PASHUPATI

# Tuning Guide

What the parameters do, and how to tune a Genisom L1W for a site

Version 2.9 · 2026-11-15

For the technician who sets up and tunes a Pashupati robot. You need an SSH shell on the robot computer and some ROS 2 basics.

# Revision history

| Version | Date | Changes |
|---|---|---|
| 1.0 | 2026-10-16 | First version: Pashupati with the Genisom L1W |
| 2.0 | 2026-10-29 | Pashupati 3 on Dharma 0.2: robot profile, stack file, sites, new parameter names |
| 2.1 | 2026-11-01 | Time synchronisation: chrony, PTP for the LiDAR, the offset to NETRA |
| 2.2 | 2026-11-14 | Field fixes on the L1W: perception as in 2.0, path shift patience and emergency slide, trip meter |
| 2.3 | 2026-11-14 | Path shift passes obstacles beside it parallel and drives narrow lanes in their middle |
| 2.4 | 2026-11-14 | Detours: turning in place toward a gap when sliding does not get the robot on |
| 2.5 | 2026-11-14 | Low obstacles from the RealSense depth camera, below the LiDAR |
| 2.6 | 2026-11-15 | The camera's measured mount, its point cloud on the Jetson, USB 3, depth_floor.py |
| 2.7 | 2026-11-15 | The L1W steps over 10 cm and passes gaps from about 0.7 m |
| 2.8 | 2026-11-15 | menuconfig for the site and env files, the console monitor and its report |
| 2.9 | 2026-11-15 | Backing up when stuck, the internet check, a console that takes taps |

Operating the robot is described in the Pashupati User Manual. Pashupati 3 runs on the Dharma 0.2 stack: every Dharma parameter is typed (unit, range, description) and listed by `ros2 param describe`. The parameter reference at the end of this guide is generated from Pashupati's parameter files. Units: m, s, m/s, rad/s, deg; 0-1 means a fraction.

# Contents


**1 How tuning works ..... 5**

1.1 Where a value comes from ..... 5
1.2 Change a value for good ..... 5
1.3 Try a value live ..... 7
1.4 What the handheld changes ..... 7
1.5 Tuning a new site, step by step ..... 8
1.6 Work safely ..... 8

**2 The Genisom L1W ..... 10**

2.1 The robot profile ..... 10
2.2 Speed limits ..... 11
2.3 Stick channels and speed scaling ..... 11

**3 Protective field ..... 13**

3.1 How it works ..... 13
3.2 Measure the braking ..... 14
3.3 Test it ..... 14

**4 The patrol ..... 16**

4.1 Speed and motion ..... 16
4.2 Stop points ..... 16
4.3 Leaving and joining the route ..... 17

**5 Obstacles ..... 18**

5.1 Path shift ..... 18
5.2 What the LiDAR sees ..... 20
5.3 What the camera sees ..... 21

**6 Sites, maps and localization ..... 24**

6.1 Sites ..... 24
6.2 AprilTags ..... 24
6.3 Odometry ..... 25

**7 Inspection and people ..... 26**

7.1 Pace and queue ..... 26
7.2 Model and memory ..... 27
7.3 Finding types ..... 27
7.4 People counting ..... 28

**8 Handheld, camera, video, NETRA, health and time ..... 29**

8.1 Handheld link ..... 29
8.2 Camera source ..... 30
8.3 Video ..... 30
8.4 NETRA uplink ..... 30
8.5 Health monitor ..... 30
8.6 Time synchronisation ..... 31
8.7 Console monitor ..... 32

**9 The env file ..... 34**

**10 Symptoms and fixes ..... 36**

**11 Site file for the Genisom L1W ..... 40**

**12 Parameter reference ..... 41**

12.1 pashupati ..... 41
12.2 pashupati ..... 41
12.3 pashupati_bridge ..... 46
12.4 pashupati_inspection ..... 47
12.5 pashupati_fleet ..... 48

# 1 How tuning works

## 1.1 Where a value comes from

Every node starts with the defaults of its package. Pashupati then applies, in this order (later wins):

| Layer | Where | Changed by |
|---|---|---|
| Robot profile | `src/pashupati/config/robot.yaml` (or `ROBOT_PARAMS`) | a Pashupati or robot update |
| Stack file | `src/pashupati/config/stack.yaml` | a Pashupati update |
| Site file | `/etc/pashupati/site.yaml` (`SITE_CONFIG`) | you, for this robot and site |
| Launch | non-empty values of `/etc/pashupati/pashupati.env` | you (chapter 9) |
| Live changes | in the running node | handheld, `ros2 param set`, RViz; lost at a restart, except the handheld's inspection and people settings |

The **robot profile** (`robot.*`) is the one description of the robot that every node reads: kinematics, base height, footprint, speed limits and braking (chapter 2). The **stack file** holds only what Pashupati changes from the Dharma defaults; most values tuned on the robot in Pashupati 1 and 2 became the Dharma defaults.

Never change the repository on a robot: the next `git pull` conflicts with your change. Put every change in the site file. The robot driver (`l1w_driver_node`) is not a Pashupati node; its settings live in the robot package (section 2.3).

## 1.2 Change a value for good

The quickest way is **menuconfig**, the Dharma console's menu (`pconfig` in the ROS shell of the stack, `pashu`, or `scripts/console.sh menuconfig`):

```
Pashupati
  Navigation  --->
    path_follower_node  --->
      path_shift  --->
        (0.05) clearance
stack.yaml
        (0.25) centre_room                                    default
```

Every node of the stack is there with every parameter, typed. A tap selects a line and a second tap opens or edits it (a phone's SSH app needs no arrow keys); the `[Save]`, `[Find]`, `[Help]`, `[Default]`, `[Back]`, `[Monitor]` and `[Quit]` buttons do the same as the keys. `<Enter>` edits a value (or picks a choice, or toggles a `[*]` switch), `/` searches names and descriptions, `?` shows what a value does with its unit, range and value in every layer, `d` goes back to the inherited value. The tag on the right says where the value comes from: `default` (Dharma's), the package's file, `stack.yaml`, `site` (the site file) or `changed` (not saved). A value out of range or against another value is refused before it is saved. `s` lists the changes, writes the site file (a `.bak` copy first; `sudo` asks for the password) and offers to restart the stack. *Launch options* edits the env file the same way (chapter 9). Setting a value back to the inherited one removes it from the site file, which then holds only what this robot changes.

By hand:

1. Open the site file: `sudo nano /etc/pashupati/site.yaml`.
2. Find the node's section, or add one like the example below. Indent with spaces, two per level, never with tabs.
3. Restart the stack: `sudo systemctl restart pashupati`.
4. Check the start of the log for warnings:
   `journalctl -u pashupati -n 100 | grep -i warn`.

```
path_follower_node:
  ros__parameters:
    speed:
      cruise: 0.35
```

A misspelt **node** name is reported at start (*no node is called "..."*). A misspelt **parameter** is reported by the node (*undeclared parameter*), and a value out of its range stops the node with the reason. A file that is not valid YAML stops the start, so the robot never runs on defaults by accident.

## 1.3 Try a value live

```
pashu                                                    # ROS shell of
the stack
ros2 param describe /path_follower_node speed.cruise     # unit, range,
live or read-only
ros2 param set /path_follower_node speed.cruise 0.3
ros2 param dump /path_follower_node                      # every value in
use
```

Safety values (limits, margins, timeouts) are **read-only at runtime**: `ros2 param set` refuses them; change them in the site file and restart. The **Tuning** panel of RViz (`dharma_rviz`) edits the live parameters of any node; **Save YAML** writes them for the site file. RViz runs on a laptop with ROS 2 Humble on the robot's network: set `ROS_LOCALHOST_ONLY=0` in the env file, restart, and use the robot's `ROS_DOMAIN_ID` and `RMW_IMPLEMENTATION` on the laptop.

> **SET IT BACK**
>
> With `ROS_LOCALHOST_ONLY=0` anyone on the robot's network can send motion commands to the robot. Set it back to `1` and restart when you are done.

## 1.4 What the handheld changes

| On the handheld | Parameter | After a restart |
|---|---|---|
| Patrol speed slider | `path_follower_node` `speed.cruise`, from `teleop_udp_node` `patrol_speed_min` to `patrol_speed_max` (and never above `robot.limits.max_vx`) | the site file value |
| Inspect the floor | `anomaly_detector_node` `enabled` | kept |
| Pace: Continuous, Interval, Stop points | `interval_sec`: 0, the interval, -1 | kept |
| Model | `model` | kept |

| On the handheld | Parameter | After a restart |
|---|---|---|
| Finding types | `categories`, `alarm_categories`, `category_descriptions` | kept |
| Count people | `person_detector_node` `enabled` | kept |
| Speed 1, 2, 3 | 30, 60, 100 % of `teleop_udp_node` `max_vx`, `max_vy`, `max_wz` (capped by `robot.limits`) | a handheld setting |
| Inspection at a stop point | the dwell time stored in the route with each **Point** | in the route |

*Kept* values are stored in `~/.pashupati/inspection.json` and `~/.pashupati/people.json` and **win over the site file**. To go back to the site file's inspection settings, stop the service, delete `inspection.json`, and start it again.

## 1.5 Tuning a new site, step by step

1. **The robot, once:** the robot profile and the stick channels (chapter 2).
2. **Braking:** measure it and test the protective field (chapter 3).
3. **The site:** map it, print, place and record the AprilTags (chapter 6).
4. **First route:** teach a short route and patrol it at 0.3 m/s (chapter 4).
5. **Obstacles:** a cardboard box on the route, then a person walking past (chapter 5).
6. **Full speed:** raise the patrol speed step by step, one full lap per step.
7. **Inspection and people:** pace, model, finding types, people counting (chapter 7).
8. **Save:** copy every value that works into the site file, restart, drive one more lap.

Try a change on the simulator first (`ros2 launch pashupati sim.launch.py rviz:=true`): it runs the same stack with the same files.

## 1.6 Work safely

- Tune in a closed area first, never in a crowd.
- Keep the ZSBot remote in reach, and someone on the handheld's STOP.
- Change one thing at a time. Write down the old and the new value.

- Start low: speeds and accelerations below what you need, then raise them step by step.
- Repeat the protective field test (section 3.3) after every change to the robot profile.

# 2 The Genisom L1W

## 2.1 The robot profile

`src/pashupati/config/robot.yaml` describes a ~0.80 × 0.56 m quadruped, field-tested on the L1W. A robot package can ship its own profile (`ROBOT_PARAMS`); to change one value for one robot, override it in the site file under `/**:` (every node reads it).

| Parameter | L1W | What it is |
|---|---|---|
| `robot.kinematics` | omni | the L1W strafes; `diff` for a robot that cannot (then `max_vy` 0) |
| `robot.base_height` | 0.35 | m, `base_link` above the floor when standing (the floor for the LiDAR and for placing people) |
| `robot.footprint.front`, `.rear`, `.half_width` | 0.40, 0.40, 0.28 | m from `base_link` to the body's edges, legs included: the protective field, the planner, path shift and the LiDAR's self filter all use it |
| `robot.limits.max_vx`, `max_vy`, `max_wz` | 1.0, 0.5, 1.5 | nothing may command more, in AUTO or REMOTE |
| `robot.limits.max_accel`, `max_yaw_accel` | 0.5, 1.5 | m/s$^2$, rad/s$^2$ |
| `robot.limits.max_decel` | 0.6 | m/s$^2$, braking the robot reliably achieves: **measure it** (section 3.2) |
| `robot.limits.reaction_time` | 0.25 | s, from the command to the robot actually braking |

To measure the height of `base_link`, stand the robot on a flat floor, measure the height of the LiDAR above the floor, and subtract the LiDAR's height in the body: `ros2 run tf2_ros tf2_echo base_link livox_frame` (the z value). Measure the footprint again when the robot carries a payload or a cover that sticks out.

```
/**: # in the site file: a heavier L1W with a cover
  ros__parameters:
    robot:
      footprint: { front: 0.45, rear: 0.40, half_width: 0.30 }
      limits: { max_decel: 0.5 }
```

## 2.2 Speed limits

| Parameter | L1W | Limits |
|---|---|---|
| `robot.limits.max_vx`, `max_vy`, `max_wz` | 1.0, 0.5, 1.5 | every command: the control mux clamps to them |
| `teleop_udp_node` `max_vx`, `max_vy`, `max_wz` | 1.0, 0.5, 1.5 | the handheld at speed level 3 (never above the profile) |
| `path_follower_node` `speed.cruise` | 0.4 | the patrol speed (the handheld's slider changes it) |
| `l1w_driver_node` `stick_max_vx`, `stick_max_vy`, `stick_max_wz` | 1.0, 0.5, 1.5 | the command that means a full stick (next section) |

## 2.3 Stick channels and speed scaling

The L1W takes motion as joystick values, like its own remote. `l1w_driver_node` turns each speed command into a stick position: a command of `stick_max_vx` is a full stick forward. Its settings are in the robot package, `src/xtras/robots/genisom_l1w_ros2/l1w_driver/config/l1w_driver.yaml`. Edit them there and restart the service; they are not live.

| Parameter | L1W | Meaning |
|---|---|---|
| `stick_max_vx`, `stick_max_vy`, `stick_max_wz` | 1.0, 0.5, 1.5 | speed that maps to full stick deflection |
| `stick_yaw_axis`, `stick_lateral_axis` | 2, 1 | stick channels for turning and walking sideways |
| `stick_invert_yaw`, `stick_invert_lateral` | false | true when the robot turns or walks sideways the wrong way |

| Parameter | L1W | Meaning |
|---|---|---|
| `cmd_timeout` | 0.5 | s; without a new command the stick returns to zero |
| `auto_control_right` | true | take motion control from the ZSBot remote when the driver connects |

**Check the directions once** (REMOTE, speed level 1): left stick to the left must walk the robot to its left; right stick to the left must turn it to the left. If one is wrong, set its `stick_invert_*` to true.

**Check the speed scale:** the speed the L1W reaches at full stick depends on its speed gear (*Slow*, *Normal*, *Fast speed* in the robot actions). Patrol a straight route and compare the patrol speed with the measured speed: `ros2 topic echo /l1w/odom --field twist.twist.linear.x`. If the robot is slower or faster, set `stick_max_vx` to its value × measured speed ÷ patrol speed, and scale `stick_max_vy` and `stick_max_wz` the same way. Always patrol in the gear you checked.

# 3 Protective field

## 3.1 How it works

`control_mux_node` is the only node that commands the robot. It limits every command to a speed at which the robot can still stop a margin short of the nearest obstacle in the path it is about to sweep, in every direction (the LiDAR's 360° safety scan), from the robot profile's `max_decel` and `reaction_time` and an assumed walking speed of a person toward the robot. The Dharma guide *Safety* gives the formula.

The field always works in AUTO. In REMOTE the operator watches the robot: Pashupati drives REMOTE without the field ( `field.in_remote: false` , a field-tested decision); the speed limits and the e-stop still apply. Without a fresh scan the robot does not drive itself at all.

| `control_mux_node` parameter | Default | What it does |
|---|---|---|
| `field.in_remote` | false | true: limit REMOTE too (with `field.remote_margin` ), e.g. for operators new to the robot |
| `field.margin` | 0.15 | m, gap left to the obstacle in AUTO |
| `field.remote_margin` | 0.10 | m, the same in REMOTE |
| `field.lateral_margin` | 0.05 | m, extra width of the swept path on each side (Dharma: 0.10; section 5.1) |
| `field.human_speed` | 0.5 | m/s, a person walking toward the robot |
| `field.min_speed` | 0.02 | m/s, a slower allowed speed stops instead |
| `field.remote_degraded_speed` | 0.15 | m/s, the REMOTE cap while the scan is stale or blind |
| `field.scan_timeout_sec` | 0.5 | s; an older scan stops AUTO and caps REMOTE |
| `use_health_limit` | true | scale every command by the health monitor's speed limit (section 8.5) |

| control_mux_node parameter | Default | What it does |
|---|---|---|
| remote_sources | [handheld, dashboard] | who may drive in REMOTE ( control/remote/<source>/cmd_vel ) |
| remote_timeout_sec , nav_timeout_sec | 0.3, 0.5 | s; a silent source gets zero speed |

A lower max_decel , a longer reaction_time or a bigger margin make the robot slow down earlier and stop further from obstacles. Too low, and it crawls in narrow aisles.

## 3.2 Measure the braking

1. Teach a straight route of at least 8 m, with free space beyond it.
2. Set the patrol speed on the handheld to the fastest you will patrol at, for example 0.6.
3. Put a tape mark on the floor in the middle of the route.
4. Start the patrol. Press **STOP** when the front of the robot passes the mark.
5. Measure the distance *d* from the mark to the front of the stopped robot.
6. max*decel* = $speed^2 \div (2 \times (_d - \text{speed} \times 0.25))$.
7. Repeat three times. Use the smallest result as robot.limits.max_decel .

Your own reaction is part of *d*, so the result errs on the safe side. At 0.6 m/s:

| Stopping distance *d* | 0.40 m | 0.50 m | 0.60 m | 0.80 m | 1.00 m |
|---|---|---|---|---|---|
| robot.limits.max_decel | 0.72 | 0.51 | 0.40 | 0.28 | 0.21 |

Measure on every new kind of floor: a polished floor brakes worse than a rough one.

## 3.3 Test it

1. In the site file set control_mux_node field.in_remote: true and restart.
2. Put a cardboard box in an open area.
3. In REMOTE at speed level 1, drive straight at the box. The robot must slow down and stop about field.remote_margin (10 cm) before it. Try level 2, then from the side and backwards.
4. Set field.in_remote back to false (or keep it on) and restart.

If the robot gets closer than the margin, lower `robot.limits.max_decel` or raise `robot.limits.reaction_time`. Never test with a person in front of the robot.

# 4 The patrol

## 4.1 Speed and motion

All in `path_follower_node`; the speed values are live. *Pashupati* is the value with the stack file (the Dharma default where Pashupati sets nothing).

| Parameter | Pashupati | What it does · when to change |
|---|---|---|
| `speed.cruise` | 0.4 | m/s, the patrol speed; keep it low in crowds |
| `speed.min_speed` | 0.1 | m/s, the slowest the speed profile goes |
| `speed.lateral_accel` | 0.3 | m/s$^2$, the speed in curves: lower is slower in bends |
| `speed.accel`, `speed.decel` | 0.5, 0.3 | m/s$^2$, starting and braking into stops (Pashupati: 0.5 to start) |
| `speed.approach_speed` | 0.15 | m/s when arriving at stop points and the route end |
| `speed.smoothing` | 0.4 | 0-1, lower is smoother and slower to react |
| `controller` | holonomic | x, y and yaw together; strafes, keeps its heading along the route |

## 4.2 Stop points

At each stop point the robot converges (`approach.*`), calls the inspection hook, waits for its answer, dwells for the route's dwell time, optionally backs up, and walks on.

| Parameter | Pashupati | What it does |
|---|---|---|
| `approach.position_tolerance`, `approach.yaw_tolerance_deg` | 0.05, 3.0 | m, deg: how close the robot converges |
| `approach.min_speed` | 0.04 | m/s, beats the robot's joystick deadband |
| `approach.timeout_sec` | 20 | s, then it dwells where it is |

| Parameter | Pashupati | What it does |
|---|---|---|
| `hooks.inspection` | [/anomaly_detector_node/check_now] | services called at each stop; a hook whose node does not run is skipped, so `INSPECTION=false` never blocks the patrol |
| `hooks.inspection_timeout_sec` | 20 | s; VLM inference on the Jetson takes a few seconds |
| `inspection.reverse_distance_m` | 0 | m; > 0 backs up this far after the inspection (the LiDAR sectors look 360° for it) |
| `loop_route` | true | a closed route starts its next lap |
| `low_battery`, `critical_battery` | 0.2, 0.1 | 0-1: below `low_battery` the patrol ends its lap; below `critical_battery` it pauses |

## 4.3 Leaving and joining the route

| Parameter | Pashupati | What it does |
|---|---|---|
| `join.max_distance` | 3.0 | m; a robot started beside the route walks onto it from here |
| `safety.off_path_slow`, `safety.off_path_stop` | 0.6, 1.5 | m of cross-track error: slow down, stop (*off the route*) |
| `safety.blocked_timeout_sec` | 30 | s standing in front of an obstacle before *Blocked* (Dharma: 10) |
| `require_localization` | true | **Start** is refused until the robot is localized |

# 5 Obstacles

## 5.1 Path shift

The robot passes an obstacle by sliding sideways, never further than `path_shift.max_offset` from the route, and stops short when there is no gap. When it has waited `path_shift.patience_sec` with no gap (a person who does not step aside, a box wider than the offset allows), it looks for a way up to `path_shift.search_offset` from the route, left or right, and goes around; with no way there either it keeps waiting (*Blocked* after `safety.blocked_timeout_sec`). The status shows `searching` meanwhile.

Something beside the robot (a person, a bench, a corridor wall) only stops it when it touches the body: driving on parallel never comes closer, and the protective field allows the same. When there is no offset with the full clearance, the robot keeps its own line as long as nothing is ahead on it, and returns to the route once it can; it always may slide away from something beside it. In a lane bounded ahead on both sides (a narrow corridor, between a person and a wall) with less than `path_shift.centre_room` beside the body, it drives in the middle, to the centimetre.

A gap ahead needs `path_shift.clearance` beside the body on both sides, and the motion guard (`safety.lateral_margin`) and the protective field (`field.lateral_margin`) keep theirs too. Pashupati sets all three to 5 cm (Dharma: 10 cm), field-tested on the L1W: the body, 0.56 m wide, passes gaps from 0.66 m, about 0.7 m in practice (0.76 m with 10 cm). The guard's margin must not be below the field's, or the guard lets the robot into a gap the field then stops it in. The 360° safety scan has 144 sectors of 2.5° (`safety.num_sectors`, Dharma: 72): wider sectors place the sides of a narrow gap nearer the path than they are.

When sliding does not get it on (walls too close for the guard to let it slide, a gap it passes only facing it, a crowd), the robot makes a **detour**: stuck for `detour.after_sec` (slower than 0.1 m/s on average, 1.5 s), it turns in place toward the gap that takes it furthest along the route, within `detour.max_angle_deg` of the route direction and no further from the route than path shift goes, and drives a straight run of up to `detour.length` at `detour.speed`. The turn and the run are checked like every command (room to turn, the swept body, the protective field); the run ends as soon as the way along the route is open again, and path shift brings the robot back. The status reads `detour: turning to +45 deg off the route, toward a gap`. Near a stop point there is no detour: the robot waits on the route.

When none of this gets the robot further along the route for `backtrack.after_sec` (30 s: wedged where it can neither slide nor turn, someone who stays in front of it), it **backs up** `backtrack.distance` (1 m) along the way it came, at `backtrack.speed`, keeping the heading it had there, and tries again from there: path shift and detours then have the room they lacked. The rear is checked on the 360° safety scan and by the protective field like every motion; a rear blocked for 3 s ends the back-up. The status reads `stuck 30 s: backing up along the driven path 0.40 / 1.00 m (attempt 1/3)`. After `backtrack.max_attempts` (3) at the same place the robot stays *Blocked* and waits for help; getting past that place starts the count again. Pauses (the handheld, an e-stop) do not count as stuck.

Something in the emergency cone of `obstacle_avoidance_node` (`shared.emergency_distance` 0.75 m ahead, `shared.emergency_cone_deg` 40° in Pashupati) stops the forward motion at once. Dharma's 60° caught the sides of a gap narrower than about 0.7 m within those 0.75 m and held the robot in front of it; at ±20° the sides are left to the motion guard and the protective field, which check the whole swept body: something beside the cone still stops the robot short of it. With `path_shift.emergency_slide` the robot may still slide sideways out of the cone (a pole right in front, room beside it); the protective field and the motion guard check every slide. The status starts with `EMERGENCY: ..., forward held` while it does.

| `path_follower_node` parameter | Default | What it does |
|---|---|---|
| `avoidance_mode` | path_shift | or a reactive policy of `obstacle_avoidance_node` |
| `path_shift.max_offset` | 0.8 | m, furthest from the route |
| `path_shift.clearance` | 0.05 | m, gap kept between the robot's side and an obstacle (Dharma: 0.10) |
| `safety.lateral_margin` | 0.05 | m kept free beside the guarded path (Dharma: 0.10), >= `field.lateral_margin` |
| `path_shift.stop_margin` | 0.40 | m, stop this far before an obstacle |
| `path_shift.switch_cost`, `path_shift.switch_margin` | 0.5, 0.05 | keeps its side around an obstacle |

| path_follower_node parameter | Default | What it does |
|---|---|---|
| path_shift.patience_sec, path_shift.search_offset | 3.0, 1.5 | s, m: wait with no gap, then search this far (0.8: off) |
| path_shift.emergency_slide | true | an emergency stops only the forward motion; false: all |
| path_shift.centre_room | 0.25 | m beside the body: a narrower lane is driven in its middle |
| detour.after_sec, detour.speed | 1.5, 0.25 | s stuck before a detour (0.5 at least), m/s on its run |
| detour.max_angle_deg, detour.length | 75, 2.0 | deg off the route direction, m of one straight run |
| backtrack.enabled, backtrack.after_sec | true, 30 | back up when stuck, s without progress along the route |
| backtrack.distance, backtrack.speed | 1.0, 0.15 | m back along the way it came, m/s |
| backtrack.max_attempts | 3 | attempts at one place, then it waits (*Blocked*) |

## 5.2 What the LiDAR sees

lidar_sector_node turns the LiDAR into sectors (front, for the avoidance) and the 360° safety scan (for the protective field). The band between filter.height_min and filter.height_max above the floor (the floor is robot.base_height below base_link) counts as obstacles; the robot's own footprint, grown by filter.self_margin, is ignored.

What is lower than filter.height_min the L1W steps over: a 10 cm step, a cable taped to the floor, a dip. Pashupati sets 0.12 m (Dharma: 0.10), which leaves 2 cm for an uneven floor and the sensors' error. A box is seen once enough of it rises above the band's bottom: in a simulated depth cloud the camera finds a 14 cm box from about 2.3 m away, a 20 cm one from 3 m, and a 10 cm step stays free. A site file copied from an older site.example.yaml may still set height_min: 0.10, and its robot then stops for 10 cm steps: set 0.12 or remove the line.

| Parameter | Pashupati | What it does · when to change |
|---|---|---|
| `filter.height_min` | 0.12 | m above the floor, lower is stepped over (Dharma: 0.10); too low sees the floor on slopes |
| `filter.height_max` | 1.55 | m above the floor (the L1W's top plus margin) |
| `safety.num_sectors` | 144 | sectors of the 360° safety scan (Dharma: 72): the sides of a narrow gap where they are |
| `filter.self_margin` | 0.05 | m around the footprint: *Slowed* or *Blocked* with nothing around, widen it 2-3 cm |
| `front.num_sectors`, `front.fov_deg` | 32, 180 | 64, 360 when the robot backs up after an inspection ( `reverse_distance_m` > 0) |
| `filter.level_source` | none | `tf` levels the band with the odometry's tilt: only on slopes, with a level odometry |
| `memory.enabled` | false | true keeps obstacles below the lowest beam for 10 s (odom frame) |

## 5.3 What the camera sees

The Mid-360 sits on top of the body and looks level: within a metre it sees nothing lower than about 40 cm, within 3 m nothing lower than about 16 cm. The RealSense D435i at the front looks 18° down and sees the floor from about 0.3 m in front of the body to 3 m ( `depth.range_max` ), a 15 cm box from about 0.15 m in front of the body. `lidar_sector_node` merges what it sees into both scans with the LiDAR's height band ( `filter.height_min` , `filter.height_max` ): a box, a bag, a low bench or a step taller than the band's bottom that the LiDAR misses stops the robot, or path shift passes it; a 10 cm step or a taped cable it walks over. The camera's point cloud comes from the robot package ( `l1w_bringup` `realsense_params.yaml` : depth 640 × 480 at 15 fps, decimated to 160 × 120 points, no texture). On the Jetson its parameters are `pointcloud__neon_.*` : librealsense names its ARM block "Pointcloud (NEON)", and `pointcloud.enable` alone publishes no cloud there.

Against ghost obstacles on shiny or tiled floors:

- **The floor is fitted in every cloud**, and heights are measured above it, not above the floor the camera's TF predicts. The mount in the URDF was measured from the floor, robot

standing (18.3° down, 1.85° rolled left; the mesh estimate was 10° and put the floor 27 cm up at 2.5 m). Once a minute the log shows how far the floor says the mount is off: `depth camera floor fitted: the mount looks this far off its TF pitch_deg=1.1 roll_deg=0.5`. A stand-up varies by about a degree; the same value for minutes of driving is a mount error: add it, in radians, to the pitch and roll of the camera's `rpy` in `l1w.urdf.xacro`. Beyond `depth.floor_max_tilt_deg` (5°) the fit is not used, the log says `nominal floor kept reason='floor points on a line ...'` and the far floor turns into obstacles: measure the mount with `python3 scripts/depth_floor.py` (robot standing, 3 m of free floor ahead, after `source scripts/env.sh`) and correct it first.

- **An obstacle needs 4 points in one 5 cm cell** (`depth.cell_points`), **and the cloud before it** must have one within `depth.confirm_radius` (0.15 m): a wrong depth on a glossy tile lasts one frame. A new obstacle shows one camera frame later (67 ms).
- Nothing farther than 3 m from the camera, where the depth error reaches about 4 cm.

The camera loses a low box when it is about 0.2 m in front of the body. So that path shift does not slide back into it while passing, the box stays in the safety scan for `depth.memory_sec` (6 s) or until the camera sees its place again. A camera that stops (USB, driver restart) leaves the scans to the LiDAR alone after 0.3 s: the log says `depth camera not merged: scans from the LiDAR alone`, and health shows `/camera/camera/depth/camera_info` stale.

The D435i belongs on USB 3. On USB 2.1 (the camera node logs `Device USB type: 2.1`) it runs, with `control_transfer returned error` warnings and now and then no cloud for more than 0.3 s (the LiDAR alone meanwhile): use a USB 3 port and a USB 3 Type-C cable.

To see it: RViz, `/perception/obstacles`, namespace `depth` (blue) is what the camera adds, `memory` (orange) what it remembers. `ros2 topic hz /camera/camera/depth/color/points` shows about 15 Hz; `python3 scripts/depth_floor.py` prints the heights the cloud shows above the floor by distance and the mount's offset from its TF.

| Parameter | Pashupati | What it does · when to change |
|---|---|---|
| `depth.enabled` | true | false: the LiDAR alone (a robot without the camera, or to compare) |
| `depth.range_max` | 3.0 | m from the camera; 2.5 when ghosts show up far ahead |
| `depth.cell_points` | 4 | points in one 5 cm cell; 6 on very shiny floors, 3 to see smaller things |

| Parameter | Pashupati | What it does · when to change |
|---|---|---|
| `depth.confirm_radius` | 0.15 | m to the obstacle in the cloud before; 0: off (one frame sooner, more ghosts) |
| `depth.memory_sec` | 6.0 | s a passed box stays beside the robot; 0: off |
| `depth.floor_fit`, `depth.floor_max_tilt_deg` | true, 5.0 | the fitted floor; a mount further off its TF is not corrected |
| `depth.timeout_sec`, `depth.required` | 0.3, false | s without a cloud before the LiDAR alone; true: every sector BLOCKED instead |

Neither sensor sees drops (stairs going down, a platform edge). Glass, black matte and mirror-like surfaces may return no depth, and right in front of the body (lower than the camera's lowest ray) and beside it the camera sees nothing.

# 6 Sites, maps and localization

## 6.1 Sites

A **site** is one place: its occupancy map, its AprilTag poses and its taught routes, in `SITES_DIR/<site>/` (`~/.pashupati/sites`). `SITE` in the env file picks the site at start (empty: the last one). `MODE=teach` maps the site from the start; drive slowly through the whole place in REMOTE, then save the map (`/mapping/save`). Taught routes go to `<site>/routes/`, routes from NETRA to `<site>/netra/`.

## 6.2 AprilTags

The tag detector (`apriltag`, from dharma_perception) must know every tag on the site's walls; localization uses the tags' frames:

```
apriltag:
  ros__parameters:
    tag:
      ids: [2, 4]
      frames: [base_map, charging_dock]
      sizes: [0.15, 0.15] # MEASURE the printed black square
localization_node:
  ros__parameters:
    tag:
      frames: [base_map, charging_dock]
```

Place tags at about the camera's height along the route, where the robot passes 1-2 m from them. Record them while localized: `/localization/record_tags` (the handheld's **Tags**) averages the visible tags over `tag.record_duration_sec` and writes the site's `tags.json`.

| `localization_node` parameter | What it does |
|---|---|
| `tag.auto_localize` | localize from the first known tag after start-up |
| `tag.auto_correct` | correct the pose from tags while patrolling |

| localization_node parameter | What it does |
|---|---|
| `tag.max_distance_m`, `tag.max_view_angle_deg` | tags further away or seen at a steeper angle are ignored |
| `estimator.sigma_xy_per_sqrt_m` | how fast the position is assumed to drift between tags |

## 6.3 Odometry

FAST-LIO2 ( `fastlio_mapping_node` ) starts once the robot stands still. `odometry_gate` accepts a standing quadruped's lean ( `max_tilt_deg: 8` ), and `odometry_monitor_node` reports a tilted odometry only beyond that ( `tilt.max_deg: 9` ). The on-line extrinsic estimation stays off ( `mapping.extrinsic_est_en: false` , Dharma's default): on the L1W in the open mall it let the odometry tilt and drift in z.

`odometry_monitor_node` also counts the trip meter of the handheld (distance, moving time, speeds) from the odometry, in whole 5 cm steps ( `trip.step_m` ) so that a standing robot adds nothing, and keeps it in `trip.file` ( `~/.pashupati/trip.json` ) across restarts. A jump or a restart of the odometry is never counted as distance.

# 7 Inspection and people

## 7.1 Pace and queue

`anomaly_detector_node` sends camera images to a vision-language model (Ollama, on the robot) and gets back one finding type or *none*.

| Parameter | Default | What it does · when to change |
|---|---|---|
| `enabled` | true | live; off also unloads the model from memory and answers the stop-point hook at once |
| `interval_sec` | 0 | live: 0 continuous, > 0 one image every N s, < 0 only at stop points |
| `queue_max` | 50 | interval pace: images waiting for the model; when full, the oldest is dropped |
| `queue_min_change` | 12 | grey levels the picture must change to be queued again; 0 = every image |
| `queue_dir` | `~/.pashupati/inspection_queue` | waiting images survive a restart; "" = memory only |
| `cooldown_sec` | 60 | s; continuous and interval pace do not report the same type again within this time |
| `max_image_age_sec` | 2.0 | s; older camera images are not inspected |
| `max_image_width` | 640 | px; images are shrunk first |
| `ollama_url` | `http://127.0.0.1:11434` | the Ollama server ( `OLLAMA_URL` in the env file) |

## 7.2 Model and memory

| Parameter | Default | What it does |
|---|---|---|
| `model` | qwen3-vl:2b-instruct | live; the model in use |
| `models` | three models | offered on the handheld (plus any other vision model in Ollama) |
| `keep_alive` | "-1" | keep the model loaded: "-1" forever, or e.g. "30m" (quoted) |
| `num_ctx` | 2048 | tokens of context; many long descriptions: 4096 (more memory) |
| `timeout_sec` | 30 | s, longest wait for one answer |
| `use_schema` | true | structured answers; false only for models that cannot do them |

A model must be pulled before it can be used ( `ollama pull <model>` , or `scripts/setup.sh install ollama` with `VLM_EXTRA_MODELS` ). `setup.sh install ollama` also turns off Ollama's prompt cache, which otherwise grows to several GB of RAM.

## 7.3 Finding types

| Parameter | Default | What it does |
|---|---|---|
| `categories` | [trash, spill, stain] | live; the types the model chooses from (plus *none*), up to 12 |
| `alarm_categories` | [trash, spill] | live; only these are reported to NETRA |
| `category_descriptions` | [""] | live; own text per type, `"name: what to look for"` |

`trash` , `spill` , `stain` , `floor_damage` and `fallen_person` have built-in descriptions. Good descriptions are short and concrete, name what is visible, and say what is **not** a finding (*not shadows, reflections or floor patterns*). `fleet_uplink_node` sends findings with at least `anomaly_min_confidence` (0.5).

## 7.4 People counting

`person_detector_node` finds people with YOLO11n (`PEOPLE=true`; the handheld switches it on and off). It places people on the floor `robot.base_height` below `base_link`.

| Parameter | Default | What it does · when to change |
|---|---|---|
| `enabled` | true | live |
| `confidence` | 0.4 | live, 0-1; boxes on things that are not people: raise it |
| `rate_hz` | 0 | live; detections per second, 0 = 5 on the GPU, 2 on the CPU |
| `max_range` | 12 | m; people further away are not placed on the map |
| `backend` | auto | `onnxruntime`, `opencv` or `ultralytics` |

NETRA's crowd map uses `fleet_uplink_node` `crowd_cell_size` (1 m) and `crowd_window_sec` (60).

# 8 Handheld, camera, video, NETRA, health and time

## 8.1 Handheld link

`teleop_udp_node`, read at start:

| Parameter | Default | What it does |
|---|---|---|
| `token` | "" | shared secret; the same in the handheld. **Always set it** |
| `robot_name` | "" | name on the handheld's start screen (empty: the computer's name) |
| `port` | 9870 | UDP port of the link and of robot discovery |
| `client_timeout_sec` | 2.0 | s; a silent handheld is dropped |
| `robot_commands` | [stand_up, sit_down, emergency_stop] | robot actions on the handheld: `/<ROBOT_NS>/<name>` services |
| `patrol_speed_min`, `patrol_speed_max` | 0.1, 0.6 | m/s range of the patrol speed slider |
| `lidar_hz`, `lidar_voxel`, `lidar_max_points` | 2, 0.1, 3000 | LiDAR points on the handheld's map |
| `internet_target` | 8.8.8.8 | pinged for the internet check (Google DNS); "" turns it off |
| `internet_period_sec`, `internet_timeout_sec` | 5, 2 | s between pings, s a ping may take |

The internet check pings `internet_target` through the robot's modem every `internet_period_sec`: the handheld shows *Internet 38 ms*, or *Internet offline* in red after two pings in a row got no answer, and `pmonitor` an *Internet* line (`internet/status`). A host name instead of the address tests the modem's DNS too.

## 8.2 Camera source

`CAMERA=driver` uses the camera of the robot package ( `CAMERA_TOPICS` , `/camera/camera/color` ). `CAMERA=rtsp` starts `rtsp_camera` for an IP camera at `CAMERA_URL` , published on the same topics: `calibration_file` (a camera_calibration YAML, see `config/calibration.example.yaml` ) rectifies the image; without one, `default_hfov_deg` (90) gives an approximate `camera_info` , enough for inspection and video but not for exact tag poses.

## 8.3 Video

The handheld gets H.264 over WebRTC from `video_stream_node` , started while it watches: `bitrate_kbps` (1500; weak Wi-Fi: 800), `fps` (15), `max_width` (1280), `keyframe_sec` (1.0), `encoder` (auto). Without WebRTC ( `WEBRTC=false` or no MediaMTX) the handheld gets JPEG frames: `teleop_udp_node` `camera_fps` (8), `camera_jpeg_quality` (60), `camera_max_width` (640).

## 8.4 NETRA uplink

`fleet_uplink_node` : `server_url` ( `CC_URL` ), `token` ( `CC_TOKEN` ), `robot_id` ( `ROBOT_ID` wins), `telemetry_hz` (1.0), `upload_period_sec` (5.0), `verify_tls` (true; keep it on), `spool_dir` ( `~/.pashupati/spool` : findings and routes wait there while NETRA is unreachable), `clock_sync_period_sec` (10: how often the offset to NETRA's clock is measured) and `clock_warn_ms` (200: a warning in the log above it), see section 8.6.

## 8.5 Health monitor

`health_monitor_node` publishes `/health/status` (level, issues, speed limit), shown on the handheld and in NETRA. `control_mux_node` scales every command by its speed limit.

| Parameter | Pashupati | Issue when |
|---|---|---|
| `sensors.topics` | LiDAR 5, odometry 8, sectors 5, camera_info 5, robot odometry 10 | a topic runs slower ( `"topic:Hz[:max_age]"` ) |

| Parameter | Pashupati | Issue when |
|---|---|---|
| `degradation.rules` | LiDAR, odometry or TF stale: speed × 0.5 | a degraded part slows the robot |
| `nodes.expected` | empty | a listed node is missing (add the driver's nodes) |
| `system.max_temp_c`, `system.max_memory_percent`, `system.min_disk_free_gb` | 85, 92, 5 | the computer is hot, full or out of disk |
| `startup_grace_sec` | 30 | s after start without missing-node issues |
| `clock.enabled`, `clock.max_offset_sec` | true (false in the simulator), 0.05 | the clock is off its time source ( `clock:ntp` ) |

## 8.6 Time synchronisation

Every stamp must come from one clock: the LiDAR's points, the IMU, the camera, the odometry, TF and the events sent to NETRA. Two clocks that differ by a few hundred milliseconds already make TF "too old" (the robot stops, `TF_STALE` ), make the odometry jump, and put findings and patrol events at the wrong time in NETRA.

| Clock | Kept by | Check |
|---|---|---|
| Robot computer | chrony ( `scripts/setup.sh install timesync` ): follows `TIME_SERVERS` , serves `TIME_ALLOW` | `scripts/timesync.sh check` , `chronyc tracking` ; health `clock:ntp` |
| LiDAR (MID-360) | PTP ( `scripts/setup.sh install ptp` ): `ptp4l` as master on the LiDAR's interface (the route to `LIVOX_IP` , or `PTP_IFACE` ), service `lidar-ptp` | `systemctl status lidar-ptp` ; no "in the future" or "steady" issue on `/livox/lidar` |
| Robot motion computer | its own; Pashupati stamps the driver's messages on arrival | |

| Clock | Kept by | Check |
|---|---|---|
| NETRA | the server's NTP | `telemetry.clock.offset_ms` in NETRA; *clock differs from NETRA* in the log |

Set it up once per robot, in `/etc/pashupati/pashupati.env`:

```
TIME_SERVERS=netra.example.com   # NETRA's host, or the NTP server NETRA
uses; empty: Ubuntu's pool
TIME_ALLOW=                      # subnets of other computers on the
robot that sync to this one
LIVOX_IP=192.168.123.20
PTP_IFACE=                       # empty: the interface that routes to
LIVOX_IP
```

Then `scripts/setup.sh install timesync ptp` (both are in the default install) and `scripts/timesync.sh check`:

```
chrony: offset 0.000231 s to C0A87B01 (netra.example.com)
ptp: PTP master on eth1 (software timestamps)
```

The offset must stay below 0.05 s (`TIME_MAX_OFFSET`); with a time server on the same network it is below a millisecond. At boot the stack waits up to `TIME_SYNC_WAIT` (60 s) for the first synchronisation, so that the clock does not jump under a running stack.

**The offset to NETRA.** The uplink sends `time_sync` after connecting and every `clock_sync_period_sec`; NETRA answers with its own times and the robot computes the offset like NTP. It is in every telemetry message as `clock` {offset_ms, rtt_ms, age_sec} (null while NETRA does not answer), so NETRA can place the robot's events on its own clock. Above `clock_warn_ms` the robot logs *clock differs from NETRA*: fix the time source, not the parameter.

## 8.7 Console monitor

`pmonitor` (or `scripts/console.sh monitor`) shows the whole robot in one terminal, over SSH too (a phone's SSH app as well): one line per part with `[ OK ]`, `[WARN]` or `[ERR ]` and its worst problem: Robot (mode, e-stop, protective field, battery), Navigation, Localization, Health (every check with its rate or age, instead of `ros2 topic hz`), System, Internet, Inspection, People and the newest warnings of every node. A tap on a line (or

`<Enter>`) shows it in detail, `[Back]` returns; System's detail lists the busiest processes (instead of `top`). CPU is averaged over 3 s; the load (processes running or waiting, over a minute) is shown against the cores: below their number nothing waits. `[Menuconfig]` switches to menuconfig. `[Report]` writes a report file to `~/.pashupati/reports/`: every panel, the log since the monitor started, the commit, the service state and its last 300 log lines. Ask an operator for a screenshot of the monitor or for that file when the robot misbehaves; `pmonitor --once > report.txt` writes it without the screen.

# 9 The env file

`/etc/pashupati/pashupati.env` says what runs. Restart the service after a change. Quote values with spaces, and never put a comment after a value.

| Key | Genisom L1W | Meaning |
|---|---|---|
| `ROBOT_ID` | e.g. `l1w-01` | the robot's name in NETRA |
| `ROBOT_DRIVER` | `l1w_bringup` | robot package started with the stack (was `ROBOT_BRINGUP`) |
| `ROBOT_PARAMS` | empty | robot profile; empty: `config/robot.yaml` |
| `ROBOT_NS` | `l1w` | namespace of the driver's topics and services |
| `ROBOT_PRESTART` | `"/usr/local/bin/robot-launch stop driver-livox forward"` | stops the vendor's own LiDAR driver and motion stack first |
| `ROBOT_SETUP`, `ROBOT_IP` | the robot package's installer, `192.168.234.1` | the ZSBot SDK, the network check |
| `MODE` | `patrol` | `teach`: map `SITE` from the start |
| `ODOMETRY` | `fast_lio` | or `leg_kilo` |
| `SITE`, `SITES_DIR` | the mall's site, `~/.pashupati/sites` | the site at start (was `ROUTE` and `TAGS`) |
| `CAMERA`, `CAMERA_URL`, `CAMERA_TOPICS` | `driver`, empty, `/camera/camera/color` | section 8.2 |
| `ROS_DOMAIN_ID`, `RMW_IMPLEMENTATION` | `0`, `rmw_fastrtps_cpp` | Pashupati's own DDS; the vendor stack uses another |
| `ROS_LOCALHOST_ONLY` | `1` | `0` only for RViz from a laptop (section 1.3) |

| Key | Genisom L1W | Meaning |
|---|---|---|
| `PEOPLE`, `INSPECTION`, `OLLAMA_URL` | `true`, `true`, empty | people counting, the VLM inspection |
| `SITE_CONFIG` | `/etc/pashupati/site.yaml` | the site file |
| `WEBRTC`, `MEDIAMTX` | `true`, empty | WebRTC video for the handheld |
| `CC_URL`, `CC_TOKEN` | NETRA's address, the robot's token | NETRA uplink; empty: findings are kept on the robot |
| `TIME_SYNC_WAIT` | `60` | s to wait at boot for the network clock |
| `TIME_SERVERS`, `TIME_ALLOW` | empty (Ubuntu's pool), empty | chrony's time sources, the subnets it serves (section 8.6) |
| `LIVOX_IP`, `PTP_IFACE` | `192.168.123.20`, empty (the route to `LIVOX_IP`) | the LiDAR follows this computer's clock over PTP |
| `OPENBLAS_NUM_THREADS`, `OMP_NUM_THREADS` | `1` | keep at 1: more threads waste CPU |

`scripts/setup.sh check` reports a missing key, or a line that breaks the rules above.

# 10 Symptoms and fixes

| Symptom | Look at | Try |
|---|---|---|
| Weaves left and right on straight parts | `speed.smoothing`, the `holonomic.*` section | smoothing 0.3 |
| Too slow in gentle curves | `speed.lateral_accel` | 0.4 |
| Jerky starts and stops | `speed.accel`, `speed.decel` | 0.3, 0.2 |
| Walks slower or faster than the patrol speed | `stick_max_*` (section 2.3), speed gear | rescale; patrol in the checked gear |
| Turns or walks sideways the wrong way | `stick_invert_yaw`, `stick_invert_lateral` | true |
| Stops just short of a stop point and waits 20 s | `approach.min_speed`, `approach.position_tolerance` | 0.06, 0.08 |
| Leaves stop points too late | dwell time of the route, `hooks.inspection_timeout_sec`, pace | shorter dwell, faster model |
| *Slowed* or *Blocked* with nothing around | `filter.self_margin`, `filter.height_min`, floor slopes | margin +0.02; height_min +0.03 |
| Ignores low objects (bags, boxes) | `filter.height_min`, the camera (section 5.3) | lower 0.02 at a time (a 10 cm step then stops it), check the floor is not seen |
| Stops for a 10 cm step or a taped cable | `filter.height_min` in the site file, the mount offset in the log (5.3) | 0.12 (5.2); correct the mount |

| Symptom | Look at | Try |
|---|---|---|
| Stops or swerves for nothing right ahead | blue `depth` points in RViz, the mount offset in the log (5.3) | clean the camera; correct the mount; cell_points 6 |
| Ghost obstacles far ahead, only with the camera | the mount offset in the log, `depth.range_max` | correct the mount; range_max 2.5 |
| `nominal floor kept ... on a line` every minute | the camera mount, more than 5° off its TF (5.3) | `scripts/depth_floor.py`, correct the URDF |
| `depth camera not merged ... no cloud for` now and then | the camera on USB 2.1 (5.3) | a USB 3 port and cable |
| Stops far before obstacles | `path_shift.stop_margin`, `field.margin`, `robot.limits.max_decel` | small steps; repeat the test of section 3.3 |
| Gets too close to obstacles | `robot.limits.max_decel`, `robot.limits.reaction_time` | measure the braking again (section 3.2) |
| Does not pass gaps that look wide enough | the three margins and the cone (5.1), `path_shift.max_offset`, the footprint | 0.05 each, cone 40; larger offset |
| Waits long in front of a person or a box | `path_shift.patience_sec`, `path_shift.search_offset` | 2.0; 2.0 where the aisle is wide |
| Goes far around where it should wait | `path_shift.search_offset`, `path_shift.patience_sec` | 0.8 turns the search off |
| Leaves the route in corridors wider than needed | `path_shift.centre_room` | 0.15; 0.05 (= clearance) turns it off |

| Symptom | Look at | Try |
|---|---|---|
| Turns toward gaps too often in a crowd | `detour.after_sec` | 2.5-3; `detour.enabled: false` turns it off |
| Waits in a tight spot it could turn out of | `detour.after_sec`, `detour.max_angle_deg` | 1.0; 90 |
| *Blocked* too soon or too late | `safety.blocked_timeout_sec` | to taste |
| **Start** refused: robot too far from the route | `join.max_distance` | drive closer, or raise it |
| **Start** refused: not localized | a tag in view, `/localization/start` | **Localize** on the handheld |
| *No loc*: tag not found | `apriltag` `tag.sizes`, `tag.ids`, light, `detector.decimate` | measure the tag, decimate 1.0 |
| Same trash reported again and again | `cooldown_sec` | 120-300 |
| False findings | type description, `alarm_categories`, `anomaly_min_confidence` | describe what is not a finding; 0.6-0.7 |
| Inspection queue always full | `interval_sec`, `model`, `max_image_width` | longer interval, smaller model |
| Robot slow everywhere, health *warn* | `/health/status` issues, `degradation.rules` | fix the stale sensor |
| Video stutters | `bitrate_kbps`, `fps` | 800, 10 |
| Handheld map lags | `lidar_hz`, `lidar_voxel`, `lidar_max_points` | 1, 0.2, 1500 |
| Findings missing in NETRA | `CC_URL`, `CC_TOKEN`, `alarm_categories`, `anomaly_min_confidence` | the log of `fleet_uplink_node` |

| Symptom | Look at | Try |
|---|---|---|
| Robot stops with `TF_STALE`, odometry jumps | `/health/status` (`clock:ntp`, `/livox/lidar` "in the future") | `scripts/timesync.sh check` (section 8.6) |
| Events in NETRA at the wrong time | `telemetry.clock.offset_ms`, *clock differs from NETRA* in the log | `TIME_SERVERS`, `scripts/setup.sh install timesync` |

# 11 Site file for the Genisom L1W

A starting point for `/etc/pashupati/site.yaml` (the repository's `src/pashupati/config/site.example.yaml` is the same). The robot itself is the robot profile; the site file holds this robot's token and actions and this site's values.

```
teleop_udp_node:
  ros__parameters:
    token: "change-me" # the same in the handheld
    robot_commands:
      [
        stand_up,
        sit_down,
        balance_stand_mode,
        move_mode,
        lock_mode,
        slow_speed,
        normal_speed,
        fast_speed,
        sdk_control_right,
        emergency_stop,
      ]

apriltag: # the tags on this site's walls
  ros__parameters:
    tag:
      ids: [2, 4]
      frames: [base_map, charging_dock]
      sizes: [0.15, 0.15] # MEASURE the printed square

path_follower_node:
  ros__parameters:
    speed:
      cruise: 0.35 # patrol speed for this site
    low_battery: 0.25

fleet_uplink_node:
  ros__parameters:
    robot_id: l1w-01 # ROBOT_ID in the env file wins
```

Each node may appear only once in the file: add more parameters under its existing heading. After a restart, `ros2 param dump /control_mux_node` (and so on) shows the values in use.

# 12 Parameter reference

The parameters Pashupati sets, read from its parameter files of this version (the robot profile, the stack file, and the defaults of the Pashupati packages). *Meaning* is the comment in the file. Every other Dharma parameter keeps its default: `ros2 param describe /<node> <name>`, and the Dharma manual lists them all.

## 12.1 pashupati

robot profile: footprint, limits, base height · `src/pashupati/config/robot.yaml`

```
/**
```

| Parameter | Default | Meaning |
|---|---|---|
| `robot.name` | `quadruped` | model, for logs |
| `robot.kinematics` | `omni` | legged: vx, vy, wz; diff for a robot that cannot strafe (max_vy 0) |
| `robot.base_height` | `0.35` | m, base_link above the floor when standing |
| `robot.footprint.front` | `0.40` | |
| `robot.footprint.rear` | `0.40` | |
| `robot.footprint.half_width` | `0.28` | |
| `robot.limits.max_vx` | `1.0` | m/s, nothing may command more |
| `robot.limits.max_vy` | `0.5` | m/s |
| `robot.limits.max_wz` | `1.5` | rad/s |
| `robot.limits.max_accel` | `0.5` | m/s^2 |
| `robot.limits.max_decel` | `0.6` | m/s^2, MEASURE: braking from full speed on the site's floor |
| `robot.limits.max_yaw_accel` | `1.5` | rad/s^2 |
| `robot.limits.reaction_time` | `0.25` | s, command to the robot actually braking (sensing + transport) |
| `robot.frames.map` | `map` | |

| Parameter | Default | Meaning |
|---|---|---|
| `robot.frames.odom` | `odom` | |
| `robot.frames.base` | `base_link` | |

## 12.2 pashupati

the Dharma stack as tuned for Pashupati · `src/pashupati/config/stack.yaml`

### odometry_gate

| Parameter | Default | Meaning |
|---|---|---|
| `max_tilt_deg` | `8.0` | a standing quadruped leans a little; catches a sitting robot |

### odometry_monitor_node

| Parameter | Default | Meaning |
|---|---|---|
| `trip.file` | `~/.pashupati/trip.json` | with the robot's other state; reset from the handheld |
| `tilt.max_deg` | `9.0` | FAST-LIO's frame starts as tilted as the IMU (it does not level to gravity): a start the gate accepts (up to 8 deg) is no drift; only more than that counts as one |

### lidar_sector_node

| Parameter | Default | Meaning |
|---|---|---|
| `safety.num_sectors` | `144` | 2.5 deg sectors: the sides of a narrow gap where they really are |
| `filter.height_min` | `0.12` | m above the floor: lower is stepped over (10 cm steps, taped cables) |
| `filter.height_max` | `1.55` | m above the floor (base_link + 1.20 m) |

| Parameter | Default | Meaning |
|---|---|---|
| `filter.level_source` | `none` | As Pashupati 2.0: a fixed height band on base_link. Leveling with the odometry's tilt turned a slightly tilted odometry into ghost obstacles (the floor a metre or two away): the robot stopped and weaved around nothing. The mall floor is flat. |
| `memory.enabled` | `false` | as 2.0: remembered points (odom frame, 10 s) kept those ghosts around |
| `depth.enabled` | `true` | the camera's low obstacles join both scans; false: the LiDAR alone |
| `depth.topic` | `/camera/camera/depth/color/points` | PointCloud2 of realsense2_camera |
| `depth.memory_sec` | `6.0` | s, a low box beside the robot (out of the camera's view) is kept |

## `localization_node`

| Parameter | Default | Meaning |
|---|---|---|
| `tag.frames` | `[base_map, charging_dock]` | the AprilTag frames (dharma_perception tag.frames) |

## `path_loader_node`

| Parameter | Default | Meaning |
|---|---|---|
| `remember_file` | `~/.pashupati/last_route` | the route loaded last comes back after a restart |

## `path_follower_node`

| Parameter | Default | Meaning |
|---|---|---|
| `loop_route` | `true` | patrol: a closed route starts its next lap (not on low battery) |
| `speed.accel` | `0.5` | m/s^2 (Pashupati 2.0 linear_accel_limit) |

| Parameter | Default | Meaning |
|---|---|---|
| `speed.smoothing` | `0.4` | |
| `path_shift.clearance` | `0.05` | m kept beside the body in a gap |
| `safety.blocked_timeout_sec` | `30.0` | s standing in front of an obstacle before BLOCKED |
| `safety.lateral_margin` | `0.05` | m, >= control_mux_node field.lateral_margin |
| `hooks.inspection` | `[/anomaly_detector_node/check_now]` | Inspection at every stop point: the VLM check answers (or, switched off, says so at once); a hook whose node does not run is skipped, so INSPECTION=false never blocks. |
| `hooks.inspection_timeout_sec` | `20.0` | VLM inference on the Jetson takes a few seconds |
| `inspection.reverse_distance_m` | `0.0` | > 0: back up this far after the inspection (360 deg scan) |

## `control_mux_node`

| Parameter | Default | Meaning |
|---|---|---|
| `use_health_limit` | `true` | scale commands by health/status speed_scale |
| `remote_sources` | `[handheld, dashboard]` | control/remote/<source>/cmd_vel |
| `field.in_remote` | `false` | REMOTE without the protective field: the operator watches the robot (field-tested decision of Pashupati 1.0); the envelope and the e-stop still apply. sim.yaml keeps it on. |
| `field.lateral_margin` | `0.05` | m, extra width of the swept path (the follower's is not below) |

## obstacle_avoidance_node

| Parameter | Default | Meaning |
|---|---|---|
| shared.emergency_cone_deg | 40.0 | +-20 deg: in Dharma's 60 deg emergency cone the sides of a gap narrower than about 0.7 m stood within 0.75 m and held the robot in front of it. The guard and the field check the whole swept body. |

## docking_node

| Parameter | Default | Meaning |
|---|---|---|
| hooks.sit | [""] | e.g. /robot/sit |
| hooks.charge | [""] | e.g. /robot/charge/start |
| hooks.stop_charge | [""] | |
| hooks.stand | [""] | |

## health_monitor_node

| Parameter | Default | Meaning |
|---|---|---|
| sensors.topics | [/livox/lidar:5.0, /Odometry:8.0, /perception/sector_scan:5.0, /camera/camera/color/camera_info:5.0:0, /camera/camera/depth/camera_info:5.0:0, /{robot_ns}/odom:10.0] | "topic:min_hz[:max_age_sec]"; {robot_ns} is the driver namespace. The camera on; camera_info: subscribing the JPEG topic would make the driver encode every frame. |

| Parameter | Default | Meaning |
|---|---|---|
| `degradation.rules` | `["sensor:/livox/lidar=0.5", "sensor:/Odometry=0.5", "tf:*=0.5"]` | |
| `clock.enabled` | `true` | the offset of this computer's clock to its time source (scripts/setup.sh install timesync) |

### `teleop_udp_node`

| Parameter | Default | Meaning |
|---|---|---|
| `token` | `""` | set it in the site file, and the same in the handheld |

## 12.3 pashupati_bridge

handheld link, video, RTSP camera · `src/pashupati_bridge/config/bridge.yaml`

### `teleop_udp_node`

| Parameter | Default | Meaning |
|---|---|---|
| `token` | `""` | shared secret: set it in the site file and the same in the handheld app |
| `robot_commands` | `[stand_up, sit_down, emergency_stop]` | Commands shown on the handheld, each a std_srvs/Trigger <robot_command_prefix><name> of the robot driver (the bringup sets the prefix to /<robot_ns>/); add the robot's own in the site file (gait, speed modes, ...). |
| `sites_dir` | `~/dharma/map` | the handheld lists and loads the taught routes of every site here |

### video_stream_node

| Parameter | Default | Meaning |
|---|---|---|
| fps | 15.0 | video frames per second |
| bitrate_kbps | 1500 | weak Wi-Fi: lower it |

### rtsp_camera

| Parameter | Default | Meaning |
|---|---|---|
| url | "" | the bringup sets camera_url:=; e.g. rtsp://192.168.1.120:8554/test |
| calibration_file | "" | camera_calibration YAML (config/calibration.example.yaml): rectified |
| default_hfov_deg | 90.0 | camera_info without a calibration (tag poses approximate) |

## 12.4 pashupati_inspection

VLM inspection, people · src/pashupati_inspection/config/inspection.yaml

### anomaly_detector_node

| Parameter | Default | Meaning |
|---|---|---|
| model | qwen3-vl:2b-instruct | any Ollama vision model; the handheld offers `models` |
| categories | [trash, spill, stain] | a-z 0-9 _, the model also answers "none" |
| alarm_categories | [trash, spill] | reported to NETRA; stain is only shown (shiny floors) |
| category_descriptions | [""] | own text per type, "name: what to look for"; trash, spill, stain, floor_damage and fallen_person have built-in texts |
| keep_alive | "-1" | keep the model loaded while the node runs ("-1"), or e.g. "30m" |

| Parameter | Default | Meaning |
|---|---|---|
| `interval_sec` | `0.0` | 0 back to back, > 0 every N s through the queue, < 0 stop points only |
| `queue_dir` | `~/.pashupati/inspection_queue` | interval frames kept across restarts |

### `person_detector_node`

| Parameter | Default | Meaning |
|---|---|---|
| `model_path` | `~/dev/Pashupati/models/yolo11n-416.onnx` | YOLO11n exported to ONNX at 416 px (scripts/setup.sh install people_model): runs without PyTorch (ONNX Runtime, CUDA when available, else OpenCV); .pt / .engine use Ultralytics. |
| `rate_hz` | `0.0` | live: detections per second; 0 = 5 on the GPU, 2 on the CPU |

## 12.5 pashupati_fleet

NETRA uplink · `src/pashupati_fleet/config/fleet.yaml`

### `fleet_uplink_node`

| Parameter | Default | Meaning |
|---|---|---|
| `server_url` | `""` | NETRA, e.g. https://netra.example.com (empty: spool only); CC_URL |
| `token` | `""` | robot token NETRA issued (Pengaturan -> Robot -> Tambah); CC_TOKEN |
| `robot_id` | `robot-01` | ROBOT_ID |
| `spool_dir` | `~/.pashupati/spool` | anomalies and taught routes wait here until NETRA has them |
| `sites_dir` | `~/dharma/map` | patrol routes from NETRA go to <site>/netra/<route_id>.csv |

| Parameter | Default | Meaning |
|---|---|---|
| `clock_sync_period_sec` | `10.0` | The offset to NETRA's clock, measured like NTP (time_sync), in telemetry.clock; a warning when it exceeds clock_warn_ms (the fix: scripts/setup.sh install timesync) |
| `clock_warn_ms` | `200.0` | |