DeepRob Final Project Report: Project Title

Yuzhen Chen
College of Engineering
University of Michigan
Ann Arbor, Michigan
yuzhench@umich.edu

Yeheng Zong
College of Engineering
University of Michigan
Ann Arbor, Michigan
yehengz@umich.edu

Anran Li
College of Engineering
University of Michigan
Ann Arbor, Michigan
anranli@umich.edu

Jiamu Liu
College of Engineering
University of Michigan
Ann Arbor, Michigan
dliujm@umich.edu

***Abstract*—In this project, we present a real-time, language-guided 6D object pose estimation pipeline that integrates modern vision-language models with classical 3D geometric techniques. Our pipeline begins with a fine-tuned Grouding-DINO model for language-guided object localization and a fine-tuned Segment Anything Model 2 (SAM2) for instance segmentation. To achieve accurate 3D reconstruction, we performed extrinsic calibration between an Azure Kinect RGB-D camera and a motion capture system. Point cloud data are generated from the depth information provided by Kinect, from which points belonging to the target object are selected by back-projecting the segmented 2D image region. This is followed by key-point extraction, FPFH-based correspondence matching, and coarse pose estimation using RANSAC-based algorithms. The final pose refinement is then obtained through an Iterative Closest Point (ICP) algorithm. We validate the full pipeline through experiments and report pose estimation accuracy and performance rate, demonstrating the system's ability to operate effectively in real-time.**

The project page is available at: https://anranli2003.github.io/DeepRobFinal/.

I. Introduction

Accurate and efficient 6D object pose estimation is a critical capability in robotics and a popular research topic due to its fundamental role in a robot's ability to understand and interact with the physical world. Estimating an object's translation and orientation in 3D space is essential for tasks such as manipulation, navigation, augmented reality, and scene understanding. Significant progress has been made in recent years, but real-time pose estimation in unstructured and dynamic environments still remain a challenging problem.

Traditional approaches often depend on prior knowledge, such as the CAD model or categorical information of the target object, which limits their application in real-world scenarios. While recent deep learning methods have improved drastically in generalizability to novel objects and robustness to noise, they introduced new challenges, such as high computational costs, dependence on manual inputs, and slow inference times.

In this paper, we propose a language-guided, real-time 6D pose estimation pipeline. Our framework incorporates modern vision-language models for target localization, which eliminates the requirement of knowing the object's category and the reliance on manual segmentation. By integrating fine-tuned Grounding-DINO for language-driven object detection, SAM2 for instance segmentation, and efficient 3D point cloud processing algorithms, such as key-point matching, RANSAC-based coarse alignment, and ICP refinement, we achieve a scalable pipeline capable of running at interactive speeds (5 Hz).

This report entails the design and implementation of our system, as well as our experimental setup to evaluate performance. It will demonstrate how combining semantic guidance with geometric methods can yield accurate pose estimations at a speed suitable for real-world robotics applications.

II. Related Work

Recent advancements in 6D object pose estimation have focused on improving generalization to novel objects, robustness to occlusion and other challenging conditions, and efficiency in both inference and 3D reconstruction. Some most recent approaches in this domain include MegaPose, BundleSDF, and GenPose++, where each has its unique methods and limitations.

MegaPose focuses on 6D pose estimation of novel objects (objects that have never been seen by the model during training) using a render-and-compare strategy [1]. It requires the CAD model of the interested object at inference time. Its pipeline consists of a coarse pose estimator that compares rendered pose hypotheses against the observed object in image and a pose refinement network which iteratively improves the pose estimation. Both components have been trained on a dataset of millions of synthetic images of diverse 3D models. MegaPose achieved breakthrough in generalization to novel objects and performs well in cluttered or occluded environments. However, one of its drawbacks is that MegaPose's performance is heavily dependent on the accuracy of the coarse pose estimator. If the coarse estimator's initial guess has too large of an error, the refinement network often fails to correct it. Additionally, the coarse stage is computationally intensive and cannot be deployed in real time. In the original paper, the authors reported that it took 2.5 seconds for the coarse estimator to render and evaluate 520 pose hypotheses.

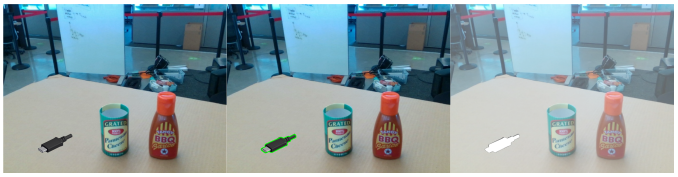

Fig. 1. Megapose6D output visualization on USB-C

BundleSDF proposes a framework of simultaneous real-time 6D pose tracking and 3D reconstruction of unknown objects using RGB-D videos [2]. This approach requires no CAD model of the object, and it only requires a segmentation mask in the first frame to track an object's motion through a video. BundleSDF contains two simultaneous processes: an online pose graph estimation for object tracking and a Neural Object Field (NOF) for learning the object's signed distance field (SDF). A key innovation of BundleSDF is it's hybrid SDF representation, which allows it to perform better under subpar conditions such as occlusion and noisy segmentation. A dynamic memory pool of keyframes was also adopted by the authors which mitigated tracking drift often seen in traditional approaches. A primary limitation of BundleSDF is that it is reliant on an external source to provide a 2D segmentation mask at the start of the video. This missing link in the pipeline could introduce inefficiency and reduce the level of automation. Furthermore, BundleSDF demands significant hardware resources, making it difficult to deploy on edge devices.

GenPose++ is a diffusion-based framework for category-level 6D pose estimation and tracking of unknown objects [3]. The model uses a probabilistic diffusion process which samples pose hypotheses instead of regressing a single pose. This approach allows GenPose++ to handle ambiguous cases such as object symmetries, occlusions, and varying materials. A key feature of GenPose++ is its semantic-aware feature extraction stage, where geometric information from depth data is integrated with semantic features derived other vision transformers from RGB images. GenPose++ generalizes across diverse object categories and achieved state-of-the-art accuracy. However, its reliance on diffusion-based processes lead to slow performance at inference time, limiting its application potential in real-time systems. Similar to BundleSDF, GenPose++ has high complexity, resulting in a high computational demands.

III. Algorithmic Extension

IV. Method Overview

The pipeline follows a layered design: *language* → *image* → *point cloud* → *pose*. Below we detail each component.

A. Sensor Layout and Extrinsic Calibration

We use an Azure Kinect (RGB-D), an AprilTag rigidly attached to the robot wrist, and an optical MoCap system. MoCap yields the tag pose in the world frame, ${}^{w}\mathbf{T}_t^{(k)}$, while EPnP on the image corners $\mathbf{x}_i^{(k)}$ gives ${}^{c}\mathbf{T}_t^{(k)}$. The camera extrinsics ${}^{w}\mathbf{T}_c$ are solved via a batched Procrustes least-squares

$$\min_{{}^{w}\mathbf{T}_c} \sum_k \sum_i \left\| {}^{w}\mathbf{T}_c \, {}^{c}\mathbf{T}_t^{(k)} \mathbf{X}_i - {}^{w}\mathbf{T}_t^{(k)} \mathbf{X}_i \right\|_2^2, \tag{1}$$

followed by an SVD projection to ensure $\mathbf{R} \in \mathrm{SO}(3)$. Calibration is performed once at the start of the session.

B. Language-driven 2-D Target Localization

A user utterance S is fed to a fine-tuned **Grounding-DINO**. We add a vision–language alignment loss

$$\mathcal{L}_{\mathrm{VL}} = -\big\langle \phi_v(B),\, \phi_l(S) \big\rangle$$

to improve retrieval of industrial parts. The network returns a set of candidate bounding boxes $\mathcal{B} = \{b_k\}$; the top-confidence box proceeds to segmentation.

C. Instance Segmentation and 2-D–3-D Mapping

SAM, fine-tuned on the target domain, produces a binary mask $M(u, v)$ for the chosen box $\hat{b}$. For every pixel in the mask,

$$\mathbf{p}_c = D(u,v)\, K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad \mathbf{p}_w = {}^{w}\mathbf{T}_c \begin{bmatrix} \mathbf{p}_c \\ 1 \end{bmatrix}, \tag{2}$$

yielding the real-scene dense point cloud $P_r = \{\mathbf{p}_w\}$.

D. Key-point Matching and Coarse Registration

We deliberately cap the number of key points to **200–300** per cloud. After ISS extraction we randomly down-sample the sets $\mathcal{K}_r, \mathcal{K}_g$ to that range.

a) Descriptor construction.: For every key point we compute the 33-dimensional Fast Point Feature Histogram (FPFH) descriptor $f \in \mathbb{R}^{33}$.

Algorithm 1 FPFH-based correspondence construction

Require: Real-scene descriptors $\{f_i^r\}_{i=1}^{N_r}$; CAD descriptors $\{f_j^g\}_{j=1}^{N_g}$; distance threshold τ
Ensure: Correspondence set $\mathcal{C}$

```
Build a KD-tree on {f_j^g}_{j=1}^{N_g}
C ← ∅
for i ← 1 to N_r do
    j* ← arg min_j ‖f_i^r − f_j^g‖_2
    if ‖f_i^r − f_{j*}^g‖_2 < τ then
        C ← C ∪ {(i, j*)}
    end if
end for
```

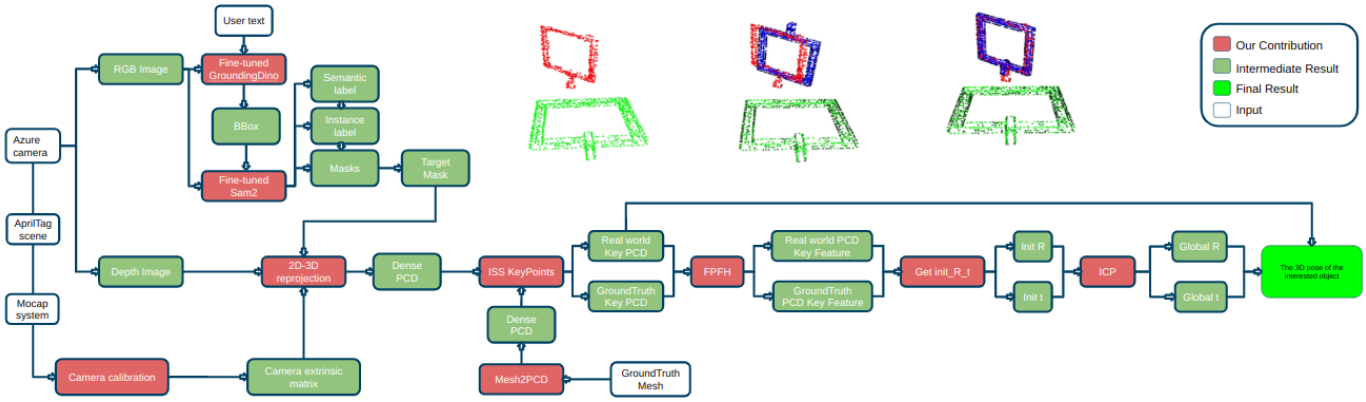

Fig. 2. End-to-end pipeline for language-guided real-time 6-DoF pose estimation. red blocks are our own contributions, green blocks are intermediate/final results, white blocks denote external inputs. The output is the object pose ${}^{w}\mathbf{T}_{o} = [\,\mathbf{R}\,|\,\mathbf{t}\,] \in \mathrm{SE}(3)$ in the world frame.

Algorithm 2 RANSAC–Kabsch pose estimation

Require: Correspondences $\mathcal{C}$, max iterations M, inlier threshold δ
Ensure: Pose $(R^\star, \mathbf{t}^\star)$

bestInliers $\leftarrow \varnothing$
for $m \leftarrow 1$ **to** M **do**
  Randomly sample 3 pairs from $\mathcal{C}$ and solve $(R, \mathbf{t})$ by Kabsch
  inliers $\leftarrow \varnothing$
  for all $(\mathbf{k}_i^r, \mathbf{k}_j^g) \in \mathcal{C}$ **do**
    if $\|R\mathbf{k}_i^r + \mathbf{t} - \mathbf{k}_j^g\|_2 < \delta$ **then**
      inliers $\leftarrow$ inliers $\cup\, (\mathbf{k}_i^r, \mathbf{k}_j^g)$
    end if
  end for
  if $|\text{inliers}| > |\text{bestInliers}|$ **then**
    bestInliers $\leftarrow$ inliers, $(R^\star, \mathbf{t}^\star) \leftarrow (R, \mathbf{t})$
  end if
end for

$i \leftarrow 10$
if $i \geq 5$ **then**
  $i \leftarrow i - 1$
else
  if $i \leq 3$ **then**
    $i \leftarrow i + 2$
  end if
end if

Finally, the inlier-maximising pose $(R^\star, \mathbf{t}^\star)$ is supplied to the next ICP stage (Sec. IV-E).

E. ICP Fine Registration

Starting from $(R_0, \mathbf{t}_0)$ we run point-to-plane Iterative Closest Point:

$$E(R, \mathbf{t}) = \sum_{\mathbf{p} \in P_r} \Big[\mathbf{n}_{\mathbf{q}}^{\top} \big(R\mathbf{p} + \mathbf{t} - \mathbf{q} \big) \Big]^2, \tag{3}$$

where $\mathbf{q}$ is the nearest neighbour of $\mathbf{p}$ and $\mathbf{n}_{\mathbf{q}}$ its normal. Gauss–Newton linearisation yields the optimum $(R^\star, \mathbf{t}^\star)$, i.e. the final object pose ${}^{w}\mathbf{T}_{o}$.

F. Runtime Analysis

Module	Parallelism	Notes
Grounding-DINO	GPU	batch 1
SAM	GPU	box prompt
2-D–3-D mapping	CPU	1028× 720 depth
Key-pts & FPFH	GPU	Open3D/CUDA
ICP	CPU	≤15 iters
Total	Mixed	≈3 Hz

TABLE I
END-TO-END RUNTIME BREAKDOWN.

In summary, the design combines vision–language models with classical 3-D registration while sustaining ˜5 Hz.

V. Experiments

To illustrate the performance of our proposed pipeline in the real world scenarios, we set up our own workspace and collect our own data to finetune the model.

A. Experimental Setup

Fig 3 shows the experimental set-up we have, which is taken by the Azure Kinect Camera that settled in front of the workspace. We attached an april tag at the end effector of the kinova arm and did hand-eye camera calibration by moving it to different positions. Three different target objects, square, circle, and triangle, are hung at different positions with at least three markers on each of them. We activated the OptiTrack system and collected the ground truth position of the target objects.

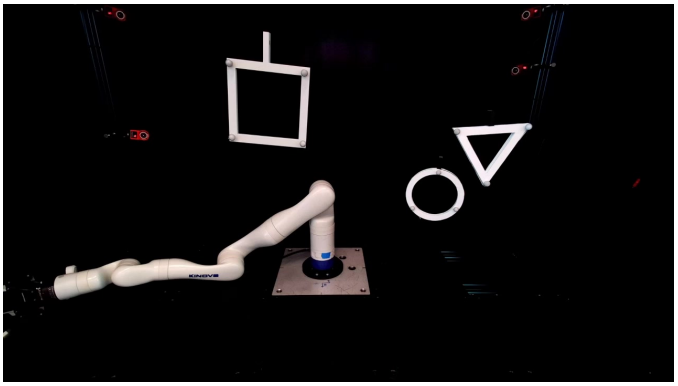

Fig. 3. example of the experiment set up we have

B. Data Preparation

We activated Azure Kinect camera and OptiTrack to obtain both the RGB-D sequence and the 3D trajectory of the three moving target objects. We did the following data preprocessing to better fit our goal of fine-tuning pretrained models and test its performance in real world scenarios: synchronization of RGB-D sequence and 3D trajectory and mask annotation.

1) Synchronization: Although we started using the Azure Kinect camera and the OptiTrack system roughly at the same time, we need more accurate synchronization to minimize the camera calibration error when lifting two pixel points to 3D. By selecting key features from the RGB-D sequence and OptiTrack results (e.g. when does the square object begin to move;), we synchronized the two sequences. Considering their difference in fps (Azure Kinect camera in 30 fps and OptiTrack system in 100 fps), we also define a bidirectional mapping between the camera time index and the OptiTrack time index by using two known matches and their respective frame rates.

2) Mask Annotation: Big models eat data. To ensure the quality of data we used to fine-tune SAMv2, we annotated the mask of the objects by utilizing both the rgb image and the depth information. Thanks to the color contrast of the target objects and the background, we first annotated each frame with a coarse mask. However, this coarse mask cannot distinguish between the three target objects and contains some part of the kinova arm. To solve this issue, we utilized the depth information as a filter. We intentionally placed three target objects at distinct positions, each moving within its own specified depth range. By thresholding the depth, we obtained the mask of each target object. Using the segmentation mask of each target object, we further padded its edges to obtain the bounding box.

C. Fine-tunning GroundingDINO

With prepared data, we followed this instruction to fine-tune GroundingDINO by formatting data correctly (ODVG Dataset Format) and feeding them into training. We freezed the backbone but only fine-tuned the detection head using the data we collected. There are 1800 images in total with well-annotated bounding boxes for each target objects. Our starting learning rate is 1×10^{-4} and ending learning rate is 1×10^{-6}. We use Adam Optimizer with weight decay 1×10^{-4}. To make training converge, we adpated a cosine annealing scheduler and the batch size we chose is 16. Considering we are using a small dataset, we applied crop; horizontal and vertical filps; $90°$, $180°$, and $270°$ rotation; and color-channel reversal to our dataset. Each augmentation is applied independently with random probability, and it is possible for more than one augmentation to be applied to the same image data. However, we did not see a significant improvement before or after the fine-tuning. We suspect that the pretrained network can already handle our dataset effectively because the target object has clear geometric shape and the background is clean.

D. Fine-tunning SAMv2

The large model of SAMv2 with 1024×1024 image resolution already generates accurate masks of each object. However, the inference speed using such a large model with high image resolution is slow, which takes about 1s to generate the mask. In order to reach our real-time goal, we use the tiny model of SAMv2 and decrease the image resolution to 512×512. We finetuned the mask decoder following this instruction using the collected rgb images with annotated masks and and AdamW optimizer with learning rate 1×10^{-5} and weight decay 4×10^{-5}.

VI. Results

A. Camera Calibration

By mounting the AprilTag on the end effector and moving the robot arm into different position, we calibrate the camera using OpenCV. We detected the corners of AprilTags and macthed them with 3D trajectory capture by OptiTrack. With a known camera intrinsics and distortion coefficients, we obtained the extrinsic matrix using the solvePnP() function of OpenCV. The projection error in pixel space is 0.94 pixels.

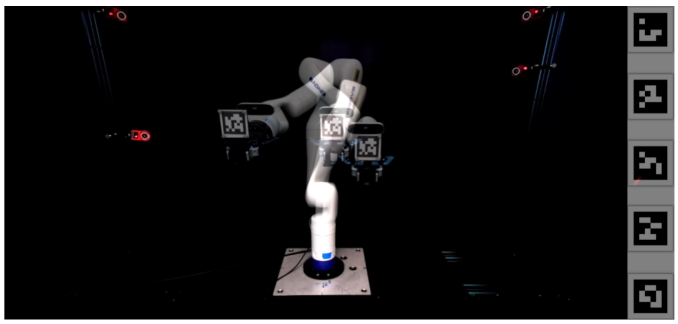

Fig. 4. illsutration of camera calibration process

B. Evaluation

We evaluated the performance of our proposed pipeline using the dataset we collected. We select two different metrics. The first is the degree difference in rotation between the ground truth normal vector and the predicted normal vector; the second is the RMSE between the ground truth center

and the predicted center. We noticed that after lifting things to the real world, the error is higher than we expected. We suspect that imperfect camera calibration and inaccurate depth information may increase the error. For qualitative evaluation example, please check our website.

	Square	**Circle**	**Triangle**
Rotation Difference (°)	9.58	12.46	11.34
RMSE (cm)	5.83	7.12	6.63

TABLE II
ROTATION ERROR AND RMSE FOR EACH TARGET OBJECT

VII. CONCLUSIONS

We have presented a real-time, language-guided 6D pose estimation pipeline that combines fine-tuned Grounding-DINO for object localization, SAM2 for instance segmentation, and efficient 3D point-cloud reconstruction (key-point matching, RANSAC, ICP) to achieve interactive (3 Hz) performance with rotation errors under 15° and RMSE under 7 cm. Although the error is not small enough to enable real-world robotic application, we believe a more careful engineering set-up can reduce the error significantly.

REFERENCES

[1] Y. Labbé, L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpentier, M. Aubry, D. Fox, and J. Sivic, "Megapose: 6d pose estimation of novel objects via render & compare," in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[2] B. Wen, J. Tremblay, V. Blukis, S. Tyree, T. Muller, A. Evans, D. Fox, J. Kautz, and S. Birchfield, "Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects," *CVPR*, 2023.

[3] J. Zhang, W. Huang, B. Peng, M. Wu, F. Hu, Z. Chen, B. Zhao, and H. Dong, "Omni6dpose: A benchmark and model for universal 6d object pose estimation and tracking," 2024. [Online]. Available: https://arxiv.org/abs/2406.04316