IFARLAB — Intelligent Factory and Robotics Laboratory

Eskişehir Osmangazi University

Automotive Surface Inspection — Vehicle Doors

Coverage planning, quality reporting and operator-driven defect handover

Table 1: Scenario at a glance.

Item	Value
Task	Optical surface inspection of two vehicle doors, both faces
Robots	UR10e on a Festo linear axis + Kawasaki RS005L on a rail-guided AGV
Sensors	Two SICK Visionary-T Mini time-of-flight cameras, one per wrist
Planned coverage	95.7 % of the sampled surface in 12 viewpoints (9 UR10e + 3 Kawasaki)
Measured coverage	94.0 % in simulation; 62.9 % on the physical cell (see Section 6)
Operator interface	Web dashboard: start, confirm, re-inspect, stop; live defect table
Maturity	Complete scan-to-cleaning chain demonstrated; real-cell coverage limited by a measured part-placement deviation

1. The industrial problem

Surface quality control on body panels is a task where the inspection itself is easy to automate and the reporting is not. A robot can be driven over a door in a few minutes; the difficult questions are which fraction of the surface was really measured, where the remaining gaps are, and what happens to a defect once it has been found.

This use case answers all three. The same planning and execution pipeline that inspects a bus chassis is applied, without modification, to a pair of vehicle doors; the output is not a trajectory but a quality report; and the report feeds an operator dialogue that hands the confirmed defects to a cleaning pass executed by one of the two arms.

2. The part and the cell

Two doors — one with a window aperture, one without — are mounted on a platform between the two arms. The arms approach from opposite sides, so the division of work follows from reachability rather than from a hand-written rule: the surfaces facing the UR10e fall to the UR10e, the far faces fall to the Kawasaki.

Table 2: The inspected parts, measured from the CAD meshes.

Property	Window door	Windowless door	Merged target
Mesh triangles	773 109	129 462	902 571
Surface area	2.880 m²	3.941 m²	6.821 m²
Geometry	Solid, approx. 6 cm thick	Zero-thickness double-sided shell	—
Units	metres	metres	metres (planner runs at scale 1.0)

Why one mesh and not two. The two doors are concatenated into a single target mesh before planning. In one mesh

each door blocks rays to the other exactly as the real pair does, so the occlusion model is correct and one combined tour is produced instead of two independent ones.

3. What is reused and what is new

The engineering value of this use case lies as much in what was not rewritten as in what was added. The planner node, both execution nodes, the trajectory cache, the joint-unwinding and the nearest-branch inverse kinematics are the chassis-validated components, called as they are. The door package contributes only the part-specific pieces.

Table 3: Only these components are door-specific; everything else is reused.

Door-specific component	What it does
Mesh preparation	Concatenates the two door meshes into one cached target mesh, rebuilt automatically when a source file changes
Door configuration	Sensor model, standoff rings, coverage threshold, per-arm viewpoint budgets and the workspace restriction for this part
Reference-surface generation	Divides each door into a 3 × 4 patch grid sampled at 100 000 points/m², which is the ground truth of the coverage report
Quality report builder	Builds the belief and occupancy maps at 2 cm resolution and prints per-patch, per-door and overall coverage
Mission orchestrator	Owns the inspection launch as a child process, tracks progress, publishes the defects and drives the cleaning handover
Operator dashboard	Separate web application on its own port, so it can run next to the cell-wide dashboard

4. Operating sequence

Table 4: The operator sequence as implemented.

Stage	Trigger	What happens
Cell bring-up	START SYSTEM	The unified cell launch is started, with or without mock hardware
Mission start	Robots are ready	The mission launch is started with the chosen camera and replanning options
Inspection	START SENSING ROBOT	Both arms run the planned coverage tour; progress is reported per arm
Reporting	automatic	Tour completion is detected, the inspection launch is torn down and the defect list is published
Confirmation	CONFIRM	The operator accepts the defect list; the UR10e alone drives to each defect while the Kawasaki stays parked
Stop	STOP	Running trajectories are cancelled, zero velocity is awaited, and only then is the launch shut down

The stop behaviour deserves a sentence of its own. Cancelling the goals is not sufficient, because a trajectory controller continues to execute its last accepted trajectory after the client disappears. The orchestrator therefore cancels, waits until both arms report zero velocity, and only then tears the launch down.

5. Planned coverage

Table 5: Plan statistics, read from the stored plan file.

Quantity	Value
Viewpoints	12 total — 9 UR10e, 3 Kawasaki
Coverage achieved by the plan	95.65 % of sampled target points
Coverage after the first Kawasaki viewpoint	35.79 %
Coverage after four viewpoints	87.90 %
Geometric ceiling without an arm budget	100.0 % (4 959 of 4 960 target points)
Standoff rings used	1.0 / 1.2 / 1.4 / 1.6 m
Kawasaki viewpoint budget	3 (deliberate: the slower arm sets the wall-clock time)

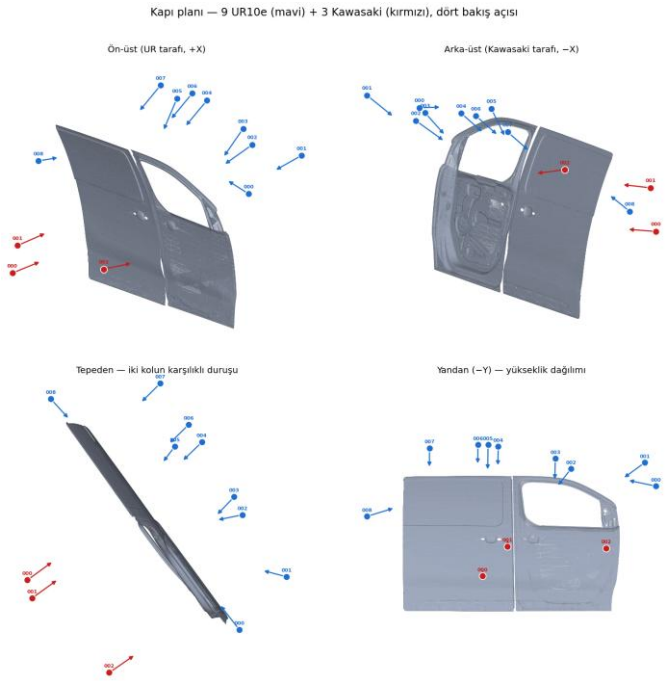


Figure 1: The planned tour from four viewpoints: 9 UR10e positions and 3 Kawasaki positions on the merged door mesh, with the view direction of each camera pose.

Capping the slower arm at three stops rather than lowering the speed of both is a deliberate scheduling choice: two arms running in parallel cost the time of the slower one, so the only effective lever is the number of stops that arm makes. On the physical cell the UR10e finished all of its stops while the Kawasaki was still on its second.

6. Measured coverage — simulation and physical cell

The quality report compares the captured clouds against the reference patch grid at 2 cm voxel resolution. In simulation the result is close to the planned value; on the physical cell it is not, and the reason was measured rather than guessed.

Table 6: Coverage measured from the occupancy maps.

Measurement	Window door	Windowless door	Total
Simulation	94.20 %	93.81 %	94.04 % (14 054 / 14 945 voxels)
Physical cell, raw	28.4 %	46.5 %	36.0 %

Physical cell, after alignment compensation	51.5 %	78.3 %	62.9 % (9 401 voxels)

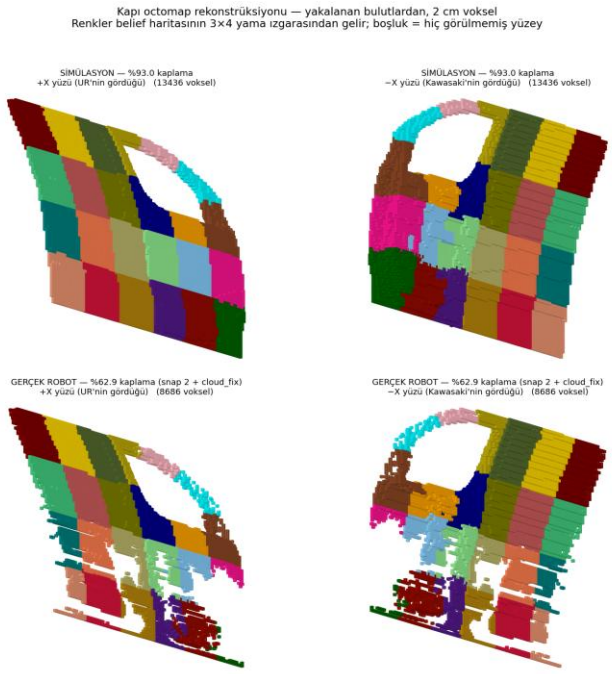


Figure 2: Occupancy reconstruction of both door faces at 2 cm voxels: simulation above, physical cell below. Colours follow the reference patch grid; grey regions were never measured.

The cause was identified by measurement. The median distance between the captured cloud and the CAD model is 3.0 mm in simulation and 37.8 mm on the physical cell. Fitting 183 740 points from twelve real captures gives a rigid transform of roughly 5° in pitch and 10 cm in translation, and reduces the residual from 41.5 mm to 18.6 mm. Both arms measure the same offset independently — the UR10e reports −81.4 mm at the bottom edge, the Kawasaki −81.5 mm — which means the doors, not the robots, deviate from the model.

What this means in practice. The gap is a fixturing problem, not a perception problem. A door fixture that holds the part within a few millimetres of its nominal pose, or an automatic part-registration step before planning, is the next step; the alignment compensation used above is a reporting correction, not a fix.

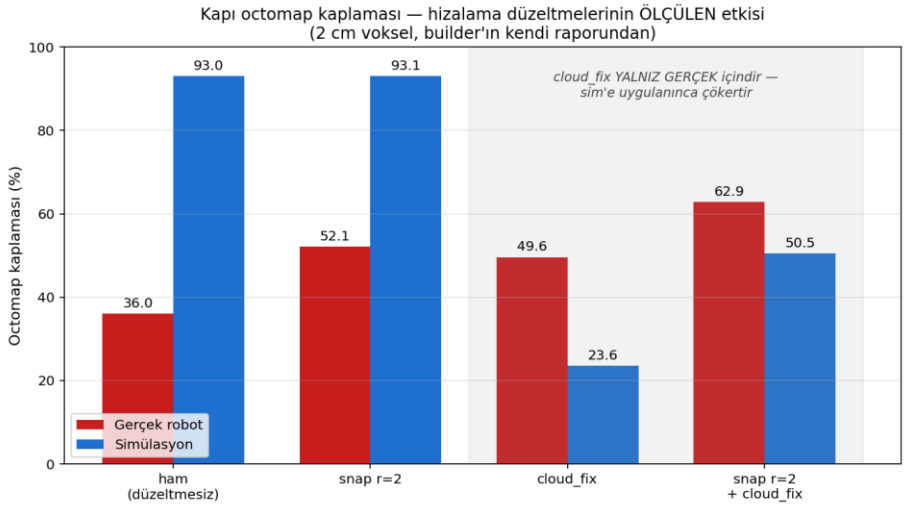


Figure 3: Measured coverage for the raw clouds and for each compensation step, simulation against the physical cell.

7. Operator interface and defect handover

The dashboard is a small web application that runs on its own port so that it can be open beside the cell-wide dashboard. It exposes exactly the decisions an operator has to make and nothing else: three toggles — mock hardware, simulated cameras only, forced replanning — and four buttons for the mission itself.

During and after the tour the panel shows the current mode, the number of defects detected, a defect table with the identifier, type and world-frame coordinates in metres, and a status per row that moves from detected to cleaning to cleaned. Live camera views from the cell and from the simulated twin sit side by side, with joint plots underneath.

Table 7: The four scenario defects and the measured positioning error of the cleaning pass.

Defect	Type	Position (x, y, z) in m	Measured approach error
DEF-01	Protrusion	−1.100, 0.800, 1.520	61.5 mm — at the reach limit
DEF-02	Protrusion	−1.100, 0.300, 1.530	68.3 mm — at the reach limit
DEF-03	Dent	−0.918, 0.274, 1.040	1.1 mm
DEF-04	Dent	−0.892, 1.680, 1.000	1.0 mm

The defect coordinates are not redefined by the dashboard; the mission node imports them from the same definition the cleaning scenario uses, so the two cannot drift apart. The measured approach errors show the honest picture: two defects are reached to about one millimetre, and two sit high on the door where the arm runs out of reach by roughly six centimetres.

Timing from a recorded session: from the start command to the published defect list took 1 min 45 s, and the cleaning pass over four defects took 69 s.

8. Current limitations

- The physical doors deviate from the CAD model by about 5°, which costs roughly 30 points of measured coverage. Alignment compensation in the report recovers part of it but does not correct the capture.
- Two of the four defect positions lie at the edge of the arm’s reach, with a measured shortfall of about 6 cm.
- Cached trajectories are replayed without re-checking collisions; a forced replanning run is required after any change to the collision model.
- The coverage figure of a zero-thickness part counts a voxel as covered when either face has been seen, so that door reads slightly optimistically.
- The plan currently stored on disk predates the latest sensor-distance settings; a fresh planning run is the first step before any new measurement campaign.

9. Running the scenario

Each stage is a separate launch, so a plan can be reused across many inspection runs.

```
# Plan (merges the two door meshes, then waits for the service call)
ros2 launch doors_inspection doors_planning.launch.py
ros2 service call /doors_planner_node/plan std_srvs/srv/Trigger

# Execute on the physical cell
ros2 launch doors_inspection doors_inspection.launch.py only_sim:=false

# Build the quality report from the captured clouds
ros2 launch doors_inspection doors_octomap.launch.py mode:=real

# Operator mission (dashboard drives this one)
```

```
ros2 launch doors_inspection doors_mission.launch.py
```