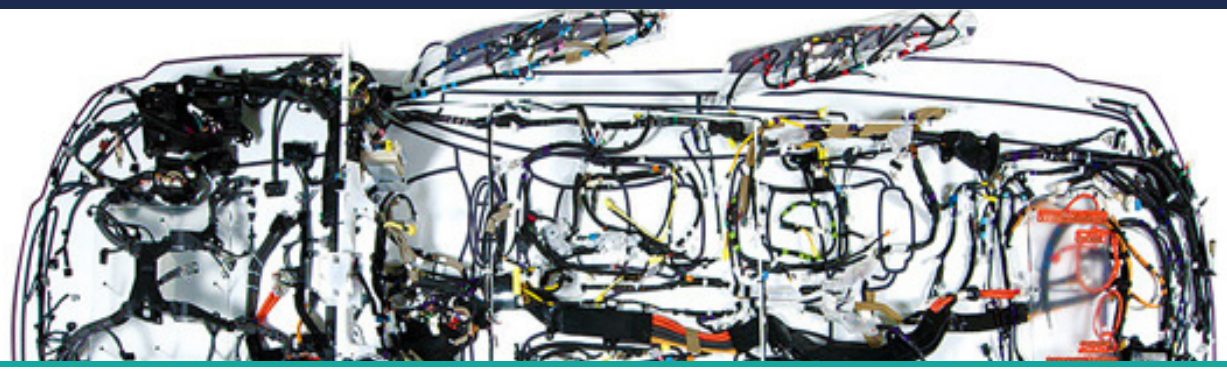


CASE STUDY · AUTOMOTIVE MANUFACTURER

Wire Harness Manipulation and Installation

A single control model handling arbitrary harness poses, no per-scenario programming for an F100 automotive manufacturer.

6 of 6	~2 ms	< 200 KB
SCENARIOS SOLVED, GRASP TO INSERTION	ONLINE BELIEF UPDATE, NO RETRAINING	MODEL BUILT FROM 30 S OF DATA

1 SUMMARY

Wire harness automation does not fail on any one case. It fails on the long tail: each exception is solvable, but innumerable, and the engineering cost never amortises. A Fortune 100 automotive manufacturer hired ThoughtForge to test whether a single model could absorb large parts of that tail instead of handling it case-by-case. One UR5e arm, one fixed RGB-D camera, and a ThoughtForge controller took harness connectors from arbitrary poses, staged them, reoriented them where required, and mated them under force control, every scenario on one unified model with no scenario-specific tuning.

The controller stores no trajectories and no hand-written exception handlers. It carries a running set of expectations about what a correct grasp and insertion should look and feel like, and updates them from vision and force/torque feedback roughly every 2 milliseconds. Because adaptation happens online, the exceptions do not have to be enumerated in advance: the same model extends to new harness variants, new connector geometries, and situations it has never seen, without retraining or reprogramming.

2 PROBLEM

Conventional industrial systems perform well when geometry is fixed and the environment is tightly controlled. Inverse kinematics and preprogrammed trajectories are reliable when parts are where they are expected to be. Wire harnesses are not: a harness is a deformable linear object with effectively infinite degrees of freedom, so it bends, twists, folds, and coils under its own weight, and small disturbances produce large changes in shape. Connectors arrive face down, rotated to arbitrary yaw, tangled with neighbouring cables, or with the mating face hidden by the harness itself.

Any one of those situations can be programmed around. The difficulty is the tail: each new variant, connector family, and cable combination produces exceptions nobody enumerated, and each one costs engineering time to find, hard-code as another waypoint or exception handler, and maintain thereafter. The system grows more brittle as the handlers accumulate, and the work never finishes. The alternative, a generalised machine learning model, moves that cost rather than removing it: it requires millions of representative samples, delivers accuracy of roughly 60%, still fails on unseen connector families, and demands continuous offline retraining as new cable combinations enter the line.

Vision also degrades as cameras get closer to harnesses: depth sensing struggles with thin cylindrical shapes, glare and shadow cause intermittent detection failures, and fiducial markers shift or peel on flexible jackets, so they cannot be treated as absolute truth. The customer needed a control system that absorbs the tail rather than enumerating it, one that stays stable when perception is uncertain and the part does not behave as modelled.

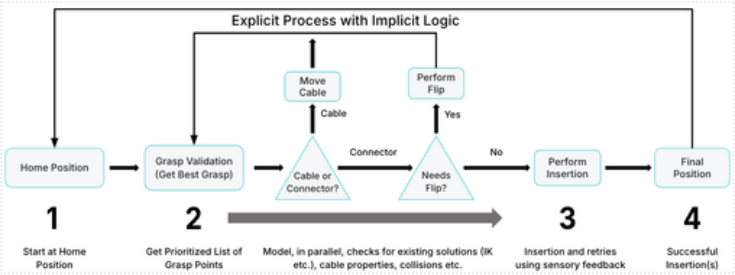
3 REQUIREMENTS

Six presentation scenarios, all to be handled without new code paths, bespoke scripts, or per-scenario trajectories:

- **Simple.** Mating face visible on the tray; direct approach, grasp, grip verification, insertion.
- **Occluded.** Mating face against the tray; automatic reorientation strategy before insertion
- **Random orientation.** Corrected by in-hand micro-rotation, or staged rotation where in-hand risked slip.
- **Upside-down.** Vertical reorientation where inverted insertion is mechanically impossible or risks damage. (*flipping required*)
- **Hanging.** Harness suspended rather than laid flat; pose estimation, grasp affordance generation, controlled lift.
- **Obstacle avoidance.** Obstacle boundaries and free corridors identified, with trajectory refinement on approach.

> Insertion also had to hit line tolerances. Alignment within millimetres, angles within a few degrees, no damage to the connector or jacket. The system had to know when it was done, reliably enough that downstream stations could proceed without manual rework.

4 APPROACH



Model Size: 200kb
Data Used: 5 Simulated Attempts

Sensory Inputs:
- Grasp Location/Orientation
- Insertion Location/Orientation
- Robot Joint States
- Torque

Performance Parameters:
- Connector Insertion Distance
- Expected Insertion Force
- Maximum Insertion Force
- Workspace
- Harness Length

All manipulation is performed by a UR5e with an integrated force/torque sensor and a fixed RGB-D camera covering the tray, the staging area, and the insertion point. AprilTags along the harness give the arm a set of ranked grasp points, with the terminal connector ranked highest and intermediate points ranked by how close they are to it. When the mating face is hidden, the arm does not wait for a better view: it grasps the nearest tagged point and works its way up the cable, moving to progressively higher-ranked grasps until the connector is exposed.

The controller maintains a generative model of the task, predicting the visual and tactile sequence a correct insertion should produce, and selects actions that reduce surprise. If the expected force profile fails to appear, it backs off a few centimeters, rechecks pose, and retries with adjusted approach parameters. Retry behaviour is not a coded exception handler: it emerges from the same drive to minimise prediction error that governs every other action.

Rather than reacting to contact forces after they arrive, the controller predicts how the harness should feel at each stage and corrects proactively, before errors grow large. It also sidesteps the singularity problems affecting inverse kinematics, generating smooth continuous motor commands rather than solving for joint configurations that can become numerically unstable.

5 ONLINE LEARNING

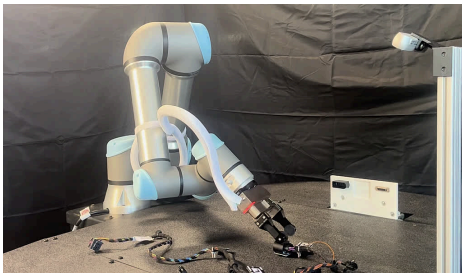
The model was built from roughly fifteen thousand samples (thirty seconds of data captured at 500 Hz), updates online in approximately two milliseconds and consumes under 250 KB. Meeting a stiffer harness jacket, an unfamiliar friction condition, or a connector that seats with a different force signature, it updates its belief state within the cycle rather than flagging a case for offline learning. No labelled dataset is collected, no training run is scheduled, and the line does not stop.

6 TRANSFERABILITY

The architecture separates a general-purpose model from explicit task parameterisation. The model consumes only grasp and insertion pose and robot joint states; everything physical that differs between harnesses is a handful of per-cable parameters — insertion distance, maximum and expected insertion force, minimum flip height, and workspace location and size. A new harness family is a parameter change, not a development project.

The work is decomposed into manipulation, flipping, and insertion modules, each specified only by its objective and sensory modalities; rotation sequences, approaching a tangled cable, and finding insertion points emerge from minimising prediction error. Perception is equally interchangeable: the controller consumes world-space pose and is agnostic to its source.

7 RESULTS



Single Arm Wire Harness Flip

The system succeeded at full task level (grasp, manipulation, and insertion) in every scenario.

Where the controller met an unfamiliar configuration, it slowed, staged the connector into the fixed working space, and used tactile readings together with pose confirmation from vision to verify or correct the final approach. Alignment held inside the millimetre and few-degree tolerances industrial connectors demand, harnesses were handled without damage, and completion was logged so downstream steps could proceed without manual rework. Seating was confirmed by matching the observed force profile against the expected one — including the characteristic snap-fit signature — rather than by assuming success once a position was reached.

WHAT THIS MEANS FOR THE LINE

Edge cases stop being projects. Variability in connector type, presentation, and surroundings was absorbed by one controller operating online, without reprogramming between scenarios.

New variants are a parameter change. A handful of per-connector values replaces waypoint authoring, so a new harness family does not restart the integration cycle.

The line keeps running. Adaptation happens inside the control cycle, so a stiffer jacket or unfamiliar friction is absorbed live instead of becoming an offline training run.

ABOUT THOUGHTFORGE

ThoughtForge is a robot control system for both simple and complex tasks.
Using our patented AI, ThoughtForge-powered robots easily adapts in real time to new parts, dynamic changes, and varying environments, planning motion and paths on-the-fly with unmatched accuracy and robustness.