ROB 498/599 · 3D Robot Perception · Fall 2026

L11 Lifting 2D foundation models into 3D

3D labels are scarce and 2D models are strong: run SAM, CLIP and DINOv2 on every frame, then lift what they say onto the map with depth and pose.

Mon Oct 12, 2026 • WRITE-IN build

Lifting 2D foundation models into 3D

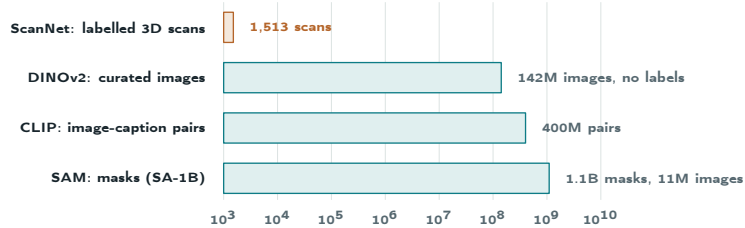
Unit 1 gave the robot a map with shape. Today the map learns what things are.

Nobody trains that from 3D labels. We borrow it from **2D models** trained on enormous image collections, and carry their answers onto the map with the depth and poses we already have.

Today

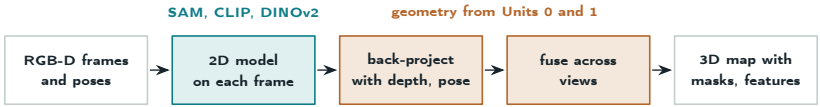
1. **Why lift:** how little labelled 3D data exists, and the recipe that works around it.
2. **The 2D models:** SAM and SAM 2 say *where*, CLIP says *what* in words, DINOv2 says *same as*. Then the rest of the zoo (Grounding DINO, OWL, LSeg, SigLIP, BLIP-2, RAM) and how to choose.
3. **Lifting, carefully:** back-project a pixel and a mask, test visibility, fuse features across views, associate fragments.
4. **OpenScene in depth:** fusion, distillation, the ensemble, results and the long tail. Then ConceptFusion, and what a feature map costs.

3D labels are scarce



- ScanNet, the standard labelled indoor set, has 1,513 scans. SAM was trained on 1.1 billion masks, about 100 per image.
- Labelling 3D is slow (every point, in a 3D tool), and capturing it needs a depth sensor in a real room. Photos and captions already exist by the billion.
- So: **let 2D models do the recognising, and use 3D only for geometry.**

The recipe



- Run a 2D model on **every frame**: masks, or a feature vector per pixel.
- **Back-project** each pixel with its depth and the camera pose (Unit 0) to land it on a 3D point.
- **Fuse** what different frames said about the same point, because each frame sees only part of the object and is sometimes wrong.
- The result is a map you can query: *where is the chair?*, *what is soft?*, *is this the same object as that?*

SAM: segment anything you point at

Kirillov et al., ICCV 2023

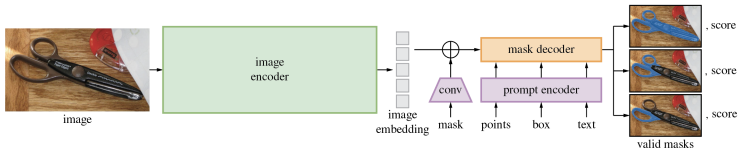


Figure: Meta AI, segment-anything repository, Apache 2.0

- **Promptable:** give a point or a box, get a mask. The heavy image encoder runs once per image; the decoder is cheap, so many prompts are cheap.
- A click is **ambiguous** (the shirt, or the person?), so SAM returns three nested masks, each with a predicted quality score.
- Prompting a grid of points gives every object in the image, with **no class names**.

What SAM does not give you

- **No names.** A mask is a set of pixels, not *chair*.
- **No identity across frames.** Mask 7 in one frame and mask 7 in the next are unrelated numbers.
- **Granularity is a choice.** Whole chair, seat, or cushion: the three masks disagree, and so will two frames.
- Names come from CLIP, next. Identity across frames comes from SAM 2 (right: one colour per object, kept across frames), or from 3D (later today).



Figure: Meta AI, sam2 repository, Apache 2.0

SAM 2: masks that follow the object

Ravi et al., 2024

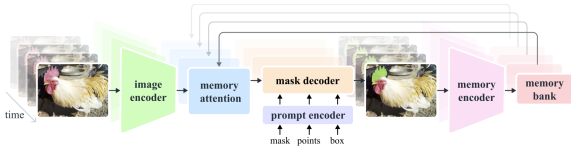


Figure: Meta AI, sam2 repository, Apache 2.0

- Prompt the object once; SAM 2 tracks its mask through the video, a **masklet**.
- Each new frame **attends to a memory bank** of earlier frames' features and masks: L10's cross-attention, with the past as keys and values.
- It can lose the object behind an occluder and recover it when it reappears. Images are just one-frame videos.

CLIP: images and words in one space

Radford et al., ICML 2021

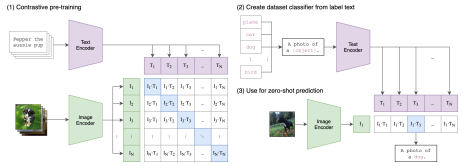


Figure: OpenAI, CLIP repository, MIT licence

- Two encoders: one turns an image into a vector, one turns a caption into a vector of the **same length**.
- Trained on 400 million web image-caption pairs: each image's vector should point the same way as **its own** caption's, away from the rest of the batch.
- Afterwards, any text is a query: compare it to an image by the cosine of the angle (right of the figure).

Matching by cosine similarity

A point's feature $\mathbf{f} = (0.6, 0.8, 0)$ and three text embeddings, all unit length (real ones have 512 or more numbers):

chair $= (0.8, 0.6, 0)$; *table* $= (0, 0.6, 0.8)$; *floor* $= (0, 0, 1)$.

Cosine of $\mathbf{f}$ with each prompt. Which label wins?

Real CLIP cosines are small and close together (0.2 to 0.3 is typical), so the **ranking** is what matters. Always normalise before comparing.

CLIP sees whole images, not pixels

- CLIP returns **one vector per image**. A map needs a vector per pixel, or at least per object.
- **Fix 1: crop.** Run CLIP on each SAM mask's crop. Simple, no training, but a crop loses context (a white rectangle: fridge or wall?). ConceptFusion's fix, later.
- **Fix 2: train a per-pixel head** that outputs CLIP-space features for every pixel: LSeg and OpenSeg. OpenScene uses these.
- Either way, the 3D map ends up storing **a CLIP-space vector per point**, and text queries work on it directly.

DINOv2: features without words

Oquab et al., 2023

- Self-supervised: a ViT trained on 142 million curated images with **no labels and no captions**, by making a student network's features match a teacher's on different crops of the same image.
- Its **patch features** are excellent: the same part of an object gets a similar vector across viewpoints, instances and lighting.
- So it is the right tool for **is this the same thing?**: matching across views, associating fragments, tracking. It cannot answer a text query.
- FoundationPose (L12) and many 3D trackers build on DINO-style features.

More 2D models: what each one hands you



Before choosing a model, decide what you want back. A **mask** says where; a **box with a phrase** answers one question; an **embedding** can be stored and asked anything later; **tags** and **captions** are words an LLM can read.

Grounding DINO: boxes from a phrase

Liu et al., ECCV 2024

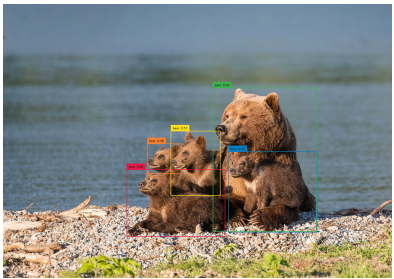


Figure: IDEA Research, Grounded-Segment-Anything repository, Apache 2.0

- **In:** an image and text, a class list or a phrase (*the red mug on the left*). **Out:** boxes, each tied to words of the text, with a score.
- A DETR-style detector with text fused into the image features; trained on detection, grounding and captioned images.
- 52.5 AP on COCO **without any COCO training data**.
- **Bad at:** masks; and it finds *something* even when nothing matches, so threshold the score.

Grounded-SAM: a phrase to a mask

Ren et al., 2024

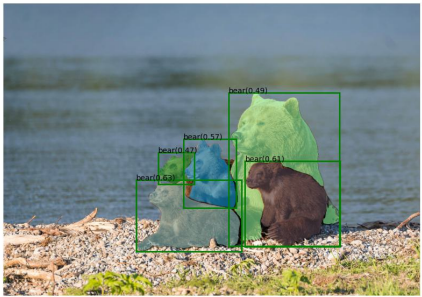


Figure: IDEA Research, Grounded-Segment-Anything repository, Apache 2.0

- Not a new model: **Grounding DINO's boxes become SAM's prompts.** Text in, masks with names out.
- Each piece does what it is best at: the detector reads the words, SAM draws the boundary.
- The standard way to get *the mask of the red mug* for back-projection.
- Errors chain: a wrong box gives a clean, confident, wrong mask.

OWL-ViT and OWLv2

Minderer et al., ECCV 2022; NeurIPS 2023

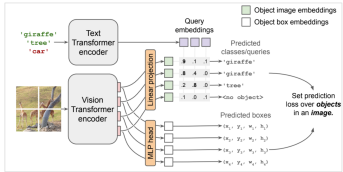
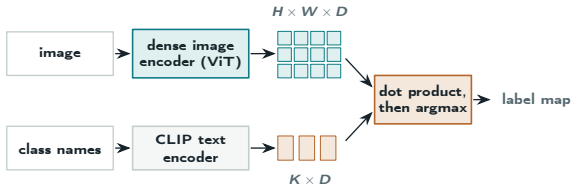


Figure: Google Research, scenic repository, Apache 2.0

- CLIP with a detection head: every output token of the image ViT gets a **box** and an **embedding** in CLIP space. Scoring a box against a text query is a dot product.
- So the box embeddings can be **stored and queried later**, and an *image* of an object works as a query too.
- OWLv2 self-trains on over a billion web images labelled by OWL-ViT itself: rare-class AP on LVIS from 31.2 to 44.6.

LSeg and OpenSeg: an embedding for every pixel

Li et al., ICLR 2022; Ghiasi et al., ECCV 2022



- **LSeg:** a dense encoder trained so each pixel's feature matches the CLIP text embedding of its label. Labelled segmentation data, so its vocabulary leans toward those classes.
- **OpenSeg:** proposes masks, then aligns *words from captions* to masks. Captions cover far more words than label sets.
- Both give the per-pixel CLIP-space features OpenScene fuses. The price: lower resolution than SAM's masks, blurry boundaries.

SigLIP: CLIP with a different loss

Zhai et al., ICCV 2023

CLIP's loss is a softmax over each row: *which caption is mine?* SigLIP scores every image-caption pair **on its own**, with a sigmoid: *is this pair a match, yes or no?*

- **Logits $s = \begin{bmatrix} 3 & -1 \\ 0 & 2 \end{bmatrix}$, matches on the diagonal. Row softmax, then sigmoid.**

- No normalising across the batch, so training needs less communication and works with smaller batches.
- For us: a drop-in replacement for CLIP, same kind of embedding, usually better scores. Its similarities are calibrated per pair, so a threshold means more.

BLIP and BLIP-2: captions and answers

Li et al., ICML 2022; ICML 2023

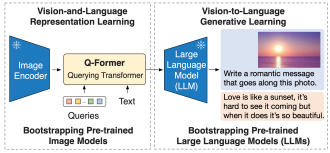


Figure: Salesforce Research, LAVIS repository, BSD 3-Clause

- **BLIP:** captioning and question answering, trained on web pairs it filters and re-captions itself.
- **BLIP-2:** a **frozen** image encoder and a **frozen** LLM, joined by a small trained **Q-Former**: learned query tokens cross-attend to the image (L10) and hand the LLM a few visual tokens.
- **In a map:** a caption per object, as text an LLM planner can read. ConceptGraphs captions each object with LLaVA, a model of this family, and summarises with GPT-4 (L13).

Tags: RAM, and Florence-2

Zhang et al., 2023; Xiao et al., CVPR 2024



Figure: Recognize Anything repository, Apache 2.0

- **RAM** lists what is in an image (*couch, dog, lamp, blanket*), trained on tags parsed from captions. Compare the three tag lists above: what each model **misses** is what a detector will never be asked for.
- **Florence-2:** one sequence-to-sequence model, prompted by task (caption, detect, ground, segment, OCR), trained on 5.4 billion annotations of 126 million images.

The zoo at a glance

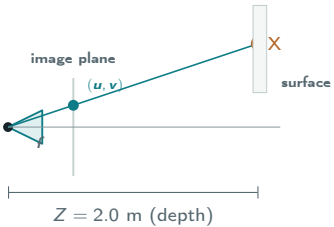
Model	In	Out	Trained on	Good at / bad at
SAM, SAM 2	image (video), point or box	masks (tracked)	1.1B masks	boundaries / no names
CLIP, SigLIP	image or text	one embedding	web image-text pairs	any word / not where
DINOv2	image	patch features	142M images, no labels	matching / no words
LSeg, OpenSeg	image	per-pixel embedding	labels; captions and masks	queryable maps / soft edges
Grounding DINO	image and text	boxes with phrases	detection, grounding, captions	phrases / one query per run
OWL-ViT, OWLv2	image, text or image query	boxes with embeddings	web pairs, self-training	store and re-query / boxes only
RAM, Florence-2	image (and task prompt)	tags; many outputs	parsed captions; 5.4B labels	what to ask for / misses
BLIP-2, LLaVA	image and question	caption or answer	web pairs, frozen LLM	words for an LLM / slow, can invent

Which model for which robot task?

Task	Pick	Why, and the trade-off
Find *the red mug*, now	Grounding DINO, then SAM (Grounded-SAM)	phrases with colour and position work; one pass per query, so slow for many queries
Segment everything, no names needed	SAM's point grid, or SAM 2 on video	best boundaries; you still need names (CLIP on crops) and identity (SAM 2 or 3D)
A map you can query later with any word	per-pixel or per-object CLIP or SigLIP features (LSeg, OpenSeg, or SAM crops)	store once, ask anything; memory per point, soft boundaries
Describe a scene to an LLM planner	tags (RAM) and captions (BLIP-2, LLaVA) per object	text an LLM can read; slow, and can invent details
Track one object as it moves	SAM 2 masklet, DINOv2 features to re-identify	masks through occlusion; loses look-alikes after long gaps
Fixed class list, fast, on board	a closed-set segmenter (Lab 1, Part B)	fastest and most accurate on its classes; blind to anything else

Back-projecting a pixel

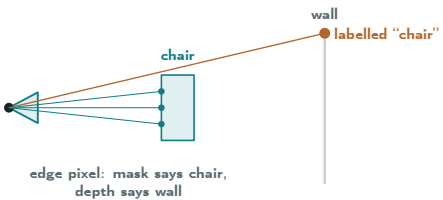
Unit 0, once more



$f_x = f_y = 500$, $(c_x, c_y) = (320, 240)$. Pixel $(u, v) = (420, 290)$ has depth $Z = 2$ m. Camera pose (to world): R = I, t = $(1, 0.5, 0.8)$. X_c, X_w: the point in camera and world frames.

The point in the camera frame, then in the world

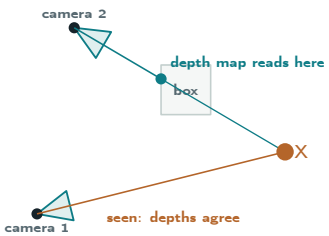
Back-projecting a mask



- Back-project every pixel in the mask: a **3D segment**, a partial point cloud of the object.
- **Edges bleed.** A boundary pixel can take the depth of the wall behind, and the wall gets labelled *chair*.
- **Depth holes:** dark, shiny or thin surfaces return no depth, so parts of the mask go nowhere.

Standard fixes: **erode** the mask by a few pixels, drop pixels near a **depth discontinuity**, and remove 3D outliers (points far from the segment's other points).

Which views actually see a point?

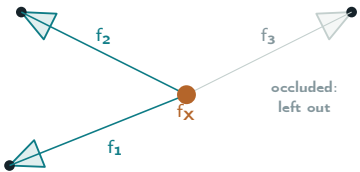


Project X into each camera (L02). At that pixel, compare the point's depth z with the depth map's D. Visible if $|z - D| \leq \tau$; here $\tau = 0.05$ m.

Camera 1: $z = 3.00$, $D = 3.02$. **Camera 2:** $z = 2.50$, $D = 1.60$.

Without this test, the box's label gets written onto everything behind it.

Fusing what the views say



View j gives point X the feature f_j at its pixel, with weight $w_j \geq 0$:

$$f_X = \frac{\sum_j w_j f_j}{\sum_j w_j} \quad \text{over the views that see X}$$

$f_1 = (0.8, 0.6)$, $f_2 = (0.6, 0.8)$. **Equal weights, then** $w_1 = 2$, $w_2 = 1$.

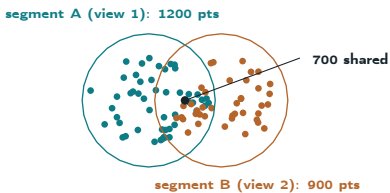
Choosing the weights

- **Equal:** OpenScene's choice. Simple, and fine with many views.
- **Distance and angle:** a view from 1 m, head-on, is more trustworthy than one from 5 m at a grazing angle; weight by inverse distance or by cos of the viewing angle.
- **Model confidence:** SAM's mask score, or how central the pixel is in its mask (edges are least reliable).
- **Running weight caps,** as in a TSDF: let old evidence fade so a moved chair can be relabelled.
- No weight fixes **pose error**: 2 cm of it is a 2 cm halo of wrong labels on every edge.

Trap

Average **features**, not **labels**. Two views saying *chair* and *armchair* average to something chair-like; voting on the label strings throws that away.

Associating fragments across views



Two 3D segments from different frames may be **the same object**. Voxelise them into voxel sets A, B; $|A|$ counts voxels, $A \cap B$ is the shared ones:

IoU of A (1,200 voxels) and B (900), sharing 700

- Merge when geometric overlap **and** feature similarity (CLIP or DINOv2 cosine) are both high. Overlap alone merges a cup with the table it sits on.
- Merging is a running average again: union of points, weighted mean of features.

An open-vocabulary map, in code

```
feats, weight = zeros(N, D), zeros(N)       # one slot per map point
for rgb, depth, T_wc in frames:
    F = clip_pixel_features(rgb)             # H x W x D, unit vectors
    X_c = backproject(depth, K)              # H x W x 3, camera frame
    X_w = X_c @ T_wc[:3,:3].T + T_wc[:3,3]   # to the world
    for p, (u, v) in visible(points, T_wc, depth, tau=0.05):
        feats[p] += F[v, u]; weight[p] += 1  # running sum
feats = normalize(feats / weight[:, None])   # average, then unit length
scores = feats @ text_encoder('a chair')     # cosine per point
```

- Every line is a slide from today: per-pixel features, back-projection, T_{wc}, the visibility test, the running average, normalisation, cosine.
- Line 6 goes the other way: it projects map points **into** the frame (L02) and keeps those the depth agrees with.

Where the features live

Unit 1's maps, with meaning

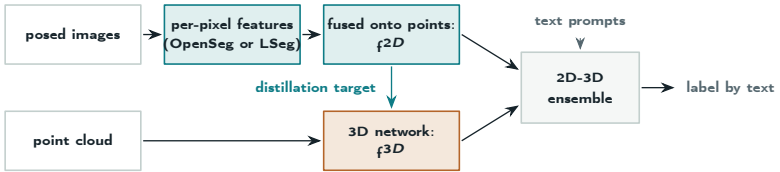
Map (Unit 1)	Feature stored on	Fusion	Example
Point cloud	each point	running average per point	OpenScene, ConceptFusion
TSDF voxels	each voxel near the surface	the TSDF weight update (L07)	semantic TSDF maps
Gaussians	each Gaussian, low-dimensional	learned by rendering feature images (L09)	LangSplat, Feature 3DGS
Objects	each object	average over its segments	ConceptGraphs (L13)

Same recipe each time: a 2D model's per-pixel output, carried onto whatever element the map is made of.

OpenScene

Peng et al., CVPR 2023

Goal: label every point of a 3D scene by **any text query**, with **no labelled 3D data at all**.



- **2D fusion:** per-pixel CLIP-space features, back-projected and averaged over the views that see each point. Today's first half, exactly.
- **3D distillation:** train a 3D network on point clouds alone to reproduce those features.
- **Ensemble:** at each point, use whichever of the two answers is more confident.

OpenScene's 2D fusion, precisely

- **2D model:** OpenSeg (768-number features) or LSeg (512). Run on every RGB frame; pixel features are upsampled to image resolution.
- **Pairing:** project each 3D point into each frame. With depth (ScanNet, Matterport3D), keep the pair only if the point's depth matches the depth map, the visibility test from earlier. Outdoors (nuScenes) the LiDAR points are projected into the camera images.
- **Fusion:** plain **average pooling** over the frames that see the point. No learned weights.
- **3D backbone:** MinkUNet18A, a sparse-convolution U-Net (L06, and Lab 1 Part B), with its output width set to the feature length.

Every piece is something you built in Units 0 and 1. The new part is only *what* gets fused.

Distilling into a 3D network

P: the points at least one image saw; p: one of them. Net, the 3D network, maps the point cloud to a feature f_p^{3D} per point, trained toward the fused 2D feature f_p^{2D}. No labels anywhere. The loss $\mathcal{L}$ averages $1 - \cos$ over the $|P|$ points:

$$\mathcal{L} = \frac{1}{|P|} \sum_{p \in P} \left(1 - \cos\left(\mathrm{f}_p^{3D},\ \mathrm{f}_p^{2D}\right)\right)$$

One point: $\mathrm{f}^{3D} = (0.6, 0.8)$, $\mathrm{f}^{2D} = (0.8, 0.6)$. **Its loss?**

Cosine, not squared distance: only the **direction** of a CLIP feature carries meaning, and fused features have shrunk lengths.

Why distil at all?

	2D fused feature	**3D distilled feature**
Needs at test time	posed images and depth	only the point cloud
Strong at	small objects, texture, fine categories	shape: walls, floors, large furniture
Weak at	points few views saw; inconsistent views	things defined by appearance, not shape
Consistency	can flicker point to point	smooth over a surface

They fail in different places, so combine them.

The 2D-3D ensemble

Encode each class c's prompt as a text vector t_c. At each point, with its 2D feature f^{2D} and 3D feature f^{3D}, score both features against every prompt and keep the feature with the **higher best score**.

cosine with	chair	table	floor
f^{2D}	0.30	0.25	0.10
f^{3D}	0.22	0.21	0.20

Which feature does the ensemble use here, and what label results?

The 3D feature here is nearly undecided (0.22, 0.21, 0.20), so the 2D one wins. On a bare wall it is often the other way round.

What it produces

zero-shot, no 3D labels

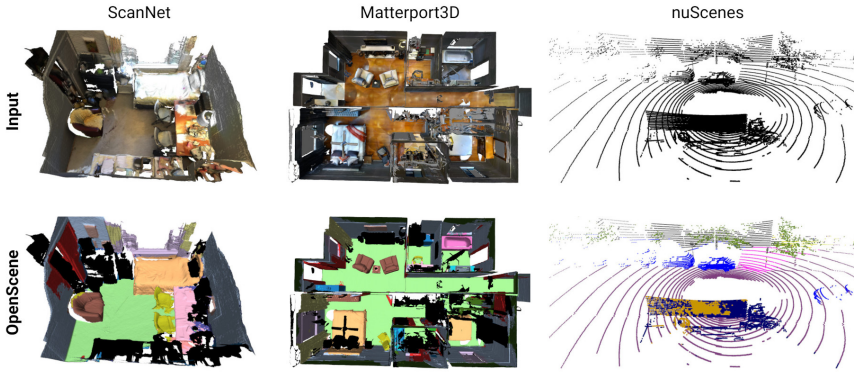
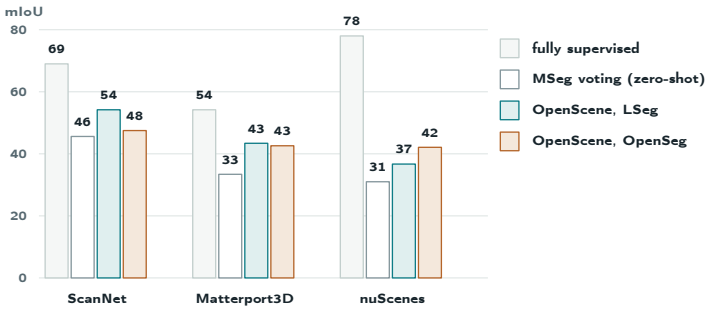


Figure: OpenScene repository (Peng et al.), Apache 2.0

How good is zero-shot?



- On ScanNet's 20 classes, OpenScene reaches 79% of a fully supervised network's mIoU with **no 3D labels**.
- It beats the previous zero-shot approach (MSeg voting) everywhere.
- Outdoors the gap is large: LiDAR is sparse and the images cover less.

Where zero-shot wins: the long tail



- Evaluate on more and more Matterport3D classes, $K = 21$ to 160.
- Supervised accuracy falls fast: rare classes have few labelled examples. OpenScene falls more slowly; it **overtakes** from $K = 40$.
- A robot's world is mostly long tail.

Querying the map with text

- **Segmentation:** prompts for a class list; each point takes the class with the highest cosine. Zero-shot, any list.
- **Search:** one prompt, *a soft place to sit*. Colour points by cosine; the sofa and bed light up.
- **Beyond objects** (the paper's examples): materials (*glass, fabric, metal, wood*), affordances (*comfy*, *hot*, *play*), activities (*cook, dine, sleep, wash*), room types, even *cluttered*.
- **Image queries:** a photo of a pool table, through CLIP's image encoder, finds the pool table.
- Prompts matter: OpenScene turns each class *chair* into *a chair in a scene*, worth 2.3 mIoU on ScanNet. Rare words and long phrases work worse.

Limit

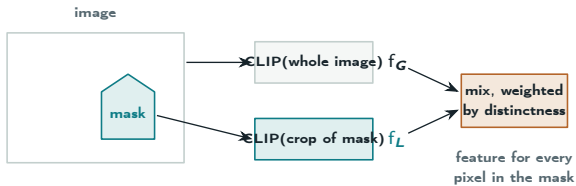
A per-point feature has **no notion of an object**. *How many chairs?* and *the chair next to the door* need instances and relations. That is L13.

OpenScene's limits

- **No objects:** per-point labels, no instances, so no counting, no *the chair by the door*.
- **Bag of words:** CLIP features ignore word order and negation; *not a chair* lights up chairs.
- **Evaluated only on closed sets.** The open queries are shown qualitatively; there is no benchmark for *comfy*. The authors say so.
- **Inherits 2D blind spots:** what LSeg or OpenSeg cannot see, fusion cannot recover; the 3D network only smooths it.
- **Static scenes, offline:** features are fused over a whole recorded scan.

ConceptFusion: no training at all

Jatavallabhula et al., RSS 2023



- Masks from a class-agnostic segmenter; CLIP on each mask's **crop** and on the **whole image**.
- Mix the two per mask: a distinctive region keeps more of its own feature, an ordinary one borrows more context from the whole image. Every pixel in the mask gets the mix.
- Fuse into a 3D map by back-projection and running averages: today's first half again, with **no 3D network**.
- Any encoder in CLIP's space can query it: text, an image, a click on the map, even audio.

What a feature map costs

1,000,000 **points, a** 512**-number feature each, 4-byte floats. And the coordinates alone?**

- The features are **170 times** the geometry. Lab 1's memory column for a feature map would be dominated by them.
- Fixes: half precision, PCA to fewer dimensions, a **feature per object** instead of per point (L13), or features on Gaussians (LangSplat, Feature 3DGS) decoded on demand.

OpenScene and ConceptFusion side by side

	OpenScene	**ConceptFusion**
2D features	per-pixel (OpenSeg, LSeg)	CLIP on mask crops plus the global image
3D training	a distilled 3D network	none
Test-time input	point cloud (plus images for the ensemble)	posed RGB-D stream
Queries	text	text, image, click, audio
Map	a feature per point	a feature per point, built incrementally

Learn it again, differently

the explanations worth your time

- **OpenScene project page** (pengsongyou.github.io/openscene): the interactive queries are the fastest way to see what per-point CLIP features can and cannot do.
- **SAM 2 demo** (sam2.metademolab.com): click an object in a video and watch the memory carry it through occlusion.
- **OpenAI's CLIP post** (openai.com/index/clip): zero-shot classification by prompt, with its failure cases.
- **DINOv2 demo and paper figures** (dinov2.metademolab.com): the PCA colouring that shows matching parts across animals.

What you should be able to do

- Explain why 3D perception borrows from 2D models, and say what SAM, CLIP and DINOv2 each contribute.
- Say what each 2D model in the zoo outputs, and pick one for a robot task: a stored embedding, or a one-shot answer?
- Back-project a pixel and a mask to world coordinates, and name what goes wrong at mask edges.
- Run a visibility test against a depth map, and fuse features across views with a weighted average.
- Describe OpenScene's three stages, compute its distillation loss, apply its ensemble rule, and explain why it overtakes supervised training on the long tail.
- Compute cosine similarity against text prompts, contrast OpenScene with ConceptFusion, and estimate a feature map's memory.

Next

L12, Wed Oct 14

- **Response due Sun Oct 18:** OpenScene. SAM 2, DINOv2 and ConceptFusion are also assigned.
- **L12, object-centric perception:** from labelled points to objects with a 6-DoF pose. FoundationPose, shape completion, articulated objects.
- **Project proposals,** naming a target paper and an evaluation, are due **Fri Oct 23**.