Open-Vocabulary pose estimation using implicit feature fields

1 Types of tasks

We categorize the pose estimation tasks into following categories:

- `pose-end`: where the final goal is to the reach the specified pose, *e.g.* "Go and stand near the chair facing the dining table"
- `downstream`: where pose inference is an intermediate (but crucial) step which is followed by a downstream task
  - `downstream-vqa`, *e.g.* "What magazine is kept on the table?" where the robot needs to first get to the correct pose to be able to see the magazine on the table
  - `downstream-action`, *e.g.* "Press the elevator button to go up." where the robot first has to reach the appropriate pose in the elevator to be able to press the button
  - `downstream-dynamic` abbreviated as `downstream-dyn`, *e.g.* "Wait in the lobby for a visitor to come by and then greet them" where the thing of interest is not immediately visible, but need to reason about the best pose where the robot would be most likely to see that when it happens

Each of these can lead to a single unique pose (*e.g.*, "Go and pick up the mug on table" is unambiguous) or multiple poses (*e.g.*, "Go and pick up the mug" is possibly ambiguous), and the algorithm should be able to output *all*.

2 Proposed Approach

2.1 Scene Representation

We will use a distilled feature field representation as our scene prior S, with 2D CLIP features distilled into a NeRF-like MLP. Concretely, given a 3D location $\mathbf{x}$ and a viewing direction $\mathbf{d}$, the feature field F_θ predicts the RGB color (radiance), the volumetric density at the location, and a D-dim CLIP feature at the location.

$$(\mathbf{x}, \mathbf{d}) \xrightarrow{F_\theta} \big(\mathbf{c}(\mathbf{x}, \mathbf{d}) \in \mathbb{R}^3,\ \sigma(\mathbf{x}) \in \mathbb{R}_{\geq 0},\ \mathbf{f}(\mathbf{x}) \in \mathbb{R}^D\big) \tag{1}$$

Once F_θ is learned using a process similar to [1], the way F_θ is used to render a RGB-image and a feature image for any camera pose is as follows. Given a camera pose $\mathbf{T}_c = (\mathbf{R}_c, \mathbf{t}_c) \in SE(3)$ and camera intrinsics $\mathbf{K}_c \in \mathbb{R}^{3\times3}$, each pixel $\tilde{\mathbf{p}}_i = (u_i, v_i, 1)^\top$ corresponds to a single ray $\mathbf{r}_i$ with origin $\mathbf{o}_i$ and direction $\mathbf{d}_i$ given as:

$$\mathbf{r}_i(t) = \mathbf{o}_i + t\,\mathbf{d}_i \tag{2}$$

$$\text{where} \quad \mathbf{o}_i = -\mathbf{R}_c^\top \mathbf{t}_c \tag{3}$$

$$\text{where} \quad \mathbf{d}_i = \frac{\mathbf{R}_c^\top \mathbf{K}_c^{-1}\,\tilde{\mathbf{p}}_i}{\big\|\mathbf{R}_c^\top \mathbf{K}_c^{-1}\,\tilde{\mathbf{p}}_i\big\|_2}, \quad \tilde{\mathbf{p}}_i = (u_i, v_i, 1)^\top \tag{4}$$

Now, using standard volume rendering technique, we render the color $\hat{\mathbf{C}}(\mathbf{r}_i)$ and the feature $\hat{\mathbf{F}}(\mathbf{r}_i)$ for this ray (and hence the pixel), and doing this for all pixels, renders the entire RGB image $\hat{\mathbf{C}}_I$ and feature image $\hat{\mathbf{F}}_I$:

$$\hat{\mathbf{C}}(\mathbf{r}_i) = \int_{t_n}^{t_f} T(t)\,\sigma\big(\mathbf{r}_i(t)\big)\,\mathbf{c}\big(\mathbf{r}_i(t), \mathbf{d}\big)\,dt \tag{5}$$

$$\hat{\mathbf{F}}(\mathbf{r}_i) = \int_{t_n}^{t_f} T(t)\,\sigma\big(\mathbf{r}_i(t)\big)\,\mathbf{f}\big(\mathbf{r}_i(t)\big)\,dt \tag{6}$$

$$\text{where} \quad T(t) = \exp\!\left(-\int_{t_n}^{t} \sigma\big(\mathbf{r}_i(s)\big)\,ds\right) \tag{7}$$

where integral is performed over the near and far bounds t_n and t_f.

Algorithm 1 Overall pipeline v0

Require: Prompt L, Scene S, Judge J
Ensure: Optimal poses p^* (one or many)

$\{\mu_{\text{prop}}\}$, $M_{\text{task}} \leftarrow$ GetTaskClustersMasks(L, S, J) ▷ task-level execution
$p^* \leftarrow [\,]$ ▷ initialize optimal poses list
for $\mu_p \in \{\mu_{\text{prop}}\}$ **do**
    $\pi^*_{\text{cluster}} \leftarrow$ SynthClusterCostProgram(L, S, J, μ_p, M_{task}) ▷ synthesize cost program for cluster
    p^*.append(CMAES(π^*_{cluster}, S)) ▷ optimize program using CMA-ES
end for
return p^* ▷ return optimal poses

Algorithm 2 Overall pipeline v1 (CEGIS-style feedback)

Require: Prompt L, Scene S, Judge J
Ensure: Optimal poses p^* (one or many)

$\{\mu_{\text{prop}}\}$, $M_{\text{task}} \leftarrow$ GetTaskClustersMasks(L, S, J) ▷ task-level execution
$p^* \leftarrow [\,]$ ▷ initialize optimal poses list
for $\mu_p \in \{\mu_{\text{prop}}\}$ **do**
    $\delta^g_f \leftarrow$ None ▷ initialize feedback
    repeat
        $\pi^*_{\text{cluster}} \leftarrow$ SynthClusterCostProgram(L, S, J, μ_p, M_{task}, δ^g_f) ▷ synthesize cost program
        $p \leftarrow$ CMAES(π^*_{cluster}, S) ▷ optimize program using CMA-ES
        δ^g_b, $\delta^g_f \leftarrow J(L, S, \pi^*_{\text{cluster}}, p)$ ▷ judge returns bool & feedback
    until δ^g_b
    p^*.append(p)
end for
return p^* ▷ return optimal poses

2.2 Algorithm

- cluster μ has all information about it: size, location, object feature etc. Think of it as an abstract class
- J is a multi-modal LLM (such as Molmo) used as a verifier / repair agent / judge
- the actual mechanism of providing feedback could be just a corrective action to take in form of NL, or an actual counter-example where it failed. This is feasible since both judge and synthesizer are multi-modal LLMs.

3 Relevant work

Table 1 summarizes the most similar works to open-vocabulary pose computation that use some kind of scene prior[1]. Some of these methods are used as baselines (section 4.1) and described there again.

The main takeways: (1) None of the current methods handle `downstream-x` type tasks; (2) the current methods predict the pose in a single feed-forward inference step (*e.g.* calculate relevancy/text-similarity map and choose the highest match) – there's no feedback loop or iterative refinement in the computation.

4 Proposed Evaluation

4.1 Baselines

Initial plan is to use mainly four baselines (not counting ablations):

1. **VLMaps** (Figure 1): It represents the scene in a top-down semantic map, where each grid cell has an associated feature vector that can be used for NL-based similarity querying. It focuses on tasks like "Navigate

[1]here, the *'using scene prior'* part separates it from RGB-only vision language navigation area

Algorithm 3 GetTaskClustersMasks

Require: Prompt L, Scene S, Judge J
Ensure: Proposed clusters $\{\mu_{\text{prop}}\}$ and task-level masks M_{task} (e.g., *floor* mask)

$\delta_f \leftarrow$ None ▷ initialize feedback
repeat
  $\pi_{\text{task}} \leftarrow \textsc{SynthTaskProgram}(L, S, \delta_f)$ ▷ synthesize task-level program
  $\{\mu_{\text{prop}}\}, M_{\text{task}} \leftarrow \pi_{\text{task}}(S)$ ▷ extract proposed clusters & masks
  $\delta_b, \delta_f \leftarrow J\big(\{\mu_{\text{prop}}\}, \pi_{\text{task}}, L, M_{\text{task}}, S\big)$ ▷ judge returns bool & feedback
until δ_b
return $\{\mu_{\text{prop}}\}, M_{\text{task}}$ ▷ return clusters and task masks

Algorithm 4 SynthClusterCostProgram v0

Require: Prompt L, Scene S, Judge J, proposed cluster μ_p, task-level masks M_{task}
Ensure: Cluster cost program π^*_{cluster}

$O \leftarrow \textsc{ExtractLocalObjectives}(L, S, \mu_p)$ ▷ extract NL objectives for this cluster
$\pi_{\text{costs}} \leftarrow [\,]$ ▷ initialize list of individual cost programs
$\{p_i\} \leftarrow \textsc{GenerateFewExamplePoses}(S, \mu_p)$ ▷ get some sample poses around cluster for verification
for all $o \in O$ **do**
  $\{\delta_{f,i}\} \leftarrow \{\text{None}\}$ ▷ initialize per-sample-pose feedback
  repeat
    $\pi \leftarrow \textsc{SynthCostProgram}\big(L, S, \mu_p, \{\delta_{f,i}\}\big)$ ▷ synthesize cost program for objective
    $\{c_i\} \leftarrow \pi(\{p_i\}, S)$ ▷ evaluate cost program on sample poses
    $\{\delta_{b,i}\}, \{\delta_{f,i}\} \leftarrow J\big(o, \pi, \{p_i\}, \{c_i\}, S\big)$ ▷ judge returns per-sample boolean & feedback
  until all $\delta_{b,i}$ ▷ repeat until every sample is judged as correct
  π_{costs}.append(π)
end for
$\pi^*_{\text{cluster}} \leftarrow \textsc{Compose}\big(\pi_{\text{costs}}, M_{\text{task}}\big)$ ▷ combine individual cost programs and task masks
return π^*_{cluster} ▷ return final cluster cost program

to the keyboard and the laptop twice" and "move in between the laptop and the bowl", and leverages a predefined DSL that specifies the meaning of spatial relations (*e.g.* hardcoded definition of in-between).

2. **CLIP-Fields** (Figure 2): It represents the scene as an implicit CLIP-based feature field. They test it on tasks such as "Go and look at X" and it leverages the distilled clip features to do NL-based similarity matching to figure out the object location and then the robot navigates to the nearby location and points the camera at the object.

3. **MobilityVLA** (Figure 3): It represents the scene as a connected graph of image frames, and utilizes a VLM as a high-level planner agent to determine the goal pose from NL query. They test it on tasks like "Take me to hand-sanitizer machine".

4. **Robopoint-feature-field based sampling**: This is the method presented last week (link[2]). In brief, it distills CLIP as well as Robopoint features and uses a sampling-based approach to determine the optimal pose.

4.2 Benchmarks

Although there is no single benchmark that exactly does what we want, we can adapt the existing benchmarks to our task and perform *a part* of the evaluation.

1. **PASTURE** and **HM3D-OVON**: These are both from the object-navigation literature, where there is a subset of tasks with the form "Go to the coffee table near the chair" which can be cast as a `pose-end` task

2. **OpenEQA-active**: OpenEQA is a embodied-QA benchmark, of which, there is a subset called OpenEQA-active where the agent has to take exploratory steps in the environment to answer the query, *e.g.* "Do we have

[2]https://docs.google.com/presentation/d/1IDunYxJMQ47euqLD5WKoRz-KMkgSSJd0Lrq67Lg6gc0/edit?usp=sharing

Algorithm 5 SynthClusterCostProgram v1 (with global feedback)

Require: Prompt L, Scene S, Judge J, proposed cluster μ_p, task-level masks M_{task}, global feedback δ_f^g
Ensure: Cluster cost program π^*_{cluster}

$O \leftarrow$ ExtractLocalObjectives$(L, S, \mu_p, \delta_f^g)$ ▷ extract NL objectives for this cluster
$\pi_{\text{costs}} \leftarrow [\,]$ ▷ initialize list of individual cost programs
$\{p_i\} \leftarrow$ GenerateFewExamplePoses(S, μ_p) ▷ get some sample poses around cluster for verification
for all $o \in O$ **do**
  $\{\delta_{f,i}\} \leftarrow \{\text{None}\}$ ▷ initialize per-sample-pose feedback
  repeat
    $\pi \leftarrow$ SynthCostProgram$\big(L, S, \mu_p, \{\delta_{f,i}\}, \delta_f^g\big)$ ▷ synthesize cost program for objective
    $\{c_i\} \leftarrow \pi(\{p_i\}, S)$ ▷ evaluate cost program on sample poses
    $\{\delta_{b,i}\}, \{\delta_{f,i}\} \leftarrow J\big(o, \pi, \{p_i\}, \{c_i\}, S\big)$ ▷ judge returns per-sample boolean & feedback
  until all $\delta_{b,i}$ ▷ repeat until every sample is judged as correct
  π_{costs}.append(π)
end for
$\pi^*_{\text{cluster}} \leftarrow$ Compose$\big(\pi_{\text{costs}}, M_{\text{task}}, \delta_f^g\big)$ ▷ combine individual cost programs and task masks
return π^*_{cluster} ▷ return final cluster cost program

Method	WHAT are they doing?									HOW are they doing it?		
	Task			outputs?		what has it been evaluated on?				scene representation		core algorithm
	category	example	domain	location	location+ orientation (pose)	pose-end	downstream-vqa	downstream-action	downstream-dyn	type	representation	
CoW-CLIP on Wheels	object navigation: navigate to a specified open-vocab object	Go to green chair near table	navigation	✓	✓	✓	✗	✗	✗	explicit	builds top-down occupancy map on-the-fly	use CLIP to match RGB image to the NL → similarity metric
VLMaps	spatial reference navigation	Go to the location between table and chair	navigation	✓	✗	✓	✗	✗	✗	explicit	top-down semantic feature map	utilize a DSL of predefined spatial relations; use semantic map for open-vocab navigation
Mobility-VLA	predict goal image frame from NL query	Take me to the cubicle	navigation	✓	✓	✓	✗	✗	✗	explicit	topological graph of image frames	use a VLM as high-level planner
CLIP-Fields / LERF	find the end goal pose where the open-vocab object is visible	Go and look at the refrigerator	navigation	✓	✓	✓	✗	✗	✗	implicit	CLIP-based distilled feature field	use relevancy/similarity maps of objects within the feature field; choose the end pose where object is visible
GeFF	open-vocab manipulation and semantic-aware navigation	Go over the grass and pick up the toy	manipulation + navigation	✓	✗	✓	✗	✗	✗	implicit	CLIP-based distilled feature field	for semantic-aware navigation, use relevancy maps for queried terrain
DFF / LERF-TOGO	predict manipulator arm grip for novel objects	soft toy	manipulation	✓	✓	–	–	–	–	implicit	CLIP-based distilled feature field	use relevancy/similarity maps of objects within the feature field; choose the grasp pose matching the demonstrated grip
Ours	open-vocab pose estimation	What is kept behind the chair?	navigation	✓	✓	✓	✓	✓	✓	implicit	CLIP-based distilled feature field	to be described later

Table 1: Comparison of open-vocabulary embodied pose computation methods.

canned tomatoes in pantry?". We can cast this as a `downstream-vqa` type task, where our pose-estimation agent has to estimate that end pose to be able to answer the downstream question. As the ground-truth end-pose is available in the benchmark, we can compare against it.

3. **ALFRED**: This is primarily a mobile manipulation benchmark for household tasks. However, a subset of it has some navigation element to it such as "Go and pick up the mug and then put it in the coffee maker", which can be cast as two separate `downstream-action` tasks, *i.e.* "Go and pick up the mug" and "Put the mug in coffee maker". Again, as the ground-truth end pose is available for both, we can evaluate our pose-estimation agent's capability in predicting the correct end pose.

Since, (1) there is no available benchmark for `downstream-dyn` type tasks, and (2) no dedicated benchmark for pose-estimation problem in general even for the other type of tasks, the proposed plan is to evaluate our proposed agent on a new benchmark in addition.

5 References

[1] Shen, William, Ge Yang, Alan Yu, Jansen Wong, Leslie Pack Kaelbling, and Phillip Isola. "Distilled Feature Fields Enable Few-Shot Language-Guided Manipulation." In CoRL. 2023.

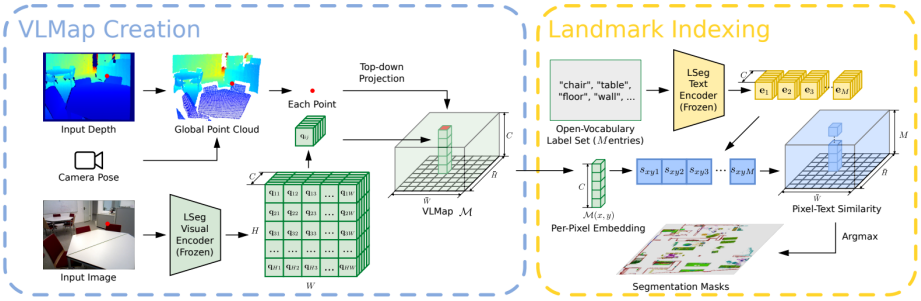

Figure 1: VLMaps

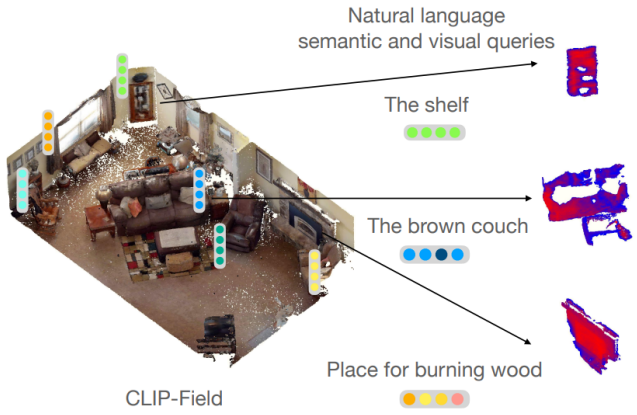

Figure 2: CLIP-Fields

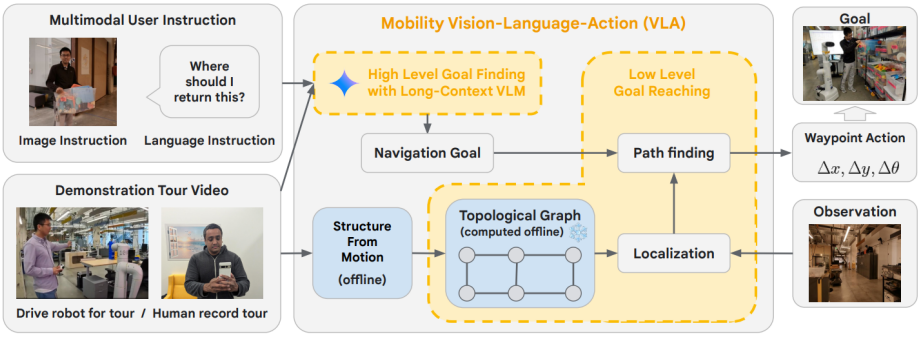

Figure 3: MobilityVLA