



WorldRover: A Scalable Synthetic Video Data Engine for World Exploration with Rich Annotations

Xiaojie Xu^{1,2*}, Zhengyuan Lin^{1,2*}, Runyi Li^{1,2}, Yihao Liu¹, Kaipeng Zhang^{1†}, Yongtao Ge^{1†}

¹Alaya Lab ²The University of Tokyo

Learning to generate or reconstruct explorable worlds requires video paired with more than RGB: camera motion, scene geometry, temporal correspondence and, for interactive models, control signals. Real capture can provide some of these signals, but dense geometry and long-range correspondence usually rely on estimation or specialised instrumentation. Rendering provides these quantities directly, yet existing synthetic resources rarely combine them on the same frames while also supporting controlled changes of viewpoint and appearance. We introduce WorldRover, a data engine for generating richly annotated, long-range explorations of artist-built environments. At its core, WorldRover-Engine is an Unreal Engine pipeline that executes and offline-renders minute-scale routes while preserving their full trajectories and scene geometry. The same exploration can be replayed from first-person, third-person, and 360° panoramic cameras under different environmental states. Using WorldRover-Engine, we construct WorldRover-10M, whose sequences pair RGB with metric depth, camera trajectories, and trajectory-derived action signals throughout each exploration. Third-person subsets additionally provide dense optical flow, long-range 2D/3D point tracks with visibility, and a character trajectory distinct from the camera trajectory. The engine can render a traversal from first-person, third-person and 360° panoramic viewpoints, under different environmental states or with a neutral white material, while preserving the route and scene geometry. WorldRover therefore turns long-horizon world exploration into a scalable data-generation problem, providing supervision for models that must build, maintain, and revisit coherent representations of an explorable world.

Project Page: <https://alayaab.github.io/WorldRover>

Code: <https://github.com/AlayaLab/WorldRover>

Data: <https://huggingface.co/datasets/xjxu21/WorldRover>

Date: August 15, 2026



1 Introduction

An explorable visual world must remain coherent as the observer moves through it. As the viewpoint changes or an action moves an agent, a model should preserve scene layout, maintain object identity through occlusion and distinguish camera motion from changes in the world. These requirements appear in interactive world generation [1, 2, 8, 11, 22, 35] and in dynamic 4D reconstruction [15]. Although the two settings have different outputs, they share a data problem: RGB alone leaves the causes of visual change entangled. Camera motion and object motion both displace pixels, illumination can change without geometry changing, and an occluded surface has no visible correspondence target. Camera trajectories, metric geometry, temporal correspondence and, when available, action signals make these factors explicit.

Data with this combination is difficult to collect. Large real-video corpora provide broad visual coverage [20], but camera pose, depth and correspondence are generally estimated after capture; pose can drift, correspondence can fail through occlusion, and instrumented capture restricts the environments and conditions that can be recorded. Rendering exposes these quantities as part of scene state and image formation, but existing synthetic resources are optimized for different goals. Task-specific datasets provide dense labels for a fixed benchmark [4, 29, 33, 39]; interactive simulators prioritize low-latency observations

^{1*}Equal contribution

^{2†}Corresponding authors: kaipeng.zhang@shanda.com, yongtao.ge@shanda.com

WorldRover-10M Dataset Overview



1 DIVERSE VIEWPOINTS



2 DIVERSE SCENE STYLES



3 DIVERSE MOTION LABELS

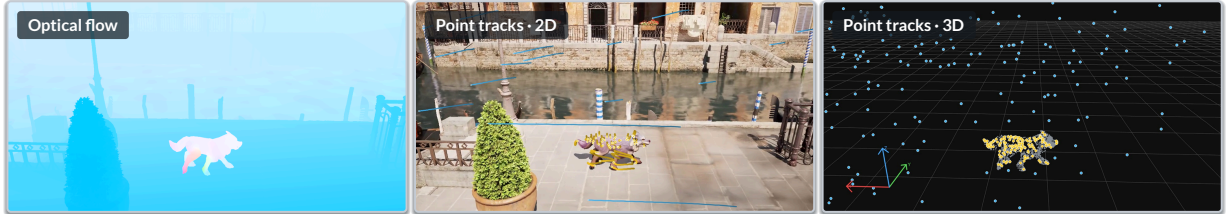


Figure 1 WorldRover-10M at a glance. Four rows, top to bottom. *Scene and routes*: one UE environment with three routes drawn as camera frustums, and four frames along each, colour-keyed to the overview. (1) *Diverse viewpoints*: the same instants observed as first-person, third-person and 360° panoramic video; every panel is RGB with metric depth inset, released with the camera that produced it. Third-person adds the character’s own trajectory and the trajectory-derived action stream. (2) *Diverse scene styles*: the first-person view re-rendered under a different weather state, a different lighting state, and as an untextured white model, with scene, motion and camera held fixed. (3) *Diverse motion labels*: dense optical flow and long-range point tracks, shown in 2D and in world space; a track stores a 3D position, its 2D projection and visibility per frame. Third-person only, with independent body and camera motion.

for a policy [6, 31]; and procedural generators increase the number of scenes [12, 26]. Across these forms, the same world traversal is seldom re-observed under controlled changes of viewpoint and appearance. The missing capability is therefore not merely more rendered frames, but a production system that scales variation while preserving the factors shared across related observations.

We introduce WorldRover, a synthetic video data engine built around one abstraction: an exploration is a reusable world-space trajectory, not a single finalized video. WorldRover-Engine prepares an artist-built UE environment once, generates minute-scale routes through it and renders those routes offline. The same exploration can then be observed from first-person, third-person and 360° panoramic cameras, or re-rendered under another environmental state or as a neutral white model. Because route, timing and scene structure are retained across these variants, the resulting pairs isolate viewpoint or appearance changes instead of confounding them with a different motion. Camera records and renderer-derived annotations remain tied to the frames they describe.

Using WorldRover-Engine, we construct WorldRover-10M: 6,003 sequences from 32 environments, comprising 21.9M rendered frames, 202.7 hours of video and 18.7TB of released data. The name refers to its 10.8M first-person frames; third-person and panoramic subsets provide additional observations of the exploration corpus. Annotation bundles are task-specific rather than universal. Depending on the subset, a sequence pairs RGB with metric depth, dense optical flow, long-range 2D/3D point tracks, rendered camera and character trajectories, and trajectory-derived actions. This organization supports both broad training data and controlled comparisons in which geometry, motion, viewpoint or appearance is held fixed.

Our contributions are:

- **A scalable data engine for world exploration.** WorldRover-Engine combines one-time scene preparation, automated and recorded route generation, offline rendering, streaming annotation extraction and parallel campaign execution over artist-built UE environments.
- **Controlled re-observation of a shared trajectory.** WorldRover renders matched first-person, third-person and panoramic observations, as well as environmental and white-model variants, while retaining the underlying route, timing and scene structure.
- **A large, richly annotated release.** WorldRover-10M provides 21.9M frames across long explorations, with subset-specific geometry, motion, correspondence, camera, character and action annotations under a common release schema.

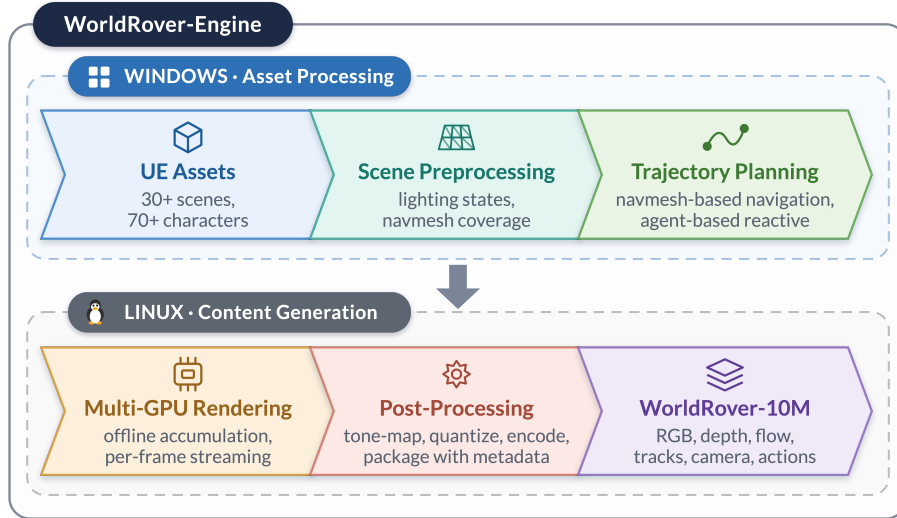


Figure 2 WorldRover-Engine overview. Asset processing runs once per scene on a Windows editor host; content generation runs once per sequence, headless, on a multi-GPU Linux host. The two halves meet at a versioned on-disk hand-off, so a campaign is replayable without an editor session.

2 WorldRover-Engine

2.1 Overview

A WorldRover sequence follows a route through a prepared environment. Sparse route samples are converted to human-scale camera motion with approximately constant travel speed, smoothed and rate-limited heading, and occasional *dwells* in which translation stops while the view rotates. The resulting trajectory is then rendered from the viewpoint specified by the target subset.

2.2 Editor side

Asset ingest. Each scene enters as an Unreal asset pack and is prepared once on the editor host into a uniform, headless-loadable form, allowing render workers to open it without an editor session. Packs target a range of engine versions, so ingest first resaves each scene for UE 5.5. Packs may also split content across streaming sub-levels or World Partition cells. Ingest loads these components before navigation baking and stylization, and the render configuration keeps the required cells resident during headless generation. A navigation mesh is then baked over the walkable surface and stored with the scene for the planner.

Scene stylization. Each appearance state of [Section 5.2](#) is authored in the editor so that a headless render reproduces it with no runtime setup. Its visual configuration is prepared there, while exposure is calibrated through the offline rendering path.

- **Environmental states.** A state is authored entirely on the scene side and saved as its own map. It replaces the scene’s shipped lighting with a fixed rig in the persistent level — a directional sun, a sky light for ambient fill, a sky-atmosphere and exponential height fog whose density and in-scattering colour define the atmospheric state — and may add rain, snow or drifting mist through persistent Niagara systems that cover the walkable area. The state does not depend on trigger volumes or level-blueprint events, so it is active when the map loads in a headless session. A campaign selects the corresponding map variant rather than configuring these effects per shot.
- **Fixed exposure.** Each environmental state uses its own calibrated fixed exposure, and auto-exposure remains disabled during rendering. A night state can therefore remain dark rather than adapting toward mid-grey. The value for a state is calibrated once: a probe frame is rendered through the same offline path used by a campaign, its mean luminance is measured, and the exposure in the map’s post-process volume is adjusted until the frame reaches the intended target. All sequences rendered from that state then use the calibrated value without per-shot adjustment.

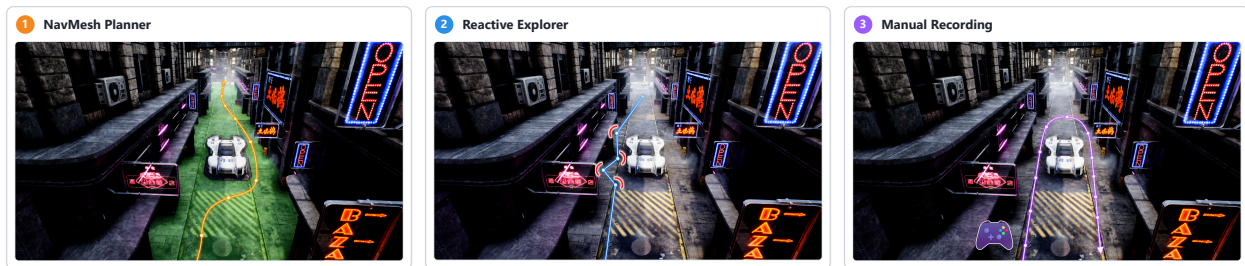


Figure 3 Trajectory generation. The three strategies for producing walking trajectories, on one scene. **(1) NavMesh planner:** a route (orange) planned over a baked navigation mesh (green); the mesh is built once and reused, so planning is fast and coverage broad, at the cost of per-scene tuning. **(2) Reactive explorer:** a path (blue) driven by local obstacle sensing (red); it needs no precomputed map, but senses step by step and can walk itself into a corner. **(3) Manual recording:** a human-driven path (purple), which yields a specific trajectory on demand at the cost of manual effort. Together they trade off automation, coverage and control.

Trajectory generation. We produce walking trajectories with a navigation-mesh (NavMesh) planner, a reactive explorer and manual recording ([Figure 3](#)). The three methods trade off automated coverage, dependence on scene preparation and direct control over the route.

- **NavMesh planner.** An agent starts at a random reachable point and walks toward sampled goals on the baked navigation mesh, with occasional changes of heading. The resulting routes remain on the walkable surface and tend to follow efficient connections through the scene.
- **Reactive explorer.** A physics-driven agent uses a fan of short ray casts to avoid nearby obstacles and favours directions it has not recently visited. It requires no precomputed navigation mesh, but its local policy may wander, backtrack or enter a dead end.
- **Manual recording.** An operator walks through the scene while the camera path is recorded. This provides routes and viewpoints selected deliberately rather than by an automatic exploration policy.

2.3 Server side

Offline rendering and dynamic illumination. Frames are produced through UE’s Movie Render Queue, which accumulates spatial sub-samples for each output frame. The pipeline targets batch production rather than interactive latency, allowing more rendering work per frame; in our configuration, accumulation reduces aliasing and temporal variation in Lumen illumination relative to the single-sample interactive path. Lighting is evaluated at render time with Lumen rather than read from a baked lightmap. Re-rendering an authored environmental state therefore updates direct and indirect illumination, including colour bleeding and shadow softness, rather than applying an image-space recolouring.

Sequence construction and streaming post-processing. SPEAR[30] provides the control plane used to load and configure the UE project, create the required actors and submit render jobs. Sparse waypoints are first converted to dense per-frame keyframes by constant-speed reparameterization, position and heading smoothing, and yaw-rate limiting. The keyframes are written to a Level Sequence consumed by Movie Render Queue; for panoramic runs, the six cube-face orientations are assigned at this stage. Movie Render Queue writes multi-layer EXR frames, while host-side workers perform tone mapping, depth extraction and, for panoramic sequences, cube-face stitching. Because these operations depend on one rendered instant at a time, each EXR is processed and deleted after it appears rather than retained until the sequence finishes, keeping intermediate storage bounded. The worker pool is sized so that its sustained consumption rate remains above the renderers’ production rate.

Parallel scheduling and recovery. Throughput was limited mainly by UE game-thread work and host-side post-processing; one render instance used only a fraction of a GPU. Multiple instances therefore share each GPU subject to video-memory limits, and RGB is encoded on the dedicated hardware encoder so CPU cores remain available for EXR processing. Render instances are reused across several sequences to amortize startup, shader compilation and lighting warm-up. A watchdog terminates instances that do not exit cleanly after a batch.

3 WorldRover-10M: Three Camera Viewpoints

The pipeline defines one first-person camera trajectory per route (Section 2), and the other viewpoints are derived from that trajectory. In third-person, an animated character walks the recorded path and a follow camera holds a fixed offset behind and above it, so the character trajectory of a third-person sequence is the camera trajectory of the corresponding first-person sequence. In panoramic mode the camera follows the recorded path and only frame assembly differs: six differently oriented cameras are rendered at each instant and stitched. This construction provides three observations of a route with matched timing, scene geometry and environmental state.

3.1 First-person

An eye-height camera rides the trajectory itself, with pitch and roll locked and heading following the direction of travel, smoothed and rate-limited so the motion reads as human locomotion rather than as a planner’s output. Here the camera *is* the agent, so the camera trajectory is also the agent trajectory, and the

trajectory-derived action stream, pose and imagery share the same motion. This is the mode that matches egocentric video and embodied navigation.

3.2 Third-person

An animated character replays the recorded trajectory in world coordinates, and a follow camera maintains a nominal offset behind and above it. Camera azimuth is driven by a slowly varying orbit target rather than copied from the character heading, so the view remains approximately stable through ordinary turns while still following long changes in travel direction. Character motion can therefore appear toward, away from or across the camera, and the trajectory-derived action stream is expressed in this camera-relative frame (Figure 4). The character turns toward its direction of travel, with a rate limit, while explicit camera-yaw controls rotate the view around a stationary character. The camera and character trajectories are distinct, although the camera trajectory is derived from the character path. The same nominal camera rig is used across characters.

These sequences contain an articulated subject observed from a moving camera, with separate recorded trajectories for the subject and camera. This separation supports evaluation of methods that must distinguish camera motion from non-rigid subject motion.

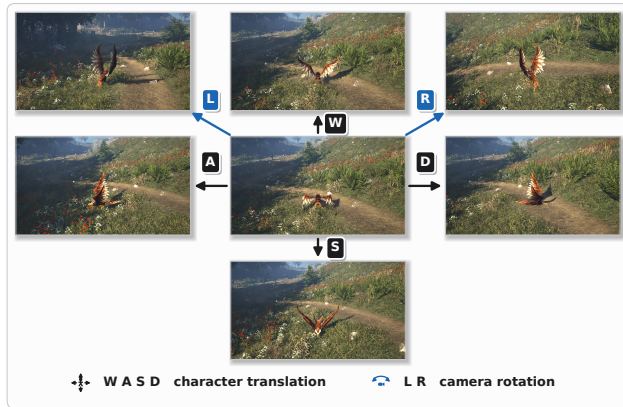


Figure 4 Third-person control semantics. Centre: the start state. **W/A/S/D** (black) translate the character along camera-relative directions — W away from the camera, S toward it, A and D to its left and right — and the character turns to face its heading while the camera follows with a slowly varying orbit azimuth. **L/R** (blue) yaw the camera about a stationary character, so the character holds its image position while the scene rotates around it.

3.3 360° panoramic

Equirectangular video at 4096×2048 is assembled from six cube faces rendered at the same instant (Figure 5), providing an omnidirectional observation around the camera. We do not use UE’s own panoramic pass, which hard-clamps scene colour during accumulation: in a tone-mapping-off render the brightest tenth of the image collapses onto a single value, and those highlights cannot be recovered afterwards. We instead render six ordinary perspective faces, which retain the measured HDR range, and assemble the panorama on the host. The faces are reprojected and cosine-feathered in linear light. Exposure is estimated from a cosine-latitude-weighted log-luminance statistic and smoothed over time for video; a large-support bilateral base/detail operator then compresses the dynamic range before sRGB conversion. Panoramic and first-person modes share camera centres and orientations, allowing a perspective view to be compared with a perspective reprojection of the panorama.

4 WorldRover-10M: Multimodal Data and Annotations

A sequence is delivered as a directory with a manifest that lists the available outputs. Annotation bundles differ by subset, but use common coordinate, timing and metadata conventions.

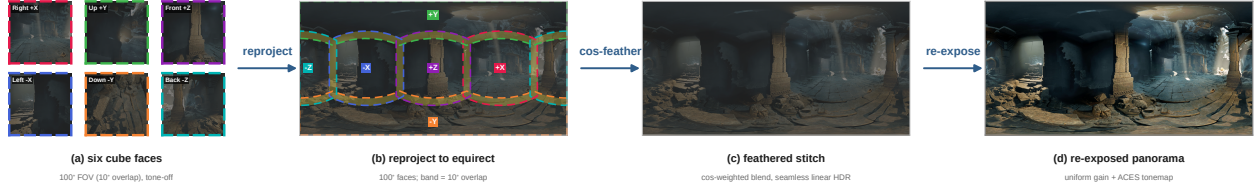


Figure 5 Panoramic assembly. (a) Six cube faces at 100° FOV, giving a 10° overlap between neighbours, rendered with tone mapping disabled so all faces share one radiometric scale. (b) Reprojection onto an equirectangular image; dashed outlines, colour-matched to (a), mark each face’s region and the yellow bands its overlap, which exposes the pincushion distortion of the side faces and the polar caps. (c) A cos-weighted feather blend merges the overlaps into a linear-HDR panorama. (d) Exposure is estimated with latitude-aware luminance weighting and smoothed across frames; local base/detail tone mapping compresses the result before sRGB conversion.

One EXR frame, three registered rasters. Movie Render Queue renders each output frame through a single configuration that writes colour, a depth layer and a velocity layer into one multi-layer EXR, accumulating two spatial samples per frame for anti-aliasing. Colour, depth and screen-space motion are therefore the same rasterization of the same instant, at the same resolution and projection; the camera annotation is the camera that produced that frame. Reading those layers is a purely intra-frame operation — each is extracted and quantized without reference to neighbouring frames. This permits the streaming process described in [Section 2](#).

4.1 RGB video

Photorealistic colour video under dynamic global illumination, rendered offline with sub-sample accumulation to reduce aliasing and temporal variation in indirect light, distant lights and specular highlights. Frames are tone-mapped from the linear EXR and encoded to H.264 on the GPU’s dedicated encoder block, keeping CPU cores available for per-frame extraction and panorama stitching. First- and third-person videos are 1280 × 720; panoramic video is 4096 × 2048 equirectangular, assembled from six 1536² cube faces.

4.2 Metric depth

Depth is read from the renderer’s depth layer, which stores radial distance from the camera centre in metres. Unlike an estimated or sensed depth map, it does not depend on image texture or sensor noise; the released representation nevertheless has a fixed valid range. Values are clipped to 0.1–200 m and quantized logarithmically into 16 bits,

$$t = \frac{\log d - \log d_{\text{near}}}{\log d_{\text{far}} - \log d_{\text{near}}}, \quad d_{\text{near}} = 0.1 \text{ m}, \quad d_{\text{far}} = 200 \text{ m},$$

and the frames are encoded losslessly with FFV1. A sidecar carries the inverse formula and encoding parameters. The logarithmic parameterization gives finer absolute precision at short range and coarser precision at longer distances.

4.3 Optical flow

Third-person subsets provide dense optical flow derived from the renderer rather than estimated from the images. A post-process material bound to an extra render pass writes the engine’s own velocity buffer into a named sublayer of the same multi-layer EXR that carries colour and depth, as a float16 buffer whose first two channels hold the screen-space displacement $(\Delta u, \Delta v)$ of each pixel between consecutive frames, normalized so that the full image width spans 2. Pixel displacements follow from the image size,

$$\Delta u_{\text{px}} = \Delta u \cdot \frac{W}{2}, \quad \Delta v_{\text{px}} = \Delta v \cdot \frac{H}{2},$$

with $+v$ pointing down the image. Because temporal anti-aliasing contaminates the velocity buffer, it is disabled in this render configuration and anti-aliasing is spatial instead — the accumulated sub-samples of [Section 2](#) — so the released flow is the raw per-pixel motion of the frame it accompanies. Its value does not

depend on texture or image matching, but it follows the engine’s velocity-buffer semantics and the storage range described below.

The field is released as one lossless FFV1 stream per sequence, rather than one file per frame. Each frame is mapped affinely onto two 16-bit channels with a single scale g , in pixels per frame, that is fixed for the whole release (64 by default and configurable per campaign),

$$u_{16} = \text{clip} \left[\frac{65535}{2} (\Delta u_{\text{px}} / g + 1) \right],$$

$$\Delta u_{\text{px}} = (2 u_{16} / 65535 - 1) g,$$

and likewise for v ; displacements beyond $\pm g$ are clipped. Thus the released stream preserves the rendered motion field within ± 64 pixels per frame under the default configuration, while larger displacements saturate unless a campaign uses a larger scale. A sidecar records g , the frame count, resolution, frame rate and the decode formula, so the stream decodes from the released files alone. Lossless video encoding exploits both the spatial smoothness and temporal redundancy of the field, avoiding the overhead of one file per frame. The payload is 16-bit and must be read with a decoder that preserves that depth.

4.4 Long-range point tracks

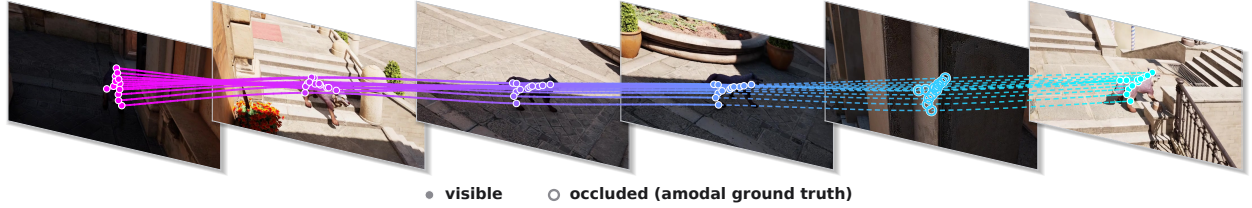


Figure 6 Point-track ground truth. Six frames 4s apart, time encoded magenta $\rightarrow$ cyan; each curve follows one 3D point on the character. Filled markers and solid curves are visible, hollow rings and dashed curves occluded and drawn at their amodal ground-truth position. In the fifth panel a wall encloses the camera and hides the character completely, yet the tracks retain their amodal positions with zero visibility flags.

Third-person sequences carry long-range point tracks that are geometric rather than matched. On a seed frame a batch of query pixels is unprojected into world coordinates using the rendered depth and the camera of that frame; on later frames those world points are projected back into the image, and the resulting pixel sequence is the track. New cohorts are seeded periodically, so surfaces revealed later still receive query points, and character and background pixels are drawn separately. Because propagation uses projection rather than image matching, it does not accumulate matching drift; accuracy instead depends on the rendered depth, camera pose and the binding of a point to the character.

Background points keep the world coordinate obtained at their seed frame. The default character tracker stores the seed point in the character’s local frame and transforms it by the per-frame root position and yaw. This rigid binding follows the character trajectory but not articulated deformation, so points on a foot, tail or head can move away from the rendered surface during an animation cycle. We also implement an LBS variant for sequences whose skinned mesh can be reconstructed from additional glTF skinning weights and per-frame bones. It binds each query to the reconstructed skinned surface and therefore follows its deformation; those mesh and skeleton inputs are not part of the sequence archive.

Occlusion does not interrupt a track. Position is written unconditionally, and visibility is a separate per-frame flag, set by comparing a track’s projected depth against the rendered depth at that pixel. The 2D trajectory is therefore *amodal*: while a point is hidden, the release still states where in the image it would be (Figure 6). Visibility and having a defined position are separate, as in PointOdyssey’s occlusion labels [37]; conflating them removes supervision during occlusion.

The rigid archive stores the seed, per-frame world and image positions, visibility, object ID and initial binding residual. The LBS archive omits world positions and binding residuals because its character

positions come from the skinned mesh. Queries can be seeded only on visible surfaces, biasing follow-camera samples toward the body’s back; background tracks also leave the frame as the camera moves, so the longest tracks are mainly on characters.

4.5 Camera, character and action streams

The remaining streams are not rasters but per-frame records derived from the trajectories the sequence was rendered along.

The **camera trajectory** gives the position, orientation and intrinsics of the camera at every frame. It is written from a sidecar emitted at render time, so it describes the camera that was actually rendered rather than the keyframes sent in. Panoramic sequences declare an equirectangular projection and keep the pose of the shared camera centre, instead of reporting a perspective intrinsic matrix that does not describe the image.

The **character trajectory** gives the position, orientation and speed of the character at every frame, and third-person sequences release it alongside the camera because the two are genuinely different motions (Section 3): they differ in path, and far more in heading, since the character turns while the follow camera holds its azimuth.

The **trajectory-derived action stream** is a control representation reconstructed from the motion rather than a log of original input events. Velocities are differenced between adjacent trajectory frames, resolved into forward, lateral, turning and vertical axes in the corresponding first-person camera frame, quantized, and written when an axis value changes. Third-person sequences retain this camera-relative stream rather than recomputing actions from the follow camera.

4.6 Release details

Table 1 Released data format. How each item is stored when a subset includes it.

Data, when present	Contents
RGB video	H.264, hardware-encoded
Depth	16-bit log-quantized, FFV1 lossless, + decode sidecar
Optical flow	16-bit affine-quantized, FFV1 lossless, + decode sidecar (third-person only)
Point tracks	per-track 3D position, 2D projection and visibility, per frame (third-person only)
Camera trajectory	per-frame pose and intrinsics of the rendered camera
Character trajectory	per-frame pose and speed (third-person only)
Trajectory-derived actions	quantized axis events + binding snapshot
Description	environment, camera, map variant and render metadata
Route preview	plan view of the trajectory

Each sequence directory holds the encoded video, the depth stream with its decode sidecar, the trajectory tables, the trajectory-derived action stream and a description manifest, plus a rendered plan view of the route for inspection. Table 1 lists how each item is stored when a subset includes it. The description manifest records the environment, the camera and sensor geometry, the projection model, the selected map variant, the trajectory source, sequence timing and the render configuration. The current schema does not expose normalized lighting and weather fields; appearance variants are identified through their map configuration. Two screens run after rendering. The first is format: the encoded video and each annotation stream must be present, decodable and of agreeing length, and the depth sidecar must carry the parameters its decoding needs. The second is quality, where files can be structurally valid while their content is unsuitable — miscalibrated exposure or too dark, a camera that has passed through geometry, a route that keeps revisiting the same part of a scene, a render that has come out degraded. Sequences that fail either screen are dropped rather than released.

5 WorldRover-10M: Diversity and Scale

Artist-built environments provide authored geometry, materials and scene layout, but are less numerous than procedurally generated scenes. WorldRover-Engine expands a fixed asset library by composing scenes, environmental states, viewpoints, characters and routes. Preparing an environment is a one-time authoring step; subsequent sequences are generated from campaign configurations and rendering compute rather than per-frame manual annotation.

5.1 Scenes

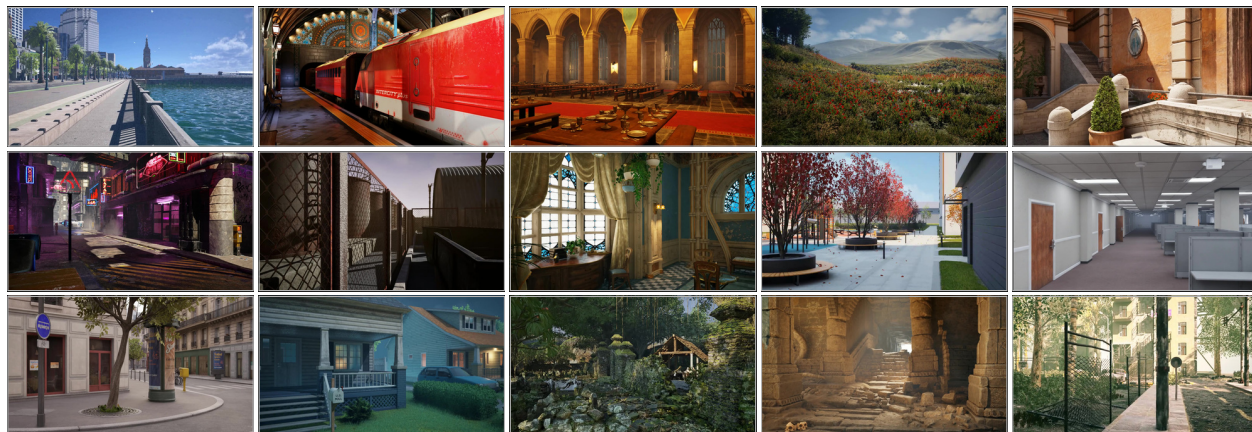


Figure 7 A sample of the WorldRover scene library. Fifteen of 30+ UE environments — waterfront city, transit interior, cathedral, alpine meadow, cyberpunk street, office interior, Paris street, suburban home, jungle ruin, and more — all rendered under dynamic Lumen global illumination.

The WorldRover-Engine scene library comprises 30+ UE environments spanning city, historical, horror, industrial, interior, fantasy, cyberpunk, mountain, street and town/village categories, covering everything from a single apartment to a full urban district (Figure 7). They are commercially licensed, artist-built environments intended for games and previsualization, with authored materials, set dressing and scene clutter.

Interior and city-scale scenes differ substantially in depth range, the density of independently moving content and the amount of occlusion. These factors affect the difficulty of depth estimation, optical flow and point tracking.

5.2 Appearances



Figure 8 Diverse appearances. The same first-person frame of *med_village*, rendered from an identical camera at the same instant under five environmental states. Because lighting is *simulated at render time* rather than baked, indirect illumination, colour bleeding and shadow softness respond correctly to each state; each state uses its own calibrated fixed exposure, with no automatic adjustment during rendering, so **Night** stays dark rather than normalised toward mid-grey. Re-rendering an instant under another state therefore changes the image physically, not by recolouring a fixed frame.

WorldRover-Engine supports two appearance controls. **Lighting** sets the time of day — day, dawn, evening, sunset and night — and **weather** sets the atmosphere — clear, overcast, fog and snow. Because lighting is

simulated rather than baked, indirect illumination, colour bleeding and shadow softness change with the authored state instead of remaining fixed in a lightmap. Each state uses its own calibrated fixed exposure, with auto-exposure disabled during rendering.

When a route is rendered under more than one state, geometry and motion remain fixed while illumination changes. Such pairs can be used to separate appearance sensitivity from changes in scene structure or trajectory.

Each state is authored as a self-contained level variant whose lighting rig is baked into the persistent level with no gameplay-gated triggers. Its fixed exposure is calibrated through the offline rendering path and is not adjusted automatically during rendering ([Section 2](#)).



Figure 9 White model. Three environments, the ordinary render above and the white-model re-render of the same frame below. With scene surfaces drawn in the engine’s default material, geometry and occlusion remain visible from large structures to fine detail — arcade columns and planters, station ironwork, individual leaves and branches — while texture, colour, decals and emissive light do not: the tiled floor flattens, the screens on the ticket machines go blank, and the burning wreck in the clearing is left as geometry alone.

White model. WorldRover-Engine can render a sequence a second time as a *white model*, with geometry and camera motion preserved and surface appearance replaced by a neutral material and its shading. This is the coarse input a coarse-to-real generator is trained against, where the structure is given and the appearance is what the model has to supply. The whitening happens in the renderer rather than through edits to individual scene components. The engine’s default-material show flag covers buildings, terrain and its procedural grass, foliage and instanced meshes, HLOD proxies, spline geometry such as rails, and buildings streamed in at runtime. Replacing materials component by component in an earlier implementation reached mesh components but missed landscape grass, level instances and HLOD buildings. The white-model render uses the same level sequence and camera configuration as its textured counterpart ([Figure 9](#)). Because the show flag replaces surface albedo but not illumination, the result is desaturated after rendering to obtain a neutral grey image while retaining shading. The sequence manifest states whether a white-model render is included.

5.3 Characters



(a) Humanoids

(b) Animals and creatures

Figure 10 Character library. Asset-review sheets — the avatars shown in one shared scene at a fixed pose — rather than frames from the released dataset. Third-person sequences are rendered with humanoids, quadrupeds and birds, which differ in body plan, gait and how they turn, so subject articulation is a diversity axis in its own right.

The WorldRover-Engine asset library contains 70+ animated characters, including humanoids alongside animals and creatures (Figure 10). Body plans differ: bipeds pivot in place, quadrupeds and birds turn along arcs, and each is authored at its native scale and ground contact. For a dataset aimed at 4D reconstruction of deforming subjects, this variety in articulation and gait is as important as scene variety.

5.4 Scalability and statistics

Table 2 WorldRover-Engine capacity and the current WorldRover-10M release. The upper block describes the engine and asset library, not the coverage of every released subset. In the lower block, scenes are counted per viewpoint and therefore do not add; 32 is the number of distinct environments.

Engine/library capacity					
Environments	30+ artist-built UE scenes, interior to city scale				
Lighting states	day, dawn, evening, sunset, night				
Weather states	clear, overcast, fog, snow				
Materials	textured render or untextured white model				
Characters	70+ animated humanoids, animals and creatures				
Routes	navmesh planner, reactive explorer, manual recording				
Scales by	scene × state × character × route × viewpoint				
Current release					
	Scenes	Sequences	Frames	Duration	Volume
First-person	23	2,910	10.8 M	100.5 h	5.9 TB
Third-person	7	1,885	8.3 M	76.5 h	4.4 TB
360° panoramic	15	1,208	2.8 M	25.8 h	8.4 TB
Total	32	6,003	21.9 M	202.7 h	18.7 TB

The current release contains 6,003 sequences from 32 environments, totalling 21.9M rendered frames, 202.7 hours of video and 18.65 TB of data. Table 2 reports the breakdown by viewpoint. The name WorldRover-10M refers to the 10.8M first-person frames; third-person and panoramic sequences add further observations of routes defined in the first-person trajectory set.

All three trajectory strategies are represented: 3,662 sequences from the navmesh planner, 1,996 from the reactive explorer and 345 from waypoint trajectories. Median durations are 115 s for planner routes, 105 s for waypoint trajectories and 81 s for reactive ones — while their speed is drawn from three settings, 1.0, 1.2 and 1.5 m/s, so its distribution is three clusters and not one (Figure 11). By scene category the largest share

is urban, at 2,128 sequences, followed by town and period settings at 1,668, interiors at 639, historical and fantasy at 633, and cyberpunk at 603.

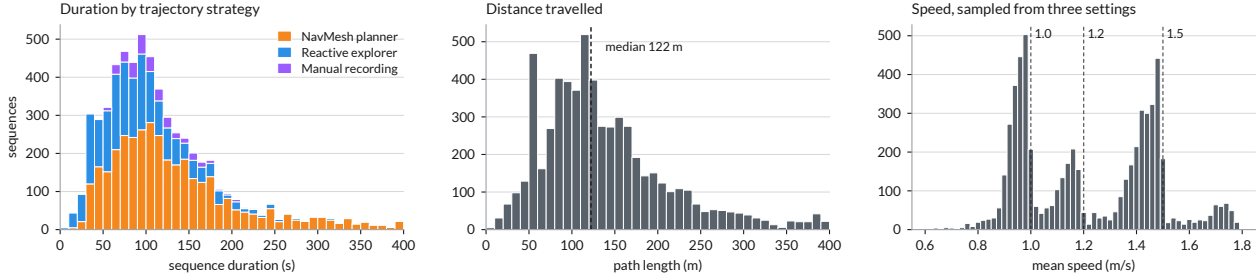


Figure 11 Distributions over the released sequences ($n = 6003$, from per-sequence metadata). Left: sequence duration, stacked by trajectory strategy in the colours of Figure 3 — reactive routes end sooner than planned ones, which run on to the long tail. Centre: distance travelled, a median of 122 m. Right: mean speed per sequence, which falls into three clusters because the motion model samples one of three walking speeds; each cluster sits a little below its setting, the cost of turning and rate limiting.

6 Related Work

6.1 Simulators and engine interfaces

A simulator advances an agent through a world and returns observations; an engine interface exposes the controls and state needed to do so. Photorealistic simulators are built on Unreal, on Unity [19] and on platforms such as Isaac Sim [23]; a parallel line is organised around physical interaction, hosting articulated objects for manipulation [34]. On UE, UnrealCV [25] opened the engine to computer vision and UnrealZoo [38] extends it with over a hundred worlds; AirSim [31] and CARLA [6] built domain-specific simulators for aerial and driving research; the Habitat line [24] did the same for indoor embodied agents; and AirSim360 [9] extends the idea to omnidirectional drone view, with equirectangular RGB, slant-range depth and automated trajectories. Many of these systems expose a curated Python API and are distributed with a particular simulator project, which can limit access to engine functionality or complicate the use of third-party UE content. SPEAR [30] instead uses UE runtime reflection and attaches to an existing project as plugins. WorldRover-Engine adopts that interface and extends it into an offline production pipeline with asset preparation, trajectory construction, Movie Render Queue rendering, streaming post-processing, parallel scheduling and recovery.

6.2 Synthetic video datasets

Rendered from existing 3D assets. Several synthetic video datasets are rendered from existing 3D assets. The BEDLAM line [3, 32] renders clothed human motion at scale, and is an asset source for later work. Dynamic Replica [18] renders thirteen animated character models for long-term correspondence. PointOdyssey [37] renders animated assets outside a game engine, specifically for long-range tracking: 104 videos averaging 2,000 frames, with deformable characters driven by real motion capture, and with characters, materials, lighting and assets randomised across sequences. Syn4D [16] composes store-sourced UE environments with animated objects and BEDLAM humans, and renders them into multi-view video with depth, dense tracks and parametric human pose; its stated aim is that any pixel can be unprojected to 3D at any time and from any camera. Its supervision is closely related to ours. Syn4D uses general multiview camera patterns, whereas WorldRover defines first-person, third-person and panoramic viewpoints with distinct traversal semantics. OmniX [15] renders 80K scenes into 1.28M multi-view videos from a UE5 data engine, with dynamic objects composited into static environments; it releases no action stream.

Recorded from video games. A second group records a finished game instead of rendering one — a technique going back to intercepting a game’s traffic to its graphics hardware [28], extended in turn to depth, cameras and instance segmentation [27] and to body joints [7]. OmniWorld [39] takes one of its four domains this way, lifting synchronised RGB and depth off the screen with a shader injector and a recorder; most of its 300M+

frames come from curated public datasets. Sekai’s game subset [20] records a UE5 landscape-photography title with a screen recorder, logging location, weather and camera trajectory as it plays, alongside a far larger subset of real video. The Generative World Renderer [14] captures 4M frames of RGB synchronised with five G-buffer channels, for inverse rendering and G-buffer-conditioned synthesis. WildWorld [21] records an action role-playing game as 108M frames with skeletons, world state, camera pose and depth, under a vocabulary of over 450 actions, compared with the four quantized axes in the WorldRover-10M trajectory-derived action stream. Gameplay recorded together with the control input that drove it is by now an established source of action-conditioned video [5, 13, 17, 36]. What can be annotated this way is decided by the capture hook, which may intercept graphics calls, record the screen, combine synchronized displays, inject a shader or read game memory. The available labels therefore differ by title and by the quantities exposed through each capture method.

7 Applications

Interactive world models. RGB, rendered camera trajectories and trajectory-derived actions provide conditioning signals without a separate pose-estimation step (Section 4.5). Third-person sequences distinguish camera motion from character motion, while matched perspective and panoramic observations provide reference content outside the perspective field of view.

Dynamic 4D reconstruction. Dense optical flow and long-range point tracks provide short- and long-range correspondence, including 3D position, 2D projection and visibility through occlusion (Section 4.4). Separate camera and character trajectories help distinguish camera motion from the motion of an articulated subject.

Generative rendering. White-model and textured pairs provide frame-aligned structural control for coarse-to-real generation [10]. Environmental variants provide further pairs in which geometry and motion are fixed while appearance changes (Section 5.2).

8 Conclusion and Future Work

We presented WorldRover, comprising WorldRover-Engine, a pipeline for rendering trajectories through artist-built UE environments, and WorldRover-10M, the resulting release. The dataset name refers to 10.8M first-person frames; including third-person and panoramic observations, the current release contains 21.9M frames in 6,003 sequences. Its task-oriented subsets provide combinations of metric depth, optical flow, long-range 2D/3D point tracks, calibrated cameras and trajectory-derived actions, together with first-person, third-person and panoramic RGB observations.

Each released annotation has a specified relationship to the pixels it accompanies. Raster annotations share a rendered instant; camera calibration describes the camera that produced the frame; a long-range track is projected through that camera and its visibility decided against the rendered depth. Each sequence declares which outputs it carries. WorldRover-Engine composes environments, routes, characters, viewpoints and environmental states, while WorldRover-10M denotes the sequences released from those configurations.

The current system has three main limitations. **(1)** Each third-person sequence contains one independently moving, trajectory-annotated character; other dynamic scene elements are not separately controlled or annotated. The follow camera also rides several metres behind that character, where the route provides no traversability guarantee. In a cluttered scene it can enter geometry, and the sequence is withheld. Populating a sequence with several tracked characters, and making the follow camera occlusion-aware, would address both. **(2)** Dynamic global illumination and artist-built assets do not model sensor effects; noise, rolling shutter and lens distortion are absent from the rendered frames. **(3)** This report describes the pipeline and the data it produces, but reports no downstream results. The next step is to train on the released data and measure the result on standard benchmarks.

References

- [1] N. Agarwal, A. Ali, J. Allen, M. Antolini, A. Aubame, A. Azzolini, J. Bai, M. Bala, Y. Balaji, J. Bapst, et al. Cosmos 3: Omnimodal world models for physical AI. *arXiv preprint arXiv:2606.02800*, 2026.
- [2] AlayaWorld Team, Alaya Lab. AlayaWorld: Interactive long-horizon world modeling — full technical report. *arXiv preprint arXiv:2607.18367*, 2026.
- [3] M. J. Black, P. Patel, J. Tesch, and J. Yang. BEDLAM: A synthetic dataset of bodies exhibiting detailed lifelike animated motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8726–8737, 2023.
- [4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012.
- [5] H. Che, X. He, Q. Liu, C. Jin, and H. Chen. Gamegen-x: Interactive open-world game video generation. In *International Conference on Learning Representations*, volume 2025, pages 37546–37593, 2025.
- [6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.
- [7] M. Fabbri, F. Lanzi, S. Calderara, A. Palazzi, R. Vezzani, and R. Cucchiara. Learning to detect and track visible and occluded body joints in a virtual world. In *European conference on computer vision*, pages 450–466. Springer, 2018.
- [8] Z. Gao, Q. Wang, J. Zhu, J. Chen, Z. Liu, Q. Bai, J. Wang, Y. Yuan, H. Wang, Y. Lu, et al. Infinite worlds with versatile interactions. *arXiv preprint arXiv:2607.07534*, 2026.
- [9] X. Ge, Y. Pan, Y. Zhang, X. Li, W. Zhang, D. Zhang, Z. Wan, X. Lin, X. Zhang, J. Liang, X. Li, W. Jiang, B. Du, M.-H. Yang, and L. Qi. AirSim360: A panoramic simulation platform within drone view. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26931–26940, 2026.
- [10] G. Gomez-Nogales, M. Comino-Trinidad, Y. Hong, C. Ge, P. Zhuang, Y. Zhou, and D. Casas. Coarse-to-Real: Generative rendering for populated dynamic scenes. *arXiv preprint arXiv:2601.22301*, 2026.
- [11] Google DeepMind. Genie 3: A new frontier for world models. <https://deepmind.google/models/genie/>, 2025.
- [12] K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, C. Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3749–3761, 2022.
- [13] J. Guo, Y. Ye, T. He, H. Wu, Y. Jiang, T. Pearce, and J. Bian. MineWorld: A real-time and open-source interactive world model on Minecraft. *arXiv preprint arXiv:2504.08388*, 2025.
- [14] Z.-H. Huang, Z. Wang, J. Tan, R. Yu, Y. Zhang, B. Zheng, Y.-L. Liu, Y.-Y. Chuang, and K. Zhang. Generative world renderer. *arXiv preprint arXiv:2604.02329*, 2026.
- [15] Y. Jiang, T. Wang, Z. Wang, C. Cao, J. Wu, W. Luo, W. Hu, J. Gao, and C. Guo. Omnix: Any-view and any-time 4d reconstruction via feed-forward trajectory fields. *arXiv preprint arXiv:2607.10840*, 2026.
- [16] Z. Jiang, Y. Lan, Y. Luo, Y. Deng, Z. Lai, E. Sucar, C. Rupprecht, I. Laina, D. Larlus, C. Zheng, and A. Vedaldi. Syn4D: A multiview synthetic 4D dataset. *arXiv preprint arXiv:2605.05207*, 2026.
- [17] A. Kanervisto, D. Bignell, L. Y. Wen, M. Grayson, R. Georgescu, S. Valcarcel Macua, S. Z. Tan, T. Rashid, T. Pearce, Y. Cao, et al. World and human action models towards gameplay ideation. *Nature*, 638(8051):656–663, 2025.
- [18] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht. DynamicStereo: Consistent dynamic depth from stereo videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13229–13239, 2023.

- [19] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.
- [20] Z. Li, C. Li, X. Mao, S. Lin, M. Li, S. Zhao, Z. Xu, X. Li, Y. Feng, J. Sun, Z. Li, F. Zhang, J. Ai, Z. Wang, Y. Wu, T. He, J. Pang, Y. Qiao, Y. Jia, and K. Zhang. Sekai: A video dataset towards world exploration. *arXiv preprint arXiv:2506.15675*, 2025.
- [21] Z. Li, Z. Meng, S. Shi, W. Peng, Y. Wu, B. Zheng, C. Li, and K. Zhang. WildWorld: A large-scale dataset for dynamic world modeling with actions and explicit state toward generative ARPG. *arXiv preprint arXiv:2603.23497*, 2026.
- [22] X. Mao, Z. Li, C. Li, X. Xu, K. Ying, and K. Zhang. Yume1. 5: A text-controlled interactive world generation model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7752–7761, 2026.
- [23] NVIDIA. Isaac Sim. <https://github.com/isaac-sim/IsaacSim>. Version 6.0.1, Apache-2.0 license.
- [24] X. Puig, E. Undersander, A. Szot, M. Dallahire Cote, T.-Y. Yang, R. Partsey, R. Desai, A. Clegg, M. Hlavac, S. Y. Min, et al. Habitat 3.0: A co-habitat for humans, avatars, and robots. In *International Conference on Learning Representations*, volume 2024, pages 15306–15336, 2024.
- [25] W. Qiu and A. Yuille. Unrealcv: Connecting computer vision to unreal engine. In *European conference on computer vision*, pages 909–916. Springer, 2016.
- [26] A. Raistrick, L. Lipson, Z. Ma, L. Mei, M. Wang, Y. Zuo, K. Kayan, H. Wen, B. Han, Y. Wang, A. Newell, H. Law, A. Goyal, K. Yang, and J. Deng. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12630–12641, 2023.
- [27] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *2017 IEEE International conference on computer vision (ICCV)*, pages 2232–2241. IEEE, 2017.
- [28] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.
- [29] M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb, and J. M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10892–10902. IEEE, 2021.
- [30] M. Roberts, R. Wang, R. Zawar, R. Dey-Prakash, Q. Leboutet, S. R. Richter, M. Müller, G. Ros, R. Tang, S. Leutenegger, et al. Spear: A simulator for photorealistic embodied ai research. *arXiv preprint arXiv:2607.06701*, 2026.
- [31] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics: Results of the 11th international conference*, pages 621–635. Springer, 2017.
- [32] J. Tesch, G. Becherini, P. Achar, A. Yiannakidis, M. Kocabas, P. Patel, and M. Black. Bedlam2. 0: Synthetic humans and cameras in motion. *Advances in Neural Information Processing Systems*, 38, 2026.
- [33] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor, and S. Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE, 2020.
- [34] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. Guibas, and H. Su. SAPIEN: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11097–11107, 2020.
- [35] X. Xu, Z. Lin, K. He, Y. Feng, X. Mao, Y. Yin, K. Zhang, and Y. Ge. WorldMark: A unified benchmark suite for interactive video world models. *arXiv preprint arXiv:2604.21686*, 2026.

- [36] J. Yu, Y. Qin, X. Wang, P. Wan, D. Zhang, and X. Liu. Gamefactorly: Creating new games with generative interactive videos. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11590–11599. IEEE, 2025.
- [37] Y. Zheng, A. W. Harley, B. Shen, G. Wetzstein, and L. J. Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19798–19808. IEEE, 2023.
- [38] F. Zhong, K. Wu, C. Wang, H. Chen, H. Ci, Z. Li, and Y. Wang. Unrealzoo: Enriching photo-realistic virtual worlds for embodied ai. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5769–5779. IEEE, 2025.
- [39] Y. Zhou, Y. Wang, J. Zhou, W. Chang, H. Guo, Z. Li, K. Ma, X. Li, Y. Wang, H. Zhu, et al. Omniworld: A multi-domain and multi-modal dataset for 4d world modeling. *arXiv preprint arXiv:2509.12201*, 2025.