Robotics SP 2025
Università della Svizzera italiana

Tower Dropper

Group Z

Paolo Deidda (paolo.deidda@usi.ch)
Francesco Costa (francesco.costa@usi.ch)
Raffaele Perri (raffaele.perri@usi.ch)

`https://github.com/USI-Projects-Collection/FinalProject-Robotics.git`

May 24, 2025

Contents

Setup

To run the code follow the setup and execution instructions provided in the `README.md` file included in this repository.

1 Introduction

1.1 Objectives and Success Criteria

2 Workflow of the Project

2.1 Global Path Planning

2.1.1 Algorithm Choice & Rationale

2.1.2 Implementation Details

2.1.3 Tests & Metrics

2.1.4 Challenges Encountered & Fixes

2.2 Orbit Controller

3 Tower Detection and Interaction System

3.1 System Overview

The tower detection and interaction system represents a sophisticated autonomous targeting solution. The system integrates multiple sensor modalities including range sensors, RGB camera, and odometry to enable a mobile robot to autonomously locate, approach, orbit, and engage tower targets in a dynamic environment. The core architecture follows a finite state machine approach with six primary operational states: *find_tower*, *position_for_orbit*, *orbit_tower*, *avoid_obstacle*, *align_tower*, and *shoot_tower*. This state-based design ensures robust behavior transitions and clear operational logic flow through well-defined transition criteria and sensor-based decision making. The main control loop executes at 60 Hz, providing real-time responsiveness while maintaining computational efficiency.
Key ROS2 topics include velocity commands (`cmd_vel`), odometry data (`odom`), individual range sensor readings (`/rm0/range_0-3`), and camera imagery (`/rm0/camera/image_color`).

3.2 Sensor Integration and Data Processing

3.2.1 Range Sensor Configuration and Calibration

The system utilizes four range sensors positioned strategically around the robot platform to provide environmental awareness:

- **Range_0 (Center-right)**: Primary sensor for orbit distance maintenance during left-side orbital maneuvers. Positioned at the robot's right quadrant with a detection range up to 10 meters.
- **Range_1 (Front-right)**: Primary sensor for initial tower detection and forward-right obstacle detection. Critical for the search phase and collision avoidance.
- **Range_2 (Center-left)**: Secondary sensor utilized during obstacle avoidance protocols, enabling the robot to maintain safe distances from obstacles while navigating around them.
- **Range_3 (Front-left)**: Forward collision prevention sensor that triggers emergency avoidance behaviors when obstacles are detected within critical thresholds.

Each sensor operates through dedicated ROS2 subscribers that continuously update distance measurements at the sensor's native frequency. The system implements sensor fusion by maintaining a real-time comparison between multiple sensor readings to determine the most reliable distance measurements and detect sensor failures or environmental interference. Readings exceeding 10 meters are considered invalid and replaced with the maximum range value. The system also implements a smoothing algorithm to reduce measurement noise:

$$d_{filtered}(t) = \alpha \cdot d_{raw}(t) + (1 - \alpha) \cdot d_{filtered}(t - 1) \tag{1}$$

where $\alpha = 0.8$ provides a balance between responsiveness and stability.

3.2.2 Computer Vision System Architecture

The vision system employs a comprehensive computer vision pipeline optimized for real-time tower detection and tracking. The process begins with image acquisition from the robot's RGB camera, followed by color space conversion and multi-stage filtering.

Color Space Processing: The system converts incoming BGR images to HSV color space to achieve robust color-based detection under varying lighting conditions. HSV provides superior color constancy compared to RGB, particularly important for outdoor or variable lighting scenarios.

Multi-Range Color Filtering: To account for the bimodal distribution of red pixels in HSV space, the system implements dual-range thresholding:

$$\text{Mask}_1 = \text{inRange}(\text{HSV}, [0, 100, 100], [10, 255, 255]) \tag{2}$$

$$\text{Mask}_2 = \text{inRange}(\text{HSV}, [160, 100, 100], [179, 255, 255]) \tag{3}$$

$$\text{Mask}_{final} = \text{Mask}_1 \cup \text{Mask}_2 \tag{4}$$

This approach captures both low-end $(0 - 10°)$ and high-end $(160 - 179°)$ red hues while maintaining robust detection across different illumination conditions.

Tower Position Calculation: The system calculates tower position using spatial moments of the binary mask:

$$M_{pq} = \sum_{x,y} x^p y^q \cdot \text{Mask}(x, y) \tag{5}$$

$$C_x = \frac{M_{10}}{M_{00}} \tag{6}$$

$$p_{tower} = \frac{C_x - W/2}{W/2} \tag{7}$$

where M_{pq} represents the (p, q)-th moment, C_x is the centroid x-coordinate, W is the image width, and $p_{tower} \in [-1, 1]$ represents the normalized tower position.

Tower Visibility Assessment: The system implements a visibility metric based on the ratio of red pixels to total image pixels:

$$R_{red} = \frac{\sum_{x,y} \text{Mask}_{final}(x, y)}{W \times H} \tag{8}$$

Tower visibility is confirmed when $R_{red} > \tau_{visibility}$, where the threshold $\tau_{visibility}$ is dynamically adjusted based on environmental conditions and historical detection performance.

3.3 State Machine Implementation and Control Logic

3.3.1 State Transition Architecture

The finite state machine implementation employs a hierarchical structure with clearly defined transition conditions and state-specific behaviors. Each state maintains internal variables for tracking progress and implementing timeouts to prevent infinite loops or deadlock conditions. State transitions are triggered by combinations of sensor readings, timing constraints, and task completion flags. The system implements hysteresis in transition conditions to prevent oscillation between states due to sensor noise or borderline threshold conditions.

3.3.2 Tower Discovery Phase: *find_tower* State

The initial search phase implements a systematic rotational search pattern optimized for comprehensive environmental scanning while minimizing search time. **Search Strategy:** The robot executes counterclockwise rotation at $\omega_{search} = 0.5$ rad/s while continuously monitoring the front-right range sensor (Range_1). This angular velocity provides optimal balance between search speed and sensor update rates, ensuring no potential targets are missed during rotation. **Detection Criteria:** Tower detection occurs when the front-right sensor measurement satisfies:

$$d_{Range1} \leq d_{target} + \delta_{tolerance} \tag{9}$$

where $d_{target} = 2.0$ meters represents the desired orbital radius and $\delta_{tolerance} = 0.1$ meters provides measurement uncertainty accommodation. **Multi-Sensor Validation:** Upon initial detection, the system performs cross-validation using adjacent sensors to confirm target authenticity and eliminate false positives from environmental artifacts or sensor noise. The validation process requires consistent readings across multiple sensor cycles to prevent premature state transitions.

3.3.3 Orbital Positioning Phase: *position_for_orbit* State

This critical state ensures proper spatial relationship between robot and tower before initiating orbital maneuvers. **Geometric Positioning Strategy:** The system executes a controlled rotation to position the detected tower relative to the robot's right side, enabling subsequent left-side orbital motion. The positioning maneuver uses angular velocity $\omega_{position} = 0.5$ rad/s while monitoring the back-right sensor (Range_0). **Position Validation Criteria:** Successful positioning requires the back-right sensor to detect the tower within acceptable bounds:

$$d_{Range0} \leq d_{target} + \delta_{position} \tag{10}$$

where $\delta_{position} = 0.5$ meters provides sufficient tolerance for initial orbital entry while maintaining proximity to the target distance. **Timeout and Recovery Mechanisms:** The state implements timeout protection to prevent infinite positioning attempts. If positioning cannot be achieved within a predetermined time window, the system reverts to the search state with modified search parameters.

3.3.4 Orbital Navigation Control: *orbit_tower* State

The orbital control system represents the most sophisticated aspect of the navigation architecture, combining multiple control loops for stable circular motion around the detected tower. **Multi-Loop Control Architecture:** The orbital controller implements two primary control loops operating in parallel:

1. **Distance Regulation Loop**: Maintains constant radial distance using range sensor feedback
2. **Angular Tracking Loop**: Centers tower in camera field of view using vision feedback

Distance Control Implementation: The distance regulation employs PID control with anti-windup protection:

$$e_d(t) = d_{measured} - d_{target} \; u_d(t) \qquad = K_p e_d(t) + K_i \int_{t_0}^{t} e_d(\tau)d\tau + K_d \frac{de_d(t)}{dt} \tag{11}$$

The integral term includes saturation limits to prevent windup:

$$\int_{t_0}^{t} e_d(\tau)d\tau = \begin{cases} I_{\max} & \text{if } \int e_d(\tau)d\tau > I_{\max} \\ I_{\min} & \text{if } \int e_d(\tau)d\tau < I_{\min} \\ \int e_d(\tau)d\tau & \text{otherwise} \end{cases} \tag{12}$$

where $I_{\max} = 5.0$ and $I_{\min} = -5.0$ prevent excessive integral buildup. **Velocity Synthesis and Coordination:** The orbital motion combines base velocities with correction terms from both control loops:

$$v_{linear} = v_{base} \cdot f_{distance}(e_d) \; \omega_{angular} \qquad = \omega_{base} + \alpha \cdot u_d + \beta \cdot p_{tower} \tag{13}$$

where:

$$f_{distance}(e_d) = \begin{cases} 0.8 & \text{if } e_d > 0.2 \text{ (too far)} \\ 1.2 & \text{if } e_d < -0.2 \text{ (too close)} \\ 1.0 & \text{otherwise} \end{cases} \tag{14}$$

The velocity adaptation function $f_{distance}$ provides reactive speed adjustments based on distance error magnitude, enhancing convergence to the target orbital radius. **Velocity Limiting and Safety Constraints:** The system implements comprehensive velocity limiting to ensure safe operation:

$$v_{linear} \in [0.05, 0.3] \text{ m/s} \; \omega_{angular} \qquad \in [-\infty, -0.1] \text{ rad/s} \tag{15}$$

The angular velocity constraint ensures consistent counterclockwise motion while preventing excessive rotation rates that could destabilize the orbital behavior.

3.3.5 Advanced Obstacle Avoidance Protocol: *avoid_obstacle* State

The obstacle avoidance system implements a sophisticated dual-objective navigation strategy that maintains safe distances from obstacles while preserving awareness of the primary tower target. **Obstacle Detection and Classification:** Obstacle detection triggers when front sensors detect objects within the critical threshold:

$$d_{Range1} \leq d_{critical} = 0.2 \text{ meters} \tag{16}$$

The system classifies obstacles based on their geometric properties and persistence, distinguishing between temporary obstructions and permanent environmental features. **Secondary Orbital Behavior:** Upon obstacle detection, the system initiates a secondary orbital pattern around the detected obstacle using the back-left sensor (Range_2) for guidance. This behavior mirrors the primary orbital algorithm but with modified parameters optimized for close-proximity navigation:

$$K_{p,obs} = 0.1 \tag{17}$$

$$K_{i,obs} = 0.0 \tag{18}$$

$$K_{d,obs} = 0.1 \tag{19}$$

$$d_{target,obs} = 0.2 \text{ meters} \tag{20}$$

The reduced proportional gain prevents oscillations in the confined space around obstacles, while the elimination of integral control prevents windup in rapidly changing obstacle scenarios. **Return Navigation Logic:** The system implements sophisticated logic for returning to the primary tower after obstacle avoidance. The return sequence monitors multiple sensor conditions:

1. Obstacle clearance detection using back-left sensor
2. Primary tower re-acquisition using back-right sensor
3. Spatial relationship validation between robot, obstacle, and tower

The return condition requires:

$$(d_{Range2} > d_{clearance}) \wedge (d_{Range0} < d_{tower,\max}) \wedge \text{flag}_{almost\_avoided} \tag{21}$$

where $d_{clearance} = 10.0$ meters indicates obstacle clearance and $\text{flag}_{almost\_avoided}$ prevents premature return attempts.

3.4 Tower Alignment and Targeting System

3.4.1 Multi-Modal Alignment Detection

The tower alignment system employs multiple complementary detection methods to ensure optimal positioning for target engagement. The system combines computer vision analysis with sensor-based geometric validation for robust alignment determination. **Computer Vision-Based Alignment:** The primary alignment detection utilizes advanced computer vision techniques to analyze tower geometry and orientation relative to the robot's position. **Contour Analysis and Shape Recognition:** The system performs morphological analysis on the binary mask obtained from color filtering:

1. **Contour Extraction**: Using OpenCV's `findContours` with `RETR_EXTERNAL` and `CHAIN_APPROX_NONE` to capture complete tower boundary information
2. **Largest Contour Selection**: Identification of the primary tower structure by selecting the contour with maximum area
3. **Bounding Rectangle Calculation**: Computation of axis-aligned bounding rectangle using `boundingRect` function

Aspect Ratio Analysis: The system calculates the aspect ratio of the detected tower structure:

$$AR = \frac{w_{bbox}}{h_{bbox}} \tag{22}$$

where w_{bbox} and h_{bbox} represent the width and height of the bounding rectangle respectively. Optimal shooting conditions are identified when:

$$AR > \tau_{aspect} = 0.5 \tag{23}$$

This threshold indicates that the robot is facing a complete side of the tower structure, providing maximum target surface area for engagement. **Advanced Width Analysis Method:** The system implements an alternative alignment detection method based on tower width analysis across image rows:

1. **Row-wise Analysis**: Scanning from bottom to top of the image to analyze tower width at each vertical level

2. **Maximum Width Extraction**: Identification of the row with maximum colored pixels, representing the tower's widest visible section

3. **Width Progression Tracking**: Monitoring tower width changes to detect complete tower faces

The width-based alignment criterion requires:

$$w_{current} \geq w_{min} + \delta_{width} \tag{24}$$

where w_{min} represents the minimum observed tower width and $\delta_{width} = 20$ pixels provides sufficient margin for reliable detection.

3.4.2 Sensor-Based Geometric Validation

To complement vision-based alignment, the system implements sensor-based geometric validation using bilateral range measurements. **Symmetry Detection:** The alignment validation compares readings from left and right range sensors:

$$\Delta_{symmetry} = |d_{Range1} - d_{Range3}| \tag{25}$$

Perfect alignment is achieved when:

$$\Delta_{symmetry} < \epsilon_{symmetry} \tag{26}$$

where $\epsilon_{symmetry}$ represents the allowable measurement tolerance for symmetric positioning. **Iterative Alignment Process:** The alignment process follows a systematic approach:

1. **Initial Rotation**: Controlled rotation at $\omega_{align} = -0.3$ rad/s to scan for geometric variations

2. **Asymmetry Detection**: Continuous monitoring of range sensor differences to identify tower edges

3. **Symmetry Convergence**: Fine positioning adjustments to achieve bilateral sensor equality

4. **Alignment Confirmation**: Final validation using both vision and sensor criteria

3.4.3 State Transition Logic and Validation

The transition from alignment to shooting state requires satisfaction of multiple criteria:

$$\text{Aligned} = (AR > \tau_{aspect}) \wedge (\Delta_{symmetry} < \epsilon_{symmetry}) \quad \wedge(\text{tower\_visible}) \wedge (\text{stable\_for} > t_{stabilization}) \tag{27}$$

where $t_{stabilization}$ ensures the robot maintains alignment for sufficient duration before engagement.

3.4.4 Projectile Launch System

The *shoot_tower* state executes the engagement sequence through integration with the CoppeliaSim physics engine. The system calculates projectile spawn position using coordinate transformation:

$$\mathbf{R} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{28}$$

$$\mathbf{p}_{projectile} = \mathbf{p}_{robot} + \mathbf{R} \cdot \mathbf{t}_{blaster} \tag{29}$$

where ψ represents the robot's yaw angle and $\mathbf{t}_{blaster} = [0.20, 0, 0.2]^T$ meters defines the blaster offset from the robot center. The projectile receives initial velocity $\mathbf{v}_{projectile} = \mathbf{R} \cdot [3.0, 0.0, 3.0]^T$ m/s, providing forward momentum with upward trajectory compensation for ballistic flight.

3.5 System Integration and Performance Optimization

3.5.1 Real-Time Processing Architecture

The complete system operates through a sophisticated real-time processing architecture designed for optimal performance and reliability in dynamic environments. **Timer-Based Control Loop:** The system utilizes a high-frequency ROS2 timer executing at 60 Hz to ensure responsive real-time behavior. This frequency provides excellent balance between computational efficiency and control responsiveness:

$$f_{control} = 60 \text{ Hz} \rightarrow T_{control} = 16.67 \text{ ms} \tag{30}$$

The control period of 16.67 milliseconds ensures adequate response time for sensor processing, state evaluation, and command generation while maintaining system stability. **Sensor Data Synchronization:** The system implements sensor data synchronization mechanisms to ensure coherent multi-modal sensor fusion:

1. **Timestamp Validation**: All sensor measurements include timestamp validation to detect and compensate for communication delays
2. **Data Consistency Checking**: Cross-validation between sensors to identify and filter erroneous readings
3. **Interpolation and Prediction**: Linear interpolation for missing sensor data and predictive algorithms for compensating sensor latency

3.5.2 State Management and Transition Logic

The finite state machine implementation includes sophisticated state management capabilities to ensure robust operation under various environmental conditions. **State Persistence and Memory:** Each state maintains internal memory variables to preserve important information across state transitions:

- **Orbital Parameters**: Preservation of orbital start position, accumulated rotation angle, and distance regulation history
- **Target Information**: Storage of tower position, dimensions, and tracking history for improved re-acquisition
- **Environmental Context**: Obstacle maps, sensor reliability metrics, and environmental adaptation parameters

Transition Hysteresis and Debouncing: To prevent state oscillations due to sensor noise or borderline conditions, the system implements transition hysteresis:

$$\text{Transition} A \rightarrow B = (Condition_B > \tau high) \quad \text{Transition} B \rightarrow A \quad = (Condition_A < \tau low) \tag{31}$$

where $\tau_{high} > \tau_{low}$ creates a hysteresis band that stabilizes state transitions.

3.5.3 Adaptive Parameter Tuning

The system incorporates adaptive mechanisms to optimize performance based on environmental conditions and operational experience. **Dynamic Threshold Adaptation:** Sensor thresholds and detection parameters adapt based on environmental conditions:

$$\tau_{adaptive}(t) = \tau_{base} + K_{adapt} \cdot \sigma_{noise}(t) \tag{32}$$

where $\sigma_{noise}(t)$ represents the current noise level estimation and K_{adapt} controls the adaptation rate. **PID Parameter Scheduling:** The control system implements gain scheduling based on operational context:

$$K_p(context) = \begin{cases} K_{p,aggressive} & \text{if large error} \\ K_{p,conservative} & \text{if small error} \\ K_{p,transition} & \text{if changing states} \end{cases} \tag{33}$$

This approach provides robust control performance across different operational phases.

3.5.4 Error Handling and Recovery Mechanisms

The system includes comprehensive error handling and recovery capabilities to maintain operational reliability. **Sensor Failure Detection:** The system monitors sensor health through multiple indicators:

1. **Range Validation**: Detection of impossible or inconsistent range readings
2. **Communication Monitoring**: Tracking of sensor message rates and communication timeouts
3. **Cross-Sensor Validation**: Comparison between multiple sensors for consistency checking

Graceful Degradation Strategies: Upon sensor failure detection, the system implements graceful degradation:

$$\text{Degraded Operation} = \begin{cases} \text{Vision-only navigation} & \text{if range sensors fail} \\ \text{Range-only navigation} & \text{if camera fails} \\ \text{Dead-reckoning mode} & \text{if multiple sensors fail} \end{cases} \tag{34}$$

Recovery and Re-initialization: The system includes automatic recovery mechanisms:

- **State Reset**: Automatic return to search state when target is lost
- **Sensor Re-calibration**: Dynamic recalibration of sensor parameters during operation
- **Emergency Stop**: Safe system shutdown and velocity zeroing in critical failure scenarios

3.5.5 Performance Metrics and Optimization

The system tracks multiple performance metrics for continuous optimization: **Navigation Efficiency Metrics:**

$$\eta_{search} = \frac{t_{detection}}{t_{search\_total}} \tag{35}$$

$$\eta_{orbit} = \frac{\text{stable orbital time}}{\text{total orbital time}} \tag{36}$$

$$\eta_{alignment} = \frac{t_{successful\_alignment}}{t_{alignment\_attempts}} \tag{37}$$

Control Performance Metrics:

$$RMSE_{distance} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_{target} - d_{measured,i})^2} \quad \sigma_{angular} \qquad = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\omega_{commanded,i} - \omega_{actual,i})^2} \tag{38}$$

These metrics enable continuous system optimization and performance validation across different operational scenarios and environmental conditions.