Shared Human-Robot Control for Wheeled Soccer Robots

Tobias Wejbora

Student Number: [redacted]
RBTBEP25003

Supervisors:
René van de Molengraft
Danny Hameeteman
Ruben Beumer

Final version

Eindhoven, 30th June 2025

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mechanical Engineering
Robotics Group

Contents

1 Introduction

In the field of robotics, human-robot interaction (HRI) is playing an increasingly more important role. In these fields of study, the aim is to create a basis for communication and understanding between robots and humans so that their individual strengths are optimized. Although fully autonomous robots are highly precise and accurate in performing repetitive tasks, humans provide essential strategic knowledge, adaptability to deal with new situations, and intuitive decision-making abilities needed to deal with dynamic and complex situations, which are, in particular, readily observed in robot soccer. Currently, the control method for the Tech United soccer robots ("TURTLEs: Tech United Robocup Limited Edition") is binary because control is either fully autonomous or fully manual in operation. In the current state, pure manual control is too challenging for operators to engage in meaningful strategic play; mainly due to low-level challenges of precise control. On the other hand, pure autonomous play, while adaptive and capable of basic strategies, does not achieve the advanced and nuanced creativity that human operators can provide.

The primary challenge this framework addresses is the "level-of-assistance dilemma". To gather meaningful data on human play, the system must be fluid and usable, which requires some level of autonomous assistance. However, if the assistance is too aggressive, it risks overriding the operator's intent and limiting the very behaviors that are important to observe. The "human play" is lost, and the system ceases to be a useful tool for research. Therefore, the purpose of this work is not to achieve maximum automation, but to create an efficient shared control environment. This framework is therefore designed to:

- **Enable Research on Human Strategy:** Provide a platform where researchers can gather data and measurements of how humans interact and play soccer with robots (HRI), capturing their strategic choices, timing, and adaptability. Future questions to be asked are the following. How do human operators strategically interact with robot teammates during the game of play? How can these strategies be interpreted and integrated by robots?
- **Balancing the Level of Assistance:** Implement assistive features (like those for passing and shooting) not as a means to perfect every action, but as tools to lower the barrier to effective play, allowing the human's strategic intent to be the primary driver. The level of assistance is a critical parameter that must be tuned, ensuring that the human remains "in the loop" in a meaningful way.
- **Serve as an Interactive Platform:** As a practical application, this framework also functions as an effective tool for public demonstration, creating engaging robot interactions with the public and hands-on experiences.

This paper details and implements a shared human-robot control (SHRC) framework. Several key features, such as assisted passing, assisted shooting, and player switching, are discussed, taking into account the balance of autonomous assistance and human contribution.

2 Background Research

Past developments in shared human-robot control have primarily been within healthcare or automotive industry where surgical robots or autonomous vehicles are partially controlled by a human. For example, surgical robots like the *da Vinci surgical system*, convert human hand movements into accurate and precise robot joint movements using image enhancement and automated tasks for invasive surgery [4]. In the automotive industry, shared-control is utilized for driver assistance, such as lane assist and adaptive cruise control, where automation continuously supports the driver at the control level with an adaptive authority [7]. Furthermore, in assistive and rehabilitation robotics, shared control frameworks have been implemented in smart wheelchairs to analyses performance and autonomous preference [2]. Within these different use cases of shared control, a key focus area is the nature of communication between the human and the robot.

> "This communication is bi-directional – the agent needs to understand the person (to infer their goals and intentions), and the human needs to understand what decisions the agent is making and why."[8].

In soccer robotics, shared control has generally remained unexplored, but it presents an opportunity to learn by demonstration. Traditionally, robots require programming skills to implement and test new strategies or plays. However, with shared-control, individuals with knowledge in soccer but no technical background can directly interact with these robots [10]. Shared control combines the strengths of both humans and robots; humans are excellent at reasoning, (re)planning, and adapting to unstructured environments, while robots are very good in performing repetitive and precise tasks [9]. This opens the possibility of capturing improved strategies through demonstration that can be further researched and implemented in soccer robotics.

3 Methodology

3.1 Overall System Methodology

The proposed system is designed as a decoupled framework to enable integration of human input with certain levels of assistance in conjunction with existing autonomous capabilities of the Tech United robots. The core principle is to use the Real-Time Database (RTDB) as the central communication channel, allowing shared control logic to operate independently from existing autonomous systems.The RTDB, designed by the CAMBADA team [3], allows for a shared memory-based database, enabling real-time data exchange between the modules controlling the robots. The shared control framework utilizing the RTDB is shown in fig. 1 is made of four primary logical components.

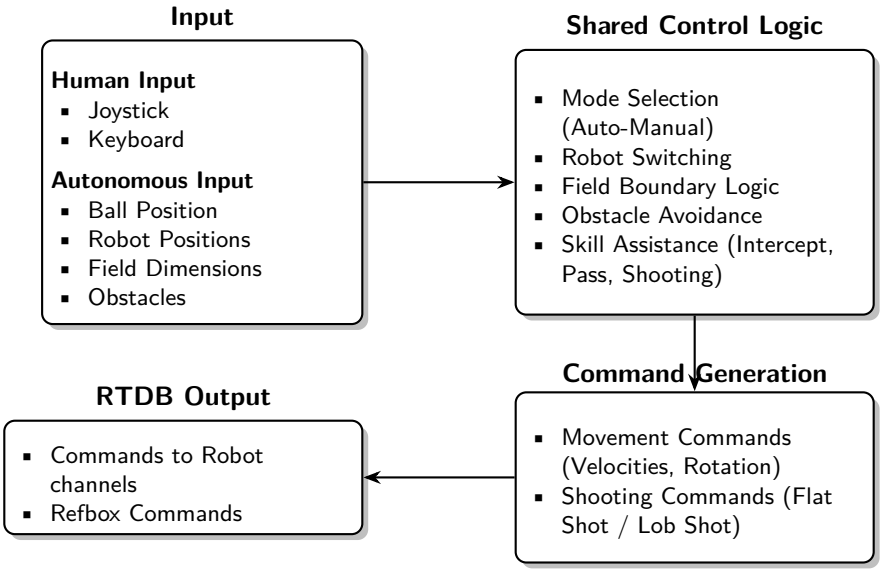

Figure 1: Block diagram of the shared human-robot control system integrating human input, autonomous data, control logic, and RTDB communication.

1. **Input Module:** This module gathers the operator's high-level strategic intent. In interfaces with input devices like joystick controllers and keyboards for multi-modal human interaction, translates raw commands(e.g., button presses, joystick movements) into executing basic skills or translational movements. Below, the keyboard controls are outlined showing how a human operator interacts with the shared control logic. Furthermore, in fig. 2, the joystick control configuration can be seen. The layout has been designed to mimic the FIFA[1] joystick control layout taking advantage of extensive user testing and experience already done by the FIFA developers. This choice within the shared control framework rewards those already familiar with FIFA controls and reduces the learning curve of new operators.
    - **W, A, S, D:** Translationally moves the manually controlled robot forward, left, backward, and right.
    - **O, P:** Rotate the active robot left (O) or right (P) at a fixed speed.

[1]FIFA (now EA Sports FC™) is a soccer video game series developed by EA Sports.

- **Space:** Initiate a lob shot (kick) with variable power based on how long the key is held.
- **F:** Initiate a flat pass/shot with variable power based on how long the key is held.
- **H:** Toggle between variable and calculated pass power modes.
- **N:** Toggle assisted passing mode on or off.
- **M:** Toggle assisted shooting (aiming) mode on or off.
- **T:** Toggle between automatic and manual robot selection modes.
- **Y:** Switch control between the magenta and cyan teams.
- **C:** Instantly switch control to the robot closest to the ball.
- **1–6:** Manually select a specific robot (by number) for direct control.

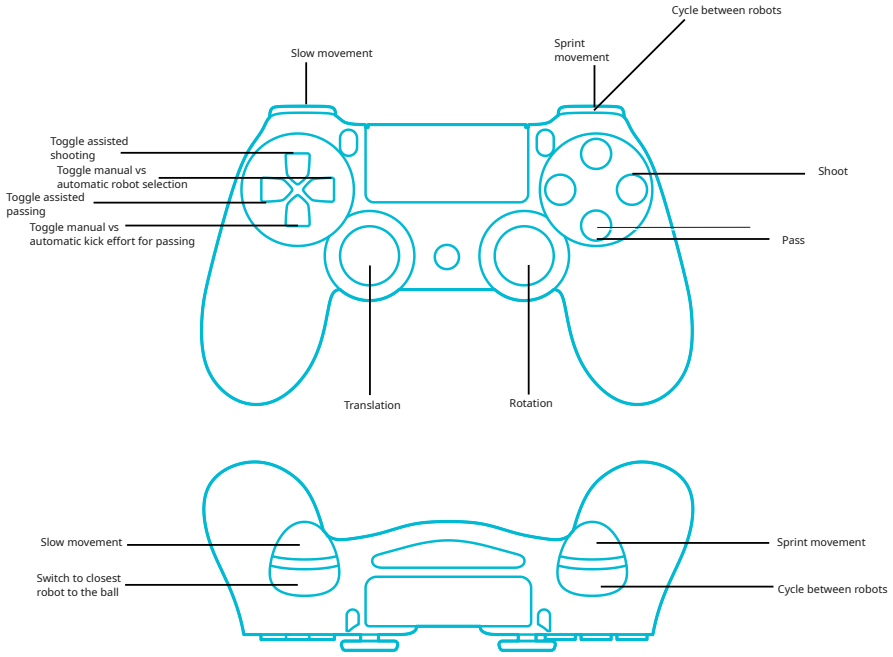

Figure 2: Implemented Joystick Controls

2. **Shared control logic(running on a central PC):** This is the core of the framework which runs software to:
   - Read state information from the WorldModel through the RTDB, which includes robot positions, ball position, and the game status. The RTDB allows the shared control logic to receive fused sensor data without direct coupling to the Vision or Motion modules.
   - Interprets raw joystick or keyboard inputs and commands from the Human Input Module.
   - Applies a set of control logic rules(assisted skills, ball intercept, field-boundary impedance) to determine the human's intent based on the game situation. This module thus determines optimal commands to fulfill the user's goal.

- Implements Proportional-Derivative (PD) control loops for robot rotation and assisted aiming, which convert angular position errors into velocity commands for the Motion module.
- Sends manual control commands through RTDB channels (COACH_FORx), bypassing existing autonomous strategy modules(DEFCON, Role Assigner, STP).
- Handles the automatic and manual switching logic of the robots where the automatic switching prioritizes ball possession and the manual switching allows for operator control of robot selection.

The shared logic has been programmed through a python file which makes use of pyRTDB, a python library to communicate with the RTDB. The communication flow of the python file running shared control logic can be seen in fig. 3, inspired by the TURTLE architecture presented in [1].

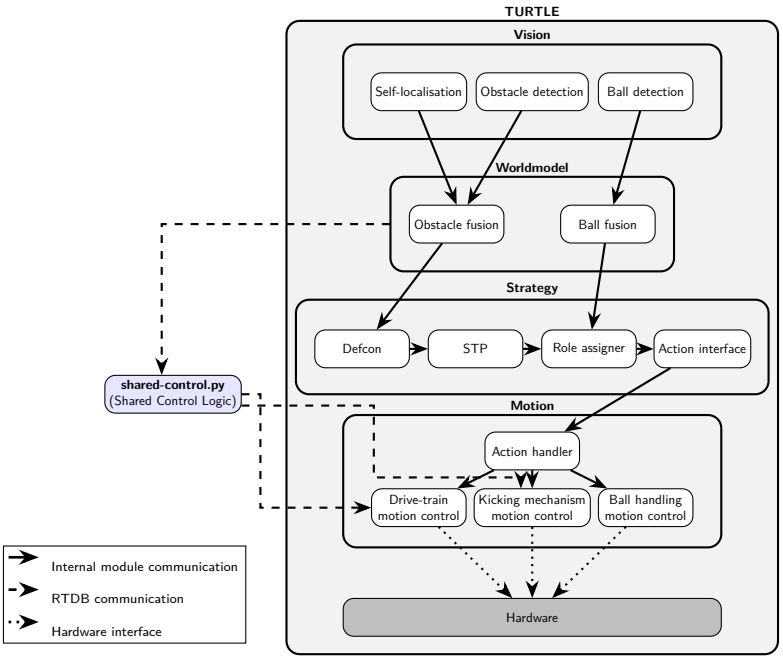

Figure 3: System architecture of the TURTLE robot including shared control logic and communication flows.

3. **Command Generation:** Decisions made by the shared control logic are converted into low-level commands such as target velocities or kick commands that the robots directly executes.
4. **Turtle (Onboard Robot Software)**
   - Reads commands from the RTDB on the robots respective channel.
   - Manages low level robot functions (motion,ball handling, kick commands)
   - Executes autonomous behaviors (e.g., obstacle avoidance, ball interception) which can be influenced or overridden by shared control commands.

- Sends robot sensor data (positions, status) and world model information (ball position) back to the RTDB.

5. **Robot(Hardware)**
    - The physical agents executing actions on the field.

The decoupled framework using the RTDB, opens possibilities as a modular framework for future HRI research. For example, incorporating different input devices or more advanced algorithms can be integrated into the shared control logic without adding changes to existing onboard software.

4 Shared-Control Logic

Effective shared control relies on integrating human strategy with robot precision for smooth and precise execution. This section details the key functions that bridge the gap between human intention and existing autonomous capabilities. These key functions range from safety constraints, such as field boundaries, to providing assistance for essential low-level tasks, such as passing and shooting. Each component aims to reduce the high cognitive workload that you would otherwise have when attempting to play successful soccer in pure manual mode, a challenge well-documented in human-robot collaboration literature [5]. Through these features, human operators can focus on high-level strategic thinking, rather than being distracted or overwhelmed by the low-level precision required for manual control.

4.1 Field Boundary Impedance

The first layer of assistance within the framework is the field boundary impedance. The purpose of this feature is to handle a non-strategic but critical task with regard to safety, keeping the robots within the field of play. The term impedance is used here because the system resists motion beyond the field boundaries, similar to how physical impedance resists velocity in mechanical systems. This safety-critical feature not only eliminates the low-level task of avoiding boundaries but it frees the operator from having to worry about safety, allowing them to focus on higher-level decisions such as positioning on the field, passing and shooting. This is one example of allowing fluid human play without interfering with strategic choices that might be important for research or learning by demonstration. In fig. 4, the Impedance logic is outlined. By monitoring the current position and velocity of a robot in a given moment, the speed of these robots can be proactively managed based on its distance to the field boundaries. The core mechanism calculates the required stopping distance based on a specified maximum deceleration rate.

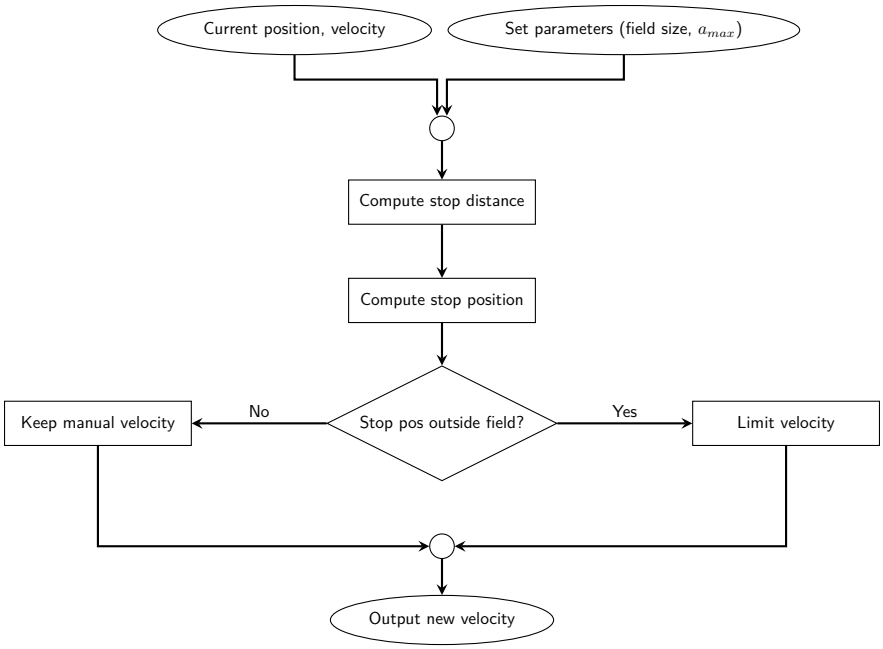

Figure 4: Flowchart for field boundary velocity impedance based on stopping distance and field limits.

4.1.1 Field Impedance Calculation

The impedance logic calculates the maximum allowable speed (v_{max}) at any point on the field to ensure that the robot can stop in time before crossing the boundary. The maximum allowable speed is calculated as a scalar limit which is then combined with the robots current direction of motion to create a vector constraint. The calculations follow these steps:

1. **Compute Stopping Distance:** The distance required for the robot to come to a complete stop based on its current velocity $\|\vec{v}\|$ and its maximum deceleration a_{max}, which is given by the kinematic equation $v_f^2 = v_i^2 + 2ad$. Here, v_f represents the robot's final velocity (which is zero, as the robot must stop), v_i is the initial velocity ($\|\vec{v}\|$), a is the constant deceleration ($-a_{\text{max}}$), and d is the stopping distance to be calculated.

$$d_{\text{stop}} = \frac{\|\vec{v}\|^2}{2a_{\text{max}}} \tag{1}$$

2. **Predict Future Position:** The future position $\vec{p}_{\text{future}}$ at where the robot would come to rest if it began decelerating immediately, can be written in terms of the direction of the unit velocity vector $\hat{v}$:

$$\vec{p}_{\text{future}} = \vec{p}_{\text{current}} + d_{\text{stop}} \cdot \hat{v} \tag{2}$$

3. **Check if Future Position is outside Boundary Limits:** If the predicted $\vec{p}_{\text{future}}$ lies outside the field boundaries, the velocity is limited.

4. **Calculate Distance to Approaching Boundary:** If the projected future position of the robot is outside the boundary limits, the system computes the maximum distance before reaching a boundary. Here, x_{boundary} and y_{boundary} are the nearest boundary based on the robot's path:

$$d_{\text{limit}} = \min\left(\frac{|x_{\text{boundary}} - p_{\text{current},x}|}{|\hat{v}_x|}, \frac{|y_{\text{boundary}} - p_{\text{current},y}|}{|\hat{v}_y|}\right) \tag{3}$$

5. **Determine Maximum Allowable Velocity:** Uses the kinematic formula again to calculate the allowable velocity:

$$v_{\text{limit}} = \sqrt{2a_{\text{max}}d_{\text{limit}}} \tag{4}$$

6. **Apply Velocity Limit:** The final velocity taken is the minimum of the velocity given by the human operator and the calculated maximum allowable velocity (v_{limit}):

$$\vec{v}_{\text{new}} = \hat{v} \cdot \min\left(\|\vec{v}\|, v_{\text{limit}}\right) \tag{5}$$

4.1.2 Limitations

This implementation still possesses a design trade-off, where absolute safety is prioritized over human strategic freedom and flexibility at the field boundaries. The current logic treats the field boundary as a hard constraint, and when near-by to a field line, a reactive force pushing away from the boundary can still be felt, possibly limiting maneuverability. Therefore, this implementation lacks context awareness to differentiate between a tactical, irresponsible, or unstable maneuver. For instance, an advanced strategy may be to have a robot run down the field outside of the field lines to overlap a defender. Another example is that, during a throw-in scenario, the robot has to slightly position itself outside of the field lines. However, the current framework does not support such edge cases, as it would require integrating more complex game-state awareness into the impedance logic. While this feature successfully allows for safe in-field play for research and demonstration, it does not consider more nuanced, infrequent strategies.

4.2 Automatic Ball Intercept

During manual operation of a turtle by a human user, it can become difficult to gain possession within close proximity to the ball. This is an example of a repetitive task that the human user has to deal with while participating in shared control. Due to the difficulty in gaining possession of the ball, assistance can be beneficial to the user. If the active turtle being controlled by the human user comes within a specified distance to the ball, it can be interpreted that the human user is inclined to intercept and gain control of the ball.

Intercepting the ball manually is a challenging low-level task that can frustrate operators and prevent them from engaging in higher-level strategic play. The framework accounts for this by including an automated ball intercept assistance, seen in fig. 5, based on the robots position relative to the ball. The system takes into account when a human-controlled robot is within a predefined radius $dist2intercept$ of the ball, at which point the manual movement commands are temporarily overridden to allow the existing autonomous commands to intercept the ball. The goal of this design choice is to eliminate the points of failure and frustration that lead to unnecessary inefficiencies in the game of play. Through this, the operator can immediately transition to focusing on strategically critical decisions like dribbling, passing, or shooting.

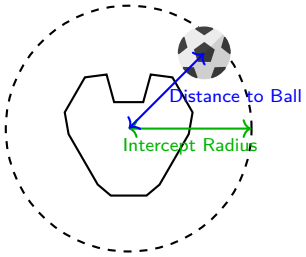

Figure 5: Ball intercept logic showing the intercept radius and current ball position relative to the turtle robot(not to scale).

If the ball comes within the intercept radius ($dist2intercept = 1000\,\mathrm{mm}$ was used during testing) of an active turtle, human input commands are no longer sent and replaced by autonomous RTDB commands. This is done through a simple command override:

```
    # Ball interception logic
if CPBrobot == 1:                                    # Turtle has
    posesson of ball
    refbox_cmd.controlData.command = JS_MOVE         # Allow human
        control
else:
    if dist2ball < dist2intercept:
        refbox_cmd.controlData.command = JS_NO_COMMAND  # Enable
            autonomous interception
    else:
        refbox_cmd.controlData.command = JS_MOVE
```

With regard to the implemented ball interception logic, it is largely reliant on fixed distance thresholds and direct movement to the current ball location; therefore, it does not consider the velocity of the ball. Although distance thresholding is effective in some scenarios, it inherently limits the effectiveness of a robot's interception during the more complex scenarios encountered in the game of play. A proposed

solution would be to incorporate the velocity of the ball with the current distance threshold logic. To move in this direction, an initial step would be to consider a more effective shape for distance thresholds. In fig. 6, an optimized ovular shape is shown for a distance threshold. The benefit of such a shape is that it reduces the amount of false intercepts of the ball when the robot is not facing the ball. Within the current implementation, it becomes more likely that a ball is automatically intercepted when simply driving past the ball.

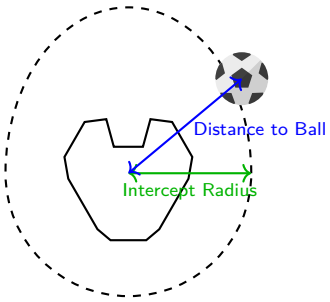

Figure 6: Optimized Ball-intercept distance threshold logic showing the intercept radius and current ball position relative to the turtle robot(not to scale).

A further future step is to take the velocity of the ball into consideration, where the trajectory of the ball determines the threshold at which the robot switches to automatically intercept the ball. This implementation can be as simple as dynamically enlarging the intercept radius when the ball is moving quickly towards the robot, or as advanced as calculating the optimal intercept point based on the future positions of the robots. Additionally, when the intended receiver of an assisted pass is known, the receiving robot can be informed to immediately attempt to intercept the ball as soon as the pass is initiated.

4.3 Design and Implementation of Assisted Low-level Skills

4.3.1 General Principles of Assistance

The philosophy behind all assisted skills is centered on providing just enough assistance to bridge the gap between the operator's high-level strategic intent and the low-level execution precision of the robot. The assistance provided is merely corrective, where parameters like aim and power are adjusted to compensate for the inaccuracies of a joystick interface. A critical design choice is the degree of autonomous correction, as too high of autonomous behavior masks the human's skill and decision making. This goes against the purpose of learning by demonstration as any collected data on human play-style gets rendered meaningless. In contrast, insufficient assistance would fail to lower the skill barrier, leading to unintended outcomes or insufficient game of play. Within the framework, it is important to create a balance between providing corrective assistance and ensuring that human strategic decisions, including mistakes, are reflected in the outcome. This principle can be seen in the "Off-Target" logic for passes and shots, as the system chooses not to assist very poorly aimed shots. This preserves the element of human error and avoids misjudging and correcting an attempted shot at the goal.

4.3.2 Assisted Passing

Interpreting Human Pass Intent: The system must first understand the intention of the human to pass and where its desired target is. This is achieved through the following.

- Pass Trigger: A dedicated joystick button which signals the intent to make a pass.
- Target Indication:
  - Directional Aim: The current direction of the analog joystick stick at the moment of the pass trigger indicated the general direction of the pass intended by the human user.
- Pass Power/Effort: The pass power can be inferred from the following:
  - The duration of the pass button press
  - Distance to recipient robot.

Autonomous Assistance for Passing: Once the pass intent has been registered by the system, the shared logic on the user's PC executes the assisted passing algorithm. The algorithm uses the current position of the ball, the robot executing the pass, and all other robot teammates to determine if a pass is considered "On, Near, or Off Target" based on distance thresholds. These thresholds are designed to be adjustable by the human operator to modify the difficulty level.

- Target refinement:
  - If the user provides a general direction to pass in, the system identifies the closest teammate (based on the angular deviation) in the passing direction.
- Execution Adjustment:
  - "On-target pass": If the human user's aim is within a small lateral distance from the recipient robot, no correction is applied to the aim, and the pass is executed.
  - "Near-target pass": If the human user's aim is within a larger but moderate lateral distance from the recipient robot, the system corrects the aim of the robot executing the pass until it aims within the "On-target" threshold of the recipient robot.
  - "Off-target pass": If the human's aim significantly deviates from the recipient robot, the system provides no correction as the aim provided by the human user is deemed a poor pass.
  - Power Adjustment Based on Distance: The system automatically calculates the appropriate kick power based on the distance to the intended target using a linear formula:

$$P = \min\left(15 + \frac{d}{100},\ 80\right) \tag{6}$$

    where P is the kick effort (in units) and d is the distance from the target in millimeters. Kick power starts at a minimum of 15 units and is capped at 80 units, based on empirical testing in the simulator.

    This adjustment plays a key role, with its behavior depending on the operator's preferences:

    * **Cognitive Offloading:** This feature eliminates the need for the human operator to manually adjust the pass power, ensuring consistent passes regardless of distance.

Examples of Assisted Passing

- Slight Aim Correction:
  - The human user aims the pass in the general direction of a teammate but is aiming off by more than 50 cm from the robot. This is deemed as a "Near-target pass" and subsequently, the aim is assisted until the aim is considered "On-Target". This results in a more accurate pass to the teammate while also acknowledging the human intent to pass to this teammate.
- No-Aim Correction:
  - The human user aims in a direction far off target from any other teammate (greater than the "Near-Target" threshold). This is deemed as an "Off-target pass", and thus the system interprets the human intent to not pass directly to a teammate.
- Power Adjustment based on Distance:
  - The human user intends to execute a pass shot with low power; however, a high kick effort/power is sent to the system due to the operator accidentally holding the pass button too long. The system, knowing the distance to the indented target, reduces the kick power to a predetermined amount based on the distance. This ensures that a receivable pass is made. Conversely, for a long pass where the user under-powers, the system might boost the power.

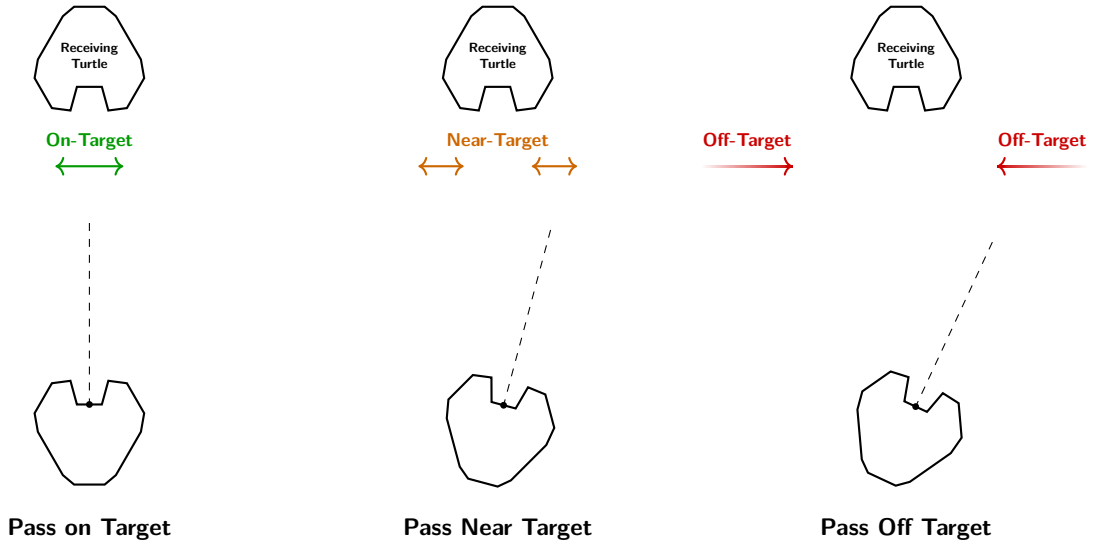

Figure 7: Visual representation of pass accuracy scenarios for receiving turtles based on lateral distance. Red fading arrows represent "Off-target" areas that continue past the field of view.

Calculations for Assisted Passing To determine whether a pass is "On", "Near", or "Off-Target", the system calculates the lateral distance from the receiving robot based on where the operator is aiming the robot. By implementing the approach seen in fig. 7, the system can ensure a consistent pass accuracy throughout the field. The human intent is determined by the system based on where the operator is aiming the robot, which in most scenarios is the intended pass receiver. The system

implements distance thresholds that adjust based on the passing distance to the target robot. This provides thinner tolerances for short passes and wider tolerances for longer passes, which require more accurate aiming.

Lateral Distance Calculation: The lateral error is determined by looking at the perpendicular distance from the receiving robot to the trajectory line of the passing robot:

$$d_{\perp} = |(\mathbf{r}_{\text{target}} - \mathbf{r}_{\text{robot}}) \cdot \mathbf{n}_{\perp}| \tag{7}$$

where $\mathbf{r}_{\text{target}}$ and $\mathbf{r}_{\text{robot}}$ are the target and robot positions, and $\mathbf{n}_{\perp}$ is the unit vector perpendicular to the robot's aiming direction.

Adaptive Threshold Calculation: The assistance thresholds are based on the target distance d_{target} using linear equations:

$$\tau_{direct} = 150 + 0.02917 \cdot d_{target} \tag{8}$$

$$\tau_{assisted} = 400 + 0.092 \cdot d_{target} \tag{9}$$

where d_{target} is the distance to the target teammate in millimeters. These coefficients can be adjusted to the operator's preferences. However, in this implementation, they are determined based on these constraints: an "On-target" aim threshold is up to 150 mm lateral error and a "Near-target" aim is up to 400 mm at 0 m distance to the target robot. Furthermore, at a distance of 12 m, the "On-target" threshold is up to 500 mm lateral error and "Near-target" is up to 1500 mm. This results in the following slopes: $m = \frac{\Delta y}{\Delta x} = \frac{500-150}{12000-0} = 0.02917$ and $m = \frac{1500-400}{12000-0} = 0.092$.

4.3.3 Assisted Shooting

Interpreting Human Shot Intent: Similarly to assisted passing, the intent of the shot can be determined by:

- Shot Trigger: A dedicated joystick button which signals the intent to take a shot.
- Aim Indication: The current direction of the analog joystick stick at the moment of the trigger of the shot indicates the general direction of the shot intended by the human user.
- Shot Power/Type: Determined by the user input through button press duration. Depending on the duration, different shot types such as flat shots (on the ground) or lob shots can be taken.

Autonomous Assistance for Shooting Once the target intent has been determined, the shared control logic uses RTDB data (ball position, shooting robot position, target location) to determine whether a shot should be assisted.

- Aim assistance
  - The system identifies the location of the opponent's goal based on team colour and only applies assistance to this goal.
  - Based on where the robot's line of fire intersects the goal line, a decision is made on whether assistance is required.

- Execution Adjustment: The system now corrects the human's shot direction based on the following scenarios seen in fig. 8:
  - "On Target Shot": If the human user's aim is within the goal frame of the opponent's goal, no correction is applied to the aim, and the shot is executed.
  - "Near Target Shot": If the human user's aim is at the goal frame or x distance outside of the goal frame, the system corrects the aim of the robot executing the shot until it aims within the frame of the goal.
  - "Off Target Shot": If the human's aim is significantly outside of the opponent's goal, the system does not correct the aim provided by the human user as it is deemed a poor shot.

Examples of Assisted Shooting

- Slight Aim Correction:
  - The human user aims the pass in the general direction of the opponent's goal but is aiming off by $1500mm$(demonstration value) or less from the goal post. This is deemed as a "Near-target shot" and subsequently, the aim is assisted until the aim is within the goal frame. This results in a more accurate shot on the opponent's goal while also acknowledging the human intent to aim on target.
- No Aim Correction:
  - The human user aims in a direction far off target from the opponent's goal posts (greater than 1500 mm). This is deemed an "Off-target shot", and thus the system chooses not to intervene, as such a shot should not be rewarded with assistance. This design respects the possibility that human intent may be strategic, such as a clearance shot, not simply a misjudged attempt.

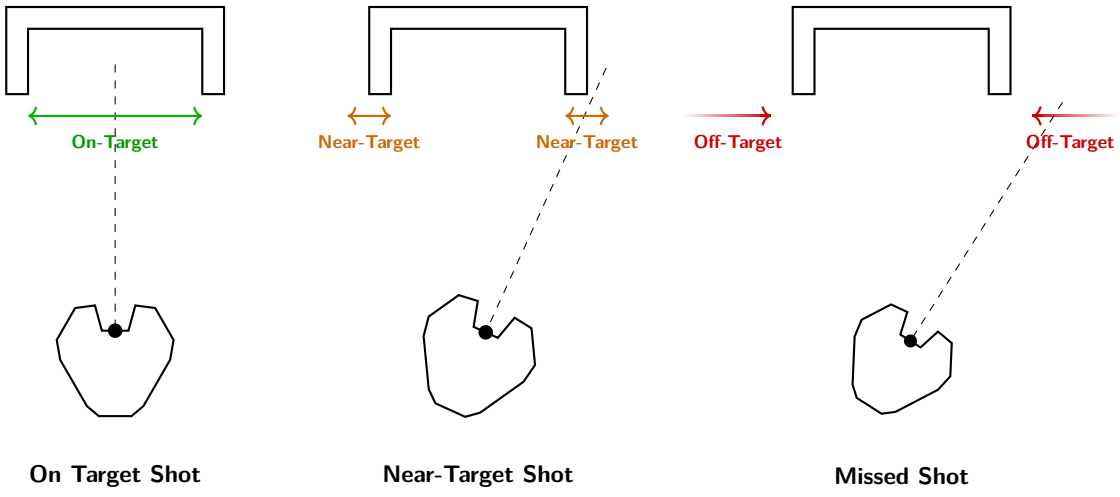

Figure 8: Visual representation of assisted shooting scenarios for "On-Target", "Near-Target", and "Missed Shots". Red fading arrows represent "Off-target" areas that continue past the field of view.

Calculations for Assisted Shooting The system uses geometric calculations to determine whether a shot is classified as "On Target", "Near Target", or "Off Target". This is achieved by computing

the intersection point of the robot's shooting direction with the opponent's goal line using parametric equations.

Let the robot position be $\mathbf{r}_{\text{robot}} = (x_r, y_r)$, and its facing direction vector be $\mathbf{d} = (d_x, d_y)$. The shooting trajectory is defined by:

$$\mathbf{P}(t) = \mathbf{r}_{\text{robot}} + t \cdot \mathbf{d} = (x_r + td_x,\ y_r + td_y) \tag{10}$$

To compute the intersection with the opponent's goal line $y = y_{\text{goal}}$, solve for t:

$$t = \frac{y_{\text{goal}} - y_r}{d_y} \tag{11}$$

Then compute the x-coordinate of the intersection point:

$$x_{\text{intersect}} = x_r + t \cdot d_x = x_r + \frac{(y_{\text{goal}} - y_r) \cdot d_x}{d_y} \tag{12}$$

A shot is classified as "On Target" if:

$$x_{\text{left\_post}} \leq x_{\text{intersect}} \leq x_{\text{right\_post}} \quad \text{and} \quad t > 0 \tag{13}$$

Here, $y_{\text{goal}} = y_{\text{goal\_line}} - \texttt{GOAL\_AIM\_OFFSET\_MM}$, accounting for the ball radius and the width of the goal post.

For "Near-target" classification, compute the lateral distance from the intersection to the closest goalpost:

$$d_{\perp} = \min\left\{ |x_{\text{intersect}} - x_{\text{left\_post}}|,\ |x_{\text{intersect}} - x_{\text{right\_post}}| \right\} \tag{14}$$

The shot is "Near Target" if:

$$d_{\perp} \leq \texttt{GOAL\_TOLERANCE} \quad \text{and} \quad t > 0 \tag{15}$$

where $\texttt{GOAL\_TOLERANCE} = 1500$ mm, a value chosen by the operator based on the desired aiming tolerance.

4.3.4 Assisted kick effort for more accurate assisted shooting

The current assisted shooting logic mainly considers the robot aim along the x-axis when determining if a shot requires assistance. It does not address the vertical component (z-axis) of the ball's trajectory, which is important to prevent shots from overshooting the goal. Accounting for the x-axis, the z-axis, or both can be considered as different levels of assistance depending on the desired level of autonomy. A valuable future improvement would be to incorporate the z-axis aim into the assisted shooting calculation, as this would allow the required kick effort to be estimated. The kick effort would be estimated based on the horizontal distance to the goal and the vertical elevation needed to clear obstacles such as defenders and goalkeepers, but stay below the crossbar. Such calculations have been made by [6]. The shot-error zones where the z axis is considered are shown in fig. 9.

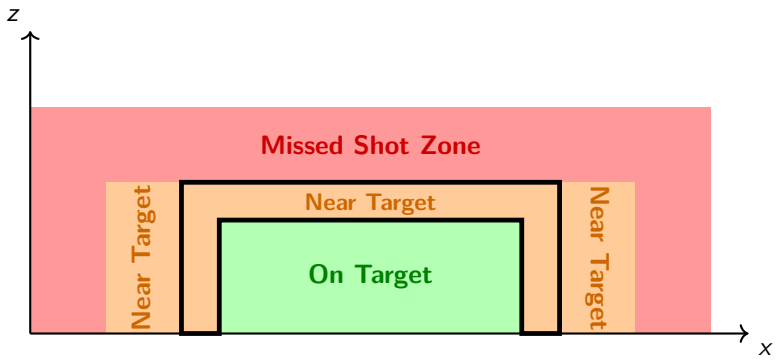

Figure 9: Shot-error zones relative to the goal taking vertical kick-effort aim into consideration: missed-shot (red), near-target (orange), and on-target (green).

An additional safety feature can be introduced that limits the maximum kick effort when the ball is aimed towards areas outside of unprotected field boundaries(e.g. sidelines, areas behind goal with no safety net). The benefit of this becomes clear when, for example, a shot is aimed towards the sideline of the field with a high kick effort; if the kick effort is too high, the system could reduce the kick effort. This should ensure that the ball can leave the field but in a controlled manner which does not cause a safety threat to persons or objects outside of the playing field.

This safety-aware modulation of power would be particularly valuable in confined or human-proximate environments, and aligns with general principles of safe human-robot interaction.

4.4 PD Controllers for Assisted Skills

To achieve an accurate and responsive angle correction for assisted passing and shooting, a proportional derivative control loop (PD) was implemented. This approach was chosen because the respective RTDB commands accept velocity commands (v_ϕ), which specify the rotation speed but not the target position. Despite the fact that the existing robot framework already has existing controllers implemented, the following PD controllers were a practical design choice that allowed the framework to be separate from the existing code. This design choice and its limitations are discussed in more detail in section 6.4. A PD controller is used to turn angular position errors into rotational velocity outputs. This is done by applying a corrective output based on the current error and the desired position of the robot, so that the robot tends towards the desired position. In this case, the control output is the robot's rotational velocity (v_ϕ), determined by the following equation:

$$v_\phi(t) = K_p e(t) + K_d \frac{de(t)}{dt} \tag{16}$$

Where:

- $e(t)$ is the angular error at time t, showing the difference between the robot's current orientation and its desired orientation .
- The Proportional term ($K_p e(t)$) applies a corrective force proportional to the current error.
- The Derivative term ($K_d \frac{de(t)}{dt}$) applies a damping force proportional to the rate of change of the error. This term helps with stability, reducing overshoot, and preventing oscillations.

Several more advanced control techniques were used during the tuning of the PD controllers to enhance performance and stability.

1. **Gain Scheduling:** Instead of relying on fixed parameters, the proportional gain (K_p) and derivative gain (K_d) are adjusted based on the magnitude of the angular error through a linear equation. For large errors, a lower K_p and a higher K_d can be implemented for smooth and stable rotation without excessive overshooting. For smaller errors, as the robot approaches its desired orientation, K_p is increased and K_d is decreased to provide a faster and responsive correction to fine-tune the final alignment. This approach allows for both large and small angular rotations depending on the level of assistance desired by the human user. In this implementation, K_p values range from a maximum of 1.5 at zero error to a minimum of 0.8 for large errors, whereas K_d values ranges from a minimum of 0.8 at zero error to a maximum of 1.2 for large errors:

$$K_p = 1.5 - 0.0156 \cdot |\theta_{\text{error}}| \tag{17}$$

$$K_d = 0.8 + 0.0089 \cdot |\theta_{\text{error}}| \tag{18}$$

   where $|\theta_{\text{error}}|$ is the magnitude of the angular error in degrees.

2. **Derivative Filtering:** The raw derivative of the error is susceptible to communication delays between sending and receiving RTDB commands, causing unstable control. By utilizing derivative filtering, specifically a low-pass filter, the rate-of-change information can be preserved while attenuating noise spikes that would otherwise cause unstable behavior.

$$\begin{aligned} d_{\text{raw}} &= \frac{\theta_{\text{current}} - \theta_{\text{last}}}{\Delta t} \\ d_{\text{filtered}} &= 0.7 \cdot d_{\text{filtered}} + 0.3 \cdot d_{\text{raw}} \\ \dot{\theta} &= K_d \cdot d_{\text{filtered}} \end{aligned} \tag{19}$$

3. **Velocity-Based Limiting**: Constraining the maximum rotational speed based on the magnitude of the angular error acts as a second layer to ensure that there are no abnormally high proportional gains causing overshoot. Thus, this acts as an adaptive cap which prevents overshoot during large corrections and ensures a smooth deceleration as the robot approaches its target. The controller therefore maintains responsiveness without sacrificing stability.

```
# Velocity-based limiting (prevents overshoot)
if error_magnitude > 30:  # Large errors
    max_speed = min(40, error_magnitude * 1.2)  # Scale with error
        but cap
elif error_magnitude > 10:  # Medium errors
    max_speed = min(50, error_magnitude * 2.0)
else:  # Small errors
    max_speed = 60
```

4.5 Switching Between Turtles

To understand how human operators manage multi-agent teams, the framework provides two modes for player switching: Manual and Automatic. Through these modes, possibilities open for analysis of

human strategy under two different levels of automation. In manual mode, the operator has control over the selection of the player at all times through the number keys or the controller buttons. This mode aims to provide a baseline for pure human strategy in team management, as decision making has to be done purely by the operator. In automatic mode, the framework offloads tasks that are required to be performed by the human operator. This is done by automating the most common and predictable switching tasks. Priority is given to robots with ball possession, or secondarily, the robot closest to the ball. By automatically handling these tasks, the system allows the operator to focus on future strategic moves that are optimal given the game scenario. This is important not only for learning by demonstration but also for comparing operator performance between the two modes. This opens possibilities for investigating how different levels of cognitive tasks affect high-level strategy in HRI.

As mentioned above, automatic switching is implemented to offload repetitive tasks and to keep the game of play smooth. In AUTO mode, the system prioritizes ball possession, automatically switching control to whichever robot currently has the ball (`CPBrobot == 1`). This allows the user to always be in control of the robot that has possession of the ball, removing the need for manual switching by the human after passes or ball interceptions. The implementation of this logic is shown below:

```
if auto_mode:
    # AUTO MODE: Ball possession takes priority
    if turtle_with_ball is not None and turtle_with_ball in
        current_team_robots:
        if active_turtle != turtle_with_ball:
            active_turtle = turtle_with_ball
```

If there is no active robot in possession of the ball, the system instead selects the robot closest to the ball. While the robot closest to the ball might not always be the most relevant, for example, a tactical marking or blocking might make a robot more relevant- distance is a simple and effective measurement of relevance and thus an implementation that takes into account most situations encountered in game play. The logic for selecting the closest robot is given below:

```
# Find the robot on the current team closest to the ball
closest_turtle = None
min_distance = float('inf')
for robot_id in current_team_robots:
    if robot_id in robot_distances and robot_distances[robot_id] <
        min_distance:
        min_distance = robot_distances[robot_id]
        closest_turtle = robot_id
```

However, manual switching is also implemented to allow the human user to have direct control over which robot they control. In MANUAL mode, the user is able to cycle between active turtles using designated keys or buttons. The numbers keys (1-6) allow direct selection of specific robots, while the PS4 controller's `R1` button cycles through available robots in ascending order. Moreover, if the human user is controlling a robot significantly far away from the area of play, a command can be sent to switch to the closest turtle to the ball using the `'c'` key or the `'L1'` trigger on the joystick.

5 Results

This section presents the results and evaluation of the performance and effectiveness of the shared control framework.

5.1 Testing Setup

Testing of the implemented shared control framework was carried out mainly within the wheeled robot simulator developed by Tech United. Within this simulator, different configurations of the number of robots, their team, and team roles were configured. Once the logic showed no signs of bugs, physical testing was also performed in a demo field.

5.2 Results

As mentioned in section 5.1, the shared control logic was tested in the simulator and on the physical wheeled robots. The results of this implementation is best visualized through videos which can be viewed through the links below. These videos demonstrate the individual components of the shared control logic as well as a combined playing scenario.

- Field Boundary Impedance: This video shows the human operator trying to move the robot beyond the field boundaries. However, the field impedance logic prevents this from happening for safety reasons.
- Assisted Ball Intercept: Here, the human operator moves the robot within the intercept radius of the ball. This initiates an automatic intercept of the ball.
- Assisted Shooting: In shooting scenarios, the control logic assists the human operator in aiming on target when aiming slightly outside of the goal area. Corrective aim is given due to factors like field of view limiting the human operator from aiming on target.
- Assisted Passing: This demonstrates how the shared control assists the operator in executing accurate shots. The user indicates the intent of passing and the target robot, while the control logic handles low level corrective aim.
- Switching Between turtles: This video demonstrates a full team of robots continuously passing to each other. The manually controlled robot is the robot with current possession of the ball, thus, when a pass is executed, the manually controlled robot switches.

Overall, the system performed consistently across simulation and physical tests. In all test cases, the robot responded smoothly to user inputs, while autonomously enhancing safety and precision. This implementation demonstrates that shared control in soccer robotics help reduce the cognitive workload of the human operator, allowing for more focus on higher level strategic plays.

6 Future Work

In this section, some potential directions for future research and expansion based on the current shared control system are provided. Although the current deployment shows strong performance in essential control and assisted behaviors, there are still significant chances to further improve the independence, safety, and user experience of the systems. The following parts discuss possible improvements in field boundary impedance for safer navigation, advanced ball intercept strategies, and the expansion of assisted skills to enable more sophisticated and natural robot behaviors.

6.1 Field Boundary Impedance

The current field boundary impedance logic robustly ensures safety by limiting a robots velocity as it reaches a field boundary. However, further enhancements can improve both the flexibility and the user experience of this function. Future work could focus on making the impedance context aware based on the game environment, allowing for temporary exemptions in specific tactical scenarios. Such tactics mentioned in section 4.1.2 are to allow a robot to make an overlap run just outside the boundary of the field or when throw-in plays are awarded. Lastly, the current implementation of the field boundary impedance logic was implemented within the Simulink motion drive-train control module, which allows for direct manipulation of the robot velocity. Ideally, the boundary impedance logic should be reimplemented in Python to integrate with the rest of the system.

6.2 Switching between turtles

An efficient and reliable automatic robot switching mechanism plays an important role in the shared control logic, especially since robot soccer is a fast paced game. The current implementation allows for the human operator to have automatic switching, by assigning control to the robot closest to the ball or with ball possession, or manual- by directly cycling/selecting the desired manually controlled robot. Although this approach provides basic functionality, there is room for improvement. Future work would include prioritizing robots based on their strategic position, role, or likelihood of influencing the game of play. For example, during the attacking phase the system would only consider passing to other attackers or during defensive situations, prioritize switching to defenders closest to the ball or best positioned to block a shot. Moreover, providing clear visual or haptic feedback for the operator is essential for situational awareness and smooth game of play when transitioning between robots. Communicating the actively manual controlled robot to the human operator is a future improvement that is essential for human-robot interaction in fast-paced game situations.

Another direction for future work is to implement multiple operators in each robot team. The current framework focuses on control by a single human operator; however, an alternative playing configuration is seen in fig. 10, where dedicated operators are assigned to each robot. This parallel human-robot control can open up research on team coordination and performance when roles are distributed between operators. In partial multi-operator scenarios(where two, three, or four joysticks are being used), the switching algorithm must also change. Each operator would be responsible for a subset of robots. The assignment of robots for each operator could be based on predefined roles, like attacking or defending operators, or dynamically switching where ball proximity or tactical play comes into consideration. This multi-operator control reduces the cognitive workload per operator, allowing each operator to communicate with each other to make strategic and tactical plays, ultimately making a more smooth and successful game.

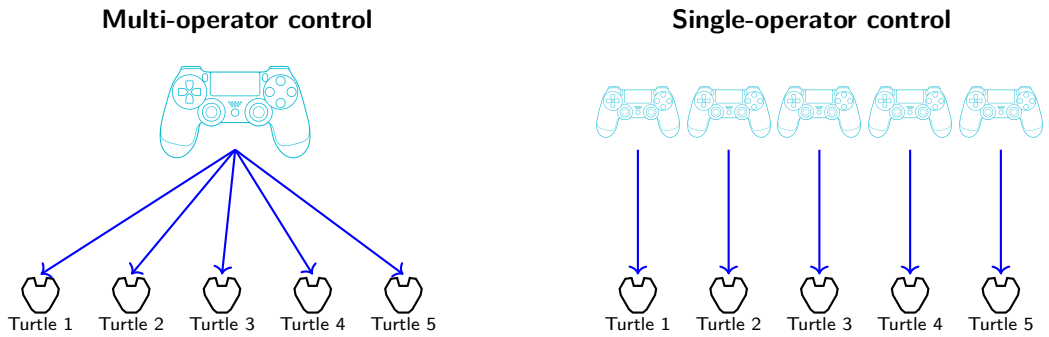

Figure 10: Two different modes of human-robot shared control. Left: One operator controls five robots via a single joystick controller. Right: Up to five operators each control one robot independently.

6.3 Assisted Obstacle Avoidance

Assisted obstacle avoidance has not been implemented in the current framework due to its complexity; however, it is a feature that is crucial to the safety of robots themselves and to have an effective game of play. Although the current framework relies on the operator to react to obstacles, future implementation would involve fusion of vision sensors and world-model data to detect static and dynamic obstacles. The robot's path or velocity can be proactively adjusted through obstacle avoidance algorithms to prevent collisions. A basic implementation could include a similar approach using impedance to avoid other obstacles, limiting the velocity in the direction of obstacles. Alternatively, a more realistic approach is a motion-planning algorithm which takes a desired setpoint and computes a collision-free trajectory that traverses around a given obstacle. An important consideration is to implement these features without severely overriding the strategic intent of the human operator. Sliding autonomy is a viable solution to this dilemma as it would allow a fraction of the input to still be retained by the human operator. By tuning this fraction of the input throughout the obstacle avoidance maneuver, the human operator can carry out their strategic intent. Including adaptive avoidance thresholds would allow the system to be more conservative in congested situations and more aggressive in spacious environments. These enhancements discussed would improve safety, but also create more fluid and confident operator control during complex game situations.

6.4 Integration of a shared-control framework within existing robot control architecture

The implementation of PD controllers was a practical design choice that served its purpose of being a functional and simple implementation. This allowed the existing low-level robot control architecture to not be modified and a higher-level shared control framework to be implemented only using data available through the RTDB. During the integration phase, it was possible to add and test new features without interfering with the existing architecture. The drawback with this design choice was that it bypassed the already well-tuned built-in control components. The implementation of PD controllers in section 4.4, while functional, could create slower control performance compared to direct integration. This concerns previously mentioned future implementations as well, particularly context-aware and more advanced strategy implementations. These implementations might benefit from direct implementation, as not all necessary information (such as robot roles, play execution state, or complex kick commands) are available through RTDB.

7 Conclusion

In this report, a Shared Human-Robot Control framework for Tech United soccer robots has been designed, implemented, and evaluated. The framework addresses the balance between autonomous assistance for precision and human strategic intent during the game of play. This is a central issue within human-robot interaction research and is recognized within soccer robotics. Fully manual control becomes impractical due to low-level precision challenges, while fully autonomous game play lacks strategic insights that would otherwise only be learned through demonstration. Through this dilemma, the framework successfully creates an environment in which human intuition and robot execution collaborate and assist each other. Furthermore, the framework, having the RTDB at its core communication channel, allows it to be decoupled from the existing autonomous system. Within a Python file, several key functions were implemented to incorporate levels of assistance: such as field boundary impedance, ball intercepting, passing, shooting, and player switching. The goal is to collect meaningful data on human strategy through demonstration. These functions not only offload repetitive tasks to the robot, but more importantly manage low-level control tasks that require speed and accuracy, which are difficult to achieve through teleoperation alone. The system keeps human strategic intent as the driving factor while offering autonomous support to maintain usability and efficient game play. Key features presented in the control logic are the following:

- Field Boundary Impedance: Automatically limits the robot velocity as it approaches field boundaries to ensure safety, allowing the human operator to focus on strategic play.
- Automatic Ball Intercept: Assists operators by automating intercepting the ball when the robot is within a defined distance to the ball, reducing.
- Assisted Low-level Skills: Implements corrective aim for passing and shooting, assisting accuracy while preserving human decision making and strategy.
- Switching Between Turtles: Allows for automatic transitioning between robots based on those in strategic positions or with ball possession, while maintaining manual override options.

However, there are several key areas that remain open for improvement. Future developments include context-aware boundary impedance and refined automatic ball interception which takes into account the ball velocity and a more optimized distance threshold. Furthermore, implementing obstacle avoidance to provide safe maneuverability for both robots and humans, while also integrating adaptive passing thresholds and assisted kick effort to improve performance when passing and shooting. Ultimately, this framework not only aims to provide a platform for future developments in researching human strategies and learning by demonstration, but also as an engaging way for people at demonstrations to interact with the soccer robots in a fun and safe environment.

8 References

[1] Ruben M. Beumer et al. "Tech United Eindhoven Middle Size League Winner 2023". In: 2024, pp. 428–439. DOI: 10.1007/978-3-031-55015-7{_}36.

[2] Ahmetcan Erdogan and Brenna D. Argall. "The effect of robotic wheelchair control paradigm and interface on user performance, effort and preference: An experimental assessment". In: *Robotics and Autonomous Systems* 94 (Aug. 2017), pp. 282–297. ISSN: 0921-8890. DOI: 10.1016/J.ROBOT.2017.04.013. URL: https://www.sciencedirect.com/science/article/abs/pii/S0921889016303670.

[3] Gerhard Goos et al. *LNCS 3280 - Computer and Information Sciences - ISCIS 2004*. Tech. rep. 2004.

[4] Ashok K. Hemal and Rajeev Kumar. "Role of Patient Side Surgeon in Robotics". In: *Robotics in Urologic Surgery* (Jan. 2008), pp. 31–37. DOI: 10.1016/B978-1-4160-2465-1.50009-2. URL: https://www.sciencedirect.com/science/article/abs/pii/B9781416024651500092.

[5] Sarah Hopko, Jingkun Wang, and Ranjana Mehta. *Human Factors Considerations and Metrics in Shared Space Human-Robot Collaboration: A Systematic Review*. Feb. 2022. DOI: 10.3389/frobt.2022.799522.

[6] C M Kengen, Y G M Douven, and M J G Van De Molengraft. *TOWARDS A MORE REPRODUCIBLE SHOT WITH THE TECH UNITED SOCCER ROBOTS*. Tech. rep.

[7] Mauricio Marcano et al. "A Review of Shared Control for Automated Vehicles: Theory and Applications". In: *IEEE Transactions on Human-Machine Systems* 50.6 (Dec. 2020), pp. 475–491. ISSN: 21682305. DOI: 10.1109/THMS.2020.3017748.

[8] Sachiko Matsumoto and Laurel D Riek. "Shared Control in Human Robot Teaming: Toward Context-Aware Communication". In: (2022). URL: www.aaai.org.

[9] Selma Musić. *Shared Control for Human-Robot Team Interaction*. Tech. rep.

[10] Harish Ravichandar et al. "Recent Advances in Robot Learning from Demonstration". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.Volume 3, 2020 (May 2020), pp. 297–330. ISSN: 25735144. DOI: 10.1146/ANNUREV-CONTROL-100819-063206/CITE/REFWORKS. URL: https://www.annualreviews.org/content/journals/10.1146/annurev-control-100819-063206.