POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Tesi di Laurea Magistrale in
Computer Science and Engineering
Ingegneria Informatica

Title of the thesis

Author:
Name Surname

Student ID:
XXXXXX

Advisor:
Prof. Name Surname

Academic Year:
20xx-xx

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

Pre Tino & goal

SLAM, process and R&D

General sensor stuff speaker, mic, VR

Legacy to ROS2 migration & ROS arquitecture

Physical changes

Testing

1 | 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	ORB-SLAM3 (Campos et al., 2021) – Robust monocular/Stereo SLAM. SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems
IMU + Wheel Encoders	Low cost; integrates motion data	Drift over time; Stewart tilt issues	Sensor Fusion for Mobile Robot Localization (Huang et al., 2018) – Kalman filtering. EKF Sensor Fusion in Practice for Mobile Robot Localization
UWB-IR	Small footprint; could work with fabric	Requires external anchors	A UWB-IR based Localization System for Indoor Robot Navigation.

External Sensing

Technology	Pros	Cons	Key Papers & Resources
UWB Anchors	High accuracy; no line-of-sight	Setup/calibration required	Robot vision ultra-wideband wireless sensor in non-cooperative industrial environments
AprilTags	Low cost; precise	Line-of-sight; limited area	AprilTag: A Robust Fiducial System (Olson, 2011).
MoCap Systems	Sub-mm accuracy	Expensive; fixed environment	OptiTrack for Robotics – Industrial use cases.

Orientation Technologies

- **Sensor Fusion**: A Review of Sensor Fusion Techniques filters for combining UWB, IMU, and encoders. (Waiting for access request)
- **UWBOri**: enabling accurate orientation estimation with ultra-wideband signals
- **NLOS Mitigation**: UWB System for Indoor Positioning and Tracking With Arbitrary Target Orientation, Optimal Anchor Location, and Adaptive NLOS Mitigation
- **RPO**: Measurement of Relative Position and Orientation using UWB

Human Detection Technologies

Onboard Sensing

Technology	Pros	Cons	Key Papers & Resources
Thermal Cameras	Works in darkness; fabric-friendly?	No depth; limited range	Thermal Human Detection – CNN-based approaches.
Ultrasonic Array	Low cost; proximity detection	No human distinction	Ultrasonic Human Tracking (In Korean).
Upgraded Camera	Wider FOV; ML-compatible	Fabric obstruction; compute-heavy	YOLOv7 (Wang et al., 2022) – Real-time object detection.

External Sensing

Technology	Pros	Cons	Key Papers & Resources
RGB-D Cameras	Depth data; multi-human tracking	Fixed installation	OpenPose: Real-Time Human Pose (Cao et al., 2019). Azure Kinect for Robotics – Performance Analysis of Body Tracking.
LiDAR	High-resolution 3D mapping	Expensive; compute-heavy	LiDAR-based Human detection (Zhi Yan et al., 2019).
WiFi/Radar	Privacy-friendly; fabric-penetrating	Lower resolution	RF-Sensing: A New Way to Observe Surroundings

Technologies for Tino Robot Implementation

Localization Technologies

Visual Odometry (VO)

- **Variants:**
  - *ORB-SLAM3* (supports RGB-D): GitHub
    - \+ Synergy with human detection via depth data
    - \+ Robust feature matching for dynamic environments
    - – Higher computational cost (requires GPU optimization)
  - *SVO* (Semi-direct Visual Odometry):GitHub
    - \+ 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, metadata 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 pe-

riodic 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.

List of Figures

List of Tables

Acknowledgements

Here you may want to acknowledge someone.