movimiento se componen de un hardware y un software especializado para el procesamiento de los datos [12], ambos, permiten la captura y análisis de parámetros de movimientos lineales, coordenadas angulares, velocidades y aceleraciones para extremidades y articulaciones [13]. Con todo esto, la utilización en conjunto de los sistemas MoCap y los SAR en la rehabilitación componen un enfoque investigativo de estudio reciente que convoca transversalmente a los profesionales de distintas áreas del conocimiento, con el único fin de apoyar, mejorar y acelerar los procesos de rehabilitación en NNcD.

De hecho, algunos autores como Elbeleidy, M. Terran & T. Williams [14], consideran que *“la criticidad de una terapia es alta teniendo en cuenta que involucra un individuo en recuperación”* e incluso discuten la idea de la favorabilidad de tener un robot totalmente autónomo en este escenario. Lo anterior significa que un ejercicio mal ejecutado puede tener consecuencias graves en rehabilitación, razón por la cual un robot teleoperado (por

un cuidador-terapeuta) puede garantizar la correcta instrucción del ejercicio. Y alternativamente, cuando el robot es teleoperado por la persona que requiere asistencia puede aumentar su aceptación [1].

Esta propuesta de investigación, aunque sólo representa un comienzo para una eventual aplicación como una tecnología de rehabilitación, muestra el desarrollo de un sistema de teleoperación unilateral con un robot NAO (avatar), un robot humanoide clasificado como SAR fabricado por Aldebaran Robotics, por medio de un sistema MoCap comercialmente conocido como Perception Neuron fabricado por Noitom Limited, con el uso del framework (Robot Operating System [ROS]), que puede traducirse como Sistema Operativo Robótico, para implementar la comunicación unidireccional entre los dispositivos en tiempo real. También, se utiliza ROS para visualizar virtualmente la imitación del robot NAO con una secuencia movimientos establecida por el usuario (cuidador-terapeuta o persona asistida).

Finalmente, se considera que los principales aportes de esta investigación consistieron en: (1) la integración de diferentes tecnologías para conseguir el registro del movimiento humano y la imitación de dicha secuencia de movimientos por un robot antropomórfico visualizado en un ambiente virtual, y (2) la creación propia de un archivo ejecutable para la teleoperación de un robot humanoide, en tiempo real o con una secuencia de movimientos pregrabada.

1. Descripción del proyecto

En este apartado se desarrollarán cuatro secciones que permiten reconocer la pertinencia del proyecto de investigación planteado. Estas secciones se dividen en: antecedentes de la investigación, planteamiento del problema, justificación y formulación de objetivos.

1.1 Antecedentes de la investigación

En un recorrido por los estudios más relevantes y significativos que soportan la propuesta de investigación presentada a lo largo de este documento, se encontró que investigadores de todas partes han desarrollado diferentes soluciones para lograr la imitación del movimiento humano en un robot [15] [16] [17] [18] [19] [20] [21] [22]. Estas propuestas, en su mayoría, se enfocan en la imitación de la parte superior del cuerpo humano con un sistema de captura de movimiento óptico sin marcadores, utilizando una solución analítica de cinemática inversa para determinar los movimientos en el robot humanoide NAO. Por supuesto, hay estudios que difieren de estas características. Todos estos se describen a continuación.

G. Emre Cemal, C. YuJung y K. ChangHwan [16] con la investigación titulada *Imitation of Human Upper-Body Motions by Humanoid Robots*, presentaron un sistema offline (que no es en tiempo real) de imitación de movimientos de la parte superior del cuerpo humano, utilizando cinemática inversa y clasificación de tareas que incluye: seguimiento de actuadores finales y seguimiento de trayectorias articulares. Los resultados se validaron con un robot humanoide NAO utilizando dos tipos de sistemas de captura de movimiento basados en: marcadores reflectivos (Motion Analisys, un sistema MoCap profesional que requiere de un laboratorio) y sin marcadores (Microsoft Kinect), ambos sistemas fueron utilizados con el objetivo de registrar los datos del movimiento humano. Luego de grabar el movimiento, se reduce el tamaño de los datos capturados en el humano al tamaño del robot, se calculan los ángulos de las articulaciones e iteran ecuaciones de cinemática

inversa. Finalmente, se utiliza ROS para transferir los valores finales de los ángulos de las articulaciones al robot NAO. La solución presentada se centró en encontrar la posición con mayor precisión del actuador final respecto a otras investigaciones, según mencionaron sus desarrolladores.

Otros autores incluso se han puesto en la tarea de diseñar y construir un robot humanoide impreso en 3D de bajo costo, como se muestra en [19]. El robot bautizado como Leonardo se controla en tiempo real, a través de ROS igual que en [16], utilizando el sistema de captura de movimiento inercial conocido como Perception Neuron, con su aplicación en Windows llamada Axis Neuron. La imitación se enfoca en el movimiento de las extremidades superiores resolviendo la cinemática inversa para la configuración de los brazos del robot, también, utiliza un dispositivo de realidad virtual que proporciona a los usuarios una experiencia inmersiva con el objetivo de que el operador pueda ver lo que el robot está viendo a través de dos cámaras de alta definición colocadas en los ojos del robot *Leonardo*. Los resultados de esta investigación fueron prometedores, pues se logró imitar la orientación exacta del brazo del operador con algunas restricciones. Los autores también pudieron identificar que con la ejecución de varios movimientos continuos el traje de captura de movimiento pierde precisión relativamente más rápido que las gafas de realidad virtual, por lo que se hace necesario calibrar constantemente el hardware del Perception Neuron.

Al igual que en [19] otros investigadores también han optado por utilizar el Perception Neuron para sus desarrollos, dado su bajo costo comparado con otros sistemas inerciales de similares características. B. M. Lütjens con su trabajo de grado titulado *Real-Time Teleoperation of Industrial Robots with the Motion Capture System Perception Neuron* [17], propone un sistema para teleoperar en tiempo real un robot UR10 que imita el movimiento de la extremidad superior, utilizando precisamiente el sistema de captura de movimiento inercial Perception Neuron. Los movimientos capturados por el traje en la extremidad superior se transfieren de Axis Neuron a ROS en otra computadora, igual que en [19] [16], por medio de una aplicación en Windows desarrollada previamente y mejorada en esta investigación. Los resultados presentados muestran un método de control intuitivo y rentable, lo que significa que un usuario no-técnico podría estar a cargo de dar las instrucciones al robot, es decir, que no se requiere de un experto para su operación.

En contraposición a los autores reseñados, I. Rodriguez, A. Astigarraga, E. Jauregi, E. Ruiz y E. Lazcano en [22] considerando que la mayoría de sistemas de teleoperación de un robot humanoide se centraban sólo en realizar la imitación del movimiento de las extremidades superiores, desarrollaron un sistema que permite a los robots humanoides imitar movimientos complejos de todo el cuerpo humano en tiempo real. Su sistema logró equilibrar el centro de masa sobre el polígono de apoyo del robot para evitar caídas, por lo que no hubo restricciones para el movimiento de las extremidades inferiores del robot. Para lograr la imitación segura del robot NAO, que también se utilizó en [16], los investigadores encontraron configuraciones estáticamente estables mediante cinemática inversa para encontrar los ángulos de articulación para cuatro cadenas cinemáticas, dadas las posiciones del actuador final; enseguida, se modificaron los ángulos de las articulaciones para que coincidieran con la posición del centro de masa del instructor humano. El traje captura de movimiento utilizado fue de tipo inercial igual que en [17] [19] , conocido comercialmente como Xsens MVN, un sistema MoCap de alto costo. Los resultados evidenciaron que un robot humanoide es capaz de imitar de forma fiable movimientos complejos, incluso con un apoyo monopodal.

El traje de captura de movimiento Xsens MNV igualmente fue utilizado en el proyecto de investigación titulado *Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning* [21] para teleoperar el robot humanoide NAO también con movimientos complejos del cuerpo humano. El enfoque novedoso de este estudio consiste en entrenar una red neuronal para cada DOF en el robot, es decir, que no se utiliza ningún modelo analítico o matemático como cinemática directa o inversa, por lo cual este desarrollo podría trasladarse a cualquier aplicación que implique el emparejamiento del sistema humano-robot. Se utilizó la técnica denominada optimización por enjambre de partículas o PSO (por sus siglas en inglés, Particle Swarm Optimization) para entrenar cada una de las redes neuronales. Este estudio demostró un nuevo método rápido, eficaz y flexible para la teleoperación de un robot humanoide basado en el uso de técnicas de inteligencia artificial.

Al igual que en [16] [17] [19] existen otras investigaciones que utilizan el entorno de trabajo de ROS para implementar la comunicación entre los dispositivos. Lo desarrollado por J. C. Cerón, M. S. Haque Sunny , B. Brahmi, L. M. Mendez, R. Fareh, H. U. Ahmed y M. H. Rahman en [20] muestra un sistema de teleoperación de un robot de asistencia social a

través de diferentes dispositivos, entre los cuales se encuentra un sensor Kinect para recuperar la información de las articulaciones del esqueleto humano, un dispositivo de realidad virtual Meta Quest y controladores de Nintendo Switch (Joycons), ambos para controlar específicamente las articulaciones del brazo y de la cabeza del robot NAO, robot humanoide también usado en [15] [16] [21]. Las gafas de realidad virtual al igual que en [19] se utilizaron para proporcionar una retroalimentación al operador, pero además se usaron para recuperar la voz del usuario para que el robot NAO pueda hablar. El estudio concluyó con algunos resultados experimentales que demostraron que la teleoperación desarrollada con el robot de asistencia social puede utilizarse en un futuro con fines terapéuticos y educativos, se trata de una investigación reciente con resultados prometedores.

En este sentido, se encuentra en [18] que el robot NAO ha sido utilizado previamente con fines terapéuticos en niños y niñas con discapacidad motriz (Figura **1-1**). La novedosa investigación titulada *A Socially Assistive Robotic Platform for Upper-Limb Rehabilitation, A Longitudinal Study With Pediatric Patients* incorporó lo que los autores llamaron NAOTherapist en los procedimientos terapéuticos de rutina de ocho pacientes pediátricos del Hospital Universitario Virgen del Rocío, ubicado en Sevilla (España), durante cuatro meses. Los autores identificaron que en las terapias de rehabilitación motriz especialmente con pacientes pediátricos puede haber perdida de interés y motivación, dada la falta de variedad en las actividades terapéuticas, por lo cual a través de una actividad de juego se utilizó la plataforma SAR para apoyar las sesiones de rehabilitación con interacción social.

Figura 1-1: Paciente con POPB en terapia con el robot NAO durante el juego de imitación o espejo. Proyecto NAOTherapist.

Fuente: Tomado de [18].

En estas terapias el robot NAO actuó como un entrenador autónomo gracias a la arquitectura de inteligencia artificial PELEA (por sus siglas en inglés, Planning, Execution and Learning Architecture); el robot dirigió la sesión (la cual se adapta a cada paciente) utilizando señales visuales y verbales, mientras recibía una retroalimentación del ejercicio ejecutado por el paciente (con ayuda de un sensor Kinect), para luego brindar un refuerzo positivo, realizar una recompensa o corregir el ejercicio ejecutado. La metodología del estudio implicó comparar en una pre fase la terapia tradicional de los miembros superiores con la terapia basada en SAR en una fase posterior. Aunque el estudio fue de corto alcance, todos los participantes (seis de estos con parálisis cerebral y dos de estos con parálisis obstétrica del plexo braquial [POPB]) informaron sentirse más comprometidos con la terapia mientras jugaban con el robot NAO, la mayoría mostró una leve mejora en sus rangos de movimiento. Por su parte, los miembros del personal clínico y familiares observaron una mejora en la actividad motriz de estos pacientes [18].

Pero, no es la primera vez que se evalúan los efectos de utilizar un SAR en una terapia convencional. En 2007 ya se evidenciaban algunos estudios como en [23], publicado en la *Journal of NeuroEngineering and Rehabilitation*, que aunque reconocía el éxito de la robótica aplicada a la recuperación de pacientes después de un accidente cerebrovascular, encontraron que estos robots sólo se enfocaban en brindar asistencia física. De modo que este estudio se orientó a la robótica de asistencia social centrada en el usuario, sin un contacto físico. Los resultados pudieron demostrar el potencial de los SAR en la recuperación integral del paciente.

Sin ir más lejos, a nivel nacional, el Hospital Militar Central ubicado en Bogotá (Colombia), a través de su Laboratorio de Innovación (INNLAB), con la línea de investigación en robótica, utilizó el robot NAO, el robot humanoide también usado en los estudios [15] [16] [18] [20] [21], como una herramienta coadyuvante para mejorar la calidad de vida de pacientes pediátricos oncológicos en niños entre 8 y 17 años y 11 meses de edad en el periodo comprendido entre febrero de 2021 y julio 2021 [24]. El robot NAO fue programado por los ingenieros con su software nativo Choregraphe, siguiendo las instrucciones dadas por el terapeuta, de modo que las rutinas terapéuticas fueron individualizadas, lo que condujo a tener un equipo de investigadores de 10 especialidades diferentes. Actualmente este robot social hace parte integral de un programa pedagógico para niños y niñas con bandera roja del espectro del autismo o con trastorno específico mixto del desarrollo del

Hospital Militar Central [24]. Sin embargo, hasta el momento sólo se han reseñado algunos resultados a través de encuestas y entrevistas realizadas a los pacientes, familiares y personal de salud tratante [25], que como se ha mencionado antes es el método más popular para evaluar el impacto de la tecnología robótica en rehabilitación [1].

Es importante destacar que los robots de asistencia social utilizados en múltiples escenarios de atención médica pueden tener una apariencia diferente a la humanoide (por ejemplo, el robot NAO), ver Tabla **2-3**. En 2022, un proyecto de investigación de la Universidad del Rosario, ubicada en Bogotá (Colombia), tuvo como resultado el diseño y fabricación de un robot con apariencia animal llamado *Robins* con el objetivo de que fuese un asistente funcional atractivo para apoyar, motivar y alentar los procesos de aprendizaje de la lectura en niños con discapacidad auditiva, aplicando una estrategia educativa apropiada que involucró a un grupo interdisciplinar [26].

Finalmente, y en función de los antecedentes mencionados, se considera que el presente estudio puede enriquecer la literatura existente. El enfoque principal de esta propuesta será la teleoperación en tiempo real de un robot de asistencia social (NAO) por medio de un sistema de captura de movimiento inercial (Perception Neuron), utilizando cinemática inversa numérica y acoplando *uno-a-uno* las articulaciones involucradas en determinadas rutinas de movimiento. Es importante mencionar que esta propuesta implica la integración de sistemas e investigaciones existentes.

1.2 Planteamiento del problema

En general, en el campo de estudio de la HRI, y específicamente en la robótica de rehabilitación se busca que los robots interactúen bidireccionalmente con los usuarios a nivel físico, o social. En cuanto a la HRI social, ésta se enfoca a menudo en poder minimizar la sobrecarga a terapeutas o cuidadores mediante el uso de robots que puedan ayudar a instruir, monitorear, motivar e involucrar a los pacientes en diversas actividades de rehabilitación, con su supervisión. De hecho, las interfaces de HRI que resultan fáciles de usar, son percibidas como positivas, de tal forma que aumentan en general la efectividad, el éxito y la aceptación por parte de los usuarios [1].

En términos ideales, el uso de un sistema robótico en rehabilitación no debe requerir de un operador experto o de una compleja capacitación para su operación [1]. Por lo cual, los avances enfocados en el campo de estudio de la HRI social se centran en el diseño de sistemas total o parcialmente autónomos, dicha autonomía puede influir positivamente en el compromiso del paciente [1]. Incluso, como se mencionaba en [17] es importante que un usuario no-técnico (como un terapeuta) pueda hacerse cargo de la instrucción del robot, para facilitar su operación. En este sentido, en [18] se concluyó, entre otras cosas, que la configuración de un robot de asistencia social (en este caso un robot NAO) utilizado para rehabilitación debería poder realizarse sin ayuda de los ingenieros, quienes además deben tener una capacitación en el manejo del software nativo del robot, o llegar a tener un conocimiento técnico especializado cuando se desea que el robot humanoide ejecute un instrucciones complejas [16]. Al final, lo que se busca es que las sesiones de terapia las configure fácilmente el terapeuta o cuidador.

Por su parte, y teniendo de precedente que una terapia de rehabilitación debe estar centrada en el usuario y en su individualidad, Chan & Nejat en 2012, como se citó en [1] definieron que la adaptación, entrenamiento y aprendizaje debe ocurrir de manera bidireccional entre el ser humano y el robot, es decir, que un sistema robótico para rehabilitación debe poder adaptarse y aprender el comportamiento que necesita para interactuar con una persona y mejorar su participación en una actividad. Esto es, adaptar el nivel de dificultad de la actividad al desempeño del usuario, teniendo en cuenta además que para que un sistema de adaptación sea más preciso es necesario considerar cada articulación de forma individualizada [18]. Para que una terapia sea personalizada, es decir, que responda a las necesidades específicas de la persona que tiene la necesidad de recuperarse, las soluciones actuales en el campo de la robótica social se orientan a tener robots teleoperados o totalmente autónomos [14]. En este sentido, Goodrick & Schultz en 2007 como se citó en [1], definieron que la autonomía de un sistema robótico se refiera a *“la capacidad de un robot para realizar tareas e implementar sus propias acciones independientemente de un ser humano”.* Varios años antes en 1978 Sheridan & Verplank [1] ya habían descrito 10 niveles de autonomía para el diseño general de una solución enfocada en robótica de rehabilitación; el nivel 1 consiste en la posibilidad de que un humano controle el robot completamente a través de un sistema de teleoperación, en este nivel es posible que el robot no proporcione asistencia; en el nivel 10 el robot debe poder actuar de forma autónoma sin requerir ningún aporte o aprobación humana. Sin

embargo, algunos autores [14] difieren de la total autonomía de un robot en un escenario terapéutico, teniendo en cuenta que la criticidad de la terapia es alta por lo que las soluciones orientadas a la teleoperación resultarían más favorables cuando se trata de rehabilitar.

De otro lado, la teleoperación con robots humanoides implica la integración de las habilidades cognitivas, físicas y sociales de los humanos con las capacidades físicas de los robots humanoides [27]. Sin embargo, los robots humanoides imponen desafíos interdisciplinarios y multidisciplinarios para la teleoperación, que van desde la cinemática, la dinámica y el control, hasta la comunicación y psicofisiología humana; de hecho, por su apariencia humana las expectativas son altas, se espera de estos robots, que sean amigables, que interactúen socialmente y que tenga un comportamiento natural [27]. La teleoperación de robots humanoides es un campo de investigación muy activo en el que cada año se proponen nuevas soluciones según se reporta en [27] para el año 2023 en que se escribe esta investigación.

Finalmente, las características físicas de los robots utilizados en rehabilitación juegan a menudo un papel importante y decisivo en la aceptación general de estos sistemas [1]. Pues ya lo identificaban Goetz, Kiesler y Powers en 2003 cuando hallaron que los usuarios adultos prefieren un robot similar a un humano para tareas de asistencia social, por ejemplo la instrucción de ejercicios, y alternativamente, los niños prefieren la interacción con un robot que no posea rasgos faciales, un juguete, según lo encontrado por Robins, Dautenhahn y Dubowski en 2006 [1].

Aunque no es objeto de esta investigación validar lo que aquí se propone con los usuarios finales del sistema, que pueden ser terapeutas, cuidadores o educadores, inclusive pacientes en condición de discapacidad, los autores consideran importante mencionar los datos consolidados en el 2018 por el Departamento Administrativo Nacional de Estadística (DANE), que indican que Colombia tiene 48,2 millones de habitantes, de los cuales 3.065.361 de personas se encuentran en condición de discapacidad, siendo las principales dificultades lo referente a movilidad, ver y oír, que en términos porcentuales corresponden al 36.9%, 18.7%, y, 11.3%, respectivamente [28]. Lo anterior, para dimensionar el tamaño de la población que puede ser impactada de forma positiva por el desarrollo de investigaciones con este enfoque. De otro lado, la Organización Mundial de la Salud (OMS)

ha lanzado la iniciativa Rehabilitación 2030 para llamar la atención sobre la necesidad insatisfecha de nuevos enfoques y recursos en rehabilitación en todo el mundo, y la necesidad también de fortalecer los sistemas de salud [29].

La propuesta que se presenta a lo largo de este documento, apunta a la teleoperación unilateral de un robot de asistencia social sin requerir de un usuario experto para la ejecución de movimientos simples o complejos, con la expectativa de que en un futuro siente las bases para una solución completa en rehabilitación que incluso pueda llegar a ser validada por terapeutas o usuarios interesados en este tipo de herramientas.

1.3 Justificación

El interés por comprender las formas naturales en que un humano puede interactuar con un robot ha dado lugar al desarrollo de los robots humanoides, que debido a su diseño (similar al humano) están destinados a operar en diferentes entornos, especialmente en aquellos compartidos por las personas. Estos robots, en un enfoque de la HRI, se han dotado con la capacidad de capturar, procesar y comprender con cierta precisión las instrucciones humanas, por ejemplo, a través de la teleoperación en tiempo real mediante la detección del movimiento humano, un área de investigación activa [22].

La imitación del movimiento humano es quizás la forma más sencilla de operar un robot humanoide [16]. Sin embargo, existen varios desafíos, entre estos: la recopilación de datos del movimiento, las diferencias físicas, el número de grados de libertad y las diferencias dinámicas. Por lo tanto, este campo de estudio se concibe como un desafío global [16]. El robot humanoide NAO, también considerado un robot de asistencia social, es ampliamente utilizado en diferentes entornos, y tiene usuarios en todo el mundo, de modo que existen numerosos recursos en línea que facilitan el desarrollo de diversas aplicaciones, siendo esta una de las razones por las cuales se optó por utilizar robot (avatar) en esta propuesta. Tiene un software integrado que permite que el robot pueda programarse en diferentes sistemas operativos como ROS, y diferentes lenguajes de programación incluidos C++ y Python. Pese a que puede programarse más fácilmente (en comparación con otros lenguajes) con el software que proporciona el fabricante Choregraphe, esto podría tomar tiempo y requerir de un conocimiento técnico cuando se trata un movimiento complejo [16], por lo que se hace factible la utilización de sensores para rastrear y reproducir el

movimiento humano con un operador que manipule directamente el robot. El requerimiento de un usuario no técnico facilita la introducción de interfaces hombre-máquina intuitivas [17].

No obstante, *"los sistemas de captura de movimiento se deben elegir según su aplicación"* [11]. Cuando los estudios requieren de precisión y de mecanismos de movimiento complejos, se prefiere el uso de un sistema inercial, que entre otras cosas permite mediciones fuera de un ambiente experimental, es de fácil portabilidad y manipulación, y, puede tener exactitud en la información cinemática capturada. Las aplicaciones más frecuentes se encuentran en el campo de la salud, deportes, rehabilitación y ergonomía. Sin embargo, su principal inconveniente es la reducción de la precisión por presencia de campos magnéticos y por errores de integración [11]. El novedoso traje MoCap Perception Neuron permite que las instrucciones al robot se den por imitación del movimiento, y en comparación a sus competidores en el mercado, se trata de un sistema inercial de bajo costo, fácilmente configurable, y funciona sin cámaras [17].

De otro lado, como se menciona en [27] los robots humanoides, como NAO, tienen una versatilidad operativa que los convierte en la plataforma ideal para gran variedad de aplicaciones en teleoperación. Este mismo estudio menciona que cuando se quiere que un sujeto interactúe con un robot teleoperado, el factor de semejanza humana resulta importante, *"dado que aumenta la aceptación, cercanía social y legibilidad de sus intenciones"* [27].

Este trabajo propone un concepto de teleoperación de un robot humanoide utilizando ROS. Como ya se ha mencionado, este propósito se logra mediante el uso de un sistema MoCap inercial para rastrear los movimientos de desplazamiento del usuario no-técnico, con un enfoque basado en la imitación del movimiento humano simple y complejo. Se transmiten los datos desde Perception Neuron en Windows a otro computador con ROS, esto se logra a partir de lo desarrollado previamente por [17], y se transfiere el movimiento a NAO con ayuda de un programa escrito Python. Lo anterior, permite el uso generalizado de los datos capturados por el MoCap a cualquier aplicación de control en ROS, cuyo uso está cada vez más extendido por la academia y la investigación, en parte debido a su integración con varios proyectos de código abierto. Esta tesis apoya un mayor desarrollo de aplicaciones de control de robots humanoides con un usuario no-técnico, cuando se imitan movimientos

humanos simples y complejos. Por ejemplo, aplicaciones en tecnología de rehabilitación, que de hecho constituyó la principal motivación para el desarrollo de esta investigación, pues es evidente la necesidad de diseñar interfaces de operación de fácil manejo y entendimiento para un terapeuta o cuidador, y así mismo, establecer nuevas herramientas terapéuticas con un enfoque diferencial, especialmente para los NNcD, de manera que se pueden proveer otras formas de asistencia, y mejorar en general el proceso de la rehabilitación, de forma que aumente la versatilidad e innovación entre sesiones de terapia [18].

1.4 Objetivos

1.4.1 Objetivo general

Diseñar un método de captura de movimiento para el registro y análisis de datos, y, su posible aplicación en plataformas de rehabilitación robóticas.

1.4.2 Objetivos específicos

- Seleccionar el sistema de captura de movimiento de acuerdo a la identificación y comparación de las tecnologías disponibles actualmente.
- Realizar la recepción y transmisión de datos desde el sistema de captura de movimiento seleccionado, hasta un entorno virtual que simula un robot social.
- Validar la teleoperación en tiempo real de un robot social, en un entorno virtual, utilizando el sistema de captura de movimiento seleccionado.

2. Marco teórico-conceptual

En este apartado se presenta un recorrido por tres ejes teórico-conceptuales que permitirán encuadrar el proyecto de investigación. El primero hace referencia a la clasificación de sistemas especializados que existen en el mercado para la captura, registro y análisis del movimiento humano; el segundo eje hace énfasis en la robótica de rehabilitación como herramienta de intervención a diferentes necesidades físicas, mentales y socio-emocionales, lo cual explica la elección del robot humanoide NAO en esta investigación. Para finalizar, se realizará una breve descripción de ROS, el entorno de trabajo utilizado para el desarrollo de esta propuesta.

2.1 Sistemas de captura de movimiento

"El movimiento es intrínseco a todos los organismos complejos. Les permite adaptarse o reaccionar ante los cambios en un entorno" [30]. De hecho, el movimiento en sí mismo resulta ser una tarea complicada influenciada por: factores internos como el peso, las lesiones, el esqueleto, la musculatura, las emociones, entre otros, y factores externos como la gravedad, los zapatos, el entorno, la ropa y más. Pero, son los factores internos los que han sido el centro de interés de diversas investigaciones, por lo que poder estudiar el movimiento ha implicado avances significativos, pues para estudiarlo primero hay que capturarlo [30].

En principio se opta por el análisis del movimiento en una silueta 2D a partir de una secuencia de imágenes obtenidas de un archivo multimedia (p.ej.: foto o video), pero actualmente la digitalización del movimiento donde el fotograma contiene información sobre las coordenadas articulares del esqueleto 3D ha abierto el camino a diversos dispositivos para la captura del movimiento que presentan los datos MoCap en formatos propios como: ASF/AMC, BVH, C3D y CSV [30].

Los sistema de captura de movimiento que registran datos espacio-temporales y los representa luego digitalmente, según [31] pueden ser clasificados según su tecnología en:

- Sistema MoCap electromecánico: Por ejemplo, Gypsy7. Generalmente estos trajes utilizan estructuras rígidas que restringen la libertad del movimiento unidas mediante potenciómetros que se colocan en las principales articulaciones, los potenciómetros capturan toda la información del grado de rotación de las articulaciones, aunque no tienen en cuenta las rotaciones complejas [31].
- Sistema MoCap electromagnéticos: Por ejemplo, Polhemus. Estos sistemas "*disponen de una colección de sensores electromagnéticos que miden la relación espacial con un transmisor cercano"* [31]. De modo tal, que los sensores colocados en el cuerpo se conectan a una unidad central que recibe la información suministrada por los receptores que a su vez han sido estimulados por un transmisor que genera un campo electromagnético de baja frecuencia [31].
- Sistema MoCap inercial: Por ejemplo, Xsens o Perception Neuron. Están compuestos por sensores inerciales que son colocados en las articulaciones del cuerpo humano para capturar datos precisos de aceleración y orientación del individuo [31].
- Sistema MoCap ópticos: Los cuales pueden ser: (1) con marcadores, por ejemplo, Vicon, se trata de los sistemas más utilizados en los laboratorios de biomecánica, y se pueden clasificar en activos o pasivos; (2) sin marcadores, por ejemplo, Microsoft Kinect, el más popular, un dispositivo periférico que a través de una cámara puede estimar la geometría 3D de la escena capturada. Ha sido adaptado a diversas aplicaciones dado su bajo costo [31].

No obstante, otros autores [30] brindan una clasificación más simple, de la siguiente manera: (1) *sistemas ópticos sin marcadores*, cuya utilización masiva en la industria de los videojuegos provocó que su producción fuese relativamente más económica, de hecho estos sistemas pueden proporcionar suficiente precisión cuando se trata del reconocimiento de acciones a corta distancia; (2) *sistemas ópticos invasivos*, nacen de la necesidad de mejorar la precisión de los dispositivos de grabación con una cámara, estos sistemas agregan marcadores al cuerpo humano para conseguir coordenadas articulares mucho más precisas y estables, pueden ser pasivos (hechos de materiales retrorreflectantes) o activos (emiten su propia luz a través de un LED). Y, (3) *sistemas no-ópticos*, los cuales se basan en sensores inerciales conectados en partes específicas del

cuerpo humano, cuya ventaja principal es poder registrar el movimiento de un sujeto en áreas de gran tamaño, incluso según menciona [30] podría capturar el movimiento de un paracaidista. En general, estos sistemas de captura de movimiento se componen de un hardware y un software especializado para el procesamiento de datos [12]. Estas técnicas de grabación de movimiento del cuerpo humano tienen aplicaciones en estudios para: optimizar el rendimiento deportivo, análisis ortopédicos, verificación de procesos de rehabilitación, animación 3D en películas y videojuegos, entre otras [11]. Sin embargo, estos desarrollos con sus fines variados, tienen fortalezas y debilidades según [11] (Tabla **2-1**).

Tabla 2-1: Características de los sistemas de captura de movimiento.

Tecnología	Fortalezas	Debilidades
Sistemas ópticos con marcadores (Sistemas líderes: Vicon y BTS)	▪ Precisión ▪ Procesamiento datos ▪ Animación 3D ▪ Realidad virtual y aumentada Abarca gran cantidad de aplicaciones ▪ Estudios de alta complejidad	▪ Tiempo de adecuación del espacio y el Sistema ▪ Costo del equipo ▪ Sistema robusto ▪ Oclusión de marcadores ▪ Estudios deportivos ▪ Espacio delimitado
Sistemas ópticos sin marcadores (Sistema líder: Kinect)	▪ Costo ▪ Fácil manipulación ▪ Procesamiento de datos ▪ Precisión en parámetros espaciotemporales ▪ Telerehabilitación	▪ Precisión en parámetros angulares ▪ Estudios de alta complejidad
IMU y sistemas magnéticos	▪ Estudios deportivos ▪ Portabilidad ▪ Fácil manipulación ▪ Realización de estudios ▪ fuera del laboratorio ▪ Costo del equipo	▪ Precisión ▪ Estudios de alta complejidad

Fuente: Tomado de [11].

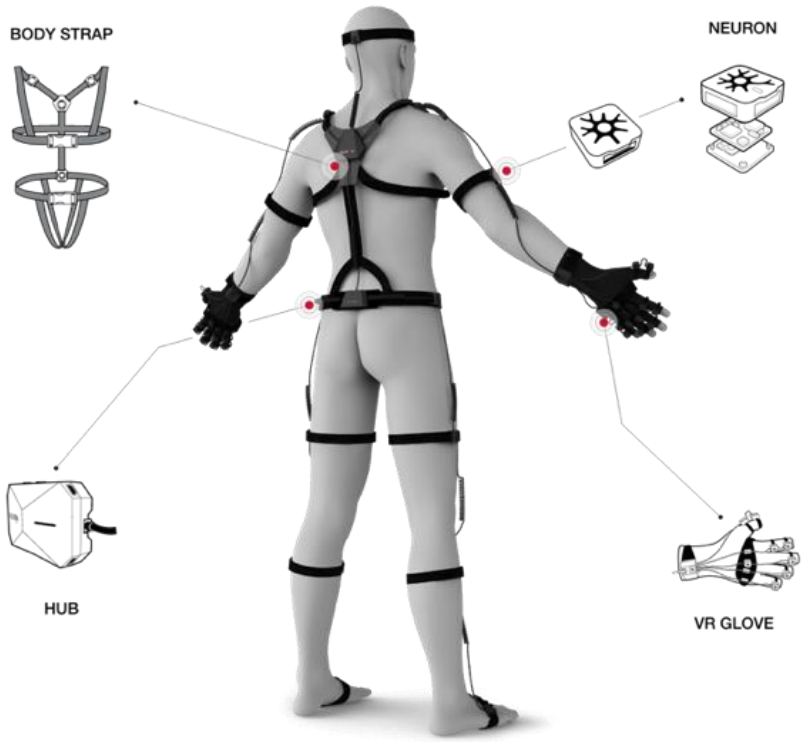
De otro lado y como se mencionó antes, los sistemas de captura de movimiento se deben escoger de acuerdo a su aplicación. En caso de que los estudios que requieran gran precisión e indagar sobre mecanismos de movimiento complejos, en lo posible, debe buscarse la utilización de tecnologías o sistemas ópticos con marcadores, pese a su costo elevado, desarrollos comerciales como Vicon lideran investigaciones en animación 3D, realidad virtual y realidad aumentada [11]. Por otra parte, si el interés de la investigación son las aplicaciones en biomecánica, telerehabilitación y rehabilitación de menor complejidad, se puede recurrir a tecnologías como el Kinect sin marcadores y de fácil acceso. Finalmente, para investigaciones sin restricciones de movimiento, la mejor opción son los sistemas inerciales y magnéticos [11].

2.1.1 Perception Neuron

Perception Neuron (Figura **2-1**) es un sistema de captura de movimiento inercial fabricado y distribuido por Noitom Limited, que puede ser utilizado en diferentes tipos de aplicaciones en los campos de (1) VFX: grabación y reproducción del movimiento; (2) Realidad virtual; (3) Análisis médico o deportivo; (4) Transmisión inalámbrica de datos en tiempo real [32]. La versión 2.0 del sistema posee los siguientes componentes (descritos en inglés, como aparecen en el manual de usuario):

- (32) Perception Neuron Sensors
- Hub
- Anti-MAG Neuron Container
- Hub Power USB Cable
- Hub Signal USB Cable
- Full Body Strap
- Set of wrist/hand/finger straps (one left and one right)
- Set of wrist/hand strap (one left and one right)
- (4) Base Cloth Gloves
- (5) Dummy Neuron
- Dummy Plug
- Check List and Warranty Card
- Universal Strap

Figura 2-1: Sistema de captura de movimiento Perception Neuron 2.0.



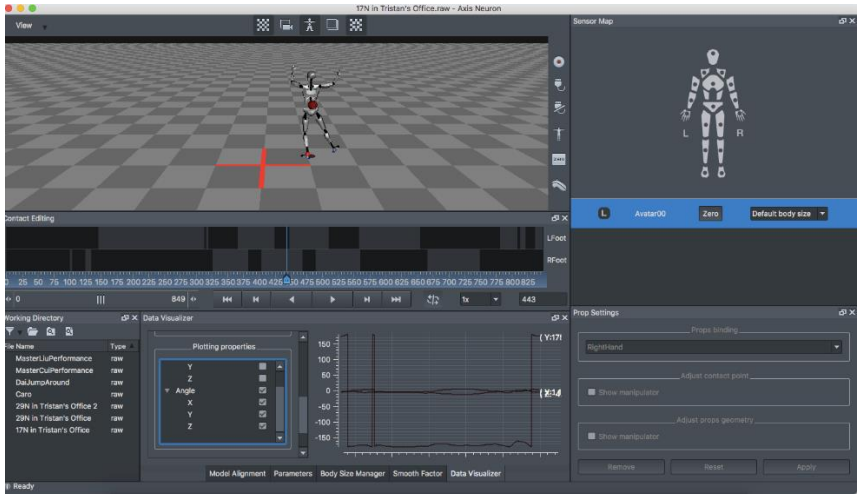
Fuente: Tomado de [32].

Cada sensor es una IMU (por sus siglas en inglés, Inertial Measurement Unit), cada IMU está compuesto por un acelerómetro, un giroscopio y un magnetómetro, con las siguientes especificaciones [33]:

- Tamaño: 12.5mm x 13.1mm x 4.3mm.
- Rango dinámico: 360 grados.
- Rango del acelerómetro: ±16g.
- Rango del giroscopio: ±2000 dps.
- Resolución: 0.02 grados.

Este traje de captura de movimiento, requiere además del software Axis Neuron, propio de Noitom Limited, para recibir y procesar los datos, también es posible realizar transmisiones sincrónicas a otros softwares [33].

Figura 2-2: Interfaz de usuario del software Axis Neuron.



Fuente: Captura de pantalla. Elaboración propia.

Con Axis Neuron se pueden obtener datos del cuerpo completo, pero también del segmento corporal que se quiere analizar, lo que inferirá en la cantidad de sensores o neurons a utilizar así como en la tasa de transmisión en fps, por ejemplo, es posible capturar en modos como: la parte inferior del cuerpo (Requiere sensores en: Cadera, ambas piernas, la parte superior de la pierna y la parte inferior de la pierna); la parte superior del cuerpo (Requiere sensores en: parte superior del brazo y parte inferior del brazo, cadera, columna vertebral); sólo el brazo (Requiere sensores en: brazo superior e inferior), módulo en el cual se pueden registrar movimientos finos que involucran los dedos de las manos (Requiere nueve sensores o *neurons*) (Figura **2-3**) [32] [33].

Figura 2-3: Hardware para la captura de movimiento de las manos y dedos. Perception Neuron 2.0

Fuente: Tomado de [32] [33].

En Axis Neuron se pueden exportar los datos en el formato BVH, cuya estructura es jerárquica. En la Figura **2-4** se muestra un ejemplo de esta estructura que consta de dos partes: el inicio de la sección de encabezado que describe la jerarquía y la pose inicial del esqueleto “HIERIARCHY”; la siguiente línea comienza por “ROOT” y enseguida el nombre del segmento raíz de la jerarquía a definir, en este caso “Hips”. Un archivo con extensión BVH puede contener cualquier número de jerarquías de esqueleto, sin embargo, el número de segmentos está limitado en la práctica. Cada segmento de la jerarquía contiene algunos datos relevantes solo para ese segmento y luego define recursivamente a sus nodos *hijos*. La primera información que se obtiene de un segmento es el desplazamiento que se especifica mediante la palabra clave “OFFSET” seguida del desplazamiento en X, Y, Z del segmento desde su nodo *padre*. La información de desplazamiento también indica la longitud y la dirección utilizadas para dibujar el segmento principal. La línea que sigue al desplazamiento contiene la información del encabezado del canal. Tiene la palabra clave "CHANNELS" seguida de un número que indica la cantidad de canales. La palabra “JOINT” que también se lee como una raíz, o “End Site” que indica que es el efector final, no tiene nodos *hijos.* Finalmente la palabra “MOTION” muestra enseguida una línea que indica el número de fotogramas o muestras de movimiento que hay en el archivo; después encontramos el “Frame time” que es la frecuencia de muestreo [34].

Para terminar, el sistema MoCap Perception Neuron (y su software propio Axis Neuron) disponible en el mercado es una *“solución rentable y fácil de usar para el análisis de movimiento”* [35]. Incluso, fue comparado con Vicon, un sistema MoCap óptico con marcadores, líder en captura de movimiento y procesamiento de datos a nivel mundial; los resultados de este estudio sugirieron que el Perception Neuron es un sistema válido para evaluar la mayoría de rangos de movimientos del cuerpo humano, especialmente en la parte superior del cuerpo con movimientos gruesos, así mismo se menciona que es necesario una cuidadosa calibración y configuración del traje para garantizar un posicionamiento correcto con márgenes de errores aceptables [35].

Figura 2-4: Ejemplo de estructura de datos en formato BVH.

```
HIERARCHY
ROOT Hips
{
	OFFSET  0.00    0.00    0.00
	CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
	JOINT Chest
	{
		OFFSET   0.00    5.21    0.00
		CHANNELS 3 Zrotation Xrotation Yrotation
		JOINT Neck
		{
			OFFSET   0.00    18.65   0.00
			CHANNELS 3 Zrotation Xrotation Yrotation
			JOINT Head
			{
				OFFSET   0.00    5.45    0.00
				CHANNELS 3 Zrotation Xrotation Yrotation
				End Site
				{
					OFFSET   0.00    3.87    0.00
				}
			}
		}
		JOINT LeftCollar
		{
			OFFSET   1.12    16.23   1.87
			CHANNELS 3 Zrotation Xrotation Yrotation
			JOINT LeftUpArm
			{
				OFFSET   5.54    0.00    0.00
				CHANNELS 3 Zrotation Xrotation Yrotation
				JOINT LeftLowArm
				{
					OFFSET   0.00   -11.96   0.00
					CHANNELS 3 Zrotation Xrotation Yrotation
					JOINT LeftHand
					{
						OFFSET   0.00   -9.93    0.00
						CHANNELS 3 Zrotation Xrotation Yrotation
						End Site
						{
							OFFSET   0.00   -7.00    0.00
						}
					}
				}
			}
		}
		JOINT RightCollar
		{
			OFFSET  -1.12    16.23   1.87
			CHANNELS 3 Zrotation Xrotation Yrotation
			JOINT RightUpArm
			{
				OFFSET  -6.07    0.00    0.00
				CHANNELS 3 Zrotation Xrotation Yrotation
				JOINT RightLowArm
				{
					OFFSET   0.00   -11.82   0.00
					CHANNELS 3 Zrotation Xrotation Yrotation
					JOINT RightHand
					{
						OFFSET   0.00   -10.65   0.00
						CHANNELS 3 Zrotation Xrotation Yrotation
						End Site
						{
							OFFSET   0.00   -7.00    0.00
						}
					}
				}
			}
		}
	}
	JOINT LeftUpLeg
	{
		OFFSET   3.91    0.00    0.00
		CHANNELS 3 Zrotation Xrotation Yrotation
		JOINT LeftLowLeg
		{
			OFFSET   0.00   -18.34   0.00
			CHANNELS 3 Zrotation Xrotation Yrotation
			JOINT LeftFoot
			{
				OFFSET   0.00   -17.37   0.00
				CHANNELS 3 Zrotation Xrotation Yrotation
				End Site
				{
					OFFSET   0.00   -3.46    0.00
				}
			}
		}
	}
	JOINT RightUpLeg
	{
		OFFSET  -3.91    0.00    0.00
		CHANNELS 3 Zrotation Xrotation Yrotation
		JOINT RightLowLeg
		{
			OFFSET   0.00   -17.63   0.00
			CHANNELS 3 Zrotation Xrotation Yrotation
			JOINT RightFoot
			{
				OFFSET   0.00   -17.14   0.00
				CHANNELS 3 Zrotation Xrotation Yrotation
				End Site
				{
					OFFSET   0.00   -3.75    0.00
				}
			}
		}
	}
}
MOTION
Frames:    2
Frame Time: 0.033333
 8.03    35.01   88.36  -3.41    14.78  -164.35  13.09   40.30  -24.60   7.88    43.80   0.00   -3.61   -41.45   5.82    10.08   0.00    10.21   97.95  -23.53  -2.14   -101.86 -80.77  -
-61.18   65.18  -1.57    0.69     0.02    15.00   22.78  -5.92    14.93   49.99   6.60    0.00   -1.14    0.00   -16.58  -10.51  -3.11   15.38   52.66  -21.80   0.00   -23.95   0.00
 7.81    35.10   86.47  -3.78    12.94  -166.97  12.64   42.57  -22.34   7.67    43.61   0.00   -4.23   -41.41   4.89    19.10   0.00    4.16    93.12  -9.69   -9.43    132.67 -81.86
-61.56   58.72  -1.63    0.95     0.03    13.16   15.44  -3.56    7.97    59.29   4.97    0.00    1.64    0.00   -17.18  -10.02  -3.08   13.56   53.38  -18.07   0.00   -25.93   0.00
```

Fuente: Tomado de [34].

2.2 Tecnología en rehabilitación

La palabra *rehabilitación* deriva del latín medieval y significa *"vuelta a la buena salud"* [36] [37]. En un principio la *rehabilitación* se definía como un entrenamiento físico para conseguir *"el más alto nivel de capacidad funcional posible",* de acuerdo con lo establecido por la Organización Mundial de la Salud (OMS) en el año de 1969 [36] [37]. Para el año 1981, el Comité de Expertos en Prevención de Incapacidades y Rehabilitación hizo énfasis en que el concepto de *rehabilitación* comprende todas las medidas dirigidas a reducir el impacto producido por las condiciones incapacitantes, dichas medidas deben posibilitar procesos de integración social de las personas afectadas [36] [37]. El concepto ha evolucionado constantemente, y hoy en día, integra diferentes dimensiones de la salud desde una perspectiva biológica, individual y social, entendiendo la discapacidad como una realidad tan compleja, que requiere una visión integradora que facilite su diferenciación [38] [37].

Ahora bien, la *tecnología en rehabilitación* según [39] se concibe como el *"conjunto de medios creados para facilitar el esfuerzo humano"* o incluso puede considerarse como "*capacidad creada".* Este mismo autor [39] hace énfasis en que para tener una mayor conceptualización del término *tecnología en rehabilitación*, es necesario reflexionar sobre las siguientes definiciones:

- Medio*:* La tecnología no es un fin en sí misma;
- Creado: Artificial en tanto es producido por personas;
- Conjunto de medios*:* Puede ser limitado o universal, según el punto de vista del analista; facilita el esfuerzo humano, y, se utiliza para incrementar el desempeño humano o permitir esfuerzos más allá de la capacidad humana.

De esta forma se puede interpretar que la tecnología es el resultado de la capacidad de creación del ser humano para transformar entornos con el fin de facilitar actividades y procesos [39]. Particularmente, la *ingeniería de rehabilitación*, en palabras de [40], *"es la aplicación de la ciencia y la tecnología para disminuir las limitaciones de individuos con discapacidad".*

Si bien existe una variada clasificación en este campo, el único interés de los autores de la propuesta de investigación mostrada a lo largo de este documento, es dar claridad al

lector sobre la *tecnología de asistencia*, que utiliza una serie de aparatos, equipos, estrategias y/o servicios, para incrementar o mejorar las capacidades funcionales de un individuo. Este tipo de tecnología puede ser de bajo costo, hasta alta tecnología, cuyo costo es elevado y de fabricación compleja [39]. En otras palabras, la *tecnología de asistencia* se refiere al dispositivo o servicio que ayuda a un individuo a realizar una actividad funcional, participando en la vida diaria de la persona [8].

La robótica hace parte de las tecnologías en rehabilitación, y puede contribuir a solucionar problemáticas como las asociadas a las terapias de rehabilitación convencional. En la Tabla **2-2** se resumen las ventajas y desventajas de la rehabilitación robótica frente a la tradicional y mixta [41].

Tabla 2-2: Ventajas y desventajas de la rehabilitación robótica frente a la rehabilitación tradicional.

Tipo de Rehabilitación	Ventajas	Desventajas
Rehabilitación robótica (RR)	▪ Mejor calidad de vida ▪ Mayor relación costo-eficacia ▪ Mayor efectividad de las terapias conocidas	▪ Requiere conocimientos de electrónica y programación ▪ Costo ▪ Disponibilidad de hardware ▪ Riesgos mecánicos
Rehabilitación tradicional (RT)	▪ Mayor estimulación ▪ Seguridad ▪ Precisión de la evaluación	▪ Insuficiente para la rehabilitación motora intensiva ▪ Aumento de estrés ▪ No aplica a las necesidades del niño
Rehabilitación mixta	Se debe definir qué parte de la sesión aplicará RR y qué parte RT.	

Fuente: Tomado de [41].

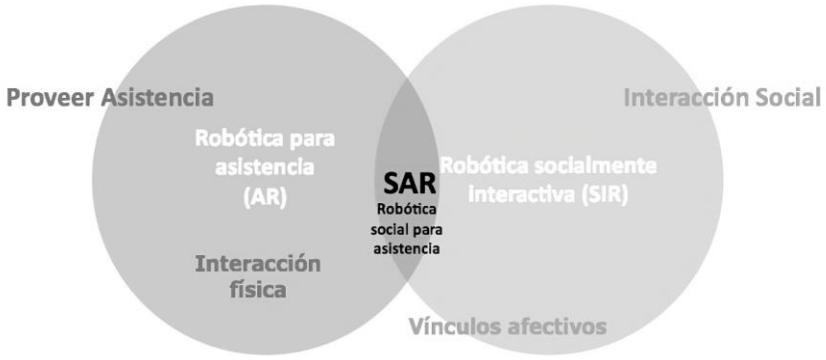
2.2.1 Robótica en rehabilitación

La robótica en rehabilitación es considerada *"una tecnología emergente para el diagnóstico y tratamiento de varios tipos de discapacidad física y mental"* [42]. Un ejemplo clásico son las sillas de ruedas motorizadas inteligentes, las cuales permiten al usuario seleccionar destinos locales, además de contar con un sistema de navegación automático; los dispositivos ortopédicos que buscan reemplazar una función motora deteriorada, son otro ejemplo de tecnología de asistencia [42].

Diferentes modelos se han propuesto para conceptualizar la tecnología de asistencia. Estos modelos se enfocan en: la actividad que la persona con discapacidad quiere realizar, la tecnología de asistencia que requiere para realizar esa actividad, y, el contexto en el cual realiza la actividad [43]. Entre los modelos teóricos tenemos: el modelo HEART (Por sus siglas en inglés, Horizontal European Activities in Rehabilitation Technology) propuesto por en Azevedo y otros en 1994 como se referenció en [43]; el modelo HAAT (por sus siglas en inglés, Human Activity Assistive Technology) propuesto por Cook & Polgar en 2015 según [43]; el modelo de Enders en 1999, y, el modelo de Círculos Dinámicos de Azevedo en 2006, como se citó en [43]. El modelo HAAT es uno de los modelos teóricos más aceptados, en una descripción breve, este modelo posee cuatro componentes básicos: el factor humano, la actividad, la tecnología de asistencia o de ayuda, y el contexto donde se produce la interacción. El componente de tecnología de asistencia es lo que le permite al individuo (que es el factor humano) llevar a cabo una actividad dentro de un contexto determinado [8]. Por lo tanto, el componente de tecnología de asistencia comprende el estudio de las interfaces humano-máquina [43].

En este sentido, se han identificado tres ramas principales de la robótica en rehabilitación, estás son: la *robótica de asistencia* o AR, que está destinada a sustituir o compensar la falta de habilidades motoras y/o sensoriales, definida en 2005 por Feil-Seifer & Mataric, como se citó en [42]; la *robótica socialmente interactiva* o SIR, la cual está involucrada principalmente con el comportamiento humano cuando tiene un compañero robótico que habla y posee gestos, término introducido por Fong, Nourbakhsh & Dautenhahn en 2003, por [42]. Finalmente, la *robótica de asistencia social* o SAR definida recientemente como un área de estudio que representa la intersección entre SIR y AR [42] (Figura **2-5**).

Figura 2-5: Definición de los robots de asistencia social o SAR.



Fuente: Elaboración propia con base en [18] [42].

El objetivo de un SAR es *"crear una interacción cercana y efectiva con un usuario humano con el propósito de brindar asistencia y lograr medir el progreso de la recuperación, rehabilitación, aprendizaje, etc."* [8]. Como dato adicional, los robots que proporcionan asistencia a los usuarios a través de la interacción física, podrían denominarse *robots de asistencia física* o PAR (por sus siglas en inglés, Physical Assistive Robot) [8].

Estas plataformas robóticas buscan en cualquier caso mejorar la motivación y compromiso de los pacientes sin que exista necesariamente un contacto físico, lo que reduce riesgos de seguridad y facilita su integración en la práctica clínica [18]. Incluso, muchos investigadores consideran que estas plataformas representan desafíos ambiciosos que no únicamente se enfocan en el monitoreo, motivación y fomento de actividades terapéuticas [18], sino en ¿Qué hacer para que la integración de los robots en terapias y la aceptación por parte de la mayoría de individuos del grupo demográfico al que está dirigido sea exitosa? [8].

A la pregunta planteada anteriormente se podría sugerir en principio que será indispensable una colaboración entre los campos clínicos y tecnológicos, para lo cual Diehl y otros autores plantearon en 2014 las siguientes consideraciones, como se citó [8]:

- Realizar una validez incremental, es decir, identificar si la presencia de un robot como co-terapeuta mejora los resultados por encima de los diagnósticos que se realizan con un enfoque empírico.
- Realizar una validez predictiva, esto es, conocer cómo las características individuales del robot y del niño o niña afectan los resultados terapéuticos.

- Identificar los roles más efectivos del robot durante la sesión, como co-terapeuta, es decir, trabaja con el terapeuta humano, o, es posible (o deseable) el uso del robot como terapeuta principal.
- Identificar la importancia de la autonomía del robot y los beneficios en los resultados terapéuticos.
- Identificar las formas más efectivas de implementar este enfoque en los casos clínicos reales y entornos como escuelas, hospitales y centros sin fines de lucro que brinden servicios de asistencia y apoyo a personas en condición de discapacidad.

▪ Robots sociales para terapias de rehabilitación

La robótica de asistencia social o SAR es un campo de estudio de la disciplina que investiga la interacción de los robots en entornos humanos o HRI, se enfoca en ayudar a las personas a través de la interacción social en lugar de la interacción física, lo que incluye un amplio espectro de aplicaciones como los robots educativos, los robots asistentes de oficina, entre otros. En otras palabras, se quiere que *"el robot actúe como una persona social"* [42].

Particularmente, se considera que los SAR, pueden ser potencialmente útiles como herramientas de intervención terapéutica en NNcD [42]. Esto porque un SAR busca replicar o modificar pero no reemplazar los beneficios terapéuticos y educativos que se derivan de la relación entre los cuidadores y la persona con discapacidad [8]. Por lo tanto un SAR debe *"Interactuar con su ambiente, exhibir un comportamiento social, y centrar su atención y comunicación en el usuario para ayudarlo o ayudarla a alcanzar su objetivo"* según Mataric y otros autores (2009) como se citó en [8].

En este sentido, se pueden agrupar cuatro grupos de estudio [8] [42]:

- Grupo uno. SAR enfocada a mejorar la función física.
- Grupo dos. SAR enfocado a mejorar la interacción social (HRI y la comunicación).
- Grupo tres. SAR enfocado en mejorar la motivación (compromiso).
- Grupo cuatro. Estudios sobre la usabilidad de SAR en distintos entornos.

En el primer grupo de estudio, *SAR enfocado en mejorar la función física*, generalmente a través de un entorno de juego, se utiliza como una herramienta terapéutica en la imitación

de tareas físicas durante las actividades de entrenamiento, desarrollos comerciales como KineTron, Ursus, NAO y Cosmobot, son un ejemplo de esto. Estos robots que se usan como entrenadores dentro de una actividad motriz [42]. Por lo tanto, el objetivo principal es alentar a los niños y niñas en el entrenamiento motor y aumentar su interés en la terapia, lo que significa disfrutar más de una sesión con una plataforma robótica en comparación de una sesión de una terapia convencional. Mantener el compromiso del paciente, es otro enfoque de gran interés de esta área de estudio [42]. Adicionalmente, se considera que este sistema HRI, debe tener las siguientes características [42]:

- El robot humanoide debe realizar las repeticiones físicas del ejercicio.
- El escenario de la terapia es uno a uno, esto quiere decir, que el robot y el participante, deben estar frente a frente durante la interacción.
- El robot debe guiar la realización del ejercicio y proporcionar retroalimentación en tiempo real.
- La HRI debe medirse en función de la respuesta de los participantes.
- No deberá existir contacto físico entre el participante de la terapia y el robot.

Por otro lado, en el grupo de estudio número dos, un SAR busca provocar comportamientos sociales positivos en la interacción humano-humano, entendiendo que la comunicación es un factor importante para el desarrollo de las habilidades sociales. El robot se convierte entonces, en objeto de atención conjunta, por parte de los niños y niñas, y sus compañeros o cuidadores. El desarrollo en este campo es prometedor, así lo demuestra la utilización de Neptuno, un robot que fue utilizado en 2011 como mediador social para la actividad de aprendizaje, con el cual se buscaba promover la inclusión social [42]. El tercer grupo, *SAR para mejorar la motivación y compromiso*, se centra en la premisa de que: *"cuanto más motivado y comprometido está un individuo en una actividad, mejor será el resultado del aprendizaje"* [42]. PETS, es un robot utilizado en una actividad de juego, que consiste en que los niños y niñas al ejecutar un movimiento puedan controlar remotamente el robot, el uso de este juguete demostró que los NNcD aumentan su compromiso y motivación para hacer la terapia, aunque es necesario añadir variabilidad y novedad, para evitar la exposición repetitiva [42]. Finalmente, el uso de los SAR en distintos entornos debe garantizar ciertos requisitos como: no tener riesgo para los niños, fácil de configurar, robustez, bajo mantenimiento, bajo riesgo de falla tecnológica, entre otros [42].

En este orden de ideas, un SAR (Tabla **2-3**), con base en los autores [42], debe tener las siguientes características:

- El robot tiene que ser atractivo para el niño o niña.
- El robot debe promover la participación activa y el juego libre.
- E robot debe tener variabilidad en medio de la actividad de juego, para promover el compromiso continuo después de sesiones repetitivas.
- El robot tiene que tener un comportamiento predictivo.
- El robot tiene que ser fácil de usar.

Tabla 2-3: Resumen de robots de asistencia social en fase de investigación y desarrollo, y en fase de comercialización para el año 2019.

	Robot	**Sistema de visión**	**Sistema de Audio**	**Sistema de movilidad**	**Nivel de madurez tecnológica o TRL**	***Status* del Desarrollo**
1	**Bobus**	Infrarrojo	Ninguno	Base con ruedas	7	En investigación (Universidad de Sherbrooke)
2	***Child-centered Adaptative Robot for Learning in an Interactive Enviroment* (CHARLIE)**	Monocular	1 altavoz	2-DOFcabeza y brazos	7	En investigación (Universidad de Carolina del Sur)
3	**C-Pac**	Infrarrojo	Altavoz	Base con ruedas	7	En investigación (Universidad de Sherbrooke)
4	**Diskcat**	Ninguno	Altavoz	Base con ruedas	7	En investigación

						(Universidad de Sherbrooke)
5	***Facial Automation for Conveying Emotions* (FACE)**	Monocular	Ninguno	32-DOF movimientos faciales	7	En investigación (Universidad de Pisa)
6	**Humanoid for Open Architecture Platform (HOAP)-2**	Estéreo	Ninguno	25-DOF con piernas humanoides	9	Comercializado (Laboratorios Fujitsu)
7	**Infanoid**	Estereoscópico	Ninguno	24-DOF torso	7	En investigación (Japón)
8	**Interactive Robotic Social Mediators as Companions (IROMEC)**	RGB-D	Micrófonos y Altavoz	Base con ruedas	7	En investigación (Unión Europea)
9	**Jumbo**	Infrarrojo	Altavoz	Base con ruedas	7	En investigación (Universidad de Sherbrooke)
10	**KASPAR**	Ninguno	Altavoz	8-DOF cabeza, 3-DOF brazos (2)	7	En investigación (Universidad de Hertfordshire)
11	**Keepon**	Estéreo	Micrófono	4-DOF	7	En investigación (Universidad Carnegie Mellon)

12	**Kismet**	Estereoscópico	Micrófono y altavoz	15-DOF expresiones faciales	7	En investigación (Instituto Tecnológico de Massachusetts)
13	**Labo-1**	Infrarrojo	Altavoz	4- Base con ruedas	7	En investigación (Universidad de Hertfordshire)
14	**Lego Mindstorms® NTX**	Ninguno	Micrófono	Base con ruedas	6	En investigación
15	**Maestro**	Infrarrojo	Altavoz	Base con ruedas	7	En investigación (Universidad de Sherbrooke)
16	**Milo**	RGB-D	8 micrófonos y altavoz	18-DOF con piernas	9	Comercializado (Robokind)
17	**NAO**	Estéreo	4 micrófonos direccionales y altoparlante	25-DOF con piernas humanoides	9	Comercializado (Aldebran Robotics)
18	**Pekee**	Infrarrojo	Ninguno	3- Base con ruedas	9	Comercializado (Wany Robotics)
19	**QueeBall**	Ninguno	Altavoz	2-DOF esfera	9	Comercializado (Que Innovation)
20	**Leka**	Ninguno	Altavoz	2-DOF esfera	9	Comercializado (Leka)
21	**Robota**	Infrarrojo	2 altavoces	5- DOF (2 piernas, 2 brazos, 1 cabeza)	9	En investigación (Escuela Politécnica Federal de Lausana)

22	**Tito**	Monocular	Ninguno	Base con ruedas	7	En investigación (Universidad de Sherbrooke)
23	**Triadic Relationship EVOking Robot (TREVOR)**	Ninguno	Micrófono y altavoz	3-DOF brazos	7	En investigación (Universidad Brigham Young)
24	**Troy**	Ninguno	Altavoz	4- DOF brazos, 2-DOF cabeza	7	En investigación (Universidad Brigham Young)
25	**Buddy**	Monocular	Micrófono y altavoz	Base con ruedas	8	Comercializado (Blue Frog Robotics)

Fuente: Tomado de [8].

- **El robot NAO**

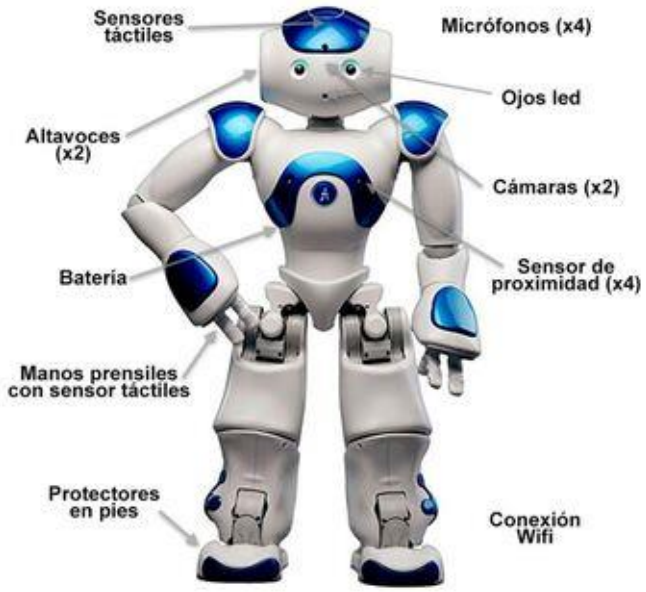
Es un robot humanoide social programable (Figura **2-6**) que fue desarrollado por Aldebaran, antes conocida como SoftBank Robotics Europe, ahora socio del grupo alemán United Robotics. Es utilizado en ingeniería de rehabilitación como un SAR [18]. El robot mide 58 cm de alto y pesa 5,5 kg. Puede ser alimentado con un cable o por baterías de iones de litio. Tiene acceso a la red vía Ethernet o por conexión Wi-Fi [44]. Además, según [44] cuenta con:

- 25-DOF en total: 2-DOF en la cabeza, 5-DOF en cada brazo, 1-DOF en la pelvis, 5-DOF en cada pierna y 2-DOF en cada mano; las juntas son accionadas por motores MAXON DC.
- Dos altavoces de un diámetro de 36 mm dispuestos en sus oídos, cuatro micrófonos colocados alrededor de su cabeza con un rango de frecuencia de 300 Hz a 8 kHz.
- 32 sensores de efecto Hall en las articulaciones, 2 girómetros de un eje, 1 acelerómetro de tres ejes y 2 parachoques ubicados en la punta de cada pie. También tiene 2 sensores de ultrasonido en el pecho y una serie de sensores capacitivos en la cabeza.

Puede capturar fotogramas a 30 fps con dos cámaras VGA CMOS. Tiene un campo de visión de 58º y un rango de enfoque desde 30 cm.

- Iluminación LED en los ojos, los oídos, el pecho y los pies.

Figura 2-6: El robot NAO.



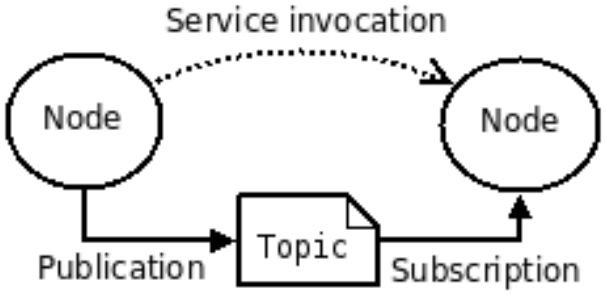
Fuente: Tomado de [45].

El robot NAO ofrece un conjunto completo de sensores, cámaras y micrófonos, que mejoran su capacidad autónoma, y mecanismos interactivos que facilitan la interacción social con personas [18].

2.3 Entorno ROS, una breve descripción

ROS es una meta sistema operativo de código abierto para sistemas robóticos que proporciona una extensa serie de servicios y librerías que simplifican la creación de aplicaciones complejas que impliquen obtener, construir, escribir y correr un código a través y mediante varias computadoras para aplicaciones complejas en robótica [46]. Comenzó como un pequeño proyecto en Stanford desarrollado por Erick Berger y Keenan Wyrobek mientras realizaban sus estudios de doctorado [47]. Es similar a otras estructuras como Player, YARP, Orocos, CARMEN, Orca, MOOS y Microsoft Robotics Studio. Actualmente, se ejecuta en plataformas basadas en Unix y en W10 mediante virtualización [46].

Figura 2-7: Estructura de ROS.



Fuente: Tomado de [48].

A continuación se definen textualmente algunos conceptos importantes, con base en [48]:

- *Packages:* Son la unidad principal, pueden contener un conjunto de datos, una biblioteca dependiente de ROS, archivos de configuración o procesos (nodos).
- *Metapackages:* Son paquetes especializados que sirven para representar un grupo de paquetes relacionados.
- *Nodes:* Se encargan de realizar cálculos; un sistema de control de robot generalmente comprende muchos nodos. En un robot, por ejemplo, un nodo realiza localización, un nodo controla los motores de las ruedas, un nodo realiza planificación de rutas, un nodo proporciona vista gráfica del sistema, etc. (Figura **2-7**).
- *Master:* El ROS Master actúa como un servicio de nombres en el gráfico de cálculo de ROS. Sin el Master, los nodos no podrías encontrarse, intercambiar mensajes o llamar (invocar) servicios.
- *Messages (.msg):* Los nodos se comunican entre sí pasando mensajes. Un mensaje es una estructura de datos, estos pueden incluir matrices y estructuras anidadas arbitrariamente.
- *Topics:* Los mensajes se enrutan a través de un sistema de transporte con semántica *publish/suscribe*. Un nodo envía un mensaje publicándolo en un tópico determinado. El tópico es un nombre que se utiliza para identificar el contenido del mensaje. Un nodo que esté interesado en cierto tipo de datos se suscribirá al tópico apropiado, (Figura **2-7**).
- *Services (.srv):* El modelo de comunicación *publish/suscribe* es muy flexible, pero su transporte unidireccional no es apropiado para las interacciones *request/reply*, que a

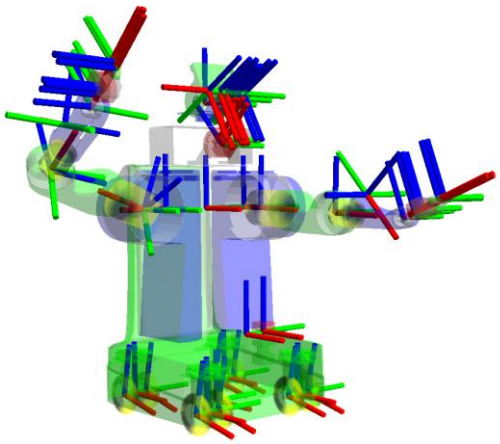
menudo se requiere en un sistema distribuido. La interacción *request/reply* se realiza a través de servicios, que están definidos por estructuras de mensajes, una para la solicitud (request) y otra para la respuesta (replay) (Figura **2-7**).

- *Bags:* Son un formato para guardar y reproducir datos, por ejemplo, datos de sensores que pueden ser difíciles de recopilar.

2.3.1 Librería *tf*

La librería *tf* es considerada la solución al problema cinemático de la robótica dentro de ROS *tf* [47] [49]. Es un paquete que permite al desarrollador realizar un seguimiento de múltiples marcos de coordenadas a lo largo del tiempo. Este paquete mantiene la relación entre dos marcos de coordenada en una estructura de árbol y permite al usuario transformar puntos, vectores, etc. [50] (Figura **2-8**).

Figura 2-8: Datos del marco de referencia de un robot, utilizando la librería *tf*-ROS.

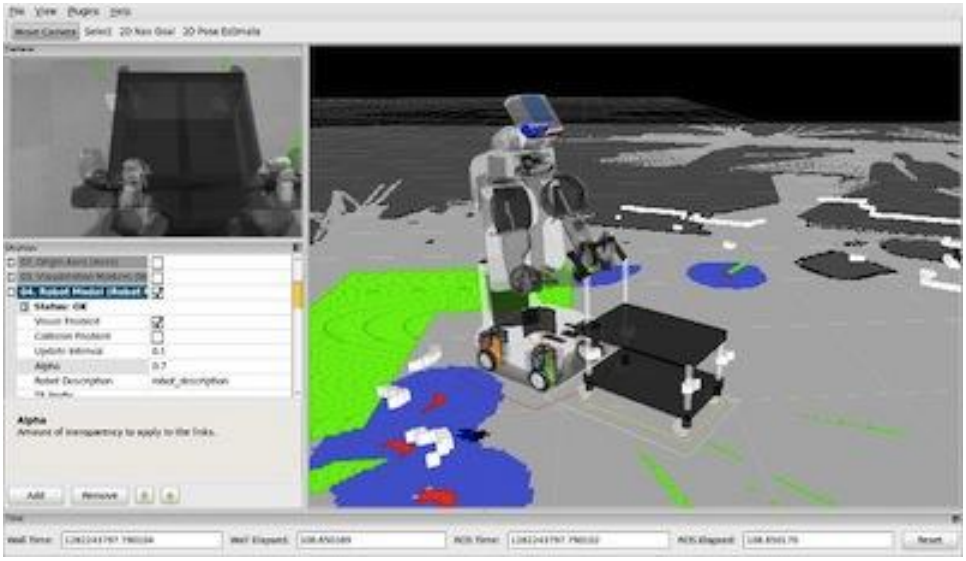


Fuente: Tomado de [50].

2.3.2 Rviz

Cuando se quiere desarrollar una aplicación en la que se quiera obtener la visualización de un robot o un entorno, la herramienta Rviz de ROS resulta muy importante (Figura **2-9**), se trata de un visualizador que puede proveer un modelo tridimensional de un robot, el cual debe estar descrito en formato URDF, este es un archivo en XML en donde se dan todas las especificaciones como tamaño, ejes de rotación, número de articulaciones, entre otras [47].

Figura 2-9: Herramienta de visualización de Rviz-ROS.

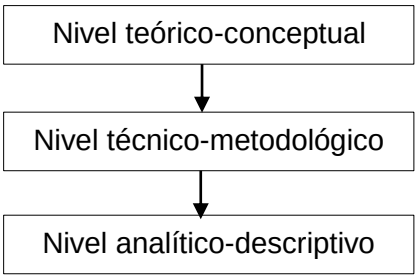


Fuente: Tomado de [47].

3. Metodología

Teniendo en cuenta las características propias de un modelo general de investigación, el diseño de este estudio se estructuró en tres niveles: un nivel teórico-conceptual, que comenzó determinando un problema, estableciendo la búsqueda conceptual y de antecedentes; un nivel técnico-metodológico, que estableció el plan para alcanzar los objetivos planteados y la delimitación de los recursos, y, finalmente, un nivel analítico-descriptivo, en el cuál se determinó la forma en la que se presentarían los resultados (Figura **3-1**). Así mismo, el proyecto general se dividió en fase preliminar, fase de desarrollo y fase final (Ver Capítulo **4.** Cronograma de actividades).

Figura 3-1: Diseño de la investigación.



Fuente: Elaboración propia.

La metodología de trabajo se estructuró a través de tareas consecutivas e interrelacionadas, agrupadas en paquetes de trabajo (PT) (Tabla **3-1**). Durante el desarrollo del proyecto, los resultados obtenidos se difundieron a través de entregables correspondientes a cada uno de los PT de manera sistemática para alcanzar los objetivos propuestos.

Tabla 3-1: Descripción de los paquetes de trabajo estructurados en el diseño metodológico.

Paquete de trabajo **N° PT1**	Inicio:	M1 (AÑO 1)
Título: Seleccionar el sistema de captura de movimiento de acuerdo a la identificación y comparación de las tecnologías disponibles actualmente.		
Entidades participantes		
Nombre corto	UN	TOTAL
Porcentaje de participación	100	100

Descripción	En la fase de desarrollo del proyecto de investigación se realizó una búsqueda de antecedentes y el planteamiento de un marco conceptual entorno al estudio de la cinemática del cuerpo humano, además de análisis comparativos para determinar el sistema de captura de movimiento adecuado teniendo en cuenta los recursos disponibles y el propósito de este estudio como ejes principales. También se requirió apropiación del conocimiento para el manejo y uso adecuado de la tecnología seleccionada en cuanto a la interfaz de operación, registro y captura. La adquisición del sistema MoCap, que se consideró apropiado para el propósito de esta investigación, fue patrocinada por la UN.
Colaboradores	Ninguno.
Tareas	**Tarea 1.1**. Análisis de fortalezas y debilidades de los sistemas MoCap. **Tarea 1.2.** Análisis y estudio según antecedentes de la investigación. **Tarea 1.3.** Análisis de costo y disponibilidad en el mercado. **Tarea 1.4.** Selección del sistema MoCap, según los propósitos del estudio. **Tarea 1.5.** Aprendizaje sobre uso y manejo adecuado el MoCap seleccionado.
Tiempo	8 meses.

Paquete de trabajo **N° PT2**	Inicio:	M3 (AÑO 2)
Título: Realizar la recepción y transmisión de datos desde el sistema de captura de movimiento seleccionado, hasta un entorno virtual que simula un robot social.		
Entidades participantes		
Nombre corto	UN	TOTAL
Porcentaje de participación	100	100

Descripción	Siguiendo con la fase de desarrollo, se establecieron lo recursos computacionales necesarios para realizar la transmisión de los datos registrados por el sistema de captura de movimiento hasta un entorno en el que los mismos pudiesen ser utilizados para crear un archivo ejecutable que recreara un movimiento imitado en el robot NAO (visualizado en un entorno virtual). También se requirió apropiación del conocimiento en cuanto al aprendizaje del entorno ROS.
Colaboradores	Asesores externos e investigaciones previas.
Tareas	**Tarea 2.1. Estudio sobre trabajos previos con objetivos similares.** **Tarea 2.2 Definición de recursos de Hardware/Software.** **Tarea 2.3. Aprendizaje del entorno ROS y principales herramientas a utilizar.** **Tarea 2.4. Recepción de datos transmitidos desde PN.** **Tarea 2.5. Generación del modelo del robot NAO** **Tarea 2.6. Creación de paquete de ROS para teleoperación.**
Tiempo	9 meses.

Paquete de trabajo **N° PT3**	Inicio:	M2 (AÑO 3)
Título: Validar la teleoperación en tiempo real de un robot social, en un entorno virtual, utilizando el sistema de captura de movimiento seleccionado.		
Entidades participantes		
Nombre corto	UN	TOTAL
Porcentaje de participación	100	100

Descripción	Para validar la propuesta del sistema de teleoperación propuesto, se realizan cuatro experimentos que corresponden a diferentes secuencias de movimientos que involucran las cadenas articulares de los brazos del robot, las piernas y la cabeza. Dicha secuencia de movimiento, que bien podría ser una secuencia de terapia, se realizó en tiempo real con el objetivo de constatar la latencia del sistema, además se consideraron posibles objetivos terapéuticos como alcanzar un objeto o favorecer la coordinación y estabilidad. De la misma forma se definieron las limitaciones del sistema por ejemplo las restricciones en los ángulos de movimientos articulares teniendo en cuenta las diferencias antropomórficas entre el instructor humano y el robot humanoide (NAO).
Colaboradores	Ninguno.
Tareas	**Tarea 3.1. Pruebas preliminares del sistema de teleoperación propuesto.** **Tarea 3.2. Verificación de restricciones y limitaciones del sistema.** **Tarea 3.3. Validación experimental de la teleoperación en segmentos articulares – motricidad gruesa.**
Tiempo	6 meses.

Fuente: Elaboración propia.

A continuación, se proporciona una descripción de cómo se desarrolló el sistema de teleoperación unidireccional del robot humanoide NAO (avatar) con el sistema de captura de movimiento inercial Perception Neuron, en tiempo real. Lo que permitirá navegar a través de diferentes posturas y movimientos del robot usando sólo el sistema MoCap (Figura **3-2**).

Figura 3-2: Demostración del sistema de teleoperación en tiempo real del robot NAO (avatar), mediante el sistema MoCap Perception Neuron y su software Axis Neuron.

Fuente: Elaboración propia. Registro fotográfico de abril 2023.

Para empezar, el instructor humano utiliza el traje Perception Neuron con 19 sensores que registran su movimiento mediante sensores IMU, estos datos son capturados con ayuda del software propio del hardware llamado Axis Neuron, también desarrollado por Noitom Limited, el cual se ejecuta en el sistema operativo de Windows, continuamente en el software se reconstruye el modelo del esqueleto humano en tiempo real. Cada cuadro del

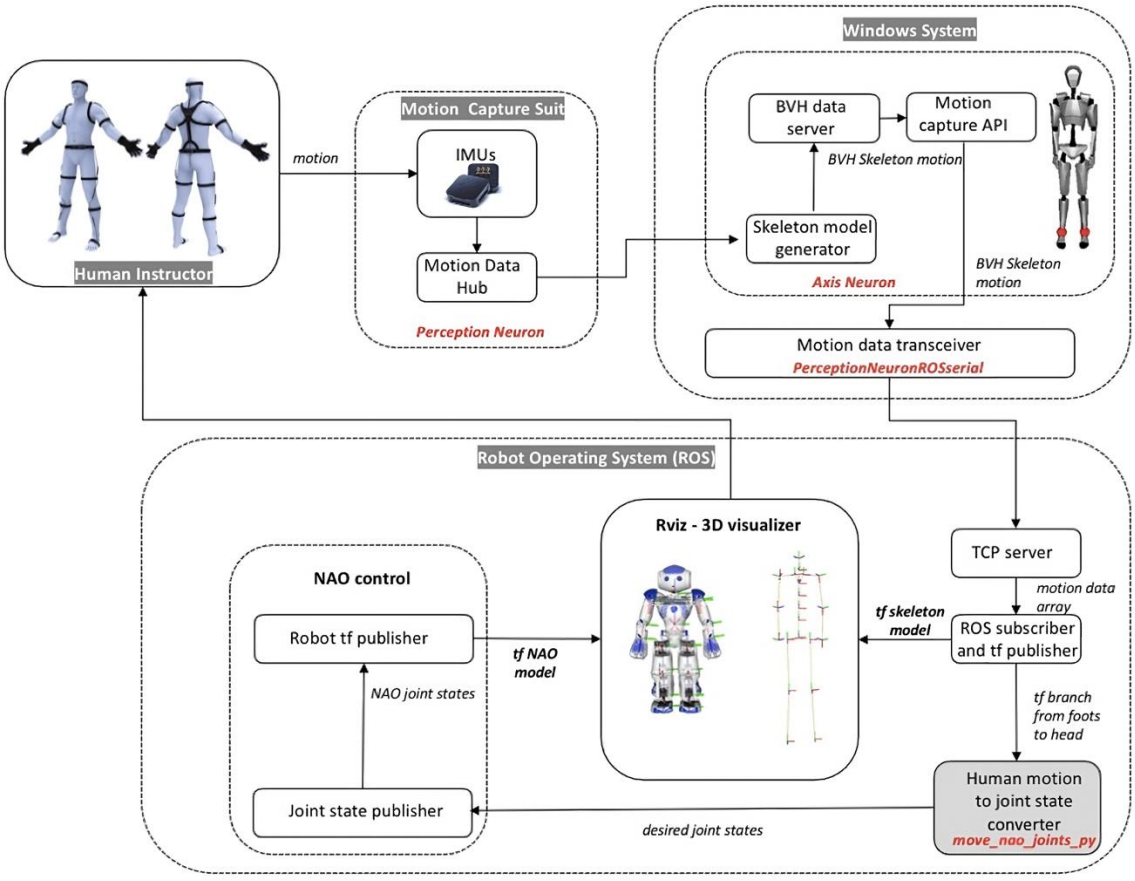
movimiento capturado se aloja en el formato de archivo BVH, el cual fue desarrollado originalmente por Biovision, y proporciona información sobre la jerarquía de datos del modelo del esqueleto humano [34]. Los datos en formato BVH son traducidos a una matriz de datos que puede ser enviada a ROS, a través de la aplicación PerceptionNeuronROSserial en Windows, se trata de un desarrollo previo (disponible en la web) que permite la transmisión de los datos de movimiento de Perception Neuron, pasando por Axis Neuron, a ROS. Fue actualizado por última vez en 2017 por Björn Lütjens con su proyecto *Real-Time Teleoperation of Industrial Robots with the Motion Capture System Perception Neuron* de la Technical University of Munich [51], quien es hoy candidato a doctorado en el Massachusetts Institute of Technology (MIT), su mejora permitió una tasa de transmisión más alta y la posibilidad de poder transmitir todo el modelo del esqueleto, y no sólo una sección del esqueleto como fue desarrollado originalmente por Simon Haller de la University of Innsbruck [52]; la última actualización garantiza la transmisión del total de datos capturados por todos los sensores del sistema MoCap inercial, lo que incluso permite futuras y diversas aplicaciones en robótica, utilizando ROS.

La conexión entre Windows y ROS se realiza a través de una arquitectura de cliente-servidor con el protocolo de comunicación TCP/IP. En ROS (Figura **3-6**) el nodo /*socket* actúa como receptor de la información transmitida desde Axis Neuron, este nodo publica los datos en el tópico de comunicación /*perception_neuron/data* al cual se suscribirá el nodo */perc_publisher* donde se realizarán las operaciones para conocer cómo se deberían mover los marcos de referencia del esqueleto humano transmitido desde Axis Neuron, el nodo */perc_publisher* publicará estos datos en /*tf*. En /tf el usuario puede realizar el seguimiento de múltiples marcos de referencia y podrá acceder a todos los datos de movimiento del esqueleto.

En el nodo /*move_nao_joints_py* (archivo ejecutable propio de la autora de esta tesis) ocurren las operaciones a partir de los datos recibidos, para conseguir el movimiento de las articulaciones del robot a través del tópico de comunicación /*joint_states.* Específicamente, para conseguir el movimiento de los brazos del robot NAO se encontró una solución de cinemática inversa numérica, mientras que para mover las piernas y la cabeza del robot humanoide se logra con la relación *uno-a-uno* entre las articulaciones.

Finalmente, en Rviz, una herramienta de visualización 3D de ROS, se observa por un lado el modelo del esqueleto humano, y por otro, el modelo del robot. El robot NAO está modelado completamente y se dispuso en ROS para su visualización y análisis, este modelo virtual fue actualizado por última vez en 2019, desarrollado originalmente por el Humanoid Robots Lab de la University of Freiburg y Arming Hornung, básicamente permite obtener un paquete de funciones del robot NAO en un entorno virtual (estabilidad y posicionamiento en el espacio, localización del sonido, síntesis de texto a voz, visión artificial, entre otros) y visualizar las versiones 1.14 y 2.1 de la API NaoQI de Aldebaran, fabricantes y distribuidores de NAO [53]. Para ver la arquitectura general del sistema descrito diríjase a la Figura **3-3**.

Figura 3-3: Arquitectura general del sistema.



Fuente: Elaboración propia, con base en [17].

3.2 Hardware

Para la captura del movimiento se utilizó el sistema Perception Neuron (Tabla **3-2**), que consiste en un traje que consta de: 32 IMUs, llamados *Neuron* por el fabricante, cada uno de estos sensores está compuesto por un giroscopio, un acelerómetro y un magnetómetro, tienen una frecuencia de muestreo de 60 fps si hay 18 nodos o más, y de 120 fps si hay menos de 18 nodos, y, un tiempo de cálculo entre 10 y 13 ms; se colocan a lo largo de las articulaciones del cuerpo humano utilizando correas elásticas que se encuentran etiquetadas para conocer la ubicación corporal correcta. El traje también contiene un *Neuron Hub*, este dispositivo concentra y recopila todos los datos de movimiento registrados por los sensores, a través de un cable; se comunica con la aplicación de software Axis Neuron, y envía estos datos a un computador a través de una USB (con un tiempo de transmisión de 1 a 2 ms) o por conectividad inalámbrica. La aplicación Axis Neuron (en Windows), recibe, corrige y procesa los datos de movimiento (con un tiempo de cálculo de 3 a 5 ms), genera un modelo del esqueleto humano completo con una resolución de 0.002 grados [54], que puede ser exportado a otros softwares como Unity, Maya y MotionBuilder o acceder a ellos a través de una API de C++ [33]. La latencia general del traje varía entre 15 y 20 ms.

Tabla 3-2: Traje Perception Neuron 2.0 de Noitom Limited.

	Sensor Neuron	**Hub**	**Correas adhesivas**
	*"Sensor IMU compuesta por un giroscopio, un acelerómetro y un magnetómetro"* [33].	*"El Hub recopila los datos de movimiento de los sensores Neuron. Envía los datos a la computadora a través de USB o por conectividad inalámbrica"* [33].	*"Las correas elásticas aseguran los sensores Neuron al cuerpo. La etiqueta en la parte posterior especifica la ubicación de la correa"* [33].
			

Fuente: Elaboración propia a partir de [33].

Como ya se ha mencionado en el Capítulo **2**. Marco teórico-conceptual, existen variadas opciones para registrar el movimiento humano, por lo que a continuación se explica la selección del sistema MoCap para esta investigación. *"El movimiento representado por los datos de un sistema MoCap, se entiende como una secuencia de poses o un conjunto de trayectorias. Además, la pose se puede entender como un conjunto de coordenadas 3D asignadas a cada articulación alternativamente como un vector de coordenadas 3D, donde cada componente del vector corresponde a la articulación específica en un esqueleto"* [30]. El proceso de grabación del movimiento humano muestra y captura el movimiento para que pueda representarse como la secuencia de cuadros individuales o frames, donde cada frame contiene información sobre: (1) la pose de una persona (una configuración estática del cuerpo) y, opcionalmente, (2) la posición de una persona y (3) la orientación dentro de un espacio registrado [30].

El sistema óptico sin marcadores aplica un método basado en la apariencia para extraer una secuencia de siluetas, luego se ajusta un modelo de esqueleto en la silueta de cada frame registrado y se extraen las características de este modelo. Una de las desventajas que representa este sistema es que los modelos de movimiento humano generados son sólo bidimensionales y están influenciados por el ángulo de visión de la cámara, se puede obtener una modelo tridimensional si se utilizan múltiples cámaras sincronizadas ubicadas alrededor del sujeto que es grabado [30]. Cabe aclarar que la precisión de estos métodos depende de la resolución de la cámara utilizada, además de otros factores externos como el fondo o la ropa del sujeto. El ejemplo clásico de este tipo de dispositivos es el Microsoft Kinect, aunque no se ha definido un estándar para la toma de medidas, se puede utilizar para aplicaciones que requieran un menor grado de exactitud sin dejar de ser fiables. Su principal ventaja es su bajo costo y fácil acceso [11].

Ahora bien, uno de los principales problemas del sistema anterior es su precisión, *"el proceso de estimación conjunta y ajuste del esqueleto es inestable para los dispositivos de grabación sin marcadores"* [30], esto hace que la longitud de los huesos varíe dentro de una secuencia que no es cierta en el mundo real. Por esta razón los sistemas ópticos con marcadores compensan esa desventaja con la obtención de coordenadas articulares más precisas y estables. A su vez, estos marcadores se pueden clasificar en pasivos (hechos de materiales retrorreflectantes) y activos (generan su propia luz y puede controlarse, a través de un diodo emisor de luz [LED]). Estos sistemas como el Vicon

MX40+ son los más precisos y profesionales, pero son de alto costo [30]. Finalmente, se tienen los sistemas de captura de movimiento no-ópticos, que pueden ser: (1) por posicionamiento mecánico, que miden directamente las rotaciones de los ángulos de las articulaciones del cuerpo y no se ven afectados por interferencias magnéticas; (2) posicionamiento por radiofrecuencia, funcionan similar a un radar, trabajan a frecuencias altas de 50 GHz, pero, su precisión es baja, y, (3) basado en sensores inerciales dispuestos en partes específicas del cuerpo humano, la mayoría de estos sistemas utilizan giroscopios y acelerómetros para medir las rotaciones de los ángulos de las articulaciones, los datos son transmitidos de forma inalámbrica generalmente, una de sus desventajas es su error de medición acumulado en el tiempo, por lo cual se recomienda la calibración continua de los sensores, no obstante, es portable, de bajo costo en comparación con los sistemas ópticos sin marcadores y puede capturar a un sujeto que se mueve en grandes áreas [30].

Todo esto para asegurar que *"la selección de un sistema de captura de movimiento depende de su aplicación y presupuesto"* [30]. Por lo cual los sistemas de bajo costo que son más asequibles, aunque menos precisos que los sistemas profesionales, son suficientes para algunos propósitos como estudios deportivos y aplicaciones de robots en movimiento, visión artificial, entre otras [30].

Tabla 3-3: Cuadro comparativo de sistemas MoCap.

Característica:	**Sistemas ópticos con marcadores (p.ej. Vicon)**	**Sistemas ópticos (p.ej. Kinect)**	**Sistemas no-ópticos basados en IMU (p.ej. Perception Neuron)**
Captura información del ambiente	Sí	Sí	No
Precisión	Alta	Media-baja	Media-baja
Fácil manipulación	No	SÍ	Sí
Costo	Alto	Bajo	Medio-Bajo
Portabilidad	Baja	Media	Alta
Realizar estudios fuera del laboratorio	No	Sí (Oclusión de objetos)	Sí (incluso al aire libre)
Tipo de aplicación (p.ej. Biomecánica y rehabilitación)	De alta complejidad y alta especificidad	De media-baja complejidad y baja especificidad	De media-baja complejidad y media-baja especificidad
Requiere personal especializado	Sí	No	No

Fuente: Elaboración propia con base en: [11] [17] [30] [31].

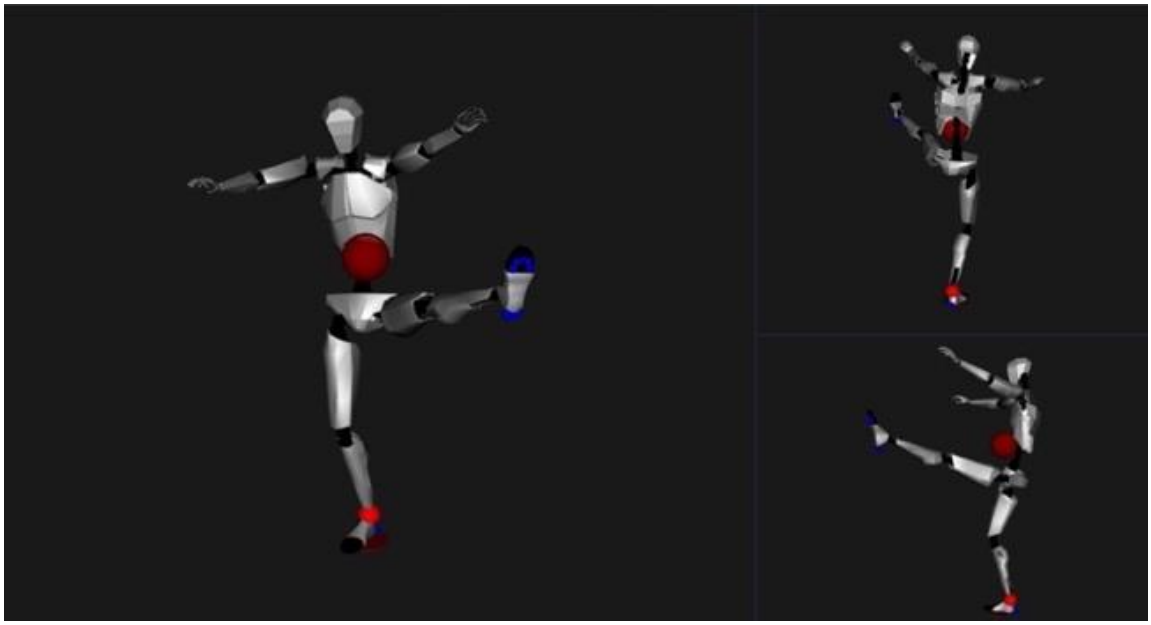
La Tabla **3-3** explica por qué se opta por el sistema de captura de movimiento Perception Neuron 2.0, se trata de un sistema no-óptico inercial, de fácil y rápida configuración, de bajo costo en comparación con un sistema óptico con marcadores o incluso de similares características como el Xsens MVN, ideal para proyectos con bajo presupuesto. Además, puede tener mayor precisión sin el uso de una cámara externa a diferencia de los sistemas ópticos (de una sola cámara), que tienen sin duda un costo más bajo al sistema seleccionado. Otra razón, es que se espera que a futuro el alcance de esta investigación tenga una aplicación en el contexto de la robótica de rehabilitación. Se optó por adquirir el Perception Neuron 2.0 sobre otros sistemas MoCap inerciales comercializados en la actualidad como Rokoko, NANSENSE, Xsens MVN y Chordata (prototipo).

3.3 En Windows (Software)

El software Axis Neuron, que recupera los datos capturados por el traje Perception Neuron, crea un modelo de esqueleto humano 3D (Figura **3-4**), el cual se puede guardar localmente o exportar en formato FBX (Filmbox) o BVH (Biovision). Noitom Limited también proporciona un Software Development Kit (SDK), conocido como MocapAPI SDK antes NeuronDataReader (NDR) SDK [55], propio, para exportar los datos de movimiento desde el software Axis Neuron hacia otras aplicaciones. El SDK NeuronDataReader fue diseñado para usarse en C++ o C#, proporciona al usuario la posibilidad de recibir y utilizar el flujo de datos de los segmentos articulares en formato BVH [17]. A través del protocolo de comunicación TCP/IP el servidor envía los datos de cada cuadro de movimiento a la aplicación en C++ PerceptionNeuronROSserial en Windows, la cual fue una contribución de otros autores. Se utiliza el protocolo TCP/IP para evitar la pérdida de datos [17] [52].

La aplicación PerceptionNeuronROSserial (Figura **3-5**) recupera los datos de movimiento del traje Perception Neuron 2.0 desde la interfaz Axis Neuron (que genera los datos en formato BVH) y los envía a un servidor de ROS [52] [17]. El formato de datos BVH incluye la jerarquía de datos del esqueleto humano y la posición inicial de cada sensor *neuron* con respecto a su nodo principal. El modelo del esqueleto consta de 59 segmentos en total, cada segmento tiene 6 valores tipo float: 3 datos de desplazamiento (X Y Z) y 3 datos de rotación (Y X Z), por lo tanto el modelo en total consta de 354 valores tipo float32 [56] [17]. El software Axis Neuron ordena los datos en jerarquía de acuerdo con la siguiente secuencia de datos mostrada en la Figura **3-5**).

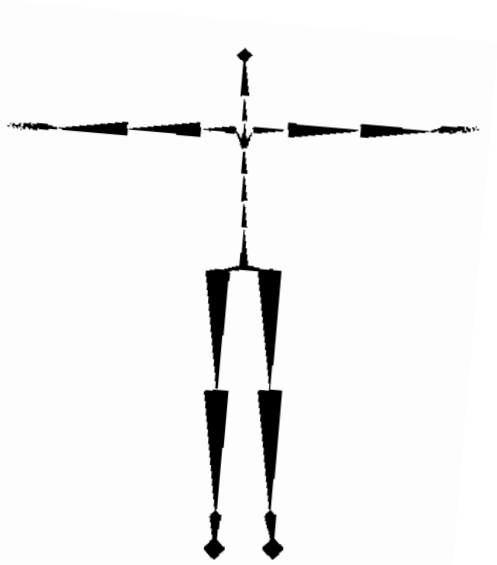
Figura 3-4: Modelo del esqueleto humano generado en Axis Neuron.



Fuente: Elaboración propia. Captura de imagen del archivo *MasterCuiPerformance.raw* utilizando el software Axis neuron.

Figura 3-5: Secuencia de datos del modelo del esqueleto transmitido desde Axis Neuron.

	Bone Name	Sequence In Data Block
Body	Hips (Position)	0
	Hips	1
	RightUpLeg	2
	RightLeg	3
	RightFoot	4
	LeftUpLeg	5
	LeftLeg	6
	LeftFoot	7
	Spine	8
	Spine1	9
	Spine2	10
	Spine3	11
	Neck	12
	Head	13
	RightShoulder	14
	RightArm	15
	RightForeArm	16
	RightHand	17
Fingers	RightHandThumb1	18
	RightHandThumb2	19
	RightHandThumb3	20
	RightInHandIndex	21
	RightHandIndex1	22
	RightHandIndex2	23
	RightHandIndex3	24
	RightInHandMiddle	25
	RightHandMiddle1	26
	RightHandMiddle2	27
	RightHandMiddle3	28
	RightInHandRing	29
	RightHandRing1	30
	RightHandRing2	31
	RightHandRing3	32
	RightInHandPinky	33
	RightHandPinky1	34
	RightHandPinky2	35
	RightHandPinky3	36
Body	LeftShoulder	37
	LeftArm	38
	LeftForeArm	39
	LeftHand	40
Fingers	LeftHandThumb1	41
	LeftHandThumb2	42
	LeftHandThumb3	43
	LeftInHandIndex	44
	LeftHandIndex1	45
	LeftHandIndex2	46
	LeftHandIndex3	47
	LeftInHandMiddle	48
	LeftHandMiddle1	49
	LeftHandMiddle2	50
	LeftHandMiddle3	51
	LeftInHandRing	52
	LeftHandRing1	53
	LeftHandRing2	54
	LeftHandRing3	55
	LeftInHandPinky	56
	LeftHandPinky1	57
	LeftHandPinky2	58
	LeftHandPinky3	59

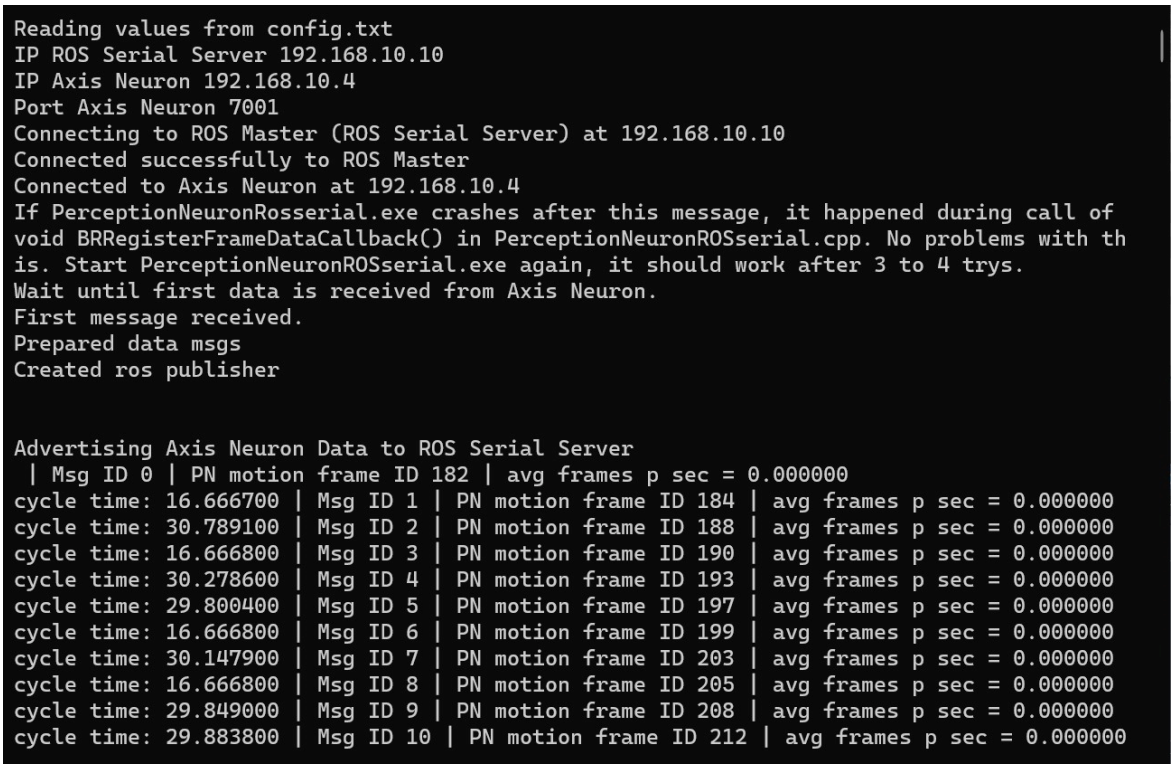


Referencia visual para encontrar los segmentos articulares (huesos)

Fuente: Tomado de [56].

Es importante mencionar, que PerceptionNeuronROSserial (Figura **3-6**) fue una contribución de Simon Haller de la University of Innsbruck [52], que está disponible en https://github.com/smhaller/perception-neuron-ros, fue actualizado por última vez en 2017 por Björn Lütjens y por el Dr. Emmanuel Carlos Dean-Leon en su proyecto de grado en la Technical University of Munich [51] [17], está disponible en https://github.com/blutjens/perc_neuron_ros_ur10, una de las mejoras más importantes es la inclusión de un temporizador de alta resolución, que detiene el bucle principal para adaptarse a la velocidad de transmisión de 60 Hz. La aplicación utiliza también el paquete de ROS *rosserial,* para transferir los mensajes de datos de Windows a ROS, este paquete fue desarrollado por Michael Ferguson en Willow Garage para generalizar los protocolos de comunicación de varios hardware a un ROS Master [17], está disponible en: https://github.com/ros-drivers/rosserial.

Figura 3-6: Demostración de la ejecución de la aplicación PerceptioNeuronROSserial para la transmisión de los datos desde Axis Neuron a ROS.

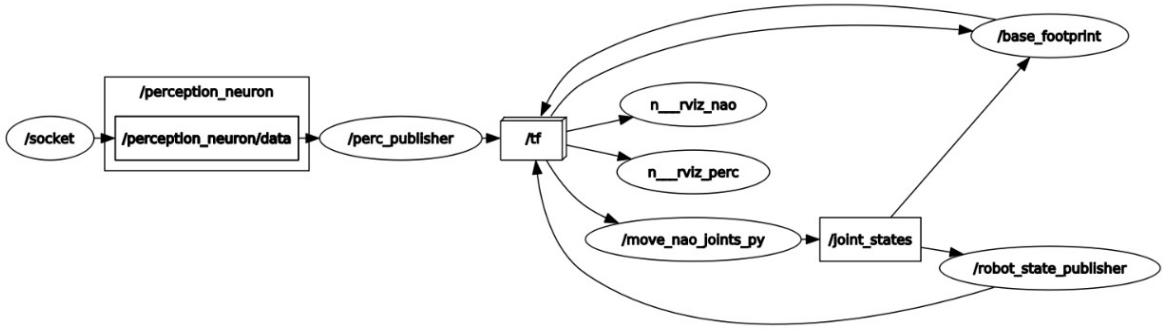


Fuente: Elaboración propia. Captura de imagen.

3.4 En ROS (Software)

Se utiliza el concepto *publicador/suscriptor* para gestionar la comunicación entre el sistema MoCap y el robot NAO (visualizado en Rviz), a través de varios nodos y tópicos.

Figura 3-7: ROS rqt_graph, nodos y tópicos del programa.

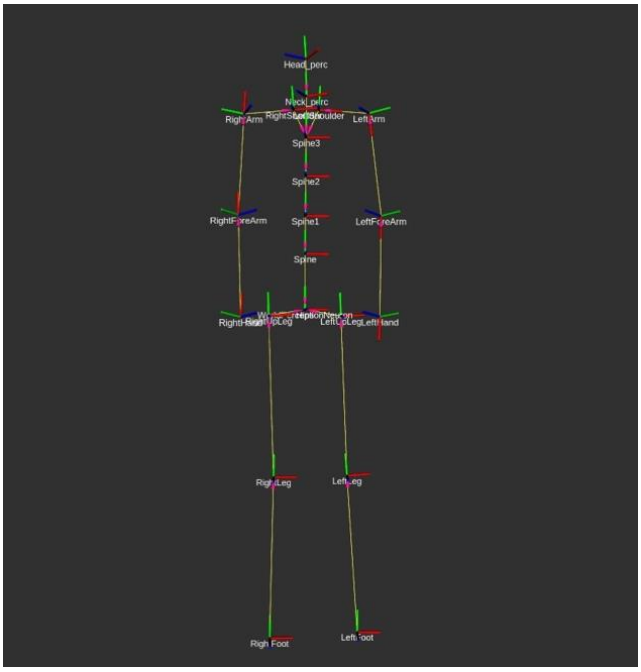


Fuente: Elaboración propia.

En la Figura **3-7** se observa la estructura de nodos y tópicos del programa. El nodo */socket* está recibiendo en ROS los datos transmitidos desde un servidor en serie desde la aplicación PerceptionNeuronROSserial (en Windows) utilizando el protocolo de comunicación TCP/IP, y está publicando los datos al tópico */perception_neuron/data*; el nodo */perc_publisher* se suscribe al tópico de comunicación mencionado, transforma los datos y permite conocer los marcos de referencia del modelo del esqueleto humano (de Axis. Neuron), a su vez este nodo publica los datos en el tópico de comunicación */tf.* El nodo */move_nao_joints_py* se suscribe al tópico de comunicación */tf* y publica los datos en el tópico */joint_states*, al cual finalmente se suscribirán el nodo */robot_state_publisher* para conocer las coordenadas articulares que debe imitar robot NAO.

A continuación, se muestra el modelo del esqueleto humano con los marcos de referencia, el cual debió girarse 90º alrededor de su eje, de modo que tiene la misma orientación que el modelo *tf* del robot NAO (Figura **3-8**).

Figura 3-8: Diagrama de marcos de referencia del modelo del esqueleto transmitido desde Axis Neuron a ROS.

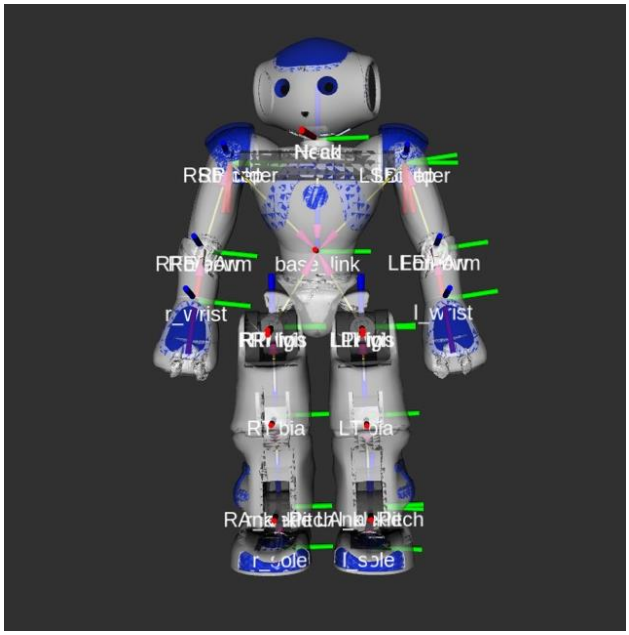


Fuente: Elaboración propia.

El programa usa cada cuadro de movimiento recibido de Perception Neuron y determina los marcos de referencia gracias al tópico de comunicación *tf* (Figura **3-7**), en el cual se representan las relaciones entre marcos de coordenadas en ROS. En este formato *"cada transformación representa una posición relativa de la articulación del esqueleto y la orientación de sí misma con respecto a la articulación anterior. Juntos crean la jerarquía de nodos del modelo de esqueleto humano"* [17]. De tal manera, que el nodo */move_nao_joints_py* puede acceder al árbol de datos completo del esqueleto humano (Figura **3-8**), desde la articulación que funciona como nodo *raíz* hasta los nodos *hijos* de cada segmento articular.

El nodo */move_nao_joints_py* permite al usuario un control intuitivo de los movimientos del robot NAO (visualizado en Rviz), imita la posición y orientación del operador. En términos generales, en el programa MoveNaoJoints.py se crea un objeto para escuchar todas las transformaciones entre los marcos de referencia recibidas desde Axis Neuron, se crea luego el publicador del tópico de comunicación */joint_states*, desde el cual se enviarán los valores articulares al robot NAO (Figura **3-9**).

Figura 3-9: Robot NAO visualizado en Rviz (ROS).



Fuente: Elaboración propia.

- **¿Cómo imita el robot NAO los movimientos del operador con el sistema MoCap?**

Para empezar, se han definido cinco cadenas cinemáticas para el robot NAO, las cuales se describen en la Tabla **3-4**.

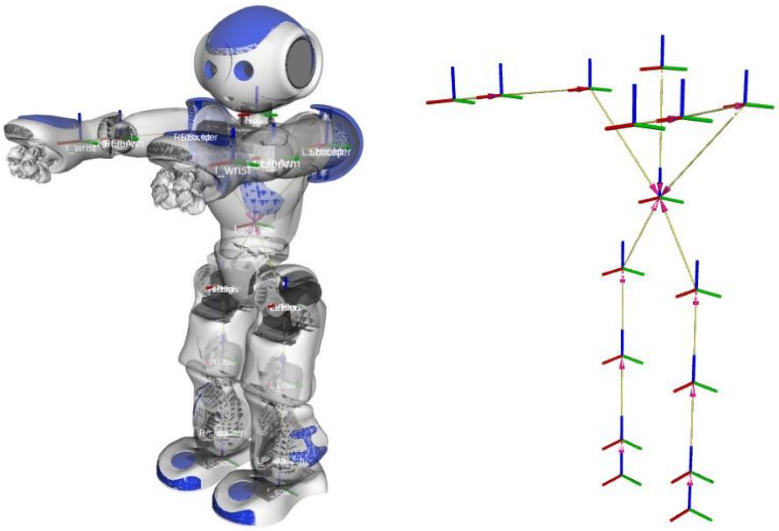
Tabla 3-4: Cadena cinemática y grados de libertad del robot NAO.

Cadena cinemática	Segmento articular	DOFs
Cabeza	*Head (pitch y yaw)*	**2 DOFs**
Brazo	*Shoulder (pitch y roll)*	2 DOFs (R) + 2 DOFs (L) = **4 DOFs**
	Elbow (roll y yaw)	2 DOFs (R) + 2 DOFs (L) = **4 DOFs**
	Wrist (yaw)	1 DOF (R) + 1 DOFs(L) = **2 DOFs**
	Hand (open/close)	1 DOF (R) + 1 DOF (L) = **2 DOFs**
Torso	*Pelvis (yaw/pitch)*	**1 DOF**
Pierna	*Hip (pitch y roll)*	2 DOFs (R) + 2 DOFs (L) = **4 DOFs**
	Knee (pitch)	1 DOF (R) + 1 DOF (L) = **2 DOFs**
	Ankle (pitch y roll)	2 DOF (R) + 2 DOF (L) = **4 DOFs**
	Total:	**25 DOFs**

Fuente: Elaboración propia con base en [57].

El robot NAO comienza con la posición de cero de las articulaciones, como se muestra en la Figura **3-10** y Figura **3-11**.

Figura 3-10: Marco de referencia de la posición inicial del robot NAO.



Fuente: Elaboración propia.

La posición inicial del robot NAO, está definido en el paquete MoveNaoJoints.py en el código fuente que se muestra en la Figura **3-11**. Ahora bien, para transmitir los ángulos de las articulaciones al robot NAO cuando se realiza un movimiento, los ángulos del esqueleto humano deben mapearse a los valores equivalentes del robot NAO, así por ejemplo la articulación *RightLeg* en el modelo del esqueleto humano corresponde a la articulación *RTibia* en el robot, y, las articulaciones *LeftFoot, LeftLeg* y *LeftUpLeg* en el modelo del esqueleto humano y que registran el movimiento de la pierna izquierda corresponden a las articulaciones *LAnkle, Lknee y LHip* en el robot, es necesario realizar un escalamiento. Sin embargo, no a todas las articulaciones se les puede aplicar el método uno-a-uno, por lo que para el movimiento de los brazos para encontrar la posición del efector final se encontró una solución de cinemática inversa numérica. Es importante mencionar que con el sistema que aquí se propone no es posible rastrear movimientos que impliquen habilidades motoras finas como el movimiento de las manos al agarrar un objeto, ni es posible rastrear movimientos tan complejos como la marcha humana. Después de que se realizan estas operaciones el nodo */robot_state_publisher* se suscribe al tópico de

comunicación */joint_states* (Figura **3-7**) para conocer los ángulos de las articulaciones que deberá imitar el robot NAO.

Figura 3-11: Código fuente de la posición inicial del robot NAO.

```
30
31      #-- Variables to store the joint angles
32
33      self.joints_prev_left = [0.0, 0.0, 0.0 ,0.0, 0.0]
34      self.joints_prev_right = [0.0, 0.0, 0.0 ,0.0, 0.0]
35
36      #- Left leg
37      self.nao_LHipPitch = 0.0
38      self.nao_LHipRoll = 0.0
39      self.nao_LKneePitch = 0.0
40      self.nao_LAnklePitch = 0.0
41      self.nao_LAnkleRoll = 0.0
42
43      #- Right leg
44      self.nao_RHipPitch = 0.0
45      self.nao_RHipRoll = 0.0
46      self.nao_RKneePitch = 0.0
47      self.nao_RAnklePitch = 0.0
48      self.nao_RAnkleRoll = 0.0
49
50      #- Left arm
51      self.nao_LShoulderPitch = 0.0
52      self.nao_LShoulderRoll = 0.0
53      self.nao_LElbowYaw = 0.0
54      self.nao_LElbowRoll = 0.0
55      self.nao_LWristYaw = 0.0
56
57      #- Right arm
58      self.nao_RShoulderPitch = 0.0
59      self.nao_RShoulderRoll = 0.0
60      self.nao_RElbowYaw = 0.0
61      self.nao_RElbowRoll = 0.0
62      self.nao_RWristYaw = 0.0
63
64      #- Head
65      self.nao_HeadYaw = 0.0
66      self.nao_HeadPitch = 0.0
67
```

Fuente: Elaboración propia.

De esta forma, en el código fuente del paquete MoveNaoJoints.py, por ejemplo para encontrar los ángulos de movimiento *LHipPitch* y *LHipRoll* en el robot (Figura **3-12**), primero se busca la transformación entre dos marcos de referencia de Perception Neuron, en este caso desde *Hips* (vector I) hasta *LeftUpLeg* (vector II), ver Tabla **3-5**, los dos valores que resultan de la transformación entre estos marcos de referencia se pasan de cuaternión a ángulos de euler, y luego se obtienen los ángulos roll y pitch. Finalmente, se

establecen los límites de los rangos articulares de la articulación con un condicional *if* para evitar movimientos del robot NAO fuera de los rangos normales de angulación.

Figura 3-12: Ejemplo. Código fuente para hallar transformación entre dos marcos de referencia y ángulos roll y pitch de la cadera en el robot.

```
71      #-- Lookup transform from Hips to Left leg
72      try:
73          (_,rot_Hip_LeftLeg) = self.listener.lookupTransform('/LeftUpLeg', '/Hips', rospy.Time(0))
74          #-- Pass quaternion to euler angles
75          rot_Hip_LeftLeg = euler_from_quaternion(rot_Hip_LeftLeg)
76          #-- Get the roll and pitch
77          nao_LHipPitch = rot_Hip_LeftLeg[0]
78          nao_LHipPitch = nao_LHipPitch*-1 #-- Invert the sign
79
80          nao_LHipRoll = rot_Hip_LeftLeg[2]
81          nao_LHipRoll = nao_LHipRoll*-1 #-- Invert the sign
82
83          #-- Limit the angle to the range of motion of the robot
84          if nao_LHipPitch < -1.54:
85              nao_LHipPitch = -1.54
86          if nao_LHipPitch > 0.48:
87              nao_LHipPi
88                          Loading...
89          if nao_LHipRoll < -0.38:
90              nao_LHipRoll = -0.38
91          if nao_LHipRoll > 0.79:
92              nao_LHipRoll = 0.79
93
94          self.nao_LHipPitch = nao_LHipPitch
95          self.nao_LHipRoll = nao_LHipRoll
96
97          #-- Print the values
98          # rospy.loginfo("nao_LHipPitch: %f  nao_LHipRoll: %f", nao_LHipPitch, nao_LHipRoll)
99
100     except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
101         pass
```

Fuente: Elaboración propia.

El mapeo de los marcos de referencia de las articulaciones del Perception Neuron utilizados para encontrar los ángulos articulares en el robot NAO, se muestra a continuación en la Tabla **3-5.**

Tabla 3-5: Mapeo de los ángulos articulares.

NAO Robot			Perception Neuron	
Kinematic Chain	*Joint*	*Angle*	*Vector I*	*Vector II*
Left Leg	LHip	Pitch	Hips	LeftUpLeg
		Roll	Hips	LeftUpLeg
	LKnee	Pitch	LeftUpLeg	LeftLeg
	LAnkle	Pitch	LeftLeg	LeftFoot
		Roll	LeftLeg	LeftFoot

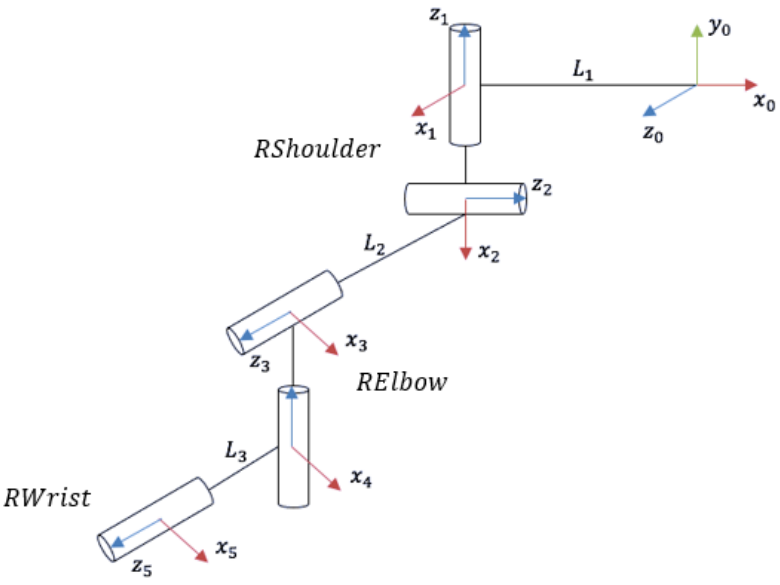
Right Leg	RHip	Pitch	Hips	RightUpLeg
		Roll	Hips	RightUpLeg
	RKnee	Pitch	RightUpLeg	RightLeg
	RAnkle	Pitch	RightLeg	RightFoot
		Roll	RightLeg	RightFoot
Left Arm	LShoulder	Pitch	LeftArm	LeftShoulder
		Roll	LeftArm	LeftShoulder
	LElbow	Yaw	LeftForeArm	LeftArm
	LElbow	Roll	LeftForeArm	LeftArm
	LWrist	Yaw	LeftHand	LeftForeArm
Right Arm	LShoulder	Pitch	RightShoulder	RightArm
		Roll	RightShoulder	RightArm
	Elbow	Yaw	RightArm	RightForeArm
		Roll	RightArm	RightForeArm
	Wrist	Yaw	RightForeArm	RightHand
Head	Head	Yaw	Head_perc	Spine3
		Pitch	Head_perc	Spine3

Fuente: Elaboración propia.

Como se muestra en los antecedentes de esta investigación (Capítulo **1**. Descripción del proyecto, Sección **1.1**.) el movimiento de los brazos en un robot humanoide es uno de los más importantes a estudiar, dado que es con las extremidades superiores con las que regularmente podemos alcanzar o agarrar objetos. Teniendo en cuenta esto, y que no era posible desacoplar el movimiento del brazo para ejecutarlo *uno-a-uno* (como en los segmentos articulares de las piernas y de la cabeza), se optó por encontrar una solución por cinemática inversa numérica para encontrar el valor de los ángulos articulares a partir de la posición del efector final, y así lograr que el robot realizará una adecuada imitación del movimiento de las extremidades superiores, se establece está una solución con el objetivo principal de conocer con gran precisión la ubicación de la mano tras un movimiento. Aclarando que aunque se pueden transmitir desde el hardware a Axis Neuron el movimiento de los dedos de las manos para agarrar un objeto por ejemplo, este movimiento fino puede ser capturado de forma inestable por el sistema MoCap [35]. De otro lado, este objetivo está fuera del alcance de este proyecto, requeriría de soluciones más complejas. Estudiar las habilidades motoras gruesas como el movimiento de los brazos y las piernas, implica que el robot NAO pueda realizar en un futuro movimientos amplios como: caminar.

Para encontrar la solución utilizando cinemática inversa numérica, primero se encuentra la cinemática directa de los brazos del robot NAO por el algoritmo de Denavit-Hartenberg con base en los ejes coordenados encontrados (Figura **3-13**), la cadena cinemática de cada brazo del robot consiste en cinco articulaciones, en la Tabla **3-6** se muestran los parámetros DH modificados. Para ver el modelo generado remítase a la Figura **7-1** en la sección de Anexos en este documento.

Figura 3-13: Diagrama con la asignación de los ejes coordenados del brazo derecho del robot NAO.



Fuente: Elaboración propia.

Tabla 3-6: Parámetros DH modificados de las articulaciones del brazo derecho del robot NAO.

Joint	L	L_{i-1}	a_{i-1}	d_i	θ_i	$off-sets$
$RShoulderRoll$	1	$-\pi/2$	$-L_1$	0	q_1	$-\pi/2$
$RhoulderPitch$	2	$-\pi/2$	0	0	q_2	$\pi/2$
$RElbowYaw$	3	$\pi/2$	0	L_2	q_3	$\pi/2$
$RElbowRoll$	4	$-\pi/2$	0	0	q_4	0
$RWristYaw$	5	$\pi/2$	0	L_3	q_5	0

Fuente: Elaboración propia.

Entonces, el código fuente del programa MoveNaoJoints.py se elaboró como se muestra en la Figura **3-15** para que el robot NAO pudiera imitar el movimiento de los brazos. En la línea 540 los valores obtenidos de la transformación entre *RightHand* (efector final) y *RightShoulder* (base) se pasan de cuaterniones a una matriz de rotación, enseguida, en la línea 545 se crea la matriz de transformación del brazo teniendo en cuenta el desplazamiento y la rotación. En la línea 547 (*ikine_LMS*) de la Figura **3-15** se utiliza cinemática inversa numérica con el algoritmo de Levenberg-Marquadt con la variante de Sugihara, o en inglés *Levenberg-Marquadt with Sugihara's tweak (*Figura **3-14***),* un método de mínimos cuadrados con amortiguación de errores, esta función se itera 50 veces hasta encontrar el valor esperado, se encuentra indexada en la librería *roboticstoolbox* [58]*,* ver la sección de anexos de este documento. Este algoritmo permite encontrar soluciones de Cinemática Inversa para una serie de poses a lo largo de una trayectoria deseada del efector final [58] [59]. Pero, es probable que no todos los movimientos que realiza el operador con los brazos puedan ser imitados fielmente por el robot, es decir es posible que no haya solución; lo que no ocurriría con el movimiento de la cabeza y las piernas donde la relación entre las articulaciones es desacoplada y se realiza *uno-a-uno*. A continuación se muestra una sinopsis del método utilizado según [59]:

Figura 3-14: Descripción del método de cinemática inversa numérica por el algoritmo de LMS, función *ikine_LMS.*

Para empezar, la operación está definida por la elección de una variable kwarg (k).
Este paso está definido cómo:

$$\boldsymbol{q}_{\mathrm{k}+1} = \boldsymbol{q}_{\mathrm{k}} + (\boldsymbol{A}_{\mathrm{k}})^{-1}\boldsymbol{g}_{\mathrm{k}}$$

$$\boldsymbol{A}_{\mathrm{k}} = \boldsymbol{J}(\boldsymbol{q}_{\mathrm{k}})^{\intercal}\,\mathbf{W}_e\,J(\boldsymbol{q}_{\mathrm{k}}) + \mathbf{W}_n$$

Dónde $\mathbf{W}_n = diag(\omega_n)(\omega_n \in \mathbb{R}^n_{>0})$ *es una matriz de amortiguación diagonal. La matriz de amortiguación garantiza que* $\boldsymbol{A}_k$ *este definido como* $no - singular\ y\ positivo$.
El rendimiento del método de $\boldsymbol{Levenberg - Marquadt}$ $(\mathbf{LM})$*depende en gran medida de la elección de* $\mathbf{W}_n$.
En este sentido, la técnica optimizada de $\boldsymbol{Sugihara}$ *propone lo siguiente*:

$$\mathbf{W}_n = E_k\mathbf{1}_n + diag(\hat{\boldsymbol{\omega}}_n)$$

Dónde $\hat{\boldsymbol{\omega}}_n \in \mathbb{R}^n, \hat{\omega}_{n_i} = l^2 \sim 0.01l^2$, *y* l *es la longitud de una sección rígida en un manipulador.*
Se proporciona la variable k *como un kwarg para ajustar el valor de* ω_n.

Fuente: Elaboración propia con base en [59].

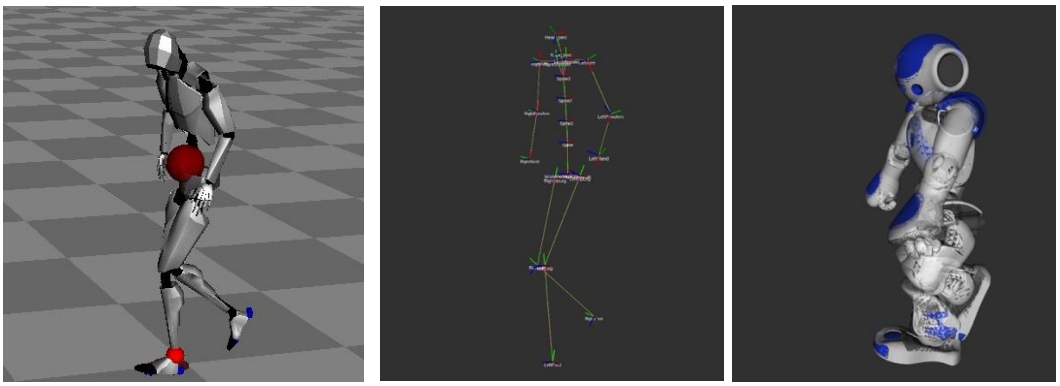
Figura 3-15: Código fuente de la solución de cinemática inversa numérica para el brazo derecho del robot NAO.

```
524    def getJointStates_Right_arm(self):
525
526        # Define robot
527        L1 = 0.128
528        L2 = 0.2655
529        L3 = 0.260
530
531        robot_model = rtb.DHRobot([
532                                   rtb.RevoluteMDH(alpha=-pi/2, a=-L1, d=0.0,  qlim=[-1.33, 0.31], offset=-pi/2),
533                                   rtb.RevoluteMDH(alpha=-pi/2, a=0.0, d=0.0,  qlim=[-2.09, 2.09], offset= pi/2),
534                                   rtb.RevoluteMDH(alpha= pi/2, a=0.0, d=L2,   qlim=[-2.09, 2.09], offset= pi/2),
535                                   rtb.RevoluteMDH(alpha=-pi/2, a=0.0, d=0.0,  qlim=[ 0.00, 1.54], offset=0.0),
536                                   rtb.RevoluteMDH(alpha= pi/2, a=0.0, d=L3,   qlim=[-1.82, 1.82], offset=0.0),
537        ])
538
539        try:
540            (trans_RightArm,rot_RightArm) = self.listener.lookupTransform('/RightShoulder', '/RightHand', rospy.Time(0))
541            # print("Distance: ", trans_RightArm)
542            #--- Pass quaternion to matrix
543            rot_matrix = quaternion_matrix(rot_RightArm)[0:3,0:3]
544
545            T_ikine = SE3.Trans(trans_RightArm[0],trans_RightArm[1],trans_RightArm[2]) * SE3.Rt(rot_matrix)
546
547            joints_ikine = robot_model.ikine_LMS(T_ikine,q0=self.joints_prev_right, mask=[1, 1, 1, 0, 0, 0], tol=1e-4, ilimit=50)
548            if joints_ikine.success:
549
550                #-- Set joint values
551                nao_RShoulderPitch = joints_ikine.q[1]
552                nao_RShoulderRoll  = joints_ikine.q[0]
553                nao_RElbowYaw      = joints_ikine.q[2]
554                nao_RElbowRoll     = joints_ikine.q[3]
555                nao_RWristYaw      = joints_ikine.q[4]
556
557                #-- Limit the angle to the range of motion of the robot
```

Fuente: Elaboración propia.

Para ver el código fuente completo puede referirse a los anexos de este documento. Finalmente, en la Figura **3-16** se muestra el resultado de la teleoperación a partir de un movimiento previamente grabado con el Perception Neuron 2.0.

Figura 3-16: Demostración del sistema de teleoperación con un movimiento ejecutado.

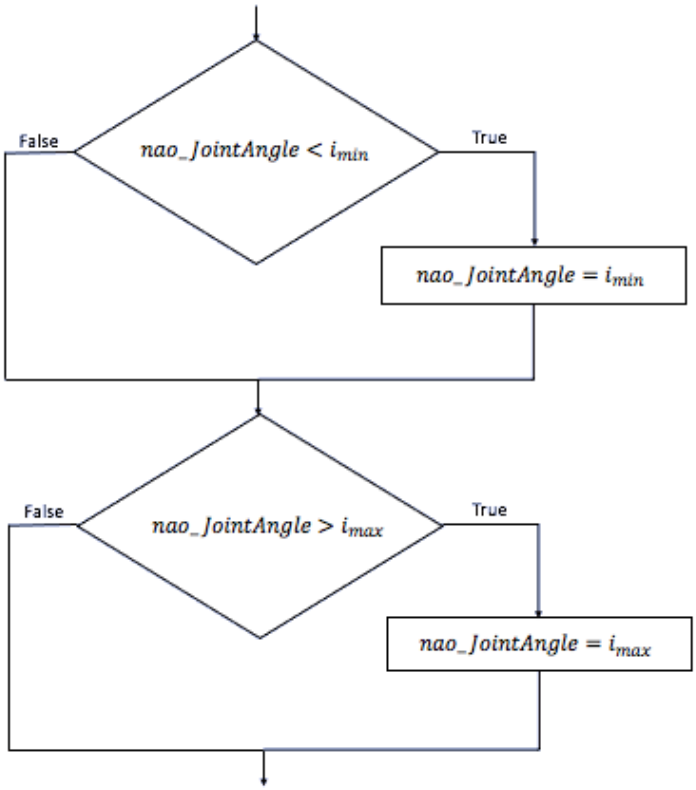


Windows *(Software Axis Neuron)*	***ROS*** *Rviz – Diagrama tf del modelo del esqueleto y el robot NAO*
Computador no. 1	***Computador no. 2***

Fuente: Elaboración propia.

Cabe aclarar que la cantidad de poses que el operador podría realizar está limitada por el rango de movimiento de las articulaciones (motores) del robot NAO, para evitar daños en un robot real deben existir reglas que limiten el movimiento al enviar ángulos que el robot no puede alcanzar. En el programa se utiliza el condicional *if (*Figura **3-17**) para definir el rango máximo y mínimo de los ángulos del robot cuando imita el movimiento del operador.

Figura 3-17: Estructura condicional de la definición de los límites del rango articular del robot NAO.



Fuente: Elaboración propia.

Experimentalmente se definieron los límites de los ángulos mínimos y máximos para las articulaciones del robot NAO, teniendo en cuenta también lo descrito por el fabricante para la operación de movimientos en NAO [60] como se muestra en la Tabla **3-7**.

Tabla 3-7: Límites de los rangos articulares en el robot NAO, definidos experimentalmente con el modelo del robot generado en Rviz-ROS.

Kinematic Chain	Joint	i_{min} (rad)	i_{max} (rad)
Left Leg	LHipPitch	-1.54	0.48
	LHipRoll	-0.38	0.79
	LKneePitch	-0.09	2.11
	LAnklePitch	-1.19	0.92
	LAnkleRoll	-0.40	0.77
Right Leg	RHipPitch	-1.54	0.48
	RHipRoll	-0.79	0.38
	RKneePitch	-0.09	2.11
	RAnklePitch	-1.19	0.92
	RAnkleRoll	-0.77	0.40
Left Arm	LShoulderPitch	-2.09	2.09
	LShoulderRoll	-0.31	1.33
	LElbowYaw	-2.09	2.09
	LElbowRoll	-1.54	0.00
	LWristYaw	-1.82	1.82
Right Arm	RShoulderPitch	-2.09	2.09
	RShoulderRoll	-1.33	0.31
	RElbowYaw	-2.09	2.09
	RElbowRoll	0.00	1.54
	RWristYaw	-1.82	1.82
Head	HeadYaw	-2.09	2.09
	HeadPitch	-0.67	0.51

Fuente: Elaboración propia.

Para terminar, el análisis de resultados se muestra comparativamente y se interpreta de forma descriptiva tras la experimentación con cuatro rutinas de movimientos que involucran diferentes cadenas articulares (Ver Capítulo **6.** Resultados). Aunque se podrá evidenciar gráficamente el movimiento articular máximo alcanzado por el instructor, es importante destacar que en las etapas tempranas de una terapia de rehabilitación no se esperaría precisión del movimiento por parte de la persona que requiere asistencia, esto es, que alcance el movimiento articular máximo con respecto al movimiento guiado por el instructor. En esta etapa se espera comprensión de la instrucción y una imitación secuencial de la rutina, por lo que el análisis de la tendencia se considera suficiente.

4. Cronograma de actividades

A continuación, se muestra el cronograma de actividades ejecutadas con base en la metodología planteada en la sección anterior.

PROYECTO: Diseño de un método de captación de movimientos para el registro, análisis y transmisión de datos a plataformas de rehabilitación robóticas, en un entorno virtual.		AÑO 1 (2019-2S - 2020-1S)												AÑO 2 (2021-1S - 2021-2S)												AÑO 3 (2023-1S)											
		M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
Actividad	**FASE PRELIMINAR**																																				
a.	Establecimiento inicial de la propuesta y validación de la idea.																																				
b.	Rastreo de antecedentes, y establecimiento del fundamento teórico y conceptual.																																				
c.	Planificación y estudio de objetivos.																																				
d.	Elaboración de la propuesta de investigación.																																				
e.	Presentación y aceptación de la propuesta de tesis de maestría.																																				
f.	Presentación de versión inicial del documento de proyecto de tesis de maestría.																																				
g.	Asesorías extracurriculares.																																				
Actividad	**FASE DE DESARROLLO**																																				
N° PT1. Seleccionar el sistema de captura de movimiento de acuerdo a la identificación y comparación de las tecnologías disponibles actualmente.																		**1**																		
Tarea PT1.1	Exploración y estudio sobre los sistemas de captura de movimiento disponibles en el mercado.																																				
Tarea PT1.2	Selección del sistema de captura de movimiento.																																				
Tarea PT1.3	Exploración y aprendizaje sobre el hardware y software del sistema de captura de movimiento seleccionado, Perception Neuron 2.0 , fabricado por Noitom Limited.																																				
Tarea PT1.4	Elaboración del informe de avance satisfactorio y presentación.																																				
N° PT2 . Realizar la recepción y transmisión de datos desde el sistema de captura de movimiento seleccionado, hasta un entorno virtual que simula un robot social.																		**2**																		
Tarea PT2.1	Exploración sobre trabajos previos con objetivos similares.																																				
Tarea PT2.2	Definición de utilización de recursos en Windows y ROS para lograr la transmisión de datos.																																				
Tarea PT2.3	Recepción de datos en Axis Neuron y transmisión a la API PerceptionNeuronROSserial.																																				
Tarea PT2.4	Transmisión de datos desde PeceptionNeuronROSserial (en Windows) a ROS.																																				
Tarea PT2.5	Recepción y tratamiento de datos en ROS para generación del esqueleto humano.																																				
Tarea PT2.6	Generación del modelo virtual del robot NAO en ROS.																																				
Tarea PT2.7	Definición del modelo matemático a utilizar para realizar la teleoperación.																																				
Tarea PT2.8	Elaboración de solicitud de modificación a la propuesta de investigación.																																				
Tarea PT2.9	Elaboración del informe de avance satisfactorio y presentación.																																				
N° PT3. Validar la teleoperación en tiempo real de un robot social, en un entorno virtual, utilizando el sistema de captura de movimiento seleccionado.																		**3**																		
Tarea PT3.1	Pruebas iniciales teleoperación del robot NAO, generado virtualmente en ROS, con el sistema de captura de movimiento Perception Neuron 2.0., en tiempo real.																																				
Tarea PT3.2	Validación de la teleoperación con diferentes movimientos articulares.																																				
Actividad	**FASE FINAL**																																				
a.	Elaboración del Informe final.																																				
b.	Elaboración del artículo científico para publicación en revista indexada.																																				
Nota: Los recuadros hacen referencia a los objetivos asociados a cada una de las fases de la propuesta de investigación																																					

5. Presupuesto y fuentes de financiación

En esta sección se muestra el presupuesto (con valor en pesos colombianos) que fue necesario para la ejecución de la propuesta de investigación presentada.

Tabla 5-1: Presupuesto y fuentes de financiación.

Detalle	Rol	Ud.	Cant.	Valor unitario	Valor total
Estudiante	Estudiante	Hora	960	$ 30.000	$ 28.800.000
Profesor	Director/Co-director	Hora	240	$ 120.594	$ 28.942.560
Colaboradores	Asesoría externa	Hora	140	$ 50.000	$ 7.000.000
Sistema MoCap	Hardware/Software	-	1	$ 12.000.000	$ 12.000.000
Computador	Linux (ROS)	-	1	$ 3.500.000	$ 3.500.000
Computador	Windows	-	1	$ 1.800.000	$ 1.800.000
Accesorios	Cable ethernet y adaptador	-	1	$ 30.000	$ 30.000
Accesorios	Batería portátil	-	1	$ 50.000	$ 50.000
				TOTAL	**$ 82.122.560**

NOTA: La Universidad Nacional de Colombia patrocinó la compra del Sistema MoCap Perception Neuron 2.0, fabricado por Noitom Limited.

6. Resultados

A continuación, se evalúa el sistema propuesto en términos de la similitud del movimiento del robot NAO con el movimiento ejecutado por el operador no-técnico. Como se ha demostrado a lo largo de este documento, para controlar el robot humanoide de forma remota fue necesario capturar el movimiento humano con un sistema MoCap inercial conocido comercialmente como Perception Neuron 2.0, los datos capturados son los datos de entrada al algoritmo de movimiento del robot NAO en ROS. No se tienen en cuenta las longitudes de las extremidades, ni el centro de masa del robot o la dinámica del robot, ya que no es objeto de este estudio, si lo fuese, tendrían que implementarse soluciones más complejas que consideraran un modelo dinámico del robot.

Teniendo en cuenta lo que se espera a futuro a partir de esta investigación, se realizó la secuencia de movimientos que se muestra en la Figura **6-1** y Figura **6-2** con base en el estudio de investigación titulado *Developing a Robot-Guided Interactive Simon Game for Physical and Cognitive Training* [61] el cual estableció una secuencia de movimientos en medio de una actividad de juego con un robot NAO real (juguete), la secuencia de movimientos fue probada con 56 niños con edades entre 5 y 6 años. Las actividades de juego utilizadas fueron: el juego de *Simón dice* con una secuencia de poses que el usuario debía recordar y realizar en el mismo orden, y el juego de *El espejo* que consistía en que el paciente imitara la pose del robot. Algunos de estos ejercicios también se identifican como parte de la secuencia de movimientos del estudio titulado *Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning*, en dicha investigación se tuvo previsto que el robot NAO actuara como un co-terapeuta [21].

Es importante mencionar, que el sistema propuesto permite que la secuencia de movimientos en el robot NAO pueda ser individualizada teniendo en cuenta las limitaciones físicas de los pacientes, de manera tal que el usuario no-técnico (terapeuta, educador o cuidador) puede personalizar el ejercicio y adaptar consecutivamente el nivel de dificultad

desde posturas simples hasta las más complejas. Lo anterior, según se menciona en [18] resulta útil para mejorar rangos de movilidad articular, evaluar procesos terapéuticos y explorar las capacidades del paciente.

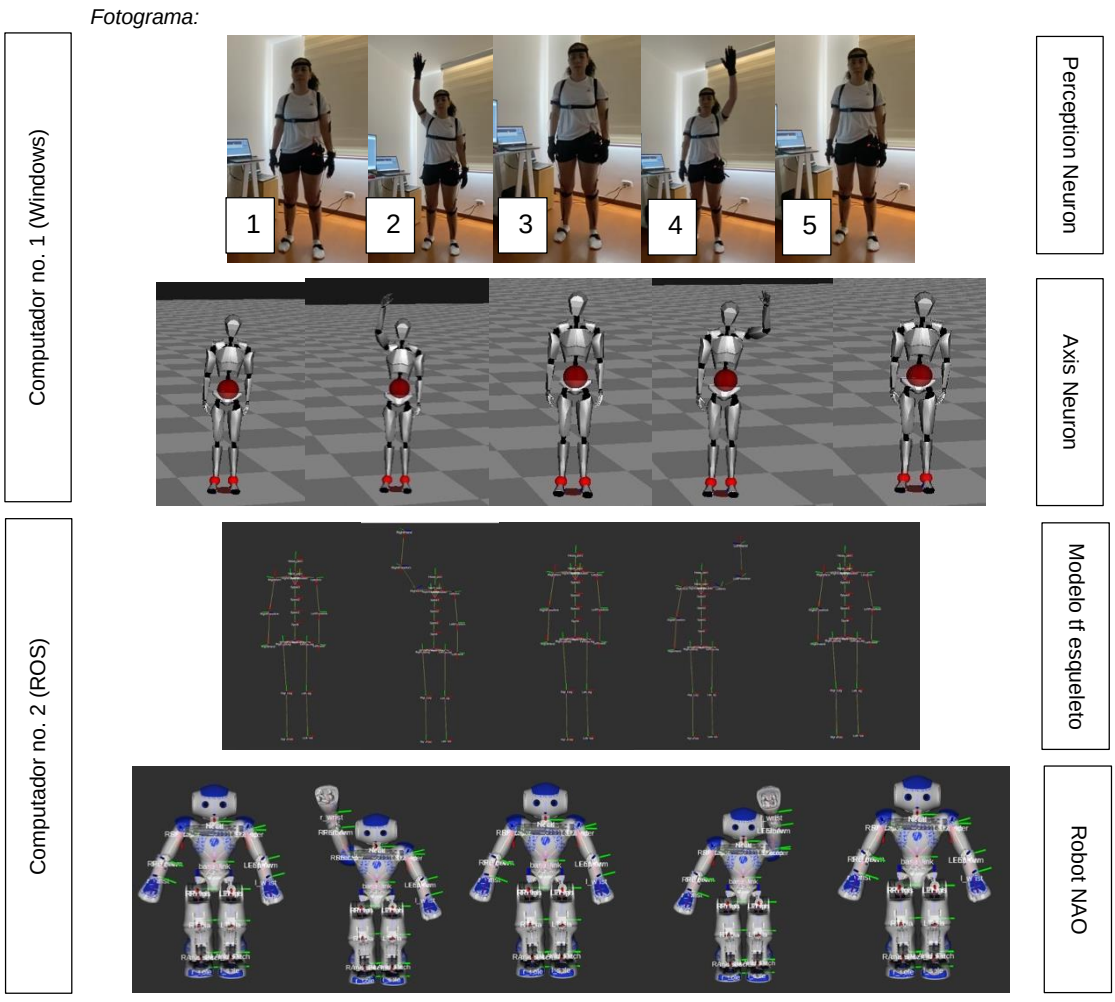
Las restricciones del diseño, estuvieron dadas por las limitaciones angulares propias del robot, si el movimiento de la cabeza, brazos y piernas del usuario no-técnico excede los límites de movimientos articulares del robot NAO, el robot tratará de encontrar una solución para acercarse al movimiento. En cuanto a la ubicación del operador no-técnico, se evidenció durante la experimentación, que éste puede ubicarse a máximo 10 metros en un espacio abierto o cerrado sin que se vea afectado de manera significativa la transmisión de los datos vía Wi-Fi desde el hardware Perception Neuron 2.0 hasta el software Axis Neuron. Dadas las ventajas de capturar los movimientos con un sistema inercial el operador no-técnico puede realizar cualquier pose y ser seguida efectivamente por el sistema sin que se vea afectado por la presencia de objetos [62], por lo cual el movimiento no se ve malinterpretado al estar oculto u oscurecido.

El sistema fue probado durante 30 min, con pausas de 2-5 min, con diferentes rutinas que involucraban el movimiento de la cabeza y de las extremidades. Tal como se evidenció en [19] cuando el sistema MoCap Perception Neuron se opera en tiempo real durante un periodo largo de tiempo es necesario calibrarlo tras cada rutina de ejercicios ejecutada para funcione de la manera adecuada, lo que no se refleja cuando se realiza la teleoperación con una rutina de movimientos previamente almacenados en el software Axis Neuron. Tampoco se evidenció una latencia significativa en la transmisión de los movimientos a partir del hardware (Sistema MoCap Perception Neuron), ni por la comunicación TCP/IP entre los sistemas computacionales, lo que resulta satisfactorio teniendo en cuenta que superar los retrasos en la comunicación de manera que el usuario identifique una imitación al instante, supone uno de los retos en este campo de investigación [27]. El fabricante ha documentado que la latencia general del traje es menor a 20 ms, la cual resulta de: el tiempo de cálculo de los sensores Neuron (<13 ms), el tiempo de cálculo de la transmisión de los datos desde el hardware (<2 ms) y el tiempo de cálculo en el software Axis Neuron (<5 ms) [17]. Sin embargo, para obtener datos experimentales se tomaron 100 registros desde la aplicación en Windows PerceptionNeuronROSserial y los datos transmitidos, en promedio se transmiten una tasa de 38,20 fps, con una

desviación estándar de 0,47 entre los datos tomados. La latencia del sistema no es percibida por el operador.

A continuación, se muestran los resultados de distintas rutinas de movimiento establecidas para verificar la imitación del robot NAO en los segmentos articulares de los brazos, de las piernas y de la cabeza.

Figura 6-1: El robot NAO imita una rutina de movimientos ejecutada por el operador no-técnico para el juego *Simón dice*, experimento no.1 en tiempo real.

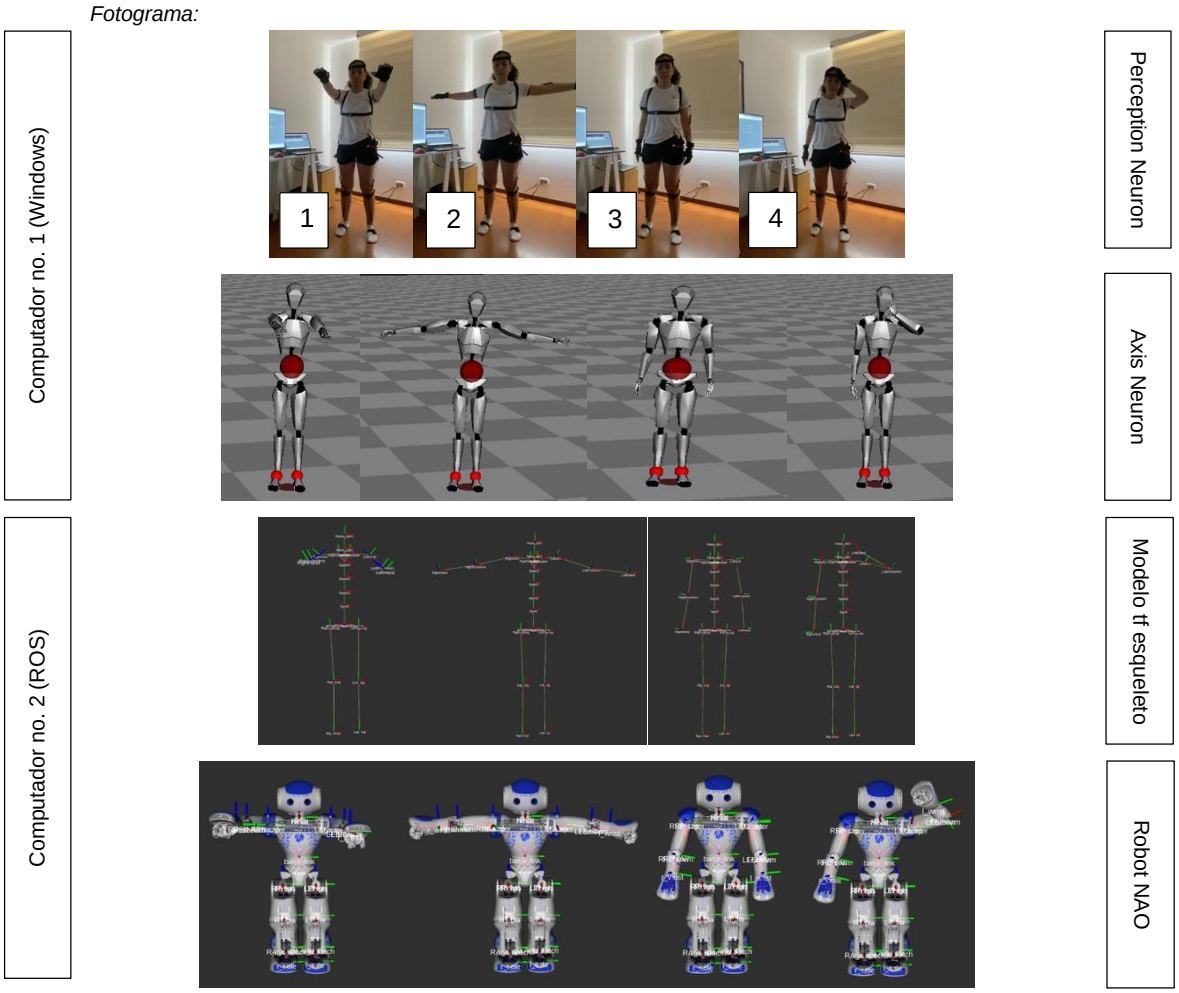


Fuente: Elaboración propia.

En la Figura **6-1**, el operador no-técnico realiza una flexión de gran amplitud del hombro derecho, retorna a su posición inicial, realiza una flexión del hombro izquierdo y vuelve a

su posición inicial. Se recuperan los datos de los ángulos del movimiento de los hombros del operador con el sistema MoCap inercial. A partir de este resultado se observa un seguimiento del movimiento de forma aceptable en el robot NAO.

Figura 6-2: El robot NAO imita una rutina de movimientos ejecutada por el operador no-técnico para el juego *Simón dice*, experimento no.2 en tiempo real.

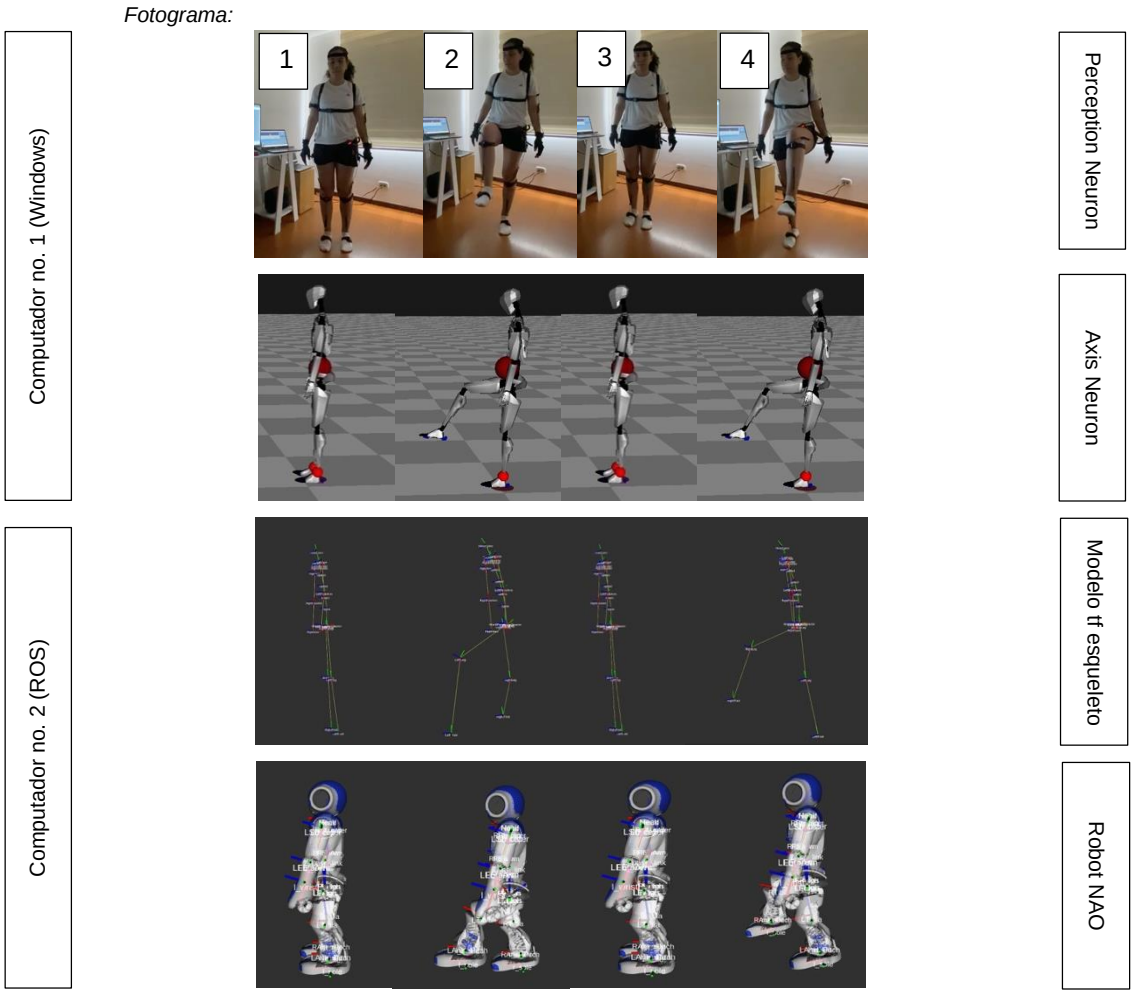


Fuente: Elaboración propia.

En la Figura **6-2**, el operador no-técnico tiene la posición inicial con los hombros flexionados a 90º, enseguida realiza una aducción de los hombros en el plano horizontal y realiza una abducción de los hombros hacia abajo desde esa posición. Luego, el usuario realiza una abducción del hombro acompañado de una flexión del codo para tocarse la cabeza. Se recuperan los datos de los ángulos con el sistema MoCap inercial. A partir de

este resultado se observa un seguimiento del movimiento de forma aceptable en el robot NAO. Aunque no hace parte de la rutina establecida en el estudio referenciado previamente [61] para la terapia de juego con los niños y niñas, los autores de esta investigación consideran importante experimentar con algunos ejercicios que impliquen el seguimiento o imitación de algunos movimientos con los segmentos articulares de la cabeza y de las piernas, para probar la estabilidad del sistema en general. La rutina establecida en la Figura **6-3** podría aplicarse a una terapia de juego para fortalecer la coordinación y el equilibrio.

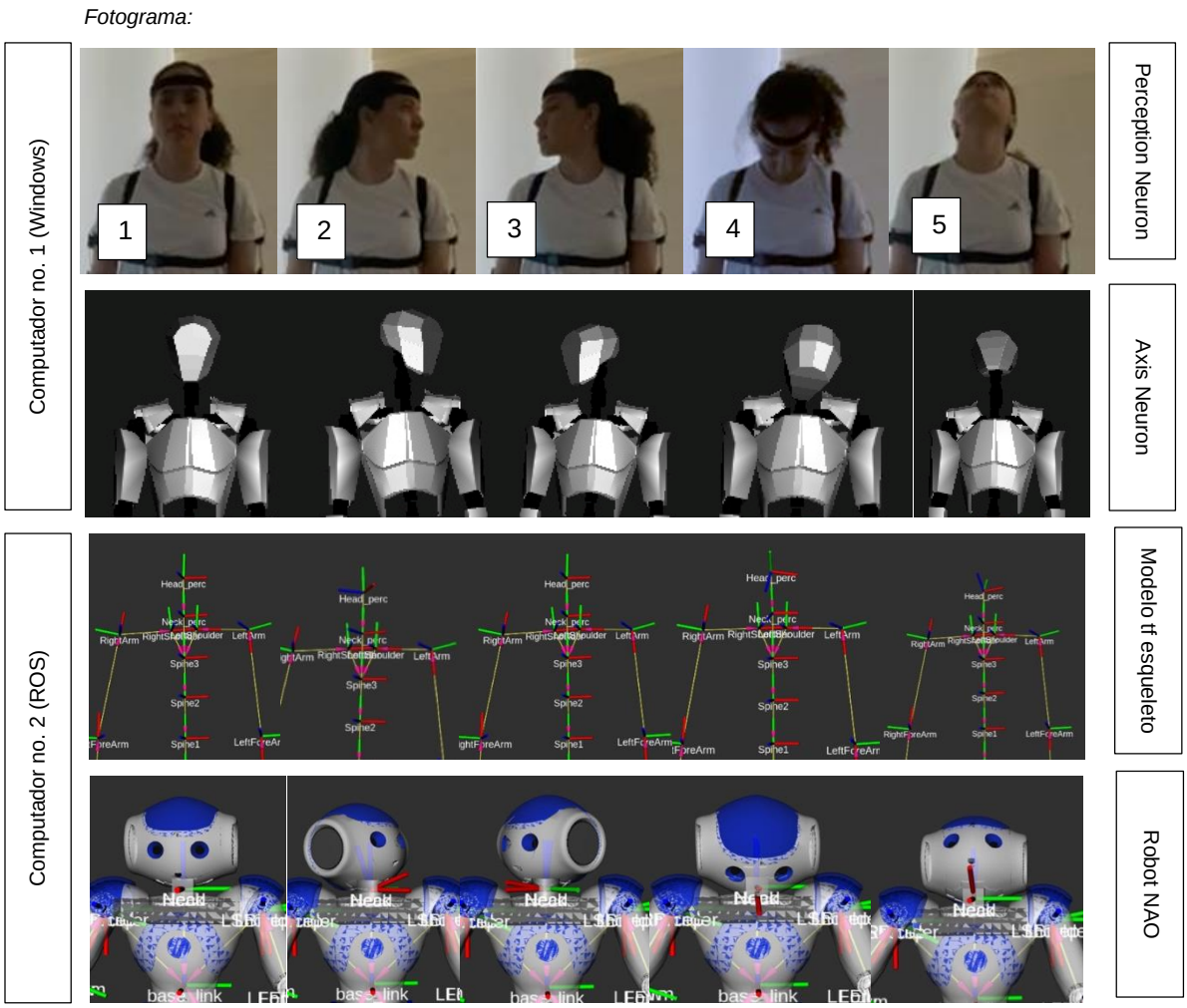
Figura 6-3: El robot NAO imita una rutina de movimientos ejecutada por el operador no-técnico, para el movimiento de las piernas, experimento no.3 en tiempo real.



Fuente: Elaboración propia.

En la Figura **6-3**, el operador no-técnico inicia con un apoyo bipodal con las rodillas en extensión y luego realiza una flexión de la rodilla derecha 90º acompañada de una flexión de la cadera, retorna a su posición inicial, y realiza una flexión de la rodilla izquierda 90º con apoyo unipodal, y vuelve a su posición inicial con las rodillas en extensión. Se recuperan los datos de los ángulos del movimiento con el sistema MoCap inercial. A partir de este resultado se observa un seguimiento del movimiento de forma aceptable en el robot NAO.

Figura 6-4: El robot NAO imita una rutina de movimientos ejecutada por el operador no-técnico, para el movimiento de la cabeza, experimento no.4 en tiempo real.



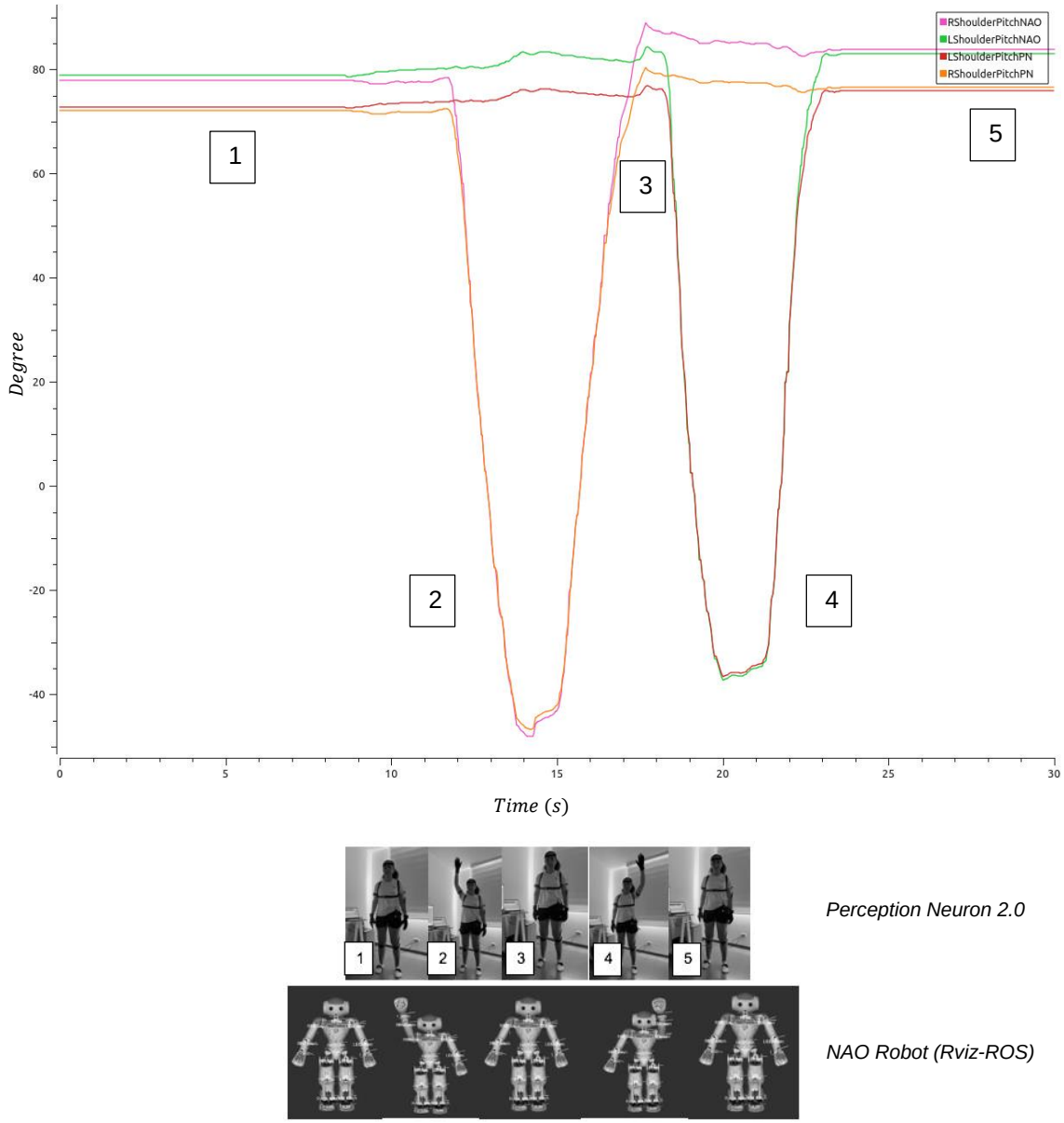
Fuente: Elaboración propia.

En la Figura **6-4**, el operador no-técnico inicia la pose con la vista hacia el frente para luego iniciar el movimiento de la cabeza con una rotación hacia la izquierda, después realiza una rotación hacia la derecha, retorna a su posición inicial y continúa con flexión y extensión del cuello. Se recuperan los datos de los ángulos del movimiento del usuario con el sistema MoCap inercial. A partir de este resultado se observa un seguimiento del movimiento de forma aceptable en el robot NAO.

En las rutinas de movimiento que se establecieron para este estudio no se incluyeron actividades que implicaban habilidades motoras finas como agarrar un objeto, ni habilidades motoras amplias que requieren la estabilización del centro de masa del robot para poder tener un apoyo monopodal y bipodal sin caídas, como caminar. En primer lugar, de acuerdo con una investigación reciente, con el cual se realizó una revisión sistemática para estudiar la aplicación de los sistemas de captura de movimiento basados en IMUs con un enfoque en rehabilitación, se encontró que 16% de las investigaciones se concentraban en el análisis de los movimientos de los brazos, y alternativamente el análisis de los movimientos finos de la mano era uno de los segmentos articulares menos estudiados con estos sistemas MoCap para rehabilitación [63]. En segundo lugar, acciones como caminar, que de hecho tiene un campo de estudio propio conocido como biomecánica de la marcha, implica el análisis de variables espacio-temporales, fuera del alcanza de este estudio por los recursos computacionales y de simulación con los que se contaba, además, calcular las posiciones del motor en el tobillo del robot es una tarea que requiere de mucha investigación o implementar algoritmos de odometría [21]. Finalmente, cabe mencionar que el visualizador Rviz de ROS únicamente considera la cinemática del robot para imitar el movimiento, y no tiene cuenta datos como la masa de un cuerpo, por lo que resulta limitado para imitar la marcha humana.

A continuación, se muestran algunos resultados de la evaluación de los ángulos articulares registrados por el robot NAO y por el operador (con Perception Neuron) tras la ejecución de las rutinas de movimiento mostradas anteriormente, con el objetivo de validar la imitación del robot NAO en diferentes segmentos articulares con el sistema de teleoperación propuesto.

Figura 6-5: Ángulos articulares resultantes de la rutina de movimientos establecida para evaluar el movimiento de los brazos, experimento no.1 en tiempo real. Secuencia de movimientos según fotogramas: (1) Posición inicial; (2) Flexión del hombro derecho; (3) Posición inicial; (4) Flexión del hombro izquierdo; (5) Posición inicial.

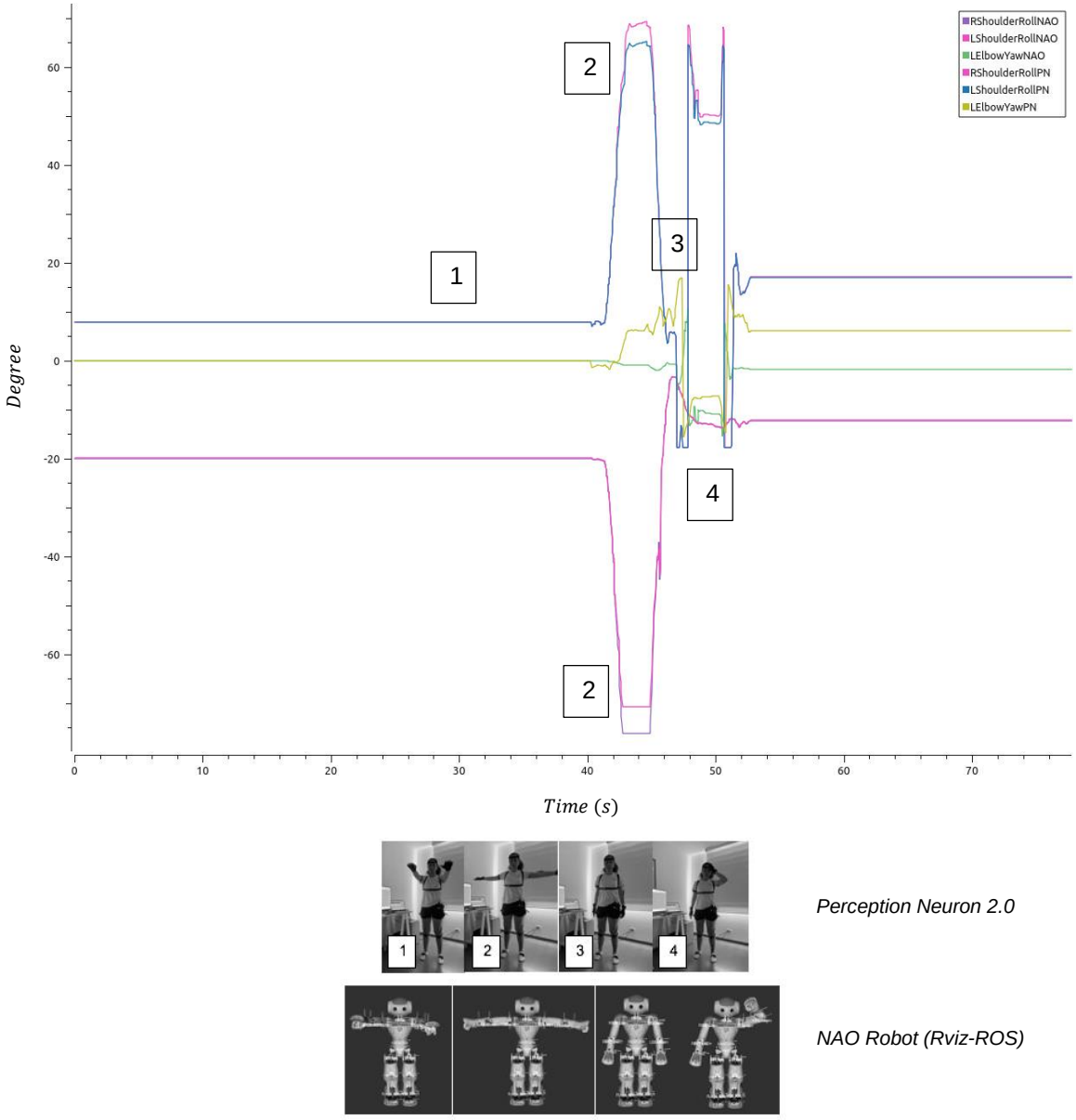


Fuente: Elaboración propia.

En el primer experimento (Figura **6-1**) se observa una serie de gráficos (Figura **6-5**) que corresponden a la imitación del robot NAO (visualizado en Rviz-ROS) del movimiento de elevación o flexión del hombro representado en el ángulo Pitch, movimiento que realiza el

operador no-técnico y que se registra con sistema MoCap Perception Neuron. El resultado es una imitación aceptable del seguimiento que realiza el robot NAO para esta rutina.

Figura 6-6: Ángulos articulares resultantes de la rutina de movimientos establecida para evaluar el movimiento de los brazos, experimento no. 2 en tiempo real. Secuencia de movimientos según fotogramas: (1) Posición inicial (hombros flexionados a 90º); (2) Aducción horizontal de los hombros; (3) Posición inicial (Abducción de los hombros); (4) Flexión del hombro izquierdo acompañado de una flexión del codo.



Fuente: Elaboración propia.

En el segundo experimento (Figura **6-2**) se observa una serie de gráficos (Figura **6-6**) que corresponden a la imitación del robot NAO (visualizado en Rviz-ROS) del movimiento aducción de los hombros en el plano horizontal y abducción de los hombres hacia abajo representado en el ángulo Roll, además de una flexión del codo izquierdo evaluado en el ángulo Yaw, movimiento acompañado de una flexión del hombro izquierdo, rutina que realiza el operador no-técnico y que se registra con el sistema MoCap Perception Neuron. El resultado es una imitación aceptable del seguimiento que realiza el robot NAO en sus articulaciones RShoulderRoll y LShoulderRoll; en la articulación LElbowYaw realiza un seguimiento continuo, y se observa una variabilidad en la estabilidad asociada a los intentos que realiza el robot para encontrar una solución a la posición que debe imitar con los brazos.

En el tercer experimento se (Figura **6-3**) observa una serie de gráficos (Figura **6-8**) que corresponden a la imitación del robot NAO (visualizado en Rviz-ROS) del movimiento flexión de la rodilla derecha e izquierda 90º, movimientos acompañados de una flexión de la cadera para permanecer en apoyo unipodal, rutina que realiza el operador no-técnico con sistema MoCap Perception Neuron. Se evalúa la imitación del robot NAO en las articulaciones RkneePitch y LkneePitch, se observa una imitación del movimiento aceptable, y se aclara que la variación que se observa de los datos durante la flexión de la rodilla derecha se debe a que durante la experimentación desde Axis Neuron ocurrió una perturbación de la transmisión de datos, con pérdida del pie de apoyo, sin embargo, esto no implica que el robot NAO no haya imitado adecuadamente la información transmitida desde el Perception Neuron.

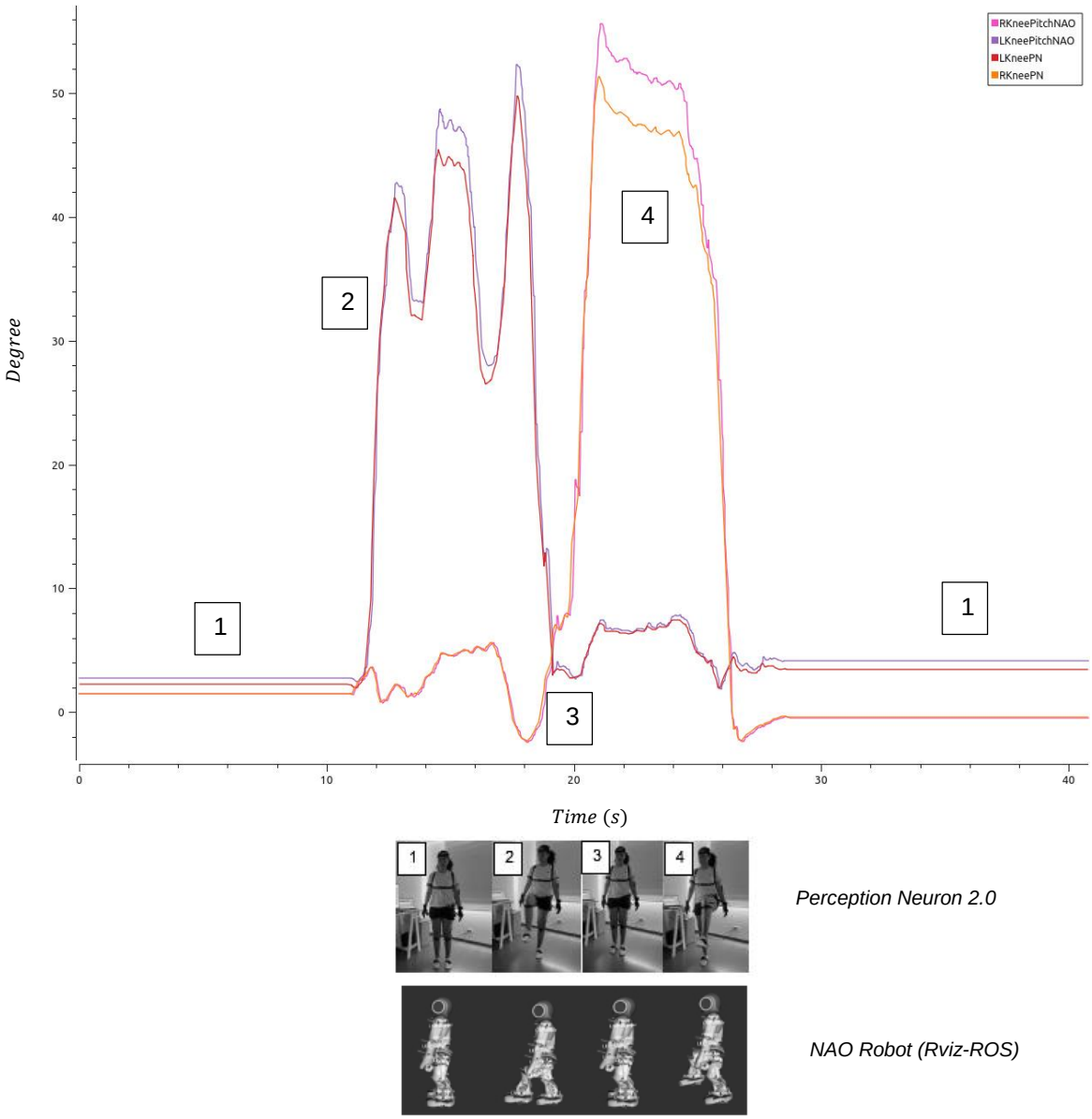
Figura 6-7: Dos escenarios para el uso del sistema de teleoperación del robot de asistencia social, en la práctica.

(a) Teleoperación del robot por la persona que requiere asistencia.

(b) Teleoperación del robot por un usuario no-técnico.

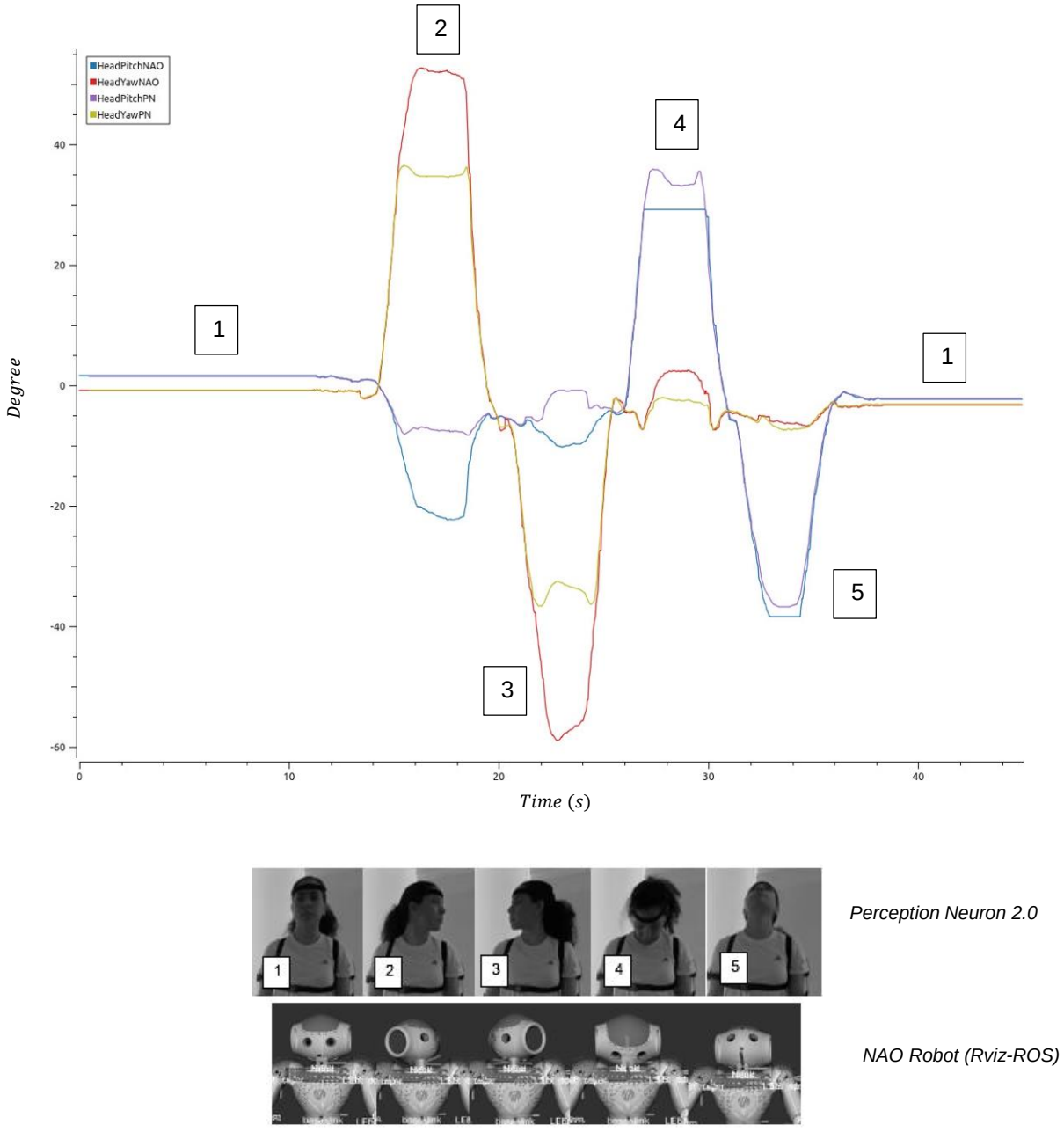
Fuente: Elaboración propia.

Figura 6-8: Ángulos articulares resultantes de la rutina de movimientos establecida para evaluar el movimiento de las piernas, experimento no.3 en tiempo real. Secuencia de movimientos según fotogramas: (1) Posición inicial (apoyo bipodal); (2) Flexión de la rodilla derecha (apoyo unipodal); (3) Posición inicial (apoyo bipodal); (4) Flexión de la rodilla izquierda (apoyo unipodal).



Fuente: Elaboración propia.

Figura 6-9: Ángulos articulares resultantes de la rutina de movimientos establecida para evaluar el movimiento de la cabeza, experimento no. 4 en tiempo real. Secuencia de movimientos según fotogramas: (1) Posición inicial; (2) Rotación izquierda del cuello; (3) Rotación derecha del cuello; (4) Flexión del cuello; (5) Extensión del cuello.



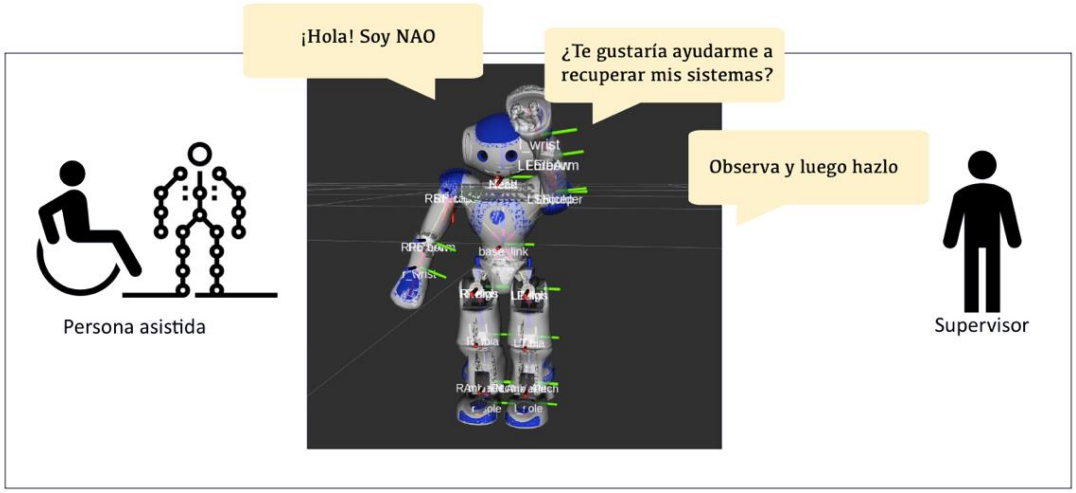
Fuente: Elaboración propia.

En el último experimento (Figura **6-4**) se observa una serie de gráficos (Figura **6-9**) que corresponden a la imitación del robot NAO (visualizado en Rviz-ROS) del movimiento de rotación hacia la derecha e izquierda de la cabeza y, flexión y extensión del cuello, rutina que realiza el operador no-técnico y que se registra con el sistema MoCap Perception Neuron. El resultado es una imitación aceptable del seguimiento que realiza el robot NAO para esta secuencia.

En términos generales, se obtienen resultados aceptables para la teleoperación del robot de asistencia social NAO (visualizado en Rviz-ROS) con un sistema de captura de movimiento inercial, y se espera que a futuro pueda aplicarse a estudios de investigación en rehabilitación. Teniendo en cuenta además que los movimientos pueden ser grabados y guardados en Axis Neuron, y, por lo tanto, pueden ser reproducidos en cualquier momento. Tal como se evidencia en [42], el uso de robots SAR ha demostrado efectos positivos en niños y niñas con trastornos del movimiento, equilibrio y postura, este estudio también evidencia que las investigaciones siguen siendo pocas para determinar el verdadero potencial de los SAR, dicho potencial debe ser aprovechado por ingenieros y personal de salud para que el intercambio de conocimientos favorezca el desarrollo de estas soluciones desde el ámbito ingenieril a los escenarios clínicos.

Para terminar, y como se revela en diversas investigaciones [63] la rehabilitación tradicional *"carece de estándares cuantitativos y sistemáticos"* por lo cual, según este mismo estudio la nueva rehabilitación debe combinar la rehabilitación convencional con equipos avanzados, como los sistemas de captura de movimiento. Con base en lo anterior, se realiza una última experimentación con el objetivo de evidenciar la comprensión de una instrucción y el seguimiento de una secuencia de poses por parte de una persona "que requiere asistencia", en adelante PA; específicamente el robot NAO hace las veces de un instructor, ejecuta una secuencia de movimientos pregrabada y posteriormente la PA debe intentar imitar la misma secuencia de poses (Figura **6-10**), en una sola oportunidad. Para monitorear y analizar la tendencia del movimiento imitado por la PA, se utiliza el traje MoCap Perception Neuron 2.0, el cual también fue utilizado para grabar la rutina en el robot NAO.

Figura 6-10: Escenario de imitación y monitoreo de una secuencia de movimientos (Figura **6-1**) realizados por el robot NAO, por la persona que requiere asistencia.



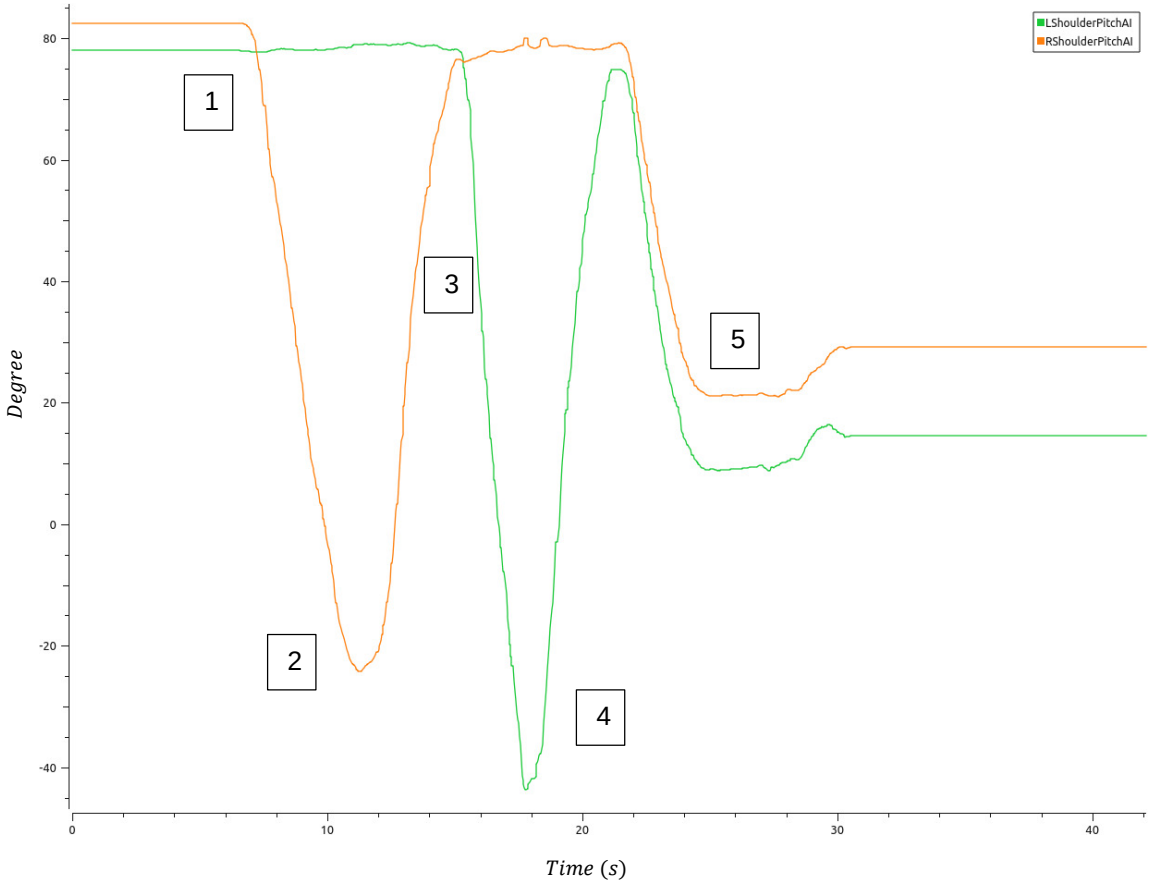
Fuente: Elaboración propia.

De esta experimentación se obtuvo el ángulo del movimiento articular máximo alcanzado por la PA al realizar una flexión del hombro derecho (FdHD) y posteriormente una flexión hombro izquierdo (FdHI), es decir una elevación de los brazos alternando ambos miembro superiores (Figura **6-11**), desde el punto de vista terapéutico el objetivo de esta rutina podría trazarse sobre la meta de alcanzar un objeto por ejemplo, lo que a su vez se relacionaría indirectamente con alguna actividad de la vida diaria. Ahora bien, si analizamos comparativamente la Figura **6-5** y la Figura **6-11**, observamos que la tendencia del movimiento es similar, aunque, el ángulo máximo aproximado alcanzado por la PA fue de -25º (FdHD) y -45º (FdHI), y, por el Instructor es -37º (FdHD) y -47º (FdHI), evidentemente diferente, lo cual corresponde a lo esperado. Pues en una etapa temprana de rehabilitación es factible que la PA no realice con precisión el movimiento, de hecho, es posible que nunca lo logre pero que cada vez más se acerque a la tendencia, sesión tras sesión, en la medida en que se recupera.

También, se realiza esta experimentación para evaluar la tendencia del movimiento imitado por la PA con los miembros inferiores como se observa en la Figura **6-12**, la PA comienza la rutina con un apoyo bipodal, posteriormente realiza una flexión de 50º de la rodilla izquierda (FdRI) (ver Figura **6-13**), vuelve a su posición inicial, y realiza una flexión de 58ºde la rodilla derecha (FdRD) (Figura **6-13**), un ejercicio que podría estar asociado a una

rutina para favorecer la estabilidad y coordinación de la PA. Si lo comparamos con los ángulos articulares máximos alcanzados por el Instructor se observa una diferencia de 2º (FdRI) y de -2º (FdRI), por lo tanto, una similitud del 96%. Adicionalmente, no se registra perturbación en el sistema cuando la PA realiza el apoyo unipodal sobre el pie derecho y posteriormente realiza un movimiento de flexión de la pelvis acompañado de una flexión la rodilla izquierda (Figura **6-13**). También es notable y esperado que los movimientos articulares máximos ejecutados por la PA no se logren en los mismos instantes de tiempo en que son ejecutados por el Instructor.

Figura 6-11: Ángulos articulares resultantes de la secuencia de poses realizada por la PA quien imitó la instrucción dada por el robot NAO. Secuencia de poses: (1) Posición inicial; (2) Flexión del hombro derecho; (3) Posición inicial; (4) Flexión del hombro izquierdo; (5) Aducción horizontal de los hombros.



Fuente: Elaboración propia.

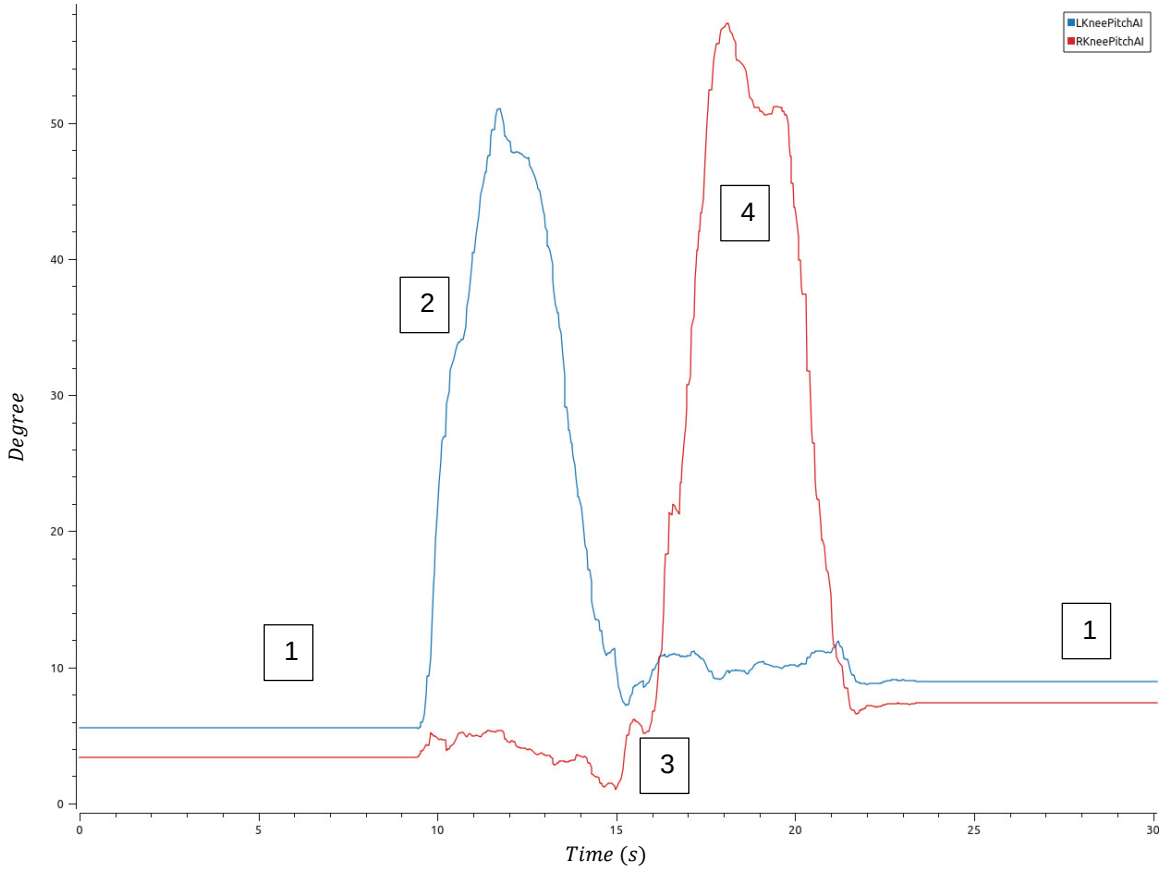
Figura 6-12: Fotografía. Escenario de imitación y monitoreo de una secuencia de movimientos realizados por el robot NAO (Figura **6-3**), por la persona que requiere asistencia (PA).

Fuente: Elaboración propia. Registro fotográfico de octubre 2023.

Dicho esto, los resultados, aunque preliminares, pueden ayudar a avizorar el potencial de uso de estas plataformas en la práctica clínica (Ver Figura **6-12**), teniendo en cuenta los siguientes aspectos: (1) La imitación del movimiento fue aceptable en las cadenas articulares evaluadas para una secuencia de movimientos establecida, con una latencia poco significativa; (2) Se pueden realizar diferentes secuencias de movimientos (con algunas restricciones), de modo tal que una rutina de terapia podría diseñarse de forma individualizada y adaptativa en la medida en que avanzan las sesiones; (3) La posibilidad

de tener rutinas previamente grabadas por un instructor disminuye la repetición constante del movimiento en una sesión de terapia, lo que puede contribuir a disminuir la sobrecarga del terapeuta (Ver Figura **6-7 (b)**); (4) La utilización del traje de captura de movimiento para teleoperar el robot humanoide NAO (avatar) no requiere una capacitación compleja, y, (5) el traje MoCap puede ser utilizado por la persona que requiere asistencia y por lo tanto sería posible monitorear el avance sesión tras sesión (Ver Figura **6-7 (a)**).

Figura 6-13: Ángulos articulares resultantes de la secuencia de poses realizada por la PA quien imitó la instrucción dada por el robot NAO. Secuencia de poses: Secuencia de movimientos (1) Posición inicial (apoyo bipodal); (2) Flexión de la rodilla derecha (apoyo unipodal); (3) Posición inicial (apoyo bipodal); (4) Flexión de la rodilla izquierda (apoyo unipodal).



Fuente: Elaboración propia.

7. Conclusiones y recomendaciones

7.1 Conclusiones

Con este trabajo de investigación se puede concluir que se alcanzaron los objetivos planteados, los cuales en términos generales consistían en el diseño de un método que permitiera la captura, registro, y transmisión de datos de movimiento a plataformas de rehabilitación robóticas como el robot de asistencia social NAO, en un entorno virtual. En este sentido, el sistema MoCap seleccionado se ajusta a los requerimientos de esta investigación y a su proyección de aplicación en el área de la rehabilitación, en un futuro; además permite el registro y evaluación del progreso terapéutico con estándares cuantitativos. De la misma forma, se puede personalizar la dimensión corporal, grabar movimientos sin limitaciones de espacio, en ambientes interiores o exteriores bajo cualquier condición de iluminación y sin la necesidad de cámaras ópticas, y, permite exportar en tiempo real este flujo de datos a otros programas. También, se realizó la recepción y transmisión de los datos del modelo del esqueleto humano capturados por el Perception Neuron 2.0 a ROS, con base en lo desarrollado previamente por otros investigadores, lo que brinda la posibilidad de diseñar una variedad de aplicaciones con distintos enfoques en los que se requiera la integración de sistemas de captura de movimiento basados en IMUs y sistemas robóticos. Finalmente, los experimentos de teleoperación en tiempo real y con movimientos pregrabados se realizaron con éxito, se visualizaron y analizaron diferentes rutinas y poses que fueron imitadas adecuadamente por el robot NAO visualizado en Rviz-ROS, aclarando que este sistema no imita el desplazamiento del robot en un espacio como la marcha humana, ni imita movimientos que impliquen habilidades motoras finas. El sistema tiene una latencia poco significativa incluso cuando el operador se ubica a 10 m del computador que captura la información. Finalmente, la solución propuesta permite que el robot NAO (avatar) ejecute una secuencia de poses individualizada y adaptativa a partir de la instrucción dada por un usuario no-técnico, por lo que no se requiere de personal especializado para operar el robot.

7.2 Recomendaciones

Establecidas las conclusiones de esta investigación se recomienda:

- Realizar pruebas de teleoperación con movimientos pregrabados de una rutina individualizada establecida por un terapeuta o educador con el objetivo de validar la facilidad de operación por parte de un usuario no-técnico, para conocer la percepción de aplicabilidad como herramienta coadyuvante en un contexto de rehabilitación.
- Realizar mejoras a la solución aquí propuesta para la teleoperación del robot NAO visualizada en Rviz-ROS, que permite el seguimiento del movimiento y representa un primer nivel de autonomía, adicionar una interfaz que posibilite una experiencia inmersiva en la visualización del entorno, la grabación de texto o de audio y la reproducción de sonidos, con el fin que el robot tenga características sociales en medio de una interacción (ver, oír, hablar). También, será necesario realizar estudios sobre la cada cinemática del tronco, para que por ejemplo el robot pueda realizar una flexión, extensión o rotación del torso, e incluso proponer soluciones para que el robot pueda desplazarse en un entorno. Es importante mencionar que aplicar el resultado de esta investigación a un robot NAO real probablemente implicará ajustes complejos para lograr por ejemplo estabilizar el centro de masa del robot y evitar caídas en la ejecución de un ciclo de marcha que implica necesariamente un apoyo monopodal, lo que por supuesto puede ser un trabajo a futuro.
- Utilizar el traje de captura de movimiento Perception Neuron 2.0 como una herramienta de monitorización y retroalimentación para el paciente o usuario que interactúa con un robot NAO (co-terapeuta), con el fin de poder identificar movimientos ejecutados incorrectamente y realizar enseguida un refuerzo positivo, lo que significa sin duda una fase avanzada de esta investigación en un escenario clínico.
- Realizar estudios complementarios que permitan tener una comparación del rendimiento de los sistemas MoCap: Kinect y Perception Neuron, para llegar a soluciones que permitan por ejemplo combinar los dos hardware para el contexto que aquí se propone u otras aplicaciones.

Finalmente, para fortalecer este tipo de investigaciones es necesario enfatizar en la importancia del diseño centrado en el usuario, así como en la necesidad de que diversas especialidades y campos de estudio estén involucrados durante el ciclo de diseño general para garantizar el éxito de su aplicabilidad en un contexto terapéutico o educativo.

A. Anexo: Instrucciones para la ejecución del programa

En primer lugar, deberá instalar en un computador con Windows:

- El software Axis Neuron, propio del sistema MoCap Perception Neuron 2.0 *Axis_Neuron_x64_3_8_42_8591_20180411034619617000_X64* disponible en el siguiente enlace: https://neuronmocap.com/pages/axis-studio
- Aplicación *PerceptionNeuronROSserial.exe* disponible en el repositorio GitHub en el siguiente enlace: https://github.com/smhaller/perception-neuron-ros
- Aplicación *PerceptionNeuronROSserial.exe* con mejoras disponible en el repositorio GitHub en el siguiente enlace: https://github.com/blutjens/perc_neuron_ros_ur10.

En otro computador con ROS, deberá instalar las herramientas de simulación del robot NAO:

- *nao_control* disponible en el repositorio GitHub en el siguiente enlace: https://github.com/ros-naoqi/nao_virtual/tree/master/nao_gazebo_plugin
- *nao_moveit_config* disponible en el repositorio GitHub en el siguiente enlace: https://github.com/ros-naoqi/nao_moveit_config
- *nao_gazebo* disponible en el repositorio GitHub en el siguiente enlace: https://github.com/costashatz/nao_gazebo

Para explorar el paquete de ROS *rosserial*, que permite la trasmisión de los datos de Windows a ROS, puede remitirse a:

- *ros-drivers* disponible el repositorio de GitHub en el siguiente enlace: https://github.com/ros-drivers/rosserial

NOTA: Para comunicar el computador con el sistema operativo de Windows y el computador con Linux (ROS) hágalo directamente con un cable de ethernet.
A continuación, se anexan las instrucciones para la ejecución del código en ROS:

```
# Thesis - Design of a movement capture method for the registration,
analysis and transmission of data to robotic rehabilitation
platforms, in a virtual enviroment

## Carolina Pimentel (cpimente@unal.edu.co), Dr. Luis Miguel Mendez
Moreno, Dr. Diego Alexander Garzon Alvarado

## Universidad Nacional de Colombia, Bogota

# Move Nao Py

## Using the movement script with RViz (display application)

To launch the mimic movement application with RViz follow the next
steps.

1. Open a terminal in Linux and execute the command "roscore"

2. Open a second terminal in Linux and source the devel/setup.bash
of the workspace. Then execute the command "roslaunch move_nao_py
launch_serial.launch"

3. Open the Axis Neuron program in Windows and launch the
"PerceptionNeuronROSserial.exe" program.

4. Open a new terminal in Linux  and source the devel/setup.bash of
the workspace. Then execute the command "roslaunch move_nao_py
display_nao_rviz.launch"

5.  Open a new terminal in Linux  and source the devel/setup.bash of
the workspace. Then execute the command "rosrun move_nao_py
move_naojoints_rviz.py"
```

En este documento, se reconoce el trabajo previo y aporte del Dr. Björn Lütjens y del Dr. Emmanuel Carlos Dean-Leon titulado *Real-Time Teleoperation of Industrial Robots with the Motion Capture System Perception Neuron*, el cual sentó las bases para el desarrollo de este proyecto de investigación, su desarrollo está disponible en el repositorio GitHub en el siguiente enlace: https://github.com/blutjens/perc_neuron_ros_ur10.

B. Anexo: Código fuente paquete *move_nao_py en ROS*

MoveNaoJoints.py

```
1  #!/usr/bin/python3
2
3  import rospy
4  import math
5  import tf
6  from tf.transformations import euler_from_quaternion, quaternion_matrix
7  import geometry_msgs.msg
8  import sensor_msgs.msg
9  import angles
10
11 import numpy as np
12 import roboticstoolbox as rtb
13 from spatialmath import *
14 import spatialmath.base as base
15 from math import pi
16
17
18 class MoveNaoJoints:
19
20   def __init__(self) -> None:
21
22     #-- Create a TransformListener object
23     self.listener = tf.TransformListener()
24
25     #-- Create publisher for joint_states
26     self.pub_joint_states = rospy.Publisher('joint_states', sensor_msgs.msg.JointState, queue_size=10)
27
28     #-- Do a transform lookup at a constant rate
29     self.rate = rospy.Rate(60.0) # 60hz
```

```
30
31      #-- Variables to store the joint angles
32
33      self.joints_prev_left = [0.0, 0.0, 0.0 ,0.0, 0.0]
34      self.joints_prev_right = [0.0, 0.0, 0.0 ,0.0, 0.0]
35
36      #- Left leg
37      self.nao_LHipPitch = 0.0
38      self.nao_LHipRoll = 0.0
39      self.nao_LKneePitch = 0.0
40      self.nao_LAnklePitch = 0.0
41      self.nao_LAnkleRoll = 0.0
42
43      #- Right leg
44      self.nao_RHipPitch = 0.0
45      self.nao_RHipRoll = 0.0
46      self.nao_RKneePitch = 0.0
47      self.nao_RAnklePitch = 0.0
48      self.nao_RAnkleRoll = 0.0
49
50      #- Left arm
51      self.nao_LShoulderPitch = 0.0
52      self.nao_LShoulderRoll = 0.0
53      self.nao_LElbowYaw = 0.0
54      self.nao_LElbowRoll = 0.0
55      self.nao_LWristYaw = 0.0
56
57      #- Right arm
58      self.nao_RShoulderPitch = 0.0
59      self.nao_RShoulderRoll = 0.0
60      self.nao_RElbowYaw = 0.0
61      self.nao_RElbowRoll = 0.0
62      self.nao_RWristYaw = 0.0
63
64      #- Head
65      self.nao_HeadYaw = 0.0
66      self.nao_HeadPitch = 0.0
67
68
69  def getJointStates_Left_leg(self):
70
71      #-- Lookup transform from Hips to Left leg
```

```
72      try:
73          (_,rot_Hip_LeftLeg) = self.listener.lookupTransform('/LeftUpLeg', '/Hips', rospy.Time(0))
74          #-- Pass quaternion to euler angles
75          rot_Hip_LeftLeg = euler_from_quaternion(rot_Hip_LeftLeg)
76          #-- Get the roll and pitch
77          nao_LHipPitch = rot_Hip_LeftLeg[0]
78          nao_LHipPitch = nao_LHipPitch*-1 #-- Invert the sign
79
80          nao_LHipRoll = rot_Hip_LeftLeg[2]
81          nao_LHipRoll = nao_LHipRoll*-1 #-- Invert the sign
82
83          #-- Limit the angle to the range of motion of the robot
84          if nao_LHipPitch < -1.54:
85              nao_LHipPitch = -1.54
86          if nao_LHipPitch > 0.48:
87              nao_LHipPitch = 0.48
88
89          if nao_LHipRoll < -0.38:
90              nao_LHipRoll = -0.38
91          if nao_LHipRoll > 0.79:
92              nao_LHipRoll = 0.79
93
94          self.nao_LHipPitch = nao_LHipPitch
95          self.nao_LHipRoll = nao_LHipRoll
96
97          #-- Print the values
98          # rospy.loginfo("nao_LHipPitch: %f  nao_LHipRoll: %f", nao_LHipPitch, nao_LHipRoll)
99
100     except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
101         pass
102
103     #-- Lookup transform from Left leg to Knee
104     try:
105         (_,rot_LeftLeg_Knee) = self.listener.lookupTransform('/LeftLeg', '/LeftUpLeg', rospy.Time(0))
106         #-- Pass quaternion to euler angles
107         rot_LeftLeg_Knee = euler_from_quaternion(rot_LeftLeg_Knee)
108         #-- Get the pitch
109         nao_LKneePitch = rot_LeftLeg_Knee[0]
110         nao_LKneePitch = nao_LKneePitch*-1 # Invert the angle
111
112         #-- Limit the angle to the range of motion of the robot
```

```
113         if nao_LKneePitch < -0.09:
114             nao_LKneePitch = -0.09
115         if nao_LKneePitch > 2.11:
116             nao_LKneePitch = 2.11
117 
118         self.nao_LKneePitch = nao_LKneePitch
119 
120         #-- Print the values
121         # rospy.loginfo("nao_LKneePitch: %f", nao_LKneePitch)
122 
123     except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
124         pass
125 
126     #-- Lookup transform from Left Knee to Left Foot
127     try:
128         (_, rot_LeftKnee_Foot) = self.listener.lookupTransform('/LeftFoot', '/LeftLeg', rospy.Time(0))
129         #-- Pass quaternion to euler angles
130         rot_LeftKnee_Foot = euler_from_quaternion(rot_LeftKnee_Foot)
131         #-- Get the roll and pitch
132         nao_LAnklePitch = rot_LeftKnee_Foot[0]
133         nao_LAnklePitch = nao_LAnklePitch*-1 # Invert the angle
134 
135         nao_LAnkleRoll = rot_LeftKnee_Foot[2]
136         nao_LAnkleRoll = nao_LAnkleRoll*-1 # Invert the angle
137 
138         #-- Limit the angle to the range of motion of the robot
139         if nao_LAnklePitch < -1.19:
140             nao_LAnklePitch = -1.19
141         if nao_LAnklePitch > 0.92:
142             nao_LAnklePitch = 0.92
143 
144         if nao_LAnkleRoll < -0.40:
145             nao_LAnkleRoll = -0.40
146         if nao_LAnkleRoll > 0.77:
147             nao_LAnkleRoll = 0.77
148 
149         self.nao_LAnklePitch = nao_LAnklePitch
151 
152         #-- Print the values
153         # rospy.loginfo("nao_LAnklePitch: %f  nao_LAnkleRoll: %f", nao_LAnklePitch, nao_LAnkleRoll)
154 
```

```
155     except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
156         pass
157 
158 def getJointStates_Right_leg(self):
159 
160     #-- Lookup transform from Hips to Right leg
161     try:
162         (_,rot_Hip_RightLeg) = self.listener.lookupTransform('/RightUpLeg', '/Hips', rospy.Time(0))
163         #-- Pass quaternion to euler angles
164         rot_Hip_RightLeg = euler_from_quaternion(rot_Hip_RightLeg)
165         #-- Get the roll and pitch
166         nao_RHipPitch = rot_Hip_RightLeg[0]
167         nao_RHipPitch = nao_RHipPitch*-1 #-- Invert the sign
168 
169         nao_RHipRoll = rot_Hip_RightLeg[2]
170         nao_RHipRoll = nao_RHipRoll*-1 #-- Invert the sign
171 
172         #-- Limit the angle to the range of motion of the robot
173         if nao_RHipPitch < -1.54:
174             nao_RHipPitch = -1.54
175         if nao_RHipPitch > 0.48:
176             nao_RHipPitch = 0.48
177 
178         if nao_RHipRoll < -0.79:
179             nao_RHipRoll = -0.79
180         if nao_RHipRoll > 0.38:
181             nao_RHipRoll = 0.38
182 
183         self.nao_RHipPitch = nao_RHipPitch
184         self.nao_RHipRoll = nao_RHipRoll
185 
186         #-- Print the values
187         # rospy.loginfo("nao_RHipPitch: %f  nao_RHipRoll: %f", nao_RHipPitch, nao_RHipRoll)
188 
189     except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
190         pass
191 
192     #-- Lookup transform from Right leg to Knee
193     try:
194         (_,rot_RightLeg_Knee) = self.listener.lookupTransform('/RightLeg', '/RightUpLeg', rospy.Time(0))
195         #-- Pass quaternion to euler angles
```

```
196         rot_RightLeg_Knee = euler_from_quaternion(rot_RightLeg_Knee)
197         #-- Get the pitch
198         nao_RKneePitch = rot_RightLeg_Knee[0]
199         nao_RKneePitch = nao_RKneePitch*-1 # Invert the angle
200
201         #-- Limit the angle to the range of motion of the robot
202         if nao_RKneePitch < -0.09:
203             nao_RKneePitch = -0.09
204         if nao_RKneePitch > 2.11:
205             nao_RKneePitch = 2.11
206
207         self.nao_RKneePitch = nao_RKneePitch
208
209         #-- Print the values
210         # rospy.loginfo("nao_RKneePitch: %f", nao_RKneePitch)
211
212     except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
213         pass
214
215     #-- Lookup transform from Right Knee to Right Foot
216     try:
217         (_, rot_RightKnee_Foot) = self.listener.lookupTransform('/RightFoot', '/RightLeg', rospy.Time(0))
218         #-- Pass quaternion to euler angles
219         rot_RightKnee_Foot = euler_from_quaternion(rot_RightKnee_Foot)
220         #-- Get the roll and pitch
221         nao_RAnklePitch = rot_RightKnee_Foot[0]
222         nao_RAnklePitch = nao_RAnklePitch*-1 # Invert the angle
223
224         nao_RAnkleRoll = rot_RightKnee_Foot[2]
225         nao_RAnkleRoll = nao_RAnkleRoll*-1 # Invert the angle
226
227         #-- Limit the angle to the range of motion of the robot
228         if nao_RAnklePitch < -1.19:
229             nao_RAnklePitch = -1.19
230         if nao_RAnklePitch > 0.92:
231             nao_RAnklePitch = 0.92
232
234             nao_RAnkleRoll = -0.77
235         if nao_RAnkleRoll > 0.40:
236             nao_RAnkleRoll = 0.40
237
```

```
238         self.nao_RAnklePitch = nao_RAnklePitch
239         self.nao_RAnkleRoll = nao_RAnkleRoll
240 
241         #-- Print the values
242         # rospy.loginfo("nao_RAnklePitch: %f  nao_RAnkleRoll: %f", nao_RAnklePitch, nao_RAnkleRoll)
243 
244       except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
245         pass
246 
247     def getJointStates_Left_arm_decoupled(self):
248 
249       #-- Lookup transform from Left Shoulder to Left Arm
250       try:
251         (_,rot_LeftShoulder_Arm) = self.listener.lookupTransform('/LeftShoulder', '/LeftArm',
252 rospy.Time(0))
253         #-- Pass quaternion to euler angles
254         rot_LeftShoulder_Arm = euler_from_quaternion(rot_LeftShoulder_Arm)
255         rospy.logwarn("Angles \t %f \t %f \t %f",
256 rot_LeftShoulder_Arm[0],rot_LeftShoulder_Arm[1],rot_LeftShoulder_Arm[2])
257 
258         #-- Get the roll and pitch
259         offset_LShoulderPitch = 1.58
260         nao_LShoulderPitch = (rot_LeftShoulder_Arm[0]/1.58)*3.14 # Scale movement
261         nao_LShoulderPitch = -1.0*nao_LShoulderPitch + offset_LShoulderPitch
262 
263         offset_LShoulderRoll = 1.83
264         nao_LShoulderRoll = (rot_LeftShoulder_Arm[2]/0.8)*1.64 # Scale movement
265         nao_LShoulderRoll = -1.0*nao_LShoulderRoll + offset_LShoulderRoll
266 
267         #-- Limit the angle to the range of motion of the robot
268         if nao_LShoulderPitch < -2.09:
269           nao_LShoulderPitch = -2.09
270         if nao_LShoulderPitch > 2.09:
271           nao_LShoulderPitch = 2.09
272 
273         if nao_LShoulderRoll < -0.31:
274           nao_LShoulderRoll = -0.31
275         if nao_LShoulderRoll > 1.33:
276           nao_LShoulderRoll = 1.33
277 
278         self.nao_LShoulderPitch = nao_LShoulderPitch
```

```
279         self.nao_LShoulderRoll = nao_LShoulderRoll
280
281         #-- Print the values
282         rospy.loginfo("nao_LShoulderPitch: %f  nao_LShoulderRoll: %f", nao_LShoulderPitch,
283 nao_LShoulderRoll)
284
285       except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
286         pass
287
288       #-- Lookup transform from Left Arm to Left Elbow
289       try:
290         (_,rot_LeftArm_Forearm) = self.listener.lookupTransform('/LeftArm', '/LeftForeArm',
291 rospy.Time(0))
292         #-- Pass quaternion to euler angles
293         rot_LeftArm_Forearm = euler_from_quaternion(rot_LeftArm_Forearm)
294
295         rospy.logwarn("Angles \t %f \t %f \t %f",
296 rot_LeftArm_Forearm[0],rot_LeftArm_Forearm[1],rot_LeftArm_Forearm[2])
297
298         #-- Get the yaw and roll
299         nao_LElbowYaw = 0.0 # rot_LeftArm_Forearm[0]
300         # nao_LElbowYaw = nao_LElbowYaw*-1 # Invert the angle
301
302         offset_LElbowRoll = 0.26
303         nao_LElbowRoll = rot_LeftArm_Forearm[2] + offset_LElbowRoll
304
305         #-- Limit the angle to the range of motion of the robot
306         if nao_LElbowYaw < -2.09:
307           nao_LElbowYaw = -2.09
308         if nao_LElbowYaw > 2.09:
309           nao_LElbowYaw = 2.09
310
311         if nao_LElbowRoll < -1.54:
312           nao_LElbowRoll = -1.54
313         if nao_LElbowRoll > 0.00:
314           nao_LElbowRoll = 0.0
315


317         self.nao_LElbowRoll = nao_LElbowRoll
318
319         #-- Print the values
320         rospy.loginfo("nao_LElbowYaw: %f  nao_LElbowRoll: %f", nao_LElbowYaw, nao_LElbowRoll)
```

```
321
322         except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
323             pass
324
325         #-- Lookup transform from Left Forearm to Left Wrist
326         try:
327             (_, rot_LeftForearm_Wrist) = self.listener.lookupTransform('/LeftForeArm', '/LeftHand',
328 rospy.Time(0))
329             #-- Pass quaternion to euler angles
330             rot_LeftForearm_Wrist = euler_from_quaternion(rot_LeftForearm_Wrist)
331             #-- Get the yaw
332             nao_LWristYaw = rot_LeftForearm_Wrist[0]
333             nao_LWristYaw = nao_LWristYaw*-1 # Invert the angle
334
335
336             #-- Limit the angle to the range of motion of the robot
337             if nao_LWristYaw < -1.82:
338                 nao_LWristYaw = -1.82
339             if nao_LWristYaw > 1.82:
340                 nao_LWristYaw = 1.82
341
342             self.nao_LWristYaw = nao_LWristYaw
343
344             #-- Print the values
345             rospy.loginfo("nao_LWristYaw: %f", nao_LWristYaw)
346
347         except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
348             pass
349
350
351     def getJointStates_Right_arm_decoupled(self):
352
353         #-- Lookup transform from Right Shoulder to Right Arm
354         try:
355             (_,rot_RightShoulder_Arm) = self.listener.lookupTransform('/RightArm', '/RightShoulder',
356 rospy.Time(0))
357             #-- Pass quaternion to euler angles
358             rot_RightShoulder_Arm = euler_from_quaternion(rot_RightShoulder_Arm)
359             rospy.loginfo("rot_LeftShoulder_Arm: Roll %f, Pitch %f, Yaw %f", rot_RightShoulder_Arm[0],
360 rot_RightShoulder_Arm[1], rot_RightShoulder_Arm[2])
361             #-- Get the roll and pitch
```

```
362         offset_RShoulderPitch = 0.63
363         nao_RShoulderPitch = -rot_RightShoulder_Arm[0] + offset_RShoulderPitch
364
365         nao_RShoulderRoll = rot_RightShoulder_Arm[2]
366         nao_RShoulderRoll = nao_RShoulderRoll*-1 #-- Invert the sign
367
368         #-- Limit the angle to the range of motion of the robot
369         if nao_RShoulderPitch < -2.09:
370            nao_RShoulderPitch = -2.09
371         if nao_RShoulderPitch > 2.09:
372            nao_RShoulderPitch = 2.09
373
374         if nao_RShoulderRoll < -1.33:
375            nao_RShoulderRoll = -1.33
376         if nao_RShoulderRoll > 0.31:
377            nao_RShoulderRoll = 0.31
378
379         self.nao_RShoulderPitch = nao_RShoulderPitch
380         self.nao_RShoulderRoll = nao_RShoulderRoll
381
382         #-- Print the values
383         # rospy.loginfo("nao_RShoulderPitch: %f  nao_RShoulderRoll: %f", nao_RShoulderPitch,
384 nao_RShoulderRoll)
385
386      except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
387         pass
388
389      #-- Lookup transform from Right Arm to Right Elbow
390      try:
391         (_,rot_RightArm_Forearm) = self.listener.lookupTransform('/RightForeArm', '/RightArm',
392 rospy.Time(0))
393         #-- Pass quaternion to euler angles
394         rot_RightArm_Forearm = euler_from_quaternion(rot_RightArm_Forearm)
395
396         rospy.logwarn("Angles \t %f \t %f \t %f",
397 rot_RightArm_Forearm[0],rot_RightArm_Forearm[1],rot_RightArm_Forearm[2])
398


400         nao_RElbowYaw = 0.0 # rot_RightArm_Forearm[1]
401         # nao_RElbowYaw = nao_RElbowYaw*-1 # Invert the angle
402
403         offset_RElbowRoll = -0.26
```

```
404         nao_RElbowRoll = rot_RightArm_Forearm[2] + offset_RElbowRoll
405
406         #-- Limit the angle to the range of motion of the robot
407         if nao_RElbowYaw < -2.09:
408             nao_RElbowYaw = -2.09
409         if nao_RElbowYaw > 2.09:
410             nao_RElbowYaw = 2.09
411
412         if nao_RElbowRoll < 0.0:
413             nao_RElbowRoll = 0.0
414         if nao_RElbowRoll > 1.54:
415             nao_RElbowRoll = 1.54
416
417         self.nao_RElbowYaw = nao_RElbowYaw
418         self.nao_RElbowRoll = nao_RElbowRoll
419
420         #-- Print the values
421         rospy.loginfo("nao_RElbowYaw: %f  nao_RElbowRoll: %f", nao_RElbowYaw, nao_RElbowRoll)
422
423     except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
424         pass
425
426     #-- Lookup transform from Right Forearm to Right Wrist
427     try:
428         (_, rot_RightForearm_Wrist) = self.listener.lookupTransform('/RightHand', '/RightForeArm',
429 rospy.Time(0))
430         #-- Pass quaternion to euler angles
431         rot_RightForearm_Wrist = euler_from_quaternion(rot_RightForearm_Wrist)
432         #-- Get the yaw
433         nao_RWristYaw = rot_RightForearm_Wrist[0]
434         nao_RWristYaw = nao_RWristYaw*-1 # Invert the angle
435
436
437         #-- Limit the angle to the range of motion of the robot
438         if nao_RWristYaw < -1.82:
439             nao_RWristYaw = -1.82
440         if nao_RWristYaw > 1.82:
441             nao_RWristYaw = 1.82
442
443         self.nao_RWristYaw = nao_RWristYaw
444
```

```
445         #-- Print the values
446         rospy.loginfo("nao_RWristYaw: %f", nao_RWristYaw)
447
448       except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
449         pass
450
451
452   def getJointStates_Left_arm(self):
453
454       # Define robot
455       L1 = 0.128
456       L2 = 0.2655
457       L3 = 0.260
458
459       robot_model = rtb.DHRobot([
460                           rtb.RevoluteMDH(alpha=-pi/2, a=L1, d=0.0,  qlim=[-0.31, 1.33], offset=-pi/2),
461                           rtb.RevoluteMDH(alpha=-pi/2, a=0.0, d=0.0, qlim=[-2.09, 2.09], offset= pi/2),
462                           rtb.RevoluteMDH(alpha= pi/2, a=0.0, d=L2,  qlim=[-2.09, 2.09], offset= pi/2),
463                           rtb.RevoluteMDH(alpha=-pi/2, a=0.0, d=0.0, qlim=[-1.54, 0.00], offset=0.0),
464                           rtb.RevoluteMDH(alpha= pi/2, a=0.0, d=L3,  qlim=[-1.82, 1.82], offset=0.0),
465       ])
466       # rot_mat_tool = np.array([[0,  0, -1],
467       #                          [ 0,  1,  0],
468       #                          [ 1,  0,  0]])
469       # tool_T = SE3.Rt(rot_mat_tool)
470       # robot_model.tool = tool_T
471
472       # rot_mat_base = np.array([[ 1,  0,  0],
473       #                          [ 0,  1,  0],
474       #                          [ 0,  0,  1]])
475       # base_T = SE3.Rt(rot_mat_base)
476       # robot_model.base = base_T
477
478       try:
479         (trans_LeftArm,rot_LeftArm) = self.listener.lookupTransform('/LeftShoulder', '/LeftHand',
480 rospy.Time(0))
481         # print("Distance: ", trans_LeftArm)

483         rot_matrix = quaternion_matrix(rot_LeftArm)[0:3,0:3]
484
485         T_ikine = SE3.Trans(trans_LeftArm[0],trans_LeftArm[1],trans_LeftArm[2]) * SE3.Rt(rot_matrix)
486
```

```
487         joints_ikine = robot_model.ikine_LMS(T_ikine,q0=self.joints_prev_left, mask=[1, 1, 1, 0, 0, 0],
488 tol=1e-4, ilimit=50)
489         if joints_ikine.success:
490             # rospy.logwarn("In funct")
491             # rospy.logwarn(joints_ikine.q)
492             # self.joints_prev_left = joints_ikine.q
493 
494             #-- Set joint values
495             nao_LShoulderPitch = joints_ikine.q[1]
496             nao_LShoulderRoll  = joints_ikine.q[0]
497             nao_LElbowYaw      = joints_ikine.q[2]
498             nao_LElbowRoll     = joints_ikine.q[3]
499             nao_LWristYaw      = joints_ikine.q[4]
500 
501             #-- Limit the angle to the range of motion of the robot
502             if nao_LShoulderPitch < -2.09:
503                 nao_LShoulderPitch = -2.09
504             if nao_LShoulderPitch > 2.09:
505                 nao_LShoulderPitch = 2.09
506 
507             if nao_LShoulderRoll < -0.31:
508                 nao_LShoulderRoll = -0.31
509             if nao_LShoulderRoll > 1.33:
510                 nao_LShoulderRoll = 1.33
511 
512             if nao_LElbowYaw < -2.09:
513                 nao_LElbowYaw = -2.09
514             if nao_LElbowYaw > 2.09:
515                 nao_LElbowYaw = 2.09
516 
517             if nao_LElbowRoll < -1.54:
518                 nao_LElbowRoll = -1.54
519             if nao_LElbowRoll > 0.00:
520                 nao_LElbowRoll = 0.0
521 
522             if nao_LWristYaw < -1.82:
523                 nao_LWristYaw = -1.82
524             if nao_LWristYaw > 1.82:
525                 nao_LWristYaw = 1.82
526 
527             #-- Set class variables
```

```
528             self.nao_LShoulderPitch = nao_LShoulderPitch
529             self.nao_LShoulderRoll = nao_LShoulderRoll
530             self.nao_LElbowYaw = nao_LElbowYaw
531             self.nao_LElbowRoll = nao_LElbowRoll
532             self.nao_LWristYaw = nao_LWristYaw
533
534
535         except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
536             pass
537
538     def getJointStates_Right_arm(self):
539
540         # Define robot
541         L1 = 0.128
542         L2 = 0.2655
543         L3 = 0.260
544
545         robot_model = rtb.DHRobot([
546                         rtb.RevoluteMDH(alpha=-pi/2, a=-L1, d=0.0,  qlim=[-1.33, 0.31], offset=-pi/2),
547                         rtb.RevoluteMDH(alpha=-pi/2, a=0.0, d=0.0,  qlim=[-2.09, 2.09], offset= pi/2),
548                         rtb.RevoluteMDH(alpha= pi/2, a=0.0, d=L2,   qlim=[-2.09, 2.09], offset= pi/2),
549                         rtb.RevoluteMDH(alpha=-pi/2, a=0.0, d=0.0,  qlim=[ 0.00, 1.54], offset=0.0),
550                         rtb.RevoluteMDH(alpha= pi/2, a=0.0, d=L3,   qlim=[-1.82, 1.82], offset=0.0),
551         ])
552         # rot_mat_tool = np.array([[0,  0, -1],
553         #                          [ 0,  1,  0],
554         #                          [ 1,  0,  0]])
555         # tool_T = SE3.Rt(rot_mat_tool)
556         # robot_model.tool = tool_T
557
558         # rot_mat_base = np.array([[ 1,  0,  0],
559         #                          [  0,  1,  0],
560         #                          [  0,  0,  1]])
561         # base_T = SE3.Rt(rot_mat_base)
562         # robot_model.base = base_T
563
564         try:
                                                                                                  and',
566 rospy.Time(0))
567             # print("Distance: ", trans_RightArm)
568             #--- Pass quaternion to matrix
569             rot_matrix = quaternion_matrix(rot_RightArm)[0:3,0:3]
```

```
570
571         T_ikine = SE3.Trans(trans_RightArm[0],trans_RightArm[1],trans_RightArm[2]) *
572 SE3.Rt(rot_matrix)
573
574         joints_ikine = robot_model.ikine_LMS(T_ikine,q0=self.joints_prev_right, mask=[1, 1, 1, 0, 0, 0],
575 tol=1e-4, ilimit=50)
576         if joints_ikine.success:
577             # rospy.logwarn("In funct")
578             # rospy.logwarn(joints_ikine.q)
579             # self.joints_prev_right = joints_ikine.q
580
581             #-- Set joint values
582             nao_RShoulderPitch = joints_ikine.q[1]
583             nao_RShoulderRoll  = joints_ikine.q[0]
584             nao_RElbowYaw      = joints_ikine.q[2]
585             nao_RElbowRoll     = joints_ikine.q[3]
586             nao_RWristYaw      = joints_ikine.q[4]
587
588             #-- Limit the angle to the range of motion of the robot
589             if nao_RShoulderPitch < -2.09:
590                 nao_RShoulderPitch = -2.09
591             if nao_RShoulderPitch > 2.09:
592                 nao_RShoulderPitch = 2.09
593
594             if nao_RShoulderRoll < -1.33:
595                 nao_RShoulderRoll = -1.33
596             if nao_RShoulderRoll > 0.31:
597                 nao_RShoulderRoll = 0.31
598
599             if nao_RElbowYaw < -2.09:
600                 nao_RElbowYaw = -2.09
601             if nao_RElbowYaw > 2.09:
602                 nao_RElbowYaw = 2.09
603
604             if nao_RElbowRoll < 0.0:
605                 nao_RElbowRoll = 0.0
606             if nao_RElbowRoll > 1.54:
607                 nao_RElbowRoll = 1.54
608
609             if nao_RWristYaw < -1.82:
610                 nao_RWristYaw = -1.82
```

```
611             if nao_RWristYaw > 1.82:
612                 nao_RWristYaw = 1.82
613 
614             #-- Set class variables
615             self.nao_RShoulderPitch = nao_RShoulderPitch
616             self.nao_RShoulderRoll  = nao_RShoulderRoll
617             self.nao_RElbowYaw      = nao_RElbowYaw
618             self.nao_RElbowRoll     = nao_RElbowRoll
619             self.nao_RWristYaw      = nao_RWristYaw
620 
621 
622         except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
623             pass
624 
625     def getJointStates_Head(self):
626 
627         #-- Lookup transform from Hips to Left leg
628         try:
629             (_,rot_Head) = self.listener.lookupTransform('/Spine3', '/Head_perc', rospy.Time(0))
630             #-- Pass quaternion to euler angles
631             rot_Head = euler_from_quaternion(rot_Head)
632             # rospy.logwarn("%f %f %f", rot_Head[0],rot_Head[1],rot_Head[2])
633             #-- Get the roll and pitch
634             nao_HeadYaw = rot_Head[1]
635 
636             nao_HeadPitch = rot_Head[0]
637 
638             #-- Limit the angle to the range of motion of the robot
639             if nao_HeadYaw < -2.09:
640                 nao_HeadYaw = -2.09
641             if nao_HeadYaw > 2.09:
642                 nao_HeadYaw = 2.09
643 
644             if nao_HeadPitch < -0.67:
645                 nao_HeadPitch = -0.67
646             if nao_HeadPitch > 0.51:
647                 nao_HeadPitch = 0.51
648 
649             self.nao_HeadYaw = nao_HeadYaw
650             self.nao_HeadPitch = nao_HeadPitch
651 
652             #-- Print the values
```

```
653         #rospy.loginfo("nao_HeadYaw: %f  nao_HeadPitch: %f", nao_HeadYaw, nao_HeadPitch)
654 
655       except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
656         pass
657 
658     def publishJointStates(self):
659 
660       #-- Create a JointState message
661       joint_states = sensor_msgs.msg.JointState()
662       joint_states.header.stamp = rospy.Time.now()
663       joint_states.name = ['LHipPitch', 'LHipRoll', 'LKneePitch', 'LAnklePitch', 'LAnkleRoll', # Left Leg
664                   'RHipPitch', 'RHipRoll', 'RKneePitch', 'RAnklePitch', 'RAnkleRoll',  # Right Leg
665                   'LShoulderPitch', 'LShoulderRoll', 'LElbowYaw', 'LElbowRoll', 'LWristYaw', # Left Arm
666                   'RShoulderPitch', 'RShoulderRoll', 'RElbowYaw', 'RElbowRoll', 'RWristYaw', # Right Arm
667                   'HeadYaw', 'HeadPitch', 'LHipYawPitch', 'RHipYawPitch', 'LHand', 'RHand' , # Head, Hips
668 and Hands
669                   'LFinger11','LFinger12', 'LFinger13', 'LFinger21', 'LFinger22','LFinger23', 'LThumb1',
670 'LThumb2',# Left Hand
671                   'RFinger11','RFinger12', 'RFinger13', 'RFinger21', 'RFinger22','RFinger23', 'RThumb1',
672 'RThumb2'] # Right Hand
673 
674       # rospy.logwarn("In publish")
675       # rospy.logwarn("%f %f %f %f %f" ,self.nao_LShoulderPitch, self.nao_LShoulderRoll,
676 self.nao_LElbowYaw, self.nao_LElbowRoll, self.nao_LWristYaw)
677       # rospy.logwarn("%f %f %f %f %f" ,self.nao_RShoulderPitch, self.nao_RShoulderRoll,
678 self.nao_RElbowYaw, self.nao_RElbowRoll, self.nao_RWristYaw)
679 
680       joint_states.position = [self.nao_LHipPitch, self.nao_LHipRoll, self.nao_LKneePitch,
681 self.nao_LAnklePitch, self.nao_LAnkleRoll,
682                   self.nao_RHipPitch, self.nao_RHipRoll, self.nao_RKneePitch, self.nao_RAnklePitch,
683 self.nao_RAnkleRoll,
684                   self.nao_LShoulderPitch, self.nao_LShoulderRoll, self.nao_LElbowYaw,
685 self.nao_LElbowRoll, self.nao_LWristYaw,
686                   self.nao_RShoulderPitch, self.nao_RShoulderRoll, self.nao_RElbowYaw,
687 self.nao_RElbowRoll, self.nao_RWristYaw,
688                   self.nao_HeadYaw, self.nao_HeadPitch, 0.0, 0.0, 0.0, 0.0, # Head, Hips and Hands
689                   0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, # Left Hand
690                   0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] # Right Hand
691 
692       joint_states.velocity = []
693       joint_states.effort = []
```

```
694
695         #-- Publish the JointState message
            self.pub_joint_states.publish(joint_states)

        def spin(self):

            while not rospy.is_shutdown():

                # Get states of left leg
                self.getJointStates_Left_leg()

                # Get states of right leg
                self.getJointStates_Right_leg()

                # Get states of left arm
                self.getJointStates_Left_arm()

                # Get states of right arm
                self.getJointStates_Right_arm()

                # Get states of head
                self.getJointStates_Head()

                # Publish the JointState message
                self.publishJointStates()
                # rospy.loginfo("Publishing joint states")

                self.rate.sleep()
```

Move_naojoints_rviz.py

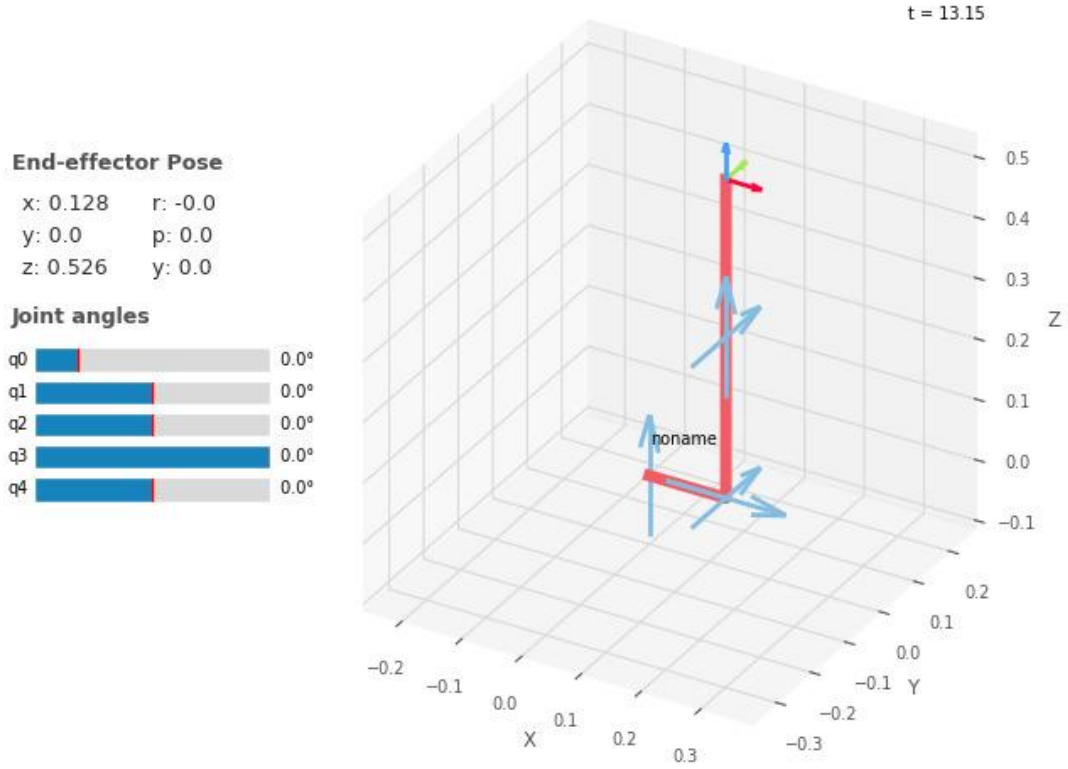
```
1   #!/usr/bin/python3
2
3   import rospy
4   import math
5   import tf
6   from tf.transformations import euler_from_quaternion
7   import geometry_msgs.msg
8   import sensor_msgs.msg
```

```
9  import angles
10 import MoveNaoJoints
11
12 if __name__ == '__main__':
13
14    #-- Initialize node
15    rospy.init_node('move_nao_joints_py')
16
17    rospy.loginfo("Init node")
18
19    #-- Create the MoveNaoJoints object
20    move_nao_joints = MoveNaoJoints.MoveNaoJoints()
21
22    #-- Do a transform lookup at a constant rate
23    move_nao_joints.spin()
```

test_arm_model.py

```
1  import numpy as np
2  import roboticstoolbox as rtb
3  from spatialmath import *
4  import spatialmath.base as base
5  from math import pi
6
7  # Modified Denavit-Hartenberg parameters
8  # Robotics Toolbox - Peter Corke
9
10 L1 = 0.128
11 L2 = 0.2655
12 L3 = 0.260
13
14 robot_model = rtb.DHRobot([
15                    rtb.RevoluteMDH(alpha=-pi/2, a=L1, d=0.0,  qlim=[-0.31, 1.33], offset=-pi/2),
16                    rtb.RevoluteMDH(alpha=-pi/2, a=0.0, d=0.0, qlim=[-2.09, 2.09], offset= pi/2),
17                    rtb.RevoluteMDH(alpha= pi/2, a=0.0, d=L2,  qlim=[-2.09, 2.09], offset= pi/2),
18                    rtb.RevoluteMDH(alpha=-pi/2, a=0.0, d=0.0, qlim=[-1.54, 0.00], offset=0.0),
19                    rtb.RevoluteMDH(alpha= pi/2, a=0.0, d=L3,  qlim=[-1.82, 1.82], offset=0.0),
20 ])
21
22 robot_model.teach()
```

Figura 7-1: Modelo por cinemática directa del brazo del robot NAO, a partir de los parámetros DH modificados. Código fuente *test_arm_model.py.*



Fuente: Elaboración propia.

Bibliografía

[1] L. Wing-Yue Geoffrey , M. Sharaf y N. Goldie, «Human-Robot Interaction for Rehabilitation Robots,» de *Robotic Assistive Technologies: Principles and Practice*, Boca Raton, CRC Press, Taylor & Francis Group, 2017, pp. 26-27, 40.

[2] L. H. Thomas, B. French, J. Coupe, N. McMahon, L. Connell, J. Harrison, C. J. Sutton, S. Tishkovskaya y C. L. Watkins, «Repetitive task training for improving functional ability after stroke (Review),» *Stroke,* vol. 48, nº 4, pp. 102-103, 2017.

[3] J. W. Burke, . M. D. J. McNeill, D. K. Charles, P. J. Morrow, J. H. Crosbie y S. M. McDonough , «Optimising engagement for stroke rehabilitation using serious games,» *The Visual Computer,* vol. 25, nº 12, pp. 1085-1099, 2009.

[4] A. Jaume-i-Capó, P. Martínez-Bueso, B. Moyà-Alcover y J. Varona, «Interactive rehabilitation system for improvement of balance therapies in people with cerebral palsy,» *IEEE Trans Neural Syst Rehabil Eng.,* vol. 22, nº 2, pp. 419-427, 2014.

[5] A. Gil-Agudo, I. Dimbwadyo-Terrer, B. Peñasco-Martín, A. de los Reyes-Guzmán, A. Bernal-Sahún y A. Berbel-García, «Experiencia clínica de la aplicación del sistema de realidad TOyRA en la neuro-rehabilitación de pacientes con lesión medular,» *Rehabilitación,* vol. 46, nº 1, pp. 41-48, 2012.

[6] P. Wang, . L. Li, M. Yan, F. Ru y Y. Ju , «Repetitive control for the periodic walking training in a gait rehabilitation robot,» *Artificial Life and Robotics,* vol. 20, nº 2, pp. 159-165, 2015.

[7] E. Martinez-Martin, M. Cazorla y S. Orts-Escolano, «Machine Learning Techniques for Assistive Robotics,» *MDPI: electronics,* vol. 9, nº 5, p. 821, 2020.

[8] C. Bodine, L. Sliker, M. Marquez, C. Clark, B. Burne y J. Sandstrum, «Social Assistive Robots for Children with Complex Disabilities,» de *Robotic Assitive Tecnologies: Principles and Practice*, Boca Raton, CRC Press, Taylor & Francis Group, 2017, pp. 263, 295-297.

[9] R. Baker, «Gait analysis methods in rehabilitation,» *Journal of NeuroEngineering and Rehabilitation,* vol. 3, nº 4, pp. 1-7, 2006.

[10] R. B. Davis III, S. Õunpuu, D. Tyburski y J. R. Gage, «A gait analysis data collection and reduction technique,» *Human Movement Science,* vol. 10, nº 5, pp. 575-487, 1991.

[11] L. L. Gómez Echeverry, A. M. Jaramillo Henao, M. A. Ruiz Molina, S. . M. Velásquez Restrepo, C. A. Páramo Velásquez y G. J. Silva Bolívar, «Human motion capture and analysis systems: a systematic review,» *PROSPECTIVA Vol. 16 - No. 2,* pp. 24-34, 2018.

[12] P. Kopniak, «Motion capture using multiple Kinect controllers,» *Przeglad. Elektrotechniczny,* vol. 91, nº 8, pp. 26-29, 2015.

[13] B. Gabbasov, I. Danilov, I. Afanasyev y E. Magid, «Toward a human- like biped robot gait: Biomechanical analysis of human locomo- tion recorded by Kinect-based Motion Capture system. 2015 - 10th International Symposium on Mechatronics and its Applications.,» *10th International Symposium on Mechatronics and its Applications (ISMA),* pp. 8-13, 2015.

[14] S. Elbeleidy, T. Mott y T. Williams, «Practical, Ethical, and Overlooked: Teleoperated Socially Assistive Robots in the Quest for Autonomy,» *17th ACM/IEEE International Conference on Human-Robot Interaction (HRI),* 2022.

[15] J. Koenemann, F. Burget y M. Bennewitz, «Real-time Imitation of Human Whole-Body Motions by Humanoids,» *IEEE International Conference on Robotics and Automation (ICRA),* 2014.

[16] G. Emre Cemal, C. YuJung y K. ChangHwan , «Imitation of Human Upper-Body Motions by Humanoid Robots,» *16th International Conference on Ubiquitous Robots (UR),* p. 24, 2019.

[17] B. M. Lütjens, «Real-Time Teleoperation of Industrial Robots with the Motion Capture System Perception Neuron,» Technische Universität München, Munich, 2017.

[18] J. C. Pulido, C. Suárez-Mejías, J. C. González, A. Dueñas Ruiz, P. Ferrand Ferri, M. E. Martínez Sahuquillo, C. Echevarría Ruiz De Vargas, P. Infante-Cossio y C. L. Parra Calderón, «A Socially Assistive Robotic Platform for Upper-Limb Rehabilitation: A Longitudinal Study With Pediatric Patients,» *IEEE Robotics & Automation Magazine,* vol. 26, nº 2, pp. 24-39, 2019.

[19] C. Girard, D. Calderón de León, A. Arafat Lemus, V. Ferman y J. Fajardo, «A Motion Mapping System for Humanoids that Provides Immersive Teleprescence

Experiences,» *6th International Conference on Mechatronics and Robotics Engineering (ICMRE),* 2020.

[20] J. C. Cerón, M. S. Haque Sunny , B. Brahmi, L. M. Mendez, R. Fareh, H. U. Ahmed y M. H. Rahman, «A Novel Multi-Modal Teleoperation of a Humanoid Assistive Robot with Real-Time Motion Mimic,» *Micromachines,* vol. 14 , nº 2, p. 461, 2023.

[21] C. Stanton, A. Bogdanovych y E. Ratanasena, «Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning,» *Proceedings of Australasian Conference on Robotics and Automation, 3-5 Dec 2012,,* 2012.

[22] I. Rodriguez, A. Astigarraga, E. Jauregi, E. Ruiz y E. Lazcano, «Humanizing NAO robot teleoperation using ROS,» *14th IEEE-RAS International Conference on Humanoid Robots (Humanoids),* 2014.

[23] M. J. Matarić, J. Eriksson, D. J. Feil-Seifer y C. J. Winstein, «Socially assistive robotics for post-stroke rehabilitation,» *Journal of NeuroEngineering and Rehabilitation, Bio Med Central,* vol. 4, nº 5, pp. 1-9, 2007.

[24] S. Matiz y A. De La Barrera, Interviewees, *Grabación del Webinar- Robot NAO como colaborador terapéutico educativo y social.* [Entrevista]. 17 abril 2023.

[25] Hospital Militar, «Laboratorio de Innovación INNLAB,» 03 febrero 2022. [En línea]. Available: https://www.hospitalmilitar.gov.co/index.php?idcategoria=70789. [Último acceso: 19 junio 2023].

[26] Ó. J. Perdomo Charry, H. Bernal, Á. D. Orjuela, S. Cadavid Espinha, O. Müller, K. Aguia Rojas y L. M. Leal Villamizar, «'Robins' enseña el español a niños y niñas sordos,» *Revista Divilgación Científica. Ciencia y Tecnología,* nº 6, pp. pp. 01-06, 2022.

[27] K. Darvish, L. Penco, J. Ramos, R. Cisneros, J. Pratt, E. Yoshida, S. Ivaldi y D. Pucci, «Teleoperation of Humanoid Robots: A Survey,» *IEEE Transactions on Robotics,* pp. 1-22, 2023.

[28] DANE, «Discapacidad,» Noviembre 2020. [En línea]. Available: https://www.dane.gov.co/index.php/estadisticas-por-tema/demografia-y-poblacion/discapacidad. [Último acceso: 27 julio 2023].

[29] World Health Organization, «Rehabilitation 2030 Initiative,» World Health Organization, [En línea]. Available: https://www.who.int/initiatives/rehabilitation-2030. [Último acceso: 27 julio 2023].

[30] J. Valčík, «Similarity Models for Human Motion Data,» Masaryk University, Brno, 2016.

[31] D. Bravo, C. Rengifo y W. Agredo, «Comparison of two Motion Capture Systems by means of joint Trajectories of human gait,» *Revista mexicana de ingeniería biomédica,* vol. 37, nº 2, p. 55, 2016.

[32] Noitom Limited, Axis Neuron User Guide, Miami, 2020.

[33] Noitom Limited, Axis Neuron User Manual V3.8.1.5, 2023.

[34] University of Wisconsin-Madison, «Biovision BVH,» University of Wisconsin-Madison, 2023. [En línea]. Available: https://research.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html. [Último acceso: 19 junio 2023].

[35] R. Sers, S. Forrester, . E. Moss, S. Ward, J. Ma y M. Zecca, «Validity of the Perception Neuron inertial motion capture system for upper body motion analysis,» *Measurement,* vol. 149, nº 1, p. 9, 2019.

[36] D. Bauer, Rehabilitación : enfoque integral : principios prácticos, Barcelona: Masson Salvat Medicina, Ediciones científicas y técnicas S.A.S., 1992.

[37] C. Pimentel, F. E. Pinchao , L. C. Rodriguez y R. A. Espinosa, «Ludic technology for the rehabilitation of upper limb in young people,» *Visión Electrónica,* pp. 215-225, 2018.

[38] Minsalud, «Política Pública Nacional de Discapacidad e Inclusión 2013-2022,» Julio 2020. [En línea]. Available: https://www.minsalud.gov.co/sites/rid/Lists/BibliotecaDigital/RIDE/DE/PS/politica-publica-discapacidad-2013-2022.pdf. [Último acceso: 17 junio 2023].

[39] M. M. Matheus y A. Ríos Rincón, «Technology in Rehabilitation: A Conceptual Approach,» *Rev. Cienc. Salud. Bogotá (Colombia),* vol. 4, nº 1, pp. 98-108, 2006.

[40] J. Reswick, «What is Rehabilitation Engineering?,» *Ann. Rev. Rehabilitation ,* vol. 2, 1982.

[41] J. A. Acevedo Londoño, E. Caicedo Bravo y J. F. Castillo García, «Application of robotics rehabilitation technologies in children with upper limb disabilities,» *Revista de la Universidad Industrial de Santander. Salud, vol. 49, núm. 1,* pp. 103-114, 2017.

[42] A. M. Norjasween, F. A. khtar Hanapiah, R. A. Abdul Rahman y H. Yussof, «Emergence of Socially Assistive Robotics in Rehabilitation for Children with Cerebral Palsy: A Review,» *International Journal of Advanced Robotic Systems,* pp. 1-7, 2016.

[43] P. Encarnação, «Fundamentals of Robotic Assistive Technologies,» de *Robotic Assistive Technologies: Principles and Practice*, Boca Raton, CRC Press, 2019, pp. 1-25.

[44] S. Fojtu, M. Havlena y T. Pajdla, «Nao Robot Localization and Navigation Using Fusion of Odometry and Visual Sensor Data,» *Center for Machine Perception, Department of Cybernetics, FEE, CTU in Prague,* vol. 121, nº 35, 2012.

[45] Revista de Robots, «ROBOT NAO PARA EMPRESA Y EDUCACIÓN,» Revista de Robots, 8 junio 2023. [En línea]. Available: https://revistaderobots.com/robots-y-robotica/robot-nao-caracteristicas-y-precio/?cn-reloaded=1. [Último acceso: 2023 junio 24].

[46] ROS wiki, «ROS.org,» 18 agosto 2021. [En línea]. Available: http://wiki.ros.org/es/ROS/Introduccion#:~:text=ROS%20es%20una%20estructura%20distribuida,ser%20intercambiados%2C%20compartidos%20y%20distribuidos.. [Último acceso: 2023 junio 24].

[47] J. I. L. Aya y D. F. Romero Ibañez, «Simulación de un robot hexapodo con ros y gazebo,» Universidad Jorge Tadeo Lozano, Bogotá, 2020.

[48] ROS wiki, «ROS.org,» ROS wiki, 10 febrero 2021. [En línea]. Available: http://wiki.ros.org/es/ROS/Conceptos. [Último acceso: 24 junio 2023].

[49] T. Foote, «tf: The transform library,» *IEEE Conference on Technologies for Practical Robot Applications (TePRA),* pp. 1-6, 2013.

[50] ROS wiki, «ROS.org,» ROS wiki, 02 octubre 2017. [En línea]. Available: http://wiki.ros.org/tf. [Último acceso: 2023 junio 24].

[51] B. Lutjens, «perc-neuron-ros-ur10,» 2019. [En línea]. Available: https://github.com/blutjens/perc_neuron_ros_ur10. [Último acceso: 19 Noviembre 2021].

[52] S. Haller, «perception-neuron-ros,» 2017. [En línea]. Available: https://github.com/smhaller/perception-neuron-ros. [Último acceso: 19 Noviembre 2021].

[53] Open Robotics, «Open Robotics,» 2019. [En línea]. Available: http://wiki.ros.org/nao. [Último acceso: 16 Febrero 2022].

[54] Noitom Limited, «Perception Neuron,» Noitom Limited, 2023. [En línea]. Available: https://neuronmocap.com/pages/perception-neuron-studio-system. [Último acceso: 19 Junio 2023].

[55] Noitom Limited, «Perception Neuron,» Noitom Limited, 2023. [En línea]. Available: https://neuronmocap.com/pages/mocap-api. [Último acceso: 17 Junio 2023].

[56] Noitom Technology Co.,Ltd., «Neuron Data Reader Runtime API,» Noitom Technology Co.,Ltd., 28 08 2020. [En línea]. Available: https://shopcdn.noitom.com.cn/newimage/0c51544770fb49d79f814e9446875121.pdf. [Último acceso: 02 julio 2023].

[57] L. M. Contreras Tapias, J. A. Galicia Valdovinos, J. C. Mayoral Baños y A. Parrales Salinas, «Análisis Cinemático Robot NAO Aldebaran,» pp. 1-5.

[58] P. Corke, «Kinematics,» Git Hub, 23 Enero 2021. [En línea]. Available: https://github.com/petercorke/robotics-toolbox-python/wiki/Kinematics. [Último acceso: 2023 noviembre 07].

[59] P. Corke, «Robot.ikine_LM,» Git Hub, [En línea]. Available: https://petercorke.github.io/robotics-toolbox-python/IK/stubs/roboticstoolbox.robot.Robot.Robot.ikine_LM.html. [Último acceso: 2023 noviembre 06].

[60] Aldebaran, «NAO Joints,» Aldebaran, [En línea]. Available: http://doc.aldebaran.com/2-4/family/robots/joints_robot.html. [Último acceso: 28 julio 2023].

[61] M. Turp, J. C. González y F. Fernández, «Developing a Robot-Guided Interactive Simon Game for Physical and Cognitive Training,» *International Journal of Humanoid Robotics,* vol. 16, nº 01, pp. 1-25, 2019.

[62] I. Almetwally y M. Mallem, «Real-time Tele-operation and Tele-walking of Humanoid Robot Nao using Kinect Depth Camera,» *10th IEEE international conference on networking, sensing and control (icnsc),* pp. 1-4, 2013.

[63] C. Gu, W. Lin, X. He, Z. Lei y M. Zhang, «IMU-based motion capture system for rehabilitation applications: A systematic review,» *Biomimetic Intelligence and Robotics,* vol. 3, nº 2, pp. 1-13, 2023.