

Trident: Design and Simulation of a Differential-Drive Pick and Place Robot

Joshua John Leo Valan Johnson

Roll No. 110123044

National Institute of Technology Tiruchirappalli

October 2026

Abstract

Trident is a small differential-drive robot that I designed and built for a competition task: to follow a line and then to carry coloured coins, each to the bin of its colour. This report presents the mechanical design of the robot and a simulation of it in which the whole task is carried out autonomously.

The robot is a printed plate of 180×150 mm on two N20 gearmotors and a ball caster, designed in SolidWorks and completed with its electronics in fifteen days. Its gripper does not lift. Two jaws with a parabolic profile, coupled by a pair of equal involute gears and driven by one servo, cage a coin on the floor, and the robot slides it to its destination; ridges on the jaws push the coin clear on release. A sliding condition for a coin on a parabolic jaw is derived as the design rule for the profile. The mass properties of the CAD model are then used to check the design. They show that the centre of mass lies inside the support triangle but that the wheels carry only 18% of the weight, which limits the acceleration to 1.8 m/s^2 at a friction coefficient of one and leaves a tipping margin of a factor of two. The depth camera cannot see the last 0.15 m of floor in front of the robot, and this constraint shapes the software.

The robot is modelled in Gazebo from the CAD export, with models of the depth camera, the inertial sensor and an eight-channel infrared array. The floor markings of the mission arena are dimensioned from the pitch of that array. Mapping, localisation and navigation use RTAB-Map and Nav2; a line follower, docking sequences and a mission state machine were written for the robot. The damping of the line follower is shown to be set by the distance of the array ahead of the axle.

In twelve simulated missions with four obstacle layouts the robot delivered all 36 coins, in 300 s per mission on average, scored by a referee on ground truth. The line was followed within 6 mm, the pose in the map was accurate to 10 mm RMS, the robot stopped on the pick markers within 7 mm, and coins were placed within 21 mm of the centres of their squares.

All measurements were taken in simulation. The printed robot won the district round of the competition and failed in the state round when its infrared array stopped working; no measurements exist for the hardware.

Keywords: mobile robot design, differential drive, caging gripper, line following, visual SLAM, navigation, Gazebo, ROS 2.

Contents

Nomenclature	4
1 Introduction	5
1.1 Context	5
1.2 Problem Statement and Objectives	5
1.3 Contributions	6
1.4 Organisation	6
2 Related Work	6
3 Design Evolution	7
3.1 Version 1: a Test Rig	7
3.2 Version 2: the Competition Robot	7
3.3 From the Hardware to the Simulation	8
4 Mechanical Design	8
4.1 Layout	8
4.2 Mass Properties and Their Consequences	9
4.3 Drive	10
4.4 Gripper	11
4.4.1 Caging in Place of Lifting	11
4.4.2 Jaw Profile	11
4.4.3 Release Ridges	12
4.4.4 Actuation	12
4.5 Sensor Placement	13
4.5.1 Depth Camera	13
4.5.2 Infrared Array	13
4.6 Manufacture	14
5 Electronics and Sensing	14
5.1 Actuators	14
5.2 Sensors	14
5.3 Power and Control	14
5.4 The Failure at the State Round	14
6 Simulation Model	15
6.1 From CAD to a Robot Description	15
6.2 Drive and Contact	15
6.3 Gripper	16
6.4 Sensor Models	16
6.5 What the Model Leaves Out	16
7 Arena Design	16
7.1 Layout	16
7.2 Floor Markings Dimensioned from the Array	17
7.3 Stations Placed for the Blind Zone	17
7.4 QR Codes and Wall Textures	18
7.5 One Source for the Dimensions	18

8	Autonomy	18
8.1	Mission and Architecture	18
8.2	Odometry	19
8.3	Mapping and Localisation	20
8.4	Navigation	20
8.5	Line Follower	21
8.6	Docking	21
8.7	Mission State Machine	22
8.8	Referee	23
9	Results	23
9.1	Method	23
9.2	Outcome of the Mission	23
9.3	Line Following	24
9.4	Localisation	25
9.5	Docking	26
9.6	The Simulated Robot	26
9.7	Summary	26
10	Verification	29
10.1	Unit Tests	29
10.2	Scoring Against Ground Truth	29
10.3	Recording	29
11	Discussion and Limitations	29
11.1	Validity of the Results	29
11.2	Known Issues	30
11.3	What I Would Do Differently	31
11.4	Future Work	31
12	Conclusion	31
	Appendices	34
A	Parameters	34
B	Running the Simulation	34

Nomenclature

Frames and mechanics (Sections 4 and 6)

x, y	robot frame with origin at the midpoint of the axle: x forward, y to the left
m, W	mass and weight of the robot
x_c, x_k	distance of the centre of mass and of the caster ahead of the axle
N_w, N_k	static load on the two wheels together and on the caster
μ	coefficient of friction
$a_{\max}$	largest acceleration that the wheels can transmit without slip
d, h	distance of the centre of mass from an edge of the support triangle, its height above the floor
r, b	wheel radius and wheel separation
ω_l, ω_r	angular speed of the left and the right wheel
v, ω	forward speed and turn rate of the robot
f	focal length of the parabolic jaw profile
α	angle between the jaw surface and the direction of travel
φ	friction angle, $\arctan \mu$
$h_c, x_{\min}$	height of the camera above the floor, nearest floor point that it sees

Line following and mission (Sections 7 to 9)

p	pitch of the infrared channels
V_i, V_{ref}	voltage of channel i , voltage on a white floor
d_i	darkness of channel i , from 0 on white to 1 on black
e	offset of the line from the centre of the array, positive to the left
k_p, k_d	proportional and derivative gain of the steering law
$\omega_{\max}$	limit of the commanded turn rate
L	distance of the array ahead of the axle
ψ	heading of the robot relative to the line
ω_n, ζ	natural frequency and damping ratio of the line-following loop
station pose	pose in front of a stem at which the navigation hands over to the line follower
marker	floor marking wider than the array: the start square, a crossbar or a drop square

Abbreviations

CAD	computer-aided design
EKF	extended Kalman filter
IMU	inertial measurement unit
IR	infrared
PD	proportional and derivative (control)
RMS	root mean square
SLAM	simultaneous localisation and mapping
URDF	Unified Robot Description Format

1 Introduction

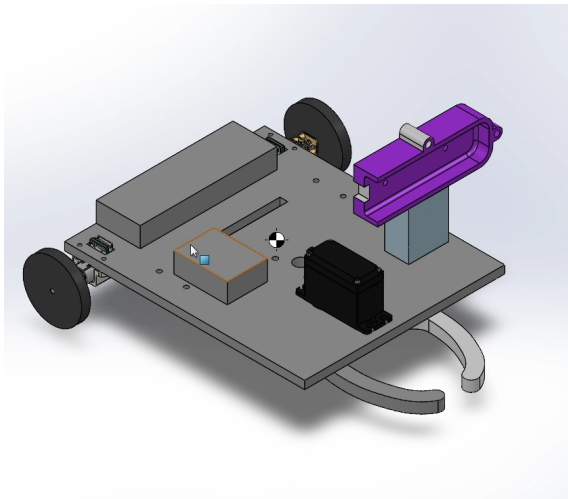
1.1 Context

Trident is a small differential-drive robot that I designed and built for a robotics task of the WorldSkills competition. The task has two parts. The robot first follows a black line on the floor. It then enters an area in which plastic coins, cylinders about 40 mm across and 15 mm high, each of one colour, have to be carried to the bin of the same colour. The first part is solved with an infrared reflectance array, the second with a camera, so the task combines line following with vision-based navigation and a simple manipulation.

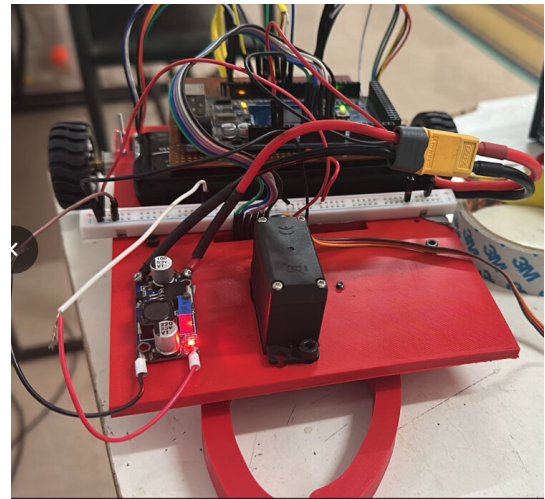
The project has passed through two versions.

1. **Version 1** was a test rig: a laser-cut acrylic base with a lidar and a depth camera, built to learn ROS 2 and Gazebo and to see the same software run on real sensors (Section 3).
2. **Version 2** is the competition robot (Fig. 1). Its chassis and gripper were designed in SolidWorks and printed, and the mechanical design and the electronics were completed in fifteen days before the state round of the competition.

The robot won the district round. In the state round it did not complete the task, because its infrared array stopped working. This report documents the design of Version 2 and a simulation of it in which the whole task is carried out autonomously.



(a) CAD model



(b) Printed robot during wiring

Figure 1. Version 2 of Trident: two driven wheels at the rear, a ball caster under the plate and a pair of geared jaws at the front. The holder at the side of the plate carries the depth camera.

1.2 Problem Statement and Objectives

A robot of this size has little room for mechanism, and a robot built in two weeks has little time for trial runs. Both facts push decisions into the design: the gripper has to work without a lift axis and without feedback, and the software has to be tested before the hardware is free for it. The work reported here therefore has two objectives:

1. to design a chassis, a drive and a gripper that move a coin of known size reliably with one servo and no sensing in the jaws;
2. to build a simulation of the robot, from the same CAD model, in which the complete mission can be run, scored against ground truth and repeated under varied conditions.

1.3 Contributions

The contributions of this report are:

- the mechanical design of the robot: its layout, drive, sensor placement and a gripper that cages a coin between two curved jaws and slides it along the floor (Section 4);
- an analysis of that design from the mass properties of the CAD model, which gives the wheel loads, the traction limit and the tipping margin, and which points to two improvements of the layout (Section 4.2);
- a simulation model of the robot for Gazebo, built on the CAD export, with models of the depth camera, the inertial sensor and the infrared array (Section 6);
- a mission arena whose floor markings are dimensioned from the pitch of the infrared array (Section 7);
- an autonomy stack that combines an existing visual SLAM system and navigation framework with a line follower, docking sequences and a mission state machine written for this robot (Section 8);
- measurements of the simulated mission over repeated runs with different coin orders and obstacle layouts (Section 9).

I am the sole author of the design, the hardware and the software. Where the software uses existing open-source packages, Section 8 states which.

1.4 Organisation

Section 2 places the work in context. Section 3 describes the two versions. Sections 4 and 5 give the mechanical design and the electronics. Sections 6 and 7 describe the simulation model and the arena, and Section 8 the autonomy stack. Section 9 reports the measurements, Section 10 the tests, and Section 11 the limitations.

2 Related Work

Differential-drive platforms. The differential drive, two independently driven wheels on one axle with a passive third support, is the standard layout of small indoor robots, and its kinematics and odometry are textbook material [1]. Frame conventions for such platforms in ROS are fixed by REP 105 [2], which this work follows.

Line following. Following a line with a row of reflectance sensors is a feedback problem in which the lateral offset of the line under the row is the measured error. A proportional and derivative law on that error is the common solution [3]. The distance of the sensor row ahead of the axle acts as a look-ahead, as it does in pure pursuit path tracking [4].

Pushing and caging. A gripper that surrounds an object without clamping it is a cage: the object can move, but not escape [5]. Moving an object by pushing it along a support surface, and the way a pushed object slides along the pusher until the contact is stable, were analysed by Mason [6]. The gripper of this robot relies on both effects in place of a lift axis.

Simulation. Gazebo [7] is the simulator most widely used with ROS. This work uses its current generation, Gazebo Fortress, with ROS 2 Humble [8]. The robot description is exported from CAD in the URDF format.

Localisation and mapping. RTAB-Map [9] is an open-source SLAM library that builds a graph of camera views, detects when a place is seen again from visual features, and can run in a localisation mode against a saved map. It accepts the colour and depth streams of a stereo depth camera such as the Intel RealSense D435 [10]. Wheel odometry and an inertial sensor are commonly fused beforehand by an extended Kalman filter; the `robot_localization` package [11] is the usual implementation. The underlying theory is covered by Thrun *et al.* [12].

Navigation. Nav2 [13] is the navigation framework of ROS 2. It combines layered costmaps [14] with a global planner, a path-tracking controller and recovery behaviours, sequenced by a behaviour tree [15]. This work uses its NavFn planner, which is Dijkstra’s algorithm [16] on the costmap, and the Regulated Pure Pursuit controller [17].

Position of this work. None of the components above is new. The contribution lies in the design of a robot around a specific task, and in the way the limits of that robot (no lift axis, a camera that cannot see the floor directly in front of it) determine how the existing components are combined with task-specific ones.

3 Design Evolution

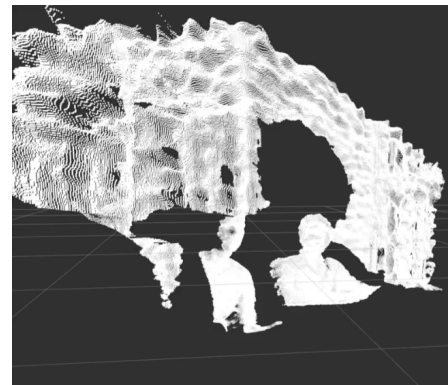
3.1 Version 1: a Test Rig

The first version was not designed for a task. It was a base cut from acrylic sheet, with a cardboard spacer and a lidar held by bolts, two driven wheels and a caster, and an Intel RealSense depth camera in a printed holder (Fig. 2a). I built it while learning ROS 2 and Gazebo, in order to see the packages that I had run in simulation work with real sensors. The same robot was modelled in Gazebo Classic, driven from a game controller and used for lidar mapping.

On the hardware, the output of the lidar and of the depth camera was verified (Fig. 2b). Version 1 never carried a payload and had no manipulator.



(a) The rig with lidar and depth camera



(b) Depth image of the camera shown as a point cloud

Figure 2. Version 1, a test rig for sensors and for ROS 2.

3.2 Version 2: the Competition Robot

Version 2 was designed for the task of Section 1, and from this point the design was deliberate. The requirements were as follows.

1. The centre of mass lies inside the support triangle of the two wheels and the caster, so that the robot stands on all three supports.

2. The robot is small. Motors with high torque do not fit a robot of this size, so N20 gearmotors drive the wheels.
3. The gripper holds a cylindrical coin about 40 mm across and 15 mm high, and it does so wherever the coin first touches it.
4. The parts are printed, with tolerances that suit the bolts and nuts in use.

The mechanical design and the electronics were completed in fifteen days. In that time the chassis was printed in more than four iterations, each testing a different tolerance, and the complete robot went through four iterations. The experience of Version 1 made this pace possible.

Figure 3 shows the robot on a test track for the line-following part of the task.

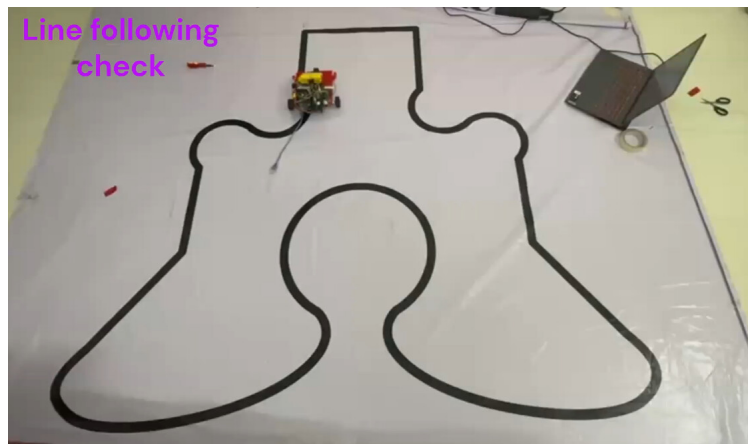


Figure 3. Version 2 following a line on a test track.

3.3 From the Hardware to the Simulation

The simulation reported here was written after the competition, from the CAD model of Version 2. It serves two purposes. It is a place in which the whole mission can be developed and repeated without the hardware, and it is a check on the design: quantities that were judged by eye during the fifteen days, such as the load that the wheels carry, can be read from the model and their consequences calculated. Section 4.2 does this.

4 Mechanical Design

The robot was modelled in SolidWorks. This section gives the layout, the mass properties and what follows from them, the drive, the gripper and the placement of the sensors. Dimensions and masses are those of the CAD model as exported for the simulation (`trident.urdf.xacro`); none of them was measured on the printed robot.

4.1 Layout

The chassis is a single printed plate of $180 \times 150 \times 6$ mm (Fig. 4). Two N20 gearmotors are fixed under its rear edge and drive two wheels of 22 mm radius, 195.5 mm apart. A ball caster under the plate, 85 mm ahead of the axle, is the third support. The servo of the gripper stands on the plate near the front edge, with its shaft pointing down through the plate, and the two jaws turn in a horizontal plane below the plate and reach forward of it. The infrared array hangs under the plate between the axle and the caster, and the depth camera sits in a holder

on a post at the left side of the plate. The battery and the control electronics occupy the rear of the plate above the axle.

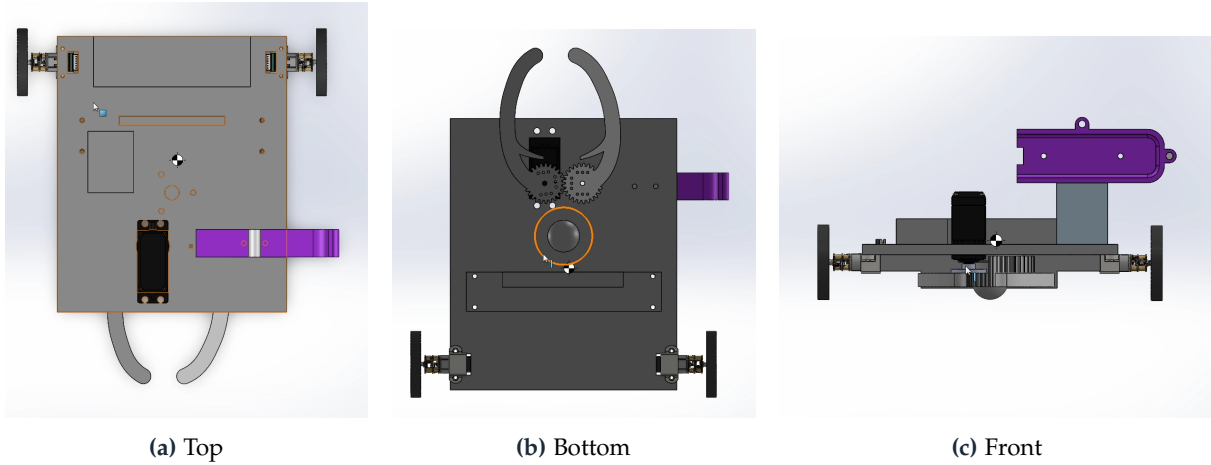


Figure 4. CAD model of Version 2. The black and white disc marks the centre of mass calculated by the CAD program. In the bottom view the caster is the dome at the centre and the two gears of the gripper are in front of it.

Table 1 lists the principal dimensions.

Table 1. Principal dimensions of the robot, from the CAD model.

Quantity	Value
Chassis plate	$180 \times 150 \times 6$ mm
Wheel radius, wheel separation	22 mm, 195.5 mm
Caster ahead of the axle	85.0 mm
Jaw pivots: ahead of the axle, apart	120.0 mm, 25.0 mm
Reach of a jaw from its pivot	90 mm
Infrared array: ahead of the axle, above the floor	48 mm, 7.4 mm
Infrared channels, pitch	8, 15 mm
Camera: above the floor, left of the centre line	85 mm, 54 mm
Total mass of the model	0.64 kg

4.2 Mass Properties and Their Consequences

The first requirement of Section 3 concerns the centre of mass. The CAD export gives a mass and a centre of mass for every part, so the requirement can be checked (Table 2). The origin is the midpoint of the axle, x points forward and y to the left.

Wheel loads. The centre of mass lies 69.3 mm ahead of the axle and the caster 85.0 mm. With $W = mg = 6.28$ N, the caster at x_k and the centre of mass at x_c , the static loads are

$$N_k = W \frac{x_c}{x_k} = 5.12 \text{ N}, \quad N_w = W \left(1 - \frac{x_c}{x_k}\right) = 1.16 \text{ N}, \quad (1)$$

where N_w is the sum over both wheels. The centre of mass is inside the support triangle, as required. It lies closer to the caster than to the axle: in the model the wheels carry 18 % of the weight and the caster 82 %. The camera with its holder, 131 g at the front of the plate, and the jaws are responsible. The CAD views of Fig. 4 agree with this: the centre-of-mass mark lies next to the caster.

Table 2. Mass budget of the model. The chassis assembly is the plate with the motors, the servo and the blocks that stand for battery and electronics in the CAD model.

Part	Mass [g]	x [mm]	y [mm]
Chassis assembly	425.6	50.2	−5.5
Depth camera (D435)	72.0	129.5	54.3
Camera holder	58.8	116.2	58.1
Jaws (two)	34.8	146.4	0.3
Wheels (two)	20.2	0.0	0.0
Infrared array	20.0	49.7	0.0
Caster	8.9	85.0	0.0
Robot	640.3	69.3	7.8

Traction limit. The wheel load bounds the force that the wheels can transmit. With a coefficient of friction μ between tyre and floor,

$$a_{\max} = \frac{\mu N_w}{m} = \mu g \left(1 - \frac{x_c}{x_k}\right) = 1.8 \mu \text{ m/s}^2, \quad (2)$$

so that even with $\mu = 1$ the robot cannot accelerate or brake at more than 1.8 m/s^2 without slip. This limit is visible in the simulation. With no limit on the deceleration, the wheels of the simulated robot stopped at once on a stop command and the robot skidded and turned; the drive is now limited to 1.0 m/s^2 (Section 6.2). The force needed to slide a coin, by contrast, is small: a coin of 23 g with a friction coefficient of 0.35 resists with 0.08 N.

Tipping margin. On three supports the robot tips about an edge of the support triangle (Fig. 5). The triangle is narrow at the caster, and the centre of mass is both far forward and, because of the camera, 7.8 mm to the left. Its distance to the left edge of the triangle is $d = 6.7 \text{ mm}$ and to the right edge 16.9 mm. With the centre of mass at a height $h = 42 \text{ mm}$ above the floor, a horizontal acceleration a directed across the left edge tips the robot when $a > g d/h = 1.6 \text{ m/s}^2$. A braking deceleration acts along x , at an angle to that edge, and tips the robot above 2.1 m/s^2 . Both values are above the 1.0 m/s^2 that the drive is allowed, but the margin is a factor of two and not more.

What this means for the design. The layout meets its requirement, and the analysis shows where it can be improved. Two changes would raise the wheel load and the tipping margin: moving the caster forward, towards the gripper pivots, which widens the triangle around the centre of mass and transfers load to the wheels; and mounting the camera on the centre line, which removes the lateral offset. Two reservations apply. The masses are CAD values, and the blocks that stand for the battery and the electronics need not carry their true mass, so the real centre of mass may lie further back than Table 2 says. And the printed robot has not been weighed on its three supports, which would settle the question in a few minutes.

4.3 Drive

The robot is a differential drive. With wheel radius r , wheel separation b and wheel speeds ω_l , ω_r , the forward speed and the turn rate are

$$v = \frac{r}{2}(\omega_r + \omega_l), \quad \omega = \frac{r}{b}(\omega_r - \omega_l). \quad (3)$$

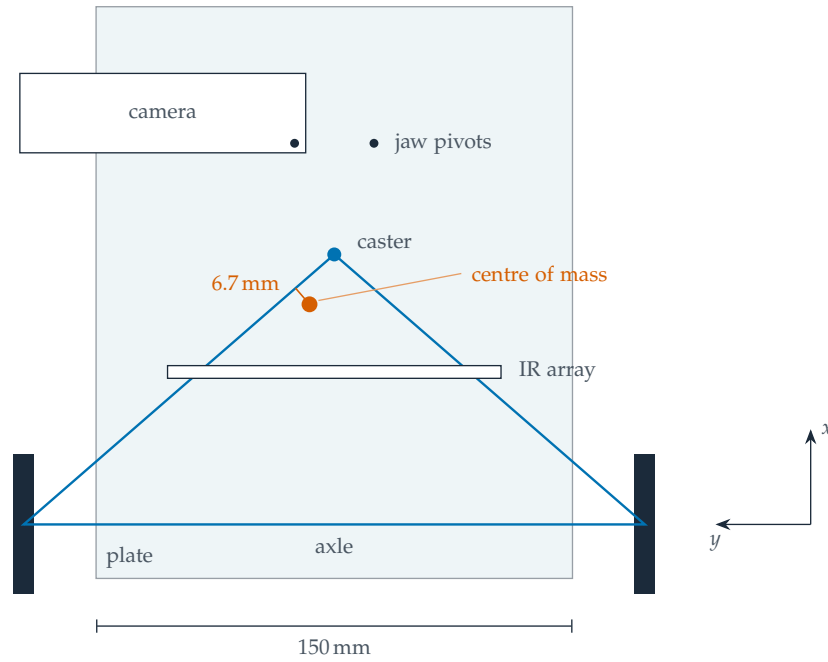


Figure 5. Support triangle of the two wheels and the caster, seen from above, to scale. The centre of mass of the model lies inside it, 16 mm behind the caster and 6.7 mm from the left edge.

N20 gearmotors were chosen because motors of higher torque do not fit under a plate of this size. The simulation assumes a variant with a no-load speed of 300 rpm at 6 V and a stall torque of 0.05 N m at the wheel; both are placeholders in the model and have to be replaced by the data of the motors fitted. On these values the robot has a top speed of $r\omega = 0.69$ m/s, and the mission uses at most 0.20 m/s, less than a third of it. The two motors together can push with 4.5 N at stall, about four times the 1.16 N that the wheels can transmit by Eq. (1). The drive is therefore limited by traction and not by torque, which is a further reason to move load onto the wheels.

The wheels are narrow, and the caster is a ball that slides. In the simulation the caster is given no friction, so that turning on the spot is resisted only by the wheels.

4.4 Gripper

4.4.1 Caging in Place of Lifting

The robot is light, and a coin of unknown mass held out in front of the caster could tip it forward. The gripper therefore does not lift. Two jaws close around the coin on the floor, and the robot slides the coin to its destination. The jaws need not clamp: it is enough that the coin cannot leave the space between them. This removes the lift axis and its actuator, and it removes the need to know the mass of the coin. It also sets a condition on the floor, which must be flat enough for a coin 15 mm high to slide; on the printed robot the grip held on mildly uneven ground.

The jaws are 8 mm thick and their mid-plane is 12 mm above the floor, so they meet the upper half of the rim of a coin.

4.4.2 Jaw Profile

The jaws were drawn with a parabolic profile (Fig. 6a). The intent is that a coin which meets a jaw anywhere along its length is led inward, to the region around the focus of the parabola, where the closed jaws hold it. The robot therefore does not have to arrive exactly in line with

the coin.

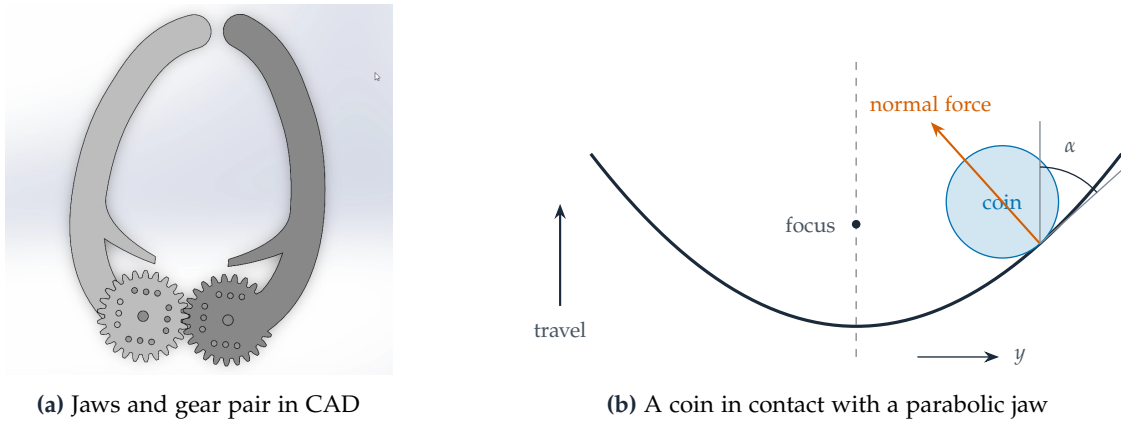


Figure 6. The gripper. A coin touched by a jaw is pushed along the normal of the profile, which has an inward component everywhere except at the vertex.

The mechanics of this can be stated simply (Fig. 6b). Let the profile be the parabola $y^2 = 4fx$ with focal length f , opening in the direction of travel, and let the robot advance onto a coin that lies at a lateral offset y from the axis. The jaw meets the coin with a surface inclined at an angle α to the direction of travel, where

$$\tan \alpha = \frac{dy}{dx} = \frac{2f}{y}. \quad (4)$$

The contact force lies within the friction angle $\varphi = \arctan \mu$ of the surface normal. The coin slides along the jaw, towards the axis, as long as the direction of travel lies outside that friction cone, that is, as long as $90^\circ - \alpha > \varphi$. With $\tan \alpha = 2f/y$ this gives

$$|y| > 2\mu f. \quad (5)$$

A coin that meets the jaw further from the axis than $2\mu f$ is carried inward; inside that band friction is able to hold it where it is. Two conclusions follow. The centring improves with a short focal length and a slippery jaw, and the tips of the jaws, where α is small, are where it works best, which is where a misaligned approach makes first contact. For the friction coefficient of 0.35 used for the coin in the simulation, the band is $0.7f$ on either side of the axis. I have not measured the focal length of the printed jaws, and Eq. (5) is given as the design rule and not as a verified figure. When the jaws close, the coin is centred in any case, since the closed jaws leave it little room.

4.4.3 Release Ridges

Each jaw carries a ridge on its inner side, near the pivot (Fig. 6a). When the jaws open, the ridges swing forward and push the coin out of the throat. Without them a coin could stay against the gears as the robot backs away and be drawn back with it. With them the coin is left where it was released. This detail decided whether a coin placed in a bin stayed there.

4.4.4 Actuation

One MG995 servo drives the gripper. Its shaft turns one jaw directly, and a pair of involute gears with equal numbers of teeth turns the other jaw by the same angle in the opposite sense. The pivots are 25 mm apart, so each gear has a pitch diameter of 25 mm. An involute profile

keeps the velocity ratio constant through the mesh [18], and equal gears make the motion of the two jaws symmetrical without any linkage.

At 4.8 V the servo is rated at 8.5 kg cm, which is 0.83 N m, and at 60° in 0.20 s, which is 5.2 rad/s. At the tip of a jaw, 90 mm from the pivot, the stall torque corresponds to a force of 9 N, two orders of magnitude above the 0.08 N needed to slide a coin. The jaws travel between an open position at -1.0 rad and a closed position at 0.09 rad of the driven jaw; the jaws touch one another at about 0.096 rad. The stroke of 1.09 rad takes 0.21 s at the rated speed, and the mission software allows one second for it. In the pose in which the assembly was exported, the second jaw is 7.25° away from the mirror image of the first, so the jaws meet slightly to one side of the centre line. This is an error of assembly in the CAD model that the simulation reproduces.

4.5 Sensor Placement

4.5.1 Depth Camera

The camera looks forward from a height of 85 mm. Its depth image has a vertical field of view of 58°, so the lowest ray leaves the lens 29° below the horizontal and reaches the floor at a distance

$$x_{\min} = \frac{h_c}{\tan 29^\circ} = \frac{85 \text{ mm}}{0.554} \approx 0.15 \text{ m} \quad (6)$$

ahead of the lens (Fig. 7). Depth is in addition not reported below a range of 0.2 m. A caged coin lies about 4 cm ahead of the lens. The camera therefore never sees the coin it is about to pick, nor the last 15 cm of floor in front of the robot. This is the most important constraint that the mechanical design places on the software, and Section 8.6 describes how the infrared array takes over where the camera ends.

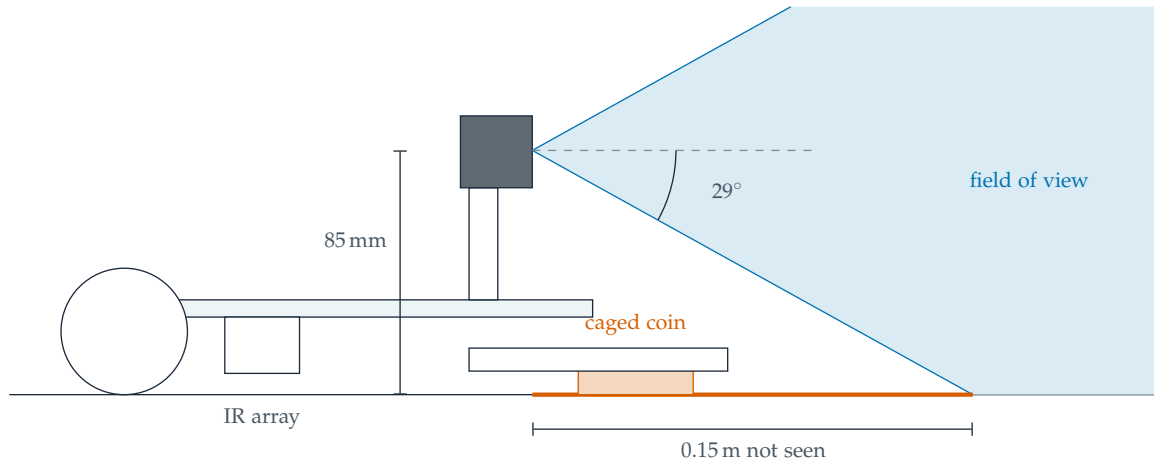


Figure 7. Side view of the front of the robot, to scale. The floor closer than 0.15 m to the lens is below the field of view of the depth camera, and so is a coin between the jaws.

The holder places the camera 54 mm to the left of the centre line, beside the servo. The offset is known to the software through the robot description, so it does not bias the map, but it is the cause of the lateral shift of the centre of mass found in Section 4.2.

4.5.2 Infrared Array

The array has eight reflectance channels in a row across the robot, 15 mm apart, which gives it a span of 105 mm between the outer channels. Its sensing face is 7.4 mm above the floor. Two properties of its position matter later. It is 48 mm ahead of the axle, and this distance acts as

a look-ahead that damps the line follower (Section 8.5). And it is 72 mm behind the gripper pivots, so that a marking on the floor under the array fixes the position of the jaws relative to that marking; the pick and drop markings of the arena are placed accordingly (Section 7).

4.6 Manufacture

The chassis, the jaws, the gears and the camera holder are printed in PLA. Holes and pockets were toleranced for the bolts and nuts in use, and the chassis was printed in more than four iterations to find clearances that the printer reproduced. All bolts have hexagon socket heads, which take a higher tightening torque from a key than a slotted or cross head does from a screwdriver and are standard in prototype work.

5 Electronics and Sensing

This section is short by intent. It records what the design notes, the photographs of the robot and the simulation model establish about the electronics, and nothing beyond that. No circuit diagram or parts list of the competition robot was available when this report was written.

5.1 Actuators

Two N20 gearmotors with encoder boards drive the wheels, and one MG995 servo drives the gripper (Sections 4.3 and 4.4). The motors were selected for their size.

5.2 Sensors

Infrared array. Eight analog reflectance channels at a pitch of 15 mm, read as voltages between 0 V on a black surface and 5 V on a white one. The array serves the line following and, in the simulated mission, the final approach to every station.

Depth camera. An Intel RealSense D435, which provides a colour image and a depth image. In the simulation it is the only sensor used for mapping, localisation, obstacle detection and the reading of QR codes.

Wheel encoders. One on each motor, the source of wheel odometry.

Inertial sensor. The simulation model carries an inertial sensor of the MPU6050 class, flat on the plate between the wheels, whose yaw rate is fused with the wheel odometry (Section 8.2).

5.3 Power and Control

The photograph of the robot (Fig. 1b) shows a battery with an XT60 connector at the rear of the plate, a step-down converter module, and a microcontroller board on a prototyping board with a solderless breadboard for the signal wiring. Both the electronics and the mechanical design were checked extensively before the competition.

5.4 The Failure at the State Round

In the state round the infrared array stopped working and the robot could not follow the line, so the task was not completed. The cause was not established afterwards, and I do not speculate on it here. Two consequences are drawn for the design. The line following depended on a single sensor with no fallback. And the photograph shows signal wiring on a solderless breadboard, which is not a reliable connection on a moving robot. A soldered

or connectorised harness, and a check of all eight channels against known black and white references at power-up, are the corrections proposed in Section 11.

6 Simulation Model

The simulation runs in Gazebo Fortress with ROS 2 Humble, inside a Docker container that is not privileged and receives only the graphics card, the display socket and the joystick devices of the host. One script builds the image, starts the container and launches every configuration described below.

6.1 From CAD to a Robot Description

The CAD assembly was exported from SolidWorks as a URDF description with one mesh per link. An export of this kind is a starting point and not a usable model. The description used here is maintained by hand on top of the export, and it differs from it in the following ways.

- A frame `base_footprint` on the floor was added below `base_link`, which stays at the midpoint of the axle, as REP 105 prescribes [2].
- The wheel axes were turned so that a positive rotation drives the robot forward.
- The right wheel was made coaxial with the left one. The export had placed it 0.5 mm higher.
- The limits, effort, velocity and damping of the gripper joints, all zero in the export, were filled in from the data of the servo.
- The body of the depth camera, which was not part of the CAD assembly, was added as a box of $90 \times 25 \times 25$ mm and 72 g, together with the depth and optical frames that the camera driver publishes.
- The infrared array was given a mass. The export had assigned none, and the simulator rejects a link without mass.

Collision geometry. The visual mesh of the chassis has 277 864 triangles. For contact it is replaced by primitives: a box for the plate, a box for the volume above it, a cylinder for each wheel and a sphere for the caster. The jaws and the camera holder keep their meshes, of about 2 000 and 700 triangles, because the shape of the jaws is what the gripper depends on.

The caster. In the CAD model the bottom of the caster dome is 0.74 mm above the plane on which the wheels stand, so the robot would rock on it. The collision sphere of the model is sized to touch the floor exactly. The description records this as a correction to be made on the physical part.

6.2 Drive and Contact

The wheels are driven by the differential-drive system of Gazebo, which converts a velocity command into wheel speeds by Eq. (3) and publishes wheel odometry at 30 Hz. The physics step is 1 ms. The friction coefficients are 1.0 for the wheels, 0 for the caster, 0.8 for the jaws and 0.35 for the coins.

The drive is limited to 1.0 m/s^2 and 8 rad/s^2 , in acceleration and in braking. The braking limit was added after the first runs: without it the wheels stopped at once, and the robot skidded and turned. Section 4.2 gives the reason, a wheel load of 18 % of the weight, and the limit of Eq. (2) that the chosen value respects.

6.3 Gripper

The two jaw joints are position-controlled at the rated speed of the servo. Gazebo Fortress ignores the mimic relation by which a URDF couples one joint to another, so the gear pair is reproduced in software: a node receives the commanded angle of the driven jaw, clamps it to the joint limits, rejects values that are not finite, and sends the angle to one jaw and its negative to the other.

6.4 Sensor Models

Depth camera. The camera is simulated as an RGB-D sensor at the position of the left infrared imager of the D435, with the depth field of view of the real camera, $87^\circ \times 58^\circ$, an image of 848×480 pixels, a depth range from 0.2 to 10 m and Gaussian image noise with a standard deviation of 0.007. The topic names match those of the driver of the real camera, so that the same nodes can run on the hardware. The point cloud is computed in ROS from the depth image, because the cloud that the simulator produces is expressed in a different axis convention under the same frame name.

Infrared array. Gazebo has no reflectance sensor. Each channel is therefore modelled by a camera of 8×8 pixels that looks straight down from the sensing face and covers a spot about 8 mm across on the floor, comparable to the spot of a reflectance sensor at that height. A node averages the brightness of each small image, scales it to a voltage between 0 and 5 V, adds Gaussian noise, and publishes the eight voltages at 20 Hz. A channel that lies over the edge of a line therefore returns an intermediate voltage, as the real sensor does, and it is this that allows the position of a line to be interpolated between channels (Section 8.5). The arena is lit evenly and without shadows, so that the readings do not depend on the heading of the robot. In the mission runs the noise has a standard deviation of 0.05 V.

Inertial sensor. The inertial sensor is sampled at 100 Hz, with Gaussian noise of 0.002 rad/s on the angular rate and 0.02 m/s^2 on the acceleration.

6.5 What the Model Leaves Out

The motors are ideal velocity sources within the acceleration limits: there is no model of the gear train, of backlash or of the motor driver. The servo is an ideal position source. The encoders have no quantisation. The infrared model has no dependence on ambient light or on the height of the array above the floor. The floor is flat. These omissions bound what the results of Section 9 can say about the hardware, and Section 11 returns to them.

7 Arena Design

The arena is a model of the task and does not reproduce the competition field. It keeps the structure of the task: a line to follow, then a room in which coins are collected at one wall and delivered, by colour, to another. It was designed together with the robot model, and most of its dimensions follow from two numbers of Section 4: the pitch of the infrared array and the blind zone of the camera.

7.1 Layout

The field measures $3.6 \times 2.4 \text{ m}$ and is enclosed by walls 0.3 m high (Fig. 8). A partition with a doorway 0.8 m wide divides it into an entry lane and a room. The robot starts in the lane on a black line, which leads through an S-bend of two arcs of 0.6 m radius, through the doorway,

to a black start square. Three pick bays lie along the left wall and three drop zones along the front wall. Up to three boxes of $100 \times 100 \times 150$ mm stand in the room as obstacles.

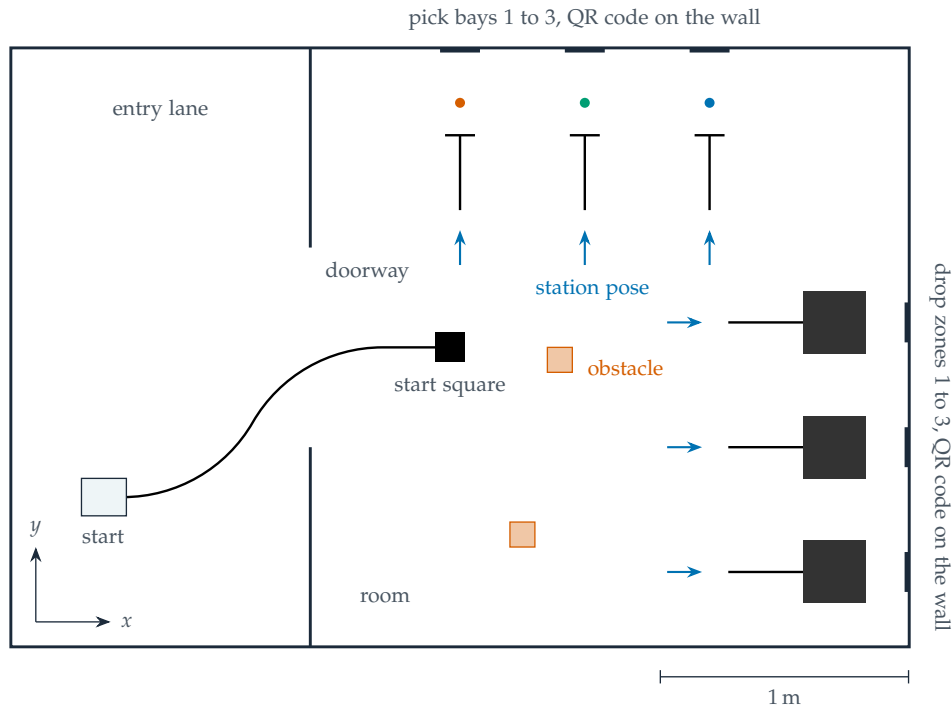


Figure 8. Plan of the arena, to scale, with obstacle preset a. The arrows are the station poses at which the navigation hands over to the line follower.

7.2 Floor Markings Dimensioned from the Array

The infrared array has eight channels at a pitch of $p = 15$ mm and a span of 105 mm. Three kinds of marking are derived from these two numbers (Fig. 9).

Lines are 12 mm wide. A line narrower than one pitch can never cover two channels completely. At most two neighbouring channels respond to it, each partially, and the ratio of their readings gives the position of the line between them.

Markers are 120 mm wide. The start square and the crossbars of the pick bays are wider than the whole array, so that all eight channels go dark together when the array reaches one. Nothing else on the floor can produce that pattern, and a line, which darkens two channels at most, cannot be mistaken for it.

Drop squares are 250 mm wide. The robot releases a coin when the array reaches the near edge of the square. The coin then lies 0.13 m further in, which is 5 mm from the centre of a square of this size.

7.3 Stations Placed for the Blind Zone

The camera does not see the last 0.15 m of floor (Eq. (6)). A pick bay is therefore a T on the floor: a stem 0.30 m long that leads to a crossbar, with the coin 0.13 m beyond the crossbar. The distance of 0.13 m is the distance from the array to the space between the open jaws, so that a robot which stops with its array on the crossbar has the coin between its jaws. A drop zone is a stem of the same length that ends in the drop square.

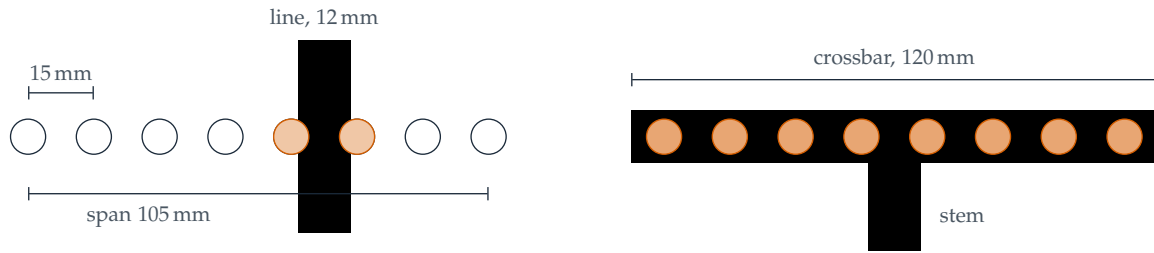


Figure 9. The infrared array over a line and over a marker, to scale. The circles are the spots of the eight channels. A line darkens two channels at most, a marker all eight.

The navigation brings the robot to a station pose on the extension of the stem and short of its foot, 0.15 m for the bays and 0.175 m for the drop zones, and facing along it. From there the line follower drives straight until the array finds the stem (Section 8.6). The length of the stem is what absorbs the error of the navigation: a lateral error of a few centimetres at the foot is steered out before the crossbar is reached.

7.4 QR Codes and Wall Textures

A QR code of 160 mm, centred 0.11 m above the floor, is mounted on the wall behind every station. It encodes the name of a colour. At a pick bay it names the colour of the coin, and at a drop zone the colour that the zone accepts. The codes are generated with the highest level of error correction and a wide quiet zone, so that they remain readable when seen at an angle. The assignment of colours to bays and to zones is a launch argument, which makes each of the 36 combinations of pick order and drop order a different mission.

Plain walls have no visual features, and a visual SLAM system cannot localise against them. The walls therefore carry generated textures with dense, irregular detail. They exist only for this purpose.

7.5 One Source for the Dimensions

The world is written as a parametrised file whose dimensions are defined once at its head. Two other files repeat a few of them: the list of station poses read by the mission, and the positions of the drop squares read by the referee. Both name the parameters that they mirror. A change of the arena therefore requires edits in three places, which is a known weakness of the present arrangement.

8 Autonomy

8.1 Mission and Architecture

The mission has four parts.

1. Follow the line from the start to the start square.
2. Visit the three drop zones and read at each the colour that it accepts.
3. For each pick bay in turn: approach it, read the colour of its coin, cage the coin, carry it to the drop zone of that colour and release it.
4. Throughout, drive around obstacle boxes that are not in the map.

The software is a set of ROS 2 packages (Fig. 10). Odometry, mapping, localisation, path planning and path tracking are taken from existing open-source packages and configured for

this robot. The line follower, the docking sequences, the mission state machine and the referee are written for it (Table 3). The packages for navigation and for the mission have no knowledge that they run in a simulator: everything that exists only in simulation, including the emulation of the infrared array and the referee, is kept in a separate package.

Table 3. Origin of the components of the autonomy stack.

Function	Solution	Source
Odometry	filter of wheel speed and yaw rate	robot_localization [11]
Mapping, localisation	RGB-D graph SLAM	RTAB-Map [9]
Planning and tracking	NavFn, Regulated Pure Pursuit	Nav2 [13]
QR decoding	bar code reader on the colour image	ZBar [19]
Line following	PD steering on the line offset	this work
Pick and drop sequences	infrared array and odometry	this work
Mission sequencing	state machine	this work
Scoring	referee on simulator ground truth	this work

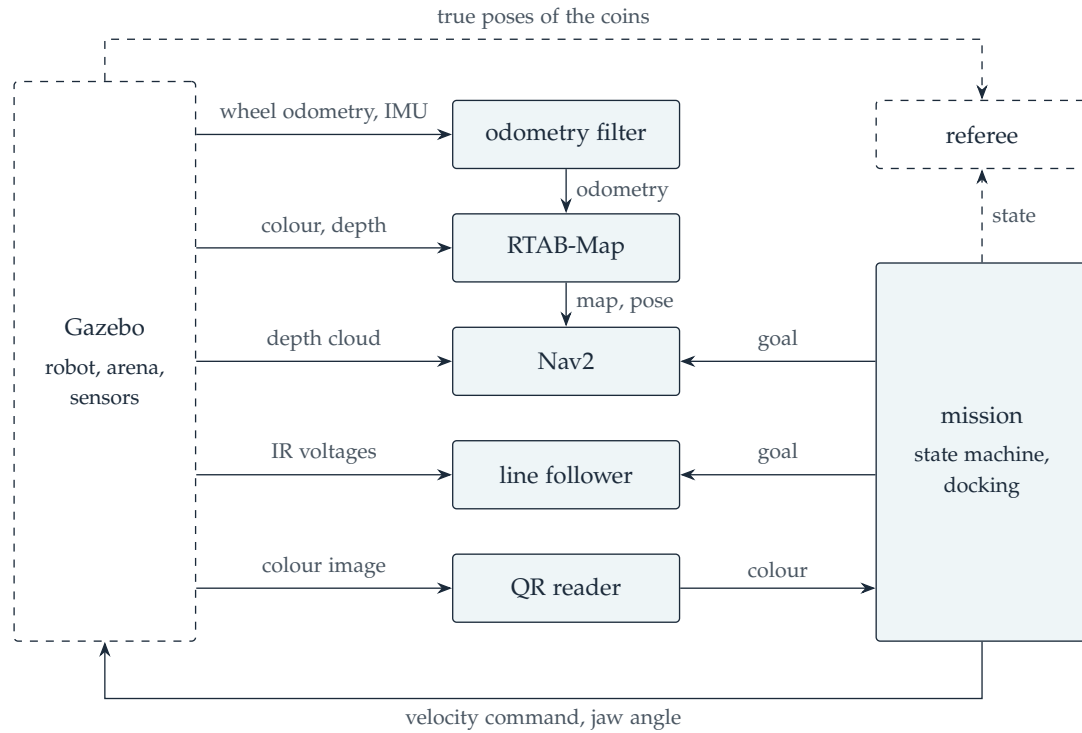


Figure 10. Software architecture. Solid boxes run unchanged on a real robot; dashed boxes exist only in simulation. Velocity commands are issued by Nav2, by the line follower or by the mission itself, one at a time, because the mission waits for each step to finish before it starts the next.

8.2 Odometry

An extended Kalman filter in planar mode fuses the forward speed from the wheel encoders with the yaw rate from the gyroscope, at 30 Hz. The yaw rate of the wheel odometry is deliberately left out: when the robot turns on the spot the wheels slip, and the gyroscope is the better source. The filter publishes the transform from the odometry frame to the robot.

8.3 Mapping and Localisation

Building the map. The map is built once, in the empty arena. A survey node sends a fixed route of seven stops to the navigation and turns the robot through a full circle at each stop, so that the camera sees every wall, while RTAB-Map adds a node to its graph every 10 cm or 0.2 rad of travel. The route ends at the start pose and closes the loop there. Because the survey starts at the start pose, the map frame coincides with the frame of the arena, and the station poses can be written in arena coordinates. The map is constrained to the plane, and its occupancy grid has cells of 2 cm. Points below 3 cm are treated as floor, which keeps coins out of the map.

Localising in it. A mission run loads the whole map and starts at its origin, where the robot stands. Each time RTAB-Map recognises a view, twice a second at most, it corrects the transform from the map to the odometry frame. Between corrections the pose is carried by the odometry filter.

Look-alike views. The arena contains the same QR code in two places, once at a pick bay and once at a drop zone. Seen from close by, a code fills much of the image, and a match of the features of the code alone was enough for RTAB-Map to conclude that the robot stood in front of the other wall. The pose then jumped across the room. The remedy is a condition on how the matched features are spread over the image: a view of the room matches across the whole width of the walls, a view of a code only within the code. The walls fill a low band of the image, and during development real views scored between 0.004 and 0.014 on this measure and look-alikes below 0.0025; the threshold is set at 0.005. The choice is a compromise. A threshold at the lower end of the real range rejects some true matches, and one set higher rejects them all, after which the robot runs on odometry alone. In addition, every new fix is checked against the last fifty views and the odometry between them.

8.4 Navigation

Between stations the robot is driven by Nav2. The global planner is NavFn on a costmap of 2 cm cells built from the saved map, and the path is planned again once a second, which is what steers the robot around boxes that are not in the map. The path is tracked by the Regulated Pure Pursuit controller at 0.15 m/s with a look-ahead of 0.20 m, and the robot is turned on the spot when the path departs from its heading by more than 0.5 rad. A goal counts as reached within 2 cm and 0.05 rad.

Obstacles enter the costmaps from the depth image, reduced to every fourth pixel and to cells of 2 cm. Points between 3 cm and 0.40 m above the floor mark obstacles, and rays through free space clear them. The robot is treated as a disc of 0.13 m radius, and obstacles are inflated by 0.25 m.

Three properties of the robot shape the configuration.

- The camera looks forward only. The controller is not allowed to reverse, and the recovery that backs the robot up is removed from the behaviour tree. The remaining recoveries clear the costmaps, turn the robot through a quarter circle to look around, and wait.
- Low objects do not reach the costmap. This is intended for coins and sensor noise, and it means that an obstacle lower than 3 cm would not be avoided.
- Turns on the spot are slow, 0.3 to 0.6 rad/s, so that successive camera views overlap and the localisation can follow.

8.5 Line Follower

Line position. Each voltage V_i is converted to a darkness $d_i = 1 - V_i/V_{\text{ref}}$ between 0 on white and 1 on black. Readings below 0.15 are set to zero, so that noise on the white floor carries no weight. The offset of the line from the centre of the array is the weighted mean

$$e = p \frac{\sum_i d_i (i - 3.5)}{\sum_i d_i}, \quad i = 0, \dots, 7, \quad (7)$$

with p the pitch. It is positive when the line is to the left. Because a line narrower than the pitch darkens two neighbouring channels partially (Section 7.2), e varies continuously as the line moves across the array, and the resolution is much finer than the pitch.

Steering. The robot drives at a constant speed v and steers with

$$\omega = \text{sat}_{\omega_{\max}}(k_p e + k_d \dot{e}), \quad (8)$$

with $k_p = 45 \text{ rad/s per metre}$, $k_d = 2 \text{ rad/s per metre per second}$ and $\omega_{\max} = 1.5 \text{ rad/s}$, at 20 Hz. If the line leaves the array, the robot turns at 0.6 rad/s towards the side on which the line was last seen. If no line has been seen yet, it drives straight, so that a goal may start a little short of a stem.

Why the position of the array matters. Let the array be a distance L ahead of the axle, let ψ be the heading of the robot relative to a straight line, and let y be the lateral position of the array relative to the line, so that $e = -y$. For small angles,

$$\dot{y} = v \psi + L \omega, \quad \dot{\psi} = \omega. \quad (9)$$

With the proportional part of Eq. (8), $\omega = -k_p y$, this gives

$$\ddot{y} + L k_p \dot{y} + v k_p y = 0, \quad (10)$$

a second-order system with natural frequency $\omega_n = \sqrt{v k_p}$ and damping ratio $\zeta = \frac{1}{2} L \sqrt{k_p / v}$. The damping comes entirely from L : an array on the axle ($L = 0$) would give an undamped oscillation under proportional control, whatever the gain. With $L = 48 \text{ mm}$ and $k_p = 45$, the damping ratio is 0.51 at the entry speed of 0.10 m/s and 0.72 at the docking speed of 0.05 m/s . The derivative term raises the first value slightly, to 0.53. The mechanical decision to hang the array ahead of the axle is thus what makes a simple controller sufficient, and driving the stems slowly is what makes the approach to a marker well damped.

Markers. A marker is reported when at least six channels have a darkness of 0.5 or more in two successive samples. A line darkens two channels at most, so only a square or a crossbar can do this, and the requirement of two samples rejects a single noisy reading. Two sample periods at 20 Hz are 5 mm of travel at the docking speed.

The follower is offered as an action. A goal names a maximum distance and a speed, and the result states whether a marker was reached and how far the robot drove.

8.6 Docking

A station is approached in two stages (Fig. 11). Nav2 brings the robot to the station pose. The line follower then drives straight until the array finds the stem, steers onto it, and stops at the marker. Everything after that is a timed movement of the jaws or a straight move measured by odometry.

Pick. Open the jaws. Follow the stem to the crossbar, at which point the coin is between the jaws. Read the QR code on the wall. Close the jaws. Back out by the distance just driven.

Drop. Follow the stem until the array enters the square. Open the jaws, which pushes the coin clear. Back out, then close the jaws.

Read. Follow the stem to its marker, read the QR code and back out. This is used for the survey of the drop zones.

The robot backs out along the stem that it has just driven and by the distance that it measured, which is the only reverse movement in the mission. It needs no sensing behind the robot because the path has just been traversed.

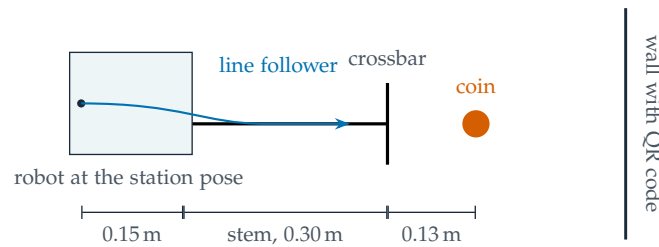


Figure 11. Approach to a pick bay, to scale. The navigation leaves the robot at the station pose with an error of a few centimetres; the line follower removes it along the stem and stops the array on the crossbar, with the coin between the open jaws.

QR codes are read at the station only. The decoder reports the text of a code but not where the code is in the image. A code is therefore trusted only when the robot stands at the marker in front of it, where no neighbouring code is in view, and only after three identical decodes in a row. Decodes received before the robot arrived are discarded, so that a code passed on the way is never taken for the code of the station.

8.7 Mission State Machine

The mission is a sequence of blocking steps, and the state machine only decides what happens next (Fig. 12). Its failure handling is deliberately plain.

- If the start square is not found within 3 m of travel, the mission fails.
- If a marker is not found at a station, the robot backs out and tries the station once more.
- A drop zone whose code cannot be read is left out of the table of colours.
- A coin whose code cannot be read is left in its bay, with the jaws still open, since without a colour there is nowhere to take it.
- A coin whose colour has no known drop zone is set down where the robot stands.

The mission reports success only when all three coins have been delivered.

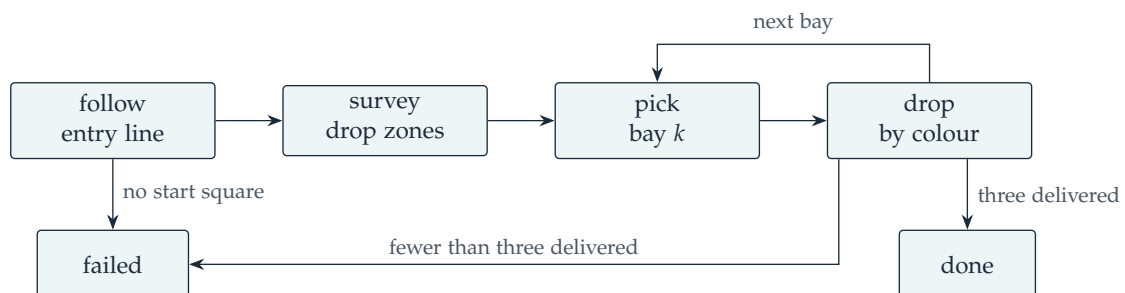


Figure 12. States of the mission.

8.8 Referee

A run is scored by a referee that reads the true positions of the coins from the simulator. A coin is home when its centre lies inside the drop square whose QR code names its colour. The referee has no connection to the robot other than the state of the mission, which tells it when to give its verdict, and nothing on the robot may read the ground truth. A run passes when all three coins are home at the end.

9 Results

All results in this section were obtained in the simulation of Section 6, in the arena of Section 7, with the parameters of Section A. Times are simulated time. Positions called true are the ground truth of the simulator.

9.1 Method

The mission was run twelve times without a display: three runs for each of the four obstacle layouts (none and the presets a, b and c), with a different combination of pick order and drop order in almost every run (Table 4). All runs used the same map, built once in the empty arena, and the same start pose. The noise on the infrared readings was 0.05 V. Each run was recorded as described in Section 10 and scored by the referee.

9.2 Outcome of the Mission

All twelve runs passed: in every run the referee found all three coins inside the drop squares of their colours, 36 coins of 36. No station needed a second attempt, no QR code was missed, and the navigation entered no recovery behaviour in any run.

Table 4. The twelve mission runs. Pick and drop give the colours of bays 1 to 3 and of drop zones 1 to 3. Map error is the RMS distance between the pose in the map and the true pose; line error is the RMS offset of the array from the entry line.

Run	Pick	Drop	Obstacles	Coins home	Time [s]	Path [m]	Map error [mm]
1	RGB	BRG	a	3 of 3	301	18.4	8.4
2	RGB	BRG	none	3 of 3	300	18.0	8.8
3	GBR	RGB	b	3 of 3	307	18.7	10.7
4	BRG	GRB	c	3 of 3	305	17.9	10.6
5	RBG	RGB	none	3 of 3	299	17.7	11.8
6	GRB	BGR	a	3 of 3	299	18.4	11.8
7	BGR	RBG	b	3 of 3	304	18.6	9.2
8	RGB	GBR	c	3 of 3	292	17.6	10.0
9	GBR	BRG	none	3 of 3	298	17.5	9.7
10	BRG	RGB	a	3 of 3	295	17.8	9.8
11	RBG	GRB	b	3 of 3	300	18.5	9.7
12	GRB	RBG	c	3 of 3	298	17.6	8.2

A mission takes 300 s on average, between 293 and 307 s (Fig. 14): 19 s on the entry line, 83 s for the survey of the three drop zones and 198 s for the three coins, or 66 s per coin. The robot covers 18.1 m, which is an average of 0.06 m/s; most of the time goes into turning on the spot,

the slow approach along the stems and the waiting for the jaws and for the QR codes. The obstacles cost little. The mean time is 299 s without obstacles and 304 s with the three boxes of preset b, and the paths differ by less than a metre. Figure 13 shows the true path of one run.

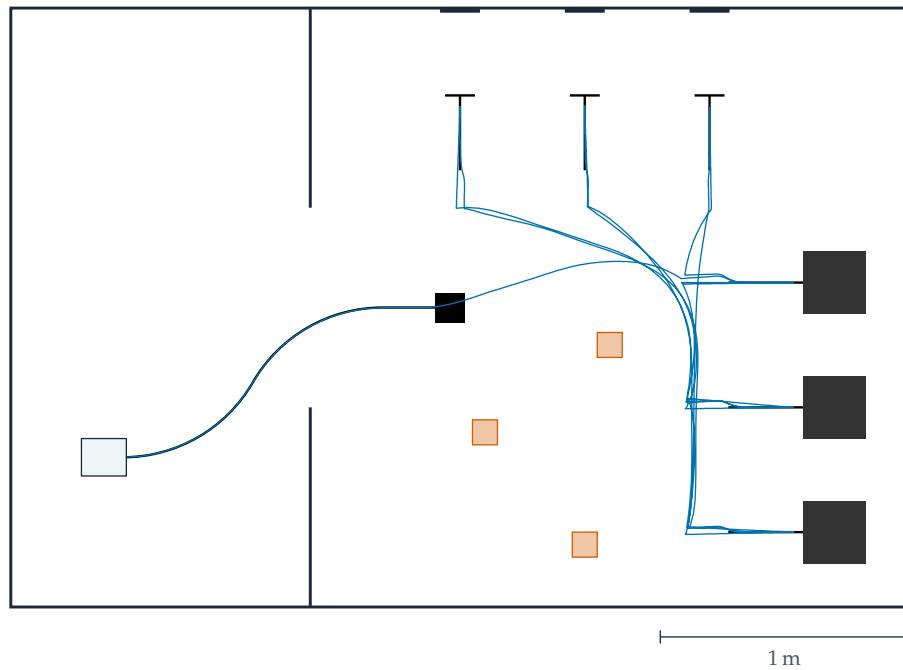


Figure 13. True path of the robot in run 3 (obstacle preset b, three boxes). The path follows the entry line, visits the three drop zones, and then alternates between the pick bays and the drop zones, passing the boxes on whichever side the planner finds free.

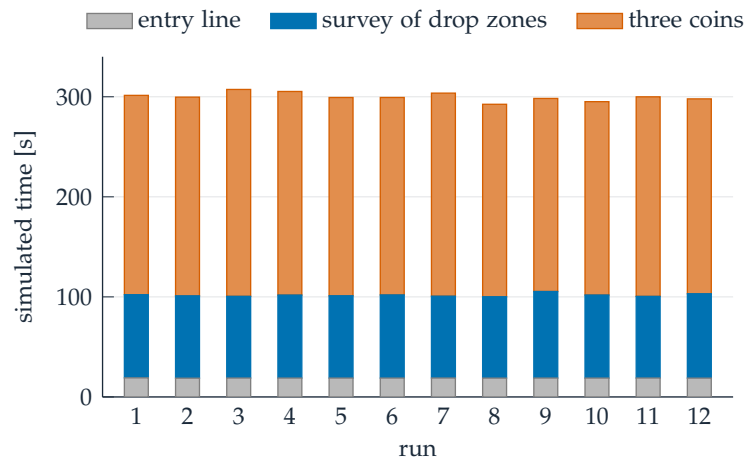


Figure 14. Duration of the three parts of the mission in each run.

9.3 Line Following

On the entry line of the twelve missions, the centre of the array stayed within 6.0 mm of the centre of the line, with an RMS offset of 1.6 mm (1.4 to 2.1 mm over the runs). The line is 12 mm wide and the channels are 15 mm apart, so the offset is a tenth of the pitch: the interpolation between partially covered channels (Eq. (7)) works as intended. In every run the array stopped on the start square.

The line follower was also run on its own on the entry line, once for each of several noise

levels and speeds (Table 5 and Fig. 15). Two results stand out. The follower is insensitive to noise up to 0.2 V, four times the level used in the missions, because readings below the floor threshold carry no weight. At 0.5 V noise crosses that threshold and the offset grows fourfold, and at 1.0 V it reaches 46 mm, close to the half span of the array, although the line was not lost. And the offset grows quickly with speed: 1.1 mm at the docking speed, 1.7 mm at the entry speed, and 17.6 mm at 0.30 m/s. This agrees in direction with Eq. (10), where the damping ratio falls with the square root of the speed, from 0.72 at 0.05 m/s to 0.29 at 0.30 m/s. The speeds chosen for the mission are on the well-damped side.

Table 5. Line follower alone on the entry line, one run per row. Offset of the array centre from the centre of the line.

Noise [V]	Speed [m/s]	RMS offset [mm]	Largest offset [mm]	Start square reached
0	0.10	1.5	4.0	yes
0.05	0.10	1.7	4.3	yes
0.2	0.10	1.9	4.1	yes
0.5	0.10	6.8	19.7	yes
1.0	0.10	18.0	45.9	yes
0.05	0.05	1.1	2.8	yes
0.05	0.15	3.3	7.7	yes
0.05	0.20	7.4	16.3	yes
0.05	0.30	17.6	37.9	yes

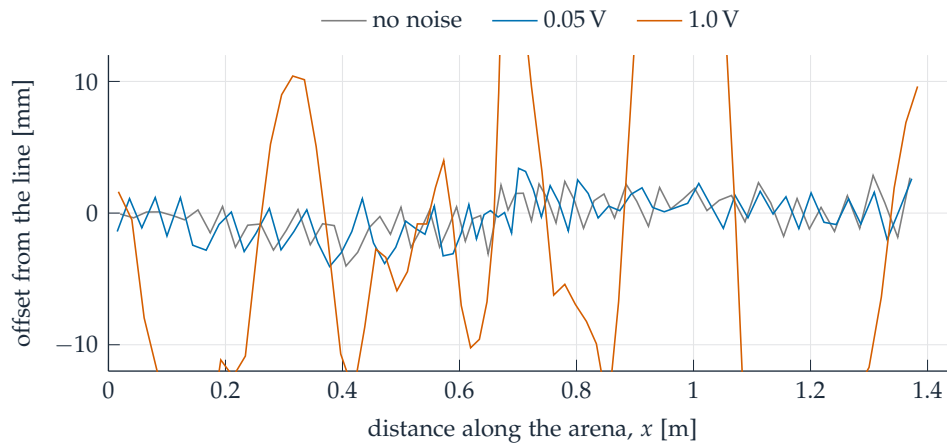


Figure 15. Offset of the array centre from the entry line at 0.10 m/s, for three levels of noise on the infrared readings. The two arcs of the S-bend lie between $x = 0.15$ and 1.19 m.

9.4 Localisation

Over the part of each mission that is driven in the room, the pose in the map differed from the true pose by 9.9 mm RMS (8.2 to 11.8 mm over the runs). The 95th percentile was 19 mm on average, the largest error in any run 39 mm, and the heading error 0.7° RMS. These errors are below the goal tolerance of the navigation, 20 mm, for most of the time, and well below the lateral error that a stem can absorb.

This result has to be read with a second one (Fig. 16). The odometry filter alone, without any correction from the map, was never more than 30 mm from the truth in any run, and ended

a mission of 18 m with an error of 16 mm on average. In this simulation the odometry is nearly ideal: the encoders have no quantisation, the floor is uniform and the drive stays inside its traction limit. The map therefore has little to correct, and its corrections are of the same size as the error they remove. The runs show that the localisation is stable over five minutes and does not jump, which was the failure described in Section 8.3; they do not show how much it would contribute on a real floor, where wheel odometry drifts.

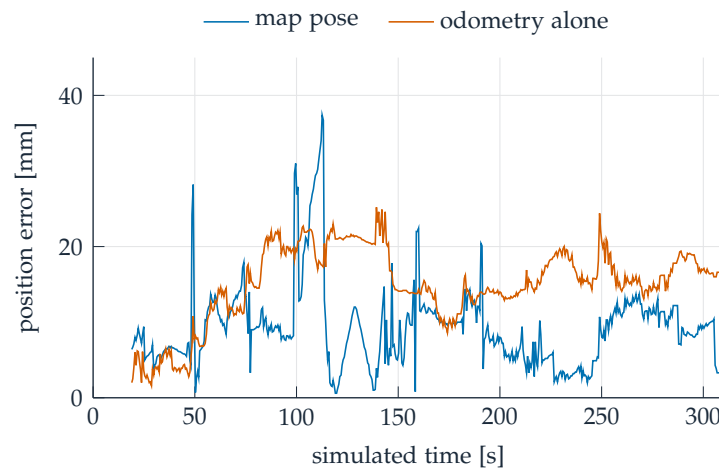


Figure 16. Distance between the estimated and the true position of the robot in run 3, for the pose in the map and for the odometry filter alone.

9.5 Docking

Figure 17 gives the accuracy of the two movements on which the manipulation depends, over all 36 picks and 36 drops.

Pick. At the moment the jaws closed, the centre of the array was 2.7 mm RMS to the side of the stem, with a largest offset of 7.1 mm, and on average 2.9 mm past the centre of the crossbar (0.9 to 6.0 mm), that is, always on the bar. The heading differed from the direction of the stem by 2.9° RMS and by 7.1° at most. The heading is the least well controlled quantity: a stem of 0.30 m removes a lateral error faster than it removes the turn that the correction itself introduces. All 36 coins were nevertheless caged, which is the tolerance that the shape of the jaws was designed to provide.

Drop. The coins came to rest 9.0 mm from the centres of their squares on average and 20.7 mm at most. The square extends 125 mm from its centre, so the margin is a factor of six. The offsets are not centred: every coin lies between 5 and 11 mm short of the centre along the stem, which is the distance by which the coin sits further back between the jaws than the nominal 0.13 m.

9.6 The Simulated Robot

Figure 18 shows the simulation during a mission.

9.7 Summary

- The mission succeeded in twelve of twelve runs, with four obstacle layouts and eleven combinations of colours, in 300 s on average.
- The line follower holds the line within 6 mm at the speeds used, tolerates four times the nominal sensor noise, and degrades with speed as the analysis of its damping predicts.

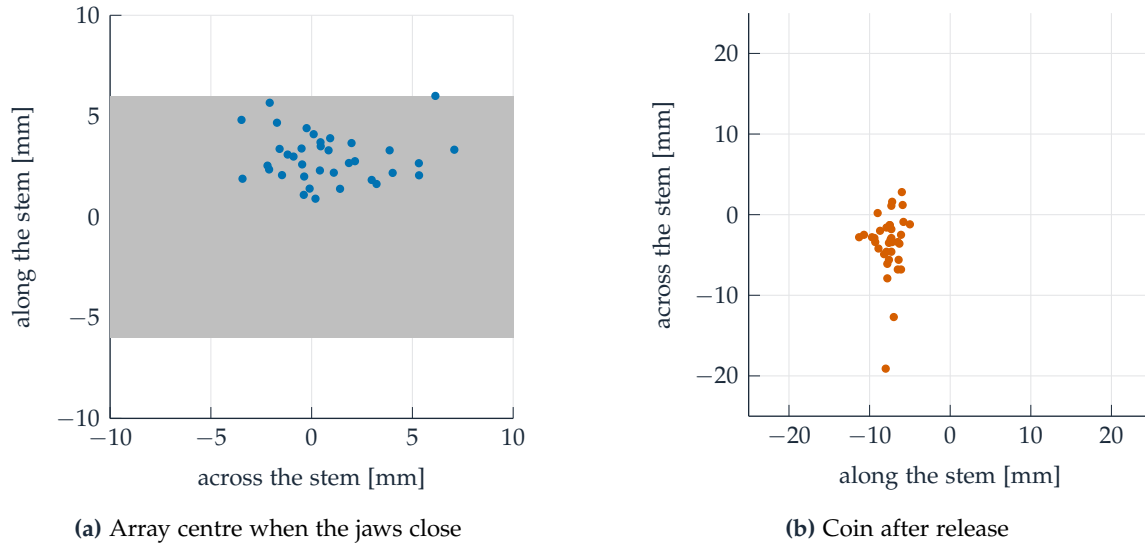
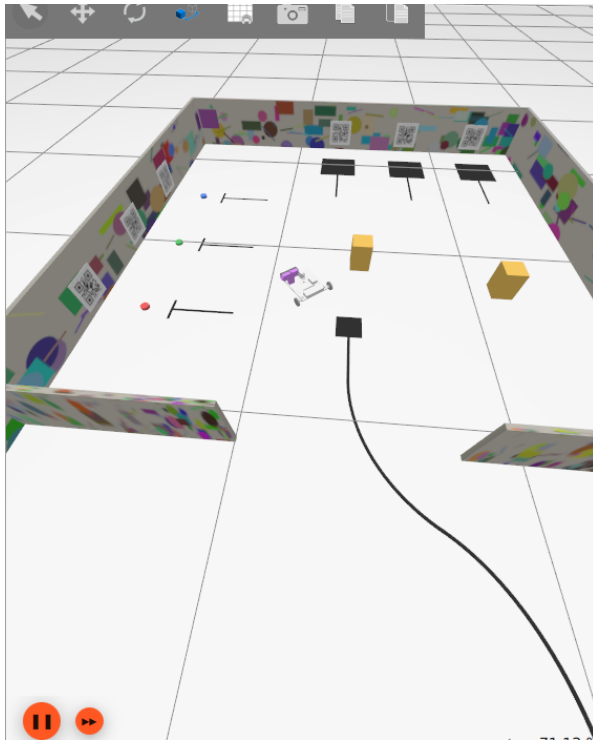


Figure 17. Docking accuracy over all picks and drops. (a) Position of the centre of the infrared array relative to the centre of the crossbar at the moment the jaws close; the grey band is the crossbar, 12 mm wide. (b) Final position of each coin relative to the centre of its drop square, which extends to ± 125 mm.

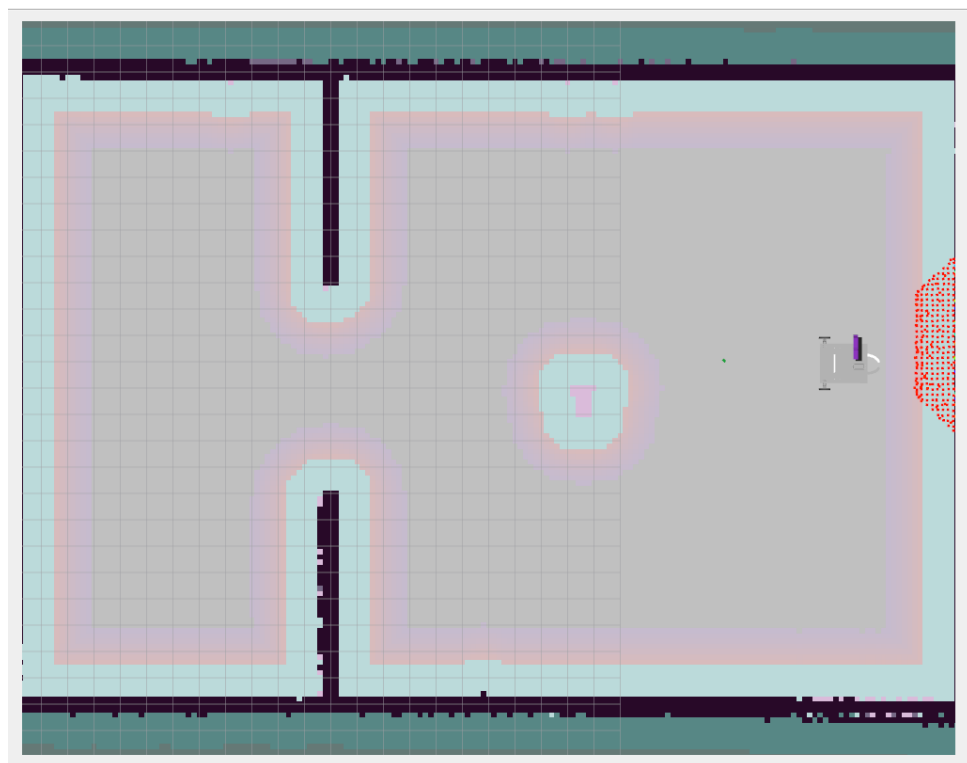
- The pose in the map is accurate to 10 mm RMS, but the simulated odometry is nearly as good, so the simulation does not test the value of the map.
- The robot stops on a crossbar within 7 mm laterally and places coins within 21 mm of the centre of a square of 250 mm.



(a) Entry line and room in Gazebo



(b) Robot in the room, seen from above the entry lane



(c) Map and inflated costmap in RViz, with the robot at a drop zone

Figure 18. The simulated mission.

10 Verification

10.1 Unit Tests

The arithmetic of the mission is kept in two modules that do not import ROS 2: the line mathematics (offset, marker detection and steering) and the table of stations and colours. They are covered by 13 unit tests, which run in under a second without the simulator. The tests fix the behaviour that the rest of the mission relies on:

- a floor without a line gives no offset, and a line between the middle channels gives an offset of zero;
- a line under the leftmost channel gives a positive offset and one under the rightmost a negative offset, which fixes the sign convention against the channel order of the robot description;
- readings below the floor threshold are ignored;
- a marker requires its number of dark channels in successive samples, and a single sample of a line in between resets the count;
- the steering turns towards the line, saturates at its limit, and its derivative term opposes an error that is closing;
- the station file holds three bays and three drop zones, the table matches colours to zones, and an unknown colour is rejected.

All 13 tests pass on the code as measured in Section 9.

10.2 Scoring Against Ground Truth

A mission can report success and still have failed, for example when a coin slips out of the jaws on the way and the robot releases nothing. The verdict on a run is therefore not taken from the mission but from the referee of Section 8.8, which reads the true positions of the coins. The two are separated in the software as well: the ground truth is bridged from the simulator on a topic of its own, and no node of the robot subscribes to it.

10.3 Recording

The measurements of Section 9 were taken by a recording node that only listens. It logs, at 10 Hz of simulated time, the true pose of the robot, the pose estimated by the localisation and the pose of the odometry, together with every change of mission state, every command to the jaws and every line of the referee. The scripts that run the missions and evaluate the recordings are kept with the report, so that every number in it can be reproduced.

11 Discussion and Limitations

11.1 Validity of the Results

Four limitations apply to the results of Section 9 and determine how far they may be generalised.

Simulation only. Every measurement in this report was taken in the simulation. None was taken on the printed robot, for which there is no recorded data beyond the photographs and the outcome of the two competition rounds. The simulated motors are ideal within their limits, the floor is flat and evenly lit, and the infrared model has no dependence on ambient light. The

results show that the software is correct with respect to its own model; they do not predict the performance of the hardware.

A model built on CAD values. The mass properties on which Section 4.2 rests are those that the CAD program calculated. The robot was not weighed, and the motor data in the model are placeholders. The conclusions drawn there are large enough to survive moderate errors, since a wheel load of 18 % would have to be wrong by a factor of several before the conclusion changed, but the figures are not measurements.

A favourable arena. The arena was designed together with the software. Its markings have the widths that the array needs, its walls have the textures that the SLAM system needs, and its lighting casts no shadows. A competition field offers none of these by arrangement. The arena shows that the approach works when its assumptions hold, and Section 7 states what those assumptions are.

A small number of runs. The mission was run twelve times and each line-following condition once. The runs differ in the order of the colours and in the obstacles, not in the start pose, the map or the lighting, and the map was the same in all of them. Differences of a few millimetres or a few seconds between conditions are within what a repetition could change.

11.2 Known Issues

The wheels are lightly loaded. The model puts 82 % of the weight on the caster (Section 4.2). The simulation works within the resulting traction limit only because the drive is limited to 1.0 m/s^2 . On the hardware, the caster should be moved towards the gripper and the camera to the centre line, after the robot has been weighed on its three supports.

The caster does not reach the floor in CAD. The dome ends 0.74 mm above the wheel contact plane (Section 6.1). The simulation corrects this in its collision geometry; the physical part needs the same correction.

The jaws do not meet on the centre line. The second jaw is 7.25° off the mirror image of the first in the exported assembly (Section 4.4).

The footprint used for planning is a disc. The navigation treats the robot as a disc of 0.13 m radius about the midpoint of the axle. The closed jaws reach 0.21 m ahead of that point and lie outside the disc. The inflation of the costmap by 0.25 m covers the difference in the present arena; a polygonal footprint would be the correct description.

Low obstacles are not seen. Anything lower than 3 cm is treated as floor, and nothing closer than 0.15 m in front of the robot is seen at all.

Localisation depends on a threshold. The condition that rejects look-alike views (Section 8.3) is a single number chosen for this arena and this camera height. It rejects some true matches, and a different arena would need it chosen again.

Dimensions are repeated in three files. The station poses and the drop squares mirror the arena file by hand (Section 7).

No feedback from the jaws. The mission assumes that a coin is caged when the array is on the crossbar and the jaws have had time to close. It cannot tell that a coin was missed or lost on the way. Only the referee sees this.

One line sensor, no fallback. This is the failure of the state round (Section 5), and the simulated mission has the same dependency: every approach ends on the infrared array.

11.3 What I Would Do Differently

Three changes follow from this work, in order of value.

1. *Check the mass properties during the design, not after it.* The calculation of Section 4.2 takes minutes once the CAD model exists. Done during the fifteen days, it would have moved the caster and the camera before the first print.
2. *Treat the line sensor as a part that can fail.* A check of all eight channels on a black and a white reference at power-up, a soldered harness, and a degraded mode that crosses the line section on odometry and vision are the measures that address a failure of this kind.
3. *Build the simulation first.* The simulation was written after the competition. Written before it, it would have allowed the mission software to be developed while the parts were printing.

11.4 Future Work

- Weigh the printed robot on its supports, measure the motors, and replace the placeholder values of the model.
- Move the caster and the camera as proposed and repeat the analysis of Section 4.2.
- Run the mission on the hardware, beginning with the line follower and the docking, which need only the infrared array and the encoders.
- Add a contact or current signal from the gripper so that a missed coin is detected.
- Derive the station poses and the referee squares from the arena file.
- Vary the start pose, the lighting and the floor friction in the simulation, and repeat each condition several times.

12 Conclusion

Trident is a small robot designed around one task, and the design decisions that matter most are the ones that removed something. The gripper has no lift axis and no sensor: two curved jaws on a pair of equal gears cage a coin and slide it, and ridges on the jaws leave it behind on release. The camera cannot see the floor in front of the robot, and no second camera was added: the infrared array that follows the line also guides the last part of every approach, and the floor markings are dimensioned from its pitch so that it can.

Two lessons recur. The first is that a model answers questions that were settled by eye during the build. The centre of mass lies inside the support triangle, as intended, but the mass properties of the CAD model put 82 % of the weight on the caster, and this, and not the motors, limits the acceleration of the robot. The simulation showed the consequence as a skid before the calculation explained it. The second is that the position of a sensor is a control parameter. The damping of the line follower is set by the distance of the array ahead of the axle, and an array on the axle would have needed a different controller.

The simulated mission is complete: the robot follows the line, maps and localises with its camera, drives around unmapped obstacles, and delivers the three coins by colour, scored by a referee that it cannot see. The steps that remain are on the hardware: to weigh the robot on its supports and move the caster and the camera accordingly, to replace the breadboard wiring of the infrared array, and to run the mission on the printed robot.

References

- [1] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. MIT Press, Cambridge, MA, 2nd edition, 2011.
- [2] Wim Meeussen. REP 105: Coordinate frames for mobile platforms. ROS Enhancement Proposal, 2010. <https://www.ros.org/reps/rep-0105.html>.
- [3] Karl Johan Åström and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton, NJ, 2008.
- [4] R. Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1992.
- [5] Alberto Rodriguez, Matthew T. Mason, and Steve Ferry. From caging to grasping. *The International Journal of Robotics Research*, 31(7):886–900, 2012. doi: 10.1177/0278364912442972.
- [6] Matthew T. Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986. doi: 10.1177/027836498600500303.
- [7] Nathan Koenig and Andrew Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154, 2004. doi: 10.1109/IROS.2004.1389727.
- [8] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66): eabm6074, 2022. doi: 10.1126/scirobotics.abm6074.
- [9] Mathieu Labbé and François Michaud. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019. doi: 10.1002/rob.21831.
- [10] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel RealSense stereoscopic depth cameras. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1267–1276, 2017. doi: 10.1109/CVPRW.2017.167.
- [11] Thomas Moore and Daniel Stouch. A generalized extended Kalman filter implementation for the Robot Operating System. In *Intelligent Autonomous Systems 13*, volume 302 of *Advances in Intelligent Systems and Computing*, pages 335–348. Springer, 2016. doi: 10.1007/978-3-319-08338-4_25.
- [12] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.
- [13] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. The Marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2718–2725, 2020. doi: 10.1109/IROS45743.2020.9341207.
- [14] David V. Lu, Dave Hershberger, and William D. Smart. Layered costmaps for context-sensitive navigation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 709–715, 2014. doi: 10.1109/IROS.2014.6942636.
- [15] Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, Boca Raton, FL, 2018.

-
- [16] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. doi: 10.1007/BF01386390.
 - [17] Steve Macenski, Shrijit Singh, Francisco Martín, and Jonatan Ginés. Regulated pure pursuit for robot path tracking. *Autonomous Robots*, 47:685–694, 2023. doi: 10.1007/s10514-023-10097-6.
 - [18] Richard G. Budynas and J. Keith Nisbett. *Shigley's Mechanical Engineering Design*. McGraw-Hill Education, New York, 10th edition, 2015.
 - [19] Jeff Brown. ZBar bar code reader. Open-source software. <https://zbar.sourceforge.net>.

Appendices

A Parameters

Tables 6 and 7 list the parameters of the simulation model and of the mission as used for the measurements of Section 9.

Table 6. Parameters of the simulation model.

Parameter	Value
Physics step	1 ms
Friction: wheels, caster, jaws, coin	1.0, 0, 0.8, 0.35
Drive limits	1.0 m/s ² , 8.0 rad/s ²
Wheel joint limits (placeholders)	0.05 N m, 31.4 rad/s
Jaw joint: range, effort, speed	−1.0 to 0.09 rad, 0.83 N m, 5.2 rad/s
Wheel odometry rate	30 Hz
Depth camera: image, rate	848 × 480, 60 Hz
Depth camera: field of view, range	87° × 58°, 0.2 to 10 m
Infrared array: channels, rate, noise	8, 20 Hz, 0.05 V
Inertial sensor: rate, gyroscope noise	100 Hz, 0.002 rad/s
Coin: diameter, height, mass	40 mm, 15 mm, 23.4 g

Table 7. Parameters of the mission and of the navigation.

Parameter	Value
Line follower: k_p , k_d , $\omega_{\max}$	45, 2, 1.5 rad/s
Line follower: floor threshold, dark level	0.15, 0.5
Marker: channels, samples	6, 2
Speed on the entry line, on a stem	0.10 m/s, 0.05 m/s
Reach on the entry line, on a stem	3.0 m, 0.60 m
Attempts per station	2
QR code: identical decodes, time allowed	3, 4 s
Time allowed for the jaws	1 s
Navigation speed, look-ahead	0.15 m/s, 0.20 m
Goal tolerance	2 cm, 0.05 rad
Costmap cell, robot radius, inflation	2 cm, 0.13 m, 0.25 m
Obstacle height band	0.03 to 0.40 m
Replanning rate	1 Hz
Map: node spacing, loop detection rate	0.10 m or 0.2 rad, 2 Hz
Minimum inliers, minimum inlier spread	25, 0.005

B Running the Simulation

All commands are given to one script on the host. The image has to be built once.

```
./docker/run.sh image      build the Docker image
./docker/run.sh up        start the container
./docker/run.sh build      build the workspace
./docker/run.sh test       unit tests of the mission logic
./docker/run.sh map        survey the empty arena and save the map
./docker/run.sh mission    run the mission with the referee
./docker/run.sh mission pick_order:=GBR drop_order:=RGB obstacles:=b
```

The measurements of Section 9 were produced by scripts kept in the folder `docs/latex/figures/src`. The script `run_missions.sh` runs a list of missions without a display and records each, `run_line_sweep.sh` does the same for the line follower alone, and `analyse_runs.py` and `analyse_line.py` evaluate the recordings.