# Perception for Embodied AI

*From Pixels to Policies: How Robots See and Understand the World*

# Why Perception Is the Bottleneck

**Perception failures cause most real-world policy breakdowns.**

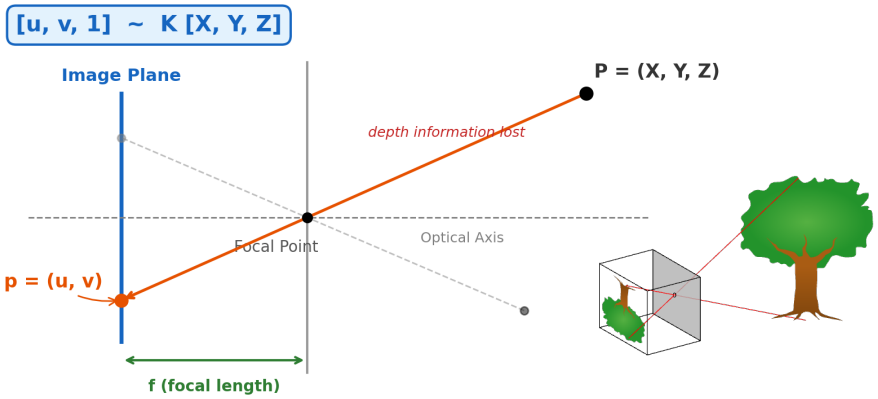
Three failure categories:

1. Sensor noise
   Depth dropouts, motion blur, lighting changes
2. Representation errors
   Incorrect point clouds, missed detections, wrong poses
3. Distribution shift
   Sim-to-real gap, novel objects, new environments

**The Sim-to-Real Gap:** A policy achieving 95% success in simulation may drop to ~40% in the real world — primarily due to perception differences, not control errors.

# The Pinhole Camera Model

| Symbol | Domain | Meaning |
|---|---|---|
| **K** | $\mathbb{R}^{3x3}$ | Camera intrinsic matrix |
| **$f_x$, f_y** | $\mathbb{R}$ | Focal lengths (pixels) |
| **$c_x$, c_y** | $\mathbb{R}$ | Principal point coordinates |
| **P = [X, Y, Z]$^T$** | $\mathbb{R}^3$ | 3D point in camera frame |
| **p = [u, v]$^T$** | $\mathbb{R}^2$ | 2D pixel coordinates |




## Pinhole Projection

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \mathbf{K} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

The ~ means "up to scale" — projection loses depth information. This is why a single RGB image is fundamentally ambiguous about 3D structure.

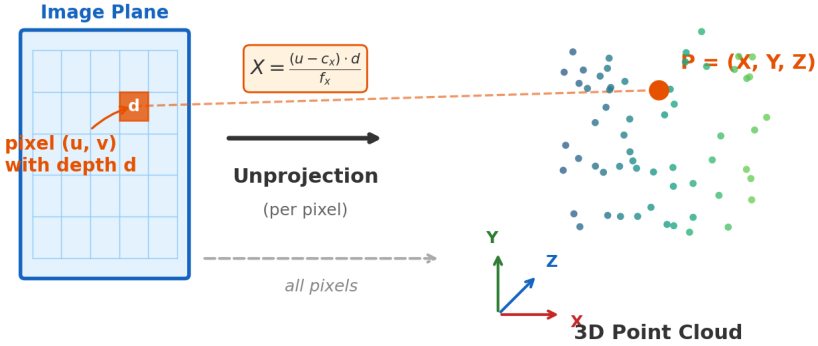
*Common misconception: assume pixels directly correspond to metric coordinates. Different cameras produce different K matrices for the same scene.*

# From 2D to 3D: Depth Unprojection

The inverse of projection: given depth d at pixel (u, v), recover the 3D point. Applied per-pixel, this generates point clouds from depth images.

## Depth Unprojection

$$X = \frac{(u - c_x) \cdot d}{f_x}, \quad Y = \frac{(v - c_y) \cdot d}{f_y}, \quad Z = d$$




## Learning Connection

Every point cloud you'll encounter in robot learning — from PerAct's voxelized inputs to DP3's sparse point clouds — begins with this unprojection step. When a depth sensor fails (returning d = 0 for transparent objects), the resulting point cloud has holes that propagate through the entire perception pipeline.

**Key Insight:** Depth quality directly determines 3D reconstruction quality. All downstream 3D perception — point clouds, voxels, scene representations — is only as good as the depth sensor.

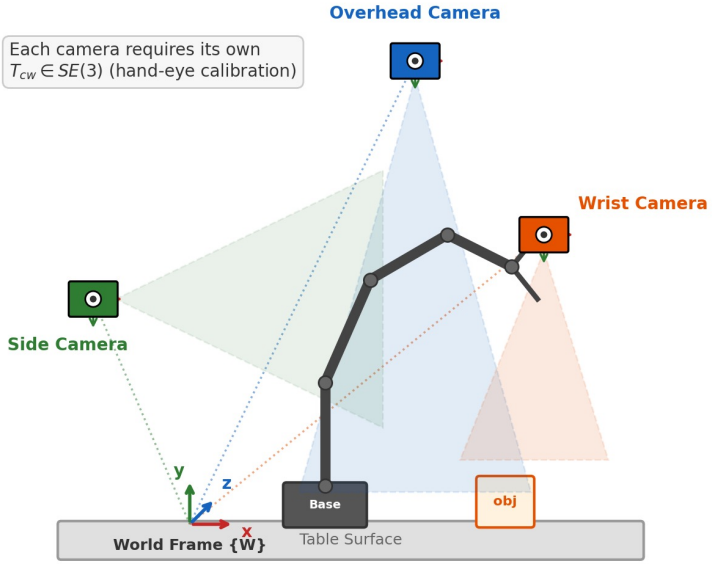
# Extrinsic Parameters and Multi-Camera Setups

## Camera-to-World Transform

$$\mathbf{P}_{\text{world}} = \mathbf{R}\,\mathbf{P}_{\text{cam}} + \mathbf{t}$$
$$\mathbf{T}_{cw} \in SE(3)$$

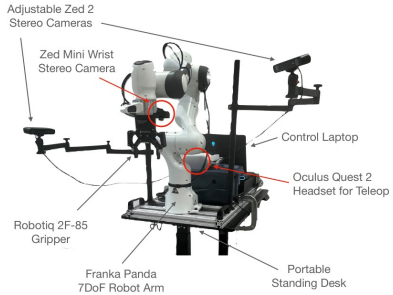
R ∈ SO(3): rotation matrix (3×3, orthogonal, det = 1)
t ∈ ℝ³: translation vector
T_cw ∈ SE(3): full rigid body transform (4×4 homogeneous)

*Connects back to SE(3) from Lecture 2 (kinematics).*




## Multi-Camera Configurations in Practice

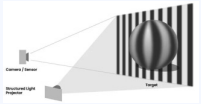
| System | Cameras | Configuration |
|---|---|---|
| **DROID (2024)** | 3 cameras | 2× ZED 2 (stereo) + 1× ZED Mini (wrist) |
| **ALOHA 2 (2024)** | 4 cameras | 4× RealSense D405 (structured light) |
| **Mobile ALOHA** | 3 cameras | 3× Logitech webcams (RGB only) |




*Hand-eye calibration required for each camera to establish T_cw accurately.*

# Depth Sensing Technologies

## Structured Light



*Example: Intel RealSense D405/D435*
Range: 0.1–10m | Best for: close-range manipulation
Projects IR pattern, triangulates depth from deformation
⚠ Fails: sunlight, transparent/reflective surfaces

## Stereo Vision

*Example: Stereolabs ZED 2*
Range: 0.3–20m | Best for: medium-range, outdoor
Two cameras compute disparity → depth via calibration
⚠ Fails: textureless surfaces, far range (disparity → 0)

## Time-of-Flight (ToF)

*Example: Azure Kinect DK*
Range: 0.5–5m | Best for: whole-body, indoor
Measures light round-trip time per pixel
⚠ Fails: multi-path interference, mixed pixels at edges

## LiDAR

*Example: Velodyne VLP-16, Ouster OS0*
Range: 1–100m | Best for: navigation, outdoor
Laser pulses measure distance at high speed
⚠ Fails: sparse density, expensive, rain/fog scatter

*Each technology has fundamentally different failure modes — sensor choice has cascading consequences for the entire perception pipeline.*

# What Sensors Do Robot Learning Systems Actually Use?

A survey of sensor configurations across major robot learning platforms:

| System | Year | Cameras | Depth? | Notes |
|---|---|---|---|---|
| **DROID** | 2024 | 2× ZED 2 + 1× ZED Mini | Yes (stereo) | 76k demos, 564 scenes |
| **ALOHA 2** | 2024 | 4× RealSense D405 | Yes (struct. light) | Close-range bimanual |
| **Mobile ALOHA** | 2024 | 3× Logitech C922x | No (RGB only) | Low-cost mobile manip. |
| **Open X-Embodiment** | 2023 | Heterogeneous (22 platforms) | Mixed | RT-X uses single-view RGB |
| **BridgeData V2** | 2023 | 1× RGBD + 2× RGB + 1× wrist | Partial | Budget-friendly setup |

**Key Insight:** There is no standard sensor configuration. The choice depends on the task, robot form factor, and whether the policy architecture requires depth. Notice the trend: newer systems increasingly use RGB-only.

# The RGB-Only Surprise

**Most state-of-the-art Vision-Language-Action models use RGB only — no depth.**

RT-1, RT-2, OpenVLA, $\pi_0$ — all RGB-only

## Why?

Pretrained VLMs are trained on billions of internet RGB images. Adding depth channels breaks pretraining alignment — the model loses its powerful visual representations. Developing fusion architectures that preserve pretrained features while incorporating 3D information remains an open research problem.

**Learning Connection**
This is a fundamental tension in modern robot learning. Pretrained VLMs provide powerful visual features but expect RGB input. Adding depth requires either training from scratch (losing billions of images of pretraining) or developing fusion architectures that preserve pretrained representations. The seemingly consensus — fusing 2D semantic features with 3D geometric features via separate encoders — is one resolution.

**The Debate:** "Should we throw away geometric information to leverage pretrained vision?"

# Challenging Conditions for Depth

## Transparent Objects

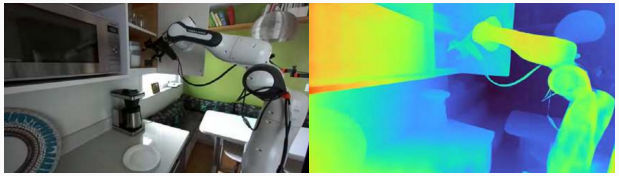
Glass, plastic bottles, clear containers
Depth sensor returns 0 or background depth
ClearGrasp, ClearDepth attempt solutions


## Reflective Surfaces

Metal, mirrors, glossy surfaces
IR/light reflects away from sensor
Causes missing or erratic depth values

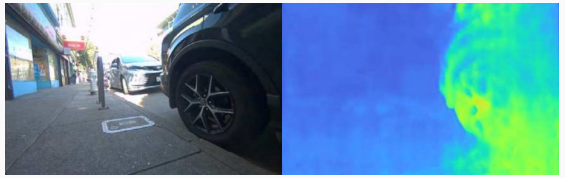
## Thin Structures

Wires, handles, pen tips
Below sensor spatial resolution
Mixed-pixel effects at edges



## Outdoor / Sunlight

IR-based sensors overwhelmed by sunlight
LiDAR less affected but expensive
FoundationStereo (CVPR '25), Depth Anything V2



# Part 1 Summary: Sensing and the Camera Model

**Key Insight:** Sensor choice is a design decision with cascading consequences for representation, pose estimation, and policy learning. There is no universally "best" sensor — only the right sensor for your task, budget, and policy architecture.

**Three things to remember:**

**1. The Camera Model Is the Foundation**
Projection (3D → 2D) and unprojection (2D + depth → 3D) underlie all visual perception. K and T_cw are the essential parameters.

**2. Depth Is Powerful but Not Universal**
Depth enables 3D reasoning but every sensor has failure modes. Modern VLAs increasingly use RGB-only to leverage pretrained vision models.

**3. Perception Failures Are the Primary Bottleneck**
Most real-world policy breakdowns trace to perception errors — sensor noise, representation failures, or distribution shift. Understanding sensors is the first step to debugging policies.

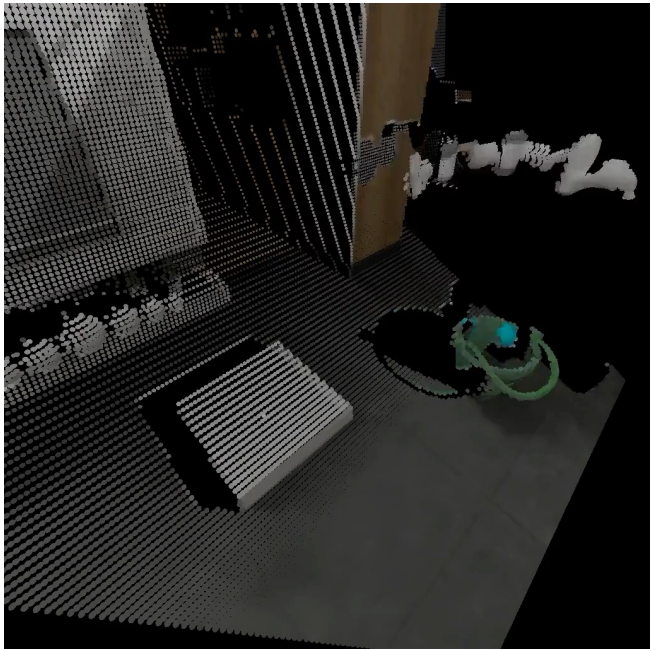
*Next: How do we represent the 3D world once we have sensor data?*

# 3D Representations for Robotics

*Choosing How the Robot Sees the World in Three Dimensions*

# Point Clouds: The Raw 3D Signal

**Definition:** An unordered set of 3D points $\{(x, y, z)_i\}$, often with color or normals.



## Key Properties

**Irregular density** — more points near surfaces facing the sensor, sparse at grazing angles

**Incomplete** — single-view captures only visible surfaces; back is always missing

**Variable cardinality** — number of points changes per frame; requires set-based networks

## Processing Networks

**PointNet (Qi et al., CVPR 2017)**
Per-point MLPs + symmetric max-pooling. Handles unordered sets directly. Foundation for 3D point cloud learning.

**PointNet++ (Qi et al., NeurIPS 2017)**
Hierarchical feature learning via set abstraction layers. Captures local geometry at multiple scales.

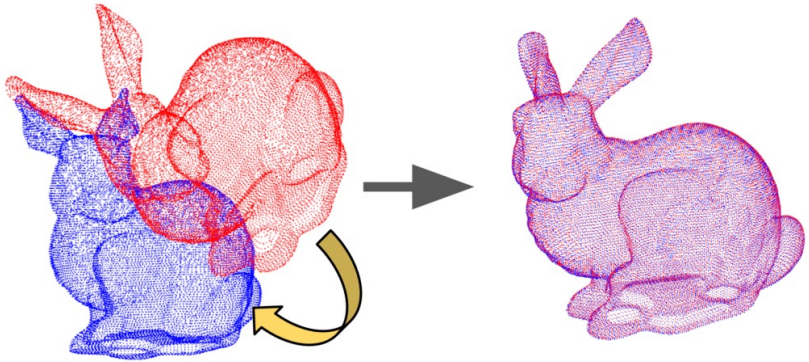
# Point Cloud Processing Fundamentals

**Normal Estimation** — PCA on k-nearest neighbors → surface orientation

**Downsampling** — Voxel grid filter or farthest point sampling (FPS)

**Plane Segmentation** — RANSAC fits planes → remove table/floor

**Registration (ICP)** — Align two point clouds by iterative closest point



### ICP Objective

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|^2$$

$p_i$: source points, $q_i$: target correspondences

### Learning Connection

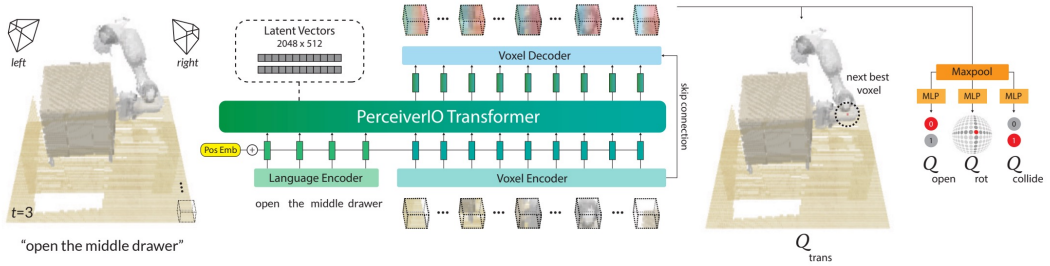
ICP is the refinement stage in classical pose estimation, the geometric backbone behind FreeZe's BOP 2024 victory (frozen features find correspondences, then ICP refines), and the evaluation metric for registration quality. Understanding ICP gives you intuition for what learned methods are trying to replace — or improve upon.

# Voxel Grids: Discretizing 3D Space

Regular 3D grid of cells, each storing occupancy, features, or semantics.

**Advantages:**
Enables standard 3D convolutions and attention
Regular structure → efficient batch processing
Natural spatial queries ("what's at this location?")

**Trade-off:**
Memory scales as O(n³) — 100³ = 1M voxels, 200³ = 8M
Resolution vs. workspace size is a fixed budget




**PerAct (Shridhar et al., CoRL 2022)**
100³ voxels + Perceiver Transformer → predicts next-best action as voxel index. 18 RLBench tasks from 100 demos.

**Key Insight:** Voxels are the "image of 3D" — they trade off the irregularity of point clouds for a regular grid that standard deep learning architectures (CNNs, Transformers) can process directly. The cost is the cubic memory scaling that limits resolution.

*PerAct showed that voxel-based manipulation policies can solve multi-step tasks from language instructions.*

# Beyond Voxels: Act3D and Adaptive 3D Features

### Act3D (Gervet et al., CoRL 2023)

Lifts 2D pretrained features into 3D via depth
Adaptive 3D feature fields — no fixed voxel grid

**+22% over PerAct on RLBench**
**3× less compute**

Key idea: use pretrained 2D vision + inject depth, rather than learn 3D from scratch.

### RVT-2 (Goyal et al., RSS 2024)

Avoids 3D construction entirely
Multi-view re-rendering from virtual cameras

**82% on RLBench**
**2× faster inference than Act3D**

Key idea: re-render from multiple virtual viewpoints, predict actions in each view.

### Learning Connection

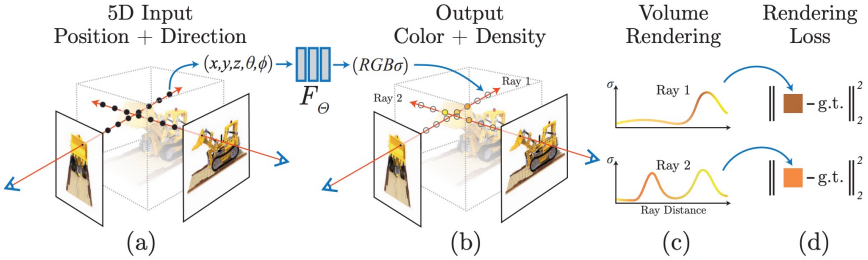
Act3D and RVT-2 illustrate a broader principle in robot learning: leveraging pretrained 2D representations and injecting 3D structure through architecture design (depth lifting, multi-view rendering) is often more effective than training 3D representations from scratch, because 2D pretraining data is orders of magnitude more abundant.

*PerAct (2022) → Act3D (2023) → RVT-2 (2024): each generation finds better ways to combine 2D features with 3D reasoning.*

# Neural Implicit Representations: NeRF

**Neural Radiance Fields (NeRF)**
A continuous volumetric representation:
a neural network maps any 3D point (x, y, z) + viewing direction to color and density.

Trained from posed images — learns to render novel views by integrating color along camera rays.




## Volume Rendering (Conceptual)

$$\hat{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\,\sigma(\mathbf{r}(t))\,\mathbf{c}(\mathbf{r}(t),\mathbf{d})\,dt$$

T(t): transmittance, σ: density, c: color, r(t): point along ray, d: viewing direction

## NeRF in Robotics

**Neural Descriptor Fields**
SE(3)-equivariant features for manipulation

**F3RM**
Language-guided grasping via NeRF features

**Key Limitation**
Slow rendering (seconds/frame) — impractical for real-time control

*NeRF showed that neural scene representations could encode rich 3D information — but its speed limitation motivated the search for faster alternatives.*

# 3D Gaussian Splatting

**Each Gaussian stores:**

Position ($\mu \in \mathbb{R}^3$) — center in 3D space
Covariance ($\Sigma \in \mathbb{R}^{3\times3}$) — shape and orientation
Color ($c \in \mathbb{R}^3$ via spherical harmonics) — view-dependent
Opacity ($\alpha \in [0,1]$) — transparency

**100+ FPS rendering | Editable | Differentiable**


**Key Insight:** 3DGS combines the strengths of classical explicit representations (editability, real-time rendering, interpretability) with the strengths of neural representations (differentiable, learnable from images). This hybrid nature explains its rapid adoption in robotics.

| | NeRF | 3D Gaussian Splatting |
|---|---|---|
| **Representation** | Implicit (MLP) | Explicit (Gaussians) |
| **Render Speed** | ~seconds/frame | 100+ FPS |
| **Editability** | Difficult | Direct manipulation |

# 3DGS for Robotics

## SplatSim

**Zero-shot sim-to-real transfer**

Reconstructs real scene as Gaussians, composites into sim for training. Policies transfer without real data.

## RoboSplat

**Demo augmentation**

Edits Gaussian scenes to create augmented training demos.

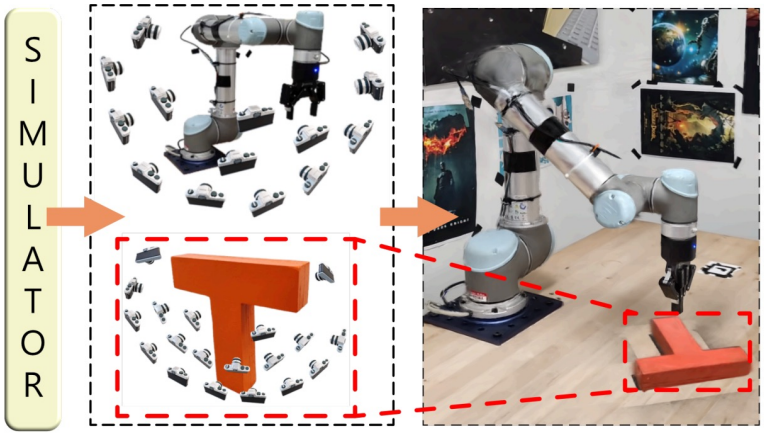
**87.8% vs. 57.2% baseline**

## ManiGaussian

**Dynamic world model**

Predicts how Gaussians move under actions. Plans by imagining futures.

**+13.1% on RLBench**




Gaussian Splats of Robot and Object

Rendering from *SplatSim* at Novel Joint Angle and Object Position

These three papers map to three fundamental uses of 3D representations in robot learning: SplatSim → sim-to-real transfer, RoboSplat → data augmentation for imitation learning, ManiGaussian → world models for planning. The 3DGS representation enables all three through its combination of photorealism, editability, and speed.

# Empirical Evidence: Point Cloud Matters

*"Point Cloud Matters" (NeurIPS 2024) — OBSBench*

## Key Findings from 125 Tasks

**Point clouds consistently outperform RGB and RGB-D**
for contact-rich manipulation tasks across all tested policies

**Greater viewpoint robustness**
3D representations maintain performance under camera perturbation

**Depth quality matters**
Noisy depth degrades point cloud advantages — sensor choice matters

| Tasks | Diffusion Policy | | | | | | |
|---|---|---|---|---|---|---|---|
| | RGB | | RGB-D | | | Point Cloud | |
| | ResNet | ViT | ResNet | ViT | MultiViT | SpUNet | PointNet |
| *ManiSkill2* | | | | | | | |
| PickCube | 0.17 | 0.24 | 0.34 | 0.58 | 0.00 | **0.71** | 0.70 |
| StackCube | 0.03 | 0.00 | 0.59 | 0.03 | 0.00 | **0.04** | 0.00 |
| TurnFaucet | 0.08 | 0.07 | 0.24 | 0.30 | 0.00 | 0.32 | **0.36** |
| Peg-Insertion-Side (Grasp) | 0.78 | 0.45 | 0.94 | 0.68 | 0.46 | 0.82 | **0.83** |
| Peg-Insertion-Side (Align) | 0.07 | 0.02 | 0.11 | 0.03 | 0.02 | 0.09 | **0.16** |
| Peg-Insertion-Side (Insert) | **0.01** | 0.00 | 0.01 | 0.00 | 0.00 | **0.01** | **0.01** |
| Excavate | 0.01 | 0.02 | 0.23 | 0.03 | 0.00 | 0.17 | **0.24** |
| Hang | 0.52 | 0.42 | 0.77 | 0.56 | 0.00 | 0.67 | **0.72** |
| Pour | 0.00 | 0.00 | 0.06 | 0.00 | 0.00 | 0.00 | 0.00 |
| Fill | 0.36 | 0.04 | 0.72 | 0.03 | 0.01 | 0.21 | **0.68** |
| *RLBench* | | | | | | | |
| open drawer | 0.00 | 0.00 | 0.12 | 0.00 | 0.08 | **0.28** | 0.12 |
| sweep to | 0.00 | 0.04 | 0.00 | 0.00 | 0.04 | 0.08 | **0.16** |
| meat off grill | 0.00 | 0.00 | 0.00 | 0.00 | **0.12** | 0.00 | 0.00 |
| turn tap | **0.24** | 0.04 | 0.04 | 0.12 | 0.16 | 0.16 | 0.16 |
| reach and drag | **0.08** | 0.04 | 0.04 | 0.00 | 0.00 | 0.04 | **0.08** |
| put money | 0.08 | 0.08 | 0.16 | 0.08 | **0.20** | 0.16 | 0.08 |
| push buttons | 0.04 | 0.00 | 0.04 | 0.00 | **0.12** | 0.04 | **0.12** |
| close jar | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| place wine | **0.04** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Mean S.R. ↑ | 0.13 | 0.08 | 0.23 | 0.13 | 0.06 | 0.20 | **0.23** |
| Mean Rank ↓ | 3.37 | 4.47 | 2.37 | 4.00 | 4.21 | 2.42 | **1.89** |

**Potential Consensus:** Fuse 2D semantic features (from pretrained vision models) with 3D geometric features (from point clouds or depth). This captures the best of both worlds — rich semantics from internet-scale pretraining, and precise geometry from depth sensing.

# 3D Representation Decision Framework

| Representation | Geometric Precision | Semantic Richness | Compute Cost | Viewpoint Robust | Pretrain Leverage | Best For |
|---|---|---|---|---|---|---|
| **Point Clouds** | ★★★ | ★ | ★★ | ★★★ | ★ | Contact-rich manip. |
| **Voxels** | ★★ | ★★ | ★ | ★★★ | ★ | Multi-step, language |
| **RGB** | ★ | ★★★ | ★★★ | ★ | ★★★ | VLAs, large-scale |
| **3DGS** | ★★★ | ★★ | ★★ | ★★★ | ★ | Sim-to-real, augment. |
| **Fused (2D+3D)** | ★★★ | ★★★ | ★ | ★★★ | ★★★ | General-purpose |

*Next: How do we determine where objects are in 3D? → Object Pose Estimation*

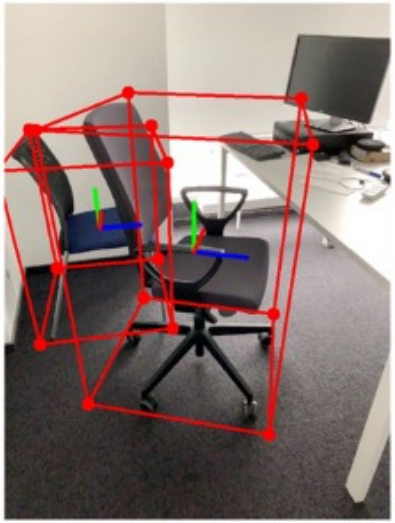
# Object Pose Estimation

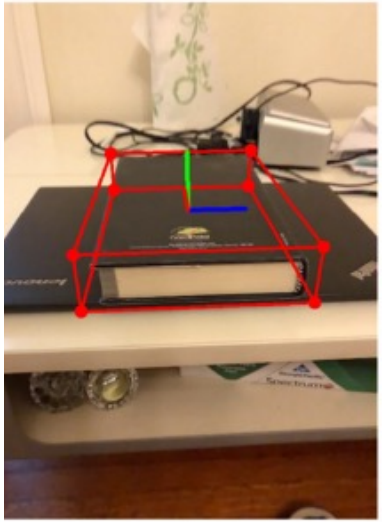
*From Geometry to Foundation Models*

# What Is 6-DoF Pose Estimation?

Determine the position and orientation of an object in 3D space — 6 degrees of freedom.

| Symbol | Domain | Meaning |
|---|---|---|
| **R** | SO(3) | Rotation matrix (3×3, orthogonal, det = 1) |
| **t** | $\mathbb{R}^3$ | Translation vector |
| **T = [R \| t]** | SE(3) | Full rigid body transform (4×4 homogeneous) |
| **AR** | [0, 100] | Average Recall — standard BOP evaluation metric |





## Two Problem Setups

**Model-Based**
CAD model is given. Match observed data against known 3D model.

**Model-Free**
Only reference images are available. Must infer geometry from appearances. FoundationPose unified both.

*Why it matters: precise grasping requires pose accuracy within millimeters; placement tasks need exact SE(3) targets; planning needs object state. Connects back to SE(3) from Lecture 2.*

# Classical Approach: Feature Matching + ICP

**1. Extract Features**
FPFH, SHOT, or other 3D descriptors computed at keypoints

→

**2. Match Correspondences**
Find nearest-neighbor feature matches between scene and model

→

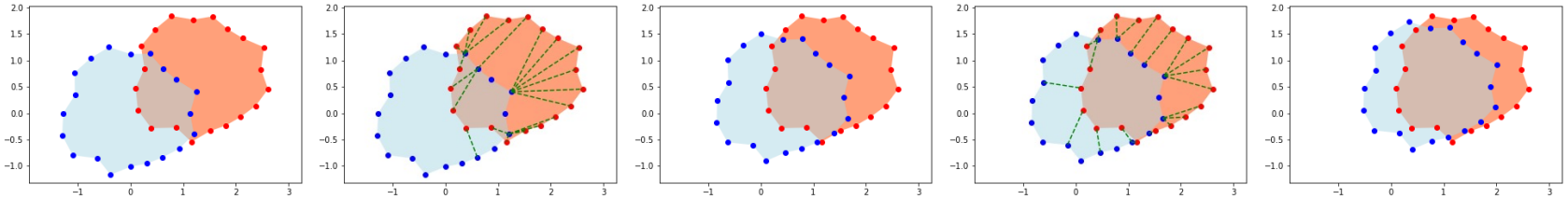
**3. RANSAC**
Random sample consensus to reject outlier matches

→

**4. Initial Transform**
Estimate R, t from inlier correspondences

→

**5. ICP Refinement**
Iterative closest point refines alignment (EQ4 from Part 2)



**Strengths:** Strong geometric priors, no training data needed, interpretable pipeline, works well for textured objects with known CAD models.

**Limitations:** Requires CAD models, fails on textureless/symmetric objects, sensitive to occlusion, slow for real-time control.

*Each limitation on the right motivates the learning-based solutions in the next slides.*

# Limitations of Classical Methods

| | |
|---|---|
| **Textureless Objects** Feature descriptors rely on texture patterns. Smooth plastic, metal parts, and uniform surfaces produce no distinctive features. | **→ CNNs learn appearance-based features** |
| **Symmetric Objects** Multiple valid pose solutions exist. A bowl looks identical under 360° rotation — classical matching can't disambiguate. | **→ Networks learn symmetry-aware loss functions** |
| **Heavy Occlusion** Partial visibility breaks correspondence matching. Feature-based methods need sufficient visible surface area. | **→ Dense prediction handles partial views** |
| **CAD Model Requirement** Every new object needs a precise CAD model. Impractical for household environments with thousands of object types. | **→ Generalizable models eliminate per-object retraining** |
| **Speed** Multi-stage pipeline is computationally expensive. Feature extraction + matching + RANSAC + ICP takes seconds per object. | **→ Single forward pass prediction** |

*These five limitations collectively drove the field toward deep learning approaches, starting with PoseCNN (2018).*

# PoseCNN and the Deep Learning Revolution

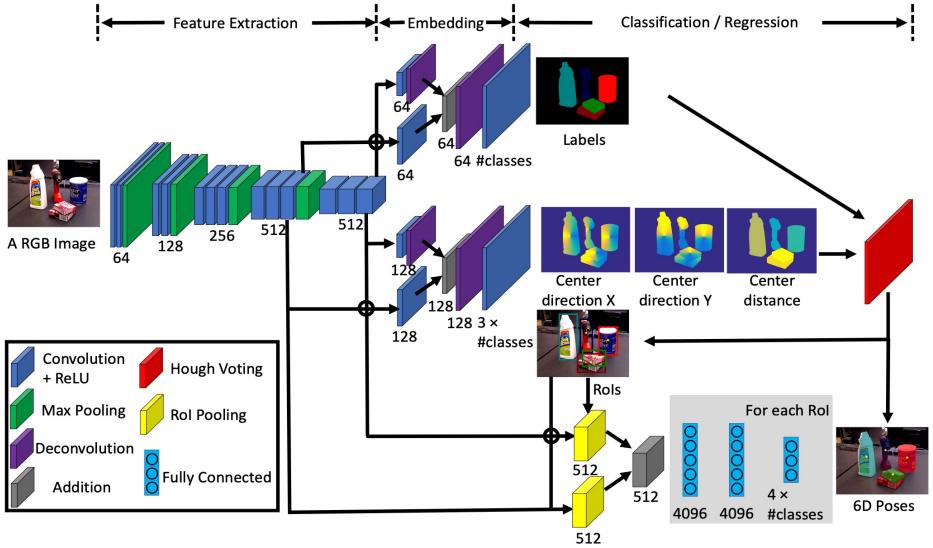
*Xiang et al. (RSS 2018) — End-to-end CNN for 6-DoF pose estimation from RGB*

## Architecture: Three Output Heads

**1. Semantic Labels**
Per-pixel object class probabilities — which object is at each pixel

**2. 3D Translation**
Hough voting: each pixel votes for the object center, then averaging estimates the 3D center

**3. 3D Rotation**
Quaternion regression — directly predicts orientation for each detected object




**Contributions**
First competitive deep 6-DoF from RGB alone. Introduced YCB-Video benchmark (21 objects, 92 sequences) — still widely used.

**Key Limitation**
Instance-specific: must retrain the entire network for each new set of objects. Cannot generalize to unseen objects at all.

# DenseFusion: Fusing RGB and Depth

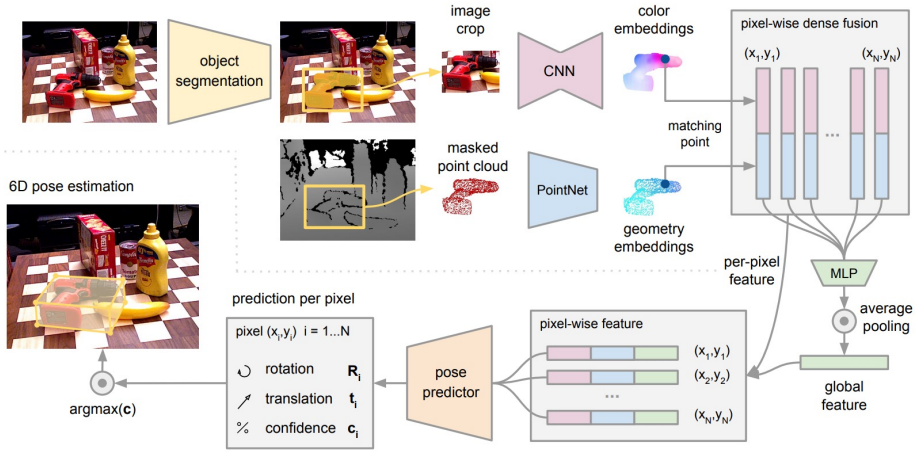
*Wang et al. (CVPR 2019) — Per-pixel two-stream fusion for robust 6-DoF pose*

## Two-Stream Architecture

**Stream 1: RGB CNN** — extracts per-pixel color/texture embeddings from cropped object region

**Stream 2: PointNet** — processes per-point 3D geometric features from depth-derived point cloud

**Dense Fusion + Iterative Refinement** — per-pixel fusion, then iterative pose refinement network

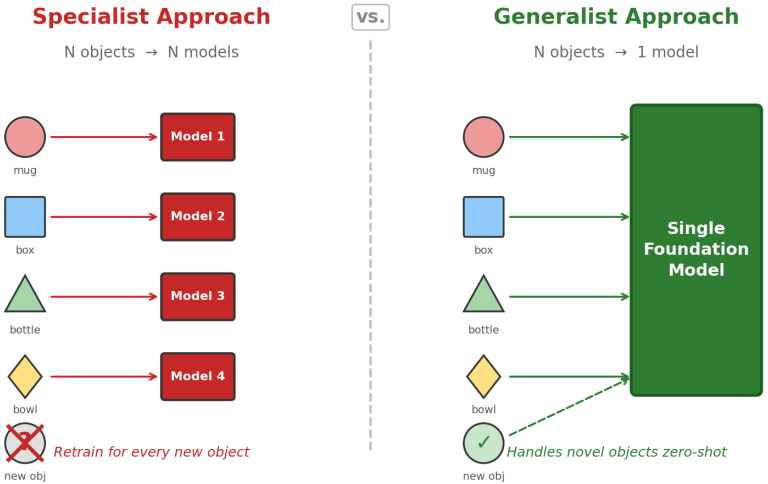



**Key Insight:** Combining RGB and depth modalities at the per-pixel level improves robustness — texture helps where geometry is ambiguous, and geometry helps where appearance is unreliable. This multimodal fusion principle extends well beyond pose estimation.

**Still instance-specific:** Like PoseCNN, DenseFusion must be retrained for each new object set. Better accuracy, same generalization barrier.

# The Generalization Problem

**Every method so far requires retraining for each new set of objects — impractical for open-world deployment.**




## The Problem in Numbers

**~100 objects**
in a typical kitchen

**Hours of training**
per object set

**Zero transfer**
to any new object

This generalization challenge parallels the journey from task-specific policies to generalist policies. In both perception and action, the field is moving from specialist systems to foundation-model-based generalists. The key enabler in both cases is training on massive, diverse data.

# MegaPose: Render and Compare at Scale

*Labbé et al. (CoRL 2022) — Generalizable pose estimation via render-and-compare*

### 1. Render Hypothesis

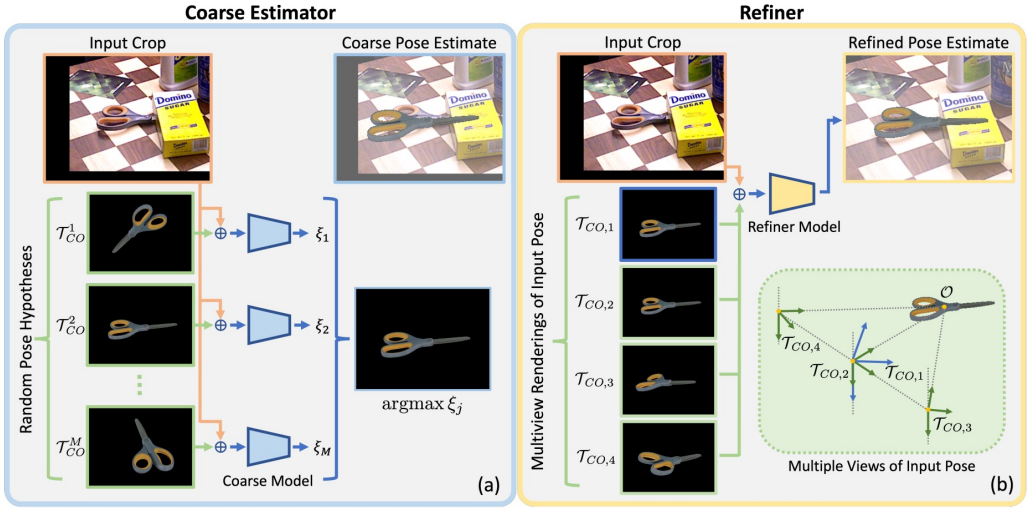
Given a CAD model, render it from many candidate poses to generate synthetic observations.

### 2. Compare

A trained network compares each rendered hypothesis against the real observation — which pose looks closest?

### 3. Refine

Iteratively update the pose estimate by rendering at the current best pose and predicting a correction.




**Critical Ablation:** Object diversity matters more than photorealism. Training on millions of diverse low-fidelity synthetic renders outperforms fewer high-fidelity renders.

**Limitation:** Model-based only — still requires a CAD model for each target object.

# FoundationPose: Unifying Model-Based and Model-Free

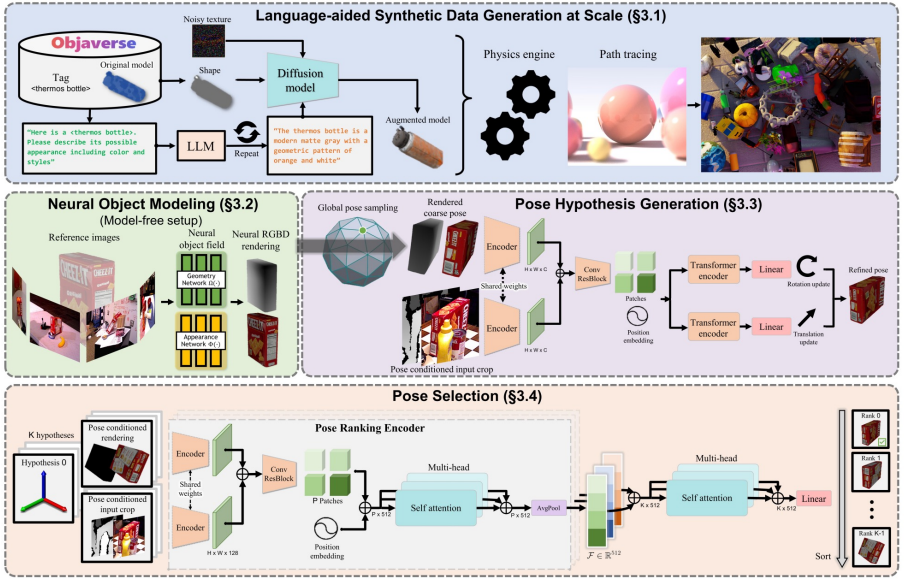
*Wen et al. (CVPR 2024)*

## Key Innovations

**Neural Implicit Bridge** Given only reference images, reconstruct a rough neural 3D model — then proceed as if a CAD model were given.

**LLM-Aided Synthetic Data** Uses language models to generate diverse object descriptions for synthetic training data at scale.

**Unified Architecture** Single model handles both model-based (CAD) and model-free (images only) setups — first to do so.




**Key Insight:** FoundationPose's neural implicit bridge between model-based and model-free setups is architecturally elegant: it asks "what if we just turn a few reference images into a rough 3D model, and then proceed as if we had a CAD model?" This shows how neural representations can unify previously separate problem formulations.

# BOP Challenge 2024: Training-Free Methods Win

## FreeZeV2.1 — Winner, Unseen Objects Track

**82.1 AR on unseen objects**
(Average Recall — the standard BOP metric)

### The Recipe

**Frozen DINOv2 Features** Pretrained vision features — no pose-specific training

**Geometric Features** Classical 3D descriptors for point correspondences

**RANSAC** Robust outlier rejection — unchanged from classical era

**ICP Refinement** Iterative closest point — the same EQ4 from Part 2

| | Date (UTC) | Method | Test image | $AR_{Core}$ | $AR_{LM-O}$ | $AR_{T-LESS}$ | $AR_{TUD-L}$ | $AR_{IC-BIN}$ | $AR_{ITODD}$ | $AR_{HB}$ | $AR_{YCB-V}$ | Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2026-03-01 | WAPR.v2 (Multi 2D Detections) | RGB-D | 0.845 | 0.781 | 0.817 | 0.961 | 0.735 | 0.798 | 0.910 | 0.915 | 50.816 |
| 2 | 2025-10-01 | FRTPose-WAPR.v2 (Multi 2D Detections) | RGB-D | 0.844 | 0.796 | 0.834 | 0.945 | 0.752 | 0.757 | 0.908 | 0.915 | 212.740 |
| 3 | 2025-05-26 | FRTPose-WAPR (Multi 2D Detections) | RGB-D | 0.840 | 0.792 | 0.826 | 0.945 | 0.743 | 0.756 | 0.907 | 0.913 | 229.326 |
| 4 | 2025-10-01 | FRTPose-WAPR.v2 (Default Detection) | RGB-D | 0.837 | 0.785 | 0.831 | 0.945 | 0.729 | 0.756 | 0.903 | 0.913 | 133.165 |
| 5 | 2025-05-26 | FRTPose-WAPR (Default Detection) | RGB-D | 0.834 | 0.777 | 0.826 | 0.945 | 0.718 | 0.758 | 0.897 | 0.913 | 116.827 |
| 6 | 2025-05-31 | FreeZeV2.2 | RGB-D | 0.833 | 0.777 | 0.799 | 0.975 | 0.711 | 0.754 | 0.895 | 0.918 | 21.709 |
| 7 | 2024-12-09 | FRTPose.v1 (SAM6D-FastSAM, NIDS, ... | RGB-D | 0.825 | 0.795 | 0.768 | 0.940 | 0.719 | 0.744 | 0.904 | 0.909 | 45.367 |
| 8 | 2025-11-02 | WAPR.v2 (Default Detections) | RGB-D | 0.824 | 0.768 | 0.826 | 0.924 | 0.707 | 0.743 | 0.891 | 0.911 | 43.068 |
| 9 | 2024-11-29 | FreeZeV2.1 | RGB-D | 0.821 | 0.771 | 0.755 | 0.976 | 0.697 | 0.742 | 0.892 | 0.915 | 24.892 |
| 10 | 2024-11-24 | FRTPose.v1 (SAM6D-FastSAM) | RGB-D | 0.818 | 0.778 | 0.766 | 0.940 | 0.702 | 0.737 | 0.896 | 0.910 | 40.090 |
| 11 | 2024-11-24 | FRTPose.v1 (Default Detections) | RGB-D | 0.818 | 0.777 | 0.763 | 0.940 | 0.705 | 0.735 | 0.896 | 0.910 | 46.533 |
| 12 | 2024-11-24 | FRTPose.v1 (MUSE) | RGB-D | 0.818 | 0.786 | 0.768 | 0.942 | 0.706 | 0.710 | 0.903 | 0.910 | 27.560 |
| 13 | 2026-02-25 | WAPR.v2.1 (MUSE) | RGB-D | 0.806 | 0.754 | 0.731 | 0.961 | 0.709 | 0.705 | 0.901 | 0.883 | 0.720 |
| 14 | 2025-09-24 | FRTPose-WAPR.v2 (MUSE) | RGB-D | 0.801 | 0.763 | 0.712 | 0.944 | 0.719 | 0.681 | 0.898 | 0.893 | 0.834 |
| 15 | 2024-09-18 | FreeZeV2 | RGB-D | 0.792 | 0.764 | 0.708 | 0.972 | 0.654 | 0.679 | 0.859 | 0.906 | 17.153 |
| 16 | 2025-05-22 | FRTPose-WAPR (MUSE) | RGB-D | 0.786 | 0.758 | 0.709 | 0.939 | 0.673 | 0.660 | 0.868 | 0.891 | 0.986 |
| 17 | 2024-09-04 | FRTPose (SAM6D-FastSAM) | RGB-D | 0.783 | 0.783 | 0.717 | 0.925 | 0.601 | 0.646 | 0.896 | 0.913 | 20.718 |
| 18 | 2024-09-07 | FRTPose (Default Detections) | RGB-D | 0.777 | 0.783 | 0.714 | 0.922 | 0.590 | 0.618 | 0.896 | 0.913 | 23.379 |
| 19 | 2024-11-25 | Co-op (F3DT2D, 5 Hypo, RGBD) | RGB-D | 0.771 | 0.738 | 0.695 | 0.929 | 0.635 | 0.629 | 0.878 | 0.898 | 6.924 |
| 20 | 2024-11-25 | Co-op (F3DT2D, 1 Hypo, RGBD) | RGB-D | 0.759 | 0.730 | 0.680 | 0.929 | 0.624 | 0.600 | 0.863 | 0.886 | 0.765 |
| 21 | 2024-11-18 | Co-op (CNOS, 5 Hypo, RGBD) | RGB-D | 0.752 | 0.730 | 0.664 | 0.905 | 0.597 | 0.613 | 0.871 | 0.887 | 7.162 |
| 22 | 2024-11-18 | Co-op (CNOS, 1 Hypo, RGBD) | RGB-D | 0.736 | 0.715 | 0.646 | 0.905 | 0.575 | 0.582 | 0.857 | 0.874 | 2.331 |
| 23 | 2024-08-27 | FoundationPose(NVIDIA) | RGB-D | 0.734 | 0.756 | 0.646 | 0.923 | 0.508 | 0.580 | 0.835 | 0.889 | 29.317 |

**Key Insight:** FreeZe's winning recipe is a remarkable callback to classical robotics: geometric features + RANSAC + ICP — with the single addition of frozen DINOv2 features replacing hand-crafted visual descriptors. The best modern pose estimation method is classical geometry powered by a foundation model backbone.

# The Emerging Frontier: 3DGS-Based Pose Estimation

*Differentiable rendering enables non-iterative pose recovery.*

## 6DGS

*ECCV 2024*

Represents objects as Gaussians, optimizes pose by differentiable splatting. Eliminates explicit correspondence matching.

## 6DOPE-GS

*ICCV 2025*

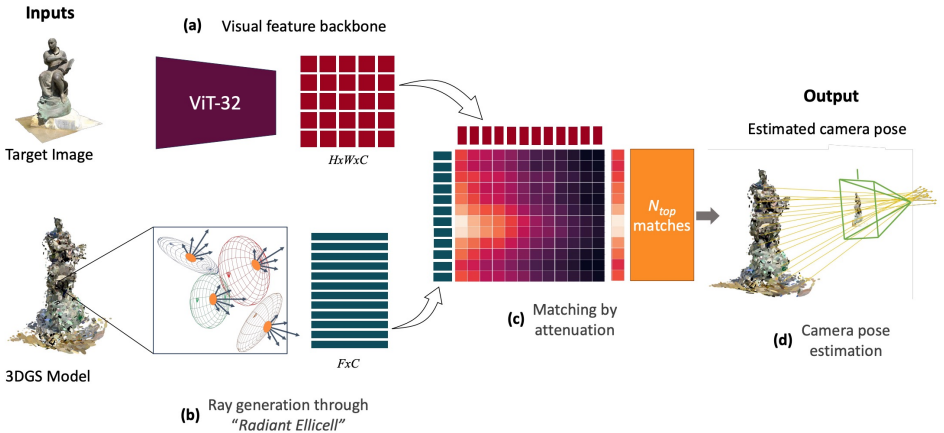
Extends 3DGS for dynamic pose estimation with occlusion-aware rendering. Handles partially visible objects.

## SAM-6D

*CVPR 2024*

Combines SAM segmentation with 6-DoF pose. Zero-shot detection and pose estimation for novel objects.

**Shared Advantage:** Differentiable rendering means pose can be optimized end-to-end by gradient descent, rather than relying on explicit feature matching → correspondence → RANSAC pipelines. This is conceptually simpler and naturally handles complex geometries.



# Pose Estimation: Lessons for Robot Learning

**1** **Geometric Reasoning Persists**
Every generation, from ICP through FreeZe, uses some form of geometric matching. The math of 3D geometry is not going away.

**2** **Render-and-Compare Is Remarkably Durable**
Introduced classically, adopted by MegaPose, refined by FoundationPose. The idea of "render a hypothesis and check" keeps working.

**3** **Foundation Model Features Enable Zero-Shot**
DINOv2 features transfer to 6-DoF pose without any pose-specific training. Frozen representations are the new hand-crafted descriptors.

**4** **Synthetic Data Diversity Drives Generalization**
Object variety matters more than rendering fidelity. This principle generalizes well beyond pose estimation to robot learning broadly.

*Next: How do we find, segment, and describe objects in open-world scenes?*

# Scene Understanding and the Perception-Action Bridge

*From Seeing Objects to Knowing What to Do with Them*

# From Closed-Set to Open-Vocabulary Perception

**Classical: Closed-Set**

Train a detector per category
Fixed vocabulary at training time
Cannot recognize unseen objects

*"I can detect the 80 COCO categories"*

Problem: a typical home has thousands of object types — retraining is impractical.

**Modern: Open-Vocabulary**

Detect anything via language query
No retraining for new categories
Zero-shot transfer to novel objects

*"Find the blue ceramic mug with the chipped handle"*

Enabled by: CLIP (2021), DINOv2 (2023), SAM (2023) — internet-scale pretraining.

# SAM and SAM 2: Segment Anything

## Evolution

**SAM (2023)** Promptable segmentation (point, box, text). Trained on SA-1B: 1.1 billion masks, 11M images.

**SAM 2 (2024)** Video segmentation. 44 FPS (6× faster). Streaming memory for temporal consistency across frames.

**SAM 2.1 (2024)** Improved occlusion handling. Better multi-object tracking in cluttered scenes.

**SAM 3 (2025)** Native text prompts — type a description, get a mask. May collapse detection + segmentation into one step.

SAM's promptable architecture — accept a hint (point, box, or text) and produce a precise mask — mirrors how robot policies receive instructions. A language-conditioned policy says "pick up the red cup"; the perception system needs SAM-like capabilities to locate and segment that cup.

*SAM is the segmentation backbone for most modular robot perception systems today.*

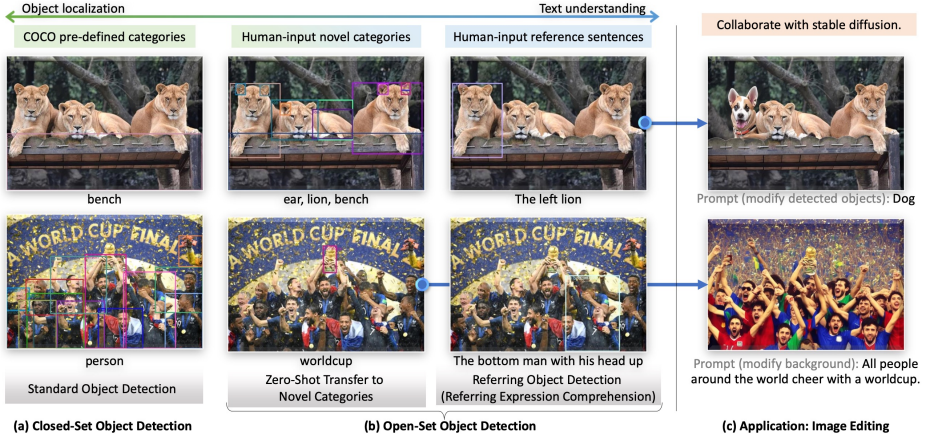
# Grounding DINO: Open-Set Detection

*Liu et al. (ECCV 2024) — Detect any object described by text*

## How It Works

Input a text query ("red cup", "screwdriver"), get bounding boxes with confidence scores. Combines a text encoder with an image encoder via cross-attention, then predicts object locations conditioned on the language.

**52.5 AP zero-shot on COCO** — competitive with supervised detectors




## Model Variants

| Model | Key Feature | Use Case |
|---|---|---|
| **Grounding DINO** | Open-set detection via text queries | General-purpose detection |
| **G-DINO 1.5 Pro** | Higher accuracy, larger model | Cloud/server deployment |
| **G-DINO 1.5 Edge** | Optimized for on-device inference | Real-time robot perception |
| **OWLv2** | Google's open-vocabulary detector | Alternative (used in OK-Robot) |
| **YOLO-World** | Real-time open-vocabulary YOLO | Speed-critical applications |

*Grounding DINO is the detection backbone — but detection alone isn't enough. We also need precise segmentation masks.*

# The Grounded SAM Pipeline

*De facto perception frontend for modular robot systems*

**Grounding DINO**

Text query → bounding boxes

"Pick up the red cup"
→ box around the cup

→

**SAM / SAM 2**

Boxes → precise masks

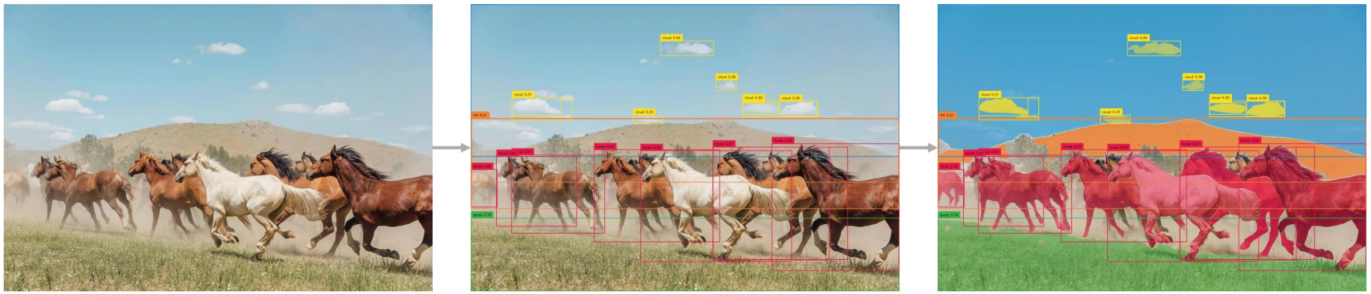
Pixel-level segmentation
for each detected object

→

**Output**

Object identity + mask

Ready for grasping,
manipulation, or navigation



**Text Prompt:**
"Horse. Clouds. Grasses. Sky. Hill."

**Grounding DINO:**
Detect Everything

**Grounded-SAM:**
Detect and Segment Everything

**Modular**
Components are swappable

**Zero-Shot**
No task-specific training

**Evolving**
SAM 3 may merge both stages

*This two-stage pipeline is the perception backbone behind OK-Robot, VoxPoser, and many other modular systems.*

# Integration Example: OK-Robot

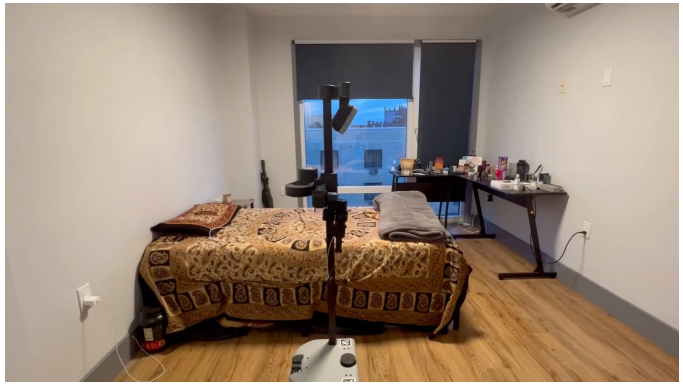
*Liu et al. (RSS 2024) — Open Knowledge Robot for Mobile Manipulation in Real Homes*

| OWL-ViT | CLIP Memory | Navigation | AnyGrasp |
|---|---|---|---|
| **Detection** | **Semantic Map** | **Motion** | **Grasping** |
| Open-vocabulary object detection from language query | Build a spatial memory of where objects are in the home | Navigate to the target object location using the map | Grasp the detected object using a generalizable grasp planner |

**Platform:** Hello Robot Stretch RE1 — low-cost mobile manipulator deployed in 10 real homes.

**58.5% success** in real homes (1.8× prior SOTA)

OK-Robot demonstrates both the power and the limits of modular perception. Open-vocabulary models enable operation in unseen homes, but their imperfect accuracy propagates through the entire system. This motivates the trend toward end-to-end perception in VLAs (Lecture 10), where the policy learns to extract exactly the visual features it needs.



# Affordance Detection: Where Can I Act?

## Where2Act

*ICCV 2021*

Per-point action success prediction from simulation. Learns where to push, pull, or grasp by training on 3D shapes with contact-based feedback.

**Dense affordance maps**

## SayCan

*CoRL 2022*

LLM "Say" (what makes sense linguistically) + affordance "Can" (what's physically possible). Grounds language in real-world feasibility.

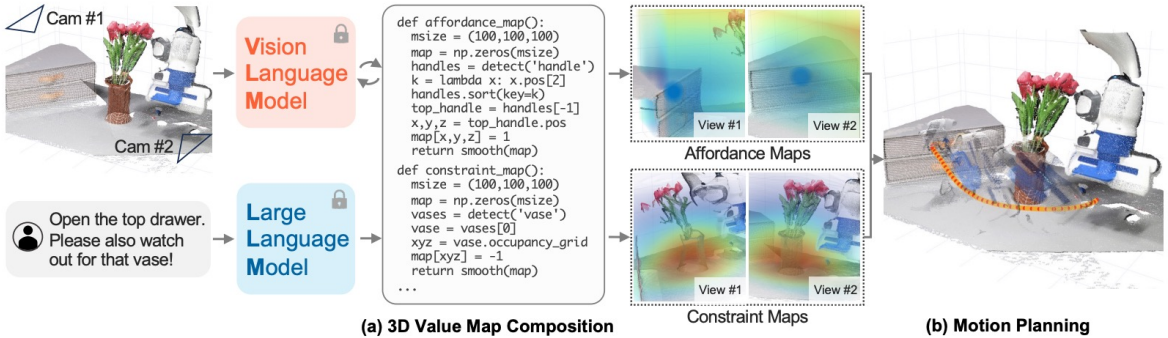
**84% plan success**

## VoxPoser

*CoRL 2023*

LLMs compose 3D voxel value maps as affordances and constraints. Directly converts language instructions into spatial cost functions for planning.

**70–90% zero-shot success**




*The evolution: explicit affordance maps (Where2Act) → language-grounded feasibility (SayCan) → spatial value fields from LLMs (VoxPoser). Next: affordance moves inside the policy itself.*

# Affordance Inside VLAs: CoA-VLA and A0

### CoA-VLA

Chain-of-Affordance: four affordance types (contact point, pre-contact pose, grasp, trajectory) as language intermediate outputs before action

**Explicit affordance reasoning**

### A0

Embodiment-agnostic affordance understanding + embodiment-specific execution. Separates "what to do" from "how to do it" — affordance transfers across robot

**Cross-embodiment transfer**

### UAD

Unsupervised Affordance Discovery (ICRA 2025 Best Paper Finalist). Learns affordances without manual labels by discovering action-relevant

**No manual annotation**

**Key Insight:** The trend is clear: affordance is moving from a separate perception module to an internal reasoning step within end-to-end policies. Just as pose estimation moved from explicit estimation to implicit encoding within visual encoders, affordance is being absorbed into the policy itself — while still benefiting from explicit affordance reasoning (CoA-VLA) during the transition.

## Standalone Modules → Integrated into Policy

Where2Act (2021): separate module → SayCan/VoxPoser: LLM interface → CoA-VLA/A0: inside the policy

# Synthetic Data for Perception Training

**Core Principle:** Object diversity matters more than photorealism. Confirmed by MegaPose and FoundationPose ablations — more diverse low-fidelity renders outperform fewer high-fidelity renders.

## Evolution of Synthetic Data

**Domain Randomization (2017)** Tobin et al. — randomize textures, lighting, camera poses. The original insight: diversity beats realism.

**BlenderProc 2 (2020+)** Procedural generation pipeline for BOP benchmark training data. Standardized synthetic data generation.

**Infinigen (2023)** Procedural generation of entire photorealistic worlds — infinite diversity in geometry, materials, and layout.

**Cosmos Transfer (2025)** Generative model-based augmentation — transform synthetic renders into photorealistic images. Best of both worlds.

**LLM-Aided Composition (2024+)** Language models generate diverse scene descriptions and object placements. Used by FoundationPose for training data at scale.

# 3D Language Fields: LERF, LangSplat, ConceptFusion

## LERF

### CLIP-in-NeRF

Embeds CLIP features into a NeRF — query a 3D scene with language and get spatial relevance maps.

⚠ *Slow (NeRF rendering)*

## LangSplat

### Language-in-3DGS

CLIP features stored in Gaussians. Real-time (100+ FPS). SAM-based hierarchical semantics for multi-scale queries.

⚠ *Requires scene reconstruction*

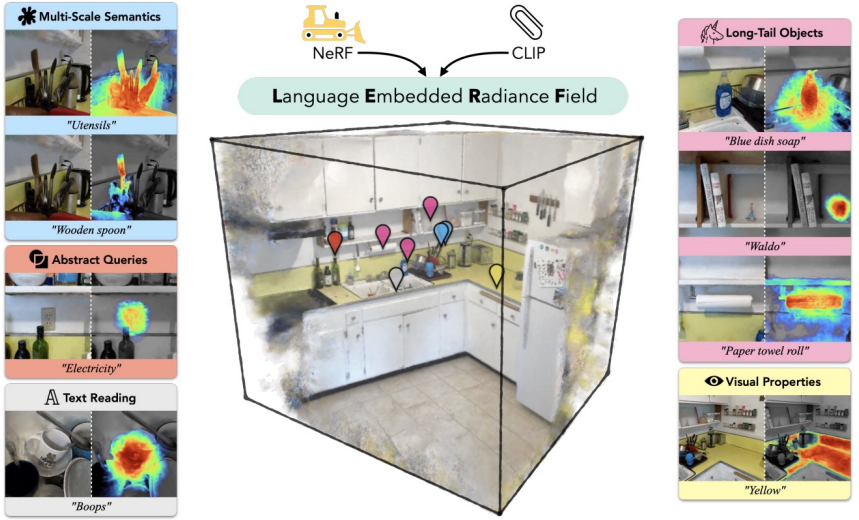
## ConceptFusion

### Multimodal 3D

Fuses features from multiple foundation models (CLIP, AudioCLIP) into a 3D map. Text, image, or audio queries.

⚠ *Complex fusion pipeline*

3D language fields represent the convergence of all the themes in this lecture. They enable a robot to look at a scene and answer "where is the thing the person asked about?" directly in 3D — the essential interface between language instructions and spatial action.



# Perception Meets Policy Learning

*What Should the Robot's Policy Actually See?*

# Visual Encoders in Robot Learning Policies

| Policy | Year | Visual Encoder | How Used | Notes |
|---|---|---|---|---|
| **RT-1** | 2022 | FiLM-EfficientNet-B3 + TokenLearner | End-to-end trained | 130k episodes needed |
| **RT-2** | 2023 | PaLI-X (ViT-22B) | Frozen VLM backbone | Actions as text tokens |
| **Diff. Policy** | 2023 | ResNet-18 | End-to-end trained | Lightweight, fast |
| **Octo** | 2024 | Transformer (scratch) | Trained on robot data | 800k trajectories |
| **OpenVLA** | 2024 | DINOv2 + SigLIP (fused) | Frozen dual backbone | 7B parameters |
| **$\pi_0$** | 2024 | PaliGemma (SigLIP) | Frozen VLM backbone | Flow matching expert |
| **HPT** | 2024 | Flexible stems → 16 tokens | Multiple frozen stems | Heterogeneous pretrain |

**Trajectory:** Task-specific CNNs (2022) → frozen pretrained VLMs (2024). The field converged on leveraging internet-scale visual pretraining rather than learning visual features from robot data alone.

Color coding — How Used: **End-to-end** **Frozen** **Robot-data trained**

# Frozen vs. Fine-Tuned vs. End-to-End

**Frozen**

**< 100 demos**

Use pretrained encoder as-is. Zero risk of catastrophic forgetting. Best when data is scarce.

*RT-2, OpenVLA, $\pi_0$*

**LoRA Fine-Tune**

**100 – 1k demos**

Low-rank adaptation of pretrained weights. Preserves pretraining while adapting to robot domain.

*Theia, many VLA variants*

**End-to-End**

**10k+ demos**

Train entire encoder on robot data. Maximum task specificity but requires massive data.

*RT-1, Diffusion Policy, Octo*

## Key Research Findings

VC-1 / CortexBench: No single frozen encoder dominates all tasks — task requirements differ.

Theia: Spatial tokens outperform CLS tokens for manipulation; fine-tuned encoders outperform fully frozen.

| | | Imitation Learning | | | | | Reinforcement Learning | | Mean | |
|---|---|---|---|---|---|---|---|---|---|---|
| # | Model | Adroit | MetaWorld | DMControl | TriFinger | ObjectNav | ImageNav | Mobile Pick | Rank | Success |
| 1 | Best prior result (any setting) | 75 | 80 | 77 | - | 70.4 | 82.0 | - | | |
| 2 | Best prior result (Frozen PVR) | 75 | 80 | 77 | - | 54.4 | 61.8 | - | | |
| 3 | Random (ViT-B) Frozen | 2.0 ± 2.0 | 0.5 ± 0.5 | 10.1 ± 0.6 | 57.8 ± 0.5 | 19.2 ± 0.9 | 42.1 ± 0.8 | 10.8 ± 1.4 | 7.2 | 20.4 |
| 4 | Random (ViT-L) Frozen | 2.7 ± 1.8 | 0.5 ± 0.5 | 9.1 ± 0.2 | 57.2 ± 0.9 | 19.3 ± 0.9 | 45.2 ± 0.8 | 20.6 ± 1.8 | 6.9 | 22.1 |
| 5 | Random (ViT-B) Fine-tuned | 44.0 ± 2.0 | 49.9 ± 7.3 | 43.5 ± 2.4 | 56.1 ± 1.3 | 28.5 ± 1.0 | 62.5 ± 0.7 | 47.6 ± 2.2 | 5.3 | 47.4 |
| 6 | MVP (ViT-B) | 48.0 ± 3.3 | 91.2 ± 2.9 | 65.9 ± 2.4 | 59.7 ± 0.3 | 51.2 ± 1.1 | 64.7 ± 0.7 | 56.0 ± 2.2 | 3.1 | 62.4 |
| 7 | MVP (ViT-L) | 53.3 ± 4.1 | 87.5 ± 3.4 | 69.2 ± 1.5 | 74.1 ± 0.3 | 55.0 ± 1.1 | 68.1 ± 0.7 | 65.4 ± 2.1 | 2.1 | 67.5 |
| 8 | CLIP (ViT-B) | 47.3 ± 3.0 | 75.5 ± 3.4 | 55.5 ± 1.4 | 62.0 ± 0.5 | 56.6 ± 1.1 | 52.2 ± 0.8 | 49.8 ± 2.2 | 3.9 | 57.0 |
| 9 | VIP (RN-50) | 54.0 ± 4.8 | 90.1 ± 2.2 | 72.5 ± 2.7 | 66.7 ± 0.2 | 26.4 ± 1.0 | 48.8 ± 0.8 | 7.2 ± 1.2 | 4.0 | 52.3 |
| 10 | R3M (RN-50) | 73.3 ± 2.0 | 96.0 ± 1.1 | 81.1 ± 0.7 | 69.2 ± 0.8 | 22.7 ± 0.9 | 30.6 ± 0.7 | 33.2 ± 2.1 | 3.4 | 58.0 |

# The Observation Space Decision

← Low Geometric Precision

High Geometric Precision →

**RGB + Pretrained VLM**

*High semantic richness, low geometric precision*

Best for: language-conditioned tasks, semantic reasoning, large-scale training. Used by: RT-2, OpenVLA, $\pi_0$

**Fused 2D + 3D ★**

*High semantic + high geometric = best of both*

Best for: general-purpose manipulation. 2025 consensus direction. Used by: Act3D, emerging VLAs.

**Proprioception Only**

*No vision — joint states, forces, torques*

Best for: repetitive tasks with fixed objects, contact-rich assembly. Limited generalization.

**Point Clouds / Depth**

*High geometric precision, low semantic richness*

Best for: contact-rich manipulation, spatial precision. OBSBench winner. Used by: PerAct, Act3D.

**Key Insight:** The observation space decision has four parts: what sensor, what representation, what encoder, and how to train the encoder. Today's lecture gave you the vocabulary and evidence to make these decisions. The next four lectures will show you how these choices play out in RL, imitation learning, diffusion policies, and VLAs.

# Perception Design Checklist

1. **Sensor Selection** What depth quality do you need? Does your downstream model expect RGB only? Budget constraints?

2. **Representation Choice** Point cloud for spatial manipulation, RGB for semantic tasks, fused for best generality.

3. **Encoder Selection** Match to your data budget. <100 demos → frozen VLM; 100–1k → LoRA fine-tune; 10k+ → end-to-end.

4. **Observation Design** Single-view or multi-view? Include proprioception? Wrist camera + overhead camera?

5. **Debugging Protocol** When a policy fails, check perception first. Visualize what the model sees.

**Key Insight:** When a policy fails, check perception first. Most robot learning failures are perception failures in disguise — wrong detections, noisy depth, poor segmentation, or distribution shift between training and deployment cameras.