PHYSICAL AI FRONTIERS

ROBOTICS TRACK · EDITION R03 · JULY 2026

One Scene, Many Solvers

OpenUSD as a foundation for Physical AI simulation pipelines — why a shared scene description layer is becoming the connective tissue of robotics simulation, synthetic data and sim-to-real transfer.

Lise DELOBEL

GenAI project manager & consultant — Physical AI, digital twins, and applied ML for industrial and extreme-environment operations. MIT IDSS Data Science & Machine Learning certificate; OpenUSD certification in progress.

LISEDELOBEL.FR · LISE.DELOBEL@ME.COM

Portable description, divergent physics

Robotics is converging on a shared scene-description and composition framework. OpenUSD — developed at Pixar over twenty-five years and open-sourced in 2016 — is becoming an important connective layer through which CAD-derived assets, robot descriptions, sensor models, environments and physics properties are composed across a growing set of tools. In December 2025 the Alliance for OpenUSD published the first normative Core Specification, replacing reliance on the behaviour of a reference implementation with a documented standard and a compliance framework. AOUSD describes ISO standardisation as the next step rather than an achieved status.

The strategic mistake available here is to read that as portability solved. It is not. A shared description makes it possible for compatible tools to exchange and interpret a common stage. It does not guarantee that every simulator implements every schema, ingests every asset, or computes equivalent physics from what it reads. The interesting engineering of 2026 is not the format's adoption — it is the work being done to make the remaining divergence explicit rather than silent.

FINDING 01

The format is winning on composition, not geometry

USD's durable advantage is not mesh exchange but its layering model: non-destructive opinions stacked in strength order, so an environment, a robot and a session override compose into one stage without any of them being rewritten. That is what lets a CAD team, a simulation team and a learning team edit the same world concurrently.

FINDING 02

Description is standardised; solver semantics are not

UsdPhysics standardises rigid bodies, joints, collision and materials. It does not standardise integrators, contact models or tolerances — because those have no common meaning across solvers. The same valid stage can therefore produce different behaviour, or fail outright, on a different engine.

FINDING 03

The divergence is documented, not hypothetical

NVIDIA publishes a list of ways a valid PhysX-authored stage breaks under the Newton backend: reversed joint parent-child order, closed kinematic chains, zero-mass links, zero-size collision placeholders, joints outside articulation roots, composition errors one parser tolerates and the other rejects, and negative scale on collision shapes.

FINDING 04

The fix is to declare the difference, not hide it

Newton's USD schemas extend UsdPhysics with a generalising layer for what transfers and a solver-specific layer for what does not — explicitly as a staging ground, so parameters that prove general can be promoted into the standard later. Honesty about divergence is the mechanism, not a concession.

Two definitions used throughout

A stage is the composed result of a set of USD layers resolved together; the layers themselves remain separate files and none is destructively edited. A schema is a typed set of properties that can be applied to a prim — UsdPhysics is the portable physics schema, while PhysxSchema and MjcSceneAPI are engine-specific extensions carrying parameters that only one solver understands.

A format became infrastructure

OpenUSD has existed since 2016 and has been in robotics tooling for years. Four things changed its status in roughly eighteen months.

Core Spec 1.0
PUBLISHED DEC 2025
First normative specification, with a compliance framework and reference conformance tests

4
SOLVER OPTIONS
Newton exposes MuJoCo, XPBD, Featherstone and semi-implicit options behind a common API

2023
AOUSD FOUNDED
Governance body now including Booz Allen Hamilton and Schneider Electric

01 **Specification replaced implementation**

Until December 2025 OpenUSD was defined largely by the behaviour of its reference implementation. The Core Specification formalised the data model, composition algorithm, value resolution, file formats and a compliance framework in one normative document. AOUSD frames it as the first step toward ISO standardisation. That changes the procurement conversation: an operator can begin to require conformance rather than a particular vendor's build.

02 **Physics acquired a shared vocabulary**

The UsdPhysics schema gives rigid bodies, collisions, joints and materials a common representation, and engine-specific extensions sit alongside it — PhysxSchema for NVIDIA PhysX, and an MJC schema developed jointly by NVIDIA and Google DeepMind for MuJoCo. Robotics formats like URDF and MJCF describe the robot; USD describes the robot inside the world.

03 **The physics engine itself became pluggable**

Newton — an Apache-2.0 Linux Foundation project initiated by Disney Research, Google DeepMind and NVIDIA, which reached 1.0 in March 2026 — is built on OpenUSD and exposes several simulation implementations and solver configurations behind a common API. Isaac Sim can switch between registered backends within one session, provided the simulation is stopped first, which turns solver choice into a configuration decision rather than a rebuild.

04 **The pipeline reaches back into design**

CAD is now a first-class entry point rather than a lossy export: cloud-native workflows connect Onshape directly to Isaac Sim using OpenUSD as the bridge, and AOUSD participants are exploring richer representations for engineering data that would preserve more design intent than a one-time mesh export.

Sources: AOUSD, Core Specification 1.0 and member announcements, 2026 · NVIDIA Isaac Sim and Isaac Lab documentation, 2026 · Newton project, Linux Foundation.

What a shared stage does and does not buy

Four properties of the format, in descending order of how safely they can be assumed.

LAYER 01

Composition — reliable

Layers stack as non-destructive opinions resolved in strength order. An environment, a referenced robot and a session override coexist without any being rewritten, which is what makes concurrent work across teams possible. This is the part of USD that genuinely transfers everywhere.

LAYER 02

Geometry and scene structure — often portable

Meshes, transforms and basic hierarchy commonly transfer well where tools support the same OpenUSD features. Cameras, materials, semantic labels and domain-specific properties require explicit schema and implementation compatibility — and instancing, negative scale, units and up-axis all remain live sources of divergence.

LAYER 03

Physics description — partially reliable

UsdPhysics gives a common vocabulary for masses, inertias, joints, collision shapes and materials. Tools that implement the schema can read what was authored, but the supported subset and the runtime interpretation differ — frequently enough to matter.

LAYER 04

Solver behaviour — not portable

Integrator choice, constraint solver algorithm, friction cone shape, tolerance and contact buffer sizes are properties of a numerical method, not of the world. A schema can standardise how such parameters are named, typed and discovered, and can expose solver-specific ones explicitly. It cannot make distinct integrators, contact models and constraint algorithms numerically equivalent.

THE LADDER OF INTEROPERABILITY

01	02	03	04
File exchange	**Shared schema**	**Shared composition**	**Declared divergence**
Export and re-import. Every transfer is lossy and every edit is destructive.	Agree what the fields mean. Geometry and mass now survive the trip intact.	Agree how files combine. Teams edit the same world without overwriting each other.	Agree what cannot be shared, and author it explicitly rather than leaving it implicit.

The strategic read

Most organisations evaluating USD are buying rung two and assuming they have bought rung four. The gap matters because the failure mode of assumed portability is not a broken file — it is a simulation that runs, looks correct, and produces subtly wrong physics. That is the same class of error as the reality gap examined in Edition R01, relocated from the actuator into the toolchain.

From CAD to a trainable environment

The design-to-simulation path, and why it used to be a rebuild

The historically expensive step in industrial simulation is not the simulation. It is reconstructing the world: taking CAD that exists, geometry that is exact, and rebuilding it in a form the physics engine will accept — usually by hand, usually lossily, and usually again from scratch the next time the design changes.

THE INTAKE

What the pipeline now ingests

Isaac Sim can receive CAD-derived assets through supported connectors and conversion pipelines, alongside URDF and MJCF robot descriptions and captured environment data, converting each into USD. PTC has demonstrated a cloud-native workflow connecting Onshape CAD directly to Isaac Sim with OpenUSD as the bridge, with the stated aim of moving from design to a realistic environment without losing data.

THE MECHANISM

Why composition changes the economics

Because layers compose rather than overwrite, a design revision can be re-referenced into an existing simulation scene without discarding the physics tuning, sensor placement or task setup layered on top of it. With a disciplined asset structure and stable prim paths, much of the expensive work — collider authoring, joint drive tuning, material assignment — can survive a geometry update. Changes to topology, hierarchy or contact geometry may still invalidate it. That difference, between a pipeline and a rebuild, is a property of the discipline applied, not of the format.

THE HAZARD

The unglamorous failure: units

The most common way this breaks is stage metrics. USD records metersPerUnit per layer, and composition does not by itself guarantee physical rescaling across every composition arc. A robot authored in centimetres brought into a metre stage is a hundredfold linear error — a millionfold error in volume, and in inferred mass wherever mass is derived from density and geometry rather than authored explicitly. The arc matters: referenced or payloaded assets can be corrected by a tool such as NVIDIA's Metrics Assembler, which authors a corrective transform on the enclosing prim, while a mismatched sublayer has no such prim and cannot be fixed the same way. NVIDIA's guidance is a metre-based root layer, robots authored in metres, and no sublayering of content in other units. Units belong to a wider class of pipeline faults that produce a running simulation rather than an error — see the Engineering Zoom on silent contact truncation.

So what

For an operator holding a digital twin, this is the question that determines whether the twin is an asset or a screenshot: can a design change flow into it without discarding the work layered on top? If every geometry revision triggers a rebuild, the twin will fall out of date and quietly stop describing the plant — which is the failure mode most twin programmes actually die of, rather than any deficiency in the simulation itself.

Sources: NVIDIA Isaac Sim documentation, 2026 · NVIDIA developer guidance on OpenUSD for robotics · AOUSD member announcements, March 2026.

The scene as a data factory

Synthetic data generation, and why the environment is the reusable asset

Edition R01 argued that simulation has moved upstream of the policy — it manufactures training data rather than validating results. That only works at scale if the environment being randomised is itself a durable, structured asset rather than a one-off scene.

THE REQUIREMENT

Randomisation needs structure to act on

Domain randomisation varies lighting, materials, poses, textures and camera parameters across thousands of renders. Doing that programmatically requires the scene to expose those properties as addressable, typed data — which is precisely what a schema-based scene description provides and what an opaque exported mesh does not.

THE PAYOFF

Labels come from the scene, not from annotators

Because object identities and semantic annotations can be associated with scene prims and their relationships, segmentation masks, bounding boxes and depth can be derived from the scene rather than drawn by hand. The economics of perception training change accordingly — though the true marginal cost includes simulation, orchestration, storage and validating that the labels are actually correct, not rendering alone.

THE DURABILITY

The asset outlives the model

Models are replaced constantly; a physically validated environment is not. An operator that has built an accurate, composable description of its own facility can retrain against it as architectures change, which is a materially different position from having trained one model well.

So what

The recurring theme across this series is that the durable asset is rarely the model. In Edition R01 it was measured actuator behaviour; in R02 it was the catalogue of what the process cannot handle; here it is a structured, physically validated description of your own environment. All three share a property: they are expensive to acquire, cheap to reuse, and almost never on anyone's balance sheet.

Sources: NVIDIA Isaac Sim synthetic data generation documentation, 2026. Cross-references Edition R01 on simulation as a training-data source.

Swapping the physics engine underneath

Newton, solver pluggability, and what portability actually delivers

The most concrete test of a shared scene description is whether you can change the physics engine without rebuilding the scene. In 2026 that became possible in a mainstream robotics stack — and the honest account of how well it works is the most useful thing in this paper.

THE SYSTEM

What shipped

Newton is a GPU-accelerated, differentiable physics engine built on NVIDIA Warp and OpenUSD. It is an Apache-2.0 open-source Linux Foundation project initiated by Disney Research, Google DeepMind and NVIDIA, and reached 1.0 in March 2026. It exposes several simulation implementations and solver configurations behind a common API — currently MuJoCo, XPBD, Featherstone and semi-implicit options, with MuJoCo Warp as the primary backend — plus deformable simulation and signed-distance-field collision.

THE GAIN

What portability delivered

Isaac Sim can switch between registered backends within a session, with the simulation stopped. Isaac Lab's solver-transitioning documentation states that for its current environments the same USD files serve both the PhysX and the Newton MuJoCo-Warp paths. That is scoped to those environments rather than to every asset, but the ability to stop, change engine and replay the same scene is genuinely new.

THE CAVEAT

What it did not deliver

The integration is documented as experimental, tested with a limited robot set including G1, H1, T1, UR5e, Allegro and Shadow Hand, and NVIDIA states that assets tuned for PhysX may not produce optimal results under Newton without adjusting contact settings, solver iterations and timestep. Vendor benchmarks reporting large speedups for MuJoCo Warp are developer-reported and measured on specific tasks and hardware.

So what

Read solver portability as a procurement option rather than a performance claim. Its value is that it removes a category of lock-in: the scene, the tuning and the task definition stop being hostage to one engine's licence and roadmap. Note what does not travel: solver-specific tuning stays coupled and must be identified, versioned and recalibrated on migration — Isaac Sim's newer asset structure keeps separate PhysX and MuJoCo definitions precisely because the tuning values differ. The right question for a vendor is not whether they support USD, but how much of your work would survive if you switched.

Sources: NVIDIA Isaac Sim, Newton Physics Backend documentation, June 2026 · Isaac Lab solver transitioning documentation · Newton project, Linux Foundation. Performance figures are developer-reported and task-specific.

The scene that loads twice and means two things

One valid USD stage, two physics backends — documented divergence

THE CHALLENGE

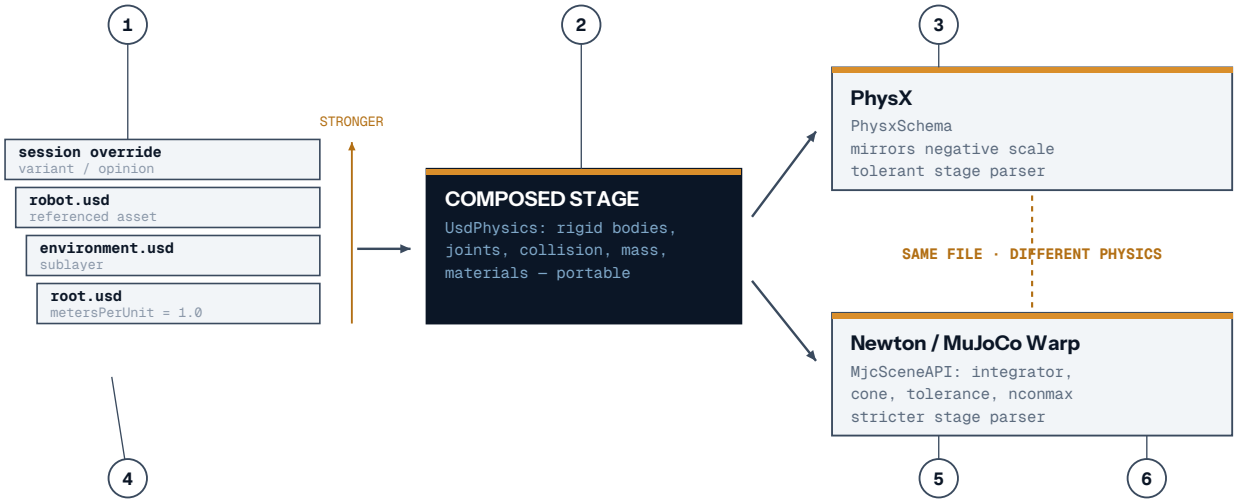
A USD stage that describes a robot is not a program; it is a set of assertions about bodies, joints, masses and collision shapes. Two solvers reading the same assertions should, naively, produce the same motion. They do not — and the ways they diverge are specific, documented and instructive.

NVIDIA publishes a compatibility list for the Newton backend that reads as a catalogue of buried assumptions: joints whose parent and child are ordered the way PhysX expects and Newton rejects; closed kinematic chains that one engine simulates and the other refuses; links with unset mass that PhysX tolerates and Newton's solver will not; zero-size collision placeholders; joints sitting outside any articulation root; and composition errors that one parser silently ignores while the other declines to start.

Why the obvious solutions fail

Tightening the standard cannot resolve this, because the parameters that differ — integrator type, constraint solver algorithm, friction cone shape, tolerance — are properties of a numerical method, not of the world being described. A schema that forced them into a common vocabulary would either be meaningless or would delete the differences that make a given solver worth choosing. And validation does not save you either: a stage can be entirely valid USD and still be wrong for the engine about to read it.

FIG. 01 — LAYER COMPOSITION AND SCHEMA DIVERGENCE (SCHEMATIC)



1 Layer stack — non-destructive opinions
2 Composition resolves to one stage
3 Solver-specific schema extension
4 Stage metrics: units and up-axis
5 Generalising layer (NewtonSceneAPI)
6 Solver parameters with no common meaning

The failure that does not fail

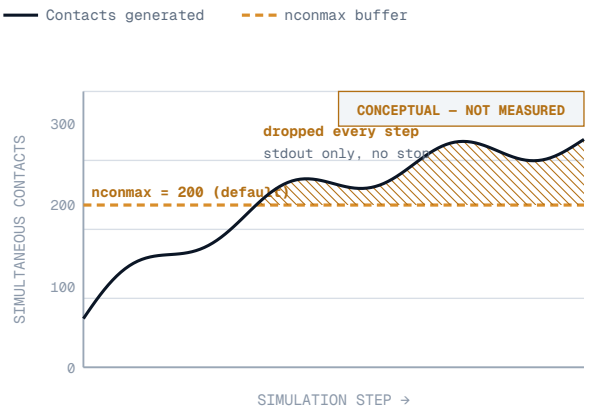
The sharpest case, the design move, and the trade-off accepted to ship

THE SHARPEST CASE

Most of the documented incompatibilities are loud: the stage refuses to load and prints a reason. One is not. The MuJoCo Warp solver pre-allocates a fixed contact buffer, defaulting to 200 simultaneous contacts. A scene that generates more than that does not stop, and does not raise an error — it drops the excess contacts on every step and continues, producing incorrect results while a message goes to standard output.

For contact-rich work this is exactly the wrong failure mode. A cluttered bin or a multi-fingered hand is where contact counts spike, and where a policy trained against truncated contacts will look convincing in simulation and behave differently on hardware.

FIG. 02 — CONTACT BUFFER TRUNCATION, CONCEPTUAL



SCENE CONFIGURATION — SCHEMA STACK

Portable schema	**UsdPhysics**
Generalising layer	**NewtonSceneAPI**
Solver-specific	**MjcSceneAPI**
Integrator options	**euler / rk4 / implicit**
Friction cone	**pyramidal / elliptic**
Contact buffer	**nconmax, default 200**

Schema layering as published for the Newton backend. Figure 02 is a conceptual illustration of the documented truncation mechanism, not measured data — the buffer limit and the silent-drop behaviour are documented; the contact profile is illustrative.

The design move, and what it costs

The response was not to paper over the difference but to make it authorable. Newton's USD schemas extend UsdPhysics with two tiers: a generalising layer carrying what has clear physical meaning across simulators — timestep, gravity, solver iterations — and a solver-specific layer carrying what does not, such as integrator type, friction cone and tolerance. The extension is explicitly a staging ground: parameters that prove to generalise become candidates for promotion into the standard itself. The cost is that the file stops being solver-agnostic — you trade the illusion of portability for reproducibility.

THE OUTCOME — DOCUMENTED, EXPERIMENTAL, HONEST

8 + 1
DOCUMENTED ISSUES
Eight asset-loading incompatibility classes, plus a separate contact-buffer truncation hazard.

2 tiers
SCHEMA LAYERING
Generalising parameters separated from solver-specific ones, both authorable in USD.

6
ASSETS TESTED
Stated coverage of the experimental integration: G1, H1, T1, UR5e, Allegro and Shadow Hand — robots and hands.

Source: NVIDIA Isaac Sim, "Newton Physics Backend," documentation last updated 5 June 2026, Asset Compatibility and Scene Configuration sections · Newton USD Schemas project.

Adopting a scene layer without believing the marketing

Durations are indicative and depend heavily on how much CAD and tooling already exists.

PHASE 01 · 0-3 MONTHS

Fix your metrics and your asset structure

Before anything ambitious, settle units, up-axis and naming, and decide what belongs in which layer. This is dull, and it is the step whose absence causes the most expensive failures later. A metre-based root layer and a documented layering convention are worth more than any amount of tooling.

PHASE 02 · 3-9 MONTHS

Make one asset pipeline non-destructive

Pick a single line, cell or vehicle and prove that a CAD revision can flow into the simulation scene without discarding physics tuning, sensor placement or task setup. If that round trip does not work, nothing built on top of it will stay current.

PHASE 03 · 9-18 MONTHS

Validate physics against measurement, not against another simulator

Establish that the simulated dynamics match instrumented reality within a stated tolerance. Comparing two simulators tells you they disagree; only hardware tells you which is wrong. This is the step that converts a visually faithful twin into a training environment.

PHASE 04 · 18 MONTHS +

Treat solver choice as a decision you may revisit

Keep the scene, the tuning and the task definition portable enough that changing engine is a configuration change rather than a project. Record which parameters are solver-specific; they are the ones that will not travel, and knowing which they are is most of the work.

Organisational readiness — and a target operating model

This capability falls between three owners who rarely share a budget line: engineering owns the CAD, IT owns the file infrastructure, and the robotics or data team owns the simulator. USD adoption usually stalls not on technology but on the absence of anyone whose job is the asset pipeline itself. The pattern that works is a small central asset-pipeline function — two to four people is typical — reporting into engineering rather than IT, because the decisions are engineering decisions: units, layering conventions, collider standards and what constitutes a validated asset. It should own three things and only three: a published conformance standard that assets must meet before entering the simulation library, an automated validation gate that enforces it, and the register recording which parameters in each scene are solver-specific. Funding it from the simulation programme rather than from IT overhead is what keeps it accountable to the people whose work it unblocks.

What goes wrong, and what it costs

Ranked by how often it is missed rather than by severity.

RISK 01

Assumed portability

A stage that loads in two engines is not a stage that behaves the same in two engines. Treat cross-engine agreement as something to be demonstrated on your own assets, never as a property of the format.

RISK 02

Non-fatal numerical truncation

Fixed buffers and solver limits can discard information without raising an exception or stopping the run — a diagnostic on stdout is easy to miss in an unattended pipeline. Any simulation feeding a training loop needs assertions on its own internals, not only on its outputs.

RISK 03

Stage metrics

Units and up-axis mismatches are the most common and most embarrassing failure, and they scale non-linearly: a hundredfold length error is a millionfold volume error, and a millionfold mass error wherever mass is inferred from density rather than authored explicitly.

RISK 04

Validity is not correctness

USD validation checks that a stage is well-formed. It cannot tell you that a mass is missing, a collider is a placeholder, or a joint is parented the way the wrong engine expects.

RISK 05

Standard versus implementation

A ratified Core Specification is not the same as conformance across tools. Ask vendors which specification version they implement and which schemas they actually honour, not whether they support USD.

RISK 06

Schema sprawl

Engine-specific extensions solve a real problem and create a versioning one. Record which parameters in your scenes are solver-specific; they are precisely the ones that will not survive a migration.

RISK 07

Experimental status read as production

Several of the most attractive capabilities here are documented as experimental, with limited tested robot coverage and features still unsupported. Plan against the documentation rather than the keynote.

RISK 08

The asset pipeline has no owner

The commonest organisational failure is that composition discipline, naming and layering belong to nobody. Without an owner, the layer structure degrades until every change becomes a rebuild again.

08 / WHERE TO START

Three questions worth asking this quarter

01 Can a CAD revision reach our simulation without discarding the tuning on top of it?

02 Which of our simulation parameters are properties of the world, and which of the solver?

03 Who owns our asset pipeline — by name?

The third question is the one that predicts the others. Format adoption is rarely limited by the format.

NEXT IN THIS TRACK

Edition R04 — The Last Hundred Microns

Precision assembly, where available clearance is smaller than the combined uncertainty of robot, tooling, fixturing and parts — and the two-point contact that makes the obvious insertion strategy fail.

SOURCES & METHOD

Primary: NVIDIA Isaac Sim documentation, "Newton Physics Backend," last updated 5 June 2026 — the Asset Compatibility and Scene Configuration sections are the source for the documented incompatibilities, the schema layering, and the contact-buffer behaviour · Isaac Lab documentation on solver transitioning · Newton USD Schemas project.

Secondary: Alliance for OpenUSD announcements on Core Specification 1.0 and membership, 2026 · Newton project under the Linux Foundation · Isaac Lab technical report, arXiv:2511.04831 · NVIDIA developer guidance on OpenUSD for robotics.

Method note: performance figures for individual solvers are developer-reported, task-specific and not independently audited, and are cited as such. Figure 02 illustrates a documented mechanism, not measured production data. Statements about experimental status and tested robot coverage are taken directly from vendor documentation and were current at the date of publication; this is a fast-moving area and readers should check the documentation before relying on any specific limitation persisting.

Cross-references Edition R01 on the sim-to-real gap and simulation as a training-data source, and Edition R02 on contact-rich manipulation.

Lise DELOBEL

GenAI project manager & consultant — Physical AI, digital twins, and applied ML for industrial and extreme-environment operations. MIT IDSS Data Science & Machine Learning certificate; OpenUSD certification in progress.

LISEDELOBEL.FR · LISE.DELOBEL@ME.COM

PHYSICAL AI FRONTIERS · ROBOTICS TRACK · EDITION R03 · JULY 2026