UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA

"ROBOT MÓVIL DE ALTO TORQUE: VISIÓN ARTIFICIAL"

LÍNEA DE INVESTIGACIÓN: Innovación y transferencia de tecnología

AUTOR: Pedro Saúl Hinojosa Rojas

DIRECTOR: PhD. Carlos Xavier Rosero Chandi

Ibarra – Ecuador

2025

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	100519543-1		
APELLIDOS Y NOMBRES:	Hinojosa Rojas Pedro Saúl		
DIRECCIÓN:	Imbabura, Otavalo		
EMAIL:	pshinojosar@utn.edu.ec		
TELÉFONO FIJO:	2-520-902	**TELÉFONO MÓVIL:**	0996548573

DATOS DE LA OBRA	
TÍTULO:	"ROBOT MÓVIL DE ALTO TORQUE: VISIÓN ARTIFICIAL"
AUTOR:	Hinojosa Rojas Pedro Saúl
FECHA:	26/02/2025
SOLO PARA TRABAJOS DE TITULACIÓN	
PROGRAMA:	■ PREGRADO POSGRADO
TÍTULO POR EL QUE OPTA:	Ingeniero en Mecatrónica
DIRECTOR:	PhD. Carlos Xavier Rosero Chandi

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN MECATRÓNICA

AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Pedro Saúl Hinojosa Rojas, con cédula de identidad Nro. 1005195431, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de integración curricular descrito anteriormente, hago entrega del ejemplar respectivo en formato digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior Artículo 144.

Ibarra, a los 26 días del mes de febrero de 2025.

EL AUTOR

Firma: .....

Nombre: Pedro Saúl Hinojosa Rojas

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de esta y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 26 días del mes de febrero de 2025.

EL AUTOR

Firma: .....

Nombre: Pedro Saúl Hinojosa Rojas

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN MECATRÓNICA

CERTIFICACIÓN DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 20 de febrero de 2025

PhD. Carlos Xavier Rosero Chandi

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de titulación, el mismo que se ajusta a las normas vigentes de la Unidad Académica de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

(f)……

PhD. Carlos Xavier Rosero Chandi
C.C.: 1002515821

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

APROBACIÓN DEL COMITÉ CALIFICADOR

El Tribunal Examinador del trabajo de titulación "ROBOT MÓVIL DE ALTO TORQUE: VISIÓN ARTIFICIAL" elaborado por PEDRO SAÚL HINOJOSA ROJAS, previo a la obtención del título de INGENIERO EN MECATRÓNICA, aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte:

(f)……

PhD. Carlos Xavier Rosero Chandi

C.C.: 1002515821

(f)……

MSc. Luz María Tobar Subia Contento

C.C.: 1002444204

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN MECATRÓNICA

AGRADECIMIENTO

A Dios, a mis padres, y a mis hermanos, por la motivación y el apoyo en mi formación tanto académica como personal.

A mis amigos Edwin, Andrew y Julián, por acompañarme a lo largo de esta etapa universitaria.

A mi director Xavier Rosero, por haberme guiado en todas las etapas de este trabajo de titulación.

Pedro Saúl Hinojosa Rojas

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN MECATRÓNICA

DEDICATORIA

A mi madre, quien ha dedicado su vida entera a guiarme con amor, fortalecido y empujado a ser mejor cada día. Eres la fuerza detrás de cada paso que he dado, la voz que me anima cuando dudo y el corazón que late en cada uno de mis logros. Este trabajo es un reflejo de tu dedicación y tu fe en mí. Gracias por ser mi inspiración eterna.

A mi padre, por ser mi ejemplo de lucha y dedicación. Me ha mostrado que no hay obstáculo que no se pueda superar. Gracias por ser mi guía, mi soporte y mi inspiración en cada paso de este camino.

Pedro Saúl Hinojosa Rojas

Resumen

Este proyecto desarrolla un sistema de visión artificial para un robot móvil de alto torque, cuya construcción, diseño mecánico y electrónico están detallados en ambos trabajos del mismo nombre. Se utiliza software de código abierto para garantizar su accesibilidad y replicabilidad. Se integra una Jetson Nano de 2 GB de RAM y un sensor de profundidad Kinect, con la implementación de drivers y nodos específicos para procesar y visualizar datos en un entorno tridimensional. La investigación se centra en la adquisición, interpretación y representación gráfica de la información de profundad en tiempo real, utilizando herramientas de visión por computadora dentro del entorno de Robot Operating System 2 (ROS 2). Los resultados obtenidos demuestran la efectividad del sistema en la captura y análisis de datos de profundidad, proporcionando una base técnica para futuras aplicaciones en navegación autónoma y toma de decisiones en robótica móvil.

Palabras clave: Visión artificial, Robótica móvil, ROS 2, Sensores de profundidad.

Abstract

This project develops a machine vision system for a high-torque mobile robot, whose construction, mechanical and electronic design are detailed in both documents of the same name. Open source software is used to ensure its accessibility and replicability. A 2 GB RAM Jetson Nano and a Kinect depth sensor are integrated, with the implementation of specific drivers and nodes to process and visualize data in a three-dimensional environment. The research focuses on the acquisition, interpretation and graphical representation of depth information in real time, using computer vision tools within the Robot Operating System 2 (ROS 2) environment. The results obtained demonstrate the effectiveness of the system in capturing and analyzing depth data, providing a technical basis for future applications in autonomous navigation and decision making in mobile robotics.

Keywords: Artificial vision, Mobile robotics, ROS 2, Depth sensors.

Índice general

Índice de figuras

Índice de tablas

Capítulo I

Introducción

1.1. Planteamiento del problema

En la era de la automatización y la inteligencia artificial, la robótica ha adquirido un rol de vital importancia en la formación de profesionales altamente cualificados. En este contexto, las instituciones académicas desempeñan un papel fundamental en la capacitación de futuros ingenieros en este campo [1]. No obstante, es imperativo abordar con urgencia la creación de un entorno eficiente y pragmático en el ámbito de la robótica, con el fin de potenciar al máximo el talento y el potencial de los jóvenes interesados en esta disciplina.

Uno de los principales desafíos que se plantean al formarse en el ámbito de la robótica es la restricción de recursos disponibles para comprender los sistemas sumamente complejos que involucran el funcionamiento de un robot. En el ámbito de la investigación robótica en las universidades locales, es común que se carezca de la potencia y versatilidad necesarias para abordar proyectos de envergadura y desafiantes. Esta limitación de recursos repercute de manera significativa en la capacidad de las instituciones de educación superior para explorar y comprender conceptos de robótica avanzada, tales como la dinámica de robots, el control de movimiento y la interacción hombre-máquina, como se menciona en [2].

La gran mayoría de los robots utilizados en este ámbito de investigación son adquiridos en el extranjero. Esto implica que carecemos de información detallada sobre el proceso de fabricación, ensamblaje y desarrollo de los sistemas, tanto en el aspecto mecánico como eléctrico y de control. El enfoque se limita, en su mayoría, a comprender únicamente el

funcionamiento básico de estos robots, quedando en un nivel superficial en cuanto a la comprensión de sus funcionalidades.

En la búsqueda de superar los desafíos en la formación de profesionales en el campo de la robótica, resulta innegable que la restricción de recursos disponibles representa un obstáculo de considerable envergadura [3]. La limitación en cuanto a la potencia y versatilidad de los recursos disponibles en las instituciones educativas locales limita de manera significativa la capacidad de emprender proyectos de complejidad, lo que a su vez afecta negativamente la exploración de conceptos avanzados en el ámbito de la robótica. La adquisición de robots en el extranjero agrava la situación al restringir el acceso al conocimiento detallado sobre su proceso de fabricación y desarrollo. Este hecho, a su vez, sitúa a los estudiantes en un nivel de comprensión meramente superficial de estos sistemas. En aras de fortalecer la formación en robótica móvil, se torna esencial abordar esta carencia de recursos y fomentar la autonomía en la investigación y desarrollo de sistemas robóticos [4].

Fomentar el interés en la robótica competitiva en el país o estimular la proliferación de competencias en Institutos de Educación Superior o inclusive niveles previos. Un robot móvil de alto torque diseñado para abordar diversas áreas del conocimiento en el campo de la robótica busca profundizar en el análisis de las características comunes que comparten los robots con alta capacidad de fuerza de avance y resistencia [5].

Hasta la fecha, en el país, no se ha establecido un enfoque de investigación dedicado a la resistencia y durabilidad en robots. En su lugar, los diseños existentes se ciñen únicamente a las especificaciones mecánicas y electrónicas mínimas, las cuales varían según el tipo de robot en cuestión. En muchas ocasiones, los robots fabricados en convenciones o competencias se centran exclusivamente en aspectos como la velocidad de desplazamiento y agilidad, sin otorgar la debida importancia a la resistencia o al torque de sus motores. Numerosos ejemplos ejemplifican esta problemática, como la creación de robots de competencia que se desempeñan en tareas que abarcan desde seguidores de línea y velocistas, hasta seguidores de destreza o robots capaces de resolver laberintos [6].

1.2. Objetivos

1.2.1. General

Desarrollar el sistema de visión artificial del robot móvil de alto torque, con la finalidad de habilitar su capacidad de percibir la realidad.

1.2.2. Específicos

- Definir los componentes a emplear para una eficaz captura de datos que sirvan como parte esencial de la lectura de variables externas.
- Diseñar el algoritmo de visión artificial que permita al robot la interpretación de su entorno.
- Realizar simulaciones para la comprobación del funcionamiento de la lógica de programación desarrollada.

1.3. Alcance

El propósito de este proyecto radica en el diseño, definición y desarrollo de un sistema de visión artificial destinado a un robot móvil de alto torque. Con el fin de alcanzar esta meta, se utilizará una tarjeta electrónica con la capacidad de interpretar los datos capturados por una cámara. Además, se llevará a cabo una implementación de software para establecer el algoritmo que permitirá al robot adquirir la habilidad de observar y comprender el entorno físico circundante.

1.4. Justificación

En el ámbito tecnológico, contribuirá al avance de la programación de inteligencia artificial para robots, incluyendo la implementación de nuevas librerías, la exploración de algoritmos novedosos, la optimización de los ya existentes y la creación de enfoques innovadores [7].

En el ámbito de la instrumentación y los sensores, se impulsará la relevancia de llevar a cabo una calibración precisa de los sensores y realizar dicha calibración de forma precisa. Además, se elaborarán recomendaciones relacionadas con el mantenimiento de estos dispositivos.

Desde un enfoque social, el desarrollo de un robot conlleva una serie de desafíos y sacrificios, los cuales a menudo se consideran superfluos, y en algunas ocasiones, el resultado final puede hacer que el fin no justifique los medios. No obstante, al enfocarnos en la contribución de material de estudio, que abarca no solo lo teórico, sino también lo experimental, se nos acerca a la posibilidad de perfeccionar nuestras habilidades en nuestras respectivas áreas de estudio [8]. Esto, a su vez, nos encamina hacia la formación de profesionales de alta calidad, porque siempre vale la pena invertir en el conocimiento.

Capítulo II

Revisión literaria

2.1. Estado del arte

La integración de la visión artificial en robots móviles ha permitido mejorar su capacidad de percepción y toma de decisiones en tiempo real, facilitando la detección de obstáculos, el reconocimiento de objetos y la interacción con el entorno [9]. Con el uso de cámaras RGB-D y sensores de profundidad, los robots pueden interpretar datos visuales de manera más eficiente sin depender de procesamiento externo. Además, el uso de entornos de desarrollo como el Robot Operating System ha optimizado la comunicación entre sensores y sistemas de control, permitiendo el desarrollo de arquitecturas modulares para la navegación autónoma. Estos avances han impulsado aplicaciones en la industria, la logística y la robótica en general, aunque persisten desafíos como la adaptación a variaciones en las condiciones de iluminación y la optimización del procesamiento en dispositivos embebidos. A medida que la tecnología evolucione, la visión artificial seguirá siendo un pilar fundamental en el desarrollo de robots móviles más autónomos e inteligentes [9].

2.2. Antecedentes

Se lleva a cabo la implementación de robots autónomos para el control de malezas en campos de arroz, así lo estipula [10] como una alternativa más productiva y sostenible en comparación con métodos tradicionales, como el uso de herbicidas químicos que son perjudiciales para el medio ambiente. Para lograr la operación completamente autónoma de control 5 de malezas, se utiliza una combinación de tecnologías, como GNSS, brújula y

visión por computadora, para guiar con precisión los robots a través de las filas de cultivo sin dañar las plantas de arroz y detectar el final de las filas. Se desarrolla un algoritmo de detección de filas de cultivo para guiar al robot con precisión en diversas condiciones, como la intensidad de las malezas, el crecimiento de las plantas y el clima. El sistema demuestra un buen rendimiento en condiciones de baja concentración de malezas, aunque su precisión disminuye a medida que aumenta la concentración de malezas, pero aun así permite que el robot navegue sin causar daños graves a las plantas.

En el trabajo que presenta [11], se describe el diseño de un robot móvil robusto, modular y rápido, capaz de operar en entornos desafiantes, con aplicaciones potenciales en ámbitos militares, de rescate e industriales. El principal desafío en el diseño del robot es superar las limitaciones de altura, lo cual se logra mediante el uso de sistemas hidráulicos y neumáticos pesados. Se emplea un sistema de elevación en espiral para cambiar la altura de la plataforma del robot, aprovechando la eficiencia de esta tecnología para levantar cargas pesadas con bajo consumo de energía y un mecanismo sencillo. El prototipo se modela y simula en entornos 3D para cambiar de altura, extender las piernas e incluso caminar por terrenos accidentados. Hay muy pocos robots diseñados para realizar tantas tareas. Una vez realizadas las pruebas, se concluye que no existe un sistema "ideal" que pueda maximizar controlabilidad, maniobrabilidad y estabilidad al mismo tiempo.

De igual manera, en [12], existe un método de navegación de robots basado en visión, utilizando una única cámara omnidireccional (catadióptrica). Se explica cómo las imágenes omnidireccionales se utilizan para generar representaciones necesarias para dos modalidades principales de navegación. Navegación topológica y seguimiento visual de trayectorias. Para abordar cambios en la iluminación, se utiliza una aproximación al espacio propio de la medida de Hausdorff. Así mismo, se presenta un método para transformar las imágenes omnidireccionales en "Bird's Eye Views" que corresponden a vistas ortográficas escaladas del plano del suelo. Estas imágenes se utilizan para controlar localmente la orientación del robot mediante servomotores visuales. El "Visual Path Following" se emplea para controlar con precisión al robot a lo largo de una trayectoria predeterminada, utilizando las vistas en "Bird's Eye" para rastrear puntos de referencia en el plano del suelo. Gracias a la geometría simplificada de estas imágenes, se puede estimar fácilmente la posición del robot y utilizarla

para un seguimiento preciso de la trayectoria. Este tipo de imágenes facilita la navegación basada en puntos de referencia, ya que estos permanecen visibles en todas las imágenes, a diferencia de una cámara estándar con un campo de visión más limitado. En pruebas realizadas en un robot preconstruido, el algoritmo logra trazar una trayectoria hasta alcanzar la posición final, en el interior del laboratorio.

De igual forma en [13], se está desarrollando un prototipo robótico móvil con capacidades multitarea, visión artificial y programación en lenguaje Python para la Universidad Politécnica Salesiana. La necesidad surge de la falta de prototipos robóticos que permitan a los estudiantes llevar a cabo prácticas para probar nuevas tecnologías en los laboratorios universitarios. El diseño del robot busca superar este desafío mediante la creación de un sistema versátil y eficiente. El robot se modela y simula en entornos 3D, permitiendo cambios de altura, extensión de patas y la capacidad de desplazarse en terrenos variados. Este enfoque innovador promete ser de gran utilidad para la comunidad universitaria en su búsqueda de soluciones tecnológicas avanzadas, esto es lo que se menciona en.

El artículo que presenta [14], se evidencia el desarrollo, implementación y pruebas de una plataforma robótica semiautónoma diseñada para fines educativos e investigativos. La plataforma se enfoca en enseñar a los estudiantes a diseñar una plataforma electromecánica estable, explorar diversos sensores para superar obstáculos, interconectar componentes electrónicos a un microcontrolador y programar el chip del microcontrolador para controlar la plataforma robótica. Los objetivos de investigación incluyen desarrollar y analizar el rendimiento de diferentes algoritmos de control para entender el comportamiento y la evitación de obstáculos. El diseño de hardware modular utiliza el bus I2C para la interconexión de sensores y controladores de motor con el microcontrolador ATMEL (AVR ATmega32). La integración del hardware se realiza en una placa de aplicación, constituyendo un sistema embebido. En el ámbito del software, se utiliza un compilador de C (ImageCraft) y se adopta un enfoque descendente para diseñar los distintos módulos. Aunque se prometen resultados experimentales para demostrar el potencial de los módulos de hardware y software, el artículo no proporciona detalles específicos sobre estos resultados ni describe la eficacia de la plataforma en términos de sus objetivos educativos y de investigación.

El proyecto en [15], introduce un sistema de navegación basado en visión artificial diseñado para un robot móvil. Este sistema se compone de un robot diferencial, un sistema de visión monocular y un mapa que incluye áreas libres y obstáculos, generado a partir de los datos captados por el sistema de visión. La realización de este proyecto implica la construcción de la plataforma diferencial y la configuración del entorno. Además, se lleva a cabo la implementación de los algoritmos de búsqueda A Estrella y Dijkstra para determinar la ruta óptima hacia un punto de destino. Se incorporan algoritmos de control automático para seguir la trayectoria establecida, y se valida el método en un entorno controlado. Las señales de control de posición y orientación se normalizan en un rango de -1 a 1 y se transmiten de forma inalámbrica a través de un protocolo TCP utilizando un microcontrolador ESP8266. Este enfoque representa una solución innovadora en la navegación de robots móviles utilizando visión artificial y control remoto.

En la investigación de [16], presenta un robot móvil que utiliza visión artificial para detectar el uso de mascarillas y medir la temperatura corporal de las personas. El robot se desplaza con tres ruedas, utiliza una Raspberry Pi y Python para el procesamiento de imágenes, y busca abordar la propagación de enfermedades respiratorias en áreas concurridas. Los resultados incluyen un robot de 59.74 kg con 8 horas de duración de batería, y la automatización de la detección de mascarillas y temperatura para garantizar la seguridad de las personas en entornos compartidos.

En el estudio que presenta [17], se analizan diversos métodos empleados para la estimación de la posición en robots móviles y vehículos autónomos. Se comienza por presentar el problema de la estimación, diferenciando entre estimadores que no dependen de la adquisición e interpretación de información del entorno (estimadores explícitos de posición) y aquellos basados en la percepción del entorno. En el primer grupo se incluyen métodos como la odometría, sistemas de navegación inercial, estimación absoluta de la posición mediante sistemas de posicionamiento global (GPS) y triangulación con respecto a balizas ambientales. Por otro lado, los estimadores basados en la percepción del entorno se dividen en el empleo de sensores activos (como láser y sonares) y sensores pasivos (como cámaras de vídeo). Además, se exploran técnicas alternativas para establecer la correspondencia entre los datos del entorno, diferenciando entre aquellas que se basan en la

extracción previa de características y los métodos icónicos. Este análisis proporciona una visión amplia de las herramientas y enfoques disponibles para abordar el desafío de la estimación de la posición en sistemas autónomos.

Se presenta en [18], el desarrollo de un sistema de control destinado a un robot móvil de configuración diferencial, con el propósito de seguir una trayectoria específica. Para abordar este desafío, se inicia implementando la cinemática directa del robot para simular su comportamiento. Luego, con el fin de alcanzar el objetivo deseado, se diseñan y aplican dos sistemas de control distintos: uno holonómico y otro no-holonómico, derivados de la cinemática inversa. Posteriormente, se llevan a cabo pruebas exhaustivas para comparar el rendimiento de ambos controladores y determinar cuál de ellos ofrece la solución óptima para el problema planteado. Este enfoque sistemático permite evaluar y seleccionar la estrategia de control más adecuada para el seguimiento preciso de la trayectoria por parte del robot móvil, contribuyendo así al avance en el campo de la robótica y la automatización.

El trabajo en [19] introduce el robot TraxBot y se detalla su completa integración en el Sistema Operativo de Robots (ROS). El TraxBot, una plataforma de robótica móvil concebida y ensamblada en el Instituto de Sistemas y Robótica (ISR) Coimbra, se presenta en como el foco central de este trabajo. Su propósito principal es agilizar el proceso de desarrollo, ofreciendo una abstracción del hardware y modos de operación intuitivos. Esto permite que los investigadores concentren sus esfuerzos en sus principales áreas de interés, como la búsqueda y rescate con múltiples robots o la robótica de enjambres. Se destacan las capacidades del TraxBot en combinación con un controlador de ROS específicamente diseñado, que facilita la utilización de diversas herramientas para el análisis de datos y la interacción entre múltiples robots, sensores y dispositivos de teleoperación. Para validar la eficacia del sistema, se llevaron a cabo pruebas experimentales exhaustivas utilizando tanto robots reales como virtuales. Este enfoque de validación integral garantiza la fiabilidad y funcionalidad del sistema TraxBot dentro del entorno operativo del ROS, proporcionando así una herramienta valiosa para investigaciones futuras en robótica y áreas relacionadas.

2.3. Bases teóricas

En el ámbito de la robótica, la visión artificial ha emergido como una disciplina clave que permite a las máquinas interpretar y comprender su entorno a través del procesamiento de imágenes y datos visuales. Desde sus inicios en la década de 1960, con la investigación en reconocimiento de patrones y procesamiento de imágenes digitales, esta tecnología ha evolucionado significativamente gracias a los avances en hardware y algoritmos computacionales [20]. El aprendizaje profundo y el aumento en la capacidad de procesamiento han permitido desarrollar sistemas de visión más precisos y eficientes. Su aplicación en la robótica móvil ha impulsado la creación de robots autónomos capaces de realizar tareas de navegación, manipulación y reconocimiento de objetos en entornos dinámicos, contribuyendo al desarrollo de soluciones avanzadas en diversas áreas industriales, médicas y de seguridad.

Desde sensores de proximidad, presión, fuerza y torque, hasta acelerómetros y giroscopios, cada uno de estos dispositivos desempeña un rol en la recopilación de datos, que están orientados hacia diversas finalidades, como medición, cuantificación, mapeo, localización, optimización o detección. La toma rápida de decisiones, fundamentada en el procesamiento de estas variables mediante esta tecnología, se postula como una alternativa a la espera de un protocolo de conexión con un servidor encargado de las tareas más intensas. En lugar de depender de dichos servicios, se aprovecha la tecnología de procesamiento actual para una respuesta más eficiente [21].

2.3.1. Robótica móvil

Los robots existen en diversas formas y cumplen una variedad de aplicaciones. Aunque los robots móviles son bastante comunes, apenas exploran la diversidad de robots que se pueden construir [22]. La robótica móvil es una de las ramas más dinámicas y avanzadas dentro del campo de la robótica, ya que permite la creación de sistemas autónomos capaces de desplazarse e interactuar con su entorno sin intervención humana directa. Su importancia radica en su amplia aplicación en sectores como la exploración espacial, la automatización industrial, la logística, la salud y la seguridad, donde los robots deben operar en entornos no estructurados y en constante cambio. Para lograr un desplazamiento eficiente y una navegación precisa, la robótica móvil integra múltiples disciplinas como el control de

movimiento, la percepción del entorno y la visión por computadora. En este contexto, la visión artificial juega un papel crucial al permitir que los robots interpreten información visual en tiempo real, rendericen mapas, detecten obstáculos y optimicen sus trayectorias, mejorando su capacidad de autonomía y toma de decisiones. Gracias a los avances en procesamiento de imágenes, los robots móviles han evolucionado para realizar tareas complejas con mayor precisión y eficiencia.

Además, se instituye como un campo interdisciplinario que amalgama la ingeniería mecánica, la informática, la ingeniería eléctrica, la psicología cognitiva, la percepción y la neurociencia, así como la mecatrónica, en una disciplina integrada que busca el desarrollo de robots capaces de interactuar con su entorno de manera autónoma, inteligente y adaptativa.

La ingeniería mecánica sienta las bases para diseñar chasis robustos y sistemas de movimiento eficientes, mientras que la informática y la ingeniería asumen la responsabilidad de programar y controlar dichos dispositivos. La ingeniería eléctrica ofrece soluciones para la integración de sensores y energía, mientras que la psicología cognitiva y la neurociencia influyen en la creación de algoritmos que imitan los procesos cognitivos humanos. En última instancia, la mecatrónica representa la convergencia de todas estas disciplinas, buscando la integración sinérgica de hardware y software para generar sistemas robóticos móviles eficientes y versátiles [22]. La colaboración entre estos campos ha propiciado avances significativos en el desarrollo de robots móviles capaces de ejecutar tareas complejas en entornos diversos y dinámicos.

2.3.2. Robot operating system

El entorno de desarrollo, con de nombre sistema operativo para robots (ROS) por sus siglas en inglés, es una plataforma de código abierto diseñado específicamente para aplicaciones robóticas. Su principal objetivo es facilitar el desarrollo de software para robots mediante una arquitectura modular que permite la comunicación eficiente entre distintos componentes, como sensores, actuadores y sistemas de procesamiento [23].

Su estructura se basa en nodos y tópicos, donde cada nodo representa una tarea específica, y los tópicos actúan como canales de comunicación para el intercambio de información [23]. Esto permite que sensores, actuadores y algoritmos trabajen de manera

conjunta y modular. Además, ROS ofrece herramientas para la simulación, visualización y depuración, lo que facilita el diseño, implementación y prueba de sistemas robóticos complejos.

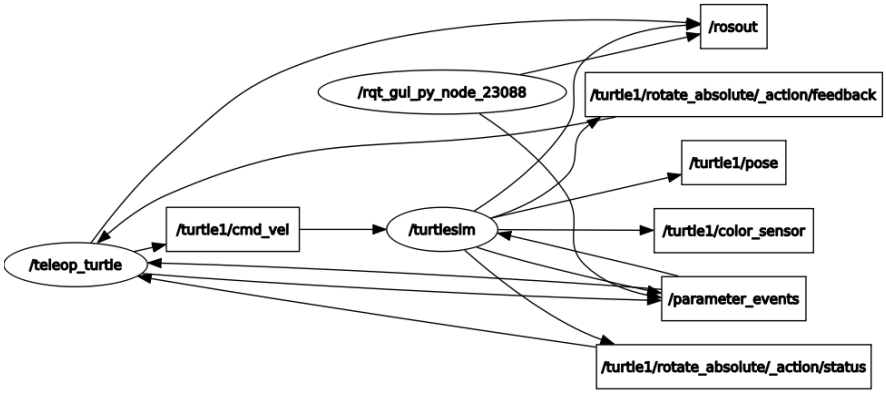


Fig. 2.1 Ejemplo de un gráfico de nodos y tópicos en ROS 2.

Esta plataforma proporciona un conjunto de herramientas y bibliotecas que soportan funcionalidades clave, como la gestión de mensajes, la visualización de datos, la simulación y el control de robots [23]. Entre estas herramientas se destacan Gazebo y RViz, que ayudan en el desarrollo de aplicaciones robóticas. Gazebo es un simulador robusto que permite recrear entornos físicos de manera realista, incluyendo la interacción entre sensores, robots y el entorno. Es especialmente útil para probar y validar algoritmos antes de implementarlos en robots reales, reduciendo costos y riesgos. Por otro lado, RViz es una herramienta de visualización que facilita la interpretación de los datos generados por los sensores del robot, la validación de modelos URDF que incluye las características para la visualización, como mallas 3D o formas geométricas simples, que permiten mostrar el robot de manera gráfica dentro de una simulación y la representación gráfica del entorno percibido, lo cual es esencial para tareas como la navegación y la planificación de trayectorias [23].

Gracias a su diseño escalable y a su amplia comunidad de usuarios, ROS se ha convertido en un estándar dentro del desarrollo de robots autónomos, permitiendo la integración de hardware diverso y la implementación de algoritmos avanzados en aplicaciones tanto reales como simuladas.

2.3.3. Visión artificial para la robótica

La visión artificial en la robótica es una disciplina que permite a los robots percibir, interpretar y responder a su entorno mediante el procesamiento de imágenes y datos visuales. Esta tecnología se basa en algoritmos avanzados de procesamiento de imágenes, aprendizaje automático y reconocimiento de patrones para extraer información relevante del entorno. Su integración en sistemas robóticos mejora la capacidad de navegación, manipulación de objetos y toma de decisiones autónoma, lo que resulta fundamental en aplicaciones como la automatización industrial, la robótica móvil y la interacción humano-robot [24].

2.3.3.1. Visión artificial y ROS

La integración de estos sistemas que permiten la comunicación eficiente entre sensores y algoritmos de procesamiento de imágenes. El uso de bibliotecas como para el análisis de imágenes y el manejo de nubes de puntos posibilita la captura y procesamiento de información visual en tiempo real, facilitando la detección de obstáculos, el reconocimiento de objetos y la interpretación del entorno [25]. A través de una arquitectura modular, los datos provenientes de cámaras y sensores de profundidad pueden ser utilizados para generar representaciones precisas del espacio circundante, mejorando la capacidad de los robots para navegar y tomar decisiones de manera autónoma. Esta integración optimiza la eficiencia de los sistemas de percepción, permitiendo un desempeño más robusto en aplicaciones de interacción con el entorno [25].

El uso de visión por computadora con el entorno de desarrollo robótico se logra mediante la implementación de nodos que capturan, procesan y transmiten información visual en tiempo real. Para ello, se emplean suscriptores y publicadores que gestionan flujos de datos provenientes de cámaras y sensores de profundidad, permitiendo aplicar técnicas de filtrado, segmentación y reconocimiento de patrones [25]. La estructura modular facilita la comunicación entre procesos, permitiendo que las imágenes capturadas sean transformadas y analizadas antes de ser utilizadas en tareas como el seguimiento de objetos o la navegación asistida. A través de la configuración de nodos específicos y la optimización de parámetros en la comunicación entre módulos, es posible lograr un procesamiento eficiente que maximice el desempeño del sistema de visión artificial [24].

2.3.3.2. Cámaras de profundidad

Las cámaras de profundidad están diseñadas para asociar una distancia o profundidad a cada píxel de la imagen capturada. Existen diversas tecnologías que permiten su funcionamiento; sin embargo, las más relevantes se pueden clasificar en tres principales categorías: luz estructurada, tiempo de vuelo y visión estéreo [26].

La tecnología de luz estructurada opera proyectando un patrón infrarrojo sobre el entorno y analizando las distorsiones que se generan en dicho patrón mediante una cámara. A partir de estas distorsiones, se determina la profundidad de los objetos en la escena. Un ejemplo representativo de esta tecnología es el sensor Kinect de la Xbox 360 (versión 1).

El método de tiempo de vuelo se basa en emitir pulsos de luz infrarroja hacia el entorno y se mide el tiempo que tardan en regresar a la cámara después de reflejarse en los objetos. A partir de este tiempo de retorno, se calcula la profundidad [26].

Para el enfoque de la visión estéreo, inspirado en el sistema de visión binocular humano, emplea dos cámaras situadas a una distancia ligeramente separada entre sí. Ambas cámaras capturan imágenes de un mismo entorno desde ángulos diferentes, y al comparar las diferencias entre ellas, se puede estimar la profundidad de los objetos en la escena. Esta tecnología ha sido implementada en diversas cámaras diseñadas para aplicaciones de percepción tridimensional.

Fig. 2.2 Nube de puntos de la cámara Helios2 ToF.

La principal diferencia entre los dos primeros métodos (luz estructurada y tiempo de vuelo) y la visión estéreo radica en que los primeros emplean una única cámara junto con una fuente de luz activa para obtener información de profundidad, mientras que la visión

estéreo se basa en dos cámaras sin necesidad de una fuente de luz adicional. Debido a esta diferencia fundamental, los primeros sistemas son conocidos como monoculares (ya que utilizan una sola cámara), mientras que el último se denomina estéreo, dado el uso de dos cámaras para la percepción tridimensional [26].

La **Fig. 2.3** muestra una simulación en Gazebo de una cámara de profundidad que captura imágenes de un cubo en un entorno tridimensional. En la parte inferior izquierda se observa un modelo 3D del cubo y el campo de visión de la cámara, representado con líneas azules. En la esquina superior derecha, se presentan las imágenes capturadas por la cámara estéreo, donde se aprecian las dos perspectivas distintas (izquierda y derecha) necesarias para estimar la profundidad utilizando técnicas de visión estéreo.

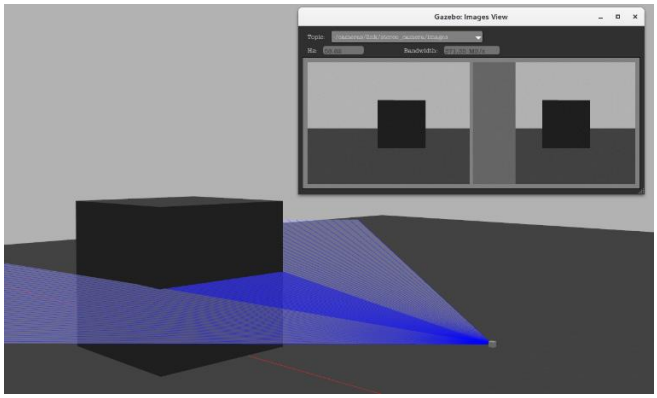


Fig. 2.3 Simulación de una cámara de profundidad en Gazebo.

2.4. Componentes del sistema

En esta sección se presentará una descripción detallada y completa de los componentes utilizados durante la realización de este proyecto, destacando sus características, funciones y su importancia dentro del sistema de visión artificial implementado.

2.4.1. Hardware empleado

2.4.1.1. Cámara kinect

La cámara Kinect de la Xbox 360 es un dispositivo de captura de movimiento y profundidad que combina una cámara RGB, un proyector de infrarrojos y un sensor de detección de tiempo de vuelo. Utiliza la proyección de un patrón infrarrojo y su posterior

análisis para determinar la distancia de los objetos en su campo de visión, permitiendo la generación de mapas tridimensionales del entorno [27].

Las especificaciones y características del sensor de profundidad se observan a detalle en la **Tabla 1**. Además, se destacan los rangos de detección óptimos en los que el sensor opera de manera eficiente. Estos parámetros están estrechamente relacionados con los límites prácticos y físicos de captura, los cuales se pueden observar en las **Fig. 2.4** y **Fig. 2.5**.

Tabla 1: Especificaciones de la cámara kinect.

Especificación	Detalles
Resolución	640 × 480 píxeles a 30 FPS
Sensor de profundidad	Tecnología de luz estructurada con proyector de infrarrojos
Campo de visión óptimo	57° horizontal, 43° vertical
Rango de detección óptimo	1.2 m a 3.5 m
Rango de valores	0 a 4095 (12 bits), donde 0 = sin detección y 4095 = distancia máxima (4 m)

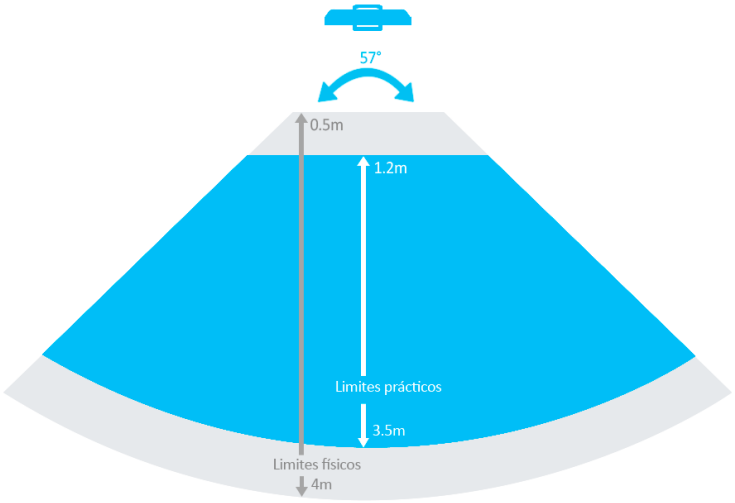


Fig. 2.4 Rangos de visión sensor kinect.

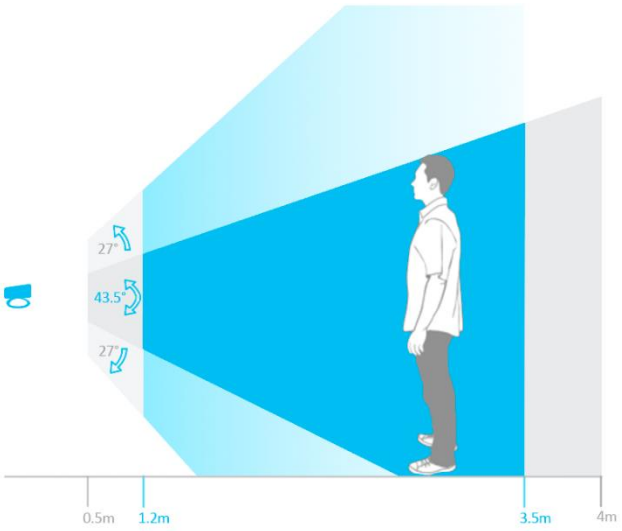


Fig. 2.5 Ángulos de visión del sensor Kinect.

Los componentes que están integrados en el sensor vienen detallados en la **Fig. 2.6**, donde se identifican sus principales elementos.

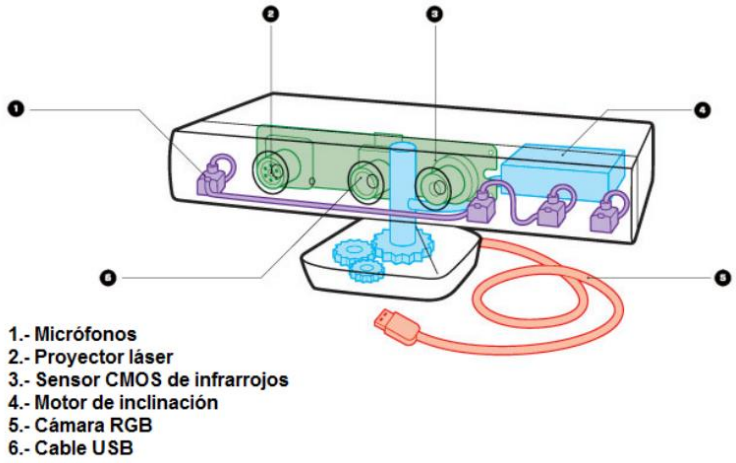


Fig. 2.6 Partes del sensor Kinect.

2.4.1.2. Jetson nano

La Jetson Nano Developer Kit es una plataforma de cómputo embebida diseñada por NVIDIA para desarrollar proyectos de visión artificial y robótica. Gracias a su capacidad para ejecutar algoritmos complejos de visión artificial, resulta ideal para integrar con sistemas operativos para la creación de robots, proporcionando una base sólida para la recolección, procesamiento y análisis de datos en tiempo real [28].

Tabla 2: Especificaciones de la Jetson Nano 2 GB Developer Kit.

Especificación	Detalles
Procesador	Quad-core ARM Cortex-A57 a 1.43 GHz
GPU	NVIDIA Maxwell con 128 núcleos CUDA
Memoria RAM	2 GB LPDDR4
Almacenamiento	microSD (requiere tarjeta para el sistema operativo)
Interfaz de Red	Gigabit Ethernet
Puertos USB	1x USB 3.0, 2x USB 2.0
Salida de Video	HDMI 2.0, DisplayPort 1.2
Soporte de Cámara	1x conector CSI (MIPI CSI-2 de 15 pines)
Interfaz de GPIO	40 pines (compatible con Raspberry Pi)
Energía	5V/3A vía USB-C o 5V/4A vía pines GPIO
Dimensiones	100 mm x 80 mm x 29 mm

Fig. 2.7 Jetson nano 2 GB developer kit.

2.4.2. Drivers y librerías

Un driver es un software que actúa como un intermediario entre el sistema operativo y el hardware, permitiendo que la plataforma de cómputo embebida reconozca y controle un periférico de manera adecuada. Sin un driver, el sistema no podría interpretar las señales del dispositivo ni interactuar con él de forma eficiente. Por otro lado, una librería es un conjunto de funciones y herramientas predefinidas que facilitan el desarrollo de software, proporcionando métodos optimizados para realizar tareas específicas, como el procesamiento de imágenes o la comunicación con dispositivos. Aunque cumplen funciones distintas, ambos son esenciales. El driver garantiza que el hardware sea accesible para el sistema, mientras que la librería simplifica la implementación de algoritmos y la gestión de datos, asegurando un rendimiento eficiente y una integración adecuada en aplicaciones avanzadas.

2.4.2.1. Librería libfreenect

Esta biblioteca de código abierto, desarrollada por la comunidad *OpenKinect*, está diseñada para habilitar el uso del sensor Kinect de Microsoft en sistemas operativos como Linux. Su principal función es proporcionar control a nivel de usuario sobre el hardware del Kinect, permitiendo el acceso a sus componentes principales, como la captura de imágenes RGB y de profundidad, el control de motores, la lectura del acelerómetro, la gestión del LED y la transmisión de audio [29]. Este proyecto surgió para superar las limitaciones impuestas por el uso exclusivo del dispositivo en consolas Xbox, facilitando su integración en aplicaciones de investigación, robótica, visión por computadora y prototipado experimental [30].

2.4.2.2. Controlador kinect_ros2

El repositorio, desarrollado por *fadlio*, es un paquete de software diseñado para integrar el sensor Kinect versión 1 (Xbox 360) en entornos de ROS 2, facilitando su uso en aplicaciones de robótica y visión por computadora. Su función principal es actuar como un driver que permite la comunicación entre el hardware del Kinect y ROS 2, publicando datos de imágenes RGB y de profundidad, junto con información de calibración de la cámara. Este proyecto se basa en *libfreenect*, una librería de código abierto para interactuar con el Kinect [31].

Capítulo III

Marco metodológico

3.1. Enfoque de la investigación

En el contexto de este trabajo de integración curricular, se lleva a cabo una investigación aplicada con el objetivo de proporcionar una solución al problema planteado, que involucra el diseño de un sistema de visión artificial para un robot móvil de alto torque, que le permita tener claro el entorno en el que se va a desempeñar.

Para abordar este objetivo, se opta por un enfoque de investigación documental, donde se recopila información de fuentes secundarias como libros, revistas y artículos científicos. Esta metodología permite la identificación, selección y organización de información relevante para la elaboración del documento y la investigación. Simultáneamente, se emplea una investigación descriptiva para definir las características de las variables del estudio y evaluar las dimensiones del contexto. La combinación de estas metodologías impulsa el progreso del proyecto al facilitar la recolección de información esencial con bases sólidas y bien fundamentadas.

En la fase final, se lleva a cabo una investigación experimental, pues mediante la corrección y pruebas, evaluamos los comportamientos del sistema y realizamos ajustes según sea necesario. Este enfoque permite la observación de los efectos en otras variables dependientes en condiciones controladas, a través de un análisis exhaustivo del rendimiento.

3.2. Diseño de la investigación

La implementación de un sistema de visión artificial requiere depender de distintas fases divididas, las cuales deben cumplir y validar el resultado final, estando directamente relacionadas con el logro de los objetivos propuestos.

3.2.1. Fase 1: Búsqueda e identificación de componentes usados para sistemas de visión por computador

En esta etapa, se fundamenta en una investigación respaldada por los conocimientos adquiridos a lo largo de la carrera para identificar y seleccionar los componentes que se emplearán en el desarrollo del proyecto.

Actividad 1.1: "Definir componentes a través de indagación"

En esta actividad, nos centraremos en la búsqueda de información de los componentes que se han venido usando para el desarrollo de sistemas de visión artificial para robots.

Actividad 1.2: " Investigación de componentes en el mercado"

Basándose en una investigación y consultas realizadas previamente, se identifican y establecen los posibles componentes que certifiquen el funcionamiento del sistema. Esto está sujeto a cambios debido a que se deben considerar el diseño del algoritmo y su programación.

Actividad 1.3: "Adquisición de componentes"

Una vez realizadas las investigaciones pertinentes, se procede a la adquisición de los componentes fundamentales para el desarrollo del sistema de visión artificial. Tras asegurarse de que cada uno ha sido evaluado en función de su propósito y funcionamiento.

3.2.2. Fase 2: Diseño de un algoritmo de visión artificial

Esta fase, nos centramos en el desarrollar un algoritmo para el sistema de visión artificial.

Actividad 2.1: "Determinamos las especificaciones del sistema"

Se determinan las características de los componentes con el fin de establecer las especificaciones técnicas que regirán el funcionamiento del sistema. Este proceso permite obtener una visión clara sobre el comportamiento esperado del sistema y evaluar su

desempeño. De esta manera, se asegura que el diseño y la implementación del sistema estén alineados con los objetivos planteados, garantizando su correcto funcionamiento durante la operación.

Actividad 2.2: "Instalación y configuración del entorno trabajo"

Establecer una plataforma que permita integrar y desplegar todos los elementos necesarios para el desarrollo del proyecto. A través de este proceso, se garantiza la disponibilidad de un entorno adecuado que facilite la implementación y ejecución de las distintas fases del proyecto, asegurando su viabilidad y eficiencia. De esta manera, se sientan las bases para el cumplimiento de los objetivos planteados y el éxito del proyecto en su conjunto.

Actividad 2.3: "Compatibilidad entre dispositivos"

Garantizar la compatibilidad entre los dispositivos involucrados en el sistema. Para esto, es necesario instalar y configurar un controlador compatible que permita la lectura de los datos capturados y su posterior publicación o visualización en el entorno de trabajo.

3.2.3. Fase 3: Pruebas y análisis

Actividad 3.1: " Ejecución de pruebas funcionales"

Realizar pruebas del sistema para evaluar su funcionamiento. Se registrarán métricas, como latencia, tasas de fotogramas por segundo y frecuencia de actualización. Esto permitirá identificar límites del sistema y su adaptabilidad.

Actividad 3.2: "Analizar los resultados obtenidos"

Evaluar los datos recolectados durante las pruebas funcionales, identificando patrones y discrepancias.

Actividad 3.3: "Desarrollo del documento correspondiente al trabajo de titulación"

Elaborar el trabajo que sintetice la investigación realizada, configuración y montaje del sistema y los resultados obtenidos. Además, se propondrán mejoras específicas, para un trabajo a futuro. Esta actividad sentará las bases para integraciones futuras o la escalabilidad del sistema.

Capítulo IV

Desarrollo y análisis

A continuación, se presentan los resultados obtenidos tras implementar los objetivos propuestos en este trabajo. Esta sección incluye, el diseño del algoritmo de visión artificial, su montaje, configuración e integración, así como la interpretación de resultados previos a las pruebas realizadas.

4.1. Algoritmo

El desarrollo del algoritmo de visión artificial se lleva a cabo simulando el funcionamiento del sistema, antes de poder implementarlo en la tarjeta electrónica. Se emplea una imagen con información de profundidad, que simula una capturada por el sensor Kinect. Posteriormente, dicha información se transforma a una nube de puntos 3D.

En la **Fig. 4.1** se ilustra el proceso para transformar las imágenes de profundidad capturadas por el sensor a coordenadas en un plano en tres dimensiones.

Este algoritmo es el que posteriormente se debe implementar en la tarjeta Jetson para lograr el reconocimiento de los alrededores. De esta manera, se consigue trasladar la información exterior a datos que pueden ser interpretados por el sistema. Al configurar correctamente las distancias focales y calibrar adecuadamente la cámara, se puede obtener con una precisión muy alta una representación en tres dimensiones de todo lo que rodea al robot.

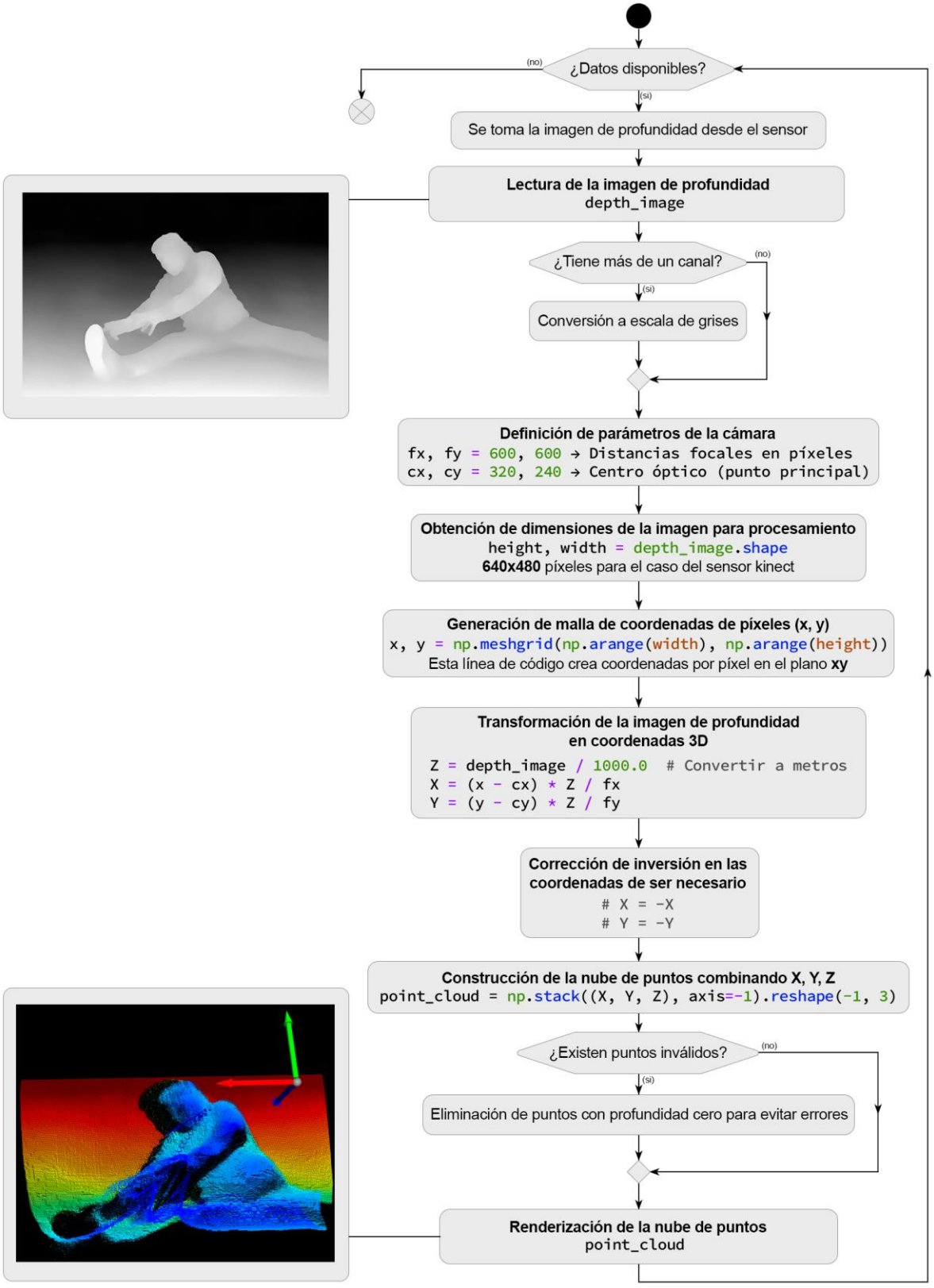


Fig. 4.1 Diagrama de flujo del algoritmo de visión artificial.

4.1.1. Transformación de la imagen de profundidad en coordenadas 3D

Para entender mejor el algoritmo, se destacan aspectos clave, el primero es que el código al leer la imagen lo convierte a una matriz de tamaño 640 x 480, que es el número de píxeles que tiene la imagen tanto de alto como de ancho y lo guarda en una variable llamada *depth_image*, y para el ejemplo en la **Fig. 4.1** equivale a

$$D = \begin{pmatrix} 5 & 6 & \cdots & 3 & 3 \\ 5 & 6 & \cdots & 3 & 3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 241 & 242 & \vdots & 238 & 242 \\ 241 & 242 & \vdots & 242 & 244 \end{pmatrix}_{640\times480} , \quad (1)$$

los valores de la matriz D representan la información de profundidad que proporciona el sensor, donde el valor 0 indica que no se pudo determinar la profundidad (por ejemplo, debido a superficies reflectantes, absorbentes de IR o fuera de rango) y el valor 4095 representa la profundidad máxima medible (4 metros) como se especifica en la **Tabla 1**.

Para obtener los puntos en la coordenada Z se aplica

$$Z = \frac{D}{1000} , \quad (2)$$

donde Z son las coordenadas en el eje del mismo nombre, a la matriz que simula los datos obtenidos por el sensor, se le aplica un factor de conversión debido a que los datos que entrega el sensor están en milímetros, esto es necesario debido a que ROS trabaja en metros.

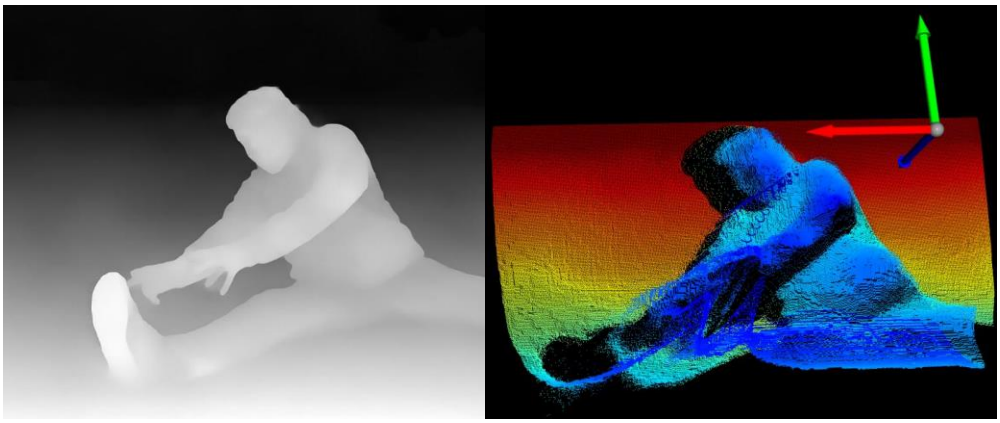
Para calcular los puntos en las coordenadas tanto en *x* como en *y*, se realiza

$$X = (x - c_x)\frac{Z}{f_x} , \quad (3)$$

$$Y = \left(y - c_y\right)\frac{Z}{f_y} , \quad (4)$$

donde x, y son las coordenadas horizontal y vertical del píxel respectivamente, c_x, c_y son las coordenadas del centro óptico de la imagen y f_x, f_y son las distancias focales vertical y horizontal respectivas del sensor.

La imagen que muestra la **Fig. 4.2** evidencia la imagen de entrada, donde las zonas más claras representan áreas más cercanas y las más oscuras, áreas más lejanas. Esta imagen es monocromática y contiene información de la distancia de cada punto a la cámara. La imagen de salida es una nube de puntos 3D, donde cada punto representa una posición en el espacio, coloreada según su profundidad, lo que permite visualizar la forma del objeto capturado con una densidad de uno a uno (un punto por píxel).



(a) (b)

Fig. 4.2 Entrada y salida del algoritmo de visión artificial. (a) Imagen de entrada con información de profundidad. (b) Representación 3D de la salida.

4.2. Montaje e integración

Para integrar y montar el algoritmo de visión con la plataforma robótica y lograr su funcionamiento adecuado, se inicia con la preparación del hardware y software necesarios para la implementación del sistema de visión artificial. Este proceso incluye la configuración de la imagen en la tarjeta SD, la instalación del sistema operativo compatible con la Jetson Nano y la implementación de la plataforma de desarrollo, en este caso, ROS 2.

Además, es necesario hacer uso de un controlador, que sea capaz de procesar los datos capturados por la cámara Kinect. Así, la configuración consiste en tres nodos, donde el principal */kinect/kinect_ros2* se encarga de la publicación de mensajes relacionados con los tópicos más importantes, como la nube de puntos.

4.2.1. Instalación del sistema operativo

En esta etapa, se establece el entorno base sobre el cual se ejecutarán las aplicaciones y algoritmos desarrollados. Para ello, se emplea *Ubuntu 20.04 LTS*, junto con el *JetPack*, un conjunto de herramientas desarrollado por *NVIDIA* que incluye las dependencias necesarias para el procesamiento de visión artificial, como *OpenCV*.

Para este proceso, se usa el software *Balena Etcher* para grabar la imagen de Linux en una tarjeta SD de 64 GB de almacenamiento. Este software facilita la creación de las particiones necesarias para garantizar el correcto almacenamiento y funcionamiento del sistema operativo.

4.2.2. Conexiones físicas

La comunicación entre el periférico y la plataforma de cómputo embebida se establece a través de una interfaz USB, que internamente opera mediante comunicación serial. Este tipo de transmisión implica un flujo de datos secuencial entre el dispositivo y la plataforma, lo que introduce una latencia inherente en la adquisición y procesamiento de la información. La imagen en la **Fig. 4.3** muestra un diagrama de conexión para un sensor Kinect, detallando los componentes necesarios para su instalación y funcionamiento. Incluye el cable de datos, el adaptador del Kinect, un adaptador USB 3.0 y una fuente de poder (batería de 12 V) para el adaptador.

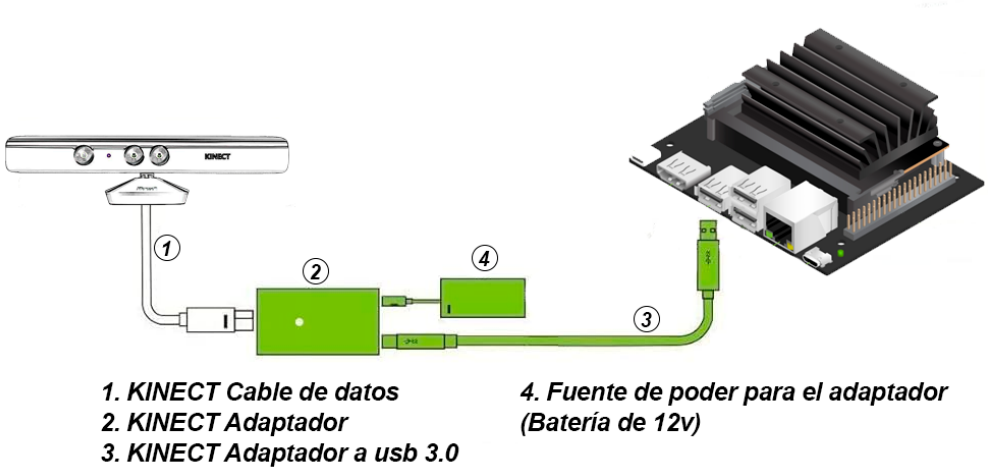


Fig. 4.3 Diagrama de conexión del sensor Kinect.

4.2.3. Configuración en ROS 2

Para configurar el sistema e integrar el algoritmo diseñado, se instala en el sistema los controladores y librerías para reconocer el sensor Kinect. Al ejecutar los comandos que se muestran en la **Fig. 4.4** se realiza la instalación completa del sistema.

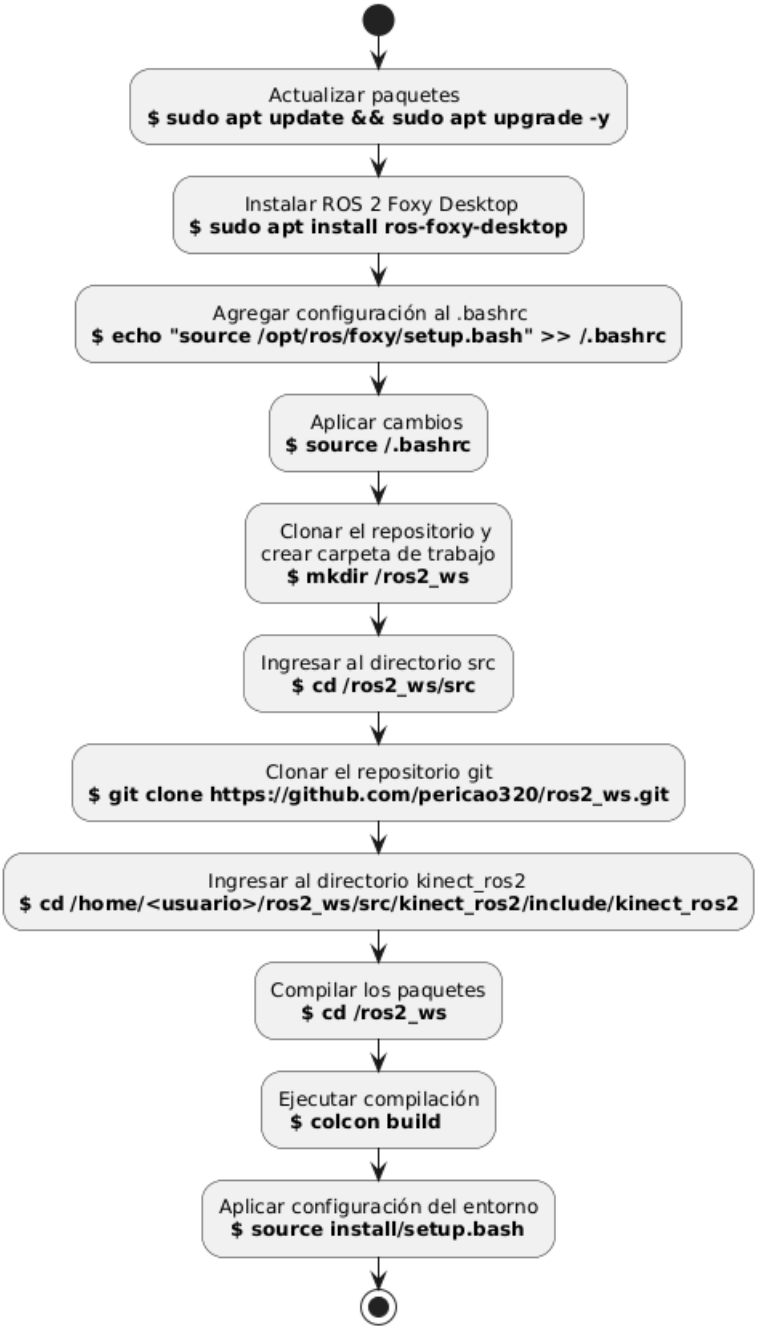


Fig. 4.4 Diagrama de instalación y configuración del sistema.

4.2.3.1. Nodos y tópicos del sistema

Dado que el sensor Kinect proporciona imágenes RGB junto con un mapa de profundidad, la transferencia de estos datos requiere un procesamiento adicional antes de ser utilizados en la aplicación de visión artificial, y de esto se encarga nuestro controlador principal el cual permite que los datos del sensor sean publicados en tópicos para su utilización en otras aplicaciones. Este driver encapsula las funciones de adquisición de imágenes y datos de profundidad, facilitando la integración con ROS 2 a través de nodos que publican información estructurada.

Tabla 3: Configuración de los nodos y tópicos del sistema.

Nodos	Tópicos	Descripción
/kinect/kinect_ros2	/kinect/depth/camera_info	Proporciona información de los datos de profundidad.
	/kinect/depth/image_raw	Publica imágenes RGB capturadas por la cámara corregidas.
	/kinect/image_raw	Publica imágenes RGB capturadas por la cámara en crudo.
	/kinect/points	Publica datos de nubes de puntos generados por Kinect.
/rviz2	/rviz	Nodo principal de RViz2, herramienta de visualización para ROS 2.
/static_tf_pub	/tf	Publica transformaciones dinámicas entre marcos de referencia.
	/tf_static	Tópico encargado de publicar la matriz de corrección con respecto a la referencia

La imagen de la **Fig. 4.5** muestra cómo queda el diagrama que representa el flujo de comunicación entre los diferentes tópicos y nodos en un sistema basado en ROS 2. Los tópicos están organizados jerárquicamente, destacando aquellos relacionados con la cámara Kinect y su procesamiento de datos. Se observa cómo algunos tópicos interactúan directamente con nodos específicos, mientras que otros permanecen aislados.

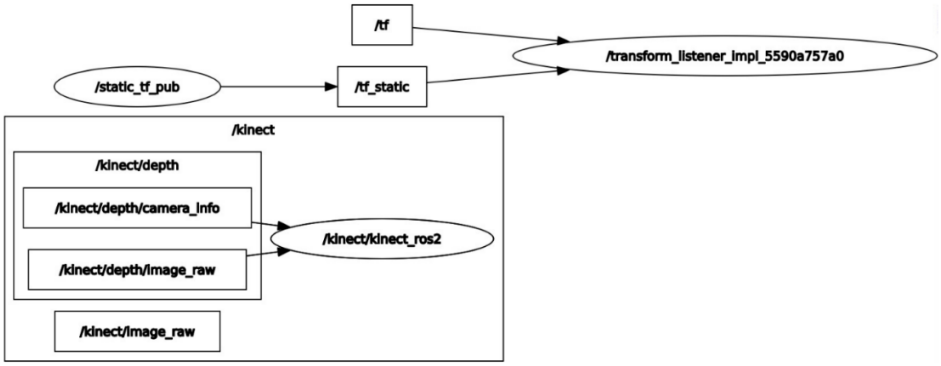


Fig. 4.5 Diagrama de flujo de tópicos y nodos extraídos del sistema.

El tópico */kinect/image_raw* no está conectado a ningún nodo, ya que contiene las imágenes en crudo capturadas por la Kinect. Estas imágenes no han sido procesadas, por lo que no son directamente utilizables en el sistema. Por otro lado, el tópico */kinect/depth/image_raw* está conectado y en uso dentro del sistema, ya que las imágenes que publica han sido procesadas mediante una matriz de distorsión y rectificación. El nodo */kinect/kinect_ros2* contiene el algoritmo de visión artificial que recibe y procesa la transformación de la imagen de profundidad a nube de puntos, este se encarga de publicar los tópicos necesarios de manera constante hasta que el programa se detenga.

4.2.3.2. Estructura de mensajes

En ROS 2, los mensajes son la unidad de comunicación entre nodos y permiten el intercambio de datos a través de tópicos. Los mensajes están predefinidos en paquetes como *sensor_msgs* y contienen información específica para cada tipo de dato. En visión artificial, los mensajes estructuran datos como imágenes, parámetros de calibración de cámaras y nubes de puntos en 3D. Cada mensaje tiene una estructura definida con distintos tipos de datos, lo que permite una interpretación precisa de la información capturada por sensores.

Cada mensaje está definido de manera estricta mediante un archivo de extensión *.msg*, aquí es donde fija el tamaño y tipo de dato que se está enviando, esto que asegura la compatibilidad y evita errores en la transmisión. Por ejemplo, como se indica en la **Tabla 4**, el paquete *sensor_msgs* proporciona mensajes de tipo *msg*/*Image* para manejar datos de cámaras o *msg/PointCloud2* para nubes de puntos en 3D, permitiendo su visualización.

Tabla 4: Tipos y estructuras de mensaje del controlador.

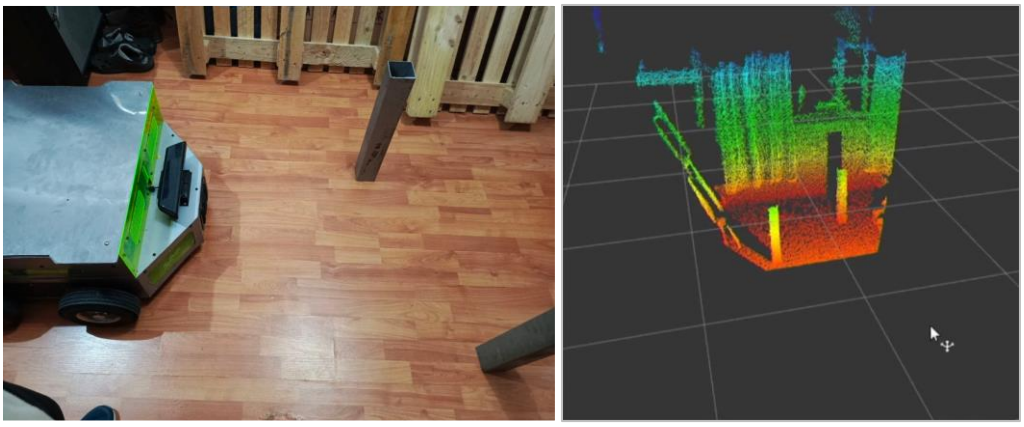
Tipo de mensaje	Estructura del mensaje	Descripción
sensor_msgs/msg/CameraInfo	std_msgs/Header header uint32 height uint32 width string distortion_model float64[] d float64[9] k float64[9] r float64[12] p uint32 binning_x uint32 binning_y RegionOfInteres	Información de calibración y parámetros de la cámara.
sensor_msgs/msg/Image	std_msgs/Header header uint32 height uint32 width string encoding uint8 is_bigendian uint32 step uint8[] data	Imagen en formato sin comprimir capturada por la cámara.
sensor_msgs/msg/PointCloud2	std_msgs/Header header uint32 height uint32 width PointField[] fields bool is_bigendian uint32 point_step uint32 row_step uint8[] data bool is_dense	Nube de puntos en 3D generada por el sensor de profundidad.

4.3. Resultados obtenidos

En esta sección se presentan los resultados obtenidos tras la implementación del sistema de visión artificial para el robot móvil de alto torque. Se detallan las pruebas realizadas para evaluar el desempeño del sistema en la captura, procesamiento y visualización de datos en un entorno tridimensional. Además, se analizan las métricas de rendimiento y retardo, destacando los alcances logrados en la interpretación del entorno por parte del robot. Estos resultados permiten validar el funcionamiento del sistema desarrollado y su potencial para futuras mejoras en el ámbito de la visión artificial aplicada a la robótica móvil.

4.3.1. Obtención y visualización de los datos de profundidad del sensor kinect

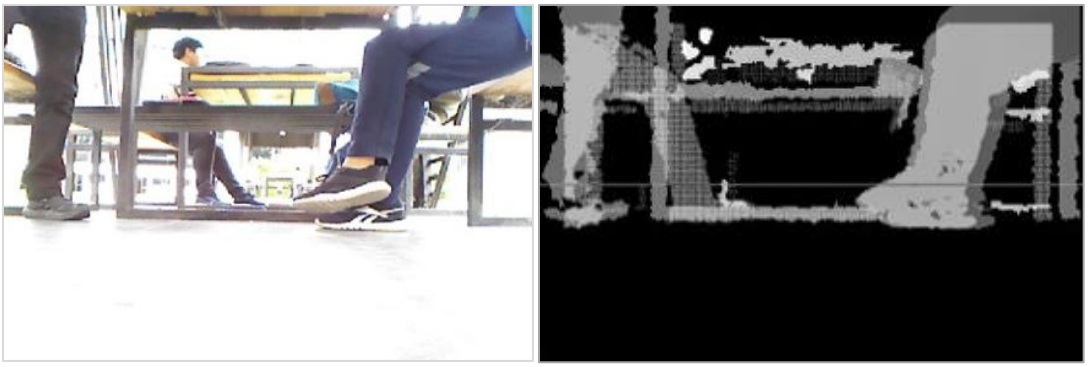
La **Fig. 4.6** muestra de lado izquierdo un entorno físico con el robot móvil sobre el piso, rodeado por elementos como tubos metálicos verticales, simulando un entorno de pruebas. En la derecha, se aprecia la representación tridimensional de este espacio capturada mediante la cámara kinect. Los datos visualizados incluyen puntos de diferentes colores que representan distancias relativas, donde tonos más cálidos como rojo y amarillo indican proximidad, y tonos más fríos como verde y azul representan objetos más lejanos. Esta visualización tridimensional permite interpretar la distribución espacial y los obstáculos en el entorno.



(a) (b)

Fig. 4.6 Datos de entorno capturados por la cámara montada en el robot móvil. (a) Vista del entorno físico con el robot y obstáculos. (b) Visualización en RViz entorno generado con la información de profundidad.

La **Fig. 4.7** se muestra una escena en la que se observa a personas sentadas bajo una mesa, con detalles visibles del entorno como los soportes de la mesa, el suelo y otros elementos cercanos. La representación de profundidad destaca las distancias relativas entre los objetos y las personas, utilizando una escala de grises donde tonos claros para los elementos más cercanos y tonos oscuros para los más lejanos.



(a) (b)

Fig. 4.7 Datos de imagen recibidos del sensor kinect. (a) Imagen RGB recibida del sensor kinect. (b) Información de profundidad recibida del sensor kinect.

En la **Fig. 4.8** se muestra un mapa generado en RViz con *slam_toolbox*, utilizando los datos de la nube de puntos. Esto permite identificar obstáculos y áreas navegables, representados en la cuadrícula como negro para obstáculos, blanco para zonas libres y azul atenuado para áreas no exploradas. Este procesamiento puede facilitar la interpretación del entorno y habilitará la planificación de rutas por parte del robot.

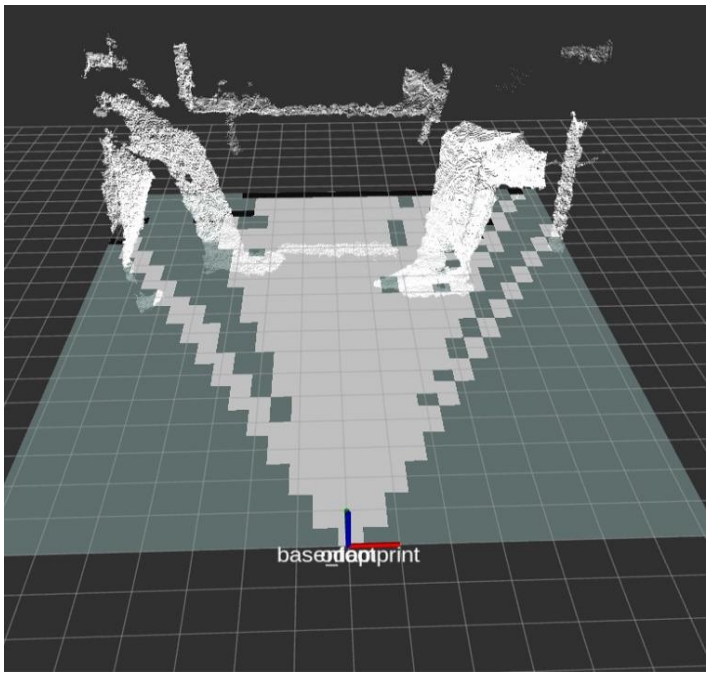


Fig. 4.8 Prueba de slam_toolbox a partir de datos de nube de puntos.

4.3.2. Evaluación de rendimiento del sistema

Acto seguido se analizan los tiempos de respuesta y las tasas de refresco del sistema, parámetros que permiten medir su desempeño durante la operación. Estos valores reflejan la rapidez con la que el sistema procesa la información capturada por los sensores y actualiza los datos en tiempo real. Además, se evalúa cómo estas métricas influyen en la interacción del sistema con su entorno, considerando su impacto en la fluidez y precisión de las tareas realizadas. Los resultados obtenidos se describen y analizan para comprender mejor el rendimiento general.

4.3.2.1. Tasas de refresco

El análisis de la frecuencia con la que se actualizan y transmiten los datos capturados por los sensores hacia el entorno de procesamiento. Este permite evaluar la consistencia y eficiencia del sistema, considerando las variaciones en los datos recibidos y su impacto en la capacidad del robot para reaccionar en tiempo real. Los resultados obtenidos ofrecen una perspectiva detallada del rendimiento en la transmisión de datos durante las pruebas realizadas.

La gráfica de la **Fig. 4.9** las frecuencias de publicación de distintos canales de datos en un sistema de visión artificial. Muestra que uno de los canales tiene la mayor frecuencia de publicación, alcanzando aproximadamente 60 Hz, seguido por otros con frecuencias de alrededor de 30 Hz, 25 Hz y 20 Hz. Esto refleja la velocidad con la que se transmiten diferentes tipos de información, destacando las variaciones según el tipo de datos procesados.

Los valores en de frecuencia de los tópicos dependen de la configuración del sensor (frecuencia máxima, resolución), el tipo de datos transmitidos (imágenes, nubes de puntos), el procesamiento interno del dispositivo, el rendimiento del hardware del sistema host y la configuración del middleware ROS. Tópicos con datos más ligeros, como */camera_info*, tienen frecuencias más altas, mientras que los que manejan datos más pesados, como nubes de puntos, tienen frecuencias más bajas.

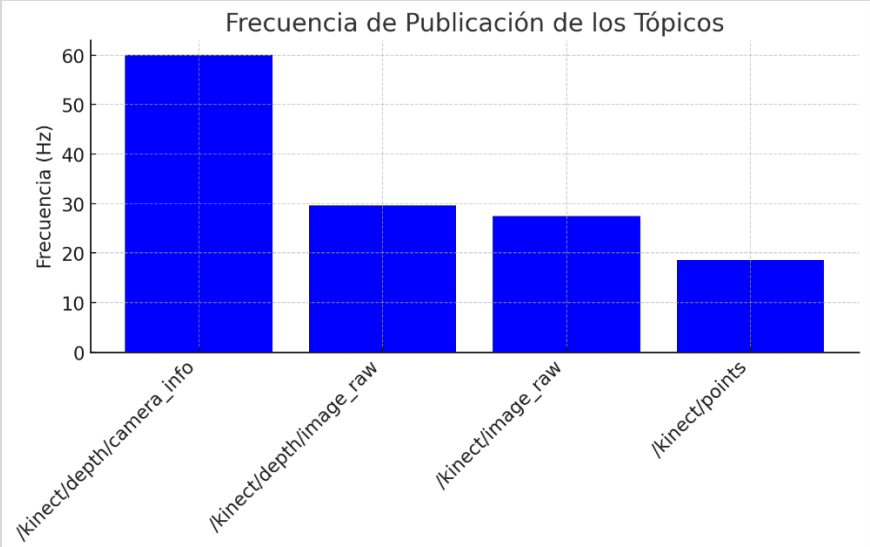


Fig. 4.9 Gráfica de tópico vs. frecuencia de publicación.

4.3.2.2. Tiempos de respuesta

La **Fig. 4.10** muestra el retardo promedio de los tópicos generados por una cámara Kinect en un sistema de comunicación robótica. El tópico */kinect/depth/camera_info* presenta el mayor retardo, alcanzando aproximadamente 0.8 segundos, lo que indica un tiempo significativo en la transmisión de información relacionada con los parámetros de la cámara.

Por otro lado, el tópico */kinect/depth/image_raw*, encargado de transmitir imágenes en crudo, tiene un retardo prácticamente despreciable. Finalmente, el tópico */kinect/points*, que transmite las nubes de puntos generadas por la cámara, muestra un retardo moderado, aunque significativamente menor al del primer tópico. Estos datos reflejan la variabilidad en el tiempo de respuesta dependiendo del tipo de información procesada y transmitida por el sistema.

El retardo promedio en los tópicos refleja las demandas de procesamiento y la naturaleza de los datos transmitidos. Los tópicos con información más frecuente o que requieren sincronización tienden a presentar mayores retardos, mientras que aquellos con datos más simples o mejor optimizados tienen retardos menores. Esto resalta la importancia de ajustar la configuración del sistema y optimizar los flujos de datos para garantizar un desempeño eficiente en aplicaciones donde la latencia puede ser crítica.

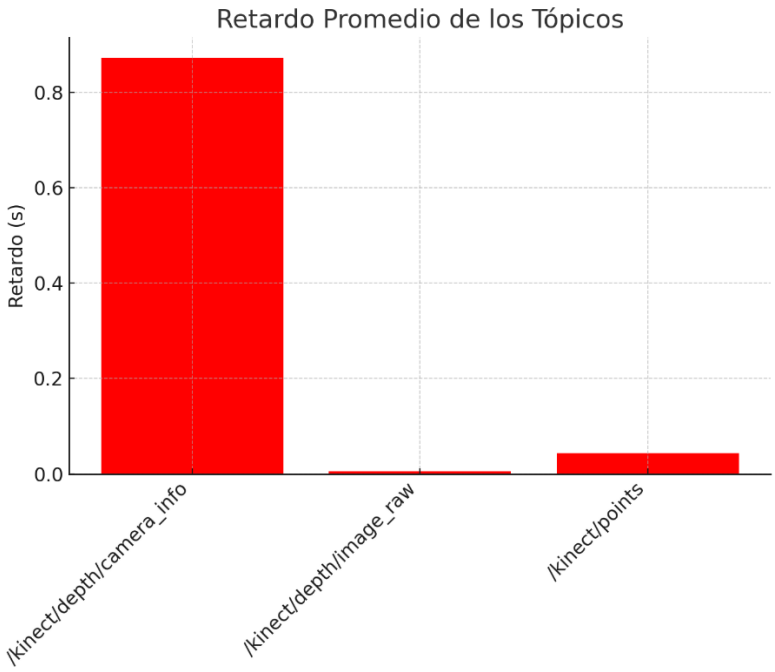


Fig. 4.10 Gráfica de tópico vs. retardo en la publicación.

Para la obtención de los datos de retraso y rendimiento véase el *Anexo C* Obtención de métricas de rendimiento.

4.4. Análisis de los resultados

El desarrollo del sistema de visión artificial para el robot móvil de alto torque permitió evaluar su desempeño en la adquisición, procesamiento y visualización de datos en un entorno tridimensional. A partir de los resultados obtenidos, se identificó que la integración entre el sensor Kinect y la Jetson Nano logró una captura de imágenes RGB y mapas de profundidad en tiempo real, lo que facilitó la interpretación del entorno del robot.

La evaluación de desempeño determinó que la frecuencia de actualización de los datos varía dependiendo del tipo de información procesada. Los tópicos de imágenes en crudo alcanzaron tasas de refresco de aproximadamente 30 Hz, mientras que los datos de nubes de puntos oscilaron entre los 18 y 27 Hz. Estas diferencias se atribuyen a la cantidad de información procesada y a la capacidad computacional del sistema embebido. Asimismo, los tiempos de respuesta mostraron que la transmisión de datos de profundidad presenta un retardo moderado, pero aceptable dentro del margen esperado para aplicaciones en visión artificial.

El análisis del sistema de comunicación evidenció una estructuración de los mensajes en ROS 2, asegurando la interoperabilidad entre nodos y facilitando el flujo de datos. Los mensajes utilizados, tales como *sensor_msgs/Image* y *sensor_msgs/PointCloud2*, permitieron gestionar eficientemente la información capturada por el sensor, optimizando su procesamiento para futuras aplicaciones de navegación y toma de decisiones del robot.

La validación del sistema mediante pruebas de captura y visualización de datos en RViz demuestra que el sistema desarrollado es capaz de representar entornos tridimensionales con un nivel adecuado de precisión. La implementación de *slam_toolbox* permite generar mapas del entorno a partir de las nubes de puntos obtenidas, lo que habilita la posibilidad de extender el sistema hacia aplicaciones de localización y mapeo simultáneo (SLAM).

Capítulo V

Conclusiones y recomendaciones

El desarrollo del sistema de visión artificial para el robot móvil de alto torque tuvo como objetivo principal dotarlo de las capacidades para la percepción del entorno, permitiéndole interpretar y procesar información visual en tiempo real. Este sistema, basado en la integración de un sensor Kinect y una Jetson Nano, complementa la estructura mecánica y electrónica del robot, cuyos detalles de diseño se encuentran especificados en los documentos técnicos correspondientes bajo el mismo nombre. A través de la implementación de técnicas de procesamiento de imágenes y análisis de datos, se logró una representación tridimensional del entorno, sentando las bases para futuras aplicaciones en navegación autónoma y toma de decisiones.

5.1. Conclusiones

- La identificación y elección de los componentes, garantizó la viabilidad del sistema de visión artificial al priorizar hardware capaz de capturar y procesar datos de imágenes con información de profundidad.
- El diseño del algoritmo de visión artificial permitió que el robot interprete con precisión los datos visuales proporcionados por el sensor, permitiendo trabajar con los datos y usarlos para construir una representación en 3D del entorno.
- La ejecución de simulaciones del algoritmo previo a su implementación ha permitido validar la fiabilidad del sistema, identificando posibles áreas de mejora.

5.2. Recomendaciones

- Se sugiere explorar técnicas de optimización en la transmisión y procesamiento de datos, como la reducción de la resolución en tiempo real o el uso de algoritmos de compresión, para mejorar la eficiencia del sistema.
- Es recomendable realizar pruebas adicionales de calibración del sensor Kinect para minimizar errores en la captura de profundidad y mejorar la precisión de los datos procesados.
- Para futuras iteraciones, se sugiere implementar algoritmos de navegación y planificación de trayectorias que permitan al robot móvil utilizar los datos de visión artificial para tomar decisiones autónomas en entornos dinámicos.

5.3. Trabajo a futuro

Una de las principales mejoras a considerar en el futuro es la optimización de la comunicación entre la Jetson Nano y el cerebro del robot, una Raspberry Pi 5, cuya integración electrónica se detalla en la tesis correspondiente al diseño electrónico del robot. Actualmente, si bien la conexión entre ambos dispositivos ha sido establecida, la comunicación se ve afectada por problemas de deserialización de datos, lo que impide el intercambio eficiente de información procesada. Resolver este inconveniente permitirá que algoritmos más complejos, como SLAM, Odometría Visual y otros métodos de percepción avanzada, puedan ejecutarse directamente en la Raspberry Pi, utilizando los datos preprocesados por la Jetson Nano. De esta manera, el robot estará habilitado para una navegación autónoma más precisa y eficiente.

Bibliografía

[1] A. M. Porcelli y A. M. Porcelli, “Inteligencia Artificial y la Robótica: sus dilemas sociales, éticos y jurídicos”, *Derecho Global. Estudios sobre Derecho y Justicia*, vol. 6, núm. 16, pp. 49–105, oct. 2020, doi: 10.32870/dgedj.v6i16.286.

[2] W. Sanz-Fernández, “Enseñanza y Aprendizaje de Robótica Industrial desde la Virtualidad”, *Revista Tecnológica-Educativa Docentes 2.0*, vol. 11, núm. 2, pp. 19–27, sep. 2021, doi: 10.37843/rted.v11i2.245.

[3] V. M. García Macías y E. Intriago, “La robótica en el ámbito educativo de Ecuador”, *Serie Científica de la Universidad de las Ciencias Informáticas, ISSN-e 2306-2495, Vol. 15, Nº. 8, 2022 (Ejemplar dedicado a: Agosto), págs. 84-93*, vol. 15, núm. 8, pp. 84–93, 2022, Consultado: el 7 de noviembre de 2023. [En línea]. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=8955512&info=resumen&idioma=ENG

[4] “Revolución Robótica en Ecuador”. Consultado: el 29 de octubre de 2023. [En línea]. Disponible en: https://www.ibecmagazine.com/TECNOLOG%C3%8DA/TabId/459/ArtMID/1165/ArticleID/1067/Revoluci243n-Rob243tica-Ecuador.aspx

[5] MSc. Juan Carlos Segovia Sánchez, “ESTRATEGIA METODOLÓGICA PARA LA ENSEÑANZA DE LA ROBÓTICA A ESTUDIANTES DE 3RO DE BACHILLERATO EN LA UNIDAD EDUCATIVA ‘SAN FRANCISCO’”, Universidad Técnica del Norte, Ibarra, 2023. Consultado: el 7 de noviembre de 2023. [En línea]. Disponible en: http://repositorio.utn.edu.ec/jspui/bitstream/123456789/14977/2/PG%201629%20TRABAJO%20GRADO.pdf

[6] “UNIVERSIDAD POLITÉCNICA SALESIANA SEDE QUITO CARRERA INGENIERÍA ELECTRÓNICA Trabajo de titulación previo a la obtención del título

de: INGENIEROS ELECTRÓNICOS TEMA DESARROLLO DE UN PROTOTIPO DE ROBOT MÓVIL DE COMPETENCIA MULTI-CATEGORÍA AUTORES LUIS ANÍBAL ALDAZ ANDRADE RUBÉN DARÍO ORELLANA TORRES TUTOR ANÍBAL ROBERTO PÉREZ CHECA Quito, agosto del 2017".

[7] B. M. Albuja Sánchez y J. L. Guadalupe Almeida, "Áreas de estudio y aplicación de inteligencia artificial en las universidades mejor puntuadas del Ecuador", *Revista Científica y Tecnológica UPSE*, vol. 9, núm. 2, pp. 58–74, dic. 2022, doi: 10.26423/RCTU.V9I2.705.

[8] R. Gutiérrez Lourdes, T. Martínez Gisela, y G. Céspedes Damarys, "Desafíos de la Educación Superior. Consideraciones sobre el Ecuador", *INNOVA Research Journal*, vol. 3, núm. 2, pp. 8–16, feb. 2018, doi: 10.33890/innova.v3.n2.2018.617.

[9] M. Á. Delgado Hernández, "Visión artificial aplicada a la robótica", Universidad de La Laguna, San Cristóbal de La Laguna, 2016. Consultado: el 30 de enero de 2025. [En línea]. Disponible en: https://riull.ull.es/xmlui/bitstream/handle/915/2914/Vision%20artificial%20aplicada%20a%20la%20robotica.pdf?sequence=1

[10] S. Kanagasingham, · Mongkol Ekpanyapong, y R. Chaihan, "Integrating machine vision-based row guidance with GPS and compass-based routing to achieve autonomous navigation for a rice field weeding robot", 123d. C., doi: 10.1007/s11119-019-09697-z.

[11] R. Shah, S. Ozcelik, y R. Challoo, "Design of a Highly Maneuverable Mobile Robot", *Procedia Comput Sci*, vol. 12, pp. 170–175, 2012, doi: 10.1016/j.procs.2012.09.049.

[12] N. Winters, J. Gaspar, G. Lacey, y J. Santos-Victor, "Omni-directional Vision for Robot Navigation".

[13] R. K. Agudo Ube y F. P. Gómez Zambrano, "Diseño e implementación de robot móvil multitarea con visión artificial y programación en lenguaje python para la Universidad Politécnica Salesiana", 2022, Consultado: el 7 de noviembre de 2023. [En línea]. Disponible en: http://dspace.ups.edu.ec/handle/123456789/23566

[14] W. G. Ali, “A semi-autonomous mobile robot for education and research”, *Journal of King Saud University - Engineering Sciences*, vol. 23, núm. 2, pp. 131–138, jun. 2011, doi: 10.1016/j.jksues.2011.03.007.

[15] O. P. Geraldo Sánchez, “Sistema de Navegación Basada en Visión Artificial para un Robot Móvil”, Instituto Tecnológico de la Paz, México, 2022. Consultado: el 7 de noviembre de 2023. [En línea]. Disponible en: https://posgrado.lapaz.tecnm.mx/uploads/archivos/GeraldoS.pdf

[16] A. Carlos *et al.*, “Diseño de robot móvil para la detección del uso de mascarillas y censado de temperatura corporal utilizando visión artificial en lenguaje Python y Raspberry Pi.”, *Universidad Ricardo Palma*, 2022, Consultado: el 7 de noviembre de 2023. [En línea]. Disponible en: https://repositorio.urp.edu.pe/handle/20.500.14138/5922

[17] J. González-Jiménez y A. Ollero, “Estimación de la Posición de un Robot Móvil”, Consultado: el 9 de abril de 2024. [En línea]. Disponible en: https://www.researchgate.net/publication/267222718

[18] L. E. Solaque, M. A. Molina, E. L. Rodríguez, L. Enrique Solaque Guzmán, M. Alejandro Molina Villa, y E. Leonardo Rodríguez Vásquez, “Seguimiento de trayectorias con un robot móvil de configuración diferencial”, *Ingenierías USBMed*, vol. 5, núm. 1, pp. 26–34, jun. 2014, doi: 10.21500/20275846.298.

[19] A. Araújo, D. Portugal, M. S. Couceiro, J. Sales, y R. P. Rocha, “Desarrollo de un robot móvil compacto integrado en el middleware ROS”, *Revista Iberoamericana de Automática e Informática industrial*, vol. 11, núm. 3, pp. 315–326, jul. 2014, doi: 10.1016/J.RIAI.2014.02.009.

[20] J. J. Sanabria S y J. F. Archila D, “Detección y análisis de movimiento usando visión artificial”, *Scientia et Technica Año XVI*, vol. 49, 2011.

[21] X. Hong y Y. Wang, “Edge Computing Technology: Development and Countermeasures”, *Chinese Journal of Engineering Science*, vol. 20, núm. 2, p. 20, 2018, doi: 10.15302/J-SSCAE-2018.02.004.

[22] GREGORY. J. DUDEK, “COMPUTATIONAL PRINCIPLES OF MOBILE ROBOTICS.”, 2024, Consultado: el 18 de enero de 2024. [En línea]. Disponible en: https://books.google.com/books/about/Computational_Principles_of_Mobile_Robot.html?hl=es&id=LoftEAAAQBAJ

[23] R. Suárez *et al.*, “Robot Operating System (ROS)”, Consultado: el 1 de febrero de 2025. [En línea]. Disponible en: http://wiki.ros.org/catkin

[24] J. Ruiz-Del-Solar, P. Loncomilla, y N. Soto, “A Survey on Deep Learning Methods for Robot Vision”.

[25] E. S. Galindo, F. B. López, R. Q. Ramirez, J. A. C. Francisco, y A. L. R. Santiago, “Modelado de visión por computadora y control de movimiento para la navegación de un robot móvil”, *Revista Científica de Ingenierías y Arquitectura*, vol. 3, núm. 1, pp. 38–51, jun. 2024, doi: 10.56643/RCIA.V3I1.174.

[26] M. Hansard, S. Lee, O. Choi, y R. Horaud, *Time-of-Flight Cameras*. London: Springer London, 2013. doi: 10.1007/978-1-4471-4658-2.

[27] C. J. Su, C. Y. Chiang, y J. Y. Huang, “Kinect-enabled home-based rehabilitation system using Dynamic Time Warping and fuzzy logic”, *Appl Soft Comput*, vol. 22, pp. 652–666, sep. 2014, doi: 10.1016/J.ASOC.2014.04.020.

[28] “Jetson Nano 2GB Developer Kit - Get Started | NVIDIA Developer”. Consultado: el 1 de febrero de 2025. [En línea]. Disponible en: https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-2gb-devkit

[29] “OpenKinect/libfreenect: Drivers and libraries for the Xbox Kinect device on Windows, Linux, and OS X”. Consultado: el 15 de febrero de 2025. [En línea]. Disponible en: https://github.com/OpenKinect/libfreenect

[30] “Libfreenect - Free Software Directory”. Consultado: el 15 de febrero de 2025. [En línea]. Disponible en: https://directory.fsf.org/wiki/Libfreenect

[31] “fadlio/kinect_ros2: C++ ROS2 driver for Kinect v1 (Xbox 360).” Consultado: el 15 de febrero de 2025. [En línea]. Disponible en: https://github.com/fadlio/kinect_ros2

Anexos

Códigos y scripts disponibles en: https://github.com/pericao320/ros2_ws

Anexo A

Lanzamiento de nodos

El script de lanzamiento en la **Fig. A.1** que inicia tres nodos principales: uno para procesar los datos de la Kinect (*kinect_ros2_node*), otro para visualizar la nube de puntos en RViz, y un tercero para publicar una transformación estática entre el sistema de coordenadas del robot (base_footprint) y la cámara Kinect (*kinect_depth*). El script también carga automáticamente una configuración predefinida de RViz para visualizar los datos.

Open — pointcloud.launch.py — ~/ros2_ws/src/kinect_ros2/launch — Save

```python
import os

import launch_ros
from launch_ros.actions.node import Node

from launch.actions.declare_launch_argument import DeclareLaunchArgument
from launch.launch_description import LaunchDescription
from launch.substitutions.launch_configuration import LaunchConfiguration


def generate_launch_description():
    pkg_share = launch_ros.substitutions.FindPackageShare(package="kinect_ros2").find(
        "kinect_ros2"
    )
    default_rviz_config_path = os.path.join(pkg_share, "rviz/pointcloud.rviz")

    return LaunchDescription(
        [
            DeclareLaunchArgument(
                name="rvizconfig",
                default_value=default_rviz_config_path,
                description="Absolute path to rviz config file",
            ),
            Node(
                package="kinect_ros2",
                executable="kinect_ros2_node",
                name="kinect_ros2",
                namespace="kinect",
            ),
            Node(
                package="rviz2",
                executable="rviz2",
                name="rviz2",
                output="screen",
                arguments=["-d", LaunchConfiguration("rvizconfig")],
            ),
            Node(
                package='tf2_ros',
                executable='static_transform_publisher',
                name='static_tf_pub',
                arguments=['0', '0', '0', '0', '1.57', '0', 'base_link', 'kinect_depth']
            ),
        ]
    )
```

Python — Tab Width: 8 — Ln 28, Col 36 — INS

Fig. A.1 Script de lanzamiento del sistema para la visualización en RViz.

Después de ejecutar el script anterior se ejecuta RViz con la visualización de los datos como se indica en la **Fig. A.2** y la **Fig. A.3**.

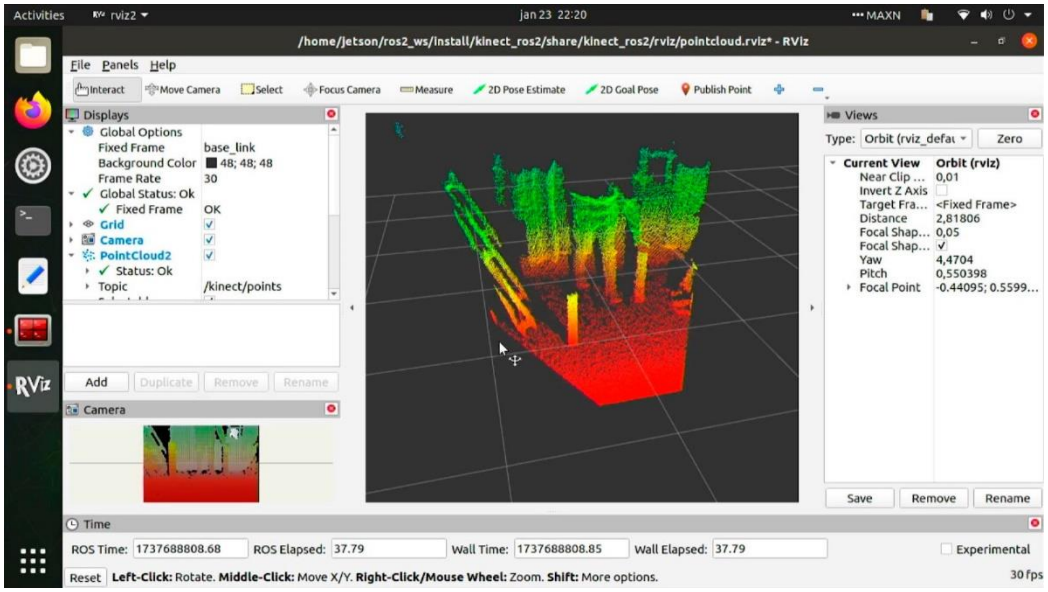


Fig. A.2 Nube puntos generada a partir del sensor Kinect.

En la **Fig. A.3** se observa la nube de con una escala de resolución más alta usando una configuración diferente en RViz.

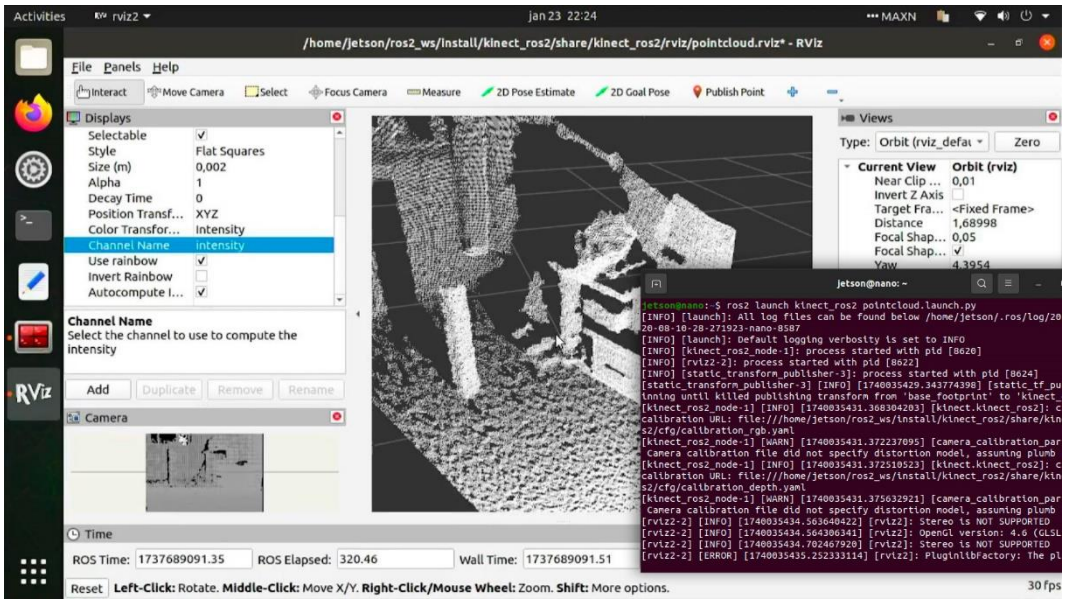


Fig. A.3 Nube puntos monocromática generada a partir del sensor Kinect.

En la **Fig. A.4** se observa el tópico que publica los puntos del sensor en la consola mientras se construye la nube de puntos en RViz.

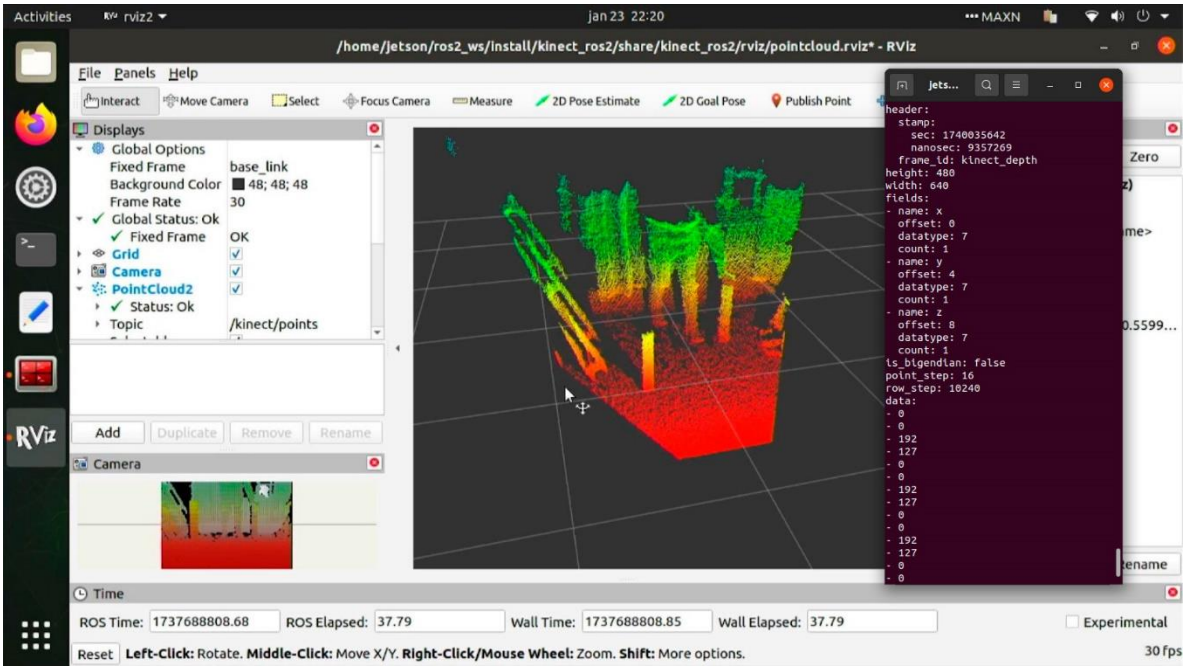


Fig. A.4 Publicación de datos a través del tópico.

Anexo B

Prueba con slam_toolbox

A partir de la información de la nube de puntos podemos hacer uso de slam_toolbox, una herramienta de ros para generar mapas como se evidencia, para esto crear dos nodos adicionales de transformación (tf2_ros) que simulen la conexión entre el robot y la cámara kinect como se observa en la **Fig. B.1**.

Open — kinect_slam.launch.py
~/ros2_ws/src/kinect_ros2/launch

```python
import os

import launch_ros
from launch_ros.actions import Node
from launch.actions import DeclareLaunchArgument, TimerAction
from launch import LaunchDescription
from launch.substitutions import LaunchConfiguration


def generate_launch_description():
    pkg_share = launch_ros.substitutions.FindPackageShare(package="kinect_ros2").find(
        "kinect_ros2"
    )
    default_rviz_config_path = os.path.join(pkg_share, "rviz/pointcloud.rviz")

    return LaunchDescription(
        [
            # Argumento para cargar configuración de RViz
            DeclareLaunchArgument(
                name="rvizconfig",
                default_value=default_rviz_config_path,
                description="Absolute path to RViz config file",
            ),

            # Publicar transformación entre Kinect y base del robot
            TimerAction(
                period=2.0,
                actions=[
                    Node(
                        package="tf2_ros",
                        executable="static_transform_publisher",
                        name="static_tf_pub_kinect",
                        arguments=["0", "0", "0", "0", "0", "-1.57", "base_footprint", "kinect_depth"],
                    ),
                ]
            ),

            # Publicar transformación entre base del robot y mapa con retraso
            TimerAction(
                period=2.5,
                actions=[
                    Node(
                        package="tf2_ros",
                        executable="static_transform_publisher",
                        name="static_tf_pub_base",
                        arguments=["0", "0", "0", "0", "0", "0", "map", "base_footprint"],
                    ),
                ]
            ),
```

Fig. B.1 Script de lanzamiento con los nodos adicionales de transformación.

Anexo C

Obtención de métricas de rendimiento

Para la obtención de datos y métricas que se observa en la sección de Evaluación de rendimiento del sistema se aplican los comandos, *ros2 topic hz <nombre_del_tópico>* para la frecuencia en Hz, *ros2 topic delay <nombre_del_tópico>* para el retardo en milisegundos, *ros2 topic bw <nombre_del_tópico>* para la cantidad de datos enviados por segundo.

```
jetson@nano:~$ ros2 topic hz /kinect/depth/camera_info
average rate: 59.716
    min: 0.001s max: 0.039s std dev: 0.01246s window: 62
average rate: 60.320
    min: 0.001s max: 0.039s std dev: 0.01372s window: 123
average rate: 59.986
    min: 0.001s max: 0.039s std dev: 0.01375s window: 184
average rate: 60.169
    min: 0.001s max: 0.039s std dev: 0.01326s window: 245
average rate: 60.135
    min: 0.001s max: 0.039s std dev: 0.01256s window: 305
average rate: 60.093
    min: 0.001s max: 0.039s std dev: 0.01182s window: 365
average rate: 60.077
    min: 0.001s max: 0.039s std dev: 0.01110s window: 425
average rate: 60.054
    min: 0.001s max: 0.039s std dev: 0.01043s window: 485

jetson@nano:~$ ros2 topic hz /kinect/depth/image_raw
average rate: 29.940
```

```
    min: 0.030s max: 0.035s std dev: 0.00131s window: 31
average rate: 29.448
    min: 0.030s max: 0.066s std dev: 0.00437s window: 60
average rate: 29.286
    min: 0.030s max: 0.066s std dev: 0.00488s window: 89
average rate: 29.234
    min: 0.030s max: 0.066s std dev: 0.00507s window: 119
average rate: 29.376
    min: 0.030s max: 0.066s std dev: 0.00459s window: 149
average rate: 29.474
    min: 0.030s max: 0.066s std dev: 0.00424s window: 179
average rate: 29.543
    min: 0.030s max: 0.066s std dev: 0.00397s window: 209
average rate: 29.599
    min: 0.030s max: 0.066s std dev: 0.00374s window: 240

jetson@nano:~$ ros2 topic hz /kinect/image_raw
average rate: 22.146
    min: 0.031s max: 0.367s std dev: 0.05900s window: 31
average rate: 23.333
    min: 0.014s max: 0.367s std dev: 0.04555s window: 56
average rate: 24.481
    min: 0.014s max: 0.367s std dev: 0.03795s window: 84
average rate: 25.496
    min: 0.014s max: 0.367s std dev: 0.03301s window: 113
average rate: 26.124
    min: 0.014s max: 0.367s std dev: 0.02965s window: 142
average rate: 26.587
    min: 0.014s max: 0.367s std dev: 0.02706s window: 172
average rate: 27.059
    min: 0.014s max: 0.367s std dev: 0.02497s window: 203
average rate: 27.444
```

```
    min: 0.014s max: 0.367s std dev: 0.02347s window: 234

jetson@nano:~$ ros2 topic hz /kinect/points
average rate: 20.475
    min: 0.028s max: 0.163s std dev: 0.03452s window: 23
average rate: 19.437
    min: 0.028s max: 0.256s std dev: 0.04411s window: 42
average rate: 18.726
    min: 0.025s max: 0.256s std dev: 0.04078s window: 60
average rate: 19.974
    min: 0.025s max: 0.256s std dev: 0.03814s window: 84
average rate: 19.461
    min: 0.023s max: 0.256s std dev: 0.03861s window: 102
average rate: 18.422
    min: 0.023s max: 0.256s std dev: 0.04237s window: 115
average rate: 18.705
    min: 0.019s max: 0.256s std dev: 0.04063s window: 136
average rate: 18.598
    min: 0.019s max: 0.256s std dev: 0.03959s window: 154

jetson@nano:~$ ros2 topic delay /kinect/depth/camera_info
average delay: 908794007.752
    min: 0.002s max: 1738562450.062s std dev: 868382742.67548s window: 44
average delay: 885998171.717
    min: 0.002s max: 1738562451.063s std dev: 869120470.34214s window:
104
average delay: 879882215.890
    min: 0.002s max: 1738562452.065s std dev: 869216582.74517s window:
164
average delay: 873179347.670
    min: 0.002s max: 1738562453.033s std dev: 869272485.35219s window:
223
average delay: 872352890.944
```

```
    min: 0.002s max: 1738562454.035s std dev: 869275798.84532s window:
283


jetson@nano:~ros2 topic delay /kinect/depth/image_raw
average delay: 0.005
    min: 0.004s max: 0.006s std dev: 0.00057s window: 29
average delay: 0.005
    min: 0.004s max: 0.007s std dev: 0.00069s window: 58
average delay: 0.005
    min: 0.004s max: 0.007s std dev: 0.00064s window: 88
average delay: 0.005
    min: 0.004s max: 0.007s std dev: 0.00071s window: 118
average delay: 0.005
    min: 0.004s max: 0.008s std dev: 0.00073s window: 147


jetson@nano:~ros2 topic delay /kinect/image_raw
average delay: 1738562545.873
    min: 1738562545.377s max: 1738562546.309s std dev: 0.26537s window:
27
average delay: 1738562546.374
    min: 1738562545.377s max: 1738562547.307s std dev: 0.55069s window:
57
average delay: 1738562546.874
    min: 1738562545.377s max: 1738562548.307s std dev: 0.83818s window:
87
average delay: 1738562547.359
    min: 1738562545.377s max: 1738562549.304s std dev: 1.11919s window:
116
average delay: 1738562547.865
    min: 1738562545.377s max: 1738562550.305s std dev: 1.41523s window:
146
average delay: 1738562548.369
    min: 1738562545.377s max: 1738562551.307s std dev: 1.70664s window:
176
```

```
jetson@nano:~$ ros2 topic delay /kinect/points
average delay: 0.044
    min: 0.039s max: 0.050s std dev: 0.00325s window: 23
average delay: 0.044
    min: 0.037s max: 0.050s std dev: 0.00338s window: 37
average delay: 0.045
    min: 0.033s max: 0.067s std dev: 0.00490s window: 58
average delay: 0.044
    min: 0.030s max: 0.067s std dev: 0.00570s window: 71
average delay: 0.045
    min: 0.030s max: 0.067s std dev: 0.00582s window: 86
average delay: 0.044
    min: 0.030s max: 0.067s std dev: 0.00585s window: 97
average delay: 0.044
    min: 0.030s max: 0.067s std dev: 0.00578s window: 109

jetson@nano:~$ ros2 topic bw /kinect/depth/camera_info
Subscribed to [/kinect/depth/camera_info]
22.09 KB/s from 100 messages
    Message size mean: 0.37 KB min: 0.37 KB max: 0.37 KB
22.12 KB/s from 100 messages
    Message size mean: 0.37 KB min: 0.37 KB max: 0.37 KB
22.12 KB/s from 100 messages
    Message size mean: 0.37 KB min: 0.37 KB max: 0.37 KB
22.15 KB/s from 100 messages
    Message size mean: 0.37 KB min: 0.37 KB max: 0.37 KB
22.15 KB/s from 100 messages
    Message size mean: 0.37 KB min: 0.37 KB max: 0.37 KB
22.18 KB/s from 100 messages
    Message size mean: 0.37 KB min: 0.37 KB max: 0.37 KB
jetson@nano:~$ ros2 topic bw /kinect/depth/image_raw
```

```
Subscribed to [/kinect/depth/image_raw]
16.77 MB/s from 100 messages
    Message size mean: 0.61 MB min: 0.61 MB max: 0.61 MB
18.46 MB/s from 100 messages
    Message size mean: 0.61 MB min: 0.61 MB max: 0.61 MB
18.46 MB/s from 100 messages
    Message size mean: 0.61 MB min: 0.61 MB max: 0.61 MB
18.47 MB/s from 100 messages
    Message size mean: 0.61 MB min: 0.61 MB max: 0.61 MB
jetson@nano:~$ ros2 topic bw /kinect/image_raw
Subscribed to [/kinect/image_raw]
27.92 MB/s from 100 messages
    Message size mean: 0.92 MB min: 0.92 MB max: 0.92 MB
27.90 MB/s from 100 messages
    Message size mean: 0.92 MB min: 0.92 MB max: 0.92 MB
27.33 MB/s from 100 messages
    Message size mean: 0.92 MB min: 0.92 MB max: 0.92 MB
27.34 MB/s from 100 messages
    Message size mean: 0.92 MB min: 0.92 MB max: 0.92 MB
27.33 MB/s from 100 messages
    Message size mean: 0.92 MB min: 0.92 MB max: 0.92 MB
jetson@nano:~$ ros2 topic bw /kinect/points
Subscribed to [/kinect/points]
79.67 MB/s from 100 messages
    Message size mean: 4.92 MB min: 4.92 MB max: 4.92 MB
88.64 MB/s from 100 messages
    Message size mean: 4.92 MB min: 4.92 MB max: 4.92 MB
92.62 MB/s from 100 messages
    Message size mean: 4.92 MB min: 4.92 MB max: 4.92 MB
87.92 MB/s from 100 messages
    Message size mean: 4.92 MB min: 4.92 MB max: 4.92 MB
```