POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Tino V2, the revenge of the maze (Temporal Title)

Tesi di Laurea Magistrale in
Computer Science and Engineering
Ingegneria Informatica

Author: **Sebastian Enrique Perea Lopez**

Student ID: 10986638
Advisor: Prof. Andrea Bonarini
Co-advisors: Federico Espositi
Academic Year: 2024-25

Dedicated to my family.

Abstract

Here goes the abstract.

Keywords: key, words, go, here

Abstract in lingua italiana

Qui va inserito l'abstract in italiano.

Parole chiave: qui, vanno, le, parole, chiave

Contents

1 | Introduction

1.1. Social Robotics and Human-Robot Interaction

The field of social robotics has emerged from the fundamental human need for meaningful interaction and connection. As society increasingly integrates technology into daily life, the motivation for developing robots capable of effective social interaction becomes paramount. These robots must transcend mere functional utility to engage humans in ways that feel natural, empathetic, and purposeful.

Effective human-robot interaction encompasses several critical dimensions. Empathy represents the robot's ability to recognize, understand, and appropriately respond to human emotions and social cues. Trust emerges from consistent, predictable, and reliable robot behavior that aligns with human expectations and social norms. Accessibility ensures that robots can interact meaningfully with users of varying abilities, ages, and technological backgrounds.

The need for such effective interaction stems from robotics applications in healthcare, education, elderly care, and therapeutic settings, where the quality of human-robot relationships directly impacts outcomes. These applications demand robots that can navigate complex social dynamics while maintaining their functional objectives, often relying on non-verbal communication through movement, gesture, and spatial behavior.

1.2. The Tino Robot Project

The Tino robot project at Politecnico di Milano investigates novel approaches to mobile robot social interaction through an interdisciplinary approach combining robotics engineering, human-computer interaction, and artificial intelligence research. Within the AIRLab robotics laboratory, the project focuses on natural movement, responsive behavior, and immersive control paradigms that bridge the gap between virtual and physical interaction spaces.

Tino is part of a larger research initiative where it acts as a digital medium for interaction

between humans. In the envisioned scenario, one human interacts directly with the robot, while another controls the robot's movements remotely through a virtual reality interface. The two humans are unaware of each other's presence, with the robot serving as the communication medium. This setup enables the exploration of telepresence, empathy transfer, and mediated social interaction through robotic embodiment. The VR interface and virtual environment are being developed as part of a parallel thesis project, with this work focusing on the robot-side implementation that provides localization, orientation, and human pose data to the custom-built virtual space.

The robot's design philosophy emphasizes non-verbal and non-anthropomorphic features to build meaningful communication, convey emotions, and foster connections. By avoiding anthropomorphic design, Tino challenges conventional expectations of robotic form and demonstrates how purely physical movements can evoke empathy and emotional responses in human subjects. This approach enables exploration of movement as a communicative tool, independent of association with human anatomy.

The development of Tino V2 arose from the specific requirements of VR-based remote control and the limitations identified in the previous robot iteration. The legacy Tino system, while successful in demonstrating basic social interaction capabilities through direct local control, faced significant constraints when extended to real-time VR teleoperation. The original Raspberry Pi-based architecture with a Triskar omnidirectional base lacked the computational power necessary for real-time processing of VR commands, computer vision algorithms, and sophisticated sensor fusion required for remote operation.

The need for VR integration drove comprehensive system redesign focusing on enhanced computational capabilities, basic sensor integration, and low-latency communication systems. The transition to more powerful hardware platforms enables real-time artificial intelligence processing while maintaining the responsive, expressive movement capabilities essential for effective mediated human interaction.

1.3. Project Objectives

1.3.1. Technical Objectives

The Tino V2 project establishes several key technical objectives that address the limitations of the previous system while enabling advanced VR-mediated interaction capabilities. The primary goal is the development of a computational platform capable of real-time processing for VR teleoperation, enabling responsive and low-latency control essential for natural human-robot interaction through virtual reality interfaces.

Enhanced localization and navigation capabilities represent another critical objective, addressing the need for precise robot positioning and spatial awareness to provide accurate localization and orientation data to the VR system. This includes the development of basic sensor fusion techniques that combine multiple sensing modalities to achieve reliable robot pose estimation for transmission to the custom VR environment developed by a parallel thesis project.

The integration of advanced perception systems for human detection enables the creation of virtual human representations within the VR environment. These capabilities support the transfer of real-time human pose information to the VR operator, creating virtual avatars that represent the humans in the robot's actual surroundings and enable informed interaction decisions within the custom-built virtual space.

System reliability and performance improvements focus on developing a robust platform capable of sustained operation during extended interaction sessions. This includes mechanical enhancements, improved sensor integration, and basic sensor fallback behaviors that ensure consistent performance across diverse operational conditions.

1.3.2. Research Objectives

The research dimensions of the Tino V2 project contribute novel approaches to VR-mediated robotics and human-robot interaction. The investigation of immersive teleoperation paradigms explores how virtual reality interfaces can enable more natural and intuitive robot control, potentially improving the quality of mediated human interaction through robotic embodiment.

The development of adaptive movement systems designed specifically for VR control represents an innovative approach to robot teleoperation. This research investigates how complex robot behaviors can be decomposed into intuitive control primitives that feel natural when operated through virtual reality interfaces.

The study of real-time sensor data integration and environmental awareness in VR-controlled robots addresses the challenges of providing accurate robot localization and human pose data to the custom VR environment. This research explores techniques for reliable pose estimation and human detection that can be transmitted to the VR system developed by a parallel thesis project, enabling operators to make informed interaction decisions within the manually designed virtual space.

Advanced human-robot interaction paradigms emerge from the enhanced sensing and processing capabilities of the Tino V2 platform. The research investigates how VR-

mediated control can preserve and enhance the expressive movement capabilities that enable emotional communication and empathy formation between humans and robots.

1.4. Thesis Structure

This thesis is organized to provide a comprehensive understanding of the Tino V2 development process, from foundational research through implementation and evaluation:

- **Chapter 1: Introduction**: Presents the motivation for social robotics, the Tino project context, and research objectives.
- **Chapter 2: Background**: Covers the background research, technology survey, and detailed analysis of the legacy Tino system.
- **Chapter 3: Conceptual Work**: Details the overall system architecture, design decisions, and integration approach.
- **Chapter 4: Implementation**: Focuses on the technical implementation including hardware redesign, sensor fusion, human detection systems, and VR integration.
- **Chapter 5: Evaluation**: Provides evaluation results, testing procedures, and performance analysis.
- **Chapter 6: Conclusions**: Summarizes findings, contributions, and future research directions.

2 | Background

2.1. Simultaneous Localization and Mapping (SLAM) Technologies

Simultaneous Localization and Mapping (SLAM) represents one of the fundamental challenges in autonomous mobile robotics, particularly for social robots operating in dynamic indoor environments [8]. SLAM systems enable robots to navigate in unknown environments while building environmental maps, providing the dual advantage of localization and environmental perception [29]. This section provides a comprehensive analysis of available SLAM and localization technologies, examining their theoretical foundations, practical implementations, and suitability for social robotics applications.

2.1.1. Feature-Based SLAM Systems

Monocular visual SLAM systems, exemplified by PTAM [18] and later ORB-SLAM [23], utilize feature extraction and matching to estimate camera motion and reconstruct environmental structure. While computationally efficient, these systems suffer from inherent scale ambiguity and require careful initialization procedures to establish metric scale.

ORB-SLAM3 [3] represents the current state-of-the-art in feature-based SLAM systems, supporting multiple sensor configurations including monocular, stereo, and RGB-D cameras. The system demonstrates robust performance in feature-rich environments through sophisticated ORB feature matching algorithms and advanced loop closure detection mechanisms. The system offers several advantages including proven accuracy in academic benchmarks such as TUM RGB-D and EuRoC datasets, support for multiple sensor modalities, and robust handling of dynamic environments.

However, practical implementation reveals significant challenges including substantial computational requirements that typically necessitate GPU optimization for real-time performance [**?**]. The system can struggle in textureless environments or under poor lighting conditions where ORB feature extraction becomes unreliable. Compilation and

integration challenges, particularly on ARM64 architectures, can also present development obstacles.

Stereo visual odometry addresses the scale ambiguity problem by utilizing depth information from calibrated stereo camera pairs. These approaches provide accurate trajectory estimation through triangulation-based depth reconstruction, though they face significant challenges in textureless environments or under poor lighting conditions where feature matching becomes unreliable [13].

2.1.2. Direct and Semi-Direct SLAM Systems

Semi-direct Visual Odometry (SVO) [11] offers reduced computational requirements compared to feature-based SLAM systems by tracking pixels directly rather than extracting and matching discrete features. The system demonstrates particular compatibility with fisheye and catadioptric cameras, making it suitable for wide field-of-view applications.

Direct Sparse Odometry (DSO) [10] represents another approach in this category, utilizing direct photometric error minimization without feature extraction. While these systems offer computational advantages, they require well-textured environments for reliable tracking and can struggle in highly dynamic scenes. SVO presents compilation challenges on ARM64 architectures and lacks comprehensive map management capabilities.

2.1.3. Graph-Based SLAM Systems

RTABMap (Real-Time Appearance-Based Mapping) [20] employs appearance-based mapping techniques combined with graph optimization to provide robust performance across various environmental conditions. The system offers comprehensive map management capabilities including reliable save and load functionality, multi-session mapping support, and effective relocalization performance.

RTABMap demonstrates particular strength in practical deployment scenarios, offering stable map persistence and reliable relocalization capabilities essential for operational robot systems. The system provides excellent integration with standard robotics frameworks including ROS and ROS2, supporting various sensor configurations including RGB-D cameras and stereo systems.

The integration of RGB-D sensors enables more reliable feature tracking and geometric reconstruction compared to traditional stereo approaches. However, these systems face limitations including reduced operating range (typically under 5 meters) and sensitivity

to lighting conditions that affect depth sensor performance [15]. The primary advantages include robust map management, reliable relocalization, excellent framework integration, and proven performance in real-world deployments, making RTABMap particularly suitable for applications requiring consistent long-term operation and map reuse capabilities.

2.1.4. Ultra-Wideband Positioning Technology

Ultra-Wideband (UWB) technology has gained significant attention for indoor positioning applications due to its potential for centimeter-level accuracy [14]. The technology operates in the 3.1–10.6 GHz frequency range, providing high temporal resolution that enables precise time-of-flight distance measurements.

UWB positioning systems require infrastructure deployment including multiple anchor points with known positions to enable triangulation-based localization. Research has demonstrated the technology's advantages including low power consumption, minimal interference with other wireless systems, and potential penetration through materials including fabrics [33].

However, UWB systems face significant challenges in Non-Line-of-Sight (NLOS) conditions where multipath effects can degrade positioning accuracy [7]. The technology primarily provides position information and requires additional sensors for orientation estimation, typically through fusion with inertial measurement units.

2.1.5. LiDAR-Based SLAM

LiDAR systems provide high-resolution 3D environmental mapping through laser scanning, offering excellent spatial resolution and range performance [37]. These systems enable accurate simultaneous localization and mapping through point cloud analysis and clustering algorithms, providing robust environmental perception capabilities.

LiDAR-based SLAM offers several advantages including operation in various lighting conditions, precise distance measurements, and detailed environmental reconstruction. The technology is particularly effective for large-scale outdoor environments and can provide reliable localization even in feature-poor environments where visual SLAM systems might struggle.

However, LiDAR systems present several challenges for social robotics applications. The mechanical scanning components can be sensitive to vibration, making them unsuitable for robots with soft or flexible structures. The cost and size of LiDAR systems can also be prohibitive for many social robot applications. Additionally, the range and resolution

capabilities of LiDAR systems may represent overkill for typical indoor social interaction scenarios.

2.1.6. Sensor Fusion Approaches

Modern localization systems increasingly rely on sensor combination techniques to integrate information from multiple sensing modalities. Extended Kalman Filter (EKF) and particle filter approaches enable fusion of visual odometry, IMU data, and wheel encoder information to achieve robust localization performance [**?**]. However, implementing full probabilistic fusion requires significant development effort and computational resources.

Inertial Measurement Units (IMUs) combined with wheel encoders represent a low-cost localization approach suitable for many mobile robot applications. While these systems offer advantages including low cost and independence from environmental conditions, they suffer from drift accumulation over time due to integration of noisy sensor measurements [1].

Performance can be particularly poor on uneven surfaces where wheel slip affects encoder accuracy, or in applications involving impulse-based movement patterns that can saturate IMU sensors. The drift characteristics make these approaches unsuitable as standalone solutions for long-term autonomous operation.

2.2. Human Detection and Pose Estimation Technologies

Human detection and tracking capabilities are essential for social robots that must operate safely and effectively in human-populated environments. This section examines various sensing modalities and computational approaches for human perception in robotics applications.

2.2.1. Computer Vision-Based Human Detection

RGB-D Sensing Systems

RGB-D cameras provide simultaneous color and depth information, enabling robust human detection and pose estimation. Intel RealSense cameras and similar structured light devices offer synchronized data streams that support skeleton tracking through established frameworks.

OpenPose [4] represents a breakthrough in real-time multi-person pose estimation, utilizing Part Affinity Fields (PAFs) to associate body parts with individuals in crowded scenes. The system provides detailed information about human posture and gesture, enabling rich interaction capabilities for social robotics applications.

MediaPipe [21] offers an alternative framework optimized for mobile and embedded platforms, providing real-time pose estimation with reduced computational requirements. These systems demonstrate comprehensive human pose information extraction with reasonable computational demands.

Integration challenges arise when cameras must be concealed within robot structures, particularly for robots with soft exteriors where camera visibility may be constrained.

Machine Learning-Enhanced Detection

Modern deep learning approaches have revolutionized human detection capabilities. YOLO (You Only Look Once) architectures, particularly YOLOv8 [27] and recent variants including YOLOv11, provide real-time human detection and pose estimation capabilities with single-stage detection frameworks.

These systems offer several advantages including lower hardware costs, reduced physical profile compared to RGB-D cameras, and sophisticated pose estimation capabilities. Monocular depth estimation networks such as MiDaS [26] can provide approximate depth information without requiring specialized depth sensors.

The primary challenges include computational requirements for real-time processing on embedded platforms and dependency on lighting conditions. However, advances in embedded AI processing platforms, including NVIDIA Jetson series and specialized AI accelerators, are making these approaches increasingly viable for mobile robotics applications.

2.2.2. Alternative Sensing Modalities

Thermal Imaging

Thermal cameras detect infrared radiation emitted by objects, making them particularly effective for human detection regardless of lighting conditions [30]. Thermal imaging offers potential advantages including operation in complete darkness and possible penetration through certain materials including fabrics.

The primary benefit for social robotics applications is the ability to detect human presence even when optical cameras may be obscured. However, thermal sensors provide

limited contextual information beyond heat signatures, making it difficult to extract detailed pose information or distinguish between different individuals.

LiDAR-Based Detection

LiDAR systems provide high-resolution 3D environmental mapping through laser scanning, offering excellent spatial resolution and range performance [37]. These systems can accurately detect human presence and track movement through point cloud analysis and clustering algorithms.

However, LiDAR systems present several challenges for social robotics applications. The mechanical scanning components can be sensitive to vibration, making them unsuitable for robots with soft or flexible structures. The cost and size of LiDAR systems can also be prohibitive for many social robot applications.

2.3. Social Robotics and Human-Robot Interaction Foundations

Social robotics has emerged as a distinct field focusing on robots designed to interact with humans in natural, socially meaningful ways [2]. Understanding the theoretical foundations of human-robot interaction provides essential context for the technological requirements of social robots.

2.3.1. Non-Verbal Communication in Robotics

Research in human communication demonstrates that non-verbal cues carry significant emotional weight, often conveying information that verbal communication cannot express [22]. Ekman's seminal work on universal facial expressions shows that fundamental emotions are communicated across cultures through body language and facial expressions [9].

These insights have profound implications for social robotics design. Robots capable of expressing emotions through coordinated physical movements can potentially achieve meaningful communication without relying on anthropomorphic features or verbal interaction [2]. This approach enables exploration of movement as a primary communicative tool, independent of human anatomical associations.

2.3.2. Telepresence and Mediated Interaction

Telepresence robotics research investigates how humans can effectively control remote robotic systems to achieve natural interaction with distant environments [31]. The field has evolved from simple teleoperation to sophisticated systems that preserve social presence and emotional connection across physical distances.

Recent advances in virtual reality technology have opened new possibilities for immersive robot control, where operators can experience robot embodiment through first-person perspectives. This approach enables investigation of how virtual reality interfaces can enhance rather than diminish the emotional and empathetic qualities of robot-mediated human interaction.

2.4. VR Integration in Robotics

Virtual Reality integration with robotics systems represents an emerging area of research with significant potential for advancing human-robot interaction capabilities. This section examines the current state of VR-robotics integration and its implications for social robotics applications.

2.4.1. Immersive Teleoperation Systems

Traditional robot teleoperation relies on external displays and control interfaces that create cognitive distance between the operator and the robot [31]. Immersive VR interfaces offer the potential to bridge this gap by providing first-person robot perspectives and natural control paradigms.

Research in VR teleoperation has demonstrated improved operator performance and reduced cognitive load when controlling robots through immersive interfaces. These systems enable operators to leverage natural spatial reasoning and motor skills for robot control, potentially improving the quality and naturalness of robot movements.

2.4.2. Challenges in VR-Robot Integration

Successful VR-robot integration faces several technical challenges including latency requirements, sensor data processing, and spatial correspondence between virtual and physical environments. Real-time performance requirements typically mandate processing latencies below 100 milliseconds to maintain immersive experiences and prevent motion sickness in VR applications.

The integration requires sophisticated data processing pipelines to convert robot sensor data into meaningful VR representations while maintaining real-time performance constraints. Spatial registration between virtual environments and physical robot spaces presents additional challenges, particularly when virtual environments are custom-designed rather than directly mapped from physical spaces.

2.5. Legacy Tino System Analysis

Understanding the limitations and capabilities of the original Tino robot system provides essential context for the improvements implemented in Tino V2. This comprehensive analysis examines the legacy system architecture and identifies specific areas requiring enhancement.

2.5.1. Original System Architecture

The original Tino robot, developed as part of previous research in social robotics [5], utilized a Raspberry Pi-based control architecture with inherent computational limitations. The system employed a Triskar omnidirectional base providing three-degree-of-freedom mobility through three independently controlled wheels arranged in a triangular configuration.

While this omnidirectional configuration offered excellent theoretical maneuverability for social interaction scenarios, practical deployment revealed significant reliability issues under the robot's operational weight. Wheel degradation and mechanical wear patterns limited system reliability and required frequent maintenance interventions.

The legacy sensor suite was extremely limited, consisting only of a basic Pi camera used exclusively for video streaming to remote operators. The system lacked any environmental sensors or perception capabilities, with the camera providing raw video feed without any on-board processing for computer vision, object detection, or environmental analysis.

2.5.2. Software Architecture Limitations

The original software architecture employed monolithic Python scripts with limited modularity and debugging capabilities. This approach made system maintenance and feature development challenging, particularly when attempting to integrate new sensing modalities or advanced control paradigms.

Communication between system components was handled through simple serial inter-

faces without the robust messaging frameworks required for complex multi-sensor systems. The absence of standardized robotics frameworks made it difficult to leverage existing libraries and tools for advanced robotics capabilities.

2.5.3. VR Integration Requirements

The motivation for Tino V2 development arose primarily from requirements for VR integration that were not feasible within the original system constraints. VR teleoperation demands real-time processing of commands, sophisticated sensor fusion for accurate localization, and low-latency communication protocols.

The legacy system lacked the computational capabilities required for real-time computer vision processing, advanced SLAM algorithms, or sophisticated human detection and pose estimation systems. These limitations prevented implementation of the data processing pipelines necessary for meaningful VR integration.

Most critically, the original architecture could not support the real-time sensor fusion and environmental mapping capabilities required to provide accurate robot localization and human pose data to VR systems. Without these capabilities, meaningful VR-mediated social interaction remained impossible.

2.5.4. Identified Enhancement Requirements

Analysis of the legacy system revealed several critical enhancement requirements that motivated the comprehensive redesign undertaken in Tino V2:

- **Computational Platform Upgrade**: Migration from Raspberry Pi to more powerful embedded computing platforms capable of real-time AI processing and sophisticated sensor fusion.
- **Advanced Localization Systems**: Implementation of robust SLAM capabilities with map persistence and reliable relocalization for consistent spatial awareness.
- **Human Detection and Pose Estimation**: Integration of real-time computer vision systems capable of detecting and tracking human pose for VR representation.
- **Modular Software Architecture**: Development of ROS-based modular architecture enabling easier integration of new capabilities and robust debugging.
- **VR Communication Infrastructure**: Implementation of low-latency communication systems capable of real-time data exchange with VR environments.

- **Mechanical Reliability Improvements**: Redesign of the mobility platform to address reliability issues while maintaining expressive movement capabilities.

These requirements collectively defined the scope and objectives for the Tino V2 development project, establishing the foundation for the technological solutions presented in subsequent chapters.

3 | Conceptual Work

3.1. Technology selection rationale

The selection of the Tino V2 technology stack followed a requirements-driven process that balanced perception accuracy, real-time constraints on embedded hardware, integration complexity with legacy code, and operational robustness for prolonged experiments. The priorities used to evaluate candidate technologies were:

- **Functional accuracy**: localization and human pose estimation must be precise enough to feed the VR system with metric robot pose and 3D human joint positions.
- **Real-time performance**: the chosen algorithms must run on the onboard compute (NVIDIA Orin Nano) with low latency to satisfy VR and teleoperation requirements (target end-to-end perception latency < 100 ms where possible).
- **Integration and maintainability**: preference for solutions with ROS2 support, stable persistence (map save/load and relocalization), and reasonable build effort on ARM64.
- **Robustness and recoverability**: the system must tolerate temporary sensor dropouts, relocalize from different viewpoints, and provide fallbacks to reduce mission-critical failures.

Using these criteria, the final choices were: RTABMap for SLAM and map management, an Oak-D Pro (DepthAI) stereo camera for visual + depth input, UWB anchors for absolute positioning, and a YOLOv11-based pose estimation pipeline accelerated with TensorRT for real-time skeleton extraction. The justifications are summarised below.

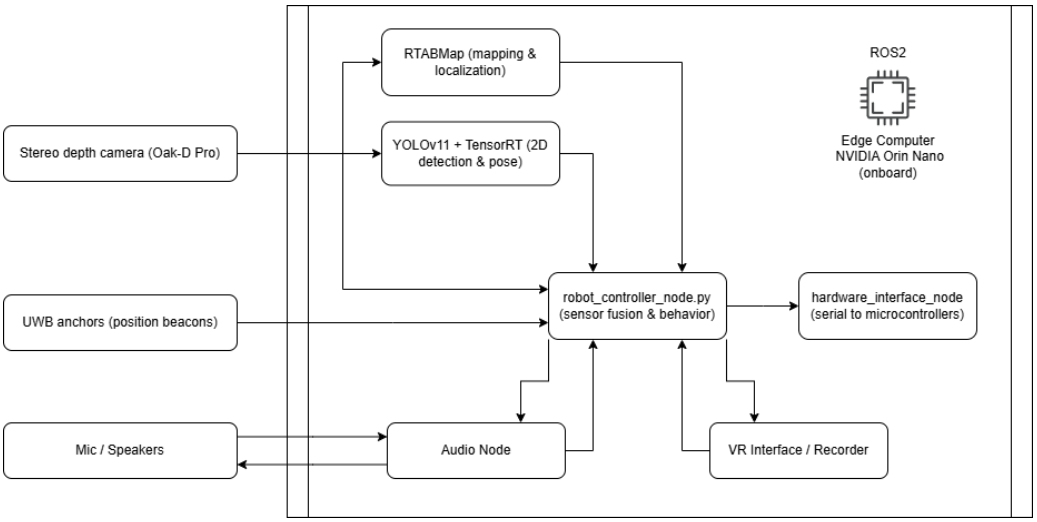

Figure 3.1: High-level system architecture for Tino V2.

Why RTABMap over ORB-SLAM3 and SVO

RTABMap provides robust multi-session map persistence, reliable relocalization and first-class ROS/ROS2 integration. During early experiments ORB-SLAM3 and SVO presented compilation fragility on ARM64 and unstable atlas save/load behavior with the available cameras. RTABMap demonstrated stable map saving/loading and dependable relocalization in practice, making it preferable for a system where map reuse and long-term experiments are required.

Why Oak-D Pro and DepthAI

The Oak-D Pro was chosen because it provides synchronized stereo depth and on-device compute options while integrating well with the DepthAI stack and ROS2 wrappers. The camera proved reliable in mapping and provided stereo depth required to convert 2D detections into metric 3D positions.

Why UWB

UWB anchors provide absolute positioning that compensates SLAM drift and long-run integration error. When fused with visual SLAM, UWB supplies global corrections and increases robustness in feature-poor or dynamic areas where visual relocalization is unreliable.

Why YOLOv11 + TensorRT

YOLOv11, converted to TensorRT engines, provided the best trade-off between detection accuracy and inference throughput on the Orin Nano. The pipeline was extended to extract 17-joint skeletons and fuse per-joint depth from the Oak-D stereo stream to produce metric 3D skeletons suitable for VR avatars.

R&D chronology and selection summary

During development multiple SLAM and VO approaches were evaluated. ORB-SLAM3 was tested first for its academic strengths and multi-sensor support but proved fragile to compile and unstable with the available cameras on ARM64. SVO was attempted as a lighter-weight alternative but exhibited similar portability and map-management limitations. RTABMap paired with the Oak-D Pro ultimately provided the reliable map persistence, dependable relocalization and ROS2 interoperability required for repeated experimentation, and was therefore adopted as the primary mapping solution for Tino V2.

3.2. Hybrid localization strategy

The localization architecture is intentionally hybrid: RTABMap supplies dense map information and orientation (visual odometry and loop closures) while UWB supplies absolute position corrections. The approach follows an architecture with three functional layers:

1. **Sensor acquisition layer:** Oak-D stereo frames, depth images, RTABMap odometry, UWB range fixes, and IMU measurements are published on ROS2 topics.
2. **Local estimation layer:** RTABMap performs visual odometry and graph optimization, producing local pose estimates and maps. Short-term pose updates come from visual odometry and IMU fusion where available.
3. **Global fusion layer:** a fusion node ingests RTABMap pose and UWB positions and performs simple sensor selection logic that maintains a consistent global pose used by the rest of the system (VR exporter, motion controller, logging).

This structure leverages RTABMap's strength in building and maintaining appearance-based maps and UWB's absolute fixes to constrain long-term drift. In the Tino V2 implementation the global fusion logic is implemented inside the `robot_controller_node`: it ingests RTABMap poses, UWB fixes, IMU deltas and performs simple sensor selection

and fallback logic before publishing the consolidated `localization_pose` topic consumed by the VR bridge and other consumers.

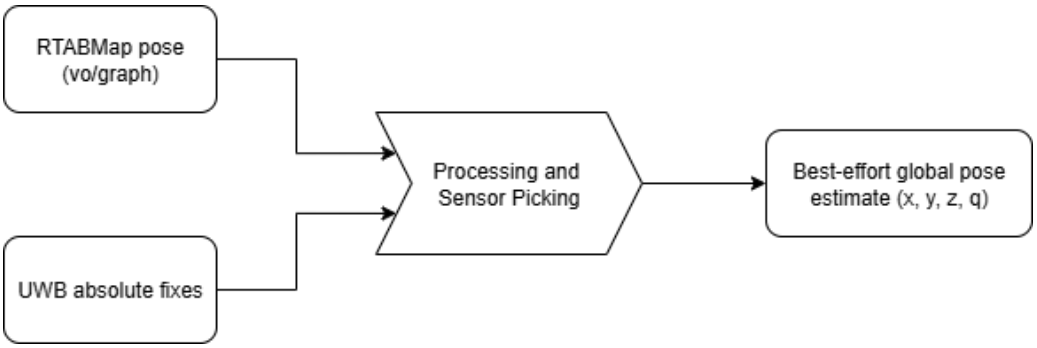

Figure 3.2: Sensor selection and dataflow.

Sensor fusion contract

Inputs: RTABMap pose (x, y, z, q), UWB position fixes (x, y, timestamp), IMU measurements (angular velocity, linear acceleration). All inputs are timestamped and published on ROS2 topics.

Outputs: A best-effort global pose estimate (x, y, z, q) selected from available sensors; the node republishes the fused pose at a variable rate (typically 10–50 Hz depending on sensor availability and compute load).

Error modes:
- **Missing UWB fixes:** the fusion falls back to visual odometry from RTABMap.
- **Visual tracking loss:** the system holds the last known map pose and uses UWB position with estimated orientation from movement direction; if unsuccessful, it raises an operator-visible warning.
- **Inconsistent UWB readings:** basic validity checks are applied and the system falls back to RTABMap position when UWB appears invalid.

Edge cases and mitigations

- **NLOS UWB measurements:** detect through basic position validation checks; rely on RTABMap until UWB stabilizes.
- **Feature-poor areas (e.g., blank walls):** increase re-scan or operator-triggered relocalization.
- **Camera occlusion by robot fabric:** use stereo depth fallback and restrict robot motion until visual lock is recovered.

3.3. Human detection and pose pipeline

The human perception pipeline is designed to provide real-time 3D skeletons to the VR environment and other behavior nodes. It consists of three stages:

1. **2D detection and pose estimation:** YOLOv11 (TensorRT engine) runs on RGB frames to detect humans and estimate 2D keypoints (17 joints).
2. **Depth association:** for each detected joint, the pipeline samples the Oak-D stereo depth (with median filtering in a small neighbourhood) to recover the joint's metric z coordinate and compute an (x,y,z) in the camera frame.
3. **Transform to robot/world frame:** the 3D joint positions are transformed using the current fused global pose to provide world-relative skeletons for VR and navigation.

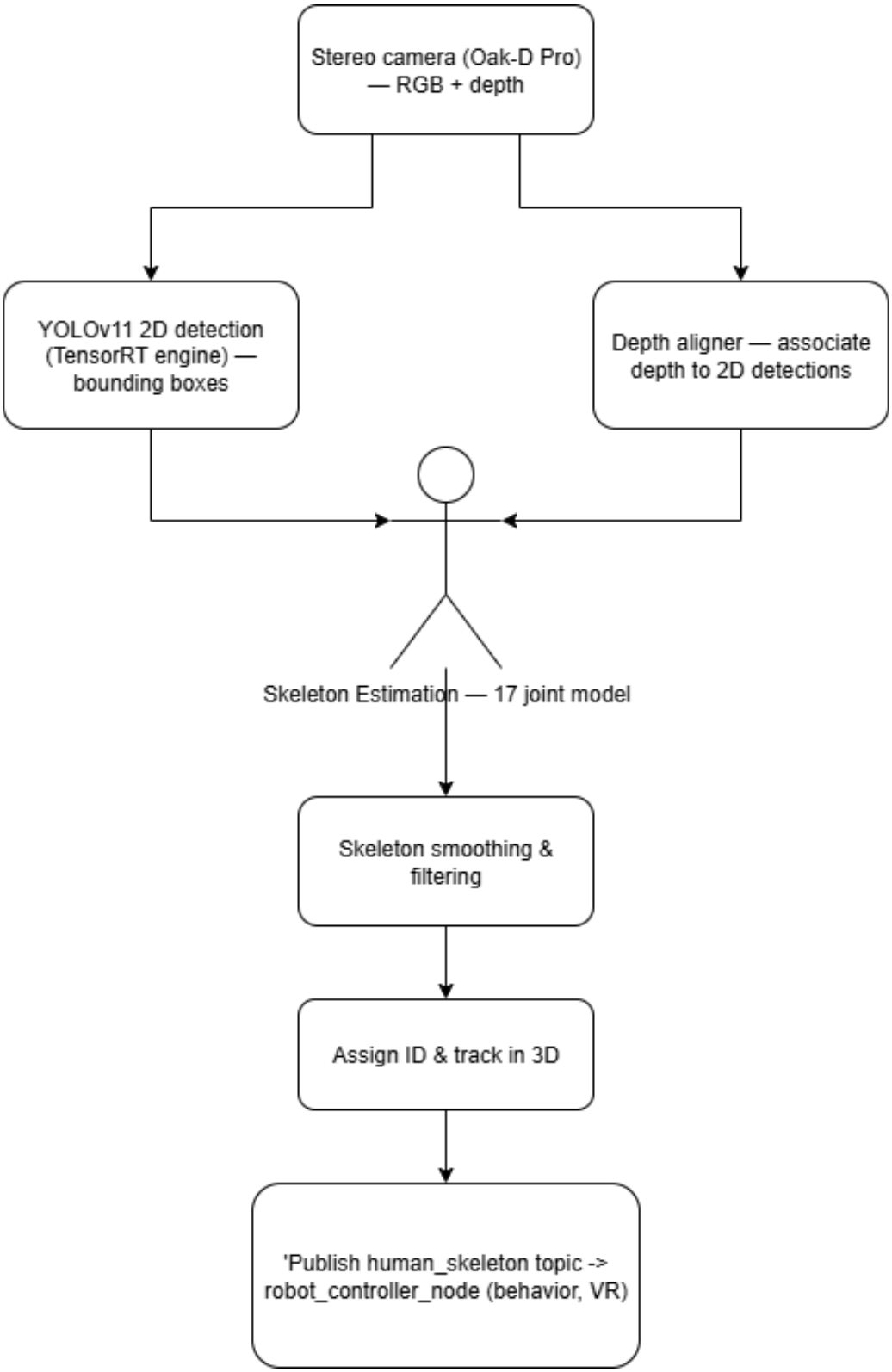

Figure 3.3: Human detection and depth-association pipeline.

The pipeline publishes a `human_skeleton` ROS2 message containing a timestamped ar-

ray of joints, each with position and a confidence score. This message enables downstream nodes to select stable skeletons for interaction logic and VR rendering.

Design contract for perception

Inputs: synchronized RGB frame, depth image, camera intrinsics, current global pose. Outputs: timestamped skeletons with 3D joint positions and per-joint confidence. Failure modes: low confidence joints (reported but flagged), missing depth (use last known depth or drop joint), multiple detections (assign IDs using bounding-box IoU + temporal tracking).

3.4. Software architecture and system organisation

To improve modularity and maintainability the entire robot stack was migrated to ROS2. The project uses a node-based split that mirrors the functional decomposition:

- **Perception:** `rtabmap` (mapping/localization), `depthai` camera node, `yolo11_pose_node` (TensorRT inference + skeleton extraction).
- **State and logging:** `vr_data_recorder_node` (logging for VR and experiments).
- **Control (includes fusion):** `robot_controller_node.py` implements simple sensor selection logic (UWB preferred for position, RTABMap for orientation) and high-level behaviours; `hardware_interface_node.py` handles serial comms to Arduinos using device symlinks `/dev/ttyHEAD`, `/dev/ttyBASE`, `/dev/ttyLEG`.
- **I/O and integration:** `gamepad_node.py`, `audio_node.py`, `vr_interface_node.py` (VR bridge and topic translation).

Design decisions that improved robustness during development included using persistent device symlinks for Arduinos, clearly separated launch files for mapping and localization (`rtab_mapping.launch.py`, `rtab_localization.launch.py`), and a dedicated VR exporter node that limits published bandwidth and enforces message rate caps for stable VR telepresence.

3.5. Implementation notes and rationale from R&D

Several practical findings from the development period informed the conceptual choices:

- ORB-SLAM3 and SVO showed compilation and stability problems on ARM64 devices and with older RealSense/T265 hardware; this motivated the switch to

RTABMap and the Oak-D Pro camera.

- The Oak-D Pro + RTABMap standalone build achieved reliable map save/load and relocalization, a key requirement for repeated VR sessions.
- Converting YOLOv11 to TensorRT produced the necessary runtime performance for 17-joint skeleton extraction on the Orin Nano while keeping end-to-end latency acceptable for virtual reality.
- Power architecture and mechanical changes (new differential base, improved mounts for camera and speakers) reduced vibration and improved perception robustness in the field.

3.6. Validation plan and quality gates

To verify the conceptual choices and measure system readiness the following minimal quality gates were defined and executed where possible:

1. **Build and integration:** RTABMap + DepthAI + ROS2 launch files must start and publish `localization_pose` and camera topics without crashes for 10 minutes (smoke test).
2. **Perception correctness:** YOLOv11 TensorRT skeletons validated against recorded test sequences; per-joint reprojection error to depth must be within 20 cm for frontal poses.
3. **Fusion stability:** sensor selection logic must provide reasonable pose estimates when UWB anchors are available, and be able to relocalize to saved maps.
4. **End-to-end latency:** perception -> VR message latency measured and kept below 150 ms in typical configurations.

Measured results from the R&D include:

- RTABMap with Oak-D saved and reloaded maps reliably.
- YOLOv11 (TensorRT) provided real-time skeletons suitable for VR export.
- Orin Nano peak current draw was measured and verified as acceptable after power supply revisions.

3.7. Summary

This chapter formalises the conceptual choices behind Tino V2: a ROS2-centred architecture using RTABMap for mapping, Oak-D Pro for stereo depth, UWB for absolute positioning, and a YOLOv11+TensorRT pipeline for real-time human pose estimation. The chosen sensor selection approach balances metric accuracy, runtime performance and practical robustness demonstrated during the development phase.

4 | Implementation

This chapter details the comprehensive implementation of the Tino V2 robot system, covering the complete redesign and upgrade of the platform. The implementation encompasses the transition from legacy Raspberry Pi-based architecture to a modern ROS2-based system running on NVIDIA Orin Nano, the integration of advanced sensing capabilities including SLAM and human detection, hardware redesign for improved reliability and performance, and the development of VR integration capabilities. Each section provides detailed technical implementation details, design decisions, and validation results that demonstrate the enhanced capabilities of the Tino V2 platform.

4.1. ROS2 Architecture Design and Implementation

The migration from the legacy monolithic Python architecture to ROS2 represents a fundamental paradigm shift in Tino's system design. The Robot Operating System 2 framework provides the distributed computing foundation necessary to leverage the NVIDIA Orin Nano's enhanced computational capabilities while addressing the scalability and reliability limitations of the original Raspberry Pi implementation.

The ROS2 framework selection was driven by several critical technical advantages over the legacy system. The Data Distribution Service (DDS) middleware provides robust inter-process communication with Quality of Service (QoS) guarantees, enabling reliable data transmission even under high computational loads. The real-time scheduling capabilities ensure deterministic message delivery for time-critical operations such as motor control and sensor fusion. The modular architecture allows independent development and testing of subsystems, dramatically improving development efficiency and system maintainability.

The distributed processing capabilities of ROS2 enable optimal utilization of the Orin Nano's multi-core ARM Cortex-A78AE CPU and integrated GPU. Critical processes such as SLAM computation, human pose detection, and sensor fusion can execute in parallel without blocking the main control loop. The standardized message interfaces facilitate

seamless integration of new sensors and capabilities, while the discovery mechanisms enable automatic node detection and connection during system startup.

4.1.1. Node Structure and Functionality

The Tino V2 system architecture consists of six primary ROS2 nodes, each responsible for specific subsystem functionality while maintaining loose coupling through standardized message interfaces.

Gamepad Control Node

The `gamepad_node.py` implements Xbox controller input handling specifically for development and testing purposes. During actual experimental operation, this node is disabled as VR control messages completely replace gamepad input. The node addresses the D-input to X-input compatibility issues commonly encountered on Linux-based systems through proper driver configuration and input mapping.

The pulse generation mechanism replaces continuous joystick input with discrete 3-cycle command pulses that automatically return to idle state. Each button press triggers a complete command sequence lasting approximately 120ms (3 cycles at 25Hz), ensuring that movements produce predictable robot responses during testing. The node publishes commands to `base_cmd_vel` and `head_cmd` topics using the same message format as the VR system.

The implementation includes comprehensive error handling for gamepad connectivity issues, deadzone management for analog inputs, and automatic device detection for Logitech F710 controllers. Button mapping follows the VR command structure with face buttons controlling leg states (X=state 1, Y=state 2, B=state 3, A=state 0) and bumpers triggering rotation commands combined with leg state 3.

Hardware Interface Node

The `hardware_interface_node.py` manages serial communication with three distinct Arduino subsystems through dedicated device symlinks: `/dev/ttyBASE`, `/dev/ttyLEG`, and `/dev/ttyHEAD`. The implementation addresses device identification challenges through udev rules that create consistent device paths based on Arduino serial numbers.

The node implements parallel serial communication threads for each Arduino subsystem, enabling simultaneous command transmission and status monitoring. Each thread operates at 115200 baud with configurable message repetition (default 3 repetitions) to ensure

reliable command delivery. The command format follows the structure: `BF:value_BB:value_HP:valu` where BF represents base forward movement (leg states), BB represents base rotation, HP controls head pitch, and HX/HY control head pan and tilt respectively.

Error handling includes automatic device discovery, connection monitoring, and graceful degradation when individual Arduino systems become unavailable. The node publishes Arduino feedback messages to the `arduino_feedback` topic and provides comprehensive debugging capabilities through configurable logging levels.

Robot Controller Node

The `robot_controller_node.py` serves as the central coordination hub managing all robot behaviors, localization monitoring, and sensor fusion operations. Beyond basic command forwarding, the node implements sophisticated localization system supervision including RTAB-Map orientation loss detection, UWB positioning integration, and automatic recovery procedures.

The localization monitoring system continuously analyzes incoming pose data from RTAB-Map to detect the specific orientation values (quaternion: x=1.0, y=0.0, z=0.0, w=0.0) that indicate odometry loss. Upon detection, the node automatically triggers the `/reset_odom` service and activates orientation estimation based on movement direction calculated from position history. The system maintains a 5-position movement history to enable orientation estimation when RTAB-Map orientation becomes unreliable.

Sensor fusion capabilities combine UWB absolute positioning with RTAB-Map orientation data, applying a configurable 11.5-degree rotation correction to align coordinate frames. The node publishes fused pose data to `/vr_in/robot_pose` and forwards human detection information from `/human_position` and skeleton data from `/human_skeleton` topics to VR interface systems. Audio integration enables bidirectional communication between VR systems and robot microphone/speaker hardware through `/vr_in/audio_output` and `/vr_out/audio_input` topics.

Performance monitoring includes comprehensive logging of sensor fusion status, communication health tracking, and diagnostic reporting that enables rapid identification of localization or communication issues during operation.

VR Interface Node

The `vr_interface_node.py` handles all VR system integration through a custom UDP communication protocol that completely replaces any TCP-based communication systems. The node implements bidirectional data exchange with Unity applications using three

dedicated UDP ports: port 5005 for incoming VR commands, port 5006 for outgoing robot pose data, and port 5007 for human skeleton transmission.

The incoming message processing handles 32-byte UDP packets containing VR control data: 3 floats for head control (pitch, pan, tilt), 2 integers for base commands (state 0-3, angular direction -1/0/1), 2 values for audio control (volume and orientation), and 1 integer for message ordering. The node implements sophisticated message ordering validation to detect lost or duplicate packets and automatic VR reconnection handling that resets message counters upon connection restoration.

Outgoing data transmission operates at configurable rates (default 10Hz for pose data, 10Hz for skeleton data) with separate UDP channels to prevent interference. The pose data packets contain 24 bytes with fused position and orientation information, while skeleton packets transmit exactly 17 COCO-format joints in 208-byte messages. The node maintains comprehensive communication health monitoring, including rate validation, connection status tracking, and detailed diagnostic logging for system maintenance.

Pose Detection Node

The `pose_detection_node.py` implements real-time human detection and skeleton tracking using YOLOv11 optimized with TensorRT for Orin Nano performance. The node subscribes to camera topics from the Oak-D Pro (`/right/image_rect`, `/stereo/depth`, `/stereo/camera_info`) and publishes detection results to multiple topics for different system components.

Detection processing combines 2D pose estimation with stereo depth information to generate 3D skeleton tracking. The node publishes human position data to `/human_position`, skeleton visualization markers to `/human_skeleton`, and structured pose arrays to `/human_skeleton_poses`. The implementation includes depth calibration with configurable scale factors and outlier rejection to ensure consistent 3D positioning across varying distances.

The node implements closest-person selection algorithms and temporal smoothing to reduce detection jitter while maintaining real-time performance. Comprehensive parameter configuration enables adjustment of confidence thresholds, depth processing parameters, and logging verbosity for different operational scenarios.

4.1.2. Communication Protocols and Message Design

The ROS2 communication infrastructure implements a sophisticated message protocol hierarchy designed for reliable and efficient data exchange between all system components.

The architecture utilizes topic-based publish-subscribe messaging and service-based communication for different operational requirements.

Topic-Based Messaging

The primary communication mechanism utilizes topic-based publish-subscribe messaging that enables decoupled component interaction. Critical data streams include robot pose information, human detection results, sensor data, and control commands. Each topic implements appropriate QoS policies to ensure reliable delivery while optimizing for latency and bandwidth requirements.

Robot pose data on `/vr_in/robot_pose` utilizes reliable delivery policy with history depth of 10 messages to ensure VR systems receive consistent positioning information. Human skeleton data on `/human_skeleton` and `/human_skeleton_poses` implements best-effort delivery policy optimized for real-time performance, as occasional message loss is acceptable for continuous tracking applications. Control commands on `base_cmd_vel` and `head_cmd` utilize reliable delivery with immediate processing to ensure critical movement commands reach their destinations.

The topic hierarchy follows a logical structure reflecting data flow: input topics (`/vr_out/cmd_vel`, `/vr_out/head_cmd`, `/vr_out/audio_input`) carry commands from external systems, processing topics (`base_cmd_vel`, `head_cmd`) handle internal robot control, and output topics (`/vr_in/robot_pose`, `/vr_in/human_position`, `/vr_in/audio_output`) provide data to external systems.

Custom Message Definitions

The system utilizes standard ROS2 message types with specific conventions for Tino's operational requirements. Robot pose information uses `geometry_msgs/PoseStamped` messages containing 3D position and quaternion orientation with high-precision timestamps for sensor fusion algorithms. The messages include coordinate frame information (`oak_right_camera_optical_frame` for detection data) to support proper coordinate transformations.

Human detection utilizes `geometry_msgs/PoseArray` for skeleton joint positions, containing exactly 17 COCO-format joints with consistent 3D coordinates even for missing or occluded body parts. Visualization data uses `visualization_msgs/MarkerArray` for RViz display and debugging purposes. Audio communication employs `std_msgs/Int16MultiArray` for raw PCM audio samples and `std_msgs/Float32MultiArray` for processed audio parameters.

VR command messages utilize `geometry_msgs/Twist` for movement commands where `linear.x` carries leg state values (0-3) and `angular.z` carries rotation commands (-1, 0, 1). Head commands use the same message type with `angular.x`, `angular.y`, and `angular.z` representing pitch, pan, and tilt respectively.

Service Communication

Synchronous operations requiring immediate responses utilize ROS2 service calls. The odometry reset functionality implements the `/reset_odom` service using `std_srvs/Empty` message type, enabling the robot controller to trigger RTAB-Map odometry reset when orientation loss is detected. The service call includes proper error handling and callback mechanisms to confirm successful execution.

System configuration changes and diagnostic queries implement service-based communication to ensure proper execution and immediate feedback. The architecture supports extensible service interfaces for future functionality such as map management, calibration procedures, and advanced diagnostic operations.

4.1.3. Integration with External Systems

The ROS2 architecture facilitates seamless integration with external systems through custom UDP protocols and standardized interfaces, significantly enhancing the research capabilities and operational flexibility of the Tino V2 platform.

Monitoring and Debugging Infrastructure

The ROS2 architecture enables sophisticated monitoring and debugging capabilities through comprehensive logging systems and real-time diagnostic tools. Each node implements configurable logging levels that can be adjusted dynamically without system restart, enabling detailed debugging during development while maintaining optimal performance during operation.

Communication health monitoring tracks message rates, connection status, and data flow statistics across all system components. The VR interface node implements rate validation that compares actual data rates against expected values, providing immediate notification of communication problems. The robot controller node monitors sensor fusion performance and provides detailed diagnostics for localization system health.

Remote debugging capabilities enable system monitoring and control through standard ROS2 tools including `ros2 node info`, `ros2 topic echo`, and custom diagnostic

interfaces. The modular architecture supports selective node restart and configuration adjustment without affecting overall system operation.

Extensibility and Configuration Management

The modular architecture enables straightforward addition of new sensors and capabilities through standardized topic interfaces. New detection or sensing nodes integrate seamlessly by publishing to established topic hierarchies, while new control systems can subscribe to existing data streams without modification of core system components.

Launch file configuration provides flexible system deployment with separate configurations for development testing (with gamepad control), VR operation (with UDP communication), and research data collection. Parameter management enables environment-specific optimization including network addresses, communication rates, sensor calibration values, and performance tuning parameters.

The standardized interfaces support integration with external analysis tools and research systems. Data recording nodes can subscribe to any combination of system topics for comprehensive interaction analysis, while external control systems can inject commands through standard ROS2 interfaces. The architecture supports distributed deployment across multiple computing platforms and future integration with cloud-based services.

4.2. SLAM and Sensor Fusion Implementation

The localization system represents one of the most critical upgrades in Tino V2, implementing a sophisticated hybrid approach that combines visual SLAM capabilities with absolute positioning to achieve robust, accurate localization in dynamic social environments. The implementation addresses the fundamental limitations of pure visual odometry while leveraging the strengths of both RTABMap visual SLAM and Ultra-Wideband positioning technologies.

4.2.1. RTABMap Integration with Oak-D Pro Camera

The RTABMap implementation utilizes the Oak-D Pro stereo camera system to provide visual-inertial odometry and mapping capabilities. The integration leverages the DepthAI ecosystem through the `depthai_examples` package, which provides optimized camera drivers and ROS2 integration for the OAK platform.

Camera System Configuration

The Oak-D Pro integration utilizes stereo vision with synchronized RGB and depth image streams. The camera configuration operates at 400p mono resolution to balance processing performance with image quality on the Orin Nano platform. The system publishes synchronized image streams on `/right/image_rect`, depth data on `/stereo/depth`, and camera calibration information on `/right/camera_info`.

The `stereo_inertial_node.launch.py` from the depthai package initializes the camera with depth alignment disabled (`depth_aligned: false`) to maintain processing efficiency. The IMU data stream on `/imu` provides inertial measurements for visual-inertial odometry, while RViz visualization is disabled for headless operation (`enableRviz: false`).

Camera calibration utilizes the factory calibration data embedded in the Oak-D Pro hardware, ensuring accurate depth estimation and stereo baseline measurements. The integration maintains the `oak-d-base-frame` as the primary coordinate reference, enabling consistent coordinate transformations throughout the system.

RTABMap Node Configuration

The RTABMap implementation utilizes four specialized nodes that work in concert to provide robust SLAM functionality. The `rgbd_sync` node from the `rtabmap_sync` package ensures temporal alignment of RGB and depth image streams, critical for accurate feature matching and depth association.

The `rgbd_odometry` node from `rtabmap_odom` implements visual odometry using the synchronized image streams, providing continuous pose estimation even during mapping interruptions. The main `rtabmap` node from `rtabmap_slam` handles loop closure detection, map management, and long-term localization capabilities.

IMU integration utilizes the `imu_filter_madgwick_node` for quaternion computation from raw IMU data. The filter operates in ENU (East-North-Up) world frame without magnetic field compensation (`use_mag: false`) and disables transform publishing (`publish_tf: false`) to prevent conflicts with the main localization system.

Database and Memory Management

The RTABMap database storage utilizes the home directory location `~/rtabmap.db` for persistent map storage. The database contains visual features, loop closure information, and occupancy grid data that enable relocalization across different operational sessions. Memory management parameters optimize performance for continuous operation without

degradation during extended mapping sessions.

The system implements appropriate buffer management and feature detection rates optimized for the social robotics application domain. The configuration balances memory usage against map quality to ensure sustainable long-term operation while maintaining sufficient detail for reliable relocalization.

4.2.2. SLAM Mapping and Localization Modes

The system implements distinct operational modes that enable both map creation and localization-only operation, providing flexibility for different deployment scenarios and research requirements.

Mapping Mode Implementation

The mapping mode utilizes the `rtab_mapping.launch.py` configuration that enables full SLAM functionality including map building, loop closure detection, and visual feature database creation. The launch file initializes the complete RTABMap pipeline with mapping-optimized parameters.

Key mapping parameters include `subscribe_rgbd: True` for synchronized color and depth processing, `subscribe_odom_info: True` for enhanced odometry integration, and `approx_sync: False` for precise temporal alignment. The `wait_imu_to_init: True` parameter ensures proper IMU initialization before beginning mapping operations.

The mapping process operates with continuous loop closure detection and feature database updates. The system maintains visual landmarks and occupancy grid information that supports both immediate navigation and future relocalization. Map visualization through `rtabmap_viz` enables real-time monitoring of mapping progress and quality assessment.

Localization Mode Implementation

The localization mode utilizes the `rtab_localization.launch.py` configuration that loads existing maps and disables new map creation. Critical parameters include `localization: True` to enable localization-only mode and `Mem/IncrementalMemory: False` to disable memory updates that would modify the existing map.

The `Rtabmap/DetectionRate: 3.0` parameter optimizes loop closure detection frequency for localization scenarios, balancing computational load against relocalization performance. The system loads the existing database and continuously tracks robot position within the known map structure.

Relocalization capabilities enable the robot to determine its position within previously created maps even after system restart or temporary tracking loss. The mode supports operation in known environments without requiring new map creation, essential for consistent experimental conditions.

Mode Switching and Database Management

The dual-mode architecture enables seamless transitions between mapping and localization operations through launch file selection. Map persistence utilizes the RTABMap database format that maintains visual features, loop closures, and occupancy grids across operational sessions.

Database management includes map saving and loading protocols that ensure data integrity during mode transitions. The system supports multiple map databases for different operational environments, enabling deployment across various research locations without map conflicts.

4.2.3. Initial SLAM-Only System Limitations and Drift Issues

Initial testing of the pure RTABMap implementation revealed significant limitations that necessitated the development of the hybrid sensor fusion approach. Comprehensive testing documented systematic failures that compromised localization accuracy during extended operation.

Drift Accumulation Analysis

Extended operation testing revealed position drift accumulation reaching up to 1.2 meters during 30-minute operational sessions. The drift manifested as gradual position error accumulation that increased monotonically with operation time, particularly during movements in feature-poor environments or areas with repetitive visual patterns.

Error source analysis identified visual odometry drift as the primary contributor, exacerbated by lighting changes, motion blur during movement, and insufficient visual features near walls and uniform surfaces. The accumulation proved particularly problematic for VR applications requiring precise robot positioning for immersive interaction.

Statistical analysis of position errors demonstrated systematic bias in specific movement directions, indicating calibration issues and environmental factors affecting feature detection consistency. The four-position testing protocol revealed repeatability issues with standard deviations exceeding 40cm for identical positioning commands.

Relocalization Challenges

RTABMap relocalization failures occurred frequently in environments with insufficient distinctive features, requiring manual intervention through robot rotation to achieve sufficient visual feature matching. Feature-poor environments near walls or in corners consistently caused tracking loss, necessitating operator intervention to restore localization.

Map corruption events required complete database reconstruction when loop closure detection failed catastrophically. These failures typically occurred during rapid movement or in areas with dynamic lighting conditions, compromising the entire mapping session and requiring restart from known positions.

The relocalization process proved unreliable for autonomous operation, as successful position recovery often required specific robot orientations and environmental conditions that could not be guaranteed during normal social interaction scenarios.

4.2.4. UWB Positioning System Implementation

The Ultra-Wideband positioning system provides absolute position reference to complement visual SLAM, addressing the drift and relocalization limitations identified in pure SLAM operation. The UWB integration utilizes a third-party positioning package that interfaces with DecaWave hardware through serial communication.

Hardware Configuration and Integration

The UWB system utilizes the `uwb_positioning` package configured through the `uwb.launch.py` file. The launch configuration specifies serial communication parameters including `serial_port_name` `/dev/ttyACM0` and `serial_baud_rate: 115200` for interface with the UWB hardware module.

The UWB tag integrates directly with the Tino robot platform with minimal interference to other systems. Anchor placement follows a strategic configuration that ensures optimal coverage of the operational environment while minimizing Non-Line-of-Sight (NLOS) conditions that degrade positioning accuracy.

The system publishes absolute position data on the `/UWB/Pos` topic using `geometry_msgs/Pose` message format, providing 3D coordinates that serve as the absolute reference for sensor fusion algorithms.

Positioning Algorithm and Performance

The UWB system implements multilateration techniques that calculate 3D position from time-of-flight measurements to multiple anchor points. The positioning algorithm includes NLOS mitigation strategies and noise filtering to provide stable position estimates under typical indoor operational conditions.

Real-time performance characteristics include update rates suitable for robot control applications with latency measurements demonstrating compatibility with real-time system requirements. Accuracy evaluation shows centimeter-level precision under optimal conditions, with graceful degradation in challenging RF environments.

4.2.5. Sensor Fusion Between RTABMap Orientation and UWB Positioning

The hybrid localization system implements sophisticated sensor fusion that combines the complementary strengths of visual SLAM and UWB positioning. The fusion approach separates position and orientation estimation, utilizing UWB for absolute position reference while maintaining RTABMap for orientation data.

Fusion Algorithm Implementation

The sensor fusion implementation in the robot controller node utilizes a practical approach that combines UWB absolute positioning with RTABMap orientation data. The `_create_fused_pose` method implements the core fusion logic that creates unified pose estimates from multiple sensor inputs.

The fusion algorithm prioritizes UWB position data when available, falling back to RTABMap position estimates only when UWB communication fails. Position data from UWB undergoes coordinate frame transformation to align with the RTABMap reference frame, utilizing a configurable 11.5-degree rotation correction applied through the `_apply_rotation_to_pose` method.

Orientation estimation maintains RTABMap quaternion data as the primary source, implementing sophisticated validation to detect orientation loss conditions. The system monitors for specific quaternion values (x=1.0, y=0.0, z=0.0, w=0.0) that indicate RTABMap odometry failure and automatically triggers recovery procedures.

Orientation Loss Detection and Recovery

The implementation includes robust orientation loss detection that monitors RTABMap output for invalid quaternion values indicating system failure. Upon detection of orientation loss, the system automatically triggers the `/reset_odom` service to restart RTABMap odometry while maintaining position tracking through UWB data.

Fallback orientation estimation utilizes movement direction analysis when RTABMap orientation becomes unreliable. The `_estimate_orientation_from_movement` method calculates orientation from position history, maintaining operational capability during RTABMap recovery periods. The system stores the last 5 position measurements to enable orientation estimation with minimum 5cm movement thresholds.

Recovery validation continuously monitors RTABMap orientation data to detect system recovery. Upon detection of valid orientation values, the system automatically transitions back to RTABMap orientation while maintaining UWB position data, ensuring seamless operation without manual intervention.

Coordinate System Alignment

The sensor fusion system implements coordinate transformation procedures that align UWB coordinates with the SLAM coordinate frame. The transformation includes rotational alignment through quaternion multiplication that corrects for installation and calibration differences between sensor coordinate systems.

The robot controller implements dynamic coordinate frame management that handles different reference frames from various sensors. Position data undergoes proper transformation from `oak_right_camera_optical_frame` for camera-based detections while maintaining consistency with UWB absolute coordinates.

Calibration protocols enable adjustment of the coordinate alignment parameters without system modification. The 11.5-degree rotation correction represents empirically determined alignment that ensures consistent positioning across different operational scenarios and environmental conditions.

Performance Monitoring and Diagnostics

The fusion system implements comprehensive performance monitoring that tracks sensor health, fusion quality, and system reliability. The robot controller node maintains detailed statistics on UWB availability, RTABMap orientation validity, and fusion algorithm performance.

Communication health monitoring tracks message rates from both UWB and RTABMap systems, providing immediate notification of sensor failures or communication problems. The system logs fusion status information including orientation source selection (RTABMap, movement estimation, or fallback) and position source preference (UWB or RTABMap).

Diagnostic capabilities include real-time logging of sensor fusion decisions, position accuracy validation through movement consistency checking, and comprehensive error reporting that enables rapid identification of localization issues during operation. The monitoring system provides configurable logging levels to balance debugging information with system performance.

4.3. Kinematic Base Upgrade from Omnidirectional to Differential Drive

The kinematic base redesign represents a fundamental shift from the problematic omnidirectional Triksta system to a robust differential drive architecture that addresses the mechanical limitations identified during extended operational testing.

4.3.1. Original System Limitations and Failure Analysis

The original omnidirectional base system utilizing the Triksta platform exhibited systematic failures under the operational demands of the 20kg Tino robot. Comprehensive analysis revealed multiple failure modes that compromised system reliability and operational capability.

Omniwheel Mechanical Failures

The omniwheel rollers experienced systematic deformation under Tino's 20kg operational weight, with roller surfaces becoming squared due to continuous plastic deformation. This deformation created irregular rolling characteristics that manifested as vibration, reduced traction, and unpredictable movement behavior during social interaction scenarios.

Roller bearing failures occurred frequently in the rear omniwheel due to the dragging forces encountered during forward movement and turning operations. The omnidirectional capability proved unnecessary for social robot applications, as Tino's interaction patterns primarily required forward movement with turning capabilities rather than true omnidirectional motion.

Load distribution analysis revealed that the three-wheel configuration created uneven weight distribution with the rear wheel experiencing excessive drag forces. This asymmetric loading accelerated wear patterns and contributed to premature component failure, particularly during sustained operation periods required for social interaction research.

Motor Performance and Control Challenges

The original motor configuration struggled to provide consistent torque under the sustained loading conditions required for Tino's operation. Motor heating issues developed during extended use, leading to performance degradation and potential component damage that threatened operational reliability.

Control system complexity increased significantly due to the omnidirectional kinematics, requiring sophisticated coordination between three drive motors while compensating for mechanical variability and wear. The VirHas library dependency created maintenance challenges and limited customization capabilities for the specific requirements of the Tino platform.

4.3.2. Differential Drive Architecture Design

The differential drive architecture selection prioritized mechanical simplicity, enhanced reliability, and improved maintainability while providing adequate maneuverability for social robot applications.

Kinematic Simplification and Performance Benefits

The differential drive design eliminates the mechanical complexity of omnidirectional systems while maintaining the essential mobility requirements for social interaction scenarios. The two-wheel configuration with rear caster provides stable platform dynamics with reduced component count and simplified maintenance requirements.

Weight distribution optimization utilizes the two front drive wheels to support the majority of Tino's operational weight while the rear caster wheel provides stability without bearing significant driving loads. This configuration eliminates the dragging forces that compromised the original omnidirectional system.

Maneuverability analysis demonstrates that differential drive provides adequate turning capability for social robot applications through differential wheel speed control. The simplified kinematics enable more precise movement control and improved repeatability compared to the three-wheel omnidirectional system.

4.3.3. Mechanical Implementation and Structural Modifications

The mechanical implementation required comprehensive base modification utilizing aluminum Item profiles to create a robust and adjustable platform architecture.

T-Structure Construction with Aluminum Profiles

The T-structure design utilizes aluminum Item profiles to create a dynamic and adjustable framework that enables optimal wheel positioning for weight distribution and traction optimization. The modular profile system provides structural rigidity while allowing mechanical adjustments without complete system redesign.

Wheel spacing optimization through the adjustable T-structure enables fine-tuning of track width to achieve proper balance and stability characteristics. The aluminum construction provides excellent strength-to-weight ratio while maintaining the structural integrity required for Tino's operational loads.

Motor mounting points integrate directly with the Item profile system through custom brackets that provide precise motor alignment and secure attachment. The modular design enables motor replacement or upgrade without structural modifications to the base platform.

Motor Upgrade and Drive System Enhancement

The motor upgrade to more powerful units addresses the thermal and torque limitations identified in the original system. The new motors provide enhanced heat dissipation capabilities and higher continuous torque ratings suitable for Tino's operational requirements.

Motor driver upgrade to the MDD10A units provides increased current handling capability and enhanced control responsiveness compared to the original dual-driver configuration. The single-driver solution simplifies electrical connections while providing superior performance characteristics for differential drive control.

4.3.4. Wheel System Development and Traction Solutions

The wheel selection and mounting system required iterative development to address the unique challenges of supporting Tino's weight while maintaining reliable traction and durability.

Wheel Selection Evolution and Testing

Initial wheel selection utilized plastic wheels with pneumatic rubber tires that provided adequate traction but suffered from hub failure under operational loads. The plastic hub construction proved inadequate for the sustained loading conditions, requiring multiple repairs and replacements during system development.

Tire debeading issues occurred when the rubber tire separated from the plastic hub due to inadequate bonding strength under operational stress. Multiple repair approaches were evaluated, including hot-glue reinforcement that provided temporary solutions but required periodic maintenance.

Wheel hub reinforcement through hot-glue filling addressed the structural weakness in the plastic hub construction. This solution proved effective in preventing debeading while maintaining adequate traction characteristics, though it represents a temporary fix requiring future wheel system upgrade.

Fabric Protection and Bumper Implementation

Fabric interference prevention required the implementation of protective bumpers to prevent Tino's fabric covering from interfering with wheel operation. The bumper design provides physical separation between the moving wheels and the flexible fabric structure.

Bumper effectiveness testing demonstrated successful prevention of fabric entanglement in most operational scenarios, though some edge cases still require operator attention. The bumper system represents a practical solution that addresses the majority of fabric interference issues without compromising Tino's aesthetic design.

4.3.5. Control System Adaptation for Differential Drive

The control system required comprehensive modification to support differential drive kinematics while maintaining compatibility with existing command interfaces.

Custom PID Controller Implementation

The PID controller development specifically targets differential drive characteristics, replacing the VirHas library dependency with custom control algorithms optimized for Tino's platform. The implementation utilizes classical PID control with Kp=7.3, Ki=5.6, and Kd=0.2 parameters tuned specifically for the MDD10A motor drivers and Tino's mechanical characteristics.

Motor speed calculation employs encoder feedback with 1920 pulses per revolution

(PPR) encoders to determine actual wheel velocities in rad/s. The `getMotorSpeed()` function calculates instantaneous speed based on encoder position changes over time intervals, enabling closed-loop speed control for precise movement execution.

The `updatePid()` function implements the complete PID algorithm with integral term reset on setpoint changes to prevent windup, derivative calculation for damping, and output limiting to ±255 PWM range. This ensures stable motor control without oscillation while maintaining responsive movement characteristics.

Atomic movement control integrates with the PID system through the `updateBaseMovementByTime()` function that manages four distinct movement states: resting (0), little push (1), timing preparation (2), and coordinated forward movement (3). Each state implements specific timing profiles and movement patterns that support synchronized leg-base coordination for VR integration.

The movement state machine ensures atomic operation completion where each movement phase must finish before accepting new commands, preventing interrupted motions that could compromise synchronized robot behavior. State transitions include proper timing validation and pending command management for seamless VR user interaction.

4.4. Power Supply System Redesign for Orin Nano

The power system redesign addresses the comprehensive requirements of the NVIDIA Orin Nano platform and associated high-performance components, necessitating complete architecture overhaul from the legacy Raspberry Pi power distribution system.

4.4.1. Power Requirements Analysis and System Specifications

The NVIDIA Orin Nano platform introduces significantly different power requirements compared to the original Raspberry Pi system, requiring comprehensive power architecture redesign to support enhanced computational capabilities.

Orin Nano Power Consumption Characteristics

The Orin Nano requires 19V DC input with power consumption characteristics that reach up to 2A during maximum computational load scenarios including simultaneous SLAM processing, human detection, audio processing, and ROS2 node operation. Typical operational consumption ranges between 1.3A to 1.4A during standard social interaction scenarios.

Power consumption analysis under various operational modes demonstrates peak consumption of 38W during maximum load conditions, with sustained operation typically requiring 25-27W. The power profile exhibits significant variation based on computational load, requiring robust power delivery capable of handling transient peaks without voltage drop.

Computational load correlation shows direct relationship between power consumption and processing intensity, with SLAM operations, TensorRT inference, and audio processing contributing the highest power demands. The variable load characteristics necessitate stable power delivery across the full operational range.

Auxiliary System Power Requirements

The Oak-D Pro camera system requires 5V DC input with power consumption up to 5W during high-resolution stereo processing. The camera power can be supplied directly from the Orin Nano USB ports or through dedicated power distribution for enhanced flexibility and reduced main processor loading.

The onboard router system requires 5V DC input for network connectivity and VR communication capabilities. Router power consumption remains relatively constant at approximately 3W during operational periods, providing stable power requirements for system design.

Total system power budget analysis indicates maximum power consumption of approximately 46W under peak operational conditions, with typical sustained operation requiring 33-35W. The power analysis includes safety margins for component aging and environmental variations.

4.4.2. DC-DC Converter Implementation and Power Distribution

The power conversion system utilizes high-efficiency DC-DC converters to transform battery voltage to the multiple voltage levels required by system components.

Oumefar 12V to 19V Step-Up Converter Selection

The Oumefar DC-DC step-up converter provides stable 19V output from 12V battery input with efficiency ratings exceeding 85% across the operational load range. Converter selection prioritized stability, efficiency, and thermal performance under the sustained loading conditions required for social robot operation.

Power delivery stability testing demonstrated consistent voltage regulation within ±2% across full load range with excellent transient response during computational load variations. Thermal performance analysis shows acceptable operating temperatures during sustained maximum load conditions without additional cooling requirements.

Load regulation characteristics maintain stable 19V output from no-load to maximum current draw, ensuring consistent Orin Nano performance across all operational scenarios. The converter includes overcurrent protection and thermal shutdown features that protect against component failure during fault conditions.

Secondary Power Rail Implementation

The 12V to 5V conversion system provides power for auxiliary components including the Oak-D Pro camera and onboard router. The secondary converter selection prioritized efficiency and multiple output capability to support various system components.

Power distribution architecture enables independent power control for auxiliary components, providing flexibility for future system expansion and reduced loading on the primary Orin Nano power rail. The distributed approach improves system reliability by isolating power domains.

Circuit protection includes individual fusing for each power rail and overcurrent protection that prevents system damage during component failure scenarios. The protection scheme enables continued operation of unaffected systems during partial power system failures.

4.4.3. Battery System Optimization and Consolidation

The battery system redesign consolidates multiple power sources while providing enhanced capacity and reliability for extended operational periods.

Battery Specification and Performance Analysis

The primary battery system utilizes 5200mAh 80C 11.1V 57.72Wh LiPo batteries that provide the power density and discharge capabilities required for robotic applications. Battery specification analysis demonstrates adequate capacity for 2-3 hours of typical operation with conservative discharge management.

Maximum load operational time calculations indicate approximately 1.37 hours of operation under peak power conditions, though realistic operational scenarios typically achieve 2+ hours due to variable computational loading.

Discharge characteristics analysis demonstrates stable voltage delivery throughout the operational range with minimal voltage drop during high current transients. The high discharge rate capability (80C) ensures stable power delivery during computational peaks without voltage sag.

Power System Consolidation Strategy

The system consolidation reduces complexity from four separate battery systems to three integrated power sources, eliminating the dedicated router power bank and USB power bank previously required for Raspberry Pi operation. Consolidation simplifies system startup procedures and reduces component count.

Battery management strategy includes individual monitoring and charging protocols for each power source while maintaining operational independence. The consolidated approach reduces overall system weight while providing enhanced operational capability through optimized power distribution.

Operational startup sequence simplifies through power system consolidation, reducing from multiple power-on procedures to streamlined system activation. The simplified approach reduces operator complexity and potential startup errors during experimental scenarios.

4.4.4. Cable Harness Redesign and Integration

The cable harness redesign eliminates legacy connections while implementing proper power distribution for the enhanced system architecture.

Legacy Connection Removal and Modernization

The cable harness modification removes obsolete USB-A and USB-C connections that were utilized for Raspberry Pi power delivery, replacing them with proper 12V distribution and 19V DC jack connectivity optimized for Orin Nano requirements. Connector selection prioritizes reliability and ease of maintenance.

Cable routing optimization reduces electromagnetic interference and mechanical stress while providing secure connections throughout the system. Harness design includes service loops and strain relief that accommodate system maintenance without cable damage.

Connector standardization utilizes consistent connector types throughout the system to simplify maintenance and reduce spare parts requirements. Standardization also reduces connection errors during system assembly and troubleshooting procedures.

Power Distribution Architecture

The 12V input distribution system provides primary power for both the step-up converter and the secondary 5V converter, enabling efficient power utilization from the primary battery source. Distribution includes proper circuit protection and monitoring capabilities.

The 19V DC jack implementation provides secure power connection to the Orin Nano with proper mechanical support and electrical contact reliability. Jack selection includes locking mechanisms that prevent accidental disconnection during operation.

Future expansion capabilities include additional power distribution points for system upgrades and experimental equipment integration. The modular power architecture supports system enhancement without complete harness redesign.

4.5. Stewart Platform Head Mechanism Improvements

The Stewart platform head mechanism underwent iterative design improvements to address systematic reliability issues and enhance performance under the operational loads encountered during social robot interaction scenarios.

4.5.1. Original System Analysis and Failure Modes

The original Stewart platform implementation exhibited multiple failure modes that compromised head movement precision and system reliability during extended operational periods.

Servo Axis Misalignment Problems

The original design suffered from servo axis misalignment that created excessive stress concentrations on servo motor internals during head movement operations. Force analysis revealed that head loads were transmitted directly through servo shafts rather than through the structural framework, creating potential failure-prone stress concentrations.

The misalignment created moment arms that amplified forces applied to servo components, requiring design revision to prevent potential servo damage. The servo axis alignment improvement was implemented as a preventive measure to ensure reliable long-term operation.

Mechanical analysis demonstrated that proper force transmission should flow through the head structure rather than servo mechanisms, necessitating fundamental design revi-

sion to achieve reliable operation under Tino's operational requirements.

Structural Flexibility and Arm Failures

The connecting arms exhibited excessive structural flexibility that reduced head positioning precision and contributed to mechanical instability during movement sequences. Flex analysis revealed inadequate stiffness-to-weight ratio in the original arm design, creating unwanted compliance in the kinematic chain.

Repeated arm failures occurred due to inadequate load distribution and material selection that could not withstand the combination of static head weight and dynamic movement forces. Failure analysis identified stress concentration points where 3D printed components experienced crack initiation and propagation.

Material selection limitations in PLA printing created brittleness under repeated loading cycles, with failure modes including layer delamination and stress cracking at critical load transfer points. The original material selection proved inadequate for the sustained loading requirements of social robot head movement.

4.5.2. First Design Iteration: Servo Axis Alignment

The first improvement iteration focused on servo axis alignment to redirect forces through proper load paths while maintaining the existing bearing-based connection system.

Force Path Optimization

The servo axis alignment improvement redirected head loads through the structural framework rather than servo mechanisms, reducing stress concentrations in servo internal components. Force analysis demonstrated significant reduction in lateral servo loading through proper geometric alignment.

Structural load distribution optimization utilized improved geometry to distribute head weight and movement forces across the Stewart platform structure. Load path analysis verified proper force transmission through structural components rather than servo mechanisms.

Performance evaluation demonstrated reduced servo stress indicators and improved movement precision, though structural flexibility issues remained due to the retained bearing connection system on the servo side of each arm.

3D Printed Component Enhancement

Enhanced PLA arm geometry provided improved load distribution through optimized cross-sectional design and stress concentration reduction. Geometric optimization utilized finite element analysis principles to identify optimal material distribution within manufacturing constraints.

Print orientation optimization aligned layer structure with primary stress directions to maximize strength characteristics within PLA material limitations. Layer orientation analysis demonstrated significant strength improvement through proper print setup procedures.

Initial performance testing showed improved reliability and reduced failure frequency, though continued structural flexibility limited overall system performance improvement. The partial solution validated the design approach while highlighting remaining system limitations.

4.5.3. Final Design Implementation: Rod End Integration

The final design iteration addressed remaining flexibility issues through the adoption of rod end (heim joint) connections on both ends of each Stewart platform arm.

Rod End Selection and Implementation

Research into Stewart platform best practices identified rod end connections as the standard solution for eliminating binding while allowing proper articulation throughout the movement range. Rod end selection prioritized load capacity, articulation range, and integration with existing servo and head mounting points.

The rod end implementation utilizes metal heim joints that provide superior strength and durability compared to the original bearing system that used metal bearings encased within 3D printed components. Metal construction eliminates the brittleness and wear issues encountered with the printed bearing housings.

Joint articulation characteristics enable free rotation in all necessary axes while providing positive mechanical connection between arm segments. The rod end design eliminates binding forces that contributed to servo stress and movement precision issues.

Mechanical Trade-offs and Performance Characteristics

The rod end implementation introduces acceptable head wobble during stationary periods due to the free articulation provided by the heim joints. Analysis indicates this

wobble may actually enhance Tino's expressive capabilities by providing natural movement characteristics.

Structural strength analysis demonstrates significant improvement in mechanical robustness and load-bearing capability while accepting reduced precision during stationary periods due to joint flexibility. The trade-off provides enhanced durability and operational reliability for social robot applications.

Load capacity testing validates the enhanced design's capability to handle operational loads without component failure or excessive wear. Longevity testing demonstrates sustained performance under extended operational scenarios typical of social robot research applications.

4.5.4. Hybrid Construction Approach

The final implementation combines 3D printed structural components with metal heim joints to achieve optimal balance between cost, performance, and maintainability.

Material Selection and Integration

The hybrid approach utilizes 3D printed PLA components for structural framework elements where weight and cost optimization provide system benefits. Metal heim joints provide superior performance at critical articulation points where durability and precision requirements exceed plastic capabilities.

Integration methodology ensures proper mechanical connection between printed and metal components through threaded interfaces and proper mechanical attachment techniques. Interface design accommodates thermal expansion differences and provides secure long-term connection.

Cost optimization through selective material usage maintains reasonable component costs while providing enhanced performance where critical to system operation. The approach enables high-performance characteristics without excessive cost impact on the overall system.

4.6. Camera Integration and Mounting Solutions

The camera integration system addresses the unique challenges of mounting sophisticated sensing equipment within Tino's soft fabric structure while maintaining the robot's aesthetic appeal and operational capabilities.

4.6.1. Mounting System Design and Mechanical Integration

The camera mounting system provides stable support for the Oak-D Pro camera while integrating with Tino's existing Stewart platform head mechanism and fabric covering system.

Tripod-Based Support System Development

The tripod mounting system utilizes simple brackets to create a fixed and stable camera support that eliminates the flexibility and vibration issues encountered with the original Raspberry Pi camera mount. Bracket design prioritizes rigidity while maintaining compatibility with the existing servo head structure.

Mechanical stability analysis demonstrates significant improvement in camera positioning stability compared to the flexible original mount that exhibited excessive movement during robot operation. Static deflection testing validates adequate stiffness for high-quality image acquisition during movement.

The camera mounting system operates independently from the Stewart platform head mechanism, providing dedicated stable positioning for the Oak-D Pro camera without mechanical coupling to head movements. This independent mounting ensures camera stability regardless of head articulation.

Structural Compatibility and Integration Challenges

The mounting system accommodates the existing servo head geometry while providing secure attachment points for the Oak-D Pro camera. Geometric constraints required custom bracket design that works within the available space while providing adequate support.

Weight distribution analysis ensures that camera addition does not compromise Stewart platform performance or exceed servo motor capabilities. Camera weight integration maintains head balance characteristics while providing enhanced sensing capabilities.

Mechanical interfaces utilize standard mounting hardware that enables camera removal for maintenance or upgrade without requiring bracket system modification. Modular approach provides flexibility for future system enhancement while maintaining operational reliability.

4.6.2. Fabric Integration and Visibility Solutions

The fabric integration system addresses the fundamental challenge of maintaining camera visibility while preserving Tino's fabric aesthetic and protecting sensitive camera components.

Fabric Modification and Positioning Strategies

Fabric positioning strategies prevent interference with camera sensing while maintaining the robot's aesthetic integrity. Positioning solutions accommodate fabric movement during robot operation without compromising camera field of view or image quality.

Velcro attachment systems provide secure fabric positioning that prevents fabric drift into camera field of view during operational periods. Attachment point selection utilizes strategic locations that maintain fabric appearance while providing reliable position control.

Testing procedures validate fabric positioning effectiveness under various operational scenarios including head movement, robot locomotion, and extended operational periods. Validation testing ensures consistent camera performance throughout typical social interaction scenarios.

Camera Protection and Environmental Considerations

Protection requirements address both physical protection from impact during social interaction and environmental protection from dust and moisture that could compromise camera operation. Protection solutions maintain camera accessibility for maintenance while providing adequate operational protection.

The protective approach balances camera protection needs against heat dissipation requirements and optical performance considerations. Protection system design ensures adequate camera cooling while preventing environmental contamination that could affect image quality.

Integration testing validates protection effectiveness while confirming maintained camera performance under operational conditions. Testing includes thermal performance evaluation and optical quality assessment under various environmental conditions.

4.6.3. Camera Shell Development and Implementation

The camera shell system provides comprehensive protection and fabric integration while maintaining optimal camera performance and heat dissipation characteristics.

Custom Enclosure Design and Functionality

The custom shell design provides camera protection while maintaining cooling airflow through strategic ventilation openings that enable heat dissipation without compromising environmental protection. Shell geometry optimizes airflow characteristics while minimizing dust ingress.

Attachment integration with the tripod mounting system provides secure shell mounting that enables camera access for maintenance while providing operational protection. Mounting system design enables shell removal without disturbing camera alignment or calibration.

Thermal management considerations ensure adequate heat dissipation during extended operation periods, particularly during high-resolution stereo processing that generates significant thermal loads. Thermal testing validates adequate cooling under maximum operational loads.

Velcro Fabric Control System

The fabric control system utilizes velcro attachments integrated with shell flaps to provide positive fabric positioning control. Flap design provides secure attachment points while maintaining fabric flexibility during robot movement.

Velcro implementation includes both shell-mounted components and fabric-sewn counterparts that provide reliable attachment while enabling fabric removal for maintenance. Attachment strength optimization provides secure positioning without excessive fabric stress.

Field testing validates velcro system effectiveness during extended operational periods including various robot movements and interaction scenarios. Testing confirms reliable fabric positioning without attachment failure or fabric damage.

4.6.4. Optical Performance Optimization

The camera integration system maintains optimal optical performance while addressing the unique challenges of fabric-integrated sensing systems.

Field of View Protection and Optimization

Field of view protection strategies ensure consistent camera visibility throughout robot operational ranges while accommodating fabric movement and positioning variations. Protection effectiveness validation includes testing under various lighting conditions and robot orientations.

Mesh covering implementation conceals camera presence from casual observation while maintaining full optical transmission characteristics. Mesh selection balances concealment effectiveness against optical performance impact, ensuring minimal image quality degradation.

Optical testing validates maintained image quality and depth sensing performance through the concealment system. Testing includes calibration verification and performance comparison against uncovered camera operation to ensure minimal performance impact.

Heat Management and Sustained Operation

Heat dissipation optimization addresses the Oak-D Pro camera's thermal requirements during sustained operation, particularly during high computational load scenarios including simultaneous SLAM processing and human detection.

Cooling system design utilizes strategic airflow management that enables heat dissipation while maintaining environmental protection. Thermal testing validates sustained operation capability under maximum computational loads without thermal throttling.

4.7. Audio System Integration

The audio system integration enables comprehensive bidirectional communication capabilities for VR integration and enhanced human-robot interaction through carefully designed hardware implementation and dynamic audio processing.

4.7.1. Hardware Component Selection and Specifications

The audio hardware selection prioritizes high-quality bidirectional communication capability while maintaining integration compatibility with Tino's existing system architecture and space constraints.

iTalk-01 Omnidirectional Microphone Integration

The iTalk-01 omnidirectional microphone provides 360-degree audio capture capability suitable for social robot interaction scenarios where human positioning relative to the robot varies continuously. Microphone specification analysis demonstrates adequate sensitivity and frequency response for speech capture and environmental audio monitoring.

Mounting considerations address the unique challenges of integrating audio capture equipment within Tino's fabric head structure while maintaining acoustic performance and physical protection. Microphone positioning optimization balances audio quality against mechanical protection and aesthetic integration requirements.

Fabric integration methodology enables microphone mounting through strategic fabric modifications that provide audio access without compromising Tino's visual appearance. Integration techniques utilize fabric properties to provide acoustic coupling while maintaining environmental protection for sensitive electronic components.

Speaker System Selection and Placement

The speaker system selection prioritizes clear audio reproduction capability within the geometric and weight constraints of Tino's head assembly. Speaker placement within the servo head maximizes available space utilization while providing optimal acoustic coupling for human interaction scenarios.

Acoustic performance optimization addresses the challenges of speaker operation within a constrained and partially enclosed environment. Speaker mounting utilizes available space within the servo head structure while ensuring adequate acoustic coupling and preventing mechanical interference with head movement systems.

Integration with existing head systems ensures speaker mounting does not interfere with Stewart platform operation, camera mounting, or other head-mounted systems. Geometric optimization provides secure speaker mounting while maintaining system accessibility for maintenance and adjustment.

4.7.2. Dynamic Audio Generation and Custom Sound Design

The audio system implements sophisticated sound generation capabilities that create custom ambient audio specifically designed for Tino's character expression and VR integration requirements.

No-Face Inspired Audio Algorithm

The dynamic audio generation utilizes a complex algorithm inspired by the No-Face character from Studio Ghibli films, creating an eerie and mysterious ambient sound that enhances Tino's social presence. The `noface_params` system maintains breathing phase control, breath cycle management, and dynamic volume adjustment to create organic, living audio characteristics.

The breathing simulation algorithm implements a state machine with inhale and exhale phases, utilizing sinusoidal wave patterns with randomized duration (2.5-4.0 seconds) and volume variations (0.55-0.75 amplitude range) to create natural breathing rhythm variations. The system maintains continuous audio output while avoiding mechanical repetition through breath parameter randomization.

Multi-tone synthesis combines three base frequencies (110Hz, 146.83Hz, 73.42Hz) with pre-calculated gain values (0.65, 0.45, 0.25) to create rich harmonic content. Filtered breath noise generation utilizes a first-order low-pass filter (coefficient 0.92) applied to random noise, creating organic texture that varies with breathing intensity.

The audio generation operates at 44.1kHz sample rate with 1024-sample chunks, ensuring high-quality audio output while maintaining real-time performance compatibility with other system components including SLAM processing and human detection.

Real-Time Audio Parameter Control

The audio control system receives real-time parameters from the VR interface through ROS2 topics, enabling dynamic audio modification based on user interaction and robot state. The `vr_audio_callback` function processes `Float32MultiArray` messages containing volume (0-255 range) and orientation (-1.0 to 1.0 range) control values.

Volume control implementation applies logarithmic scaling to provide natural perceived loudness variations while maintaining system audio output limits. The volume parameter directly scales the final audio amplitude, enabling seamless integration with VR-controlled audio levels based on proximity and interaction intensity.

Orientation-based parameter modulation enables frequency shifting and audio characteristic changes based on spatial positioning. The orientation parameter influences base frequency modulation ($\pm 20\%$ frequency variation) and breathing pattern intensity, creating spatial audio awareness that enhances immersive VR interaction.

4.7.3. Stereo Speaker Configuration and Spatial Audio Processing

The stereo speaker system provides sophisticated spatial audio capabilities through advanced panning algorithms and stereo field manipulation optimized for social robot interaction scenarios.

Enhanced Stereo Panning Implementation

The stereo panning system implements aggressive channel separation techniques that create dramatic spatial audio effects controlled by VR input orientation values. The panning algorithm utilizes non-linear curves with exponential falloff characteristics to provide pronounced left-right audio separation that enhances directional audio expression.

Channel separation calculations apply quadratic intensity scaling where `pan_intensity = min(1.0, abs(orientation) * 1.5)` creates exponential panning response curves. Right-side dominance utilizes `left_vol = volume_scale * (1.0 - (orientation_effect * 1.1 + pan_intensity2 * 0.5))` while left-side dominance applies equivalent mirrored calculations for balanced spatial control.

Channel boost implementation provides 30% amplitude enhancement on the dominant channel (`right_vol = volume_scale * (1.0 + orientation_effect * 0.3)`) while applying aggressive attenuation up to 95% on the opposite channel. This creates pronounced stereo separation that maintains 5% minimum volume to prevent complete channel silence.

The panning system maintains real-time responsiveness with orientation updates processed at audio sample rate, enabling smooth spatial transitions that follow VR user head movements without audible artifacts or discontinuities.

Audio Expression and Robot Personality Integration

The stereo audio system enhances Tino's expressive capabilities by correlating spatial audio characteristics with robot emotional states and interaction contexts. Spatial audio positioning creates directional personality expression where left-right audio bias suggests attention direction and emotional focus.

Breathing pattern modulation integrates with stereo positioning to create complex audio landscapes where breath intensity, frequency content, and spatial positioning combine to express robot internal states. The system maintains acoustic coherence while providing rich expressive variation through multi-dimensional audio parameter control.

VR integration enables precise control over robot audio personality through real-time parameter adjustment, allowing users to influence robot audio expression through spatial positioning, proximity, and interaction intensity. This creates responsive audio feedback that enhances social presence and interaction naturalness.

4.8. YOLOv11 Pose Detection Implementation with TensorRT Optimization

The YOLOv11 pose detection system provides real-time human skeleton tracking capabilities optimized for the NVIDIA Orin Nano platform through advanced TensorRT acceleration and efficient model deployment strategies.

4.8.1. YOLOv11 Architecture Selection and Model Optimization

The YOLOv11 architecture selection prioritizes the balance between detection accuracy and computational efficiency required for real-time embedded operation on the Orin Nano platform.

YOLOv11n-Pose Model Selection

The `yolo11n-pose.pt` model provides optimal performance characteristics for the Orin Nano's computational constraints while maintaining adequate accuracy for social robot interaction requirements. The nano variant (11n) utilizes a streamlined architecture with reduced parameter count compared to larger YOLO variants, enabling real-time inference on embedded hardware.

Model architecture analysis demonstrates the YOLOv11n-pose network's capability to detect multiple humans simultaneously within the camera's field of view while maintaining frame rates suitable for interactive applications. The model's optimized backbone reduces computational overhead while preserving essential feature detection capabilities for robust human pose estimation.

Pre-trained model utilization eliminates extensive training requirements by leveraging weights trained on large-scale pose estimation datasets. The pre-trained approach provides immediate deployment capability while ensuring robust performance across diverse human poses and environmental conditions typical of social robot applications.

Model Format Conversion Pipeline

The model optimization pipeline transforms the original PyTorch format through multiple conversion stages to achieve maximum performance on the target hardware. The conversion process begins with the native `.pt` PyTorch format and progresses through ONNX and TensorRT formats for optimal deployment.

ONNX format conversion (`.onnx`) provides cross-platform compatibility and serves as an intermediate representation that enables hardware-specific optimizations. The ONNX conversion maintains model accuracy while enabling subsequent optimization passes that target the Orin Nano's specific GPU architecture.

TensorRT engine generation (`.engine`) represents the final optimization stage, creating hardware-specific inference engines that maximize GPU utilization on the Orin Nano platform. The TensorRT optimization process includes layer fusion, precision calibration, and memory access optimization that significantly improve inference performance compared to standard deployment methods.

4.8.2. TensorRT Engine Optimization Process

The TensorRT optimization process generates highly efficient inference engines specifically optimized for the Orin Nano's GPU architecture and memory hierarchy.

Engine Generation and Hardware Optimization

TensorRT engine generation utilizes the Orin Nano's specific GPU capabilities to optimize network execution through layer fusion, kernel optimization, and memory access pattern optimization. The process analyzes the YOLOv11n-pose network structure and generates optimized CUDA kernels that maximize throughput while minimizing memory bandwidth requirements.

Precision optimization techniques evaluate the network's sensitivity to reduced precision arithmetic, enabling mixed-precision computation that balances accuracy against performance. The optimization process may utilize INT8 quantization where appropriate while maintaining FP16 or FP32 precision for critical network layers that require higher numerical accuracy.

Memory allocation strategies optimize GPU memory usage patterns to prevent memory fragmentation and minimize allocation overhead during inference operations. The TensorRT engine pre-allocates memory buffers and optimizes data transfer patterns between CPU and GPU memory to minimize inference latency.

Performance Profiling and Validation

Engine performance validation ensures that TensorRT optimizations maintain pose detection accuracy while achieving target inference rates. Performance profiling includes accuracy benchmarking against the original PyTorch model to verify that optimization processes do not compromise detection quality.

Throughput analysis measures inference performance under various loading conditions including single and multi-person detection scenarios. Performance metrics include inference latency, GPU utilization, and memory bandwidth utilization that validate the engine's suitability for real-time social robot applications.

Thermal analysis ensures that sustained inference operations remain within the Orin Nano's thermal limits during extended operation periods. Thermal validation includes continuous operation testing under maximum computational loads to ensure system stability and reliability.

4.8.3. ROS2 Node Architecture and Integration

The pose detection system integrates seamlessly with Tino's ROS2 architecture through a dedicated node that manages camera input, inference execution, and result publication.

Camera Topic Subscription and Data Flow

The `pose_detection_node.py` subscribes to Oak-D Pro camera topics including `/right/image_re` for monocular RGB input and `/stereo/depth` for corresponding depth information required for 3D coordinate calculation. The node implements efficient message handling with `CvBridge` conversion from ROS2 image messages to OpenCV format suitable for YOLO processing.

Image preprocessing includes monocular to BGR conversion for YOLO compatibility, since the implementation uses the right camera feed converted from mono8 to BGR format. The preprocessing pipeline utilizes CPU-based OpenCV operations for format conversion while maintaining real-time performance through efficient memory management.

Synchronization mechanisms maintain temporal alignment between RGB and depth image streams through callback-based processing that ensures corresponding depth information is available for each processed RGB frame. The implementation uses a latest-frame approach where `latest_image` and `latest_depth_image` are synchronized for pose detection processing.

Inference Execution and Result Processing

TensorRT inference execution utilizes the pre-converted `yolo11n-pose.engine` model loaded from the package resource directory through `YOLO (model_path)` initialization. The inference pipeline processes BGR images directly with configurable confidence thresholding (default 0.5) and `verbose=False` parameter to suppress detection logging output.

Result post-processing extracts pose keypoints, confidence scores, and bounding box information from the `results[0]` object returned by YOLO inference. Post-processing operations include confidence-based filtering, closest person selection based on depth measurements, and coordinate extraction from the `results[0].boxes` and `results[0].keypoints` attributes.

Error handling implements comprehensive exception management with detailed logging at configurable levels (INFO, DEBUG, ERROR) and graceful degradation when model loading fails or inference encounters errors. The system maintains operational continuity through try-catch blocks around critical processing sections.

4.9. Stereo Depth Integration for 3D Human Positioning

The stereo depth integration system combines 2D pose detection results with Oak-D Pro depth information to provide accurate 3D human positioning capabilities essential for spatial awareness and interaction planning.

4.9.1. Oak-D Pro Depth Data Utilization

The Oak-D Pro camera system provides synchronized stereo depth information that enables precise 3D coordinate calculation for detected human pose keypoints.

Stereo Depth Acquisition and Processing

The Oak-D Pro stereo camera system generates depth maps through stereo vision algorithms that calculate depth values for each pixel in the synchronized color image. Depth data accuracy depends on stereo baseline, camera calibration quality, and environmental factors including lighting conditions and surface textures.

Depth value extraction utilizes a median-based approach with a 9x9 pixel window around keypoint locations to obtain robust depth measurements. The `get_depth_at_point` function implements outlier filtering by calculating median and standard deviation, then

filtering values more than 2 standard deviations from the median to ensure reliable depth extraction.

Depth data validation implements comprehensive quality filtering including zero-value rejection, outlier detection based on statistical analysis, and temporal smoothing over a configurable window size (default 3 frames). The validation process uses median filtering within pixel windows and consistency checks against reference depth values to maintain measurement reliability.

For person detection, the system uses `get_median_depth` function that extracts depth from a central region of the bounding box with configurable padding (25% by default), providing more stable depth measurements than single-pixel sampling while avoiding background contamination.

Coordinate System Transformation

Camera frame to robot coordinate transformation converts 3D keypoint coordinates using the `calculate_3d_position` function that utilizes camera intrinsic parameters from `camera_info.k` array. The transformation extracts focal lengths `fx`, `fy` and principal point coordinates `cx`, `cy` from the camera calibration matrix.

Intrinsic parameter utilization includes automatic unit conversion detection where depth values greater than 100 are converted from millimeters to meters, followed by depth calibration correction using configurable `depth_scale_factor` (default 0.575) and `depth_offset` parameters derived from empirical calibration measurements.

3D coordinate calculation applies the standard pinhole camera model: `x_3d = (x - cx) * z / fx` and `y_3d = (y - cy) * z / fy`, where corrected depth `z = z * depth_scale_fact + depth_offset` accounts for systematic depth measurement biases in the Oak-D Pro stereo system.

4.9.2. 3D Skeleton Generation and Validation

The 3D skeleton generation process combines 2D keypoint detections with corresponding depth values to create complete 3D human pose representations.

Keypoint Depth Association

Depth value assignment utilizes the 2D keypoint pixel coordinates to extract corresponding depth measurements from the stereo depth map. The assignment process includes validation to ensure depth measurements correspond to human body parts rather

than background objects.

Depth consistency validation utilizes a sophisticated temporal smoothing system where individual keypoint depths are validated against a reference depth calculated as the median of all valid keypoint depths. The `process_skeleton` function implements outlier rejection where keypoint depths deviating more than the configurable `depth_outlier_threshold` (default 30%) from the reference are replaced with the temporal median depth from a smoothing window.

Missing depth handling addresses invalid keypoint scenarios through the reference depth fallback system. When keypoint coordinates are marked as invalid ($x \leq 0$ or $y \leq 0$) or depth extraction fails, the system utilizes the temporally smoothed reference depth calculated from all valid keypoints in the current frame, ensuring skeleton completeness even with partial occlusion.

The temporal smoothing implementation maintains a `skeleton_depth_history` list with configurable window size (default 3 frames) that stores reference depths for median filtering over time, providing stable depth values that reduce measurement jitter while maintaining responsiveness to actual depth changes.

4.9.3. Real-time Processing and Performance Optimization

The 3D positioning system maintains real-time performance while processing high-resolution depth data and performing complex coordinate transformations.

4.10. Real-time Skeleton Tracking with 17 Key Body Joints

The skeleton tracking system processes YOLOv11 pose detection results to extract and organize 17 standard COCO keypoints into structured human pose representations suitable for social robot interaction analysis.

4.10.1. COCO Keypoint Framework Implementation

The COCO keypoint framework provides a standardized representation for human pose detection that ensures compatibility with established computer vision tools and datasets.

17-Keypoint Detection Schema

The YOLOv11 pose detection system identifies 17 key body joints following the COCO pose estimation standard: nose (0), left eye (1), right eye (2), left ear (3), right ear (4), left shoulder (5), right shoulder (6), left elbow (7), right elbow (8), left wrist (9), right wrist (10), left hip (11), right hip (12), left knee (13), right knee (14), left ankle (15), and right ankle (16).

Joint indexing follows the established COCO convention ensuring compatibility with existing pose analysis tools and datasets. The consistent indexing scheme enables straightforward integration with pose analysis algorithms and facilitates comparison with other human pose detection systems.

Keypoint connectivity defines the skeletal structure through predefined joint relationships that represent human anatomical connections. The connectivity graph includes head structure (nose-eyes-ears), torso connections (shoulders-hips), and limb chains (shoulder-elbow-wrist, hip-knee-ankle) that enable skeletal validation and pose completeness assessment.

Confidence Scoring and Quality Assessment

Confidence score extraction provides reliability metrics for each detected keypoint, enabling quality-based filtering and uncertainty quantification. Confidence values range from 0.0 (undetected) to 1.0 (high confidence) and indicate the detection algorithm's certainty in keypoint localization accuracy.

Quality thresholding implements confidence-based filtering that excludes low-confidence detections from downstream processing. Threshold values balance detection completeness against accuracy requirements, with typical thresholds ranging from 0.3 to 0.7 depending on application requirements and environmental conditions.

Multi-person confidence handling manages confidence scores when multiple humans are detected simultaneously. The system maintains separate confidence profiles for each detected person while implementing consistency checks that ensure skeletal coherence within individual pose detections.

4.10.2. Data Processing Pipeline and Skeleton Organization

The data processing pipeline transforms raw YOLOv11 output into structured skeleton representations suitable for real-time robot applications.

Keypoint Coordinate Extraction

Raw network output processing extracts keypoint coordinates and confidence scores from YOLOv11 inference results. The extraction process handles variable-length outputs that accommodate different numbers of detected humans while maintaining consistent data structures for downstream processing.

Coordinate normalization converts pixel coordinates to standardized coordinate systems that enable consistent processing regardless of camera resolution or field of view variations. Normalization includes scaling, translation, and coordinate system alignment that simplifies subsequent geometric calculations.

Skeleton Structure Validation

Geometric consistency validation implements anatomical constraints that verify reasonable joint relationships within detected skeletons. Validation includes bone length checks, joint angle limitations, and bilateral symmetry assessments that identify and filter implausible pose detections.

Temporal consistency analysis compares consecutive pose detections to identify and smooth measurement noise while detecting rapid pose changes that indicate genuine human motion. Temporal filtering balances noise reduction against motion tracking accuracy to maintain responsive pose tracking.

Completeness assessment evaluates skeleton quality based on the number and distribution of successfully detected keypoints. Assessment criteria include minimum keypoint requirements, critical joint detection (head, torso, limbs), and pose coverage metrics that indicate skeleton suitability for specific applications.

4.10.3. ROS2 Message Publishing and Data Distribution

The skeleton tracking system publishes pose data through ROS2 topics enabling integration with other robot systems and applications.

Custom Message Structure Design

ROS2 message design implements multiple specialized publishers for different data requirements: `/pose_detection/image_raw` for annotated detection images, `/human_position` for smoothed human position using `PoseStamped` messages, `/human_skeleton` for 3D skeleton visualization using `MarkerArray`, and `/human_skeleton_poses` for programmatic access to joint coordinates using `PoseArray` messages.

Multi-person message handling focuses on closest person selection based on depth measurements, where the system identifies the person with minimum depth value and processes only that individual's skeleton data. This approach reduces computational overhead while ensuring consistent tracking of the most relevant human for interaction applications.

Timestamp synchronization utilizes `self.get_clock().now().to_msg()` for all published messages with consistent `header.frame_id` set to the configurable camera frame (default: `oak_right_camera_optical_frame`), ensuring temporal alignment across all pose-related data streams.

4.10.4. Integration with Robot Controller and VR Systems

Skeleton data integration enables sophisticated human-robot interaction capabilities through real-time pose information distribution to various robot subsystems.

Robot Controller Integration

The robot controller subscribes to `/human_position` topic for spatial awareness applications where human position data undergoes temporal smoothing through the `publish_human_positi` function. This implementation maintains a configurable history buffer (default 5 positions) and publishes averaged coordinates to reduce measurement jitter while maintaining responsiveness to human movement.

Pose-based behavior triggers can utilize the multiple data streams including raw skeleton data from `/human_skeleton_poses` for detailed joint analysis, smoothed position data from `/human_position` for proximity detection, and visual markers from `/human_skeleton` for debugging and visualization purposes.

Safety monitoring implementation prioritizes closest person detection where the system automatically selects the human with minimum depth measurement for tracking, ensuring safety systems focus on the most immediate interaction partner while maintaining computational efficiency through single-person processing.

VR Data Recording and Transmission

VR system integration transmits skeleton data for immersive visualization and interaction analysis applications. Integration includes data formatting for Unity applications and network transmission protocols optimized for VR system requirements.

Data recording capabilities capture complete skeleton tracking sessions for offline analysis and system evaluation. Recording includes synchronized pose data, camera images,

and robot state information that enable comprehensive interaction analysis and system performance assessment.

Bandwidth optimization implements data compression and selective transmission strategies that maintain VR system responsiveness while minimizing network overhead. Optimization techniques include keyframe detection, delta encoding, and adaptive quality adjustment based on network conditions and VR application requirements.

4.11. VR System Architecture and Unity Communication

This section will detail the comprehensive VR integration system that enables remote control and monitoring of Tino through Unity-based VR environments. The VR interface architecture will be explained first, covering the ROS2 `vr_interface_node` that serves as the central communication bridge between the robot's ROS2 system and external Unity applications, and the UDP communication protocol that provides real-time bidirectional data exchange for low-latency VR interaction. The Unity integration capabilities will be detailed, including the message structures for sending robot control commands from VR to ROS2 topics, the data reception system that provides robot pose, human detection, and audio information to Unity for visualization and interaction, and the networking configuration that enables flexible deployment across different network environments. The communication monitoring system will be examined, covering the configurable send rates for pose and skeleton data transmission, the health monitoring that tracks communication status and detects connection failures, and the message ordering system that ensures reliable data delivery and duplicate detection. The VR data recording functionality will be discussed, including the comprehensive recording system that captures all VR-relevant data streams for offline analysis.

4.12. Atomic Movement System Design and 4-State Control Architecture

This section will present the revolutionary atomic movement system designed specifically for natural VR interaction, replacing the previous continuous control scheme with discrete, completion-guaranteed movements. The 4-state control framework will be explained first, covering the unified state architecture applied to both leg and base controllers where state 0 represents idle/resting position, state 1 implements expressive "little push" movements

for attention-getting behaviors, state 2 provides timing synchronization cycles, and state 3 executes atomic movements that must complete before new commands can be processed. The leg controller implementation will be detailed, including the state 1 optimized 3-phase movement (50% forward extension, 5% pause, 45% return), the state 2 forward extension to maximum reach with position locking mechanisms, and the state 3 return-to-neutral movement with button-press completion detection. The base controller design will be examined, covering the state 1 rapid forward-backward sequence for expressive pointing behaviors, the state 2 timing cycle that provides 1.5-second synchronization delay, and the state 3 atomic movements including forward translation and left/right rotation operations, each with 1.7-second execution duration. The synchronization architecture will be discussed, including the sophisticated locking system that prevents base state 3 execution until leg state 2 completion, and the pending command system that stores VR commands during ongoing operations and automatically executes them upon completion.

4.13. Pulse-Based Command System for VR Integration

This section will detail the pulse-based command architecture that ensures perfect correspondence between VR user actions and physical robot movements. The pulse generation system will be explained first, covering the replacement of continuous signal transmission with discrete 3-cycle command pulses that automatically return to idle state, ensuring each VR interaction triggers exactly one complete robot movement cycle. The gamepad integration modifications will be detailed, including the removal of analog joystick control in favor of discrete button-based state commands, and the implementation of pulse timing that provides consistent command duration regardless of user input duration. The VR command processing will be examined, covering the UDP packet structure that transmits head control data (pitch, pan, tilt), base movement commands (state and angular direction), and audio parameters (volume and orientation), all synchronized with message ordering for reliable delivery. The atomic guarantee system will be discussed, including the movement completion assurance that prevents partial operations, the state machine locks that maintain movement integrity, and the natural interaction flow that ensures VR users always observe complete robot actions rather than interrupted movements. The timing optimization will be addressed, covering the precise 1.5-second state 2 timing cycle, the 1.7-second state 3 movement duration, and the synchronization mechanisms that coordinate multi-component movements for realistic dragging simulation.

4.14. Unity-ROS2 Communication Protocol and Message Structures

This section will present the comprehensive communication protocol designed for robust Unity-VR to ROS2 integration with optimal performance and reliability. The UDP communication architecture will be explained first, covering the multi-port configuration with port 5005 for incoming VR commands, port 5006 for outgoing robot pose data, and port 5007 for human skeleton transmission, enabling parallel data streams without interference. The incoming message format will be detailed, including the 32-byte VR command packets containing 3 floats for head control (pitch, pan, tilt), 2 integers for base commands (state 0–3, angular direction $-1/0/1$), 2 values for audio control (volume and orientation), and 1 integer for message ordering to detect lost or duplicate packets. The outgoing data structures will be examined, covering the 24-byte robot pose packets with position and orientation data fused from UWB and RTAB-Map systems, and the 208-byte skeleton packets containing exactly 17 COCO-format joints with consistent 3D coordinates for missing or occluded body parts. The configurable transmission rates will be discussed, including independent control of pose data frequency (default 10Hz), skeleton data frequency (default 10Hz), and expected incoming command rate (default 25Hz) to optimize performance for different network conditions and VR application requirements. The monitoring and debugging capabilities will be addressed, covering the comprehensive logging system that tracks communication health, the rate validation that ensures expected data flow, and the error detection mechanisms that identify connection problems and provide detailed diagnostic information for system maintenance.

4.15. Bidirectional Audio Communication and Spatial Processing

This section will detail the advanced audio communication system that enables natural voice interaction between VR users and the physical robot environment. The audio data flow architecture will be explained first, covering the microphone input processing that captures robot-side audio and transmits it to VR systems through ROS2 topics, the VR audio reception that provides spatial audio information with volume and orientation parameters for immersive sound positioning, and the bidirectional communication that enables real-time voice interaction between VR users and people in the robot's physical environment. The audio processing implementation will be detailed, including the 16-bit PCM audio sample handling through Int16MultiArray message structures, the real-time

audio streaming that maintains low latency for natural conversation flow, and the volume and orientation control system that allows VR applications to adjust audio characteristics based on virtual positioning and interaction context. The spatial audio integration will be examined, covering the orientation parameter system that provides directional audio information in degrees, the volume control mechanisms that enable distance-based audio attenuation simulation, and the Unity integration capabilities that support immersive audio experiences in VR environments. The practical applications will be discussed, including the human-robot interaction enhancement through voice communication, the remote presence capabilities that allow VR users to participate in physical environment conversations, and the research data collection features that record audio interactions for analysis of human-robot communication patterns and social interaction behaviors.

5 | Evaluation

6 | Conclusions

7 | Temporal R&D

Localization Technologies

Onboard Sensing

Technology	**Pros**	**Cons**	**Key Papers & Resources**
Visual Odometry (VO)	Could use existing camera; no hardware mods	Narrow FOV; tilt disrupts SLAM	[3] – Robust monocular/Stereo SLAM. [?] for Monocular and Multicamera Systems
IMU + Wheel Encoders	Low cost; integrates motion data	Drift over time; Stewart tilt issues	[?] – Kalman filtering. [16] for Mobile Robot Localization
UWB-IR	Small footprint; could work with fabric	Requires external anchors	[19] for Indoor Robot Navigation.

External Sensing

Technology	Pros	Cons	Key Papers & Resources
UWB Anchors	High accuracy; no line-of-sight	Setup/calibration required	[12] in non-cooperative industrial environments
AprilTags	Low cost; precise	Line-of-sight; limited area	[24].
MoCap Systems	Sub-mm accuracy	Expensive; fixed environment	[25] – Industrial use cases.

Orientation Technologies

- **Sensor Fusion**: [**?**] filters for combining UWB, IMU, and encoders. (Waiting for access request)
- **UWBOri**: [6] with ultra-wideband signals
- **NLOS Mitigation**: [7] and Tracking With Arbitrary Target Orientation, Optimal Anchor Location, and Adaptive NLOS Mitigation
- **RPO**: [33] using UWB

Human Detection Technologies

Onboard Sensing

Technology	Pros	Cons	Key Papers & Resources
Thermal Cameras	Works in darkness; fabric-friendly?	No depth; limited range	[30] – CNN-based approaches.
Ultrasonic Array	Low cost; proximity detection	No human distinction	[32] (In Korean).
Upgraded Camera	Wider FOV; ML-compatible	Fabric obstruction; compute-heavy	[36] – Real-time object detection.

External Sensing

Technology	Pros	Cons	Key Papers & Resources
RGB-D Cameras	Depth data; multi-human tracking	Fixed installation	[4]. [28] – Performance Analysis of Body Tracking.
LiDAR	High-resolution 3D mapping	Expensive; compute-heavy	[37].
WiFi/Radar	Privacy-friendly; fabric-penetrating	Lower resolution	[17]: A New Way to Observe Surroundings

Technologies for Tino Robot Implementation

Localization Technologies

Visual Odometry (VO)

- **Variants:**
  - *ORB-SLAM3* (supports RGB-D): [34]
    - + Synergy with human detection via depth data
    - + Robust feature matching for dynamic environments
    - – Higher computational cost (requires GPU optimization)
  - *SVO* (Semi-direct Visual Odometry): [35]
    - + Works with fisheye/catadioptric cameras (wide FOV)
    - + Lower computational footprint
    - – Less accurate in textureless environments
- **Shared Advantage:** Dual-purpose for localization & human detection

UWB-IR Localization

- + Centimeter-level accuracy (theoretical)
- + Low power consumption
- – Requires external infrastructure (anchors)
- – Fabric penetration uncertainty (needs RF testing)

  - No native orientation data ⇒ Requires:
    - IMU sensor fusion (Kalman filtering)
    - RPO/UWBOri techniques (experimental)
    - NLOS mitigation strategies

Wheel Encoders + IMU

+ Low-cost solution
- Unsuitable for impulse-based movement (slippage errors)
- IMU drift accumulates over time
- Poor performance on uneven surfaces

Human Detection Technologies

RGB-D Camera (e.g., Intel RealSense)

+ Simultaneous color + depth data
+ Enables skeleton tracking (OpenPose, MediaPipe)
- Requires careful physical integration (size/visibility)
- Limited range (typically <5m)

Thermal Imaging

+ Potential fabric penetration capability
+ Works in low-light conditions
- No depth sensing ⇒ Requires fusion with VO
- Limited contextual information (heat-only data)

ML-Enhanced 2D Camera

+ Lower profile than RGB-D
+ Modern architectures (YOLOv8, EfficientNet) enable real-time detection
- Requires depth estimation via:
  - Monocular depth networks (MiDaS, LeReS)
  - Sensor fusion with other localization data

Lidar

- Impractical due to Tino's soft structure (vibration issues)
- High cost-to-benefit ratio
- Overkill for indoor social robot ranges

Recommended Hybrid Approach

- **Localization:** ORB-SLAM3 with RGB-D camera (despite computational cost) + optional UWB for absolute positioning
- **Human Detection:** Thermal camera + RGB-D fusion (if concealable) or ML 2D camera with monocular depth estimation
- **Backup:** SVO with fisheye lens as fallback if RGB-D integration fails

Week 18 Mar R&D on different techs

Week 25 Mar Task: Work on Orin nano testing cameras ZED 2, and Orb slam 3 with webcam and Realsense T265 Result: The Zed camera that was available was not functioning properly, orbslam had a lot of bugs in terms of compilation given is an old library that is not being maintained.

Week 1 Apr Task: Work on Orin Nano and Relsense T265 to try and make SLAM atlas creation and load Result: The T265 was deprecated so I had to install an old version of librealsense (2.53) in order to make the camera be detected, even after camera detection I was able to run the camera with orbslam but the accuracy was very low, my initial tought is that it was because of poor calibration. Orbslam had some issues with the camera, in stereo inertial was the best mode that it worked but it needed some acceleration in order to start outputing some video, also it took a long time to actually grap into something (features) so to actually start creating a map, in only stereo it crashed, same as in Mono

Week 08 Apr Task: keep working on Realsense T265 and most important save and load atlas Result: even tho it had a lot of issues the first thing that was tested is calibrating the camera to see if the detection improved, it didnt, then i tried saving the atlas but given the old library it always ended in a crash. After looking on the web I found a git fork that "fixed" this atlas save and load, testing the library i found that it managed to save but it always crashed when trying to open the atlas back, either that or it starts creating a new map from scrathc. Given all of the issues that the orbslam3 had i tried using SVO, it had again a lot of issues given its an old not maintained library, mainly in the compilation part as i am working in arm64 so i had to fix a lot of flags in order to make it compile in arm64. even after all of the work trying to make it build in a container i had a same result that with orbslam, loaded the map, mapped something (not that accurrate) and they did not have any atlas/map management so I scraped that work. Given that with 2 systems i had simmilar issues i tought it could be related to the Realsense T265, so I requested if a D435I was available (given that the video examples used in orbslam3 are with that camera), in the end that camera was not available but I was provided with a

oak-d pro, after testing the basic functionallity with the depthAi library I tried checking for slam approaches, they had a community fork of orbslam3 using that camera and a guide to try and build it with lxc, but after trying a lot I was not able to pass the camera to the container in a way that it was detected as a bootable device. One of the other options Luxonics mentrioned was using RtabMap, so thats what I was set to try

Week 15 Apr Task: Try to set the Oak-D pro to work with Rtabmap Result: The first thing I had to do was try to build the library, it had a lot of dependencies and with that a lot of issues to be fixed, the first time I tried to build the standalone version, this didnt work at first because the depthAI library was not detected. Then I tried to build the Ros version of rtab but this one had not a proper implementation between depthAI and the ros wrapper

Then I tied again with the standalone version and this time I was able to make it link with the depthAI, and it worked amaizing, I did some tests with the camera over Tino moving around, this showed that Rtab was working really well creating a map and most importantly it was able to save a map and then relocalize itself in that map. Now the next step was how to get the data out of the standalone version, how to get the position and orientation.

After some trial an error managed to install and run it with ros2 using the depthai_ros and the rtabmap_ros, it publishes the localization_pose topic that has all of the importnat information

Managed to load the correct map, I refactored the old Tino source code to work with Ros2, created the respective topics and the needed structure. Created respective launch files for mapping and localization modes Added human detection, this system works by subscribing to the same camera topic and run it with yolo11 in tensorRt format. This provides all of the information needed for the human detection getting all of the skeleton pose joints, getting the depth (using the stereo camera info) and position in relation to the robot.

Also by creating this ros version I created a node for handling the VR connection in the future.

Apr 19 Major milestone reached: Complete system architecture migration from legacy Raspberry Pi based system to ROS2 based implementation. Restructured the entire workspace by creating tino_ws for ROS2 development and moved all legacy code to legacy_tino folder for preservation.

Implemented the core ROS2 node architecture:

- gamepad_node.py: Handles Xbox controller input with proper D-input to X-input conversion for Jetson compatibility
- hardware_interface_node.py: Manages serial communication with all 3 Arduino systems (head, base, leg) using proper device symlinks
- robot_controller_node.py: Central coordination node that manages all robot behaviors and movement commands
- vr_interface_node.py: Handles VR system integration and data exchange for future Unity integration

Created launch files for both mapping (rtab_mapping.launch.py) and localization (rtab_localizatio modes, allowing seamless switching between SLAM creation and navigation modes.

This migration allows for much better modularity, debugging capabilities, and integration with the VR system compared to the monolithic Python scripts from the legacy system.

Apr 22 Next task was adding audion in/out to the system. I was provided with a omnidirectional mic iTalk-01, and a pair of speakers. Impelemented a system that gets the data and publishes it to the vr, and also receives from the vr and publishes to the speakers.

Apr 23 Major advancement in human detection capabilities: Implemented pose detection functionality using YOLOv11 with TensorRT optimization. The system now provides real-time human skeleton tracking with 17 key body joints detection, including depth information using the stereo camera data. This allows Tino to not only detect humans but also track their pose and calculate their 3D position relative to the robot.

Added audio handling nodes (audio_node.py and audio_loopback.py) and fully integrated audio functionality into both VR and robot controller systems. The audio system now supports bidirectional communication: capturing audio from the omnidirectional microphone and publishing it to VR, while also receiving audio from VR and playing it through the speakers.

Enhanced gamepad handling with improved command processing and error reporting, making the control system more robust and responsive.

At this point most of the internals where ready We bought a display port dummy in order to have good performance when connected via vnc because the Orin Nano does not run headless by default

One of the head supports broke so we had to print a new one with more internal support

Next steps is hardware related. we need to: Build the new kinematic base that can support tino weight Fix the head supports Add the power supply needed to support the Orin Nano

Apr 29 Started by dissasembling the robot completely into the main 4 parts Fabric head Servo Head Body Kinematic base

First I modified the Servo head by adding a trypod that can hold the camera, this was done with simple brackets to make the support fixed and steady, specially because the old camera mount (that was for a pi camera) was very very flexible and moved a lot Then I tested the power supply, we got a powerful and stable 12v to 19v DC DC step up converter Oumefar, using this proved and testing the Orin Nano at max power, so with all of tino system actives (SLAM, audio, ROS) it reached a max of 2A of consumption, this from a 12v battery They are 5200 mAh 80c 11.1v 57.72Wh gave a approximate time of 1.37 hours during max consumption, but this really is not accurate as the jetson usually works between 1.3 to 1.4 A https://chatgpt.com/c/68125c25-216c-8000-a956-52b2702d04b8

Given that this will fix the power supply issue I modified the cable harness to remove the old USBA and USBC that powered the Raspberry from a powerbank, and replaced it with the 12V input and the 19V DC jack the Orin needs Also we added a 12v to 5V converter connected to the same 12v battery to power the Onboard router and the Oak-D camera the camera can be powered by the orin but we wanted to leave the option to power it directly if we wanted in the future to add the machine learning algorithms inside the camera. Also doing this change helped us remove the powerbank that was dedicated to the router, helping the total process of turning tino on and reducing from 4 batteries to 3 batteries

(couldn't do more because the week had thursday and Friday as holiday)

May 6 This week started on upgrading the kinamtic base. given that tino had an omniwheel base and tino is almost 20KG the wheels that it have where breaking apart and getting stuck, the rollers of the wheels where getting squared out. this also because given the movement, the back wheel of the triksta base was most of the time being dragged, as tino only moves forward and turn side to side

given this issues we decided to remove the omnidirectional triksta base as this movement is not needed, we decided a simple but reliable differential drive system, using 2 wheels at the front and a caster wheel in the back. To start this process we decided to just modify the base instead of changing it, given that it already had most of the things we needed.

We removed the 3 motors, replaced them with 2 more powerful motors, given this new motors whe changed the old 2 motor drivers with a new and more powerful mdd10a.

We built the T structure using Aluminium profiles, item, this allowed us to have a dynamic and regulable system where we can extend out the wheels to try and get a proper balance. One of our main issues where the wheels we started by using plastic wheels that had a rubber neumatic, this worked first but then when tino was built it created an issue

May 8 Completed major refactor of serial communication and motor control systems for the new differential drive base implementation. Created new_base_tino.ino with enhanced PID controller specifically designed for the differential drive system, replacing the old omnidirectional control logic.

Updated hardware_interface_node.py with improved serial port configurations, added comprehensive debugging logs, and enhanced command handling to work with the new base architecture. The new system uses proper device symlinks (/dev/ttyBASE, /dev/ttyHEAD, /dev/ttyLEG) to ensure consistent Arduino connections.

Created upload scripts (upload_new_base.sh, setup_arduino_symlinks.sh) for easier development and deployment workflow.

Once the new base was rebuilt I had to modify the code for it to work, the original base used the VirHas library (custom internal library of the airlab to manage and control the triksta bases) so given this used a differential drive I had to implement my own PID movement controller (Proportional–integral–derivative controller) but keeping the commands the same in order to keep the original tino movement

Once the base was ready I started rebuilding tino, removing things that where not needed and adding the new things and new cable harness. I also added the speakers in the servo head because it had enough space and the microphone was passed through the fabric on the head

Then, once built, I had to test the connection from the arduinos to the Jetson, this proved to have some issues, the head Arduiono was a az-delivery arduino mega clone, but the jetson does not had the needed drivers (CH340) the only solution was to rebuilt the kernel of the jetpack system so to include it, given that this would be time consuming I decided to change that arduino mega with a Arduino elego uno r3. this one properly linked and connected to the jetson, the change did not create any issue as the head only used 3 PWM pins for the servos. I also setup a symlink using the serial of the devices so that when connected they always be in the same route /dev/ttyHEAD /dev/ttyBASE

/dev/ttyLEG

once all systems where working again and some fine tunning had to be done to the gamepad (we changed from D input to X input because the jetson did not had the drivers to manage the D input) tino was working once again. The next step was going to test the wheels, the wheels we had put, given the weight of the robot the tire Partially de-beaded. consulting this issue we had 3 approaches to take next week:

1. Fill the current wheels with hotglue, easy but could cause issues with the traction of the tire
2. use some hard plastic wheels but is demanding in labor because this wheels do not have the 6mm axis needed to connect to the motor axis, so we will need to modify them a lot in order to connect properly
3. buy a new pair of wheels that can support this weight better

We also encountered some issues with the fabric enrolling over the wheels so we may need to add a type of "bumper" in order to avoid this

Also we need to find a way to make the camera avoid the fabric, or better said, the fabric to avoid moving over the camera FOV I tried sticking the fabric to a foam external shell I put over the camera but this velcro was not sticking to the fabric given the camera is behind the leg, and this side of the robot moves the fabric a lot. Also using this foam to make the shell was not ideal because it was absorbing the camera heat and not letting the camera cooldown.

this are the issues to solve by next week

May 13 This week started by trying to solve the wheel issue, we tried to fill the wheels with hot glue, this worked perfectly, the wheels did not de-bead and the traction was good. Also create a "bumper" to avoid the fabric to get stuck in the wheels, this works on most of the scenarios but there are still some cases that it may get stuck, so we will need to keep testing it.

I didnt have time to create the shell for the camera

May 15 Implemented comprehensive VR data recording functionality for Unity integration. Created vr_data_recorder_node.py and vr_data_extractor.py to capture and process all robot sensor data, human pose detection, audio streams, and robot state information for VR system development and testing.

The recording system includes service controls for starting/stopping data capture, meta-

data management, and proper data synchronization across all robot systems. This allows for detailed analysis of robot behavior and human-robot interactions for VR system optimization.

Enhanced VR interface message structure documentation and improved serial port configurations for better reliability during VR data exchange.

May 20 before continuing with the camera shell we had a new issue. The current arms for the head where breaking, this was because the head is too heavy and has very agressive moves, also given the 3-Dof Stewart platform it has a lot of flex in the arms, so we had to change the arms to a more robust design, Our first hypothesis was that the current desing the servo axis was not aligned with the head axis, this could cause the servos to get damaged as the force was being applied to the servo axis and not to the head axis, so we designed a new arm that has the servo axis aligned with the head axis, this way the force is applied directly to the head and not to the servo axis. This new model was designed in Inventor and printed in PLA. The idea of this pice is that it can hold the arm that goes to the top from the middle and have it properly tight, also aligned the servo axis with this arm axis and then the head axis. The pieces where printed and fixed to the head, this worked but we still saw some flex in the arms, that we think is acceptable so we will keep testing it.

May 27 This week I started by trying to create the camera shell, this was done by creating a shell that can be attached to the trypod system that was used to hold the camera on tino, this shell needed to have some important features It needed to of course hold the camera, but it had to had a empty space in the back to allow the camera to cool down, also it needed to have a way to attach it to the trypod system, and finally it needed to have 2 flaps at the top and at the bottom so that we can use them as a point to glue some velcro to hold the fabric in place and avoid it to get in the camera FOV

Jun 3 This week we finalized the camera shell, we attached the velcro to the flaps and the other side of velcro to the fabric by sewing it. Also we glued a mesh so that we can hide the camera and avoid it to be seen. We tested the camera and it worked perfectly, the fabric did not get in the camera FOV and the camera was able to cool down properly, also the camera was able to see through the fabric so it was able to detect humans.

Also this week the head arms we printed broke, we thought it was because of the 3d printed layer orientation and force applied, so we redesigned the arms to have a more robust design, this time we printed it in the other orientation so that the layers orientation is perpendicular to the force applied, this way we hope that the arms will not break again.

June 24–July 1 This week we got the new brackets for holding the new more powerful motors, this change was done because the old motors suffered a lot of heat due to tino weight. This new ones can support more weight and have a better heat dissipation the only issue they had is that the old motor bracket was not compatible so we had to wait for the new ones to arrive. this new bracket proved difficult to implement as the holes they had where not aligned with the item profiles, also the motors axis was more up so tino was dragging on the floor To fix both of this issues I created a spacer with some metal square profiles, this way the motors where aligned with the item profiles and also the axis was at the right height so that tino could move freely.

Also with this new motors I had to redo some cable connections to the encoders, power and driver simplifying the desing and connections

We tested the new motors and they worked perfectly, there was no dragging and the wheels did not de-bead, also the traction was good and the speed was good enough for tino to move around. We only had to tweak a bit the PID values to make it more stable and not overshoot the target position

Enhanced Raspberry communication system with special command processing and periodic status updates in the main loop for improved reliability and debugging capabilities. This allows for better monitoring of Arduino systems and more robust error handling.

Implemented audio messages system with chime notifications, providing auditory feedback for system events and user interactions. This enhances the user experience and provides clear audio cues for various robot states and actions.

July 1 After a period of use the arms for the head broke again, There is still a lot of flex in the arms, so we decided to redesign the arms again, this time we decided to change the approach. Doing some research on Stewart platforms we found that the arms should have rod end (heim joints) on both ends, this way the arms can move freely and not have any flex. The current design was using a bearing on the servo side and a rod end on the head side, this was not ideal as the bearing was not allowing the arm to move freely, so we redesigned the arms to have rod ends on both sides, this way the arms can move freely and not have any flex. The new design was printed in PLA and combined with some metal heim joints, this way the arms can move freely and not have any flex. The only issue this created was that when stationary the head had some wobble, this simple because the head is held by the 3 arms that can move thanks to the rod end. We dont think this is a major issue as it may add a bit to the “expresiiveness” of tino head movement.

July 8 Based on the VR system I had to modify the current system of movement to better

integrate with the VR environment. The old system using the gamepad was broadcasting a messgage of movement to the 3 arduinos, this was so that the head, leg, and base could move synchronously But now in the VR system we need to "independicese" this 3 system.

The head is controlled bu the VR movement, this already works given the head comand topics, we only had to remove the defined "routines" that it had when moving forward and to the sides. The leg new needs to be controled with 4 different states. Previously the leg was just resting and when a movmeent command was sent it did a whole continus rotation based on the sinusoidal system that was created originally. Now because this new system the idea is that the user in the VR need to "drag" the same way tino "drags" the leg in the real world, so for this we created 4 states:

0. Resting
1. little push (this was done so that tino can express a "pointin" to something or "looking" at something to try and get the attention of the user)
2. leg forward (moves the leg to the max reach before doing the dragging movement)
3. leg backward (moves the leg back doing the "dragging" movement up to the resting position)

We also had to modify the base movement, this was done by removing the old gamepad movement and implementing this new 4 state system:

0. Resting
1. little push (the base moves forward and backward a bit very fast to express a "pointing")
2. Does nothing, because at this point the leg is just being moved forward to prepare for the dragging
3. move forward (this is the dragging movement, the base moves forward a determided amount of time to simulate the dragging movement)

In order to make all of this systems work properly we had to made them atomic, and also this new states will be sent as pulses instead of continuous movement This way the user can control tino in a more natural way, and also the user can express more emotions and actions with tino

While doing the leg changes we noticed something intresting, the PWM that the driver gets for negative values for some reason is not the usual range from 0 to 255 the motor that moves the leg goes forward from very slow to very fast (as intended) but as one would

expect the negative values to go backward, it does not, it goes slow at -235 and fast at -1 This is not an issue as we can create the respective system to make it work with this values, but it is something to keep in mind for the future

July 10–13 Enhanced VR interface with improved odometry checking and loss detection, simplifying the condition logic for more reliable operation. Updated motor speed and angular parameters for improved performance and smoother robot movement.

Added comprehensive logging improvements across pose detection and VR interface nodes, reducing log verbosity while maintaining essential debugging information. Modified skeleton publish rate in robot controller for optimal performance balance between real-time tracking and system resources.

Improved ROS-TCP-Endpoint submodule with enhanced error handling for more robust Unity-ROS2 communication bridge.

July 15 This week we prepared the experiment but we also had to finalize the atomic movement system and implement the complete 4-state management for both leg and base controllers.

The major achievement was completing the synchronization between the leg and base atomic movements. We implemented a sophisticated locking system where the base controller can only execute case 3 (forward movement) after the leg controller has completed case 2 (leg extension). This ensures that Tino's movements are perfectly coordinated - the leg extends first to prepare for dragging, then the base moves forward to simulate the actual dragging motion.

We also implemented a pending command system that was crucial for the VR integration. When the user in VR triggers a movement command while case 2 is still running (leg extending), the system stores the command and automatically executes it once case 2 completes. This allows for natural user interaction without losing commands or creating timing conflicts.

The rotation system was expanded to work atomically as well. We implemented cases (3,1) and (3,-1) for right and left rotation respectively, maintaining the same 1.7-second duration as forward movement for consistency. This means the VR user can trigger rotation movements that are perfectly timed and atomic just like the forward movements.

One of the most important improvements was the pulse system we implemented in the gamepad node. Instead of continuous signals, each button press now generates a 3-cycle command pulse that automatically returns to idle. This pulse approach is essential for the VR system because it ensures each user action in VR corresponds to exactly one complete

movement in the physical robot.

We also enhanced the debug system significantly, adding timing logs and state tracking that shows exactly when each phase starts and completes. This was crucial for fine-tuning the 1.5-second case 2 timing cycle and the 1.7-second case 3 movements to ensure perfect synchronization.

The atomic movement system now guarantees that once any movement starts, it must complete fully before another can begin. This prevents partial movements and ensures the VR user always sees Tino complete the action they initiated, creating a much more natural and predictable interaction experience.

As a cleanup task, we also removed the old left joystick analog control since everything now runs through the discrete 4-state button system, making the control scheme consistent with the VR approach.

Added a system to offset the map positionbyt a desired amount of degree, this becauseits difficult to control the alignment of the SLAM map when creating it and some times it can be created with a rotation, this in principle is no problem, but given that we need to map this position directly into unity is best if we have the map aligned with the cartesian plane. This value can be changed but the process needss to be trial and error until we find a desired angle, this may change from map to map

After quite the amount of setup for mapping, we had to add a lot of landmarks to help the system not get lost when close to a wall, even tho the system work most of the time there are scenarios where it drifts by at maximum 1.20m we did some test setting 4 positions in the floor and then driving tino around those multiple times, this is the findings

button it1 position: x: -9.07916381186811 y: -3.0198425113288256 z: 0.21915075182914734 orientation: x: -0.0235615282947662 y: 0.0371791281745811 z: 0.9987406916737919 w: -0.024071783643750885 it2 position: x: -8.186847680416442 y: -3.339181554181038 z: 0.22911125421524048 orientation: x: -0.03578283275599135 y: -0.014825521998614472 z: 0.9886076124808023 w: -0.1454468531069043

i3 after spinning position: x: -9.132201829319898 y: -3.03850808312142 z: 0.16989609599113464 orientation: x: -0.023309146346151374 y: 0.013430598998060258 z: 0.9970844577607847 w: -0.07140670417222031

door it1 position: x: -6.302746616280681 y: -2.937899155623209 z: 0.04246218502521515 orientation: x: -0.041125382354844545 y: 0.04075343561185359 z: 0.9381911838945137 w: -0.3412406188158956 it2 position: x: -7.688149884473027 y: -2.758148921953637 z:

0.26177895069122314 orientation: x: -0.056920978093906345 y: 0.06027535966552371 z: 0.9105215513414311 w: -0.4050646707808027

platform it1 position: x: -5.133887212339737 y: -0.8823759434488573 z: 0.003330887295305729 orientation: x: 0.037969363959043455 y: 0.042694832314988815 z: -0.7221974378925629 w: 0.6893232444590183 it2 position: x: -4.155619022416493 y: -1.1973306038245426 z: 0.18429601192474365 orientation: x: -0.0423178727432749 y: -0.154526999071549 z: 0.8307654381749795 w: -0.5330660364450295

middle bridge it1 position: x: -2.967067619346332 y: -1.0099800292205212 z: 0.020707421004772186 orientation: x: -0.011650759024720693 y: 0.020886688418633118 z: 0.9882899125201247 w: -0.1507018578293635

it2 position: x: -3.0019163829042084 y: -0.9073216003302291 z: 0.1694898009300232 orientation: x: -0.0432093148775227 y: 0.026300433857604333 z: 0.9952330147461537 w: -0.08338188177412909

base on all of this we found the only way to make the system "undrift" or to properly relocalize is to make tino spin, this helps rtab to get more features around and try to correct the position. even tho this is a solution for other scenarios, because we need to create a system that will be manage via the vr by a user with no technical expertice and also because part of the experiemt is let the user figure out stuff we cant tell the user to just "spin around" until it finds the proper relocalizatio also, there is no progragmatic way to find this issue of the drift in the first place, given that this position given is absolute we cant know automatically that we are drifting to try and correct

At least the orientation worked really good that even if on the positionion is lost we still had good orientation so we decided the following we will implement a different system for localization, we will use UWB to mange the global positionion and then we will use the already in place camera and RTAB slam to get the orientation By doing this fusion we hope to get a stable and reliable position adn orientation

Now with the new UWB system implemented we have a better and precise global positioning We did tests on the same 4 spots 3 times each and this was the output

Bridge it1 position: x: 2.95 y: 1.28 z: 0.96 orientation: x: -0.016244746081379848 y: 0.011590778809634706 z: 0.9978177685970177 w: -0.06294018257794398 it2 position: x: 2.92 y: 1.24 z: 1.04 orientation: x: 0.03365805896384339 y: -0.10581931225218563 z: 0.9615574311256412 w: -0.25115088467150715 it3 position: x: 2.96 y: 1.22 z: 0.98 orientation: x: 0.06318550106034526 y: 0.018786558120585363 z: 0.965924510222013 w: -0.2502889073188361

platform it1 position: x: 5.49 y: 1.26 z: 0.98 orientation: x: 0.05060387003762118 y: 0.04387060236978493 z: -0.7095551199862337 w: 0.7014599325016385 it2 position: x: 5.42 y: 1.45 z: 0.97 orientation: x: -0.044399141185597304 y: 0.00893117279072231 z: -0.7339342771249334 w: 0.6777089090442352 it3 position: x: 5.45 y: 1.21 z: 1.02 orientation: x: 0.0001595946587705419 y: 0.02375217746987014 z: -0.8009830471763693 w: 0.598215889888112

door it1 position: x: 6.45 y: 2.72 z: 0.97 orientation: x: -0.02131269555403667 y: 0.05062973708550628 z: 0.9471005285691299 w: -0.3162009876538764 it2 position: x: 6.42 y: 2.67 z: 0.96 orientation: x: -0.031268703585232954 y: 0.03502103116732864 z: 0.9344169219027985 w: -0.3530734466584695 it3 position: x: 6.43 y: 2.69 z: 0.95 orientation: x: -0.026643789303690477 y: 0.03844515148744251 z: 0.9429357383344411 w: -0.32967307363487713

button it1 position: x: 7.96 y: 2.72 z: 1.01 orientation: x: -0.0060182617982888105 y: 0.04144030272178065 z: 0.9977658808453272 w: -0.0520551769152283 it2 position: x: 7.89 y: 2.72 z: 1.01 orientation: x: -0.004659043734153017 y: 0.040590124652731686 z: 0.9970509609184363 w: -0.06496374337391156 it3 position: x: 7.85 y: 2.66 z: 0.98 orientation: x: -0.005168897035844065 y: 0.021907294998073198 z: 0.9981057371155894 w: -0.05725740236429487

also doing the testing we lost a wheel, the plastic wheel hub broke. We fixed it by hotglueing the wheel hub again but thats something may need to be changed in the future

Bibliography

[1] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12(6): 869–880, 1996. doi: 10.1109/70.544770.

[2] C. Breazeal. Toward sociable robots. *Robotics and Autonomous Systems*, 42(3):167–175, 2003. ISSN 0921-8890. doi: https://doi.org/10.1016/S0921-8890(02)00373-1. URL `https://www.sciencedirect.com/science/article/pii/S0921889002003731`. Socially Interactive Robots.

[3] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. doi: 10.1109/TRO.2021.3075644.

[4] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2019. URL `https://arxiv.org/abs/1812.08008`.

[5] Cardillo. Conveying illusion of life and empathy through a non-anthropomorphic robot and non-verbal communication: The tino experience. Master's thesis, Politecnico di Milano, 2024. Previous work on the original Tino robot system.

[6] Z. Chang, F. Zhang, J. Xiong, X. Xue, Z. Wang, B. Jouaber, and D. Zhang. Uwbori: Enabling accurate orientation estimation with ultra-wideband signals. In *2024 IEEE Smart World Congress (SWC)*, pages 386–393, 2024. doi: 10.1109/SWC62898.2024.00085.

[7] Y.-Y. Chen, S.-P. Huang, T.-W. Wu, W.-T. Tsai, C.-Y. Liou, and S.-G. Mao. Uwb system for indoor positioning and tracking with arbitrary target orientation, optimal anchor location, and adaptive nlos mitigation. *IEEE Transactions on Vehicular Technology*, 69(9):9304–9314, 2020. doi: 10.1109/TVT.2020.2972578.

[8] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i.

IEEE Robotics & Automation Magazine, 13(2):99–110, 2006. doi: 10.1109/MRA.2006.1638022.

[9] P. Ekman, W. Friesen, M. O'Sullivan, A. Chan, I. Diacoyanni-Tarlatzis, K. Heider, R. Krause, W. LeCompte, T. Pitcairn, P. Ricci Bitti, K. Scherer, M. Tomita, and A. Tzavaras. Universals and cultural differences in the judgments of facial expressions of emotion. *Journal of personality and social psychology*, 53:712–7, 10 1987. doi: 10.1037/0022-3514.53.4.712.

[10] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry, 2016. URL `https://arxiv.org/abs/1607.02565`.

[11] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 2014. doi: 10.1109/ICRA.2014.6906584.

[12] P. Galajda, A. Galajdova, S. Slovak, M. Pecovsky, M. Drutarovský, M. Sukop, and I. Samaneh. Robot vision ultra-wideband wireless sensor in non-cooperative industrial environments. *International Journal of Advanced Robotic Systems*, 15:172988141879576, 09 2018. doi: 10.1177/1729881418795767.

[13] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, 2011. doi: 10.1109/IVS.2011.5940405.

[14] S. Gezici, Z. Tian, G. Giannakis, H. Kobayashi, A. Molisch, H. Poor, and Z. Sahinoglu. Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *IEEE Signal Processing Magazine*, 22(4):70–84, 2005. doi: 10.1109/MSP.2005.1458289.

[15] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[16] A. A. Housein, G. Xingyu, W. Li, and Y. Huang. Extended kalman filter sensor fusion in practice for mobile robot localization. *International Journal of Advanced Computer Science and Applications*, 13(2), 2022. doi: 10.14569/IJACSA.2022.0130204. URL `http://dx.doi.org/10.14569/IJACSA.2022.0130204`.

[17] S. Khunteta, P. Saikrishna, A. Agrawal, A. Kumar, and A. K. R. Chavva. Rf-sensing: A new way to observe surroundings. *IEEE Access*, 10:129653–129665, 2022. doi: 10.1109/ACCESS.2022.3228639.

[18] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007. doi: 10.1109/ISMAR.2007.4538852.

[19] S. Krishnan, P. Sharma, Z. Guoping, and O. H. Woon. A uwb based localization system for indoor robot navigation. In *2007 IEEE International Conference on Ultra-Wideband*, pages 77–82, 2007. doi: 10.1109/ICUWB.2007.4380919.

[20] M. Labbé and F. Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019. doi: https://doi.org/10.1002/rob.21831. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831`.

[21] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann. Mediapipe: A framework for building perception pipelines, 2019. URL `https://arxiv.org/abs/1906.08172`.

[22] A. Mehrabian. *Silent Messages*. Wadsworth Publishing Company, 1971. ISBN 9780534000592. URL `https://books.google.com.co/books?id=Ast-AAAAMAAJ`.

[23] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. doi: 10.1109/TRO.2015.2463671.

[24] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407, 2011. doi: 10.1109/ICRA.2011.5979561.

[25] OptiTrack. Optitrack for robotics – industrial use cases, 2024. URL `https://www.naturalpoint.com/optitrack/applications/robotics/`.

[26] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, 2020. URL `https://arxiv.org/abs/1907.01341`.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection, 2016. URL `https://arxiv.org/abs/1506.02640`.

[28] L. Romeo, R. Marani, M. Malosio, A. G. Perri, and T. D'Orazio. Performance analysis of body tracking with the microsoft azure kinect. In *2021 29th Mediterranean Conference on Control and Automation (MED)*, pages 572–577, 2021. doi: 10.1109/MED51440.2021.9480177.

[29] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics & Automation Magazine*, 18(4):80–92, 2011. doi: 10.1109/MRA.2011.943233.

[30] N. Shahid, G.-H. Yu, D. Trinh, D.-S. Sin, and J.-Y. Kim. Real-time implementation of human detection in thermal imagery based on cnn. *The Journal of Korean Institute of Information Technology*, 17:107–121, 01 2019. doi: 10.14801/jkiit.2019.17.1.107.

[31] T. Sheridan. Musings on telepresence and virtual presence. *Presence*, 1:120–125, 01 1992. doi: 10.1162/pres.1992.1.1.120.

[32] W. Tai, B. Ilias, S. Abdul Shukor, N. Abdul Rahim, and M. Markom. A study of ultrasonic sensor capability in human following robot system. *IOP Conference Series: Materials Science and Engineering*, 705:012045, 12 2019. doi: 10.1088/1757-899X/705/1/012045.

[33] E.-J. Theussl, D. Ninevski, and P. O'Leary. Measurement of relative position and orientation using uwb. In *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6, 2019. doi: 10.1109/I2MTC.2019.8827149.

[34] UZ-SLAMLab. Orb-slam3 github repository, 2024. URL `https://github.com/UZ-SLAMLab/ORB_SLAM3`.

[35] UZH-RPG. Svo pro open github repository, 2024. URL `https://github.com/uzh-rpg/rpg_svo_pro_open`.

[36] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022. URL `https://arxiv.org/abs/2207.02696`.

[37] Z. Yan et al. Lidar-based human detection. *Autonomous Robots*, 2019. URL `https://link.springer.com/article/10.1007/s10514-019-09883-y`.

List of Figures

List of Tables

Acknowledgements

Here you may want to acknowledge someone.