

Appendix

A Additional Experiment in Home Environment

We conduct comparative experiments in a home environment to evaluate our method in more diverse and challenging environments. We select the home, which is not included in all datasets. In the home, we choose ten objects, make five prompts for each object, and navigate the robot toward the target object, similar to our evaluation in the main paper. The appearance of the target objects and the given prompts are shown in the Appendix below.

The two baselines in the home perform better than the office environments. However, there is still an explicit advantage between our method and the baselines. Our method trained on augmented YouTube videos learns a general policy that can navigate towards novel objects in novel environments. Here, “novel” indicates that the training dataset does not include images of the object in question in the environment seen during evaluation.

More evaluations in novel environments are shown in our supplemental material video.

Table 3: **Quantitative results using a prototype real robot in home environment.** We show the goal success rate. A success is determined by the robot reaching within a 0.2 [m] radius of the target object.

Method	Total	Simple prompts	Noisy prompts	Multiple objects
CoW	0.72	0.80	0.67	0.53
Owl-ViT + ViNT	0.60	0.60	0.60	0.40
Our method	0.84	0.85	0.83	0.80

B Additional Data Ablation

In addition to the robot dataset ablation study in Fig. 6, we conduct an additional dataset ablation study for the YouTube Tour Dataset and our Human-walking Dataset. By including more data sources in our augmented dataset, the performance of the trained language-conditioned navigation policy improves. We show an improvement in the performance of the policy by adding the Indoor Navigation Dataset and our Human-Walking Dataset. We hypothesize that improvements from adding more data from YouTube saturate the least as due to the broad distribution of environments and objects within the dataset. We can use YouTube video data because of our data augmentation approach, which enables us to leverage the diverse in-the-wild video. Note that we add the test dataset to the YouTube Tour Dataset and Human-Walking Dataset due to make the balance of data between the three sources more even Fig. 7.

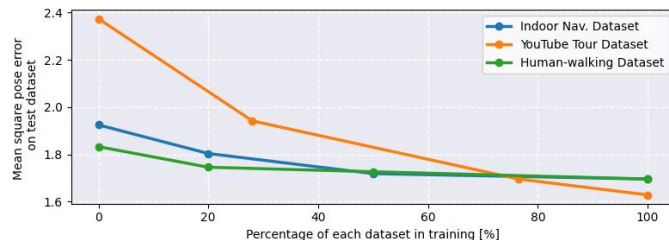


Figure 7: **Data Ablation.** An ablation of the percent of each dataset included in training data mixture, while keeping the entirety of the other datasets in the data mixture. The data ablation that studies the Indoor Navigation Dataset and Human-Walking Dataset use 76% of the YouTube dataset due to the addition of new YouTube data during the course of the project.

C Model Ablations

We also study how ablations of our model architecture impact the performance of our policy while training on the LeLaN dataset. For the visual encoder, we replace "ResNet-FiLM" in our method with "ViT-B32" and "ViT-ResNet50" of the pre-trained CLIP. For the text encoder, we employ larger pre-trained CLIP text encoder "ViT-ResNet50" instead of "ViT-B32".

In this ablation study, we evaluate each model on the test dataset. Similar to the objective in training, we calculate the mean square error between the generated virtual robot pose from the control policy and the target object pose. From Table 4, the pre-trained visual encoders from CLIP are worse than the ResNet-FiLM trained on our augmented dataset. The visual features from the CLIP visual encoder are insufficient to derive time-series velocity commands because they do not include geometric information. Furthermore, the ResNet-FiLM inserts the text features from the text encoder for low-level visual features, which helps to understand the target objects in the image view. In addition, the larger CLIP text encoder helps with learning a precise control policy. However, the advantage of a larger encoder is not significant on the test dataset. Furthermore, when navigating with a real robot, its difference was trivial and, in fact, increased the computational load on the robot controller. This not only reduced the frame rate, but also increased battery consumption. Therefore, we used the the pre-trained text encoder from the "ViT-B32" CLIP model for the main model in the paper.

Table 4: **Ablation study of our model architecture.** We use the pre-trained weights of ViT-B32 and ViT-ResNet50 from CLIP for both the visual and text encoders. When using ResNet-FiLM, we train our model from scratch.

Visual encoder			Text encoder		MSE
ResNet-FiLM	ViT-B32	ViT-ResNet50	ViT-B32	ViT-ResNet50	
✓			✓		1.291
✓				✓	1.202
	✓		✓		1.690
		✓		✓	1.673

D Target Objects and Prompts in Evaluation

In our evaluation, we select 18 objects in the university campus environment (inside and outside) and 10 additional objects in the home environment and prompt the robot to navigate towards the target objects. For each object, we feed 5 or 6 trials with different prompts (some of which are noisy) and evaluate the robustness of the policy. Here we show the overview of the target objects and the prompts in our evaluation. First 18 objects are from the university campus. The rest are from a home environment.

First, two prompts for each object are for the simple prompts and the others are for noisy prompts, which includes wrong adjectives (red) long prompts, or the prompts without the target object's noun. The red border in the image indicates the presence of multiple corresponding objects in the experimental environment. If the objects can be distinguished by prompts, success is considered only if the robot reaches the correct object; if the objects cannot be distinguished by prompts, success is considered if the robot reaches one of the objects that fits the description of the prompt.



- couch
- small rounded purple couch
- purple **carpet**
- round plush piece of furniture
- purple couch on the grey concrete floor, next to the **blue grey pillow**
- pinkish purple cushion-like stool



- fridges
- shiny fridge with a water dispenser
- **arched** fridge with a water server
- tall shiny appliance with two doors
- Fridge next to the wooden shelf
- dark silver something to storage food

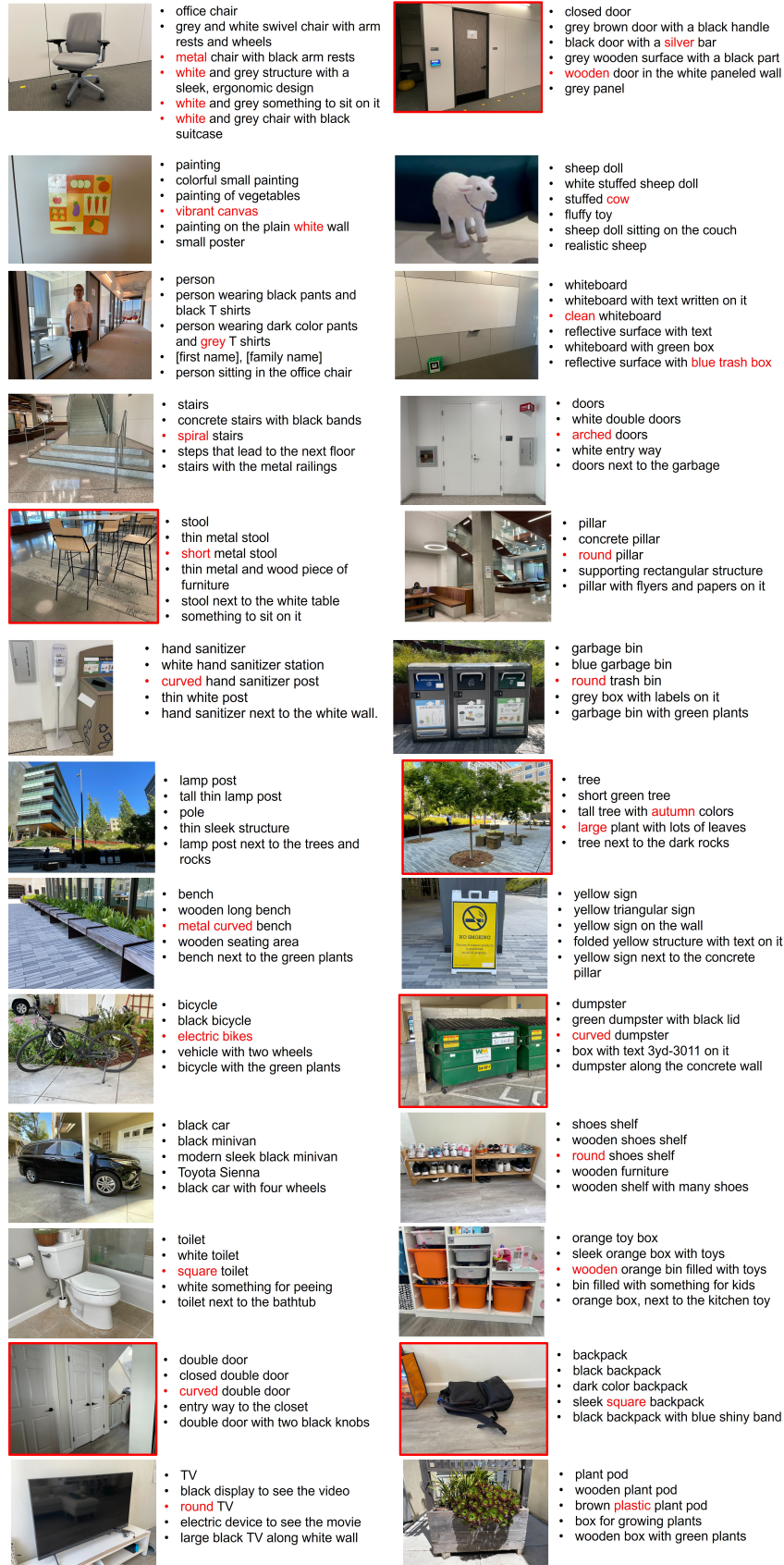


Figure 8: Overview of 28 target objects and various prompts in our evaluation.

E Baseline Method

In our evaluation, we conduct a comparative evaluation with two strong baselines trained on the internet scale datasets. Here we explain the details of each of their implementations.

CLIP on Wheels (CoW) We implement the best-performing CoW baseline with the OWL-ViT B/32 detector [51]. Similar to our method, we feed the current observation and the prompts corresponding to the target object into the OWL-ViT B/32 detector [51], which was trained an internet scale dataset to estimate the boundary box for the target object. We crop the estimated point clouds by the estimated boundary box and take the median value as the target object pose. To have a fair comparison without method using a single camera image only (no depth camera and LiDAR), we estimate the depth with Depth360 [45] and project it to estimate the point clouds. To control the robot toward the detected object, we use a state lattice motion planner to generate the linear and angular velocity commands. The details of the implemtd state lattice motion planning are shown in this appendix below. We limit the scope of instructions to object navigation for objects within view from the starting point of the robot trajectory. Therefore, we do not implement the exploration portion of CoW.

OWL-ViT + ViNT To compare our method with a learning-based method, we leverage the foundation model for the vision-based navigation, which can navigate the robot towards a goal position conditioned on a goal image view. To take a goal image view corresponding a target object, this baseline leverages Owl-ViT, a VLM trained on internet-scale data and combine it with ViNT, a control policy trained on multiple dataset collected by various mobile robots. Specifically, we feed the cropped image from the Owl-ViT into the ViNT as a goal image.

F Implementation details

We show the details of our training and the evaluation setup using a real prototype robot in language-conditioned navigation.

F.1 Training

To train our control policy, we randomly choose 256 observations from our whole dataset. Since one observation contains multiple objects and each object contains multiple prompts, in almost all cases, we randomly select the object and prompt (which is based on the object).

By feeding the observations and the prompts into the model, we calculate our model and generate a sequence of the velocity commands for $N(=24)$ steps. Then, we estimate the virtual robot pose N steps in the future via our kinematic model (integration of the velocity commands in our case). Finally, we calculate the objective J in Eqn. 1 and update our policy π_θ . Our training is with an Adam optimizer using a learning rate 0.0001 on a workstation with a Intel i9 CPU, 96GB RAM and an NVIDIA RTX 4090 GPU.

F.2 Robot experiment

Figure 9 shows the overview of a prototype mobile robot in navigation. We calculate the control policy on the edge robot controller, Nvidia Orin AGX, with the best frame rate for each method. We mount the omnidirectional camera, a RICOH Theta S on the robot and only use the front-side fisheye camera as the observation. Since we learn the visual encoder from scratch, there are no restrictions on the camera on the robot, but we use cameras with a wide FOV to reduce blind spots and make object detection easier.

In the evaluation, we provide the language instruction to the policy once at the beginning of navigation to reduce the computational load in each step. To control the real robot, we repeatedly calculate our control policy at the best frame rate on the robot edge controller and feed the first step veloc-

ity command $\{v_0, \omega_0\}$ in the generated sequence of the velocity commands $\{v_i, \omega_i\}_{i=0 \dots N}$, similar to the receding horizon control. We test our method against two baselines, CLIP on Wheels and OWL-ViT + ViNT.

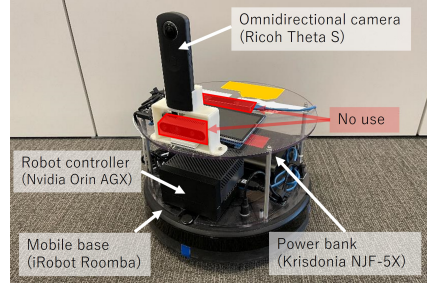


Figure 9: **Overview of the prototype mobile robot.** Note that we only use the front side camera of the omnidirectional camera, Ricoh Theta S, to navigate the robot.

G YouTube Video List

We list all URLs of the YouTube video in our YouTube Tour Dataset below.

- https://www.youtube.com/watch?v=vQU_QydOUIw
- <https://www.youtube.com/watch?v=5J2Wsvnk-Ec>
- <https://www.youtube.com/watch?v=b9thcSOI8bw>
- <https://www.youtube.com/watch?v=V511PNMx2uw>
- https://www.youtube.com/watch?v=DO-JDTu_h5I
- <https://www.youtube.com/watch?v=oQ61ijChego>
- <https://www.youtube.com/watch?v=EwJQG74b174>
- <https://www.youtube.com/watch?v=rM01DH0bv1o>
- <https://www.youtube.com/watch?v=9MWWZeCr3QE>
- <https://www.youtube.com/watch?v=-3vt2My1vsw>
- <https://www.youtube.com/watch?v=HknDp84cFBM>
- https://www.youtube.com/watch?v=l_s9YAluXBY
- <https://www.youtube.com/watch?v=k3Q1vse7In8>
- <https://www.youtube.com/watch?v=ISNDJ2Pjq34>
- https://www.youtube.com/watch?v=4jDa_5S-0W4
- https://www.youtube.com/watch?v=2O7JrGu_mVk
- <https://www.youtube.com/watch?v=je8267s9z38>
- <https://www.youtube.com/watch?v=tWovplr-ois>
- <https://www.youtube.com/watch?v=UmCbkpRUOA4>
- <https://www.youtube.com/watch?v=Ea2yExKlg7w>
- <https://www.youtube.com/watch?v=1Zu6Xct5bLQ>
- <https://www.youtube.com/watch?v=9IluzedLtYs>
- https://www.youtube.com/watch?v=lnYfw_ryOdQ
- <https://www.youtube.com/watch?v=9r5eK5JXzLo>
- <https://www.youtube.com/watch?v=LdWHy-f3jYg>
- <https://www.youtube.com/watch?v=Kcc7zuQDlpE>
- <https://www.youtube.com/watch?v=r-98ADAXxQM>
- <https://www.youtube.com/watch?v=iRfQa2SEu0Q>
- <https://www.youtube.com/watch?v=NzFbFARYhfE>
- <https://www.youtube.com/watch?v=i3QkZ0xW92Y>
- <https://www.youtube.com/watch?v=stUYODYcPCI>
- <https://www.youtube.com/watch?v=GCKyfi5LRYM>
- <https://www.youtube.com/watch?v=848EpwPmQfA>
- <https://www.youtube.com/watch?v=Bq4rmeIvJbs>

550 • <https://www.youtube.com/watch?v=uVy9TKMA-f8>
 551 • https://www.youtube.com/watch?v=_OZhGsKdBY

552 H State lattice motion planning

553 We implemented sampling-based motion planning as the local motion planner in **CLIP on Wheels**
 554 (**CoW**) baseline. We generated 15 motion primitives assuming steady linear and angular velocity
 555 commands for 8 steps (2.664 s). The pairs of linear and angular velocity commands are $(v_s, \omega_s) =$
 556 $(0.0, 0.0), (0.2, 0.0), (0.2, 0.3), (0.2, 0.6), (0.2, 0.9), (0.2, -0.3), (0.2, -0.6), (0.2, -0.9), (0.5, 0.0),$
 557 $(0.5, 0.3), (0.5, 0.6), (0.5, 0.9), (0.5, -0.3), (0.5, -0.6), (0.5, -0.9)$. We selected these 15 motion
 558 primitives by balancing computational load and navigation performance.

559 By integrating these velocity commands for 8 steps, we obtained 15 trajectories such as
 560 $\{\{^s p_i^j\}_{i=1\dots 8}\}_{j=1\dots 15}$, where $^s p_i^j$ is the i -th virtual robot pose on the j -th motion primitive. To
 561 select the best motion primitive, we calculated the following cost value for each primitive.

$$J_s^j = \min_i (\hat{p}_{\text{obj}} - ^s p_i^j)^2 \quad (2)$$

562 Here, \hat{p}_{obj} indicates the estimated target object pose in **CLIP on Wheels (CoW)** baseline. This
 563 objective calculates the squared errors between all 8 poses in the j -th motion primitive and the goal
 564 pose and selects the minimum one to evaluate the goal-reaching performance. Then, we choose the
 565 motion primitive with the minimum $\{J_s^j\}_{j=1\dots 15}$ and assign the corresponding velocity commands
 566 v_s and ω_s to control the robot during navigation.