Language-Grounded Understanding of 3D Shapes via Foundation Models

Bingchen Gong

Joint work with S. Hadgi, D. Gomez, A. Hamdi, A. Eldesokey, A. Abdelreheem, R. Sundararaman, E. Pierson, L. Li, P. Wonka, M. Ovsjanikov

IMAGINE Lab Seminar • École des Ponts ParisTech

January 28, 2026

Outline

Outline

Why Localized 3D Understanding?

Goal: Understand 3D shapes at the **sub-shape level**

- Keypoint detection (e.g. wing tip, leg joint)
- Part segmentation (e.g. wing, tail, fuselage)
- Semantic labeling without manual annotation
- Language-driven queries ("show me the wing")

Traditional approach:

- Expensive per-category 3D annotations
- Category-specific models
- Poor generalization to new shapes

Key Challenge

How can we achieve **fine-grained 3D understanding** without 3D supervision?

Our Insight

Leverage **Multi-Modal LLMs** and **2D foundation models** to bridge the gap from 2D→3D.

Contributions at a Glance

ZeroKey (ICCV 2025)

- First zero-shot 3D keypoint detector
- No 3D annotations needed
- **79.43%** IoU@0.10 (3× baselines)

PatchAlign3D (arXiv 2025)

- SOTA zero-shot 3D part segmenter
- Single feed-forward pass (0.4s)
- **56.9%** mIoU on ShapeNetPart

Common thread: Language grounding + 2D foundation models → localized 3D understanding **without 3D supervision**

Background: Foundation Models We Build On

Vision-Language Models

- **CLIP / OpenCLIP:** Contrastive image-text pre-training; shared embedding space for images and text
- **SigLIP:** Sigmoid-based variant — per-pair contrastive loss (no softmax)
- **GPT-4o:** Multimodal LLM with image understanding and reasoning

Visual Feature Extractors

- **DINOv2:** Self-supervised ViT producing dense local features; excels at spatial correspondence
- **Molmo:** MLLM trained with **pixel-level point annotations**; can output precise (x, y) coordinates

Key insight: These 2D models encode rich geometric and semantic knowledge.
Our work **transfers** this knowledge to 3D without 3D-specific supervision.

Two Complementary Approaches

ZeroKey (ICCV 2025)

- **Task:** Zero-shot 3D keypoint detection
- **Key idea:** Exploit pixel-level MLLM annotations across multi-view renderings
- **Pipeline:** GPT-4o → Molmo → back-project → HDBSCAN
- **Result:** Competitive with supervised methods; **no 3D annotations needed**

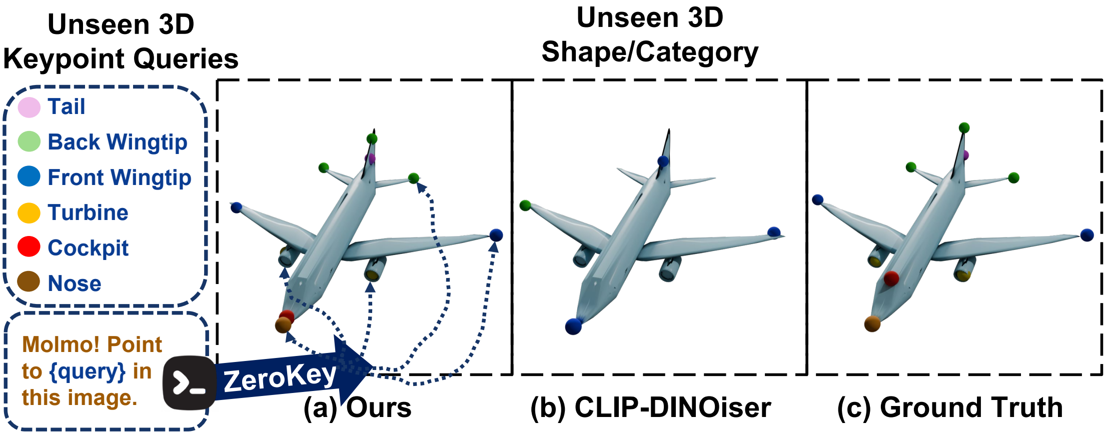
PatchAlign3D (arXiv 2025)

- **Task:** Zero-shot 3D part segmentation
- **Key idea:** Encoder-only 3D model with language-aligned patch features
- **Training:** DINOv2 distillation → SigLIP text alignment
- **Result:** SOTA across benchmarks; **single feed-forward pass**

Common theme: Foundation models + language grounding ⟶ localized 3D understanding

Outline

ZeroKey — Visual Overview



Without ground truth labels, ZeroKey leverages **pixel-level MLLM reasoning** to extract and name salient 3D keypoints — achieving competitive performance with supervised methods.

ZeroKey — Problem Setup

3D Keypoint Detection

- Given a 3D mesh, detect **semantically meaningful** points
- Schelling points: game-theoretic focal points people select independently
- E.g. wing tips, wheel centers, chair legs
- Traditionally requires dense per-point annotations

Zero-shot setting:

- No 3D keypoint labels at training time
- No category-specific fine-tuning
- Must *both* localize *and* name keypoints

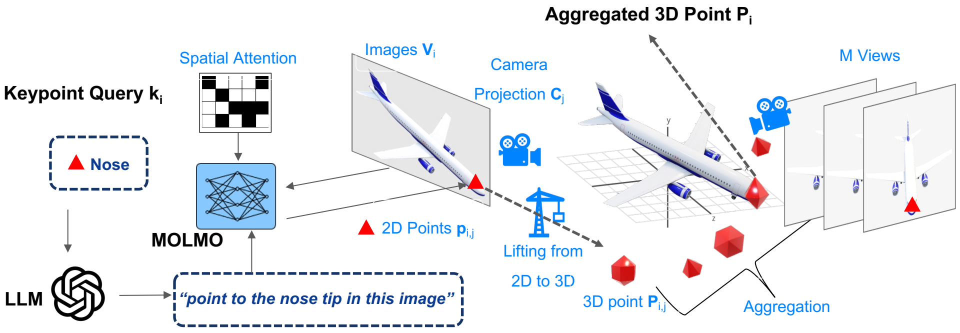
First-of-its-kind

ZeroKey is the first method to show that **pixel-level MLLM annotations** can be exploited for 3D keypoint detection *without any ground truth*.

Evaluation:

- KeypointNet benchmark
- Categories: airplane, chair, table
- Metric: IoU at geodesic distance thresholds

ZeroKey — Pipeline Overview



Stage 1: 2D Detection
Molmo localizes each named keypoint across N views

$$\mathbf{p}_{i,j} = \text{Molmo}(\mathbf{V}_j, k_i)$$

Stage 2: Soft Voting
Gaussian kernel weights for back-projection

$$\mathbf{w}_{i,j} \propto \sum \exp\left(-\frac{\|\mathbf{p}_{i,j}-\mathbf{p}\|^2}{2\sigma^2}\right)$$

Stage 3: Clustering
HDBSCAN aggregation with mutual reachability

$$d_{\text{mreach}}(a,b) = \max\{d_k(a),\ d_k(b),\ \|a-b\|\}$$

ZeroKey — Method: 2D Candidates

1. Text Candidate Generation (GPT-4o)

- **Input:** Rendered views of the shape
- **Prompt:** "List possible salient key points (in text)."
- **Output:** $\mathcal{K} = \{k_1, \ldots, k_N\}$ (e.g. "nose", "wing tip")
- Typically generates 6–10 keypoint names per shape

2. 2D Localization (Molmo)

- **Prompt:** `"Point to the {k_i} in this image."`
- **Output:** 2D coordinates $\mathbf{p}_{i,j}$ for each view $\mathbf{V}_j$
- Leverages Molmo's **point-level supervision** for precise localization
- Process across $M = 26$ views (default)

ZeroKey — Method: 3D Aggregation

3. Soft Voting Back-Projection

Stabilize ray casting with $h \times h$ patch refinement:

- Back-project patch $\mathbb{S}_{i,j}$ centered at 2D prediction $\mathbf{p}_{i,j}$
- Assign **Gaussian soft-voting weights** $\mathbf{W}_{i,j}$:

$$\mathbf{W}_{i,j} = \sum_{\mathbf{p} \in \mathbf{N}_{i,j}} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\|\mathbf{p}_{i,j} - \mathbf{p}\|^2}{2\sigma^2}\right)$$

- Higher weight for points closer to patch center
- Mitigates noise from sharp angular intersections
- $\sigma = h/3$ in implementation

4. Weighted HDBSCAN

Cluster 3D candidates $\mathcal{P}_i$:

$$d_{\text{mreach}}(a, b) = \max\{\text{core}_k(a), \text{core}_k(b), \|a - b\|\}$$

- Incorporate weights $\mathbf{W}_{i,j}$
- Filter outliers (Molmo noise)
- minPts $k = 10$ (fixed)
- **Output:** Centroid of densest cluster

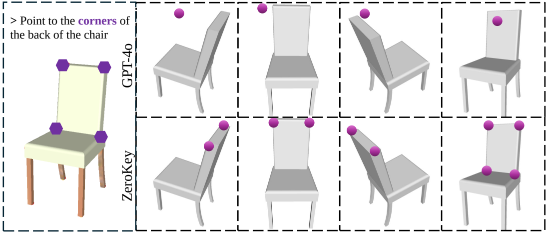
ZeroKey — Why Molmo?

Molmo is a recent MLLM trained with **pixel-level pointing data**:

- Can output precise (x, y) coordinates
- Understands natural-language spatial references
- Trained on human point annotations

Why not GPT-4o directly?

- GPT-4o reasons about images but outputs *bounding boxes*, not points
- IoU@0.10: GPT-4o $= 20.73\%$ vs. Molmo $=$ **79.43%**



(a) Ours Predicted Keypoint (b) Rendered Multiple Views

GPT-4o fails to precisely locate keypoints; Molmo succeeds due to pixel-level training.

Key insight: Point-level training is essential — scaling alone does not solve localization.

ZeroKey — Quantitative Results

KeypointNet benchmark (airplane, chair, table)

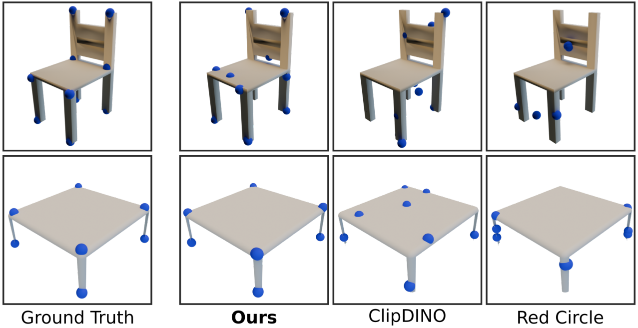
Method	**IoU (%) @ geo. dist.**		
	0.01	0.05	0.10
Supervised / Few-shot:			
UKPGAN (supervised)	6.54	26.55	46.49
FSKD (few-shot)	7.94	31.14	57.03
B2-3D (few-shot)	20.29	57.72	70.57
Zero-shot (ours):			
RedCircle	0.34	3.05	18.50
GPT-4o	0.48	6.04	20.73
CLIP-DINOiser	1.41	9.80	25.56
StablePoints	5.80	19.91	38.22
ZeroKey	**13.16**	**56.60**	**79.43**

Key Takeaways

- **79.43%** IoU@0.10 — **surpasses** supervised methods at larger thresholds
- **Zero-shot**: no 3D keypoint annotations
- **3**× improvement over CLIP-DINOiser

Note: Drop at small thresholds expected due to semantic (vs. geometric) focus.

ZeroKey — Comparison with Baselines



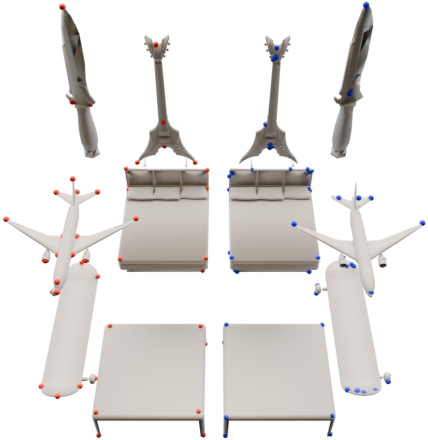
Visual Comparison:

- **CLIP-DINOiser:** Identifies prominent regions but fails to localize precisely
- **RedCircle:** Random sampling with CLIP similarity — noisy results
- **ZeroKey:** Precise localization according to text prompt

Key Difference

ZeroKey uses **point-specific prompts** + Molmo's pixel-level training for accurate localization.

ZeroKey — Qualitative Results

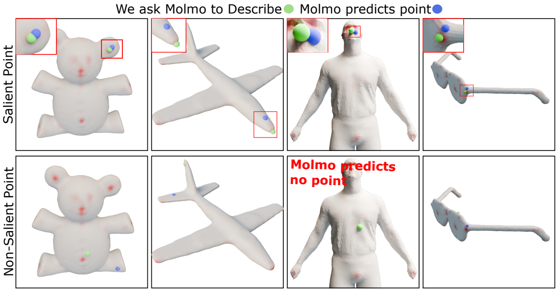


Observations:

- **Strong:** Distinctive, nameable parts (e.g., wing tips, wheel centers).
- **Weak:** Arbitrary surface points, symmetric duplicates.
- **Insight:** Detection quality correlates with how **nameable** a keypoint is.

ZeroKey — Analysis: Schelling Points & Describability

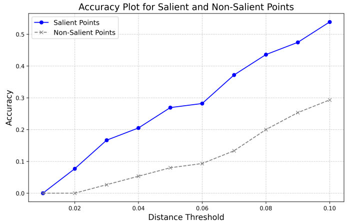
Schelling Points: Focal points people select independently due to prominence (game theory).



Ask Molmo to describe green point → use description to retrieve via ZeroKey (blue).

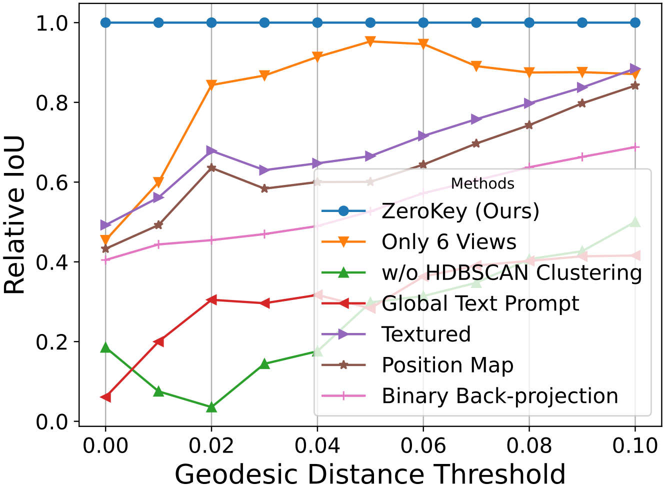
Key Findings:

- **Salient points** (semantically meaningful) are retrieved with much higher accuracy
- **Non-salient points:** ZeroKey may find similar parts or fail entirely
- Confirms: **describability** ≈ **detectability**



Salient vs. non-salient retrieval accuracy across distance thresholds.

ZeroKey — Ablation: Model Configurations



Impact of Modifications:

- **Original method** (blue)
- Global Text prompt (red)
- Alternative renderings (orange, purple, brown)
- No HDBSCAN (green)

This comparison shows the contribution of each component to overall performance.

ZeroKey — Ablation: Aggregation & Rendering

Multi-View Aggregation

6 views 26 views 46 views

- More views → more keypoints detected
- 6 views achieves **80%** of full performance
- Prompt: "corner of the table"

Key Findings

- **Clustering:** Direct averaging fails; **HDBSCAN is essential**
- **Soft Voting:** Gaussian weights outperform binary (weight=1)
- **Rendering:** Pointmap colors / mesh textures don't help (out-of-distribution for Molmo)

Robustness

Method works with **simple shaded renderings** — no special preprocessing required.

ZeroKey — Summary

Key Contributions:

1. **First zero-shot 3D keypoint detector** using MLLMs
2. Language grounding enables **both localization and naming**
3. **79.43%** IoU@0.10 — surpasses supervised at larger thresholds
4. Key ingredients: Molmo + HDBSCAN + multi-view aggregation

Limitations

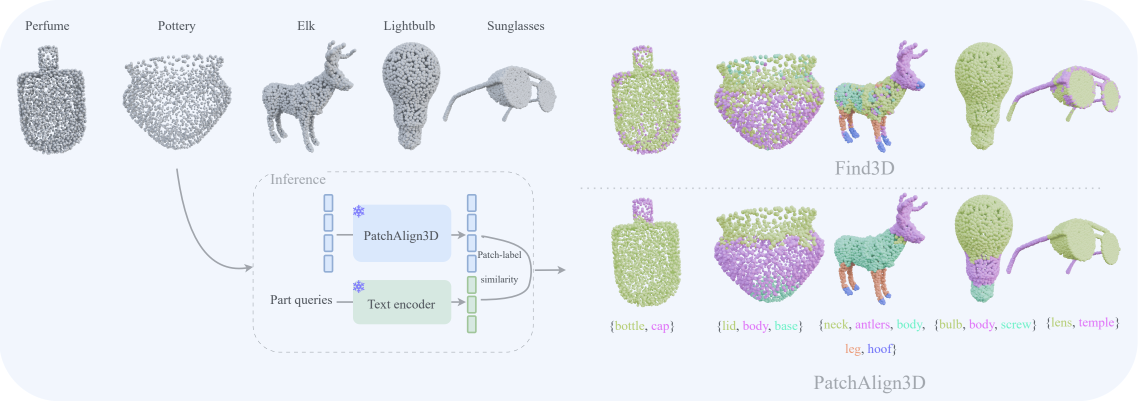
- **Fine-grained:** Lower accuracy at small distance thresholds (semantic vs. geometric focus)
- **Speed:** Requires multi-view rendering + MLLM inference per view
- **Symmetric parts:** May detect one instance of repeated keypoints

But what if we want dense part-level features instead of sparse keypoints?

$\Longrightarrow$ **PatchAlign3D**

Outline

PatchAlign3D — Visual Overview



An **encoder-only** 3D model producing language-aligned patch features — enabling zero-shot part segmentation in a **single feed-forward pass** without multi-view rendering at test time.

PatchAlign3D — Problem Setup

Zero-shot 3D Part Segmentation

- Given a point cloud + text queries (part names), segment the shape
- No test-time category-specific training
- Must generalize across diverse object types

Prior approaches (e.g. Find3D, COPS, SATR):

- Render multiple views at inference
- Run 2D foundation model per view
- Fuse predictions back to 3D
- Find3D: feed-forward but limited (**23.3%** mIoU)

→ **Slow** (>100s for SATR) and prompt-sensitive

PatchAlign3D Goal

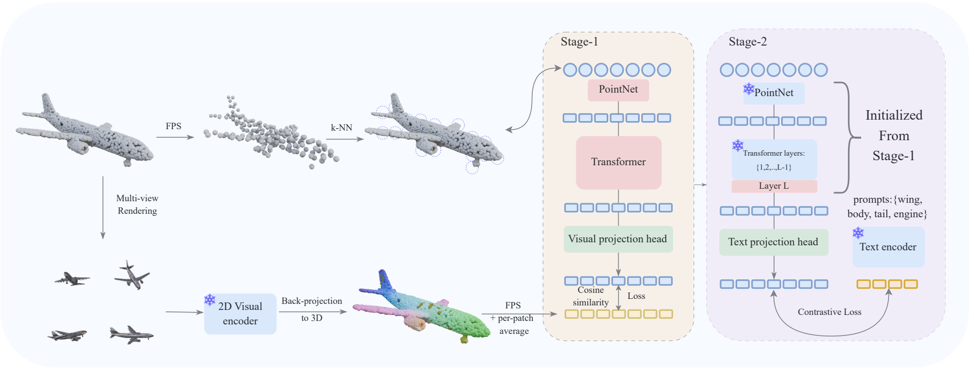
An **encoder-only** 3D model that:

- Operates in a **single feed-forward pass**
- Produces **language-aligned** patch features
- Achieves **SOTA** zero-shot part segmentation

Key Insight

Patch-level aggregation averages out annotation noise — more robust than point-level learning.

PatchAlign3D — Architecture



- **Input:** 2048 points (XYZ only)
- **Patches:** G=128 patches × 32 points
- **Encoder:** 12-layer transformer
- **Stage 1:** DINOv2 feature distillation
- **Stage 2:** Text-patch contrastive alignment
- **Inference:** $s_{i,j} = \frac{1}{\tau}\langle \mathbf{z}_i, \mathbf{t}_j \rangle + b$

PatchAlign3D — Loss Functions

Stage 1: Cosine-Similarity Regression

Align 3D patch features $\mathbf{f}_i$ with back-projected DINOv2 features $\mathbf{d}_i$:

$$\mathcal{L}_{2D} = \frac{1}{G}\sum_{i=1}^{G}\left(1 - \frac{\mathbf{f}_i \cdot \mathbf{d}_i}{\|\mathbf{f}_i\|\,\|\mathbf{d}_i\|}\right)$$

- $G = 128$ patches per shape
- DINOv2 features averaged across visible views
- Trains *all* transformer layers

Stage 2: SigLIP Contrastive Loss

Align patch features $\mathbf{z}_i$ with text embeddings $\mathbf{t}_j$:

$$\mathcal{L}_{\text{text}} = -\sum_{i,j}\left[y_{i,j}\log\sigma(s_{i,j}) + (1-y_{i,j})\log\sigma(-s_{i,j})\right]$$

- $s_{i,j} = \frac{1}{\tau}\langle \mathbf{z}_i, \mathbf{t}_j\rangle + b$ (learnable τ, b)
- **Fractional labels** $y_{i,j} \in [0, 1]$
- Freeze first 11 layers; train last block + proj. head

Why SigLIP over softmax? Sigmoid operates per-pair → naturally handles **multiple positive parts** per shape.

PatchAlign3D — Two-Stage Pre-Training

Stage 1: 2D→3D Feature Distillation

Teacher: DINOv2 (frozen)
Target: Transfer dense visual priors to 3D

1. Render N views of each shape
2. Extract dense DINOv2 features per view
3. Back-project to 3D: $\mathbf{d}(x) = \frac{1}{|V(x)|} \sum_r \mathbf{F}_r(u_r(x), v_r(x))$
4. Train with cosine-similarity regression

→ *3D patches capture local visual structure*

Stage 2: Contrastive Text Alignment

Freeze: Early 11 transformer layers
Train: Last block + projection head

1. Find3D annotations (>2M parts, 761 categories)
2. **SigLIP** contrastive loss
3. Fractional labels $y_{i,j} \in [0, 1]$ for soft matching
4. Handles noisy/overlapping annotations

→ *Patch features aligned with part-level text*

Why two stages? Joint training: 50.2% → Two-stage: **56.9%** mIoU

PatchAlign3D — Training Data

Based on Find3D data engine:

- **32,052** Objaverse shapes (28,827 train / 3,225 val)
- **761** object categories
- >**2 million** part annotations

Annotation pipeline (fully automatic):

1. 10 multi-view renderings per shape
2. SAM generates 2D part masks
3. Gemini 1.5 VLM assigns single-word part names
4. Back-project masks to point cloud

Key Advantage

- Same training data as Find3D
- But PatchAlign3D distills this into a **feed-forward encoder**
- No multi-view rendering at test time

Training details:

- 100 epochs per stage, batch size 32
- AdamW optimizer, lr 3×10^{-4}
- Input: XYZ coordinates only

PatchAlign3D — Main Results: ShapeNetPart

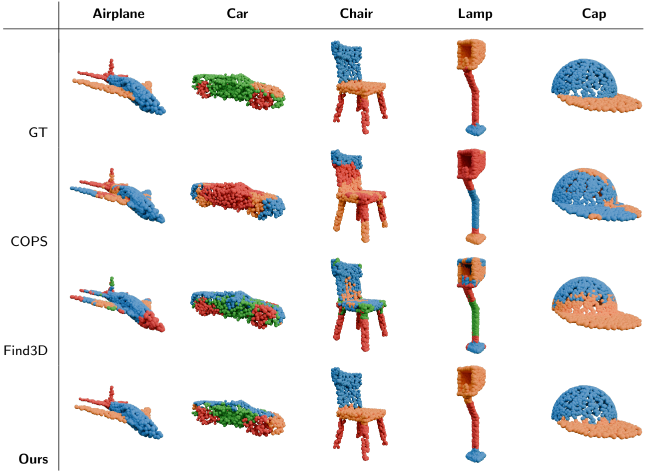
Zero-shot Part Segmentation on ShapeNetPart

Method	mIoU (%)	cIoU (%)	Type
PointCLIPv2	16.1	16.2	CLIP-based
SATR	32.8	36.3	Rendering (mesh)
Find3D	23.3	23.9	Feed-forward 3D
COPS	25.6	32.2	DINOv2 + multi-view
PatchAlign3D	**56.9**	**53.1**	**Feed-forward 3D**

+31.3% mIoU over COPS
+33.6% mIoU over Find3D

Consistent gains in **15 out of 16** categories
at **0.4s** inference (vs. 111s SATR)

PatchAlign3D — Qualitative Results: ShapeNetPart



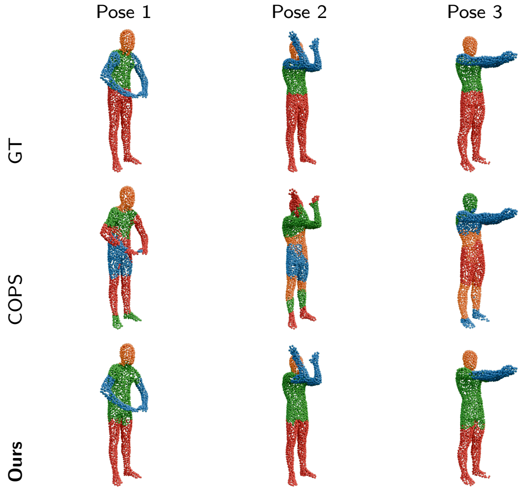
Zero-shot Segmentation

PatchAlign3D produces **sharper, more coherent** part boundaries.

- COPS: noisy, inconsistent
- Find3D: blurry boundaries
- **Ours**: clean parts

PatchAlign3D — Qualitative Results: FAUST Humans

Non-Rigid Human Body Segmentation



FAUST Results:

- **67.8%** mIoU (vs. 30.4% COPS)
- **+37.4%** improvement
- Parts: arm, head, leg, torso

Key Observation

PatchAlign3D remains **stable under pose variation**, while COPS degrades on non-rigid deformations.

PatchAlign3D — Results Across Benchmarks

	PatchAlign3D		Best Baseline	
Benchmark	mIoU	cIoU	mIoU	Method
ShapeNetPart	**56.9**	**53.1**	32.8	SATR
PartNetE	**41.4**	**42.2**	27.0	COPS
FAUST (humans)	**67.8**	—	30.4	COPS
ScanObjectNN	**22.7**	**25.3**	18.8	COPS
Objaverse (seen)	**37.5**	—	28.9	Find3D
Objaverse (unseen)	**35.6**	—	34.6	Find3D

Highlights

- **FAUST: 67.8%** mIoU on non-rigid human body segmentation (+37.4 over COPS)
- **ScanObjectNN:** Robust to **real-world noise** and partial scans
- **Unseen categories:** Strong generalization (35.6% vs. 34.6% Find3D)

PatchAlign3D — Inference Speed

Method	Time (s/shape)	Requires Multi-View?
SATR	111.0	Yes (mesh rendering)
COPS	1.38	Yes (DINOv2 per view)
Find3D	0.4	No
PatchAlign3D	**0.4**	**No**

Why is PatchAlign3D fast?

- Single feed-forward pass through 3D encoder
- No rendering pipeline at test time
- Text embeddings can be pre-computed

Practical Impact

- **$\sim$275$\times$ faster** than SATR
- Enables real-time / large-scale applications
- Point-cloud input (no mesh needed)

PatchAlign3D — Ablation Studies

Training Strategy (ShapeNetPart)

Strategy	mIoU
Stage 2 only	50.5
Joint training	50.2
Two-stage	**56.9 (+6.4)**

Text Encoder

Encoder	mIoU
SigLIP	46.4
Gemma-2-9B-it	54.8
OpenCLIP bigG	**56.9 (+2.1)**

2D Feature Encoder

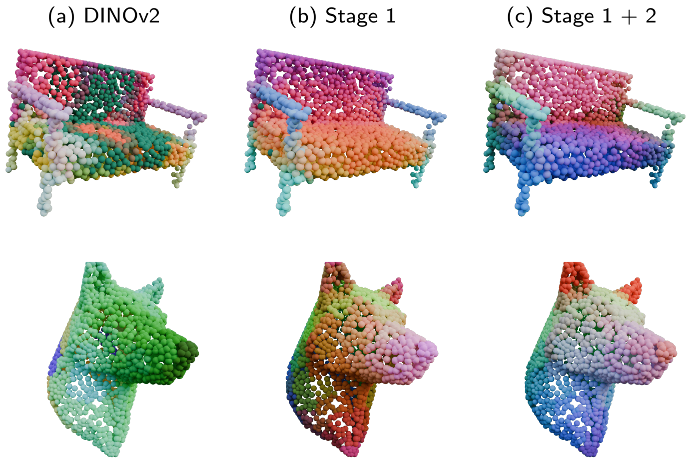
Encoder	mIoU
DINOv3	46.5
OpenCLIP bigG	49.3
DINOv1	51.8
DINOv2	**56.9 (+5.1)**

Freezing Strategy (Stage 2)

Strategy	mIoU
Full fine-tuning	49.4
Last 2 blocks	55.7
Last block	**56.9 (+1.2)**

Takeaways: Two-stage training essential • DINOv2 for vision, OpenCLIP for text • Fine-tune last block only

PatchAlign3D — Feature Evolution Across Stages



Features visualized via PCA to RGB.

Observations:

- **DINOv2:** Back-projected features are noisy, inconsistent
- **Stage 1:** Coherent, geometry-aware patterns emerge
- **Stage 2:** Preserves structure, adds text-driven semantics

Takeaway

Two-stage design is **essential**: Stage 1 refines, Stage 2 aligns.

PatchAlign3D — Keypoint Detection Extension

PatchAlign3D features also enable keypoint detection (connecting back to ZeroKey)

Method	**IoU (%) @ geo. dist.**		
	0.01	0.05	0.10
RedCircle	0.34	3.05	18.50
CLIP-DINOiser	1.41	9.80	25.56
ZeroKey	13.16	56.60	79.43
PatchAlign3D (zero-shot)	—	—	32.88
PatchAlign3D (few-shot)	—	—	**64.07**

Complementarity

PatchAlign3D's dense features + few-shot adaptation → **64.07%** IoU@0.10

Different Trade-offs

- ZeroKey: **truly zero-shot**, no training
- PatchAlign3D: requires pre-training but faster inference

PatchAlign3D — Summary & Limitations

Key Contributions

1. **First encoder-only** 3D model with language-aligned local features
2. **Two-stage pre-training:** DINOv2 distillation $\rightarrow$ SigLIP text alignment
3. **Multi-positive contrastive** with fractional labels handles noisy annotations
4. **56.9%** mIoU on ShapeNetPart (+31.3% over COPS)
5. **0.4s** inference — **275× faster** than SATR

Limitations

- **Data coverage:** Pre-trained on curated Objaverse subset (32K shapes of 800K+)
- **Fixed patching:** Not adaptive/hierarchical for varying point cloud sizes
- **Annotation quality:** Relies on SAM+VLM pseudo-labels (inherently noisy)

Future: Scale data, adaptive patching, inherit global 3D foundation model knowledge.

Outline

Limitations & Discussion

ZeroKey Limitations

- **Prompt quality dependency:** Poor keypoint names $\rightarrow$ poor detection
- **Non-salient points:** Fails on arbitrary or ambiguous locations
- **Inference cost:** Requires MLLM calls per view per keypoint
- **Symmetric objects:** Difficulty distinguishing left/right

PatchAlign3D Limitations

- **Patch resolution:** 128 patches may miss very small parts
- **Training data bias:** Performance tied to Objaverse coverage
- **Real-world gap:** 22.7% mIoU on ScanObjectNN
- **XYZ-only input:** Does not leverage color or normals

Both methods demonstrate the **viability** of foundation-model-based 3D understanding, while highlighting the **gap between synthetic and real-world** performance.

Conclusion

ZeroKey (ICCV 2025)
Sparse keypoints via MLLM
79.4% IoU, **flexible**

⟷ **Complementary** ⟷

PatchAlign3D
Dense features via distillation
56.9% mIoU, **0.4s**

Unifying Theme: Local 3D via Language

- 2D foundation models (MLLM, DINOv2) → 3D
- Language enables **localization + semantic naming**
- No 3D keypoint/part supervision needed

Open Challenges

- Real-world noise & occlusion
- Fine-grained / small parts
- Unified sparse + dense models

Future Directions

Short-Term

- Extend to articulated / deformable objects
- Finer patch resolution for small parts
- Integration with 3D scene understanding

Scaling Up

- Larger pre-training data (full Objaverse)
- Multi-granularity features (part $\rightarrow$ sub-part)
- Real-world point cloud inputs (LiDAR, depth)

Ongoing: RL from LLM Feedback

- LLM scores correspondence quality as reward
- Train DINO+LoRA via reinforcement learning
- Preliminary: **+20%** PCK on 2D matching
- **Next:** Extend to 3D keypoints

Vision

A single foundation model for 3D shapes at **any granularity**, language-grounded, no category-specific training.

Thank You!

Questions?

ZeroKey: `https://sites.google.com/view/zerokey`

PatchAlign3D: `https://souhail-hadgi.github.io/patchalign3dsite/`

Website: `https://s2.hk`

Bingchen Gong • École Polytechnique