

From Video to Digital Twin: An Exploratory Study of World Models and Gaussian Splatting on In-the-Wild Video

Somesh Mohapatra^{1*}

¹Independent Researcher.

*Address correspondence to: pikulsomesh@gmail.com

Abstract

Generative world models and 3D Gaussian Splatting promise to turn ordinary video into explorable 3D scenes, but little is reported on how they behave on uncured, in-the-wild footage. We study what these models can and cannot recover when the only input is a publicly available news walkthrough of a food bank warehouse, with no drawings, scans or site visit, and we use the result as the geometry of a digital twin. A commercial world model given 20 frames recovered camera poses and metric depth for three areas, which we fused into about 747,000 Gaussian splats for 200 credits (20 US cents). Splat models optimized locally from 49 to 72 frames per area, at no credit cost, were photorealistic near the filmed path but degraded quickly away from it. Asking the world model to generate views of areas the video never showed failed on the provider’s side twice, so unseen space was filled by a scripted Blender model instead. We describe the coordinate and packaging corrections needed to turn world-model output into a valid, metric Universal Scene Description (OpenUSD) asset using its native splat schema, and use that format as the only interface to a serverless data layer, a browser viewer, Blender and NVIDIA Omniverse. Omniverse rendered the modeled facility with RTX but drew the splat prims as blank grey. Because a sensor-fed twin brings operational technology (OT) into scope, we also give a zone-based OT security design in which the twin can read the floor but never write to it; only its read-only cloud controls are implemented. The findings suggest that today’s models recover the observed parts of a real scene cheaply and at metric scale, while extrapolating unobserved space, the step a whole-building twin needs most, remains the open problem. The work is an exploration of these methods, not a deployed product.

1 Introduction

A digital twin is a virtual representation of a physical asset that stays synchronized with it through data [1, 2]. In industry, twins are used to plan layouts, monitor equipment and rehearse changes before they are made on the floor [3, 4]. The geometry of a facility twin is usually the expensive part. It comes from terrestrial laser scanning, photogrammetry campaigns or hand-built computer-aided design (CAD) models, and specialists then import it into a simulation tool. That cost is tolerable

for an automotive plant. It is prohibitive for a regional food bank, which runs warehouse-scale cold-chain logistics on a non-profit budget.

Two developments change that cost structure. Neural scene representations, and 3D Gaussian Splatting in particular [5], produce photorealistic scenes that render in real time from ordinary images. Classical reconstruction recovers camera poses with structure-from-motion [6]; neural radiance fields [7] improved view synthesis but render slowly; learned multi-view geometry such as DUST3R [8] reduces the number of views needed. All of these reconstruct only what was observed. Generative world models [9, 10] go further: from a few views they produce a coherent, explorable 3D world and fill in regions the input never covered. World Labs’ Marble [11] exposes this capability through an API that returns Gaussian-splat worlds. For a twin, generation has a double edge. It makes a sparse walkthrough usable, but it also invents geometry.

In parallel, Universal Scene Description (OpenUSD), originally from Pixar and now developed under the Alliance for OpenUSD, has become an open, vendor-neutral scene description with layered composition and typed schemas [12, 13]. Its 26.03 release added the `ParticleField3DGaussianSplat` schema family [14, 15], so a splat reconstruction is no longer a viewer-specific file. It can sit in the same stage as sensors, equipment and annotations. NVIDIA Omniverse [16] is a set of libraries that bring OpenUSD data interoperability, RTX rendering and physics simulation into applications. Its licensing is mixed: some components are open source, while the Kit SDK is distributed under NVIDIA’s own license terms: the Kit App Template states that its software and materials are governed by the NVIDIA Software License Agreement and the Product-Specific Terms for NVIDIA Omniverse [17], not an open-source license.

Surveys of digital twins identify geometry acquisition, data integration and interoperability as the recurring barriers [3, 4]. This work addresses the first and third barriers directly and keeps the second deliberately thin. This paper is an exploratory study of the learning methods behind the first barrier. We ask three questions: what a world model and locally optimized Gaussian splats recover from uncurated, publicly available video with no CAD drawings, laser scans or site visit; whether the generative step can fill in space the camera never saw; and what it takes to carry the result into an open scene format that other tools can use. To answer them we compose existing models into a three-step pipeline (Figure 1) and apply it to one realistic input, a news walkthrough of a food bank. The steps are:

1. **Video to digital twin with world models.** Blur-aware frame extraction feeds a world model that returns a Gaussian-splat world, which is converted to an OpenUSD stage in metric, Y-up coordinates. Where the camera saw too little, a facility model scripted in Blender fills in the building, published in the same format.
2. **Connecting existing data systems on Amazon Web Services (AWS).** A serverless API exposes operational data in the twin’s coordinate frame through a small, versioned JSON contract. The reference implementation simulates cold-chain telemetry; the contract is designed so that a real sensor feed or warehouse management system can replace the simulator without changing any renderer.
3. **NVIDIA Omniverse as the visualization layer.** Omniverse renders the USD twin with

73 RTX on a rented GPU; binding live data to scene objects is the planned next step. A browser
74 viewer and Blender read the same USD and the same JSON, so the renderer is a deployment
75 choice rather than an architectural one.

76 We do not propose a new model; the contribution is an empirical account of how current models
77 behave on in-the-wild input. It consists of a measured comparison of world-model reconstruction
78 and locally optimized splats on the same footage, including where each fails; a report of the gen-
79 erative extrapolation step failing on this input; an open-source pipeline from walkthrough video
80 to an OpenUSD Gaussian-splat twin, including the coordinate and packaging corrections needed
81 to make world-model output a valid metric USD asset; an architecture in which OpenUSD and a
82 small telemetry contract are the only interfaces between reconstruction, data and rendering, with a
83 serverless backend that has no standing cost; an operations layer that models inventory, flows, docks
84 and a community kitchen behind the same API and as OpenUSD prims; cost controls that make
85 paid world-model generation safe to run unattended; an operational technology (OT) security design
86 that keeps the twin read-only toward the floor; and a food bank as the test case. Food banks are
87 logistics operations: when the Central Texas Food Bank moved into its 135,000-square-foot Austin
88 facility in 2016, the new building held seven times the refrigeration and freezer capacity of the old
89 one [18], and its newer Waco hub combines a warehouse and distribution hub, a kitchen, a food
90 market and a training center in 64,000 square feet [19]. Cold storage at that scale is exactly what a
91 twin with live temperatures is meant to watch.

92 The case study is a representative Texas food bank, not a model of any one facility. It is inspired
93 by the Central Texas Food Bank: three of its areas are captured from a public walkthrough of one of
94 that food bank’s buildings, and its operating volumes are anchored to published figures for another.
95 The rest of the building is designed for this paper, and all live data is simulated. Table 2 states
96 where every part of the twin comes from, and the figures show captured, designed and simulated
97 content so that they can be told apart. The aim is to show what a food bank’s twin looks like and
98 what it takes to build one, not to describe a particular site.

99 2 Results

100 2.1 Three steps joined by one open contract

101 The steps share data, not code. Two artifacts cross step boundaries (Table 1). The reconstruction
102 step publishes a USD package, the SPZ splat file it was converted from (for lightweight viewers)
103 and a metadata file recording scale, ground offset and provenance. The data step publishes a JSON
104 document whose positions are in meters in the same frame as the USD stage. A renderer needs
105 nothing else, which is what lets Omniverse, a browser and Blender coexist (Figure 1).

106 2.2 A walkthrough news video yields usable input for three zones

107 The input was a public news walkthrough of a new Central Texas Food Bank facility, which by its
108 broadcaster and on-screen campaign appears to be the Waco hub [19] (18 min 32 s, 1920 × 1080 at

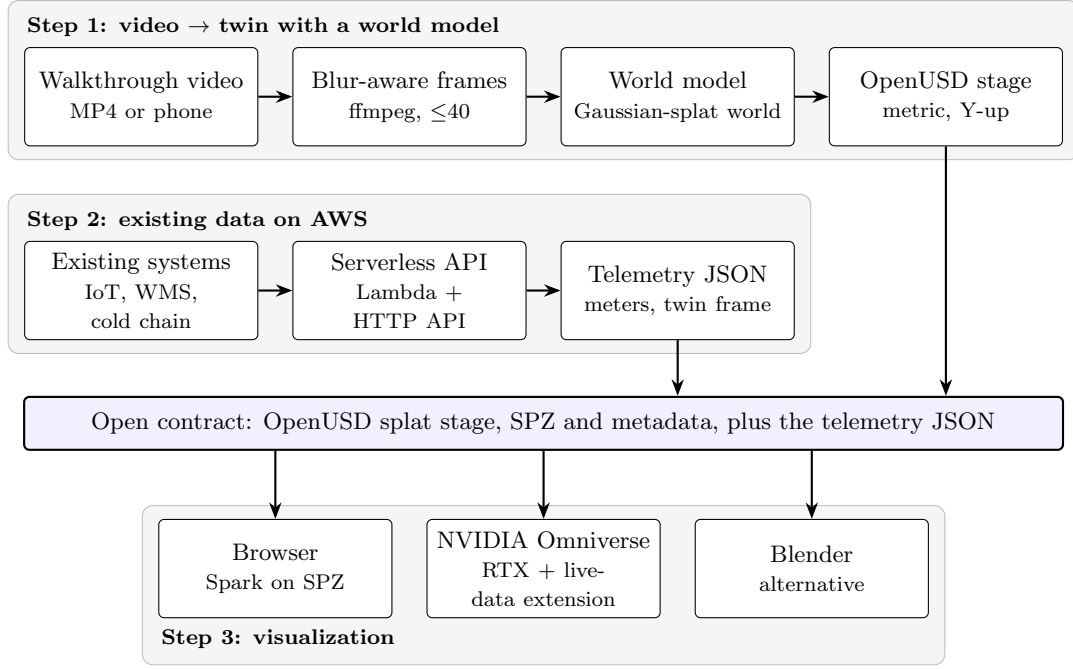


Figure 1: The Auto-Twin pipeline. The open contract is the only interface between steps: the OpenUSD stage (with the SPZ it was converted from and a metadata file) and the telemetry JSON. Any box can be replaced without changing the others.

59.94 frames per second) [20]. Two people appear in almost every shot and a lower-third graphic is burned into each frame, which makes it a realistic rather than a curated input. A multi-image world needs overlapping views of one space, so we targeted a single zone. Surveying the video at one frame every 10s gave the candidate zones in Table 3. The pallet-racking aisle was captured first, because the camera circles one bay and then looks down the aisle, and because it is the zone that matters most for inventory locations, forklift paths and telemetry. The volunteer sorting room and the client market were captured in a second run, from narrower windows in which the camera moves enough to give overlapping views. The other operating areas of the representative food bank (receiving, freezer, cooler and order staging) were not captured: they are designed footprints in the operations layer, drawn by a modeled facility, not generated or reconstructed geometry (Table 2). The community kitchen was captured later by local training only.

Automatic frame selection failed on this footage. Ranking frames by variance of the Laplacian, a standard sharpness score, rewards in-focus faces, and on this video it selected six near-identical shots of the presenters. The eight inputs were therefore chosen by hand for viewpoint spread and absence of people, spanning 6:13 to 7:33 (Figure 2): three views of one bay from different angles, one view with no people, two from the opposite side of the aisle, a wide shot under the beam and one looking down the main aisle toward the dock. The sorting room and market inputs, six frames each, were picked by hand in the same way from 40 extracted frames per zone. In every input, the two presenters were covered by hand-drawn boxes that exclude those pixels from reconstruction.

Table 1: Contracts between the pipeline’s steps. Changes to the telemetry contract are additive only.

Boundary	Artifact	Rules
1 to 3	<code>factory.usdz</code> , <code>factory.spz</code> , <code>factory.meta.json</code>	Metric (<code>metersPerUnit = 1</code>), Y-up, ground at $y = 0$; alignment and provenance in the metadata file.
2 to 3	<code>GET /telemetry</code> (JSON)	<code>zones[]</code> (temperature, setpoint, humidity, door, status, position) and <code>forklifts[]</code> (position, heading, speed, battery); positions in meters in the twin frame; <code>simulated</code> flag.

2.3 The world model recovers metric geometry, but did not yet extrapolate

The world model posed every input frame and returned metric depth with a per-pixel confidence. The depth is plausible at real scale: in the racking aisle, the first beam lands about 0.4 m above the camera, consistent with a beam about 1.9 m above the floor for a handheld camera. Fusing the posed frames gave 270,423 splats for the racking aisle, about 282,000 for the sorting room (keeping every third pixel) and about 195,000 for the client market, about 747,000 in total (Table 6). The floor was found from a dense horizontal slab of points in the sorting room and from the world model’s own ground plane in the market; in the racking aisle, where the floor is barely visible, it was placed 1.5 m below the source cameras.

The generative step did not succeed. Both requests to generate a ring of eight new views around the racking aisle returned `EXECUTION_FAILED` without further detail: the first placed every target camera at one center, the second spread them on a 0.6 m ring and centered the context principal points. Retries were paused until it is known whether failed requests are billed. The twin therefore covers only what the camera saw (Figure 3): each zone is a partial shell seen from its source viewpoints, with gaps where the presenters were masked and where the camera never pointed. These gaps are exactly where generated views are meant to go.

2.4 Locally trained splats make four zones photorealistic at no credit cost

To separate what the world model contributes from what Gaussian splatting contributes, the racking aisle was also reconstructed the conventional way, without any paid call. A 100 s segment of the tour (from 6:08) was sampled at 4 frames per second and 960 pixels wide; people were masked automatically, together with the broadcast graphic; and sharp frames in which the camera had moved were kept, leaving 72 frames, nine times as many as the world-model capture used. Camera poses came from structure-from-motion on the CPU [6, 21], and a 3D Gaussian Splatting model [5] was trained on the CPU with a trainer written for this work. Near the filmed path, the racking, wrapped pallets, shelf labels and the bottle bin read photographically, where the world-model capture shows sparse fragments (Figure 4).

The same route was then run on the sorting room, the client market and the community kitchen, which the world model never captured (Table 4). The four zones hold 105,899 to 161,425 splats in

Table 2: Provenance of each part of the representative food bank twin.

Part of the twin	Kind	Source
Racking aisle, sorting room, client market (geometry)	Captured, partial	20 frames of a public walkthrough of a Central Texas Food Bank building; world-model depth fused locally. The same three are also trained locally from 49 to 72 frames each (Table 4).
Community kitchen interior (geometry)	Captured, partial	Trained locally from 65 frames of the same video; no world-model capture.
Absolute scale	Estimated	The world model’s metric depth; not checked against a survey.
Building shell, receiving, freezer, cooler, order staging, kitchen annex footprint	Designed	Footprints chosen for this paper.
Modeled facility (racking, cold rooms, sorting lines, market, kitchen equipment, forklifts)	Designed	Built in Blender by one script from the designed footprints, with fit-out inspired by the video; a compact 26×20 m building plus an 11.5×11 m annex, much smaller than the facility whose volumes drive the operations model.
Annual volume, agencies, floor area, pallet positions, cold storage, docks, kitchen equipment, meals per day	Published	Figures for the Central Texas Food Bank’s Austin facility (Table 5).
Source and channel shares, product mix, capacities, recipes, step times	Estimated	Set for this paper and flagged as estimates in the master data.
Temperatures, door events, forklifts, daily flows and inventory	Simulated	Computed from the clock; every payload carries simulated: true .

1.6 to 2.5 MB SPZ files and 6.9 to 10.6 MB USDZ packages, with the same contract as the world-model zones, and cost 0 credits. Two changes to the trainer made the difference between a usable zone and a hazy one. The presenters walk with the camera, so the space behind them is masked in nearly every frame; the trainer now fills the masked pixels with background warped in from frames where they had moved, and every 250 steps it removes splats that sit inside their visual hull. The sorting room, white walls over a grey floor, needed exhaustive feature matching before structure from motion would converge. In the browser, all four zones sit in the representative building, the kitchen in its annex (Figure 5). Three trade-offs remain. Structure from motion has no metric scale, so each scene was scaled by assuming the camera was held 1.6 m above the floor, where the world model measured depth directly. The spot right behind the presenters was never filmed and trains to a soft blur rather than to holes or ghosts. And quality falls off quickly away from the filmed path, because the walkthrough moves mostly forward.

2.5 World-model output becomes a valid, metric OpenUSD asset

Converting the splat world to OpenUSD required four corrections, described in Materials and Methods: a coordinate transform from the world model’s camera frame, an untyped reference prim so the splat schema is not hidden, binary packaging with correct alignment, and a choice of splat

Table 3: Candidate zones in the source walkthrough and the assessment that led to the three captured zones.

Time (min:s)	Zone	Assessment
0:45–1:35	Volunteer sorting room	Captured from 0:48–1:06, where the camera moves; 6 frames.
2:10–2:50	Freezer and cooler doors	Mostly blank walls; fogged at 2:40.
5:05–6:00	Loading dock	People fill the frame.
6:10–7:45	Pallet-racking aisle	Captured. Racking, wrapped pallets and a pallet jack; the camera circles a bay and looks down the aisle; one view has no people; 8 frames.
9:20–10:00	Client grocery market	Captured from 9:16–9:50; people block the center and were masked; 6 frames.
13:50–16:40	Commercial kitchen	Reflective stainless steel, hard for splats.

Table 4: Locally trained zones. Frames are the posed frames recorded in each zone’s export metadata; sizes are the committed files. All cost 0 credits.

Zone	Video segment	Frames	Splats	SPZ (MB)	USDZ (MB)
Racking aisle	6:08, 100 s	72	128,884	2.1	8.4
Volunteer sorting room	0:52, 30 s	68	111,400	1.8	7.3
Client market	9:13, 45 s	49	105,899	1.6	6.9
Community kitchen	14:23, 34 s	65	161,425	2.5	10.6

density by target. After conversion the pipeline re-opens the package and fails unless it finds a `ParticleField3DGaussianSplat` prim with positions. In a synthetic test with 20,000 splats at spherical-harmonic degree 1, the SPZ file was 262 KB and the USDZ package 2.0 MB, a 7.7-fold increase, because SPZ is quantized and compressed while USD stores 32-bit floats and USDZ is uncompressed by specification. For the racking aisle, the 270,423-splat SPZ file (spherical harmonics of degree 0) is 2.1 MB and its USDZ package 17.5 MB, an 8.5-fold increase. The three zones are composed into one stage, `/World/Site`, in which each zone is a separately referenced layer with its own placement, so a zone can be re-captured without touching the others; the packaged site is 48.5 MB for about 747,000 splats.

2.6 Live data reaches the twin at near-zero standing cost

The reference backend serves simulated cold-chain telemetry for three zones (freezer, cooler and dry storage, with setpoints of -18°C , 3°C and 21°C) and one forklift on a closed path from the dock through the storage aisles. It has no virtual private cloud, NAT gateway, database or provisioned capacity, so an idle twin costs nothing beyond storage. The same function also serves `GET /operations`, the operations layer described next. No real data connector is deployed yet, so all telemetry in this paper is simulated. Figure 6 shows the AWS pattern: the services in the repository’s template today (object storage, the HTTP API and its function, logs and access roles) and the real-floor path that the Discussion prices, in which sensors and the warehouse management system feed the same API. Appendix A gives equivalent designs on Microsoft Azure and Google Cloud. The stack was deployed on 25 October 2026 to us-east-2, because the account’s organization policy blocks the function, API



Figure 2: The eight input frames from the racking aisle, 1920×816 pixels after cropping the broadcast graphic. The sorting room and market inputs are in the repository.

193 and deployment services in us-east-1. AWS Cost Explorer showed \$0.00 billed for October 2026 at
 194 that date: at demo traffic the function, HTTP API and bucket stay within the free tier. Over 1,000
 195 sequential requests to `GET /telemetry` from a cloud container in the United States, paced at about
 196 three per second, all 1,000 succeeded, with a median round trip of 67ms and a 95th percentile of
 197 78ms. In an earlier run of the same size, 1.4% of requests were refused by the API's deliberate
 198 throttle (HTTP 429). These are measurements from one individual AWS account under its own
 199 quotas and policies, not a user's device or an enterprise account.

200 2.7 An operations layer turns the geometry into a working food bank

201 The captured zones show what the building looks like; an operations layer models what it does,
 202 and it follows the same three steps. In the twin's metric frame, eight operational areas (receiving,
 203 product recovery or sorting, freezer, cooler, dry racking, order staging, the client market and the

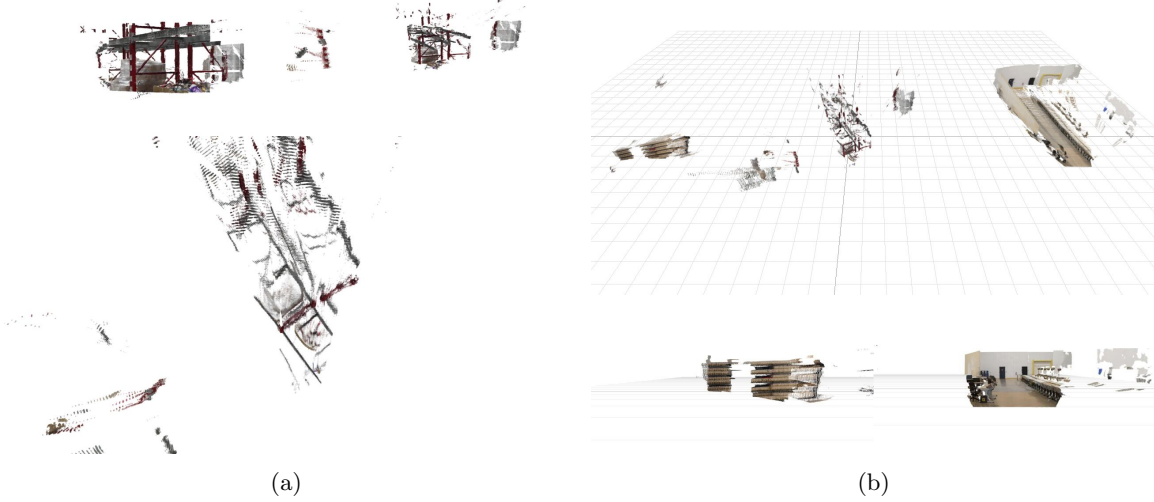


Figure 3: The fused twin rendered in Spark 0.1.10. (a) The racking aisle from two viewpoints: red racking, a wrapped pallet and the donation bin are recovered where the camera looked, and the rest of the aisle is empty. (b) Top: the three zones placed on one site plan on a 1 m grid; bottom: the market and the sorting room from their source viewpoints. No generated views are included.

community kitchen) are footprints tagged with the capture that covers them. On the data side, the same stateless function that serves telemetry also answers `GET /operations` with inventory by product family, flows between sources, areas and outbound channels, dock schedules and kitchen batches, all computed from the clock and one master-data file, with no database; a JavaScript port returns identical payloads offline, which a test enforces. For Omniverse, a script writes the layer as `operations.usda`, with one prim per area (8), kitchen station (7) and flow (16), keyed by the same identifiers as the API so that an extension can bind live values to them. One stage, `twin.usda`, composes the twin for Omniverse: the modeled facility described below (visible), the four captured zones (the locally trained versions, falling back to the world-model capture where an area has none; authored invisible so they can be switched on), and this layer. Each capture is referenced, not copied, and placed with the same configuration file the browser uses, so Omniverse, the browser and the API share one frame (Fig. 7). Each capture is cropped to its area by a 0.65 MB sublayer that only overrides the opacities of splats outside the crop box, so the capture library is never copied or edited, and a test fails if the placements in the stage and in the browser configuration drift apart. Opened with the OpenUSD library, the stage holds 452 prims, including 137 meshes and four Gaussian-splat prims, in Y-up meters.

The kitchen is modeled as a small factory, with a bill of materials and a routing per recipe: ingredients are picked from storage, and each batch of 250 meals moves through prep, cooking on a kettle, tilt skillet or combi oven, blast chilling, portioning and packing, and cold holding until dispatch. The same stance extends to the whole distribution center: inbound loads from five source types either pass through the sorting room or go straight to put-away, stock waits in its temperature zone with a days-of-supply target, and outbound pounds leave for partner agencies, mobile pantries, the market and the kitchen. Volumes are anchored to published figures for the Central Texas Food

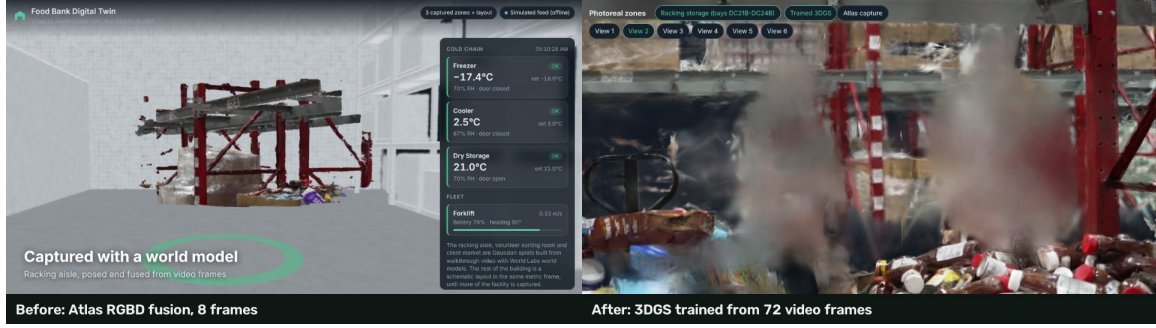


Figure 4: The racking aisle in the browser viewer. Left: the world-model capture, posed and fused from 8 frames. Right: a 3D Gaussian Splatting model trained locally from 72 frames of the same video, from a source camera, in the first training run; the grey haze is where the masked presenters stood, which the background fill and hull carving described above later reduced to a soft blur.

Table 5: Published figures that anchor the operations model. All other parameters are estimates.

Figure	Source
74 million lb distributed per year, 26.6 million lb of it produce (fiscal year 2024)	[22]
251 partner agencies in a 21-county service area	[22]
135,000 sq ft facility	[18]
More than 7,000 pallet positions, more than 35,000 sq ft of cooler and freezer, 15 docks	[23]
4,200 sq ft production kitchen with kettles, combi ovens, blast chillers and tilt skillets	[24]
About 950 kitchen meals per day for after-school meals (2018); about 2,000 a day by 2025, including summer meal sites	[25, 26]
1.2lb of food per meal	[27]

Bank’s main facility (Table 5); every other parameter, including source and channel shares, product mixes, pallet weights, capacities, time windows, equipment counts, recipes and step times, is an estimate and is flagged as one in the master data.

Over eight simulated weeks, a weekday received 231,000 to 296,000 lb (median 258,000) and distributed 220,000 to 300,000 lb (median 260,000, about 217,000 meal equivalents), with 4.49 to 4.63 million lb on hand, or 22.1 to 22.8 days of supply. Each weekday had 13 to 17 inbound trucks and 59 to 80 partner pickups, and the kitchen produced 912 to 997 meals; Saturdays run at reduced volume and Sundays are closed. These numbers follow from the published annual volume and the estimated parameters; they are a plausible operating rhythm for the twin to display, not a measurement of the food bank.

2.8 A modeled facility gives the whole-building view the captures cannot

The captures answer *what does this aisle look like*, but only from near the camera path: from an overview they break into fragments, and a phone screen showed the twin as debris over a grey massing model (Figure 8b). For the view a manager or volunteer needs first, where everything is and how it is doing, the twin’s default scene is therefore a modeled facility (Figure 8a). One script builds it headless with Blender’s Python module [28] in the telemetry frame, from the operations layer’s

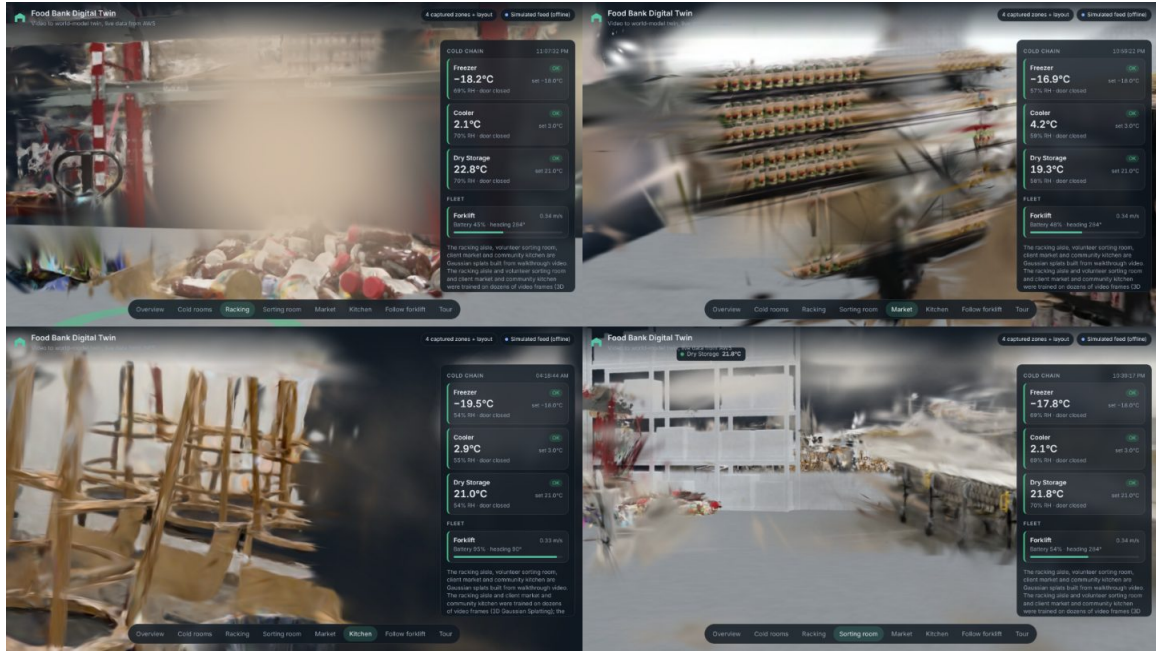


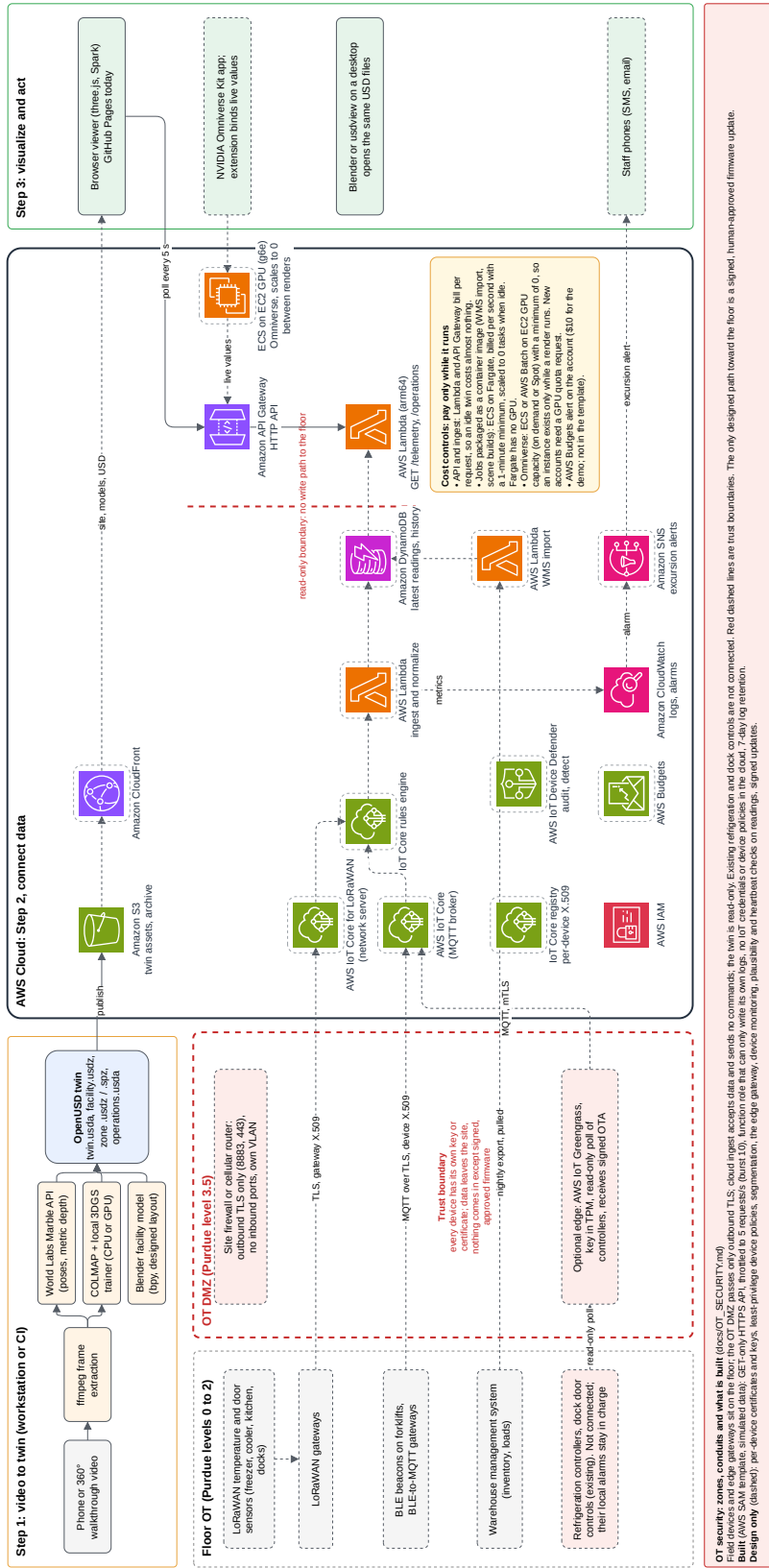
Figure 5: The four locally trained zones in the browser viewer, each from one of its source cameras: racking aisle (top left), client market (top right), community kitchen (bottom left) and volunteer sorting room (bottom right). The panels show simulated telemetry. Soft regions are where the presenters stood or the camera did not look.

footprints and the cold-room probe positions, so the labels, the forklift route and the operations page line up with it. It contains the shell and dock doors with a trailer, the freezer and cooler as cutaways with racking inside, four rows of selective pallet racking, receiving and order staging, the volunteer sorting room with tables, roller conveyors and scales, the client market with wire shelving, a produce display, a fridge case and checkout, the community kitchen annex with kettles, tilt skillets, combi ovens, blast chillers, prep tables, a pack line and a walk-in, forklifts, painted lanes and signage. The same scene is written as a 1.8MB binary glTF file for the browser and a 1.6MB USDZ package for Omniverse, which the OpenUSD library opens as 129 meshes in Y-up meters. The model is designed, not reconstructed: the fit-out is inspired by the walkthrough video and the layout comes from the operations layer, the building is a compact 26×20 m stand-in plus an annex rather than a 135,000sqft warehouse, and the viewer labels it a representative food bank, not a replica. The captures stay one switch away, in the browser and in the USD stage.

2.9 The browser shows the twin with live telemetry

The public demo at <https://pikulshomesh.github.io/foodbank-spatial-twin/> opens on the modeled facility, drawn with three.js, with labels for the freezer, cooler, dry storage and kitchen walk-in temperatures, the forklift's battery, and a cold-chain panel (Figure 9). The model's forklift moves along the position and heading the telemetry reports. Preset views open the cold rooms, kitchen, racking, sorting room and market, and a portrait layout serves phones. A URL switch

Food bank digital twin on AWS Cloud



Solid: in the repository today and deployed in us-east-2 (AWS SAM template; simulated data). Dashed: real-floor path priced in docs/COST_ESTIMATE.md from public list prices; directional, enterprise pricing may differ. Red dashed: OT trust boundaries (docs/OT_SECURITY.md).

Figure 6: The food bank twin on AWS. Solid borders are in the repository today (AWS Serverless Application Model template; simulated data); dashed borders are the real-floor path priced in Table 7. Step 1 runs on a workstation or in continuous integration and publishes the OpenUSD twin to object storage; Step 2 connects floor sensors and the warehouse management system; Step 3 is the browser, OmniVerse on an on-demand or Spot GPU instance that is stopped after each session, desktop tools and staff alerts. The cost-controls panel lists how each part bills only while it runs. Red dashed lines are OT trust boundaries (Purdue levels 0 to 2 on the floor and an OT DMZ at level 3.5); the twin has no write path to the floor, and the bottom panel separates the controls that are built from those that are design only. Drawn in draw.io with the AWS architecture icons; the source is in the repository's paper/diagrams folder.

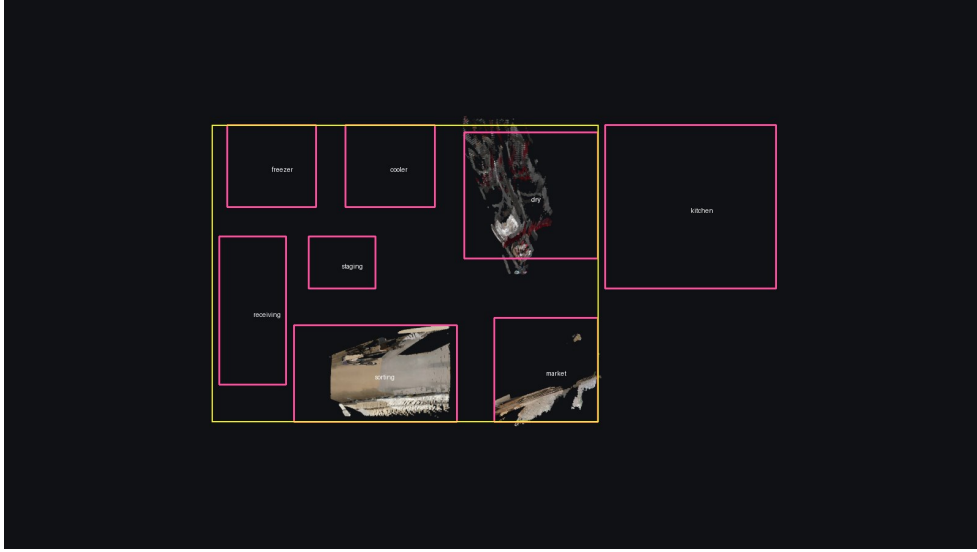


Figure 7: Alignment check: top view of an earlier version of the composed OpenUSD stage, rendered on the CPU, with the operations-layer footprints drawn over it (pink: areas; yellow: building). The world-model captures of the sorting room, dry racking and market land in their areas; the racking capture extends slightly past its footprint. The current stage keeps these placements and adds the trained zones and the modeled facility.

composites the captured zones with Spark [29], using the four locally trained zones (8.0 MB in total) in place of the world-model captures of the same areas (5.9 MB), and a separate page compares each zone from its source cameras. The site runs from static files, falls back to a local copy of the simulator when no backend is configured, and was checked in desktop and phone layouts in headless Chromium. Served locally to headless Chromium with software WebGL (SwiftShader, four CPU cores, 1280×720), the default view transferred 2.7 MB and finished loading in 1.5 s, and the view with captures transferred 60 MB in 2.7 s. Without a GPU it drew 1.1 and 0.5 frames per second, a floor rather than a device figure; frame rates on laptops and phones with a graphics processor were not measured.

2.10 The OpenUSD stage renders outside the browser

Two renderers read `site.usdz` directly on a four-core CPU machine with no GPU, from the same list of six camera shots (Figure 10). A Gaussian-splat renderer written for this work walks the stage for every splat prim, applies each prim’s transform to positions and covariances, and composites the splats in depth order; it rendered all six stills in about 20 s including loading, and a 24 s, 576-frame 1080p flythrough in a few minutes. Blender 5.2’s own USD importer brings in the transform hierarchy but turns each splat prim into an empty object, because it does not read the `ParticleField3DGaussianSplat` schema. Reading the splats with the OpenUSD library that ships inside Blender and drawing each as an emissive sphere gave renders in Cycles that match the splat renderer shot for shot, at about 5 s per 960×540 frame with 8 samples. In both, each zone reads well

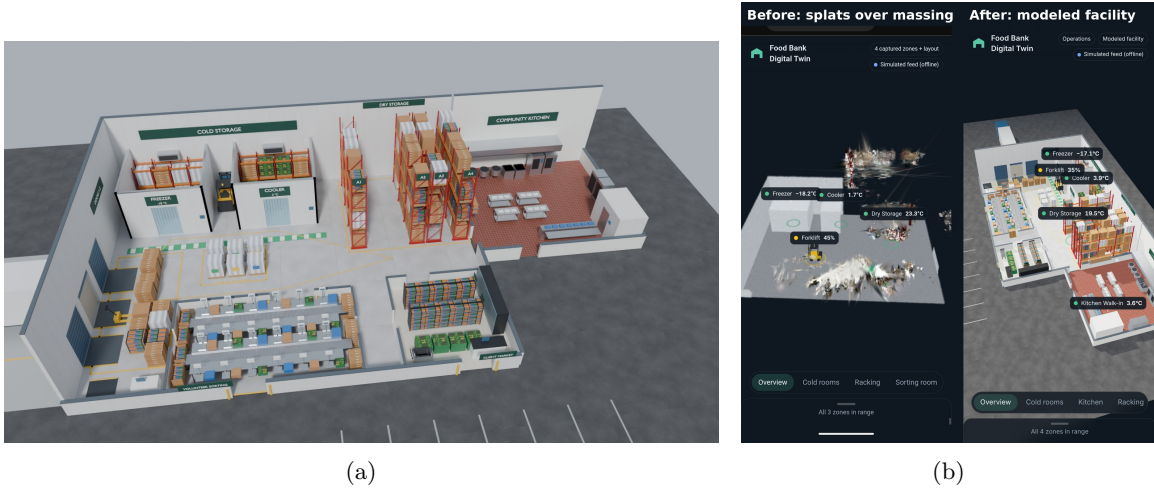


Figure 8: The modeled facility. (a) Cycles render of the Blender scene: cold storage, pallet racking, receiving and staging, the volunteer sorting room, the client market and the community kitchen annex. (b) The browser overview on a phone before (captured splats over a massing model) and after (the modeled facility), both with simulated telemetry.

only near its source viewpoints; wide shots break into fragments because the captures are sparse, which more frames or generated views would address, not a different renderer, and they are why the modeled facility is the default view. The modeled facility itself is ordinary meshes and materials written by Blender’s USD exporter, so any OpenUSD reader can open it. Omniverse rendered the modeled facility on a rented GPU after six failed or partial runs (Figure 11, Movie S3). On 25 October 2026 we ran the Isaac Sim 5.1 container (Kit 107.3) headless on EC2 in us-east-2, each run on an instance that terminated itself. On a g6.xlarge (one NVIDIA L4) with the GPU image’s 595-series driver, Kit crashed in its RTX scene database right after start, even for a bare lit cube, so neither the twin files nor the splat schema caused it. On plain Ubuntu with the 580-series driver the container is validated on, Kit opened the stage. With no g6 capacity left in the region, the final run used a g5.xlarge (one NVIDIA A10G) and rendered three 4s, 96-frame clips of the modeled facility at 1280×720 with eight RTX subframes per frame, in 18.6s, 47.7s and 49.4s, about 2 to 5 frames per second offline. Kit 107.3 drew the splat captures’ `ParticleField` prims as blank grey, so Omniverse does not yet show the photoreal zones; the browser and the CPU renderers do. The clips use a scripted forklift path, not live telemetry. Instance boot took about 1 minute, pulling the container about 4.5 minutes, and Kit start-up 75s cold and 14s warm. Across seven runs the trial used about 94 instance-minutes, about \$1.34 at on-demand list prices, and left no instances or volumes behind. Interactive frame rate in Omniverse, and binding live telemetry to zone prims there, were not measured; both need an interactive Kit session rather than headless clip rendering.

2.11 Summary of measurements

Table 6 collects the cost, time and fidelity measurements, the three properties that decide whether a small operator can adopt the approach.

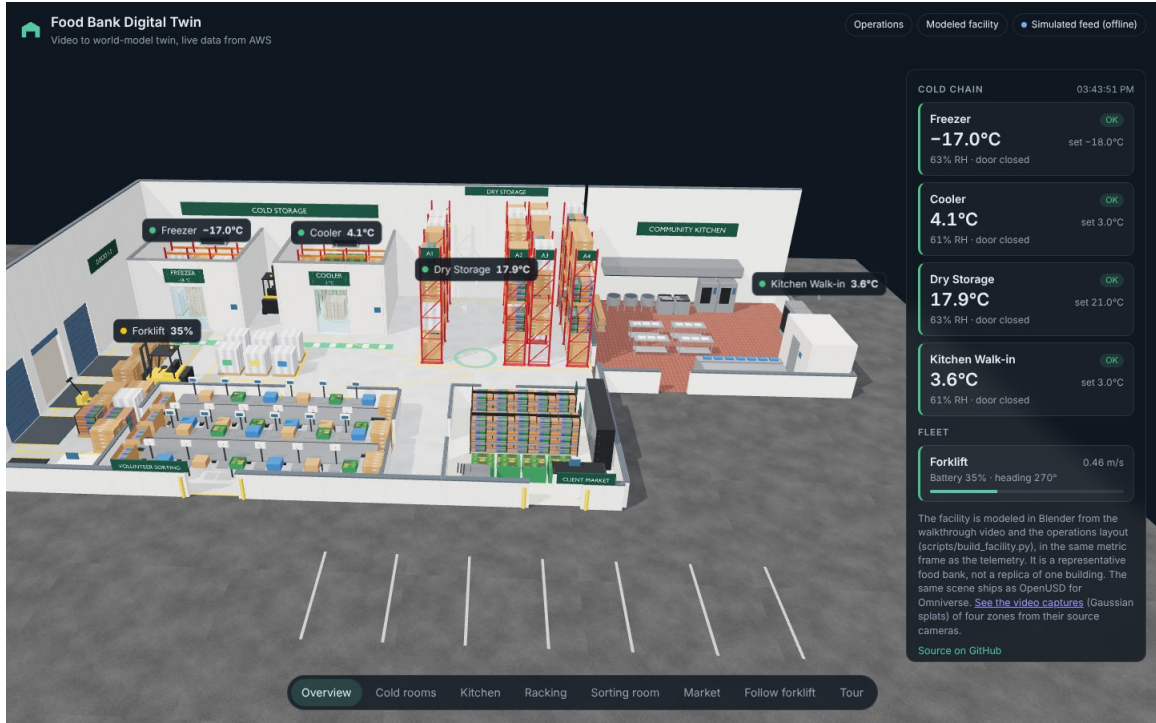


Figure 9: The browser viewer on a desktop, opening on the modeled facility with simulated telemetry labels and the cold-chain panel. The buttons open preset views; the panel links to the captured zones.

3 Discussion

This study set out to learn what current world models and Gaussian splatting recover from in-the-wild video. The main finding is that the observed parts of a real scene come back cheaply and at metric scale, while unobserved space does not yet. A consequence is that a spatial digital twin does not have to start with a CAD model or a laser scanner. Every input to this one was publicly available: a broadcast news tour, published facility figures and open-source or pay-per-use tools. No drawing, scan or site visit was used. A short walkthrough video, a world model and OpenUSD’s native splat schema produce a metric twin, Blender supplies the parts the camera never saw, and a serverless data layer connects it to the systems an operator already has for almost no standing cost. One author, working with AI coding assistants, went from the first commit to a deployed backend in seven days (19 to 25 October 2026). That speed comes from composing existing technologies through open interfaces, not from new models. Keeping OpenUSD and a small JSON contract as the only interfaces makes each step replaceable as world models, data platforms and renderers improve, and lets an organization choose a renderer by hardware, cost and license posture: OpenUSD is open source, Blender is GPL, and Omniverse combines open components with NVIDIA-licensed runtime pieces. On a public interview video, 20 frames and 200 credits of world-model calls produced a three-zone OpenUSD twin that the browser renders with live telemetry.

The main limitation in the current results is coverage. Twenty frames from an interview walk-

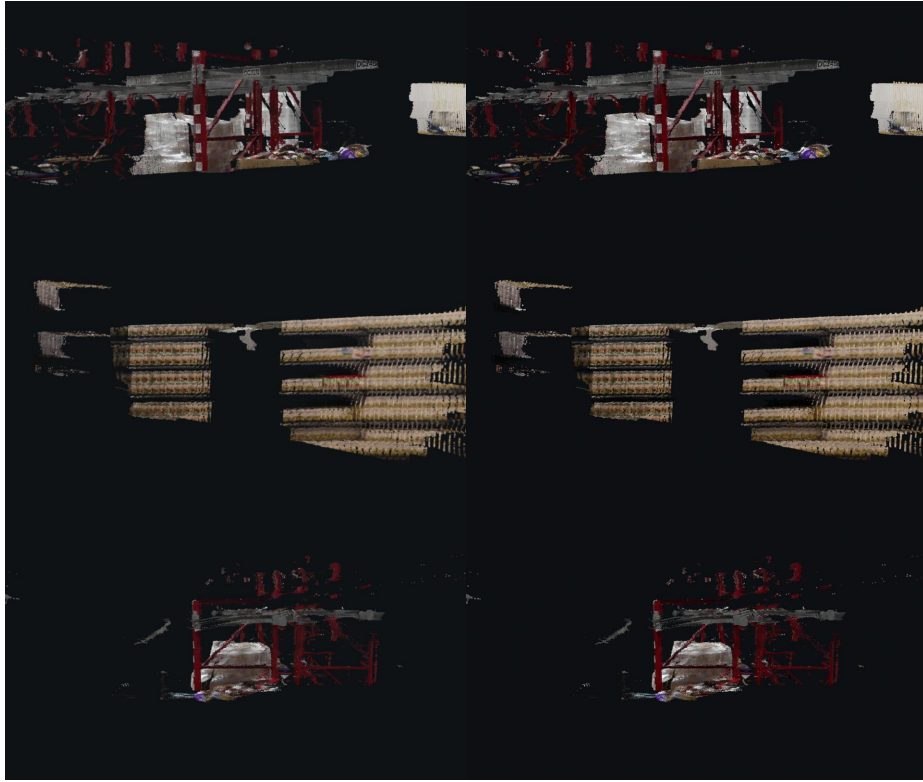


Figure 10: The same OpenUSD stage rendered on a CPU, from identical cameras: the Gaussian-splat renderer (left) and Blender 5.2 Cycles with one emissive sphere per splat (right). Rows: the racking aisle, the client market, and the racking aisle from an oblique viewpoint. All three shots are near the zones’ source viewpoints.

320 through recover the parts of three zones the camera pointed at, not whole rooms, and the generative
 321 step that should fill the rest failed on the provider’s side. Until it works, the whole-building view is a
 322 designed model, which explains the operation well but is not a reconstruction; the honest summary
 323 is that video and world models gave the twin its photographs of real places, while its map is drawn.
 324 A second limitation is that generated is not measured. A world model completes what the camera
 325 did not see. That is useful for a planning or communication twin and unsafe if someone reads a
 326 generated aisle as the real one. Generated regions should be marked in the stage, for example with
 327 an attribute on splats or a separate USD layer, so renderers can show them differently. No generated
 328 regions exist in the twin yet, so no such marking has been implemented; designed content is kept in
 329 its own model and layer so it cannot be mistaken for a capture. Scale comes from the world model’s
 330 metric estimate rather than a survey, so absolute dimensions should be checked against at least one
 331 known measurement before the twin informs layout decisions. Nor did we measure fidelity on held-
 332 out frames: the zones were judged by eye from their source cameras, so a quantitative comparison
 333 of the world-model and locally trained captures remains to be done. People, motion blur, reflective
 334 steel and repetitive racking remain hard inputs; masking people before reconstruction would help,
 335 and the world model’s API exposes masked inputs that we have not yet evaluated.



Figure 11: The modeled facility rendered by NVIDIA Omniverse (Isaac Sim 5.1, Kit 107.3, RTX) on an EC2 g5.xlarge: an orbit over the building (left), the forklift in a racking aisle (middle) and a chase view toward the receiving area (right). Frames from the three clips of Movie S3; the forklift follows a scripted path, not live telemetry, and the splat captures are not shown because this Kit version draws them blank.

Table 6: Measurements for the twin. All values are single measurements; fusion time was not recorded, and no hold-out fidelity was measured (see Discussion).

Quantity (unit)	Value	How measured
Input frames used	20 (8 + 6 + 6)	hand-picked from 40 per zone
Frame extraction, racking window (s)	88	wall clock, 4 CPU cores
World-model cost, posing (credits; USD)	200; 0.20	10 credits per frame, rate card
World-model cost, failed generation (credits)	0 to 800	billing of failures unknown
Splats, racking / sorting / market	270k / 282k / 195k	fused files
SPZ size, racking / sorting / market (MB)	2.1 / 2.2 / 1.6	committed files
USDZ size, racking / site (MB)	17.5 / 48.5	committed files
World-model generation time (min)	n/a	both runs failed
SPZ to USDZ, 270k / 182k splats (s)	5.2 / 3.9	wall clock, 4 cores
Locally trained zones (credits)	0	four zones, Table 4
Modeled facility, GLB / USDZ (MB)	1.8 / 1.6	committed files
Telemetry latency, p50 / p95 (ms)	67 / 78	cloud client, 1,000 requests
AWS cost, October 2026 to date (USD)	0.00	Cost Explorer, demo traffic
Browser download, default / captures (MB)	2.7 / 60	headless Chromium
Browser frame rate, no GPU (fps)	1.1 / 0.5	software WebGL, a floor
Omniverse offline render, 96 frames (s)	18.6 to 49.4	A10G, 720p, 8 subframes
Omniverse GPU trial, 7 runs (min; USD)	94; 1.34	list price, terminated

The twin is also a composite by design. Its captured geometry comes from a video that appears to show the food bank’s Waco hub, its operating volumes come from published figures for the 135,000sq ft Austin facility, the rest of the building is designed, and the data is simulated. That makes it a fair picture of what a Texas food bank’s twin looks like, but not a record of any one site, and its provenance (Table 2) should travel with every image of it. A twin of a real facility would replace each designed or simulated row with a capture or a live feed from that facility. Keeping these sources in one frame is what lets the twin be used at all: one USD stage composes the modeled facility, the captured zones (by reference, placed by the same configuration as the browser) and the operations layer, and cropping is a non-destructive opacity override, so the capture library is never copied. A new capture or a new data feed therefore changes one layer rather than the whole stage.

Applying the approach also needs care outside the software. Walkthrough videos capture staff, volunteers and clients, so faces should be removed or the space filmed when empty, and a facility’s

348 permission obtained before a reconstruction is published. The next steps are a first real data connec-
349 tor, most cheaply temperature loggers publishing over MQTT to AWS IoT Core [30], a person-aware
350 frame picker, and training the locally reconstructed zones at full resolution on a GPU. The first com-
351 parison with a conventional splat reconstruction already shows the division of labor: the world model
352 gives metric scale and usable geometry from eight frames, while local training needs about nine times
353 as many frames and a scale assumption but delivers photorealism.

354 What would it cost to run the twin on real floor data? We priced one representative 135,000 sq ft
355 facility from list prices on 19 October 2026 (Table 7); this is an estimate under stated assumptions,
356 not a measured bill. Battery LoRaWAN sensors report freezer, cooler, kitchen and dock temperatures
357 and door states through AWS IoT Core for LoRaWAN, BLE beacons place each forklift in an area,
358 the warehouse management system supplies inventory, and the existing API serves the result with
359 the `simulated` flag cleared. About \$2,700 of hardware and about \$12 a month of AWS services cover
360 the core, so the cloud is not the barrier; the hardware, its installation and keeping the master data
361 current are. Running Omniverse on an AWS GPU only when needed adds about \$124 a month,
362 while an always-on GPU would add about \$2,100, which is why the browser is the everyday viewer.
363 The Omniverse trial gives a measured reference point: seven runs used about 94 instance-minutes
364 on g6.xlarge (\$0.8048 per hour on demand) and g5.xlarge (\$1.006 per hour) instances, about \$1.34
365 computed from list prices because the bill lags by a day, and every instance was terminated with no
366 volumes left behind. Scaled up, on-demand rendering on ECS or AWS Batch with a GPU minimum
367 of zero costs about \$0.80 per GPU-hour actually used, plus about 6 minutes (about 8 cents) of
368 container pull and Kit start-up per cold job; ten GPU-hours a week would be about \$35 a month
369 on a g6.xlarge or \$44 on a g5.xlarge, with no software fee for the container in this trial. Scaled to
370 a complete capture of a 135,000 sq ft facility with the same tools (twelve zones, with world-model
371 poses and generated views, one GPU-hour of splat training per zone and ten hours of Omniverse
372 review), the cloud spend from video to deployed twin would be about \$25 to \$50 at list prices, before
373 staff time; the assumptions are in `docs/COST_ESTIMATE.md`. Assumptions, unit prices and sources are
374 in the repository’s `docs/COST_ESTIMATE.md`.

375 The same principle, paying only while something runs, extends to every part of the design and
376 is drawn as a cost-controls panel in each architecture diagram (Figs. 6, 12 and 13). The API and
377 ingest functions already bill per request, so an idle twin costs almost nothing. Work that is easier
378 to ship as a container image, such as a warehouse import that outlasts a function’s time limit or
379 a scene rebuild, can run on Amazon Elastic Container Service (ECS) with AWS Fargate, which
380 bills per second of virtual CPU and memory with a one-minute minimum and costs nothing at
381 zero tasks; Azure Container Apps and Google Cloud Run do the same and scale to zero on their
382 own. Containers are not automatically cheaper: an always-on Fargate task of 0.25 virtual CPU and
383 0.5 GB on ARM costs about \$7 a month at us-east-1 list prices, more than the whole demo API,
384 so containers suit jobs rather than the polling API. Fargate offers no GPUs, so Omniverse needs
385 an EC2 GPU instance, started for a session and stopped afterward; Spot capacity lowers its price
386 further but can be reclaimed mid-session, and a stopped instance’s disk still bills. On AWS the same
387 pattern can be automated by running the Omniverse renderer as an ECS service or AWS Batch job
388 on GPU EC2 capacity whose Auto Scaling group has a minimum of zero, so an instance exists only

while a render runs. A new AWS account starts with no on-demand GPU quota, so a quota request is part of deployment. Azure Container Apps and Cloud Run do offer serverless GPUs that scale to zero, which could host a headless renderer; we have not tested Omniverse on them. A budget alert on each cloud catches anything left running. All prices here are public list prices, and the measurements come from an individual AWS account and the quotas and policies it had. Enterprise accounts may differ through negotiated discounts, savings plans, quotas and region policies, so the figures are directional, not a quote.

Operational technology security

Most security discussion around a cloud twin is about IT: the account, the API and the web front end. A twin fed by floor sensors also brings operational technology (OT) into scope: the sensors, the gateways that carry their data and the path from the floor to the cloud. We therefore designed the twin’s OT boundary before any real device is connected (Figure 6), following the zone-based segmentation and Purdue levels described in NIST SP 800-82 Rev. 3 [31] and the zones-and-conduits model of the ISA/IEC 62443 series [32, 33].

The central decision is that the twin watches the floor and never drives it. The API exposes only `GET` routes, and in the design no cloud principal may publish to a device topic, so even a full compromise of the cloud account could mislead people but could not change a freezer setpoint. A twin that sends commands back would become part of the control system and would need far more assurance. For the same reason the design adds its own battery LoRaWAN sensors rather than tapping the refrigeration controllers, whose local alarms stay authoritative. Floor devices and gateways (Purdue levels 0 to 2) sit behind an OT DMZ (level 3.5) that allows only outbound TLS, and the only designed path toward the floor is a signed, human-approved firmware update.

For a read-only twin the main OT risk is integrity rather than control. A falsified “all normal” reading could hide a cold-chain failure, and falsified alarms could train staff to ignore real ones. MITRE ATT&CK for ICS lists such falsified telemetry as an unauthorized reporting message (T1692.002) [34], and NIST SP 800-82 notes that many field devices and protocols allow data to be replayed, modified or spoofed [31]. The design answers with one identity per device: an X.509 certificate for each MQTT device, with an AWS IoT policy that lets a device connect only under its own registered thing name [35] and certificates that can be revoked permanently [36], and per-device AES-128 session keys for LoRaWAN sensors. Readings would then pass plausibility, rate-of-change and data-age checks before they reach staff. One trust point should be stated plainly: AWS IoT Core for LoRaWAN decrypts sensor payloads in the cloud [37], so the cloud provider is inside the trust boundary for that data.

What is built and what is designed must be kept apart. Implemented and deployed today are the cloud-side controls: the `GET`-only HTTPS API, its throttle of 5 requests per second (burst 10), a function role that can only write its own logs, seven-day log retention, and no IoT resources or device credentials in the account. Every device-side control above is a design, because the demonstration uses simulated telemetry and no real device exists. The target security levels are reasoned, not the result of an IEC 62443 assessment, and the public API has no authentication because the data it

Table 7: Estimated cost of running the representative twin on real floor data, from public list prices checked on 19 and 26 October 2026. Excludes tax, installation and staff time. Directional: enterprise discounts and quotas may differ.

Item	One-time	Monthly
24 temperature sensors, 21 door sensors, 2 LoRaWAN gateways	\$1,850	
8 forklift beacons, 10 BLE gateways (area-level location)	\$840	
AWS: IoT Core, Lambda, API Gateway, DynamoDB, logs (20 viewers)		\$12
Optional: Omniverse on a g6e.xlarge, 10 h a week		\$124
Measured: Omniverse trial, 7 runs, 94 instance-minutes ^a	\$1.34	
Scaled: on-demand rendering on g6.xlarge, 10 h a week		\$35
Optional: forklift telematics, 8 vehicles		\$200–400

^aComputed from list price; the bill was not yet available.

serves is public and simulated; a deployment with real readings would add both.

4 Materials and Methods

4.1 Experimental Design

The objective was to build a working twin of a food bank from a public video and to measure cost, time and fidelity at each step (Table 6). The design was fixed before any paid call: one zone first (the racking aisle, eight frames), then two more zones with six frames each, the lowest-cost API tasks available, and a project ceiling of 1,000 credits. All code, the AWS template and the input frames are in the project repository.

4.2 Frame extraction

World-model APIs accept only a few images (Marble’s Auto Layout multi-image mode accepts up to eight [38]). The ingest stage samples a window of the video and, for every interval of N seconds, scores candidate frames with ffmpeg’s blur detector and keeps the sharpest; interlaced frames are deinterlaced only when flagged. A hard cap of 40 frames bounds storage and upload. For the case study, frames were taken from 5:05–7:45 at one per 4 s with the bottom 264 pixels cropped to remove the broadcast graphic, giving 40 candidates at 1920×816 in 88 s of wall-clock time on four CPU cores for the 160 s window; all inputs share one resolution, which reconstruction requires. The automatic picker splits the pool into eight equal segments and keeps the frame with the highest Laplacian variance in each; for the case study it was replaced by hand selection as described in Results. A text prompt described the space and asked for no people.

4.3 World-model reconstruction

The pipeline implements two routes on the World Labs platform. The first, a multi-image request to Marble that returns a finished Gaussian-splat world in SPZ format [39] with a metric scale factor and ground-plane offset [40], needs a Marble v1 key and was not available to the account used here. The reported twin was built with the second route, through the Marble 2 developer API, in three

452 stages. In the pose stage, `images2PosedRGBD` returns a camera pose, metric depth and confidence for
 453 each input frame (10 credits per frame on the rate card of 19 October 2026, where 1,000 credits
 454 cost US\$1). In the generate stage, `atlasGenerate` is asked for new posed RGB-D views from target
 455 cameras around the zone, with the masked presenters excluded from its context (50 credits per view);
 456 this is the step in which the world model extrapolates unseen space, and it failed in both attempts
 457 (Results). In the free, local fuse stage, every RGB-D view is unprojected, the most confident sample
 458 per 1 cm voxel is kept, masked pixels are dropped, and the points are written as isotropic Gaussians
 459 with spherical harmonics of degree 0, as PLY and SPZ version 2. The floor is taken, in order of
 460 preference, from a dense 5 cm horizontal slab of points, from the world model’s ground plane, or from
 461 a fixed 1.5 m below the cameras. A site stage then places each zone on a plan in meters (racking
 462 at the origin, sorting at +26 m and the market at −24 m on x) and writes one USD stage that
 463 references each zone’s layer. This plan is a library layout, not a survey; the composed twin re-replaces
 464 the zones as described under Visualization.

465 4.4 Locally trained zones

466 The photoreal route uses no paid service. Frames from a zone’s segment of the video are extracted at
 467 4 to 6 frames per second and 960 pixels wide. People are masked with a DeepLab v3 segmentation
 468 model run through MediaPipe, the masks are grown by 35 pixels and unioned with neighboring
 469 frames, and the broadcast graphic is masked by a fixed box. Under the masks, a background plate
 470 is built from pixels warped by homography from frames up to 15 s away in which the presenters had
 471 moved, with a blurred inpaint where no frame saw the background. Frames are kept if they are not
 472 in the blurriest 20%, at least 35% of their pixels are unmasked, and the median optical flow since the
 473 last kept frame is at least 14 pixels. COLMAP 3.9 estimates poses with masked SIFT features on the
 474 CPU, with sequential matching, or exhaustive matching and a 3° initial triangulation angle for the
 475 low-texture sorting room. The trainer implements 3D Gaussian Splatting [5] in plain PyTorch with
 476 a sparse per-pixel rasterizer, so that it runs without a GPU (about 1 s per step at 480 pixels on four
 477 cores): elliptical weighted average projection, front-to-back compositing, an L1 plus 0.2 D-SSIM loss
 478 that sees the background plate at 0.3 weight under the masks, visual-hull carving every 250 steps
 479 (a splat is removed when its center falls on a presenter mask in more than half of the five or more
 480 frames that see it), adaptive cloning, splitting and pruning, and opacity resets every 3,000 steps,
 481 at half resolution for the first quarter of the steps. The documented run trains each zone for 6,000
 482 steps, roughly 1.7 h at the rate above; per-zone wall-clock times were not recorded. Export levels
 483 the scene to the mean camera up direction, takes the floor from the lowest well-supported band of
 484 structure-from-motion points below the cameras, scales so that the cameras sit 1.6 m above it, keeps
 485 splats inside at least six camera frustums and within 80 m of the path, and writes the same SPZ,
 486 USDZ and index files as the world-model zones; the composed twin crops each zone to its room.

487 4.5 Conversion to OpenUSD

488 The splat world is converted with `usd-convert-gsplat` 0.1.15 [41] into a `ParticleField3DGaussianSplat`
 489 prim. World Labs SPZ files use an OpenCV-style camera convention (`marble_raw_opencv`, Y down)

in raw units that the world’s metric scale factor converts to meters [40]. Leaving the splat payload untouched, we place it under a parent transform, as shown in Equation 1,

$$M = R_x(180^\circ) T(0, -o, 0) S(s), \quad (1)$$

where s is the world’s metric scale factor and o its ground-plane offset, and set the stage to Y-up with `metersPerUnit=1`. With synthetic input the ground lands at exactly $y = 0$. A typed `Xform` that references the splat would override the referenced prim’s schema type, so the pipeline authors `/World/FoodBank` as the transforming `Xform` and `/World/FoodBank/Splat` as an untyped prim carrying the reference. The converter’s direct USDZ output is an ASCII layer zipped without the 64-byte alignment the USDZ specification requires, so we convert to binary `.usdc` and package with OpenUSD’s `UsdUtils.CreateNewUsdzPackage`. Conversion took 5.2s for the 270,423-splat world-model racking zone and 3.9s for a 181,556-splat first version of the locally trained one on four CPU cores. The schema family defines half-precision companions to its attributes, such as `positionsh` [42], that would roughly halve file size; the converter does not yet write them.

4.6 Cost controls for paid generation

World-model calls are the only paid, non-idempotent steps, so they are guarded in code. On the Marble 2 route, each paid stage records its operation identifier before waiting and sends an idempotency key derived from the request body, so a re-run resumes or reuses a result instead of paying again; a credit ceiling is passed per stage. The Marble v1 route adds the following guards. Only the draft model is allowed unless explicitly overridden, and the model is checked again in the payload and in the returned world. Each process makes at most one billable call. The operation identifier is saved the moment a job is accepted, so a crash resumes polling instead of paying again, and a finished world is reused unless a new one is forced. A lifetime ledger refuses any run that would exceed the budget, a live balance check requires a reserve, and the billable request is retried only on HTTP 429, which the API defines as not accepted, never on timeouts or server errors. These guards were exercised against a local mock of the API covering a 429 retry, a 503 during polling, resume in flight with exactly one generation across three runs, reuse, refusal of non-draft models and ledger accounting.

4.7 Serverless data backend

The backend is one AWS Serverless Application Model template [43]: an S3 bucket serving the twin assets and web viewer; a Python 3.12 Lambda function on arm64 with 128MB of memory and a 3s timeout behind an HTTP API with `GET /telemetry`, `GET /operations` and `GET /health`, throttled to 5 requests per second (burst 10) as a cost ceiling; and a log group with seven-day retention. The telemetry function is stateless and deterministic: every value is a function of the wall clock, so clients in the same second see the same state without shared storage. Zone temperatures follow a slow sinusoidal drift with a 15-minute period plus per-5-second jitter, doors open in about 10% of 30-second windows, and alarm status is set when the deviation from setpoint exceeds the zone’s

band. Responses are cacheable for 2s and every payload carries `simulated: true`. Zone and asset anchors are in meters in the twin frame, the frame the modeled facility is built in, so labels sit on the modeled cold rooms by construction. Each capture’s placement and crop box are set by hand in a configuration file; their residual error against the real building is not measured, because no survey exists.

4.8 Operations model

The master data lists areas with footprints and positions in meters, six inventory families with a home temperature zone, pallet weight, shelf life and days-of-supply target, five inbound source types and four outbound channels with their shares and product mixes, and the kitchen’s programs, equipment, recipes with per-meal bills of materials, and routing. Each value is either tied to a reference or marked as an estimate. Daily inbound follows the annual volume, a weekday profile and deterministic noise; outbound is set so stock tracks a slow target curve, so one day’s closing stock is the next day’s opening stock. Areas whose cold-chain sensor is in alarm report their stock as at risk; the kitchen walk-in is a fourth cold-chain sensor in the telemetry feed. Kitchen batches of 250 meals start at 06:00 local time, staggered by 45 minutes, and are dispatched between 13:00 and 15:30. Tests check that the Python and JavaScript payloads agree, that stock carries over midnight and that Sunday is closed. The weekday ranges in Results were computed by evaluating the model at the end of each day from 1 September to 26 October 2026.

4.9 Visualization

The modeled facility is built by `scripts/build_facility.py` with the `bpy` module of Blender on Python 3.13, without a display. Footprints come from the operations master data; materials use small procedural textures generated by the script, so the build needs no downloads. One mesh per area and material keeps the browser to about 150 draw calls. The script exports the scene as binary glTF (Y up, transforms applied) and as USDZ with materials, and can also save a `.blend` file and a Cycles still (48 samples, denoised, CPU). A test checks that the glTF file carries every operational area and the forklift node and that the building floor spans the telemetry frame.

The stage to open in Omniverse is `twin.usda`. A script writes it from the browser configuration: a reference to the modeled facility, one reference per capture with the viewer’s translation and rotation, a crop sublayer that sets the opacity of every splat outside the capture’s crop box to zero, and a reference to `operations.usda`; a test compares the stage’s placements with the configuration. A second script renders the captures on the CPU with the area footprints drawn over them as a visual check. In Omniverse, a small extension is planned that polls `/telemetry` and updates prims under the twin: zone markers change color with alarm status, labels show temperatures, and the forklift follows its reported position and heading. The clips in Movie S3 were rendered by `scripts/omniverse_render_clips.py` inside the Isaac Sim 5.1 container, launched by `scripts/gpu_trial.py` on an instance that uploads its output through presigned S3 URLs and then terminates itself. Because `/Facility` carries the Z-up to Y-up rotation, the forklift’s world pose is mapped into that frame before it is set.

563 The browser viewer uses three.js 0.180 and its glTF loader for the modeled facility, and Spark
564 2.2 [29] only when the captures are switched on. It deploys as static files to GitHub Pages; common
565 USD web viewers hand USDZ to AR Quick Look, which does not render USD particle fields. The
566 CPU splat renderer uses usd-core, NumPy and Numba, projects each Gaussian with an elliptical
567 weighted average filter, colors it from its degree-0 spherical harmonics and blends front to back.
568 For the captures in Blender, a script reads every splat prim with the `pxr` module, builds one point
569 cloud per prim with Geometry Nodes (radius 1.7 times the splat scale, emission color, opacity as
570 transparency) and renders with Cycles on the CPU. Spheres are not Gaussians, so edges are harder
571 than in the splat renderer.

572 The architecture diagrams (Figure 6 and Appendix A) are generated as draw.io files by
573 `paper/diagrams/make_diagrams.py` with the AWS, Azure and Google Cloud icon libraries of draw.io
574 26.2.15 and exported to PDF with draw.io desktop.

575 4.10 Statistical Analysis

576 This is a single case study, and every value reported is a single measurement: counts and file sizes
577 from the committed files, times from one wall-clock run on a four-core CPU machine, and costs from
578 rate cards or list prices. No repeated runs, hypothesis tests or confidence intervals are reported. The
579 weekday ranges of the operations model are the minimum, median and maximum over 40 simulated
580 weekdays from one deterministic run. No fidelity metric on held-out frames was computed; when
581 it is, frames from each extraction window that were not used for training should be posed against
582 the reconstruction and compared in observed regions only, because generated content has no ground
583 truth. Telemetry latency, once the backend is deployed, will be reported as the 50th and 95th
584 percentiles of 1,000 client-side requests.

585 4.11 Animal and Human Studies

586 No human subjects research was performed. The input is a publicly broadcast news video; people
587 who appear in it were excluded from input frames where possible and the world model was prompted
588 to omit them. The video is used as published by its broadcaster; the twin is a representative facility
589 inspired by it, not a record of the building.

590 Acknowledgments

591 General

592 This project was inspired by 25 News KXXV’s public video tour of the Central Texas Food Bank’s
593 new facility [19, 20] and by the food bank’s published account of its work. Neither organization was
594 involved in or endorsed this study. During the preparation of this work, the author used Claude
595 (Anthropic) in order to write and test the pipeline software and to draft the manuscript. After using
596 this tool, the author reviewed and edited the content as needed and takes full responsibility for the
597 content of the publication.

Author Contributions

S. Mohapatra conceived the idea, designed the study, conducted the experiments and wrote the manuscript.

Funding

This research received no specific funding.

Competing interests

The author declares that there is no conflict of interest regarding the publication of this article.

Data Availability

Code, the AWS template, the operations master data and the case-study input frames are available at <https://github.com/pikulsoomesh/foodbank-spatial-twin>. The source video is publicly available from its publisher [20]. The fused zone files (SPZ), the composed USD site and the per-zone placements are in the same repository.

Supplementary Materials

Movie S1. A 22s overview of the pipeline, 1080×1350 : the public news tour that inspired the twin, a world-model splat capture, the pipeline diagram, three NVIDIA Omniverse RTX shots of the modeled facility (Movie S3), the browser twin with simulated telemetry, and an end card. Omniverse shots are labeled as such.

Movie S2. A 24s 1080p flythrough of the OpenUSD stage, rendered with the CPU Gaussian-splat renderer.

Movie S3. Three 4s NVIDIA Omniverse RTX clips of the modeled facility (facility orbit, racking aisle, forklift chase), 1280×720 at 24 frames per second, rendered headless on an EC2 g5.xlarge; the forklift follows a scripted path.

Data file S1. Commands to reproduce the pipeline end to end, in three files in the repository’s `docs` folder: `CTFB_WAREHOUSE.RUN.md` (world-model zones), `PHOTOREAL.md` (locally trained zones) and `RENDERING.md` (CPU, Blender and Omniverse renders), with `scripts/build_facility.py` for the modeled facility.

Data file S2. The architecture diagrams as editable draw.io files (`paper/diagrams`).

A Equivalent designs on Microsoft Azure and Google Cloud

Nothing in the pipeline depends on AWS: the OpenUSD twin is files, and the API is a stateless function behind an HTTPS endpoint. Figures 12 and 13 redraw Figure 6 with the same layout on the other two large clouds, service for service (Table 8). Neither has been deployed or priced. Two differences matter for a food bank. Azure IoT Hub has no built-in LoRaWAN network server, so a

630 partner network server forwards the sensors' uplinks to it. Google Cloud IoT Core was retired in
631 August 2023, so on Google Cloud the sensors and gateways reach Pub/Sub through a partner or
632 self-hosted LoRaWAN network server and MQTT broker.

Table 8: The same twin on three clouds.

Role	AWS	Azure	Google Cloud
LoRaWAN network server	IoT Core for LoRaWAN	Partner network server	Partner network server
MQTT ingest	IoT Core	IoT Hub	Partner or self-hosted broker
Routing	IoT Core rules	Event Grid	Pub/Sub
Functions	Lambda	Functions	Cloud Run functions
Readings store	DynamoDB	Cosmos DB (serverless)	Firestore
Object storage	S3	Blob Storage	Cloud Storage
HTTP API	API Gateway HTTP API	API Management (Consumption)	Cloud Run
Content delivery	CloudFront	Front Door	Cloud CDN
Monitoring, alerts	CloudWatch, SNS	Azure Monitor, action groups	Cloud Logging, Cloud Monitoring
Access, budgets	IAM, Budgets	Entra ID and RBAC, Cost Management	Cloud IAM, Billing budgets
GPU for Omniverse Containers, pay per second	EC2 g6e.xlarge (L40S)	NVads A10 v5	Compute Engine G2 (L4)
Serverless GPU	ECS on Fargate (no GPU) none; EC2 on demand or Spot	Container Apps (scale to zero) Container Apps serverless GPUs	Cloud Run (scale to zero) Cloud Run GPUs (L4)

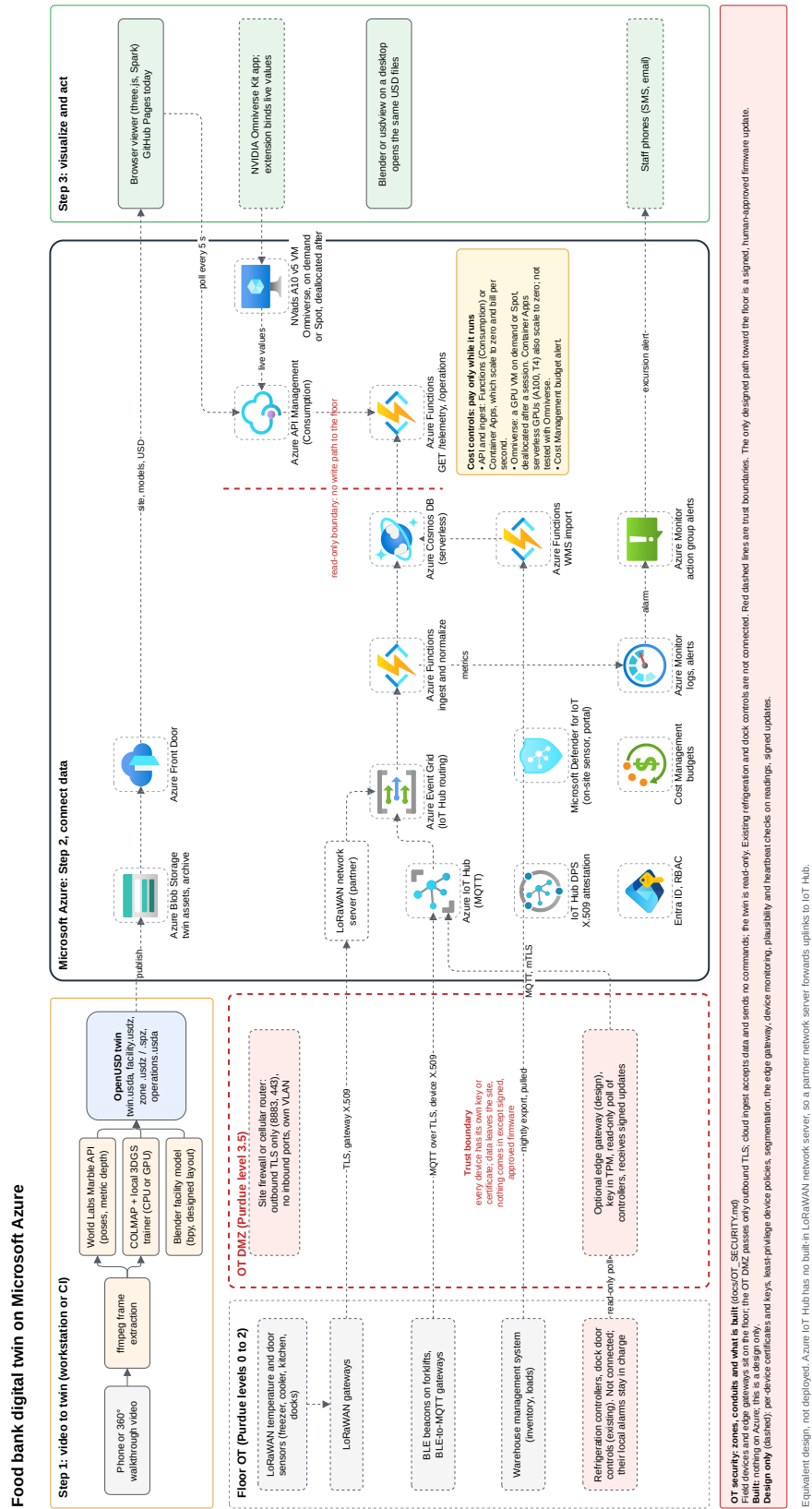


Figure 12: The food bank twin on Microsoft Azure: an equivalent design, not deployed, with the same OT zones and trust boundaries as Figure 6. Drawn in draw.io with the Azure icons.

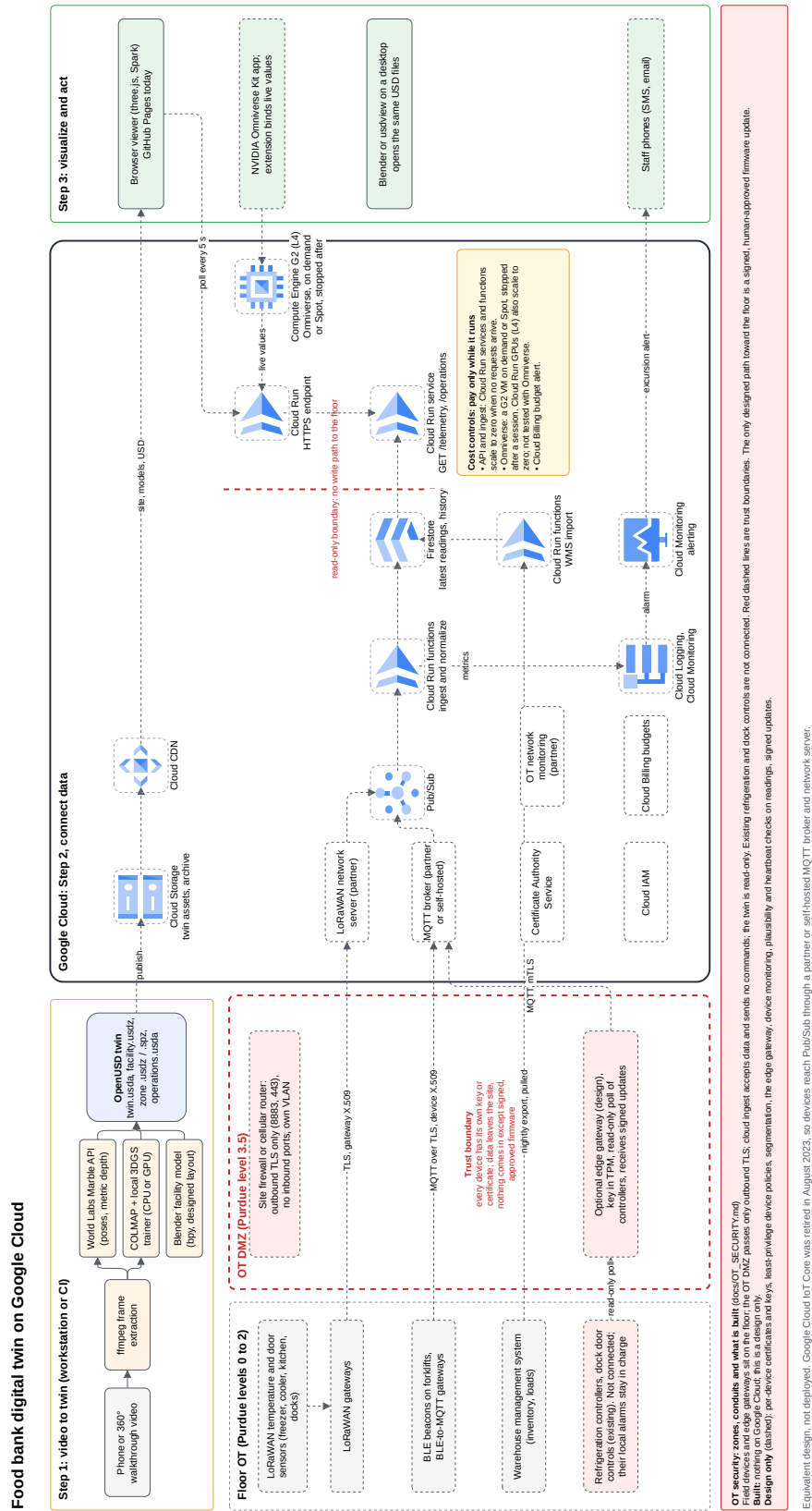


Figure 13: The food bank twin on Google Cloud: an equivalent design, not deployed, with the same OT zones and trust boundaries as Figure 6. Drawn in draw.io with the Google Cloud icons.

References

1. Grieves M and Vickers J. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In: *Transdisciplinary Perspectives on Complex Systems*. Springer, 2017:85–113. DOI: 10.1007/978-3-319-38756-7_4.
2. Kritzing W, Karner M, Traar G, Henjes J, and Sihn W. Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine* 2018;51:1016–22.
3. Tao F, Zhang H, Liu A, and Nee AYC. Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics* 2019;15:2405–15.
4. Fuller A, Fan Z, Day C, and Barlow C. Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access* 2020;8:108952–71.
5. Kerbl B, Kopanas G, Leimkühler T, and Drettakis G. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 2023;42:139:1–139:14.
6. Schönberger JL and Frahm JM. Structure-from-Motion Revisited. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016:4104–13. DOI: 10.1109/CVPR.2016.445.
7. Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R, and Ng R. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In: *Computer Vision – ECCV 2020*. Lecture Notes in Computer Science. Springer, 2020:405–21. DOI: 10.1007/978-3-030-58452-8_24.
8. Wang S, Leroy V, Cabon Y, Chidlovskii B, and Revaud J. DUST3R: Geometric 3D Vision Made Easy. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024:20697–709. DOI: 10.1109/CVPR52733.2024.01956.
9. Ha D and Schmidhuber J. Recurrent World Models Facilitate Policy Evolution. In: *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*. 2018. URL: <https://papers.nips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html>.
10. Brooks T, Peebles B, Holmes C, et al. Video generation models as world simulators. OpenAI. 2024. URL: <https://openai.com/index/video-generation-models-as-world-simulators/> (visited on 10/26/2026).
11. World Labs. World API quickstart. URL: <https://docs.worldlabs.ai/api> (visited on 10/26/2026).
12. Pixar Animation Studios. Introduction to USD. URL: <https://openusd.org/release/intro.html> (visited on 10/26/2026).
13. Alliance for OpenUSD. Alliance for OpenUSD. URL: <https://aousd.org/> (visited on 10/26/2026).
14. Alliance for OpenUSD. Announcing OpenUSD v26.03: Key Features and Improvements. 2026. URL: <https://aousd.org/blog/openusd-v26-03/> (visited on 10/26/2026).

- 669 15. OpenUSD. ParticleField3DGaussianSplat schema. URL: [https://openusd.org/dev/user_](https://openusd.org/dev/user_guides/schemas/usdVol/ParticleField3DGaussianSplat.html)
670 [guides/schemas/usdVol/ParticleField3DGaussianSplat.html](https://openusd.org/dev/user_guides/schemas/usdVol/ParticleField3DGaussianSplat.html) (visited on 10/26/2026).
- 671 16. NVIDIA. Develop Physical AI Applications: NVIDIA Omniverse. URL: [https://www.nvidia.](https://www.nvidia.com/en-us/omniverse/)
672 [com/en-us/omniverse/](https://www.nvidia.com/en-us/omniverse/) (visited on 10/26/2026).
- 673 17. NVIDIA. Omniverse Kit App Template. URL: [https://github.com/NVIDIA-Omniverse/kit-](https://github.com/NVIDIA-Omniverse/kit-app-template)
674 [app-template](https://github.com/NVIDIA-Omniverse/kit-app-template) (visited on 10/26/2026).
- 675 18. Central Texas Food Bank. Capital Area Food Bank of Texas Renames to Become Central Texas
676 Food Bank. Press release, 25 May 2016. 2016. URL: [https://www.centraltexasfoodbank.](https://www.centraltexasfoodbank.org/press/capital-area-food-bank-texas-renames-become-central-texas-food-bank)
677 [org/press/capital-area-food-bank-texas-renames-become-central-texas-food-bank](https://www.centraltexasfoodbank.org/press/capital-area-food-bank-texas-renames-become-central-texas-food-bank)
678 (visited on 10/19/2026).
- 679 19. Molineaux C. Ready to feed the need: inside Central Texas Food Bank's new Waco mega
680 facility. 25 News KXXV. URL: [https://www.kxxv.com/news/local-news/in-your-](https://www.kxxv.com/news/local-news/in-your-neighborhood/ready-to-feed-the-need-inside-central-texas-food-banks-new-waco-mega-facility)
681 [neighborhood/ready-to-feed-the-need-inside-central-texas-food-banks-new-waco-](https://www.kxxv.com/news/local-news/in-your-neighborhood/ready-to-feed-the-need-inside-central-texas-food-banks-new-waco-mega-facility)
682 [mega-facility](https://www.kxxv.com/news/local-news/in-your-neighborhood/ready-to-feed-the-need-inside-central-texas-food-banks-new-waco-mega-facility) (visited on 10/26/2026).
- 683 20. 25 News KXXV. Inside the NEW Central Texas Food Bank | See How Your Donation Changes
684 Lives. YouTube video. URL: <https://www.youtube.com/watch?v=8sQmr7JqXkI> (visited on
685 10/26/2026).
- 686 21. Schönberger JL. COLMAP: Structure-from-Motion and Multi-View Stereo. URL: [https://](https://colmap.github.io)
687 colmap.github.io (visited on 10/19/2026).
- 688 22. Central Texas Food Bank. 2023–2024 Impact Report. URL: [https://www.centraltexasfoodbank.](https://www.centraltexasfoodbank.org/media/985/download)
689 [org/media/985/download](https://www.centraltexasfoodbank.org/media/985/download) (visited on 10/19/2026).
- 690 23. Central Texas Food System. Food Processing and Distribution. URL: [https://www.](https://www.centraltxfoodsystem.org/food-processing-distribution)
691 [centraltxfoodsystem.org/food-processing-distribution](https://www.centraltxfoodsystem.org/food-processing-distribution) (visited on 10/19/2026).
- 692 24. Granato K. Kitchen Magic. Edible Austin. 2017. URL: [https://edibleaustin.com/food/](https://edibleaustin.com/food/embracing-local/kitchen-magic/)
693 [embracing-local/kitchen-magic/](https://edibleaustin.com/food/embracing-local/kitchen-magic/) (visited on 10/26/2026).
- 694 25. Central Texas Food Bank. Food Bank's After School Meals Help Central Texas Families. 2018.
695 URL: [https://www.centraltexasfoodbank.org/news/food-banks-after-school-meals-](https://www.centraltexasfoodbank.org/news/food-banks-after-school-meals-help-central-texas-families)
696 [help-central-texas-families](https://www.centraltexasfoodbank.org/news/food-banks-after-school-meals-help-central-texas-families) (visited on 10/19/2026).
- 697 26. Central Texas Food Bank. Cooking with Purpose: A Recipe for Serving Central Texas. 2025.
698 URL: [https://www.centraltexasfoodbank.org/news/cooking-purpose-recipe-serving-](https://www.centraltexasfoodbank.org/news/cooking-purpose-recipe-serving-central-texas)
699 [central-texas](https://www.centraltexasfoodbank.org/news/cooking-purpose-recipe-serving-central-texas) (visited on 10/19/2026).
- 700 27. Feeding America. How Feeding America turns \$1 into at least 10 meals. URL: [https://www.](https://www.feedingamerica.org/ways-to-give/faq/about-our-claims)
701 [feedingamerica.org/ways-to-give/faq/about-our-claims](https://www.feedingamerica.org/ways-to-give/faq/about-our-claims) (visited on 10/19/2026).
- 702 28. Blender Foundation. Blender Python API. URL: [https://docs.blender.org/api/current/](https://docs.blender.org/api/current/index.html)
703 [index.html](https://docs.blender.org/api/current/index.html) (visited on 10/26/2026).
- 704 29. World Labs. Spark: an advanced 3D Gaussian splatting renderer for three.js. URL: [https:](https://github.com/sparkjsdev/spark)
705 [//github.com/sparkjsdev/spark](https://github.com/sparkjsdev/spark) (visited on 10/26/2026).

- 706 30. Amazon Web Services. AWS IoT Core. URL: <https://aws.amazon.com/iot-core/> (visited
707 on 10/26/2026).
- 708 31. Stouffer K, Pease M, Tang C, et al. Guide to Operational Technology (OT) Security. NIST
709 Special Publication 800-82 Rev. 3. National Institute of Standards and Technology, 2023. DOI:
710 10.6028/NIST.SP.800-82r3.
- 711 32. ISA/IEC 62443 Series of Standards. International Society of Automation. URL: [https://www.
712 isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-
713 standards](https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards) (visited on 10/27/2026).
- 714 33. IEC 62443. International Electrotechnical Commission, Systems Committee Smart Energy.
715 URL: [https://syc-se.iec.ch/deliveries/cybersecurity-guidelines/security-
716 standards-and-best-practices/iec-62443/](https://syc-se.iec.ch/deliveries/cybersecurity-guidelines/security-standards-and-best-practices/iec-62443/) (visited on 10/27/2026).
- 717 34. Unauthorized Message, Technique T1692, ATT&CK for ICS. MITRE. URL: [https://attack.
718 mitre.org/techniques/T1692/](https://attack.mitre.org/techniques/T1692/) (visited on 10/27/2026).
- 719 35. Connect policy examples, AWS IoT Core Developer Guide. Amazon Web Services. URL: [https:
720 //docs.aws.amazon.com/iot/latest/developerguide/connect-policy.html](https://docs.aws.amazon.com/iot/latest/developerguide/connect-policy.html) (visited on
721 10/27/2026).
- 722 36. Revoke a client certificate, AWS IoT Core Developer Guide. Amazon Web Services. URL: [https:
723 //docs.aws.amazon.com/iot/latest/developerguide/revoke-ca-cert.html](https://docs.aws.amazon.com/iot/latest/developerguide/revoke-ca-cert.html) (visited on
724 10/27/2026).
- 725 37. Data and transport security with AWS IoT Core for LoRaWAN. Amazon Web Services. URL:
726 [https://docs.aws.amazon.com/iot/latest/developerguide/connect-iot-lorawan-
727 security.html](https://docs.aws.amazon.com/iot/latest/developerguide/connect-iot-lorawan-security.html) (visited on 10/27/2026).
- 728 38. World Labs. Multi-image prompt tips. URL: [https://docs.worldlabs.ai/marble/create/
729 prompt-guides/multi-image-prompt](https://docs.worldlabs.ai/marble/create/prompt-guides/multi-image-prompt) (visited on 10/26/2026).
- 730 39. Niantic. SPZ: a compressed file format for 3D Gaussian splats. URL: [https://github.com/
731 nianticlabs/spz](https://github.com/nianticlabs/spz) (visited on 10/26/2026).
- 732 40. World Labs. Rendering Marble SPZ files in third-party engines. URL: [https://docs.
733 worldlabs.ai/api/rendering-spz.md](https://docs.worldlabs.ai/api/rendering-spz.md) (visited on 10/26/2026).
- 734 41. NVIDIA. usd-convert-gsplat 0.1.15: NVIDIA 3D Gaussian Splat to USD Converter. URL:
735 <https://pypi.org/project/usd-convert-gsplat/> (visited on 10/26/2026).
- 736 42. OpenUSD. ParticleFieldPositionAttributeAPI schema. URL: [https://openusd.org/dev/
737 user_guides/schemas/usdVol/ParticleFieldPositionAttributeAPI.html](https://openusd.org/dev/user_guides/schemas/usdVol/ParticleFieldPositionAttributeAPI.html) (visited on
738 10/26/2026).
- 739 43. Amazon Web Services. AWS Serverless Application Model (SAM). URL: [https://aws.amazon.
740 com/serverless/sam/](https://aws.amazon.com/serverless/sam/) (visited on 10/26/2026).