Robotic Data

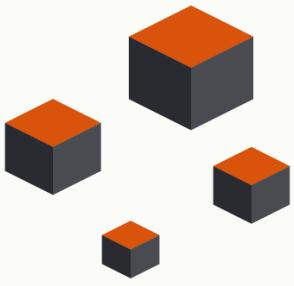
WEBINAR 02 — BUILD THE SYSTEM

Robotic Data Flywheel Framework

From Data Collection to Deployment

Building the continuous cycle that keeps physical AI improving.

How to build a scalable robotic data flywheel — a continuous cycle connecting real-world capture, data processing, model evaluation, gap discovery, and targeted recollection.



Duration: 45 minutes • Dates to be announced

Robotic Data — Data for the Physical AI Revolution

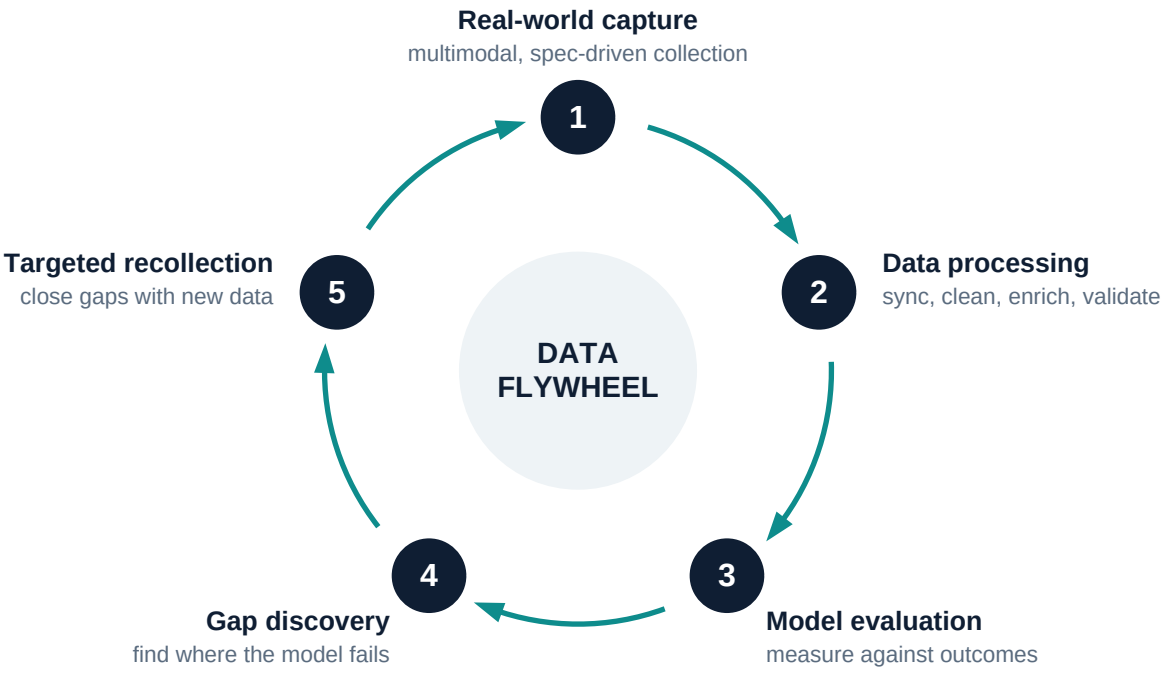
OVERVIEW

Why a flywheel, not a project

A successful data program is not a one-time collection project. It is a continuous cycle connecting real-world capture, data processing, model evaluation, gap discovery, and targeted recollection. Each turn of the wheel makes the next one cheaper, faster, and better targeted — because every deployment teaches you exactly what data to collect next.

Teams that treat data as a project collect a large batch, train a model, and stall when performance plateaus. Teams that treat data as a system instrument their models to reveal failure modes, translate those failures into precise collection specifications, and feed the results back into training on a regular cadence.

The five stages of the flywheel



Key idea

The flywheel's output is not a dataset — it is a repeatable capability: the ability to identify any model weakness and close it with data, on a predictable timeline and budget.

SECTION 01

01 Begin with the task, not the sensor

The most common failure mode in robotic data programs is starting from hardware: choosing sensors, then collecting whatever those sensors produce, then hoping a model can be trained from it. The framework inverts this. Start by writing down the task the robot must perform, in operational language, and derive every downstream decision — modalities, environments, volumes, annotations — from that definition.

Translate the task into a data specification

- **Task statement.** One sentence describing what the robot does, for whom, and under what conditions (e.g., “pick and place mixed-SKU items from cluttered bins in a warehouse aisle”).
- **Success criteria.** Measurable definitions of a completed task: accuracy, cycle time, damage rate, intervention rate.
- **Operating envelope.** The environments, objects, lighting, weather, and human interactions the system must handle — and the ones explicitly out of scope.
- **Data requirements.** Only now: which modalities, at what resolution and frequency, with what annotations, in what volume and diversity.

> **Working rule**
>
> If a data element cannot be traced back to a line in the task specification, you should question why you are collecting it. Unmotivated data inflates cost without improving the model.

SECTION 02

02 Define observations, actions, outcomes, and failure conditions

Robot learning data has a grammar. Every episode you collect should be decomposable into four elements, and your specification should define each one before collection begins.

Element	What to specify
Observations	Everything the robot perceives: camera frames, LiDAR sweeps, depth, proprioception, force/torque, audio. Define fields of view, frame rates, and reference frames.
Actions	What the robot (or demonstrator) does: joint trajectories, end-effector poses, gripper commands, base motion. Define the action space and its sampling rate.
Outcomes	The result of the episode against success criteria: task completed, partially completed, aborted. Label at episode and sub-task level.
Failure conditions	An explicit taxonomy of how episodes fail: perception errors, grasp slips, collisions, timeouts, human interventions. Failures are labeled data, not waste.

> **Why failure conditions matter**
>
> A model can only learn to avoid or recover from failures it has seen. Programs that discard failed episodes systematically starve their models of the most informative data they own.

SECTION 03

03 Design a multimodal capture program

No single sensor tells the whole story. Modalities are complementary: each one covers weaknesses in the others, and cross-modal redundancy is what makes synchronization, validation, and rich annotation possible.

Modality	What it contributes	Limitations to cover
LiDAR	Precise 3D geometry and range, robust to lighting	Sparse at distance; no color/texture; struggles with glass and rain
RGB imagery	Dense semantic detail, texture, color; cheap and ubiquitous	Sensitive to lighting; no direct depth
Video	Temporal context — motion, causality, event sequences	High storage/bandwidth; needs temporal annotation
Depth / stereo	Per-pixel range at close quarters for manipulation	Limited range; noise on reflective surfaces
Wearable sensors	Natural human demonstrations at low cost, in real workplaces	Embodiment gap to robot morphology; calibration drift
Motion capture	Millimeter-accurate ground-truth poses and trajectories	Instrumented spaces only; setup cost
Force / tactile	Contact-rich signal for grasping and insertion tasks	Hard to simulate; sensor fragility
Audio	Contact events, machine state, human speech context	Noisy environments; privacy handling required

Designing the sensor suite

- Choose modalities by tracing each element of the task specification to the sensors that observe it — not by what hardware is on the shelf.
- Pair every primary modality with at least one cross-checking modality so errors are detectable (e.g., LiDAR depth vs. stereo depth).
- Lock calibration and time-synchronization procedures before the first collection day; retrofitting sync onto captured data is rarely fully successful.

SECTION 04

04 Plan for diversity and difficult edge cases

Models generalize only across variation they have seen. Diversity is not a volume problem — ten thousand near-identical episodes teach less than one thousand deliberately varied ones. Plan variation along three axes from the start.

Environmental diversity

- Lighting: daylight, artificial, mixed, low light, glare, shadows in motion.
- Weather and surfaces: rain, dust, fog; wet floors, gravel, carpet, reflective surfaces.
- Clutter and layout: sparse to dense scenes, occlusions, moved furniture, seasonal changes.

Geographic diversity

- Multiple sites, regions, and building types — a model trained in one facility learns that facility's quirks as if they were laws of physics.
- Regional differences in objects, signage, packaging, vehicle types, and architectural norms.

Behavioral diversity

- Multiple demonstrators with different technique, speed, handedness, and body size.
- Varied strategies for the same task — including suboptimal but valid approaches.
- Human bystanders and collaborators behaving naturally, not scripted.

Edge cases: engineer them, don't wait for them

Rare events dominate deployment risk but almost never appear in convenience-sampled data. Enumerate plausible hard cases from the failure taxonomy (Section 02), then stage them deliberately: unusual object poses, partial occlusions, sensor degradation, near-collisions, interrupted tasks. Track edge-case coverage as a first-class dataset metric alongside volume.

> **Coverage over count**
>
> Report your dataset as a coverage matrix (conditions × behaviors × environments), not a single episode count. Gaps in the matrix are your next collection plan.

SECTION 05

05 Process, enrich, validate, and protect collected data

Raw capture is not training data. A disciplined pipeline turns terabytes of sensor streams into datasets a model can actually learn from — and that your organization can legally and safely hold.

Synchronize and calibrate

- Align all streams to a common clock (hardware sync where possible); verify with cross-modal events such as a clap, flash, or contact spike.
- Validate extrinsic and intrinsic calibration per session, not per project — mounts drift, lenses get bumped.

Clean and structure

- Detect dropped frames, sensor dropouts, and corrupted segments automatically; quarantine rather than delete.
- Segment continuous recordings into episodes aligned to the task grammar: observations, actions, outcomes, failure labels.

Enrich

- Add annotations the task requires: object instances, grasp points, sub-task boundaries, success/failure tags, natural-language task descriptions.
- Use model-assisted pre-labeling with human review — pure manual annotation does not scale; pure automatic annotation does not converge.

Validate

- Score every episode against the specification: completeness, sync tolerance, calibration residuals, label agreement.
- Sample-audit with a second annotator; track inter-annotator agreement as a pipeline health metric.

Protect

- Blur faces and license plates; scrub audio of personal identifiers; honor site NDAs and worker consent agreements.
- Apply access controls and provenance tracking so every training example can be traced to its consent basis and collection session.

> **Quality gate**
>
> Nothing enters the training pool without passing validation. A small, trusted dataset beats a large, contaminated one — and contamination discovered after training is expensive to unwind.

SECTION 06

06 Close the continuous improvement loop

The flywheel turns when evaluation drives collection. This requires treating model failures as a managed pipeline of their own.

Discover failure cases

- Instrument deployments: log interventions, aborts, low-confidence actions, and anomalous sensor states.
- Run structured evaluations against the outcome and failure taxonomy from Section 02, so failures land in named buckets instead of anecdotes.
- Cluster failures by cause (perception, planning, control, environment) and by condition (lighting, object class, site).

Prioritize

- Rank failure clusters by frequency × severity × fixability-with-data. Not every failure is a data problem — route control bugs to engineering.
- Estimate the collection cost of each cluster; favor gaps where a small, targeted batch moves a key metric.

Recollect and verify

- Write a narrow collection spec per gap: exact conditions, behaviors, volumes, and acceptance criteria.
- After retraining, re-run the same evaluation suite and confirm the targeted metric moved — and that nothing else regressed.

Build vs. partner

Most mature programs are hybrids. The decision is not one-time; revisit it as the program scales.

Build internally when…	The data is tightly coupled to your proprietary hardware or task; iteration cycles must be same-day; the collection volume is modest and ongoing.
Use a data partner when…	You need geographic or demographic diversity you cannot reach; volume must scale faster than you can hire; you need specialized capture (mocap, wearables, fleets) that is not worth owning; annotation throughput is the bottleneck.

> **Keep ownership of the loop**
>
> Whatever you outsource, keep the task specification, the failure taxonomy, and the evaluation suite in-house. They are the steering wheel of the flywheel.

TAKE IT HOME

✓ Program readiness checklist

Use this checklist to assess your own data program before and after the session. Every unchecked box is a concrete next step.

Specification

- ■ Written task statement with measurable success criteria
- ■ Operating envelope defined, including explicit out-of-scope conditions
- ■ Failure taxonomy documented and used to label collected episodes

Capture

- ■ Every modality traceable to a line in the task specification
- ■ Time-sync and calibration procedures validated before collection
- ■ Diversity plan covering environmental, geographic, and behavioral axes
- ■ Edge cases enumerated and deliberately staged, with coverage tracked

Pipeline

- ■ Automated quality gate scoring every episode against the spec
- ■ Model-assisted annotation with human review and agreement metrics
- ■ Privacy protections (blurring, scrubbing, consent tracking) applied by default

Loop

- ■ Deployments instrumented to log interventions and low-confidence events
- ■ Failure clusters ranked by frequency × severity × fixability with data
- ■ Each recollection batch tied to a metric and verified after retraining
- ■ Build-vs-partner decision reviewed at each scale milestone

Questions for the live Q&A

The session closes with live audience Q&A. Come prepared: the most useful questions name a specific task, a specific failure mode, or a specific scaling decision you are facing. Jot yours here.

> **Webinar 02 — Build the System**
>
> 45 minutes • Dates to be announced. This resource accompanies the session; keep it as a working reference for your data program.