Asking the right questions

Main slide 1 of 33

Talk 0:00-0:30 - 0:30 allocated



Speaker notes

What should a reconstructed program let us do? We begin with the 4DCodeBench video demonstration, define the task, and examine how the benchmark measures reconstruction quality. After its results and limitations, we raise three research questions: what makes one of several equivalent programs better, where symbolic structure should end and learned detail begin, and when a 2D sketch makes 3D program inference easier. These are open questions from our discussion. We then use SceneAgent and 3DCodeEdit to examine specialized workflows and possible experiments.

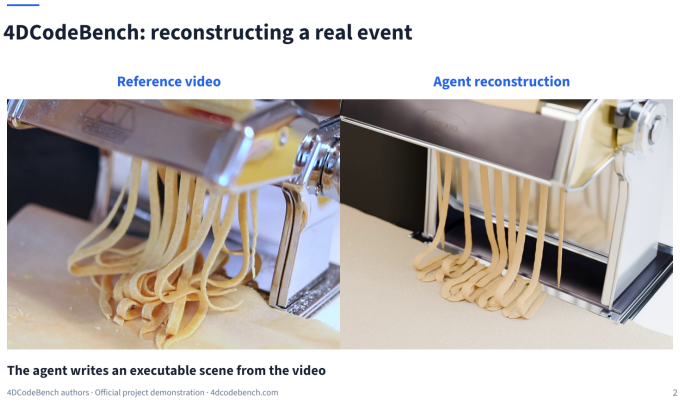
Sources

Document review, 4 November 2026 (Shared research discussion)

4DCodeBench: reconstructing a real event

Main slide 2 of 33

Talk 0:30-2:00 - 1:30 allocated



Speaker notes

Let the short clip play through once before explaining it. On the left is the real input video: pasta ribbons emerge and coil into a pile. On the right is the agent reconstruction shown in the authors' official project demonstration. The output is rendered from an executable scene. The agent must construct geometry and specify how that geometry evolves; it is not simply asked to predict another video. This side-by-side comparison combines the two original clips at their original speed and starts them together. It is one illustrative example selected by the authors, not an aggregate measure of performance. On a second loop, notice both the overall resemblance and differences in the strands and their motion. That is the motivation for our questions: what structure was recovered, what edits would it support, and what would happen under changed conditions? The PDF shows a frame at four seconds; the browser presentation and PowerPoint contain the video. Next, we define exactly what the agent receives and what its programs must produce, then examine the benchmark evidence across many scenes.

Sources

4DCodeBench official project and video demonstration (4dcodebench.com)

Original input video (4dcodebench.com/demo-pasta/gt.mp4)

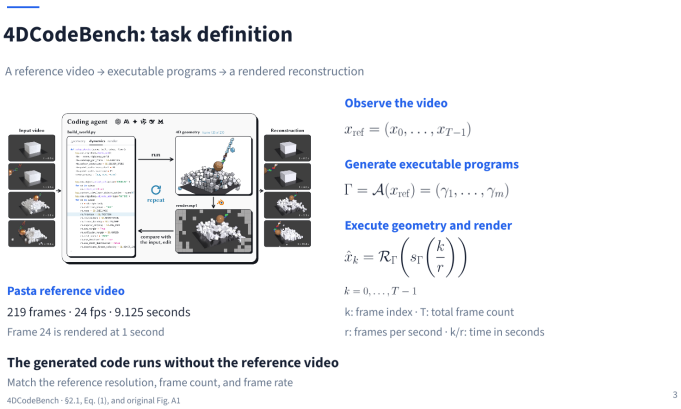
Original reconstruction video (4dcodebench.com/demo-pasta/recon.mp4)

4DCodeBench paper (arxiv.org/abs/2610.03715)

4DCodeBench: task definition

Main slide 3 of 33

Talk 2:00-4:00 - 2:00 allocated



Speaker notes

The task is executable 4D reconstruction from one RGB video. T is the total reference frame count; k is a zero-based frame index, from 0 to T-1; r is the frame rate in frames per second. Frame k represents time k/r seconds. The agent A maps the video to programs Gamma. Executing Gamma yields geometry s_Gamma(t); R_Gamma renders it using the program's camera, lighting, and materials. The programs must run without the reference video and produce the same full frame count, rate, and resolution. Analytic motion, keyframes, and simulation are all allowed. These equations specify an interface, not a required optimization loss.

The released pasta reference is 1280×960, with 219 frames at 24 fps. Thus T=219, r=24, and k=0,…,218. Its playback duration is 219/24=9.125 seconds. Frame 24 is at 1 second; the final frame is at 218/24≈9.083 seconds, followed by its display interval. Each agent receives the complete video and chooses which frames to inspect. The evaluator's 300-frame cap and the VQA/pairwise samples are separate; see the frame-sampling slide.

Figure A1 shows an agent inspecting, executing, rendering, and revising its programs. Each of 18 models runs once per scene with tools and execution loops. Proprietary models use native harnesses; others use an open agent loop, so backbone and workflow can both affect results. Successful execution does not establish correct geometry or motion. We now examine the dataset, evaluation protocol, and reconstruction results.

Sources

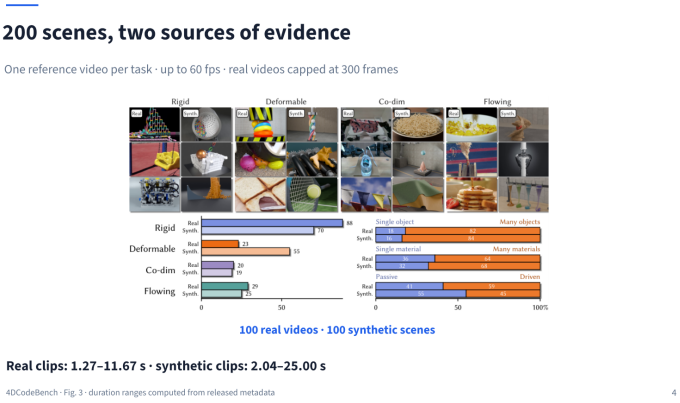
4DCodeBench · §2.1, Eq. (1): task definition (arxiv.org/html/2610.03715v1)

4DCodeBench · Fig. A1 and agent evaluation setup (arxiv.org/html/2610.03715v1)

Released real-video metadata · pasta: 219 frames at 24 fps (huggingface.co)

Official code · agent input is one complete reference video (github.com)

200 scenes, two sources of evidence



Speaker notes

The benchmark contains one hundred real videos and one hundred synthetic scenes. The two halves offer different evidence. Real footage brings appearance and dynamic complexity that are not limited to the selected simulators, but it lacks complete four-dimensional ground truth. Synthetic scenes provide known geometry and motion over time, enabling direct spatial and temporal evaluation; their realism remains conditional on the scene construction and simulation assumptions. The dataset spans rigid and articulated objects, deformable materials, fluids, grains, cloth, and rope, with categories that can overlap. Read the paper examples as a breadth illustration, not a claim that each physical regime is equally represented or equally difficult. For our discussion, the important question is which failures can be identified from images alone and which require the synthetic reference state.

Dataset construction caps every video at 60 fps; lower-frame-rate sources keep their original rate. Real videos are capped at 300 frames. There is no fixed duration shared by every case. For example, the pasta reference contains 219 frames at 24 fps (9.125 seconds). These dataset limits must not be confused with the scorer's separately configured 300-frame sampled timeline or the VLM judge's frame sampling. Each task provides one reference video, not multiple viewpoints.

Computed from the authors' released scene metadata: the 100 real videos contain 38-300 frames at 20, 24, 25, or 30 fps, with durations T/r from 1.27 to 11.67 seconds. The 100 synthetic videos contain 49-600 frames at 21, 24, 25, 30, or 60 fps, with durations from 2.04 to 25.00 seconds. Durations are playback lengths, including the last frame interval. These descriptive ranges are computed from metadata, not quoted aggregate results from the paper.

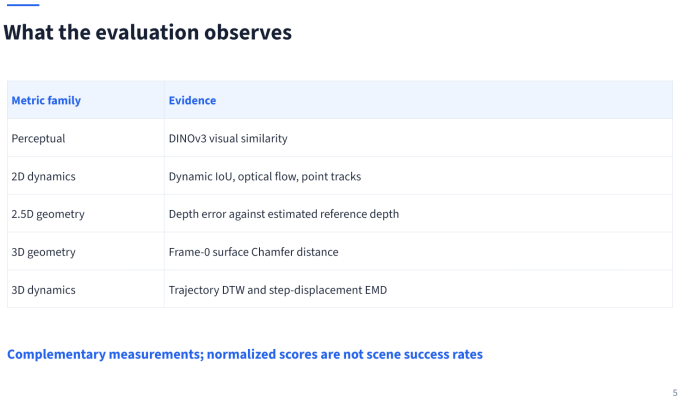
Sources

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

Released scene metadata · duration ranges computed as frame count / fps (4dcodebench.com/data/scenes.json)

What the evaluation observes



Speaker notes

The evaluation deliberately uses several views of reconstruction quality. Appearance and image-space behavior are observable across the dataset, while direct geometry and material trajectories require suitable reference information. The table shows the five separate metric families. The 2.5D family uses depth error on real videos; estimated point-map diagnostics are not part of that family's Overall contribution. Real-video depth and motion references use vision estimators; synthetic references supply 3D states. Normalized scores are not percentages of solved scenes. VQA and preference Elo are separate from the five-family Overall score. Human alignment in rankings does not make all metrics interchangeable. Ask which error each signal can detect: a convincing silhouette may conceal incorrect depth, and a plausible moving shape may contain the wrong material trajectories. These distinctions will matter when we discuss the SceneAgent image-mask IoU result.

Sources

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

Frame sampling for evaluation



Speaker notes

The code was inspected at commit 6c466be34107d7f388f30f97bf059b68ba1a21b0. One complete reference.mp4 is mounted for the agent; evaluation data and reference 3D worlds are kept separate. There is no universal fixed number of images inspected by all agents. Output render.mp4 must match the reference frame count, frame rate, width, and height. The paper uses T for total frame count; the implementation often uses F for the same quantity. The sampling module's local variable k is an integer stride, not the frame-index k in the task equation. Flow, Track2D, Depth, Semantic and GeoPhys use the shared timeline with MAX_FRAMES=300: choose the smallest stride d such that ceil(T/d)≤300, then evaluate indices 0,d,2d,… below T. For the 219-frame pasta reference this stride is 1, so the sampled timeline contains all 219 frames. This cap does not reduce the required output length. VQA samples 16 frames from the reconstruction (first, last, and 14 interior frames) without showing the reference to the judge. Pairwise judging instead samples each of three videos at 3 fps: the reference and two candidate reconstructions. These are evaluation protocols, not an agent input restriction. The pasta metadata independently confirms 219 frames at 24 fps, 1280×960, so playback duration is 9.125 seconds. Dataset construction is separately capped at 60 fps, and real videos at 300 frames. No single clip duration applies to every scene.

Sources

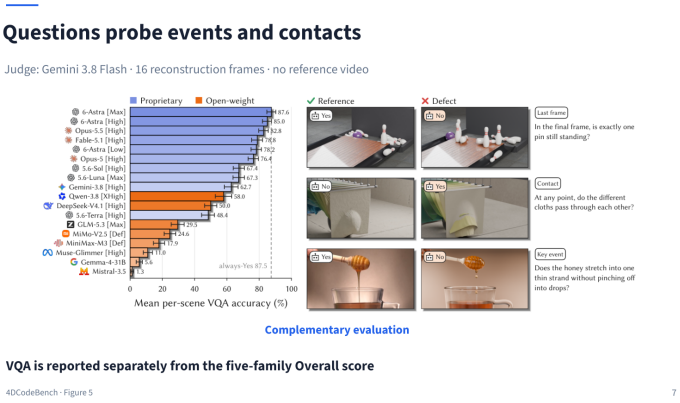
Official code · shared sampled evaluation timeline (github.com)

Official code · VQA and pairwise frame sampling (github.com)

Official code · required render settings (github.com)

Released real-video metadata (huggingface.co)

Questions probe events and contacts



Speaker notes

Gemini 3.8 Flash is the VLM judge for this VQA evaluation. It sees 16 frames of the reconstruction, including the first and last frames, and answers all of the scene's hand-written binary questions in one call. The reference video is not shown to the judge. The released code uses google/gemini-3.8-flash at temperature 0. Answers are assessed against the benchmark's supplied labels; this is separate from human pairwise preferences. The paper also asks scene-specific visual questions about reconstructed events. These questions cover initial and final states, important events, and contact, with object-presence gating. They supplement the geometric and motion metrics rather than replacing them. Use the examples in the original figure to distinguish a question about what is visible from a measurement of the hidden three-dimensional state. A correct answer may capture an important qualitative event even if a precise trajectory is imperfect; conversely, a visually plausible output can fail a specific contact or final-state requirement. The model-based evaluator also has its own limitations, which motivates comparison with human judgments on the next slide. For later research, this suggests asking which properties need language-level assessment and which can be checked directly on geometry.

Sources

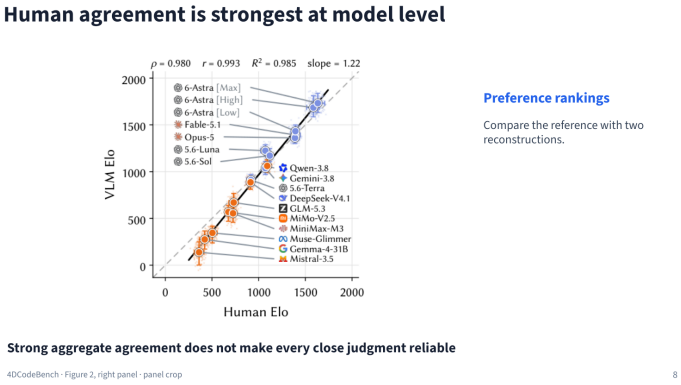
4DCodeBench · Appendix D.5: VQA judge and input (arxiv.org/html/2610.03715v1)

Released VLM judge code · model and temperature (github.com)

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

Human agreement is strongest at model level



Speaker notes

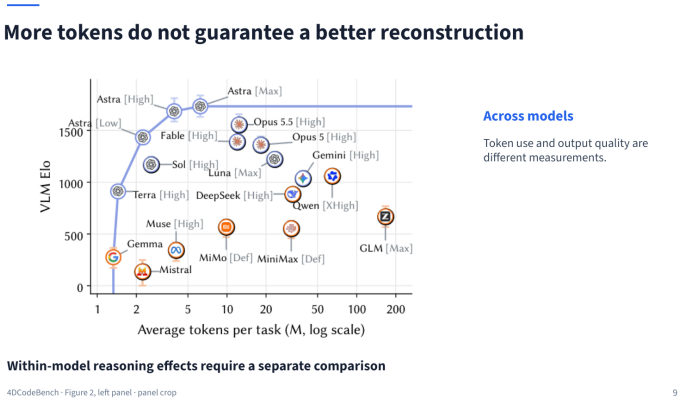
The authors compare automated rankings with human preferences to test whether their evaluation reflects perceived reconstruction quality. The displayed panel concerns model-level Elo agreement, not perfect agreement on each individual pair of videos. Human participants compare candidates against a reference, considering geometry, layout, and motion. Aggregating many such decisions can produce stable rankings even when a close individual comparison remains uncertain. This is a useful distinction when evaluating our own proposed studies: agreement at one level of aggregation does not automatically validate a checker for every failure type. Ask whether a method would still be judged correctly on small attachment errors, hidden topology, or nearly tied candidates. Those cases may require targeted measurements or an explicit uncertainty policy rather than a stronger claim about the judge.

Sources

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

More tokens do not guarantee a better reconstruction



Speaker notes

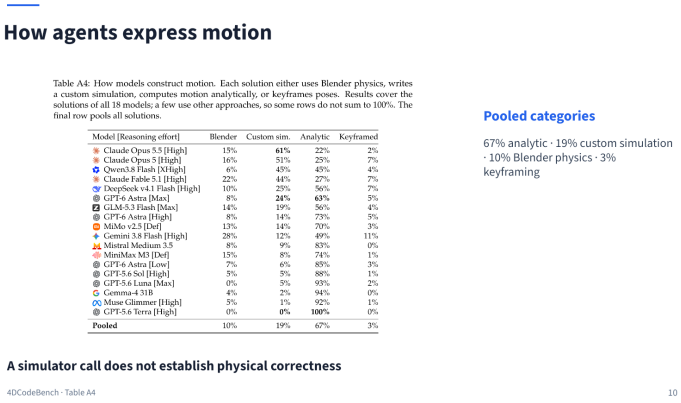
The original cost-quality panel compares models using different amounts of computation. Its message should remain separate from the effect of changing reasoning effort within one model. Across different systems, larger token consumption is not a reliable guarantee of a better reconstruction. In the paper's controlled Astra settings, however, increasing reasoning effort improves the reported reconstruction scores. These statements can both be true because backbone, harness, and reasoning configuration differ across the broader comparison. Token accounting is also not the same as elapsed time or money, especially when caching and model pricing differ. For a proposed method, report what resource changes with the intervention and compare like with like. A useful geometric measurement might save search, but that benefit must be measured rather than inferred from a shorter prompt.

Sources

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

How agents express motion



Speaker notes

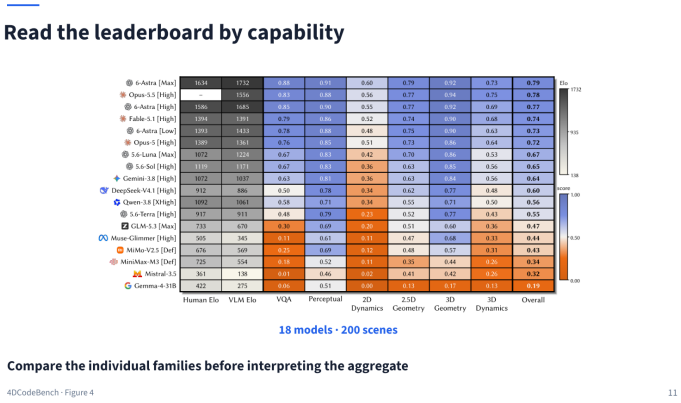
One advantage of executable outputs is that the authors can inspect how motion is implemented. Their pooled classification reports sixty-seven percent analytic motion, nineteen percent custom simulation, ten percent Blender physics, and three percent keyframing; rounded values and occasional other approaches mean rows need not sum to one hundred. The categories also vary substantially by model. More importantly, the implementation label is not itself a scientific verdict. An analytic equation may encode the correct mechanism, while a simulation may have the wrong parameters, boundary conditions, or contacts. A fitted trajectory may be perfectly acceptable for a reconstruction task that only evaluates the observed event. To assess predictive mechanisms, we need a different protocol with declared inputs and withheld interventions. Implementation choices alone do not determine reconstruction quality. Next, compare the measured outcomes in the leaderboard.

Sources

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

Read the leaderboard by capability



Speaker notes

The original leaderboard places several kinds of evidence side by side. Read the five normalized families before the Overall column, then distinguish those measurements from VQA and the human or model-based preference ratings. Each model was run once on each scene, and invalid submissions receive the worst applicable metric scores. This makes executability part of the practical result, while still leaving geometric and dynamic fidelity visible. A model can rank well overall yet have a different pattern of weaknesses from another model. Avoid reading the score as a probability that a scene is correct. The useful comparison for our purpose is which capability improves and which remains weak, because those patterns tell us what a subsequent repair or verification method would need to address.

Sources

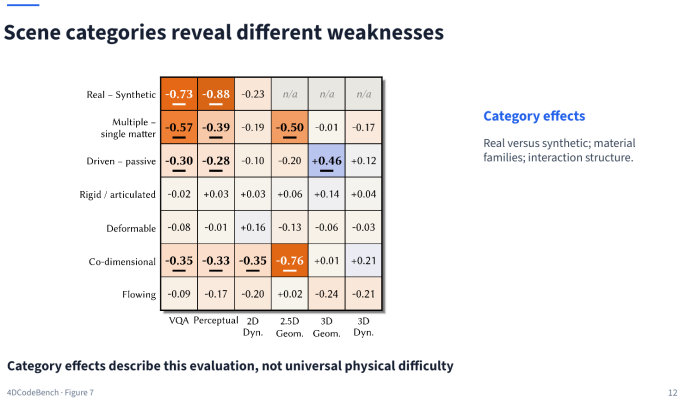
4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

Scene categories reveal different weaknesses

Main slide 12 of 33

Talk 12:30-13:30 - 1:00 allocated



Speaker notes

The original category figure examines how reconstruction scores change across subsets of scenes. Its comparisons include real versus synthetic inputs and different material or interaction categories. The paper marks effects more conservatively when their interval excludes zero and their direction recurs across most evaluated models. Do not read each colored cell as a separate causal experiment: the scene properties can co-occur, and the subsets differ in more than one respect. The figure is most useful as a diagnosis of where to inspect examples or design a controlled follow-up. For instance, a low image-space score on a material family might reflect visibility, appearance, geometry, or motion. A subsequent study should isolate those factors rather than simply declaring that one family is intrinsically harder for every representation. Across these varied scenes, motion remains weaker than appearance and static geometry; the next slide summarizes that recurring gap.

Sources

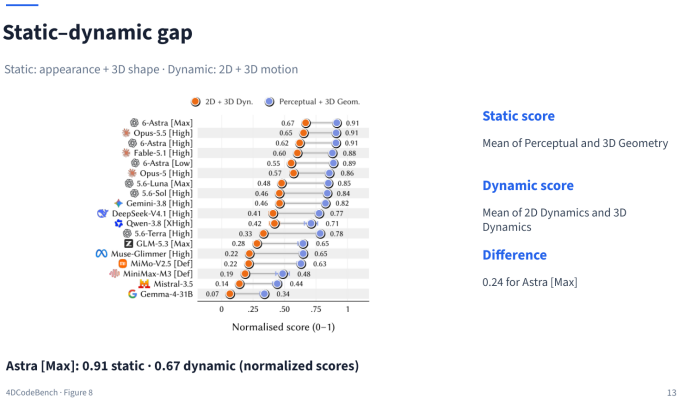
4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

Static-dynamic gap

Main slide 13 of 33

Talk 13:30-14:30 - 1:00 allocated



Speaker notes

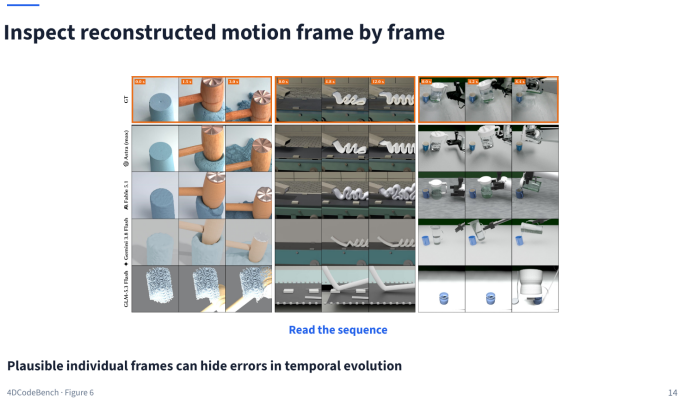
The paper finds a persistent difference between reconstructing appearance and geometry and reconstructing motion. Its strongest reported model, Astra at maximum reasoning effort, scores 0.91 over the static metric families and 0.67 over the dynamic families. Static averages Perceptual and 3D Geometry; dynamic averages 2D and 3D Dynamics. The 2.5D family is excluded. These normalized aggregates are not percentages of correctly reconstructed scenes. Use the original figure to inspect whether the pattern is isolated to one model or appears across the evaluated systems. The authors report that it recurs across all eighteen models. This supports the narrower conclusion that the benchmark exposes a dynamics bottleneck. It does not yet diagnose one universal cause or establish that any particular simulator resolves it. The same pattern persists when Perceptual is compared with 2D Dynamics and 3D Geometry with 3D Dynamics separately; the detailed comparison is retained in backup B8. These are complementary checks of the same finding, rather than two separate claims. We next inspect frame-by-frame reconstruction examples to connect the scores to visible motion errors.

Sources

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

Inspect reconstructed motion frame by frame



Speaker notes

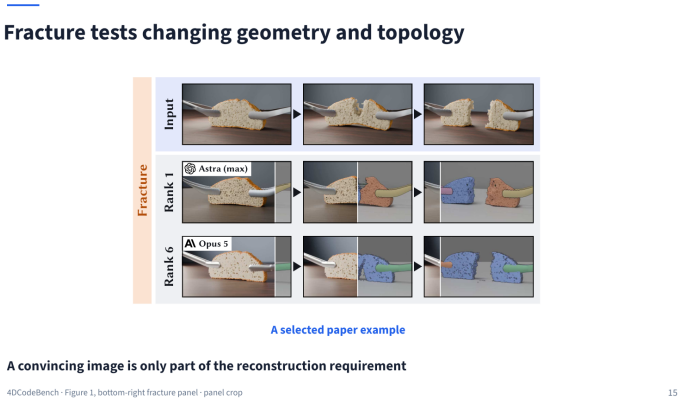
Use these original paper examples to slow down the reading of a reconstruction. First identify the reference event and the sampled times. Compare each rendered reconstruction with the reference sequence rather than judging only the first frame. Figure 6 shows rendered sequences, not a separate geometry view; hidden 3D structure and persistent material trajectories cannot be established from this figure alone. Ask the group to distinguish a misplaced shape, an incorrect deformation, a temporal offset, and the wrong interaction between materials. These failure types may look similar in a single image but require different repairs. The figure is qualitative evidence from the authors; it does not constitute a new controlled experiment or show responses to unseen forces. Avoid attributing a specific physical cause unless the displayed evidence supports it. The purpose of the exercise is to connect the metric families to observable discrepancies and to make clear why one aggregate score cannot explain every failure.

Sources

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

Fracture tests changing geometry and topology



Speaker notes

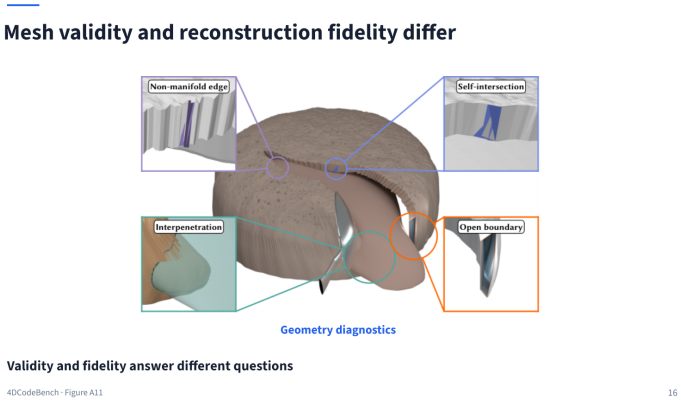
This selected fracture example gives a concrete reason to expose geometry over time. When an object separates, the reconstruction must account for the changing spatial arrangement and the geometry of its pieces, not merely retain a good initial silhouette. Read the source row labels and compare the reference with the selected model outputs. The figure illustrates the task; it does not establish a new average result or identify the material parameters responsible for a discrepancy. Fracture also anticipates the structural-edit question later in the talk: some regions persist while new interfaces become visible. The underlying goals remain different - reproducing a dynamic event here and defining semantic edit behavior later - but both show why a single correspondence or a single initial rendering may be insufficient evidence.

Sources

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

Mesh validity and reconstruction fidelity differ



Speaker notes

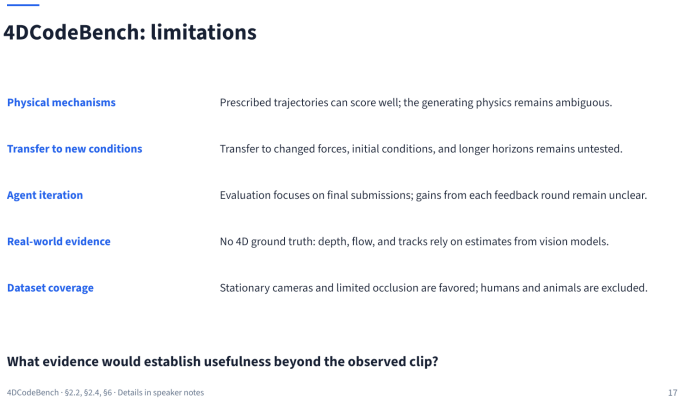
The benchmark separately inspects structural defects in the delivered meshes. The original figure shows a selected example of those diagnostics; it is not a prevalence estimate for all scenes. A reconstruction can have an open boundary or an unwanted intersection even when the rendering hides it. Conversely, a simple watertight shape can be geometrically valid while failing to match the reference event. This is why the paper reports mesh-validity diagnostics outside its Overall fidelity score. The distinction carries directly into editing evaluation. We must ask whether the requested operation happened and whether declared relationships survived, not only whether the output passes generic validity tests. Some disconnected components or intersections may be intentional, so the applicable checks must follow the task's semantics.

Sources

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)

4DCodeBench: limitations



Speaker notes

The first three points are limitations explicitly acknowledged by the authors in §6. The last two describe boundaries of the dataset and measurement design in §2.2 and §2.4. They concern what the benchmark establishes, rather than failures of a particular model.

Physical mechanisms: prescribed trajectories can achieve high reconstruction scores. Several mechanisms can explain the observed motion. Analytic motion can be appropriate; using a simulator alone does not establish the correct physics.

Generalization: the benchmark does not test changes to initial conditions or external forces, or prediction beyond the observed clip. The authors propose interventions and longer-horizon prediction as future work. To test these claims, freeze the recovered program and evaluate new conditions without another fitting round. This is our proposed evaluation principle.

Iteration: the analysis scores final submissions. Tracking intermediate reconstructions would show whether render-and-compare rounds improve quality, how much they cost, and when gains saturate. Cross-model token totals alone do not answer those questions.

Evidence and coverage: real videos lack full 4D ground truth, so depth, flow, and trajectories depend on vision-model estimates. Synthetic scenes supply geometry and motion ground truth under simulator assumptions. The real-video curation favors stationary cameras, limited occlusion, and continuous footage, and excludes humans and animals. Performance under more difficult capture conditions therefore needs separate evidence.

Judges also see finite samples: VQA uses 16 reconstruction frames; pairwise judging uses 3 fps. Our interpretation is that brief failures may escape these judges; the benchmark also includes direct numerical metrics. Aggregate rankings do not certify every contact or event. These limits motivate the next questions: which construction rule is useful, how structure should coordinate with fine detail, and which intermediate representation helps recover a useful program.

Sources

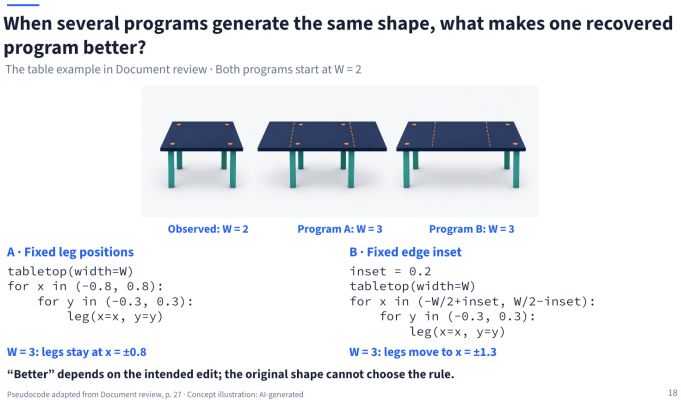
4DCodeBench, §6: authors' limitations (arxiv.org/html/2610.03715v1)

4DCodeBench, §2.2: dataset construction (arxiv.org/html/2610.03715v1)

4DCodeBench, §2.4: evaluation and ground truth (arxiv.org/html/2610.03715v1)

4DCodeBench, Appendix D.5-D.6: VLM judging (arxiv.org/html/2610.03715v1)

When several programs generate the same shape, what makes one recovered program better?



Speaker notes

This table example comes directly from the prose in Document review.pdf, page 27. The two snippets are illustrative pseudocode adapted for this slide, not verbatim code from the document and not full Blender programs. The helper functions tabletop and leg stand for identical geometry constructors in both programs. W is tabletop width; depth, height, leg dimensions, and the y coordinates stay fixed. The inset is measured from the tabletop edge to the leg center.

At W = 2, both programs place the leg centers at x = ±0.8 and y = ±0.3, so they generate the same initial geometry. Change only W to 3. Program A retains x = ±0.8. Program B uses ±(W/2 - 0.2) = ±1.3, keeping a 0.2 inset. The conceptual illustration shows these two possible edits; it is not a metric render of these snippets.

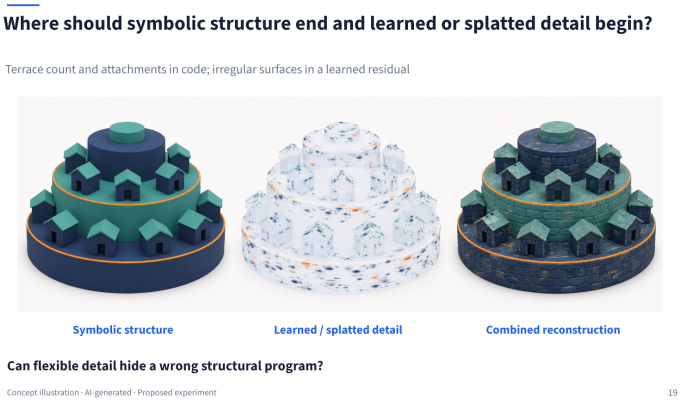
Which program is better depends on what the edit should preserve. A fixed support footprint favors A; a fixed distance from the tabletop edges favors B. Neither rule can be selected from the original shape alone. Compactness, editability, reuse, and similarity to the designer's process can prefer different programs. A second object variant, an edit example, an explicit constraint, or an instruction can help resolve the ambiguity. This is a question about recovering useful dependencies, rather than merely producing runnable code.

Keep the programs frozen while applying the declared parameter edit; letting an agent rewrite either program after seeing the desired result would test a different ability. The next slide asks how explicit construction rules should coordinate with fine detail that a simple program does not describe.

Sources

Document review, 4 November 2026 (Shared research discussion)

Where should symbolic structure end and learned or splatted detail begin?



Speaker notes

This is question 17 in the forty-question collection, on page 16 of Document review.pdf. The illustration is a conceptual decomposition, not output from a trained hybrid method. The left view assigns terrace count, radii, repeated house modules, and attachments to a structural program. The middle view suggests fine irregular surface geometry or appearance represented by learned features or splats. The right view combines them. The exact split is a research choice; splats are not inherently restricted to fine detail.

The central risk is that a sufficiently flexible residual explains most of the object, or compensates for an incorrect structural program. A good initial render therefore does not establish that the symbolic part captures useful relationships. How should we constrain residual capacity and attach detail to parts so that both components remain meaningful?

A controlled experiment would match initial reconstruction quality where possible, vary structural constraints and residual capacity, and then apply withheld structural edits. Resize a terrace, change the number of houses, or move a part. Measure whether detail stays attached, whether boundaries remain coherent, which structural constraints survive, and the full cost of the program and learned representation. Compare symbolic, continuous, and hybrid alternatives under the same observations. These are proposed tests, not established advantages of a hybrid.

The next question concerns an earlier inference choice: whether a construction sketch can make recovery of the structural program easier.

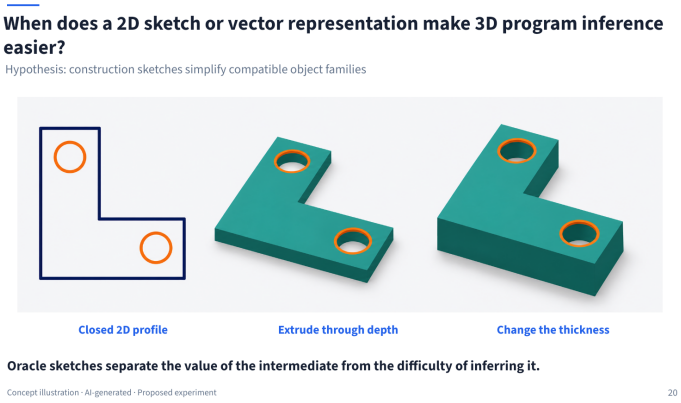
Sources

Document review, 4 November 2026 (Shared research discussion)

When does a 2D sketch or vector representation make 3D program inference easier?

Main slide 20 of 33

Talk 23:00-24:30 - 1:30 allocated



Speaker notes

This is question 13 in the forty-question collection, on page 15 of Document review.pdf, with a proposed experiment on page 22. The illustration shows one compatible family: a closed planar profile with two holes, extruded to produce a solid. Changing extrusion depth gives another variant while preserving the profile. It is conceptual and not a metric execution of a recovered program.

The hypothesis is that a construction-aligned intermediate can simplify inference for some object families. Tracing a contour in a perspective image does not automatically recover the canonical construction sketch. The system must still infer the relevant plane and profile, the camera, depth, hidden parts, and a suitable lifting operation. The claim is about inference difficulty under a construction grammar, not a general reduction of all 3D geometry to easy 2D tracing.

Use the same input images, model, output language, and compute budget. Compare direct 3D program inference, image-space vectorization, inferred canonical sketches, and oracle canonical sketches. Oracle means supplying the correct construction sketch; it is a diagnostic condition rather than a realistic input assumption. If oracle sketches help but predicted sketches do not, recovering the sketch is a bottleneck. If even oracle sketches do not help, the intermediate may omit needed information, the chosen grammar may be inappropriate, or downstream program inference may be failing.

Vary camera pose, occlusion, and construction family, including cases that are poorly described by the sketch grammar. Measure final geometric correctness, execution success, edit behavior, and total cost. The next slide broadens the discussion to feedback for repairing programs, physical mechanism recovery, and reusable rules inferred from Gaussian scenes.

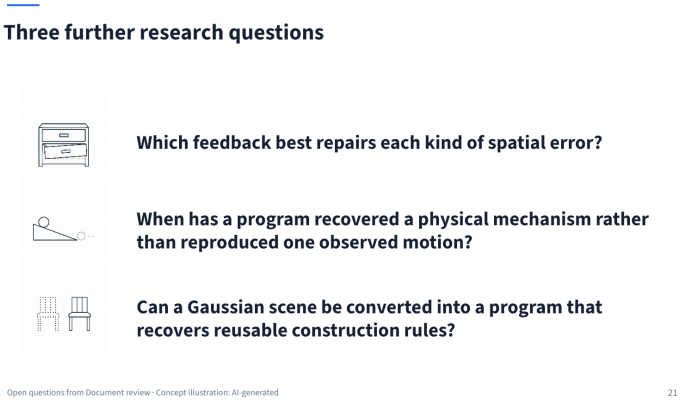
Sources

Document review, 4 November 2026 (Shared research discussion)

Three further research questions

Main slide 21 of 33

Talk 24:30-26:00 - 1:30 allocated



Speaker notes

These are questions 37, 33, and 29 from the forty-question collection in Document review, in the order requested for this slide. The illustration is conceptual, not an experimental result. Spend roughly thirty seconds on each direction, then carry the questions into the SceneAgent and 3DCodeEdit discussion.

Feedback for repair: different errors may require different observations. A missing part, a wrong dimension, a broken attachment, and a collision need not benefit from the same feedback. Compare rendered views, geometric measurements, constraints, localized language, and combinations under matched budgets. Measure successful repairs, damage to previously correct regions, and the cost of obtaining feedback. A useful result is a map from error type to effective feedback, followed by a method that selects its next observation.

Physical mechanisms: reproducing one observed trajectory can leave masses, forces, contacts, and constitutive choices ambiguous. Freeze the recovered program and test held-out initial conditions, controlled forces, or longer horizons. Determine which interventions distinguish competing explanations. This can provide evidence for mechanism recovery; it does not automatically identify a unique true mechanism, and simply calling a physics simulator is insufficient evidence. This connects directly to the limitations acknowledged by 4DCodeBench.

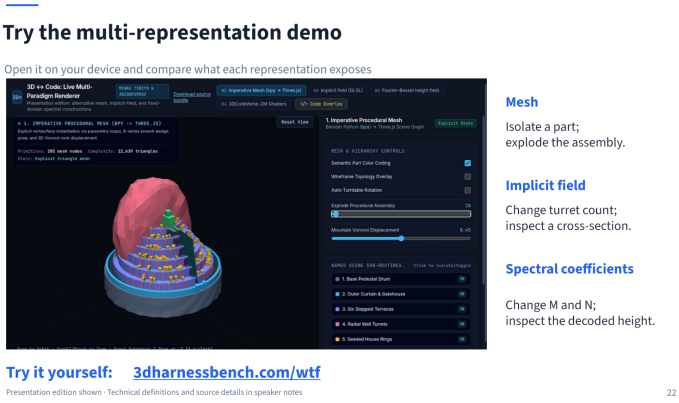
Reusable rules: a Gaussian scene might support recovery of repeated components, symmetry, attachments, and shared parameters. The target is a program that produces useful new variants, rather than a literal serialization of splat positions. Vary repetition count, dimensions, or spacing after inference and test geometric validity and consistency. Check stability across different splat reconstructions of the same object, and allow residual detail without letting it conceal a wrong structural rule. Recovering the original source file is generally not a uniquely defined target.

These questions connect recovery, intervention, and verification. Before turning to SceneAgent, the Minas Tirith example makes the representation choices concrete through definitions, code, and a live renderer.

Sources

Document review, questions 37, 33, and 29 (Shared research discussion)

Try the multi-representation demo



Speaker notes

Invite the audience to open https://3dharnessbench.com/wtf on their own devices. The slide shows the address prominently and links to it from the browser deck, PDF, and PowerPoint. Ask participants to try one operation in each representation and compare which quantities are directly controlled. The screenshot and equations in this section refer to the archived presentation edition, cited in the references; the independently hosted audience version may differ. The downloadable bundle includes the original input and the documented presentation changes, with runtime dependencies bundled locally.

Keep the audience demo open while explaining the definitions and the next three cases. Across the walkthrough, in the mesh tab, enable part colors or isolate the houses and show a small assembly explosion. This makes object identity and transforms visible. In the implicit tab, change the repeated turret count and enable the field cross-section; the solid is determined by field evaluation rather than a list of instantiated objects. In the spectral tab, change angular and radial bandwidth and reset the camera if the height range changes. The parameters index a fixed basis rather than semantic city parts.

Use the Code Overlay to compare the illustrative excerpts with the actual browser implementation cited in the references. Do not describe the three tabs as exactly the same geometry, automatic translations, or reconstructions fitted to the photograph. The exercise is to ask which operations each representation makes easy and what assumptions its decoder supplies. The fourth shader tab is optional and outside this short comparison.

First define the mesh, implicit-field, and spectral representations, then explain how each appears in this example. Return to the open demo as each case is discussed.

Sources

Audience demo: try it yourself (3dharnessbench.com/wtf)

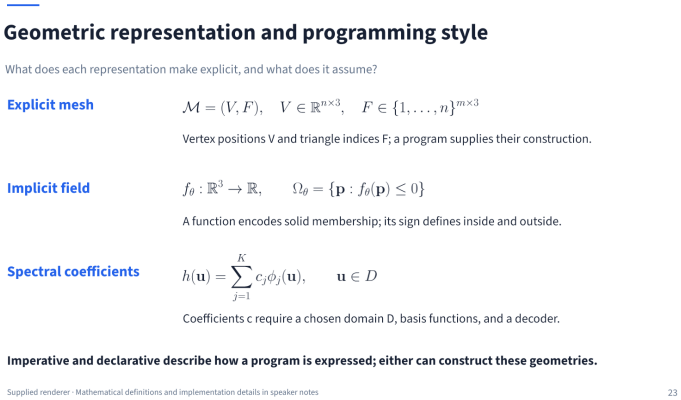
Supplied live multi-paradigm renderer (sites.google.com/view/3d-code-multi-paradigm-reconst)

Presentation demo and downloadable source (filedn.eu/lmI1NlfngedQA4sE1CySaDp/s2/talk/2026-11-05/demo)

Geometric representation and programming style

Main slide 23 of 33

Talk 28:00-29:30 - 1:30 allocated



Speaker notes

Separate the geometry being represented from the programming interface used to construct it. An explicit mesh stores vertex coordinates and triangle connectivity; a procedural generator can additionally encode part identity, repetition, and dependencies. A declarative interface can also produce a mesh. Conversely, a field evaluator can be written in imperative GLSL. The terms are not competing definitions of the same axis.

For the mesh definition, n is the number of vertices and m is the number of triangular faces; each row of F contains three vertex indices. The program may output several meshes together with object transforms. For an implicit model, a negative-inside scalar field defines the solid through an inequality. With suitable continuous defining functions, a zero level set describes its boundary. The supplied demo includes discontinuous constructions, so that stronger statement needs care.

A spectral representation stores coefficients of a function on a specified domain. The basis and decoding rule are essential parts of the representation, not information recovered from the coefficient array alone. The following demo uses a fixed disk and produces a height field. It is not the intrinsic spectrum of the city mesh.

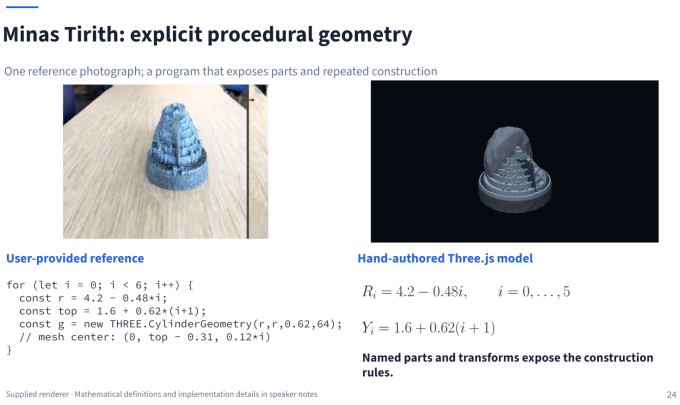
The three city tabs are independently authored approximations inspired by the photograph. They do not demonstrate lossless conversion or equal reconstruction quality. The shader examples in the fourth tab are additional programs, not a fourth independent geometric representation family.

Sources

Supplied live multi-paradigm renderer (sites.google.com/view/3d-code-multi-paradigm-reconst)

Presentation demo and downloadable source (filedn.eu/lmI1NlfngedQA4sE1CySaDp/s2/talk/2026-11-05/demo)

Minas Tirith: explicit procedural geometry



Speaker notes

The left image is the photograph supplied for this talk. The right image is captured from the actual browser renderer. This is a schematic reconstruction authored by code, not an optimization result fitted to the photograph. All dimensions shown are demo units rather than measurements of the photographed object.

The code excerpt is abridged from the running Three.js implementation. It constructs six cylindrical tiers with radii R_i=4.2-0.48i, upper elevations Y_i=1.6+0.62(i+1), thickness 0.62, and 64 radial segments. Their centers use y=Y_i-0.31 and z=0.12i. The full program creates the mesh objects, assigns materials, adds walls, houses, a wedge, and a displaced mountain, and groups these parts for inspection.

The important program information is the dependency: changing rStart or rStep changes an entire tier family. The resulting vertex and face arrays alone need not retain that construction relationship. Part identity and scene-graph transforms support isolation and assembly inspection in the demo.

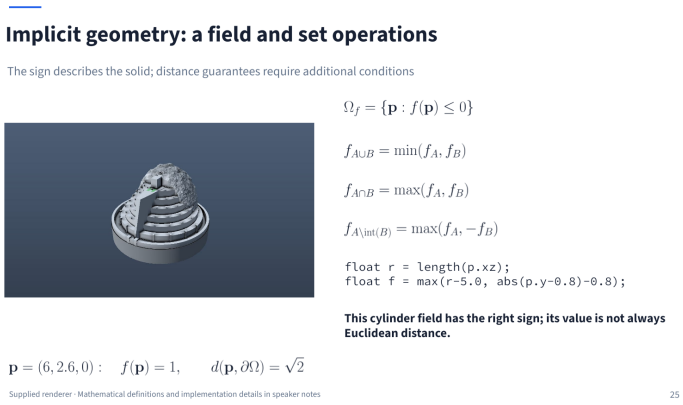
The code overlay in the app shows an illustrative Blender-style excerpt. The browser itself executes Three.js; it does not execute bpy. The complete JavaScript implementation is in the downloadable source bundle. Compare the reference and model openly: fine detail and several shapes differ, so executable code is not evidence of an accurate geometric reconstruction.

Sources

Supplied live multi-paradigm renderer (sites.google.com/view/3d-code-multi-paradigm-reconst)

Presentation demo and downloadable source (filedn.eu/lmI1NlfngedQA4sE1CySaDp/s2/talk/2026-11-05/demo)

Implicit geometry: a field and set operations



Speaker notes

The screenshot is the reviewed demo's separately authored implicit model. For negative-inside defining fields, min encodes union, max encodes intersection, and max(f_A,-f_B) encodes CSG difference with the cut boundary retained: A minus the interior of B, for defining fields whose zero set is the boundary. These are statements about membership. They do not generally preserve an exact Euclidean signed-distance value.

The two code lines come from the actual base-cylinder definition. The solid has radius 5, center height 0.8, and half-height 0.8. At p=(6,2.6,0), the point is one unit outside the radial boundary and one unit above the top. The field gives max(1,1)=1, while the nearest point on the cylinder rim is at distance sqrt(2). This is a concrete distinction between a valid defining field and an exact signed-distance function.

A signed-distance function uses minus the distance to the boundary inside and plus the distance outside. Sphere tracing requires a certified distance bound: if a field is L-Lipschitz, |f(p)|/L bounds the distance to its zero set from below. The supplied field contains floor-selected terraces, noise, scaling, and smooth combinations; the demo does not establish the required global bound. Its ray march remains heuristic. Some boundaries of a discontinuous field need not be zero level sets.

The presentation copy adds the missing lower cap on the turret field and rejects rays that exhaust their step budget without a valid hit. These repairs make the displayed finite scene clearer but do not certify the composed field as an SDF. The original source is retained alongside the reviewed copy. Field slicing exposes inside/outside structure that a single rendered view can obscure.

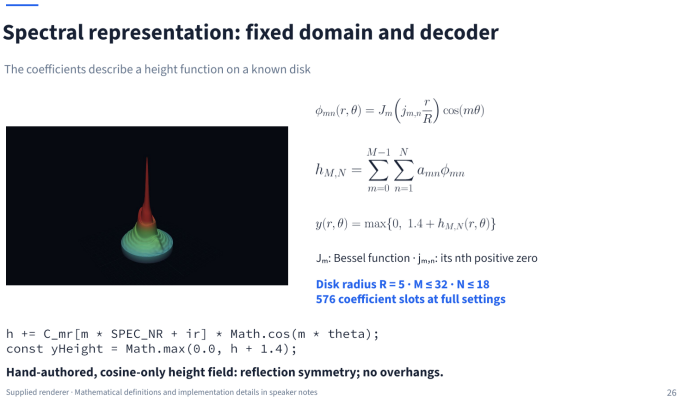
Sources

Supplied live multi-paradigm renderer (sites.google.com/view/3d-code-multi-paradigm-reconst)

Presentation demo and downloadable source (filedn.eu/lmI1NlfngedQA4sE1CySaDp/s2/talk/2026-11-05/demo)

Hart, Sphere tracing (1996) (doi.org/10.1007/s003710050084)

Spectral representation: fixed domain and decoder



Speaker notes

Let the audience inspect the basis and decoder before showing the result. The rendered example appears automatically after ten seconds in the live presentation.

The basis is defined on a planar disk of known radius R=5. J_m is the Bessel function of the first kind of integer order m, and j_m,n is its nth positive zero. The disk Dirichlet eigenfunctions have radial factor J_m(j_m,n r/R) and eigenvalues (j_m,n/R)^2. A general real expansion also includes sine modes. The actual demo uses cosine modes only, so h(r,theta)=h(r,-theta): it imposes reflection symmetry.

The program fills a 32 by 18 array with hand-authored coefficients for radial structure, directional components, high angular modes, and stochastic detail. It does not project or fit the explicit mesh into this basis. Rectangular truncation in M and N is not the same as selecting the K smallest Laplacian eigenvalues. Increasing bandwidth reveals more of this prescribed function; it does not establish convergence to the photographed city. The screenshot shows the actual resulting field, including its tall central peak.

The code first sums radial modes into C_mr, then sums their angular cosine factors and clips y=max(0,1.4+h). It evaluates an 80 by 160 polar sampling grid and adds a pedestal skirt for display. A height field assigns one height to each planar location and cannot generally represent overhangs, tunnels, or separate stacked surfaces. The domain, frame, basis, and clipping rule all belong to the decoder.

The original high-mode root calculation was inaccurate. The presentation copy uses a 256-node integer-order Bessel integral with bracketed root refinement. All 576 roots and 46,080 sampled radial basis values were independently checked against libc jn before making these captures. This is a numerical repair of the example, not evidence of better inferred geometry. The full basis and domain interpretation can be revisited in the spectral backup slide.

Sources

Supplied live multi-paradigm renderer (sites.google.com/view/3d-code-multi-paradigm-reconst)

Presentation demo and downloadable source (filedn.eu/lmI1NlfngedQA4sE1CySaDp/s2/talk/2026-11-05/demo)

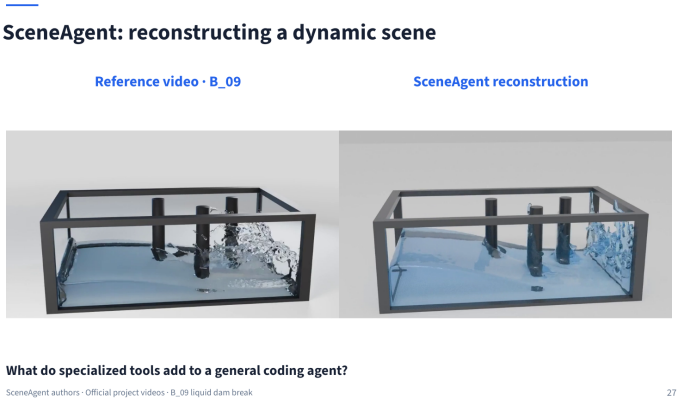
Fourier-Bessel basis on the disk, §1.1 (web.math.princeton.edu/~amits/publications/2207.13674.pdf)

NIST DLMF, integer-order Bessel integral (dlmf.nist.gov/10.9.E2)

SceneAgent: reconstructing a dynamic scene

Main slide 27 of 33

Talk 34:45-36:00 - 1:15 allocated



Speaker notes

Play the authors' reference/reconstruction pair for the liquid dam break around rigid pillars. Both source clips contain 130 frames at 24 fps (5.417 seconds), begin together, and retain their original speed. The presentation scales the two full frames equally, places them side by side, and adds white padding; the PDF poster is the combined frame at 2.5 seconds. The clip is a selected qualitative example from the official SceneAgent website. Its displayed reconstruction is not identified here with a particular model.

Ask what the specialized workflow adds: tools for reconstruction, simulation, and inspection, together with reusable guidance for coordinating them. The comparison motivates the next pipeline figure. Resemblance in this observed event does not establish response to a new force or a different initial condition.

Sources

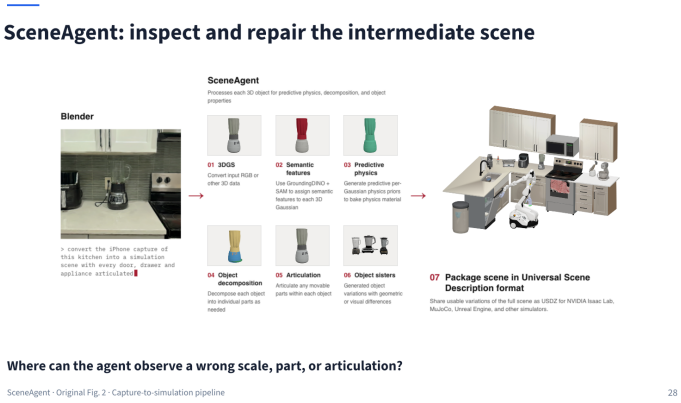
SceneAgent official project and demonstration (computationalrobotics.seas.harvard.edu/SceneAgent)

B_09 reference clip (computationalrobotics.seas.harvard.edu)

B_09 reconstruction clip (computationalrobotics.seas.harvard.edu)

SceneAgent paper (arxiv.org/html/2610.34745v1)

SceneAgent: inspect and repair the intermediate scene



Speaker notes

Use the original pipeline figure to locate segmentation, physical modeling, part decomposition, articulation, and export. A plausible output from each tool does not guarantee a consistent assembled scene: scale, coordinate frames, segmentation, or joint assumptions can be wrong at their interfaces. The workflow inspects simulator renders and repairs intermediate outputs.

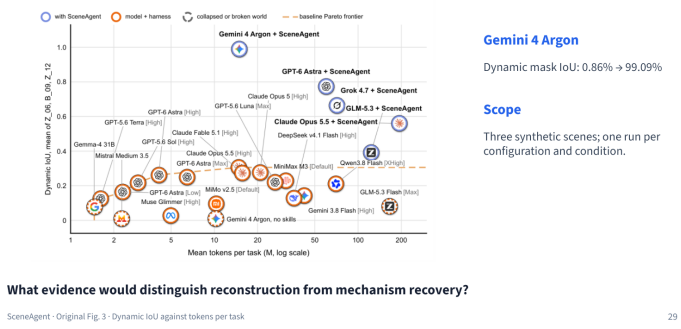
The labels in the original figure describe the authors' components. For static captures, physical properties are approximate priors; the paper explicitly leaves accurate physical parameter identification to future work. Its dynamic pathway additionally fits simulated motion to a video. This distinction matters when asking what the resulting program is justified in predicting.

Sources

SceneAgent paper, Fig. 2 and §3 (arxiv.org/html/2610.34745v1)

SceneAgent official project (computationalrobotics.seas.harvard.edu/SceneAgent)

SceneAgent: a large gain on three dynamic scenes



Speaker notes

The original plot places the selected headline alongside the other harnesses. On Z_06, B_09, and Z_12, mean dynamic image-mask IoU for Gemini 4 Argon rises from 0.86% without the skills to 99.09% with them. The evaluation is narrow: each configuration is run once per scene and condition. The baseline token averages in the plot come from the published 100-task benchmark, whereas the SceneAgent conversions use the three selected tasks.

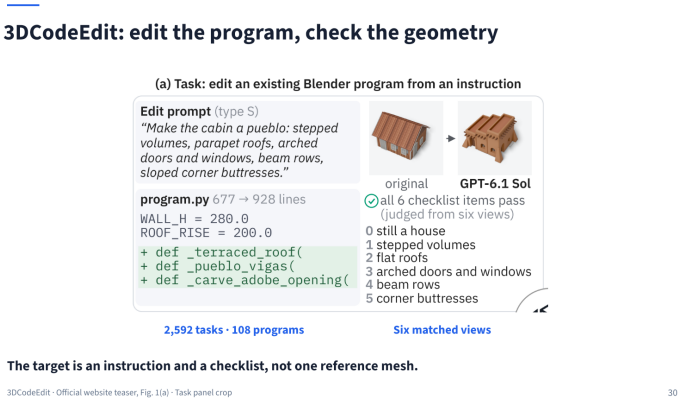
Dynamic mask overlap measures reconstructed motion in rendered images. It does not score recovered material parameters or predict behavior under held-out interventions. The useful follow-up is to specify the additional evidence that would support those stronger claims.

Sources

SceneAgent paper, Fig. 3 and §4.2 (arxiv.org/html/2610.34745v1)

SceneAgent official project (computationalrobotics.seas.harvard.edu/SceneAgent)

3DCodeEdit: edit the program, check the geometry



Speaker notes

The authors' task panel shows a source program and edit instruction, followed by the edited program and its geometry. The benchmark contains 2,592 tasks over 108 Blender programs. Both objects are rendered from six matched views. A verifier checks each requested change and preservation requirement; every item must pass.

This gives a concrete definition of edit success while allowing several valid outputs. A different shape can satisfy the same request. An unchanged object can be geometrically valid and still fail. The next example asks whether executable code also puts interacting parts in the right places.

Sources

3DCodeEdit official project (binjiew.github.io/3DCodeEdit)

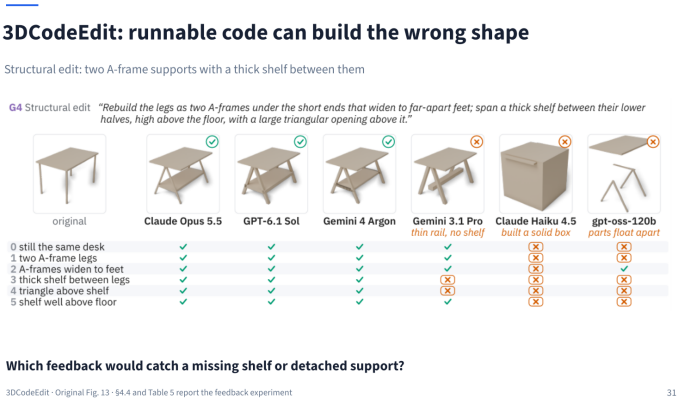
Official project teaser (binjiew.github.io/3DCodeEdit/assets/img/teaser.png)

3DCodeEdit paper, Fig. 1 and §3 (arxiv.org/html/2610.38231v1)

3DCodeEdit: runnable code can build the wrong shape

Main slide 31 of 33

Talk 40:15-42:00 - 1:45 allocated



Speaker notes

Compare the desk outputs against the original checklist. Some programs execute but leave a missing shelf, build an unintended solid box, or detach parts. This is a concrete geometric-foresight failure: code correctness alone does not establish the requested geometry.

The paper separately tests multi-turn feedback on 312 tasks with up to eight revision rounds. For GPT-6.1-sol, all-pass rate rises from 64.7% single-shot to 85.6% with an independent reviewer that describes how renders differ from the request. The reviewer is GPT-6-luna, except that GPT-6.1-sol reviews GPT-6-luna. These percentages belong to that analysis subset, not to the pictured desk alone or the full benchmark.

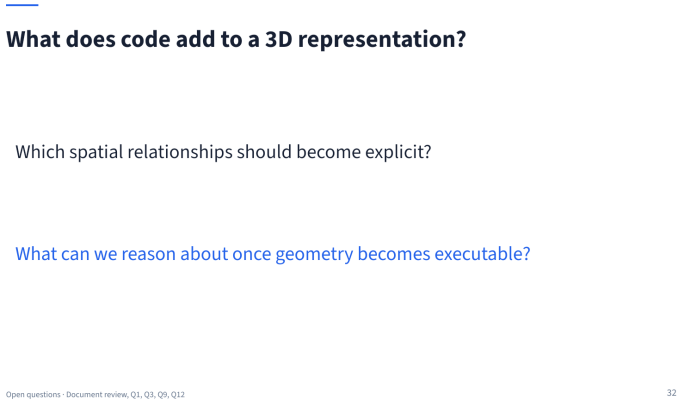
The research question is what each feedback channel makes observable: renders, localized descriptions, dimensions, connectivity, and contact checks may expose different errors.

Sources

3DCodeEdit paper, Fig. 13 and §4.4 (arxiv.org/html/2610.38231v1)

3DCodeEdit official project and interactive examples (binjiew.github.io/3DCodeEdit)

What does code add to a 3D representation?



Speaker notes

Step back from the individual benchmarks. Here 3D includes shapes, scenes, and how they behave. Code has a broad meaning: procedures, constraints, relations, or learned modules with executable interfaces. A file listing vertices is also executable through a decoder; the research question is which additional structure gives us useful capabilities. No representation family is inherently more intelligent merely because we call it code.

Which relationships should be explicit? Coordinates describe a configuration. A relation such as attachment, symmetry, or constant clearance can describe how configurations depend on one another. Choosing those concepts also introduces assumptions. Which assumptions help us infer unseen structure, ask spatial questions, or carry out meaningful changes? How much should the language guarantee, and how much should an agent discover?

What can we reason about once geometry becomes executable? Execution can expose consequences of a construction, propagate constraints, and test candidate changes. It does not by itself establish that the construction is true or the physical model is adequate. The broad opportunity is a representation whose operations make useful reasoning accessible. The question is which reasoning becomes easier, for whom, and at what computational cost.

Use these three minutes to connect the table dependencies, the multi-representation example, and the two agent systems. Do not return to a benchmark checklist. The question is what spatial knowledge should become operational. This slide synthesizes the discussion in Document review rather than reporting a new experimental finding.

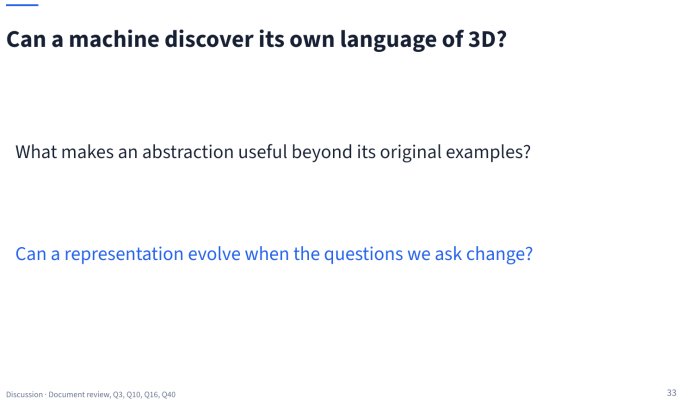
Sources

Document review, foundational research questions (Shared research discussion)

Can a machine discover its own language of 3D?

Main slide 33 of 33

Discussion 45:00-60:00 - 15:00 allocated



Speaker notes

Leave this slide up for the fifteen-minute discussion. Language means a vocabulary of spatial concepts together with ways to compose and execute them, not necessarily a textual programming language. It may include reusable functions, relational constraints, continuous fields, or learned operators. The question is whether a system can discover useful concepts rather than only fill parameters in a vocabulary chosen in advance.

What makes an abstraction useful beyond its original examples? A repeated arrangement, an attachment rule, or a motion constraint should help with new combinations or new questions. Compactly encoding familiar examples is one possible benefit; it is not sufficient evidence that the abstraction transfers. Its usefulness also depends on the tasks and on what the interpreter or library already knows. Invite different views on whether good abstractions should primarily aid explanation, construction, or interaction.

Can a representation evolve when the questions we ask change? A representation adequate for rendering may omit what is needed for reasoning about contact, assembly, or structural modification. Could a system recognize that limitation and introduce a new relation, operation, or level of detail? Such a change should retain the meaning of existing parts and previously established constraints where possible. There need not be a single best representation for every purpose.

These are open questions inspired by Document review Q3, Q10, Q16, and Q40. They do not claim that the presented systems autonomously discover a general language of 3D. Start with whichever question the group finds more consequential, and let the discussion connect geometry expertise with programming languages, learning, and models of the physical world.

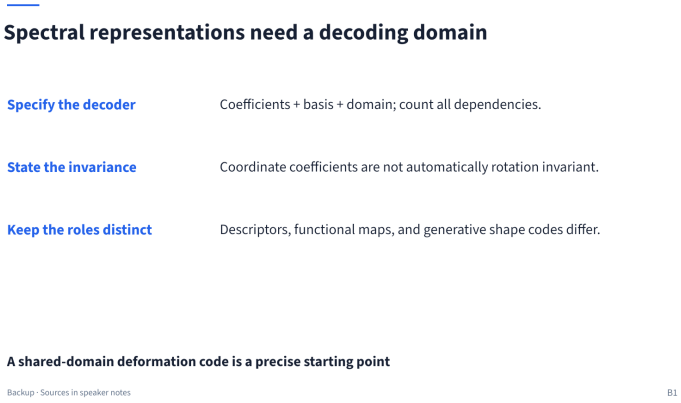
Sources

Document review, foundational research questions (Shared research discussion)

Spectral representations need a decoding domain

Backup 1 of 8 - untimed

Available for questions



Speaker notes

Spectral coefficients are meaningful relative to a basis and its domain. Projecting coordinates onto a Laplace-Beltrami basis can compactly encode a signal on an available surface, but the coefficients alone are not necessarily a complete encoding of a previously unknown shape. If the basis is computed from the target, its information and storage must be accounted for. Coordinate coefficients also rotate when centered coordinates rotate; using an intrinsic operator does not automatically make those coefficients invariant. Eigenvalues, descriptors, eigenfunctions, and functional maps have distinct roles and ambiguities. DiffuMatch supplies a learned prior over functional maps for correspondence, rather than a universal small-code mesh decoder. A productive narrower study would use a declared shared domain or partwise bases and evaluate the specific invariances and edits the representation is intended to support.

Sources

Document review, 4 November 2026 (Shared research discussion)

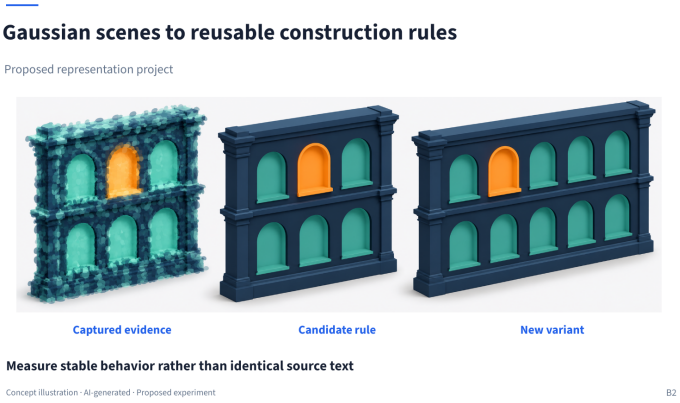
DiffuMatch (arxiv.org/abs/2507.23715)

pyFM spectral projection documentation (robinmagnet.github.io/pyFM/auto_examples/plot_03_spectrum.html)

Gaussian scenes to reusable construction rules

Backup 2 of 8 - untimed

Available for questions



Speaker notes

The appealing target is an editable procedural explanation of captured geometry, with residual detail where the program is insufficient. It is not recovery of a unique original source file. Begin with a constrained family containing repetition, symmetry, attachments, and deliberate exceptions. Use the same observations for direct inference and routes through splats, meshes, or measured parts. Vary splat reconstruction seeds and density to test whether inferred rules are stable. Then change repetition count, spacing, or dimensions and evaluate the resulting variants. ShapeCoder already discovers structural and parametric abstractions from unstructured primitives, so novelty must extend beyond applying abstraction discovery to another input format. Gaussian centers and covariance summarize a rendering representation; they do not directly provide exact surfaces, extents, or physical masses. A hybrid residual must not conceal an incorrect scaffold.

Sources

Document review, 4 November 2026 (Shared research discussion)

3D Gaussian Splatting (arxiv.org/abs/2308.04079)

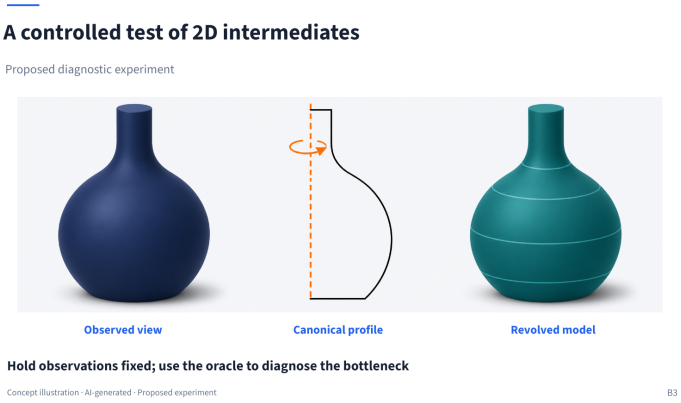
ShapeCoder (arxiv.org/abs/2305.05661)

TRELLIS (arxiv.org/abs/2412.01506)

A controlled test of 2D intermediates

Backup 3 of 8 - untimed

Available for questions



Speaker notes

Does vectorizing in two dimensions make three-dimensional coding easier? That is a useful hypothesis for compatible construction families, not a universal shortcut. A perspective silhouette is not necessarily the canonical profile required for extrusion, revolution, or lofting. Compare direct image-to-program inference, image-space vectorization, predicted canonical sketches, and oracle sketches, using the same output language and backbone. The oracle condition is diagnostic because it receives extra information. Vary view, occlusion, and construction family, and include objects that do not fit the chosen grammar. If the oracle helps but the prediction does not, sketch recovery is a bottleneck. If the oracle does not help, investigate whether it omits necessary information or whether the downstream model fails to exploit it. Include total inference cost.

Sources

Document review, 4 November 2026 (Shared research discussion)

Nearby work defines the contribution

Backup 4 of 8 - untimed

Available for questions



Speaker notes

The proposed directions become precise when distinguished from nearby prior work. Semantic editing with parametric implicit templates already transfers detailed meshes through template edits, and BeCoS provides procedural full and partial correspondence data. ParSEL constructs parameterized editing programs and propagates edits to preserve relationships. ShapeCoder learns structural and parametric abstractions. The proposed opportunities concern structural changes, incomplete evidence, stable identities, and uncertainty, not the invention of editing or abstraction discovery. For verification, 3DCodeEdit already uses instruction-derived checklists and reviewer feedback; directly executable tests for hidden geometry are a sharper extension. For dynamics, interventions already appear in 4DCodeBench's outlook. Additional specialist directions include choosing parameter fitting versus structural rewriting and studying task-appropriate spectral invariance, but each needs its own declared setting and decisive experiment.

Sources

Document review, 4 November 2026 (Shared research discussion)

Semantic Shape Editing with Parametric Implicit Templates (udaykusupati.github.io/publications/ish/ish)

BeCoS (nafieamrani.github.io/BeCoS)

ParSEL (arxiv.org/abs/2405.20319)

ShapeCoder (arxiv.org/abs/2305.05661)

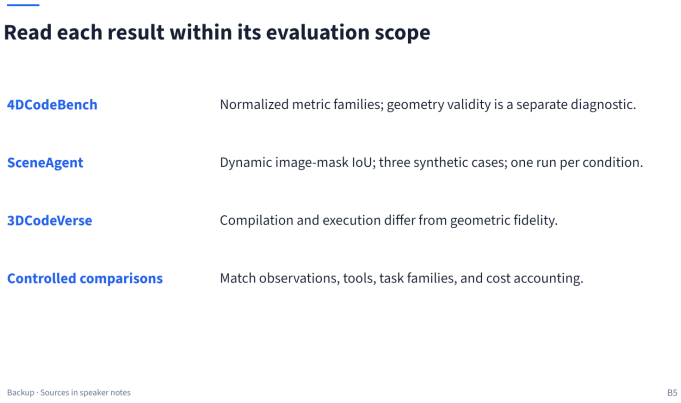
3DCodeEdit (arxiv.org/abs/2610.38231)

4DCodeBench (arxiv.org/abs/2610.03715)

Read each result within its evaluation scope

Backup 5 of 8 - untimed

Available for questions



Speaker notes

These are guardrails for questions about the paper results. First, 4DCodeBench's static and dynamic scores aggregate different normalized metric families; they are not physical error units or fractions of solved scenes. Its mesh-quality checks are separate diagnostics because a simple but inaccurate shape can still be watertight. Second, SceneAgent's headline dynamics gain is image-mask IoU over three synthetic cases, with a single conversion run for each configuration and condition. It is not a direct assessment of recovered physical parameters. Third, compilation and execution percentages in 3DCodeVerse cannot establish that one geometry representation is superior: runtime task sets, API exposure, and training differ, while geometric fidelity is evaluated separately. Finally, all method comparisons should make observations, available tools, retries, and compute explicit, rather than attributing a mixed change to one representation choice.

Sources

Document review, 4 November 2026 (Shared research discussion)

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

SceneAgent, full text (arxiv.org/html/2610.34745v1)

3DCodeVerse (arxiv.org/abs/2610.22341)

The 40-question research map

Backup 6 of 8 - untimed

Available for questions



Speaker notes

The longer collection groups forty questions into ten themes. Its question numbers refer to that collection and should not be confused with the separate twenty-seven-question list discussed at the end of Document review. The map is useful for locating follow-ups, not as a claim of forty novel research contributions. Questions two, eleven, and thirty-eight motivate evaluation through edit behavior; questions seventeen through twenty concern structure, residual detail, and topology changes; questions twenty-nine through thirty-two examine Gaussian intermediates and rule recovery; questions thirty-three through thirty-six address physical mechanisms; question thirty-seven anchors feedback and repair. Language, abstraction discovery, active observation, and learning without unique source programs connect these directions. In each case, prioritize a setting where the evidence can distinguish competing explanations and where a negative outcome identifies a useful limitation.

Sources

Document review, 4 November 2026 (Shared research discussion)

Reading and discussion sources

Backup 7 of 8 - untimed

Available for questions



Speaker notes

The principal paper is 4DCodeBench, with SceneAgent and 3DCodeEdit as companion perspectives. InfiniDream and 3DCodeVerse place them in a broader content-generation and agent-workflow landscape. The research questions concern detail persistence, recoverable dependencies, and executable verification, with the shared discussion providing their context. The editing and correspondence proposals should also be read alongside semantic shape editing with parametric implicit templates, BeCoS, and ParSEL; the rule-recovery proposal should be compared with ShapeCoder. Spectral and Gaussian background is cited on the corresponding slides. The URLs below lead to the paper or project sources, while the shared review records the proposed questions. The figures in the paper sections remain attributed to their original authors and are evidence from those works, not results of the proposed studies.

Sources

4DCodeBench (arxiv.org/abs/2610.03715)

SceneAgent (arxiv.org/abs/2610.34745)

3DCodeEdit (arxiv.org/abs/2610.38231)

3DCodeVerse (arxiv.org/abs/2610.22341)

InfiniDream (arxiv.org/abs/2610.26853)

Document review, 4 November 2026 (Shared research discussion)

Semantic Shape Editing with Parametric Implicit Templates (udaykusupati.github.io/publications/ish/ish)

BeCoS (nafieamrani.github.io/BeCoS)

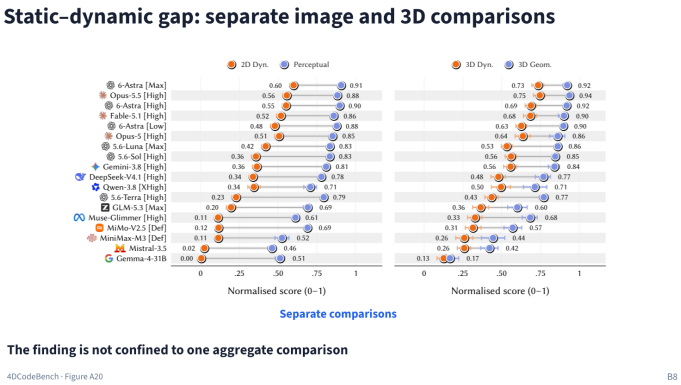
ParSEL (arxiv.org/abs/2405.20319)

ShapeCoder (arxiv.org/abs/2305.05661)

Static-dynamic gap: separate image and 3D comparisons

Backup 8 of 8 - untimed

Available for questions



Speaker notes

Backup for the main static-dynamic gap slide. The main static-dynamic comparison combines two static families and two dynamic families. This additional paper figure separates comparisons in image and three-dimensional spaces. It helps check whether the general observation depends on mixing different sources of evidence. The normalized measures are still different quantities; their numerical separation is not a physical distance or a common-unit error. The value is the recurring pattern across models and across distinct evaluation views. The three-dimensional comparison is especially useful because synthetic scenes expose reference state beyond what is visible in a rendering. Connect this result back to the task definition: producing plausible appearance and a plausible static surface does not ensure that the material follows the correct paths over time.

Sources

4DCodeBench, full text (arxiv.org/html/2610.03715v1)

Document review, 4 November 2026 (Shared research discussion)