BridgeCode: Grammar-Constrained Scan-to-Program Bridge Reconstruction for As-Built-to-As-Designed Feedback

Supplementary Material

A. End-to-End Pipeline and Tensor Shapes

We provide a reproducible, end-to-end specification of BridgeCode, including tensor shapes and the exact conditioning interface between the scan encoder, template retrieval, and the code LLM.

A.1. Input Preprocessing and Normalization

All coordinates are represented in **meters**. We do **not** normalize to a unit cube. Given an input scan $\mathcal{P} \in \mathbb{R}^{N\times 3}$: (i) we center it at the estimated deck centerline midpoint; (ii) apply PCA to canonicalize the longitudinal axis (span direction); (iii) optionally flip the axis to enforce a consistent left-to-right convention using the scan bounding box. We cap points to $N_{\max} = 2 \times 10^5$ by voxel-balanced sampling; for $N < N_{\max}$ we keep all points.

A.2. Encoder Outputs

Sparse voxelization yields a sparse tensor with n_0 active voxels. The encoder produces multi-scale feature sets $\{F_\ell\}_{\ell=1}^{4}$:

$$F_\ell \in \mathbb{R}^{n_\ell \times d_{\text{enc}}}, \quad d_{\text{enc}} = 256,$$

where n_ℓ depends on scan density and voxelization. We additionally compute a global descriptor

$$\mathbf{z} \in \mathbb{R}^{d_{\text{enc}}}$$

via attentive pooling over the highest-level features.

A.3. Template Retrieval and Conditioning Tokens

Each template τ_k is represented by an embedding sequence

$$\phi(\tau_k) \in \mathbb{R}^{L_\tau \times d_{\text{llm}}},$$

projected to the LLM hidden size d_{llm}. We retrieve top-m templates per family and compute soft weights α_k with temperature τ. The final template memory is the weighted set $\{\alpha_k \phi(\tau_k)\}_{k=1}^{m}$. In all experiments we use $m = 8$ and $L_\tau = 1$.

A.4. LLM Interface (Cross-Attention Adapters)

We condition the causal code LLM on two external memories: (i) template tokens (Sec. A.3), (ii) scan tokens obtained by projecting encoder features F_ℓ into d_{llm} and concatenating them into $M = 256$ keys/values. Cross-attention adapters inject these memories at every LLM layer (implementation details in released code).

A.5. Output Representation

The LLM emits an OP/ARG/VAL token sequence $\hat{\mathbf{y}}$ which deterministically parses into an AST and then compiles into executable CadQuery. Continuous values are produced by (bin, residual) prediction (Sec. E).

B. CadQuery DSL and Grammar

This section expands Sec. 3 by detailing the typed DSL, grammar masks, and validity checks used during decoding.

B.1. Vocabulary and Tokenization

The DSL $\mathcal{V}$ is partitioned into **OP** (operation), **ARG** (argument key), and **VAL** (typed value) tokens. Numeric values leverage the codebook+residual head from Sec. 3, annotated with explicit units.

B.2. Validity Constraints

We enforce: positive thicknesses, minimum aspect ratios, fillet radius $\leq$ local feature size, absence of self-intersections (kernel check), deck clearance above roadway, and consistent axis orientation. Grammar/type masks prune illegal continuations; the execution sandbox rejects programs violating kernel or constraint checks.

B.3. Girder Module Example

The listing in Lst. 1 is a human-readable pseudocode. In the actual DSL, repetitions are represented by explicit operators such as `ArrayLinear` (Tab. 4), so that the emitted program remains a flat OP/ARG/VAL sequence with a deterministic AST.

Listing 1. Girder module excerpt

```
# Deck scaffold
Workplane(origin=[0,0,0], normal=[0,0,1])
SketchRect(width=span_len, height=deck_thick)
Extrude(height=deck_thick, both=False)

# Girder array
for i in range(num_girders):
    Workplane(origin=[i*girder_spacing,0,0],
    normal=[0,0,1])
    SketchIShape(web=web_t, flange=flange_t,
    h=girder_h)
    Extrude(height=span_len, both=False,
    dir=[1,0,0])
```

B.4. Additional Program Excerpts

Listing 2. Truss (Warren) excerpt

Table 4. Core DSL operators and typed arguments (meters unless noted).

OP	ARG:type (unit)
Workplane	origin:vec3, normal:vec3
SketchLine	p0:vec2, p1:vec2
SketchArc	p0:vec2, p1:vec2, center:vec2
Constraint	name:str{Coincident,Horizontal,Equal,...}
Extrude	height:float, both:bool
Revolve	angle:deg, axis:vec3
Sweep	path:id, mode:str{cut,add}
Fillet	edges:set, radius:float
Shell	faces:set, thickness:float
Union	a:id, b:id
Cut	a:id, b:id
ArrayLinear	dir:vec3, count:int, pitch:float
Transform	T:mat4
SketchRect	width:float, height:float
SketchIShape	web:float, flange:float, h:float
SketchSpline	ctrl:seq[vec2]
Loft	sections:seq[id], ruled:bool
Mirror	plane:vec3, obj:id
Name	id:id, tag:str

```
Workplane(origin=[0,0,0], normal=[0,0,1])
SketchLine(p0=[0,0], p1=[L,0])       # bottom
    chord
ArrayLinear(dir=[L/(n-1),0,0], count=n,
    pitch=L/(n-1))
SketchLine(p0=[i*dx,0], p1=[(i+1)*dx,h])  #
    diagonals (typed slots)
Extrude(height=member_t, both=False)
```

Listing 3. Arch rib excerpt

```
Workplane(origin=[0,0,0], normal=[0,1,0])
SketchArc(p0=[0,0], p1=[L,0],
    center=[L/2,arch_rise])
Sweep(path=arch_axis_id, mode=add)
Shell(faces=arch_faces, thickness=rib_t)
```

Listing 4. Cable-stayed excerpt

```
# tower
Workplane(origin=tower_base, normal=[0,0,1])
SketchRect(width=tower_w, height=tower_d)
Extrude(height=tower_h, both=False)
# cables (catenary prior during training)
ArrayLinear(dir=[dx,0,0], count=n_cables,
    pitch=dx)
Sweep(path=cable_path_id, mode=add)
```

C. Dataset Generation Details

These details complement Sec. 3 and Sec. 4.1.

Mesh sampling: Poisson-disk sampling proportional to local curvature.

MMS-like: Photogrammetric pipeline with reprojection noise $\mathcal{N}(0, \sigma^2)$ and depth-biased outliers.

ALS-like: Beam divergence θ_b, range noise $\epsilon_r \sim \mathcal{N}(0, \sigma_r^2)$, incidence-based dropout, grazing sparsity.

Occlusions: Procedural vegetation, vehicles, and scaffolding; visibility flags stored per point.

Each sample stores $(\mathcal{P}, \mathbf{y})$ and per-point labels (module ID, template ID, visibility, uncertainty). Splits reserve unseen span-count/length ranges and optionally hold out one family for cross-family evaluation.

We generate $\sim 8 \times 10^4$ training, 10^4 validation, and 10^4 test programs. We report additional diversity statistics: (i) distribution of AST size (nodes) and token length; (ii) parameter coverage per slot (min/median/max); (iii) family-wise frequencies. We remove near-duplicates by hashing the normalized AST (operator + argument-key structure) and rejecting samples whose continuous parameters fall within a small tolerance of an existing program in the same hash bucket. In the released dataset we provide: program hashes, template IDs, and split metadata.

Our splits are **not** random. We reserve disjoint intervals of (span count, span length) so that test bridges fall outside the training range on at least one global degree of freedom. Optionally, we hold out one entire family for cross-family evaluation (reported separately).

D. Template Bank and Retrieval

Templates serve as retrieval keys for the hierarchical decoder (Sec. 3.2.2).

Family	# templates	Avg. AST nodes	Notes
Girder	32	180	3–8 girders, straight axis
Continuous girder	24	220	2–4 spans
Frame	16	160	Rigid frames
Truss	20	260	Pratt/Warren/K variants
Arch	20	240	1–4 ribs
Cable-stayed	16	300	A- and P-towers

Table 5. Template statistics per family.

Component	**Configuration**
Encoder	SparseConv (Minkowski) + PointNeXt blocks; 4 stages; width 256; strides 1/2/2/2
Code LLM backbone	Qwen2-1.5B (public configuration); causal decoder for OP/ARG/VAL
LLM conditioning	(i) template tokens (top-m) via cross-attention adapters; (ii) multi-scale scan tokens via cross-attention adapters
Tokenization	OP/ARG/VAL vocabularies; BOS/EOS; typed slots for $\{L_s, n_g, g, t_d, \dots\}$
Numeric head	64 log-spaced bins on $[10^{-3}, 10^{2}]$ + Huber residual (δ=0.05)
Grammar masks	finite-state frontier + type/arity/namespace masks + kernel capability mask
Beam search	B=5 (train), B=10 (test); length penalty α=0.6; temperature T annealed $0.7 \to 0.3$
Optimizer	AdamW, lr 3×10^{-4}, wd 1×10^{-2}, β=$(0.9, 0.999)$
Schedule	cosine decay 300k steps, warmup 10k, grad clip 1.0, AMP O2
SCST mix	10–20% (linearly ramped), greedy baseline, reward Eq. (10)
Sandbox	CadQuery+OCC in isolated workers; timeout 150 ms/beam; prefix cache; 8–32 workers
Batching	batch size 4 (effective 32 via accumulation); max tokens 512; max points 2×10^{5}

Table 6. Training and architecture settings.

Encoder descriptor $\mathbf{z}$ (Sec. 3.2.1) is compared with template embeddings $\phi(\tau_k)$; top-m templates condition the decoder via cross-attention.

E. Model, Training, and Sandbox

This section expands Sec. 3.

E.1. Architectural Notes

Encoder. Sparse PointNeXt-style backbone with residual blocks; multi-scale features $\{F_1, \dots, F_4\}$ feed decoder cross-attention. A global descriptor z supports template retrieval.

Decoder. Causal Transformer emits OP/ARG/VAL sequence; grammar/type masks apply at each step to prevent invalid continuations, significantly improving Exec@1 (Tab. 3).

Numeric Prediction (definition of symbols). For a continuous argument, we predict a log-spaced bin index $k \in \{1, \dots, K\}$ and a residual $r \in \mathbb{R}$. Let $\mathrm{bin\_center}_k$ be the center value of the k-th bin in log-space. The final prediction is

$$\hat{v} = \mathrm{bin\_center}_k \exp(r).$$

Ground-truth values $v^\star$ are assigned to a bin $k^\star$ with a residual $r^\star = \log(v^\star / \mathrm{bin\_center}_{k^\star})$. We train with

$$\mathcal{L}_{\text{val}} = \mathrm{CE}(k^\star, k) + \mathrm{Huber}(r, r^\star).$$

Data Augmentation. Random yaw ($\pm10^\circ$), scaling (0.95–1.05), 1 cm Gaussian jitter, up to 20% dropout, synthetic occlusion slabs, sensor mixing. PCA aligns the longitudinal axis.

E.2. Curriculum

Three stages gradually relax constraints:

- **Phase 1 (scaffold):** axis/spans/deck only, teacher forcing p_{TF}=1, strong masks.
- **Phase 2 (modules):** add girders/diaphragms/piers, enable cross-attention, $p_{\text{TF}} \downarrow 0.7$, beam B=3.
- **Phase 3 (finish):** enable finishing ops (fillet/shell/booleans), $p_{\text{TF}} \downarrow 0.5$, beam B=5.

E.3. Training Schedule (SFT → SCST Interleaving)

We train in two phases. **Phase A (SFT warm-up):** we optimize supervised objectives only for the first $S_0 = 25{,}000$ steps (no sandbox execution). **Phase B (interleaved SCST):**

Table 7. Which objectives are used in SFT vs. SCST.

Loss / term	SFT (teacher forcing)	SCST reward
$\mathcal{L}_{\text{tok}}$ (OP/ARG/VAL CE)	✓	–
$\mathcal{L}_{\text{val}}$ (numeric bin+residual)	✓	–
$\mathcal{L}_{\text{part}}$ (slot/module supervision)	✓	–
$\mathcal{L}_{\text{geo}}$ (Chamfer/NC/IoU)	optional†	✓
Exec failure penalty ($\mathbf{1}[\texttt{fail}]$)	optional‡	✓
Length penalty ($\lvert\hat{\mathbf{y}}\rvert$)	–	✓
Structural priors (catenary/per/sym/clear)	optional†	✓

†If used in SFT, applied only after the exec warm-up when sandbox is enabled.
‡Used only when a program can be executed in the sandbox.

after warm-up, we interleave SCST updates with probability $p_{\text{SCST}}(s)$ linearly ramped from 0.10 to 0.20 over $S_1 = 150{,}000$ steps:

$$p_{\text{SCST}}(s) = 0.10 + 0.10 \cdot \text{clip}\Big(\frac{s - S_0}{S_1}, 0, 1\Big).$$

On SFT steps, we apply teacher forcing; on SCST steps, we sample with beam search and execute in the sandbox. Unless otherwise stated, we update **all** trainable parameters (encoder, retrieval projections/adapters, and the LLM adapters; the LLM backbone is `fine-tuned`).

E.4. Training Objectives

Supervised XE. Cross-entropy on OP/ARG/VAL, numeric loss $\mathcal{L}_{\text{val}}$, and part-level slot supervision $\mathcal{L}_{\text{part}}$.

Executable-in-the-loop. Self-critical sequence training (SCST) interleaved with sandbox execution (Sec. 3).

$$\mathcal{R} = -\mathcal{L}_{\text{geo}}(\hat{\mathcal{M}}, \mathcal{P}) - \lambda_{\text{exec}}\mathbf{1}[\texttt{fail}] - \lambda_{\text{len}}|\hat{\mathbf{y}}|. \tag{10}$$

Gradients do not flow through the sandbox; prefix execution caches amortize cost. Failures influence rewards via λ_{exec} and length penalty.

Practical Considerations. Decoder dropout 0.1, encoder stochastic depth 0.1, label smoothing 0.05, residual clipping ± 0.3, exec warm-up (sandbox enabled after 25k steps).

Hardware. Training on 4×H200 (80 GB). Synthetic-only training ≈36 h; adding real-scan fine-tuning ≈44 h.

F. Metric Definitions

For completeness we list exact formulations used in Sec. 4. $\hat{\mathcal{Q}}$ denotes points sampled from the predicted solid $\hat{\mathcal{M}}$; $\mathcal{P}$ is the input scan.

Chamfer-L1.

$$\text{CD}_{\ell_1}(\hat{\mathcal{Q}}, \mathcal{P}) = \frac{1}{|\hat{\mathcal{Q}}|}\sum_{\hat{\mathbf{q}}} \min_{\mathbf{p}} \|\hat{\mathbf{q}} - \mathbf{p}\|_1 + \frac{1}{|\mathcal{P}|}\sum_{\mathbf{p}} \min_{\hat{\mathbf{q}}} \|\mathbf{p} - \hat{\mathbf{q}}\|_1. \tag{11}$$

Earth Mover's Distance. Balanced OT with entropic relaxation:

$$\min_{\mathbf{T}\geq 0} \ \langle C, \mathbf{T}\rangle + \varepsilon \sum_{i,j} \mathbf{T}_{ij}\big(\log \mathbf{T}_{ij} - 1\big) \tag{12}$$
$$\text{s.t. } \mathbf{T}\mathbf{1} = \tfrac{1}{n}\mathbf{1}, \qquad \mathbf{T}^\top \mathbf{1} = \tfrac{1}{n}\mathbf{1}.$$

Normal Consistency.

$$\text{NC}(\hat{\mathcal{Q}}, \mathcal{P}) = \tfrac{1}{2}\Big[\frac{1}{|\hat{\mathcal{Q}}|}\sum_{\hat{\mathbf{q}}} \big|\hat{\mathbf{n}}(\hat{\mathbf{q}})^\top \mathbf{n}(\text{NN}_{\mathcal{P}}(\hat{\mathbf{q}}))\big| + \frac{1}{|\mathcal{P}|}\sum_{\mathbf{p}} \big|\hat{\mathbf{n}}(\text{NN}_{\hat{\mathcal{Q}}}(\mathbf{p}))^\top \mathbf{n}(\mathbf{p})\big|\Big]. \tag{13}$$

Voxel Part IoU/F1. Majority-vote labels on voxel grid $\mathcal{V}_v$ yield per-class IoU_c and macro averages.

AST Edit Distance.

$$\text{AST-ED}(\hat{T}, T^\star) = \frac{\text{TED}(\hat{T}, T^\star)}{|\mathcal{N}(T^\star)|}. \tag{14}$$

Slot MAE.

$$\text{MAE}_{\text{scalar}} = \frac{1}{|\mathcal{S}|}\sum_{s} \alpha_s\,|\hat{s} - s|, \tag{15}$$

$$\text{MAE}_{\text{span}} = \min_{\pi} \frac{1}{|\mathcal{L}|}\sum_{i} |\hat{L}_{\pi(i)} - L_i|. \tag{16}$$

Overall error: $\frac{1}{2}\big(\text{MAE}_{\text{scalar}} + \text{MAE}_{\text{span}}\big)$.

Family	CD↓	IoU↑	Exec@1↑	Edit-Exec↑
Girder	0.028	0.88	0.95	0.90
Continuous girder	0.030	0.87	0.93	0.88
Frame	0.034	0.84	0.91	0.85
Truss	0.038	0.81	0.88	0.82
Arch	0.036	0.83	0.89	0.83
Cable-stayed	0.041	0.79	0.85	0.78

Table 8. Family-specific metrics (synthetic test).

Exec@B.

$$\text{Exec@}B = \frac{1}{B}\sum_{j=1}^{B} \text{exec}(\hat{\mathbf{y}}^{(j)}; \tau). \tag{17}$$

Real-scan Tolerances. Slot accuracy within engineering tolerances Δ_s (e.g., ± 1.0 m span, ± 0.10 m spacing).

G. Extended Experiments

G.1. Family-wise and Conditioned Results

We provide per-family breakdowns (Tab. 8) and stratifications by occlusion rate, sensor, and point density in the accompanying supplementary tables.

G.2. Failure Modes and Edit Stability

We also analyze edit stability beyond executability. Given a predicted program with named member identifiers (via `Name` in the DSL), we define *ID preservation* under an edit as the fraction of identifiers whose tags remain present and correctly mapped after re-execution. We find that topology-preserving edits (e.g., span-length or girder-count changes) preserve most IDs, while edits that alter topology (e.g., switching families or introducing strong axis curvature) can require remapping; we therefore keep IDs stable by (a) hierarchical naming (span/module/member) and (b) deterministic ordering of repeated arrays.

H. Notes on Baseline Fairness

All methods are evaluated on *the same* input point clouds and the same geometric metrics (CD/NC/IoU/EMD). BridgeCode uses structural priors *only* as training regularizers and decoding constraints; at evaluation time, geometric metrics are computed solely from the predicted solids without using priors. To isolate the effect of priors, we additionally report ablations that remove (i) structural priors and (ii) grammar/type masks, showing that gains are not solely attributable to handcrafted constraints.

Setting	Encode (ms)	Decode+Exec (ms)
MMS, N=50k	24.6	158.3
ALS, N=200k	81.7	302.5

Table 9. Runtime summary.

I. Runtime and Scaling

Complexity. Encoding $\mathcal{O}(N)$ (sparse), decoding $\mathcal{O}(T)$ tokens with beam width B, execution amortized via prefix cache.

Throughput. Measured per sample with B=10.

J. Real Data

This section complements Sec. 4 by documenting the real-world deployment pipeline. PLATEAU provides *real-world city-scale bridge geometry* (CityGML) rather than raw sensor scans. We therefore use PLATEAU to test generalization to *out-of-distribution, imperfect real-world geometry* (varying LoD, missing members, merged structures), while generating LiDAR-like point clouds through controlled sampling and simulation for reproducible evaluation. We explicitly do **not** claim this is a substitute for true terrestrial LiDAR scans; instead, it complements synthetic tests by exposing modeling artifacts and LoD-induced sparsity.

J.1. Source Data and Conversion

We download CityGML tiles for Tokyo, Osaka, and Nagoya from Japan's PROJECT PLATEAU, filter `Bridge` objects, and convert to meshes using the official PLATEAU GIS converter. Meshes are de-aggregated, metric-scaled, centered, and oriented via PCA to estimate the longitudinal axis.

We sample point clouds by (i) Poisson-disk sampling of mesh surfaces ($N \approx$ 50k–200k) and (ii) LiDAR simulation with beam divergence, range noise $\mathcal{N}(0, \sigma_r^2)$, and grazing-angle sparsity to mirror synthetic sensors.

The synthetic-only BridgeCode checkpoint (no PLATEAU fine-tuning) is decoded with beam B=10 under grammar/type masks. We report Chamfer, normal consistency, Exec@1/Exec@B. AST-ED is undefined without ground-truth programs.

Pseudo-ground-truth spans derive from deck projections along the longitudinal axis; deck width from cross-sections; girder count only when underside members are present. We evaluate tolerance-banded accuracy (span ± 1.0 m, width ± 0.20 m).

J.2. Results

Errors stem from semantic sparsity (LoD1/2 lack members), geometric abstraction, and context bias (merged ramps).

Metric	CD↓ [m]	NC↑	Exec@1↑	Slot tol-Acc↑
All bridges	0.045 [0.039–0.056]	0.93 [0.91–0.95]	0.88 [0.84–0.92]	–
Span length (± 1.0 m)	–	–	–	0.84 [0.79–0.89]
Deck width (± 0.20 m)	–	–	–	0.81 [0.74–0.87]
Girder count (exact)[†]	–	–	–	0.57 [0.46–0.66]

[†]Evaluated only when underside members are discernible.

Table 10. PLATEAU evaluation (median [IQR], LiDAR-like scans, $N \approx 100$k).

Template retrieval mitigates span drift; conservative masks prevent invalid code when underside detail is absent. Round-trip STEP/IFC export succeeds on representative cases (cf. Sec. 4).

K. Dataset and Code Availability

If accepted, we will release: (i) the procedural generator and DSL specification; (ii) the full synthetic paired dataset (programs, simulated scans, labels, split metadata); (iii) evaluation scripts and the CadQuery sandbox wrapper. We will also release a lightweight inference checkpoint and instructions for reproducing all tables. We will include licensing terms and any necessary restrictions (e.g., third-party assets) in the final release package.