

[Supplementary Material]

ObjectReact: Learning Object-Relative Control for Visual Navigation

Anonymous Author(s)

Affiliation

Address

email

1 In this document, we provide additional details related to our method and experimental setup.
2 We also include additional results and analyses, especially in reference to the attached video re-
3 sults from both real world and simulator. We will make the source code, task datasets, and
4 video demonstrations publicly available through the (currently anonymized) project page: <https://object-react.github.io/>
5

6 A1 Method

7 **WayObject Costmap Representation:** We elaborate on our proposed learnable representation of
8 the segmentation masks and their corresponding path lengths. Given a variable number of binary
9 object masks as a tensor $\mathbf{M}_{HW \times N_m}$ in the current image and their path lengths $\mathbf{p} \in \mathbb{R}^{N_m}$ to the long-
10 horizon goal, we obtain the path length encoding and the WayObject Costmap \mathcal{W} representation as
11 below:

$$\mathcal{W} = \mathbf{M}_{HW \times N_m} @ \mathbf{E}_{N_m \times D} \quad (1)$$

$$E(p)_i = \begin{cases} \sin\left(\frac{p}{Z^{i/D}}\right) & \text{if } i \text{ is even} \\ \cos\left(\frac{p}{Z^{(i-1)/D}}\right) & \text{if } i \text{ is odd} \end{cases} \quad \forall i \in [0, D] \quad (2)$$

12
13 where $\mathbf{M} \in \{0, 1\}^{HW \times N_m}$ represents ℓ_1 -normalized N_m segmentation mask arrays with height H and
14 width W corresponding to the currently-viewed objects. \mathbf{E} represents a D -dimensional positional
15 encoding of the path lengths for each of the objects, with $D = 8$ and $Z = 10000$. The path lengths are
16 normalized per image and re-scaled to $[1, 100]$ before encoding such that 100 represents the shortest
17 path length. We use $p = 0$ to represent an outlier corresponding to undetected and unmatched image
18 segments.

19 **Model Architecture and Loss functions:** We base the training of our navigation controllers on
20 GNM [1]. The GNM architecture consists of two convolutional encoders: one for the current ob-
21 servation image along with a history of 5 previous images, and the other for the goal image (fused
22 channel-wise with the other inputs). The output of these two encoders is concatenated and decoded
23 using an MLP and two linear projection heads to predict: a) *local waypoints*, that is, position and
24 yaw relative to the current robot position, and b) *distance* to the image goal. Our object-relative
25 controller uses a modified version of this architecture with a *single* goal encoder for the WayOb-
26 ject Costmap, which is based on a custom ResNet model with D input channels. Our WayObject
27 Costmap has the same resolution as the images used by GNM ($W = 85$, $H = 64$), but each pixel
28 consists of a positional encoding E with $D = 8$ to represent the locally normalized path lengths (see
29 Equation 2). We predict a trajectory rollout of 10 future 2D waypoints in the BEV space of the
30 robot’s frame of reference. As our object-relative encoders do not require an explicit goal image, we
31 ignore the distance prediction output.

32 A2 Experimental Setup

33 In this section, we provide implementation details for our proposed pipeline and the baselines. We
34 also discuss the episodes which were excluded during the evaluation, which highlights the limita-
35 tions and implementation overheads of the underlying simulator.

36 A2.1 Implementation Details

37 **Mapping:** We used the automatic mask generator of SAM2 [2] (ViT-L) to obtain segmentation
38 masks, sampling 16 points per side and using 1 additional cropping layer with a downsample factor
39 of 2. We used CLIP [3] to remove segments if their masked image embedding matched with text
40 string ‘floor’ or ‘ceiling’. We used Superpoint-Lightglue [4] to match every current frame with 3
41 previous frames to obtain keypoint correspondences. To enable batching, we used a fixed number of
42 SuperPoint keypoints (2048) and disabled early stopping and point pruning in LightGlue. To obtain
43 a local 3D projection of a segment, we used the indoor model checkpoint from Depth Anything [5].

44 **Localization and Planning:** We used FastSAM [6] from Ultralytics for segmentation during the
45 execution phase, using a mask confidence threshold of 0.5. For localization, we used a submap cen-
46 tered at the reference map image that is closest to the agent’s current state in the simulator and used
47 a radius of 16 frames with a subsampling factor of 2 to perform pairwise image matching between
48 the query and the submap images. For real-world experiments, the submap center is calculated as
49 the reference map image that has the most segments matched with the current query image and its
50 history of 8 frames. For efficient global path planning, we precompute all path lengths from all
51 the map nodes to the goal node, where the goal node is estimated based on the maximum IoU of
52 the predicted segments with the ground truth segmentation mask of the goal object – if $\text{IoU}=0$, the
53 episode is deemed a failure, though it occurs rarely. For each matched segment pair between the
54 query and a reference map image, we look-up the path length from the precomputed data. As some
55 of the query segments get matched with multiple map nodes, we only consider the match that has the
56 least path length. For the query segments that could not be localized, we assign them a path length
57 through a tracking mechanism: we perform pairwise image matching between the current image and
58 its history of 8 frames, and use these segment correspondences to track path lengths and assign a
59 median path length to the current query segment that was not localized but could be tracked. Finally,
60 any undetected, or unlocalized and untracked, image segments are marked as outliers and encoded
61 as described in Section A1.

62 **Control Execution:** We broadly follow GNM [1] in terms of execution of the predicted waypoints.
63 We used the last waypoint from the predicted trajectory rollout to calculate linear and angular veloc-
64 ities. We clip the linear velocity between 0 and 0.05 m/s , and the angular velocity between -0.1 and
65 0.1 rad/s . We compute a moving window average of these velocity values over predictions from the
66 past 5 image observations. For the real-world experiments, we clip the linear velocity between 0.15
67 to 0.4 m/s ; we execute the predicted velocities on the Unitree Go1 robot dog using its HighCmd
68 control interface.

69 **Image resolution, field-of-view and agent radius:** For our simulator experiments, we consistently
70 used 120° field-of-view images. For the mapping and execution phases, we used an agent radius
71 of 0.75m and an image resolution of 320×240 . For controller training, we used an agent radius of
72 0.3m , and an image resolution of 85×65 – the same as that used in GNM [1].

73 A2.2 Baselines

74 **PixNav [7]:** PixNav is a transformer-based imitation learning method for local navigation [7], which
75 utilizes a patch of goal pixels representing either the final destination or intermediate navigation
76 targets. This goal patch is initially input to the model as a mask alongside the corresponding RGB
77 image, and the model then selects an action from a discrete set: {Stop, MoveAhead, TurnLeft,
78 TurnRight, LookUp, LookDown}. At each following step, the model uses the current RGB image, a
79 collision signal, the sequence of previous images, and the initial goal mask to predict the next action.

80 Unlike RoboHop and our proposed method ObjectReact, which are continuous controllers with fixed
 81 cameras, PixNav operates as a discrete controller with a movable camera. To accommodate these
 82 differences and align with PixNav’s design, its evaluations were configured to begin with a visible
 83 intermediate goal determined by our global path planner. The goal is then updated either when the
 84 model outputs the ‘Done’ command or when the memory buffer reached capacity. We used the
 85 official model checkpoint provided by the original authors.

86 **RoboHop [8]:** We obtained the source code from the original authors. We use its zero-shot con-
 87 troller based on path length-weighted 2D pixel centers of the object segments:

$$\Delta\phi = \frac{g}{w} \sum_j \alpha_j (m_j - o); \quad \alpha_j = \frac{e^{\beta d_j}}{\sum_j e^{\beta d_j}} \quad (3)$$

88 where m_j denotes the centroid of segment S_j , and o represents the image center. Each d_j is the
 89 min-max normalized path length associated with segment S_j . The weights α_j are computed using a
 90 softmax function with temperature parameter $\beta = 5$. The term w refers to the image width, and g is
 91 a fixed gain constant set to 0.4.

92 A2.3 Evaluation

93 **Excluded episodes:** We procedurally generate object goals for the challenging tasks of Alt-Goal,
 94 Reverse, and Shortcut. Through manual inspection, we then exclude episodes for a task in cases
 95 where the goal is invalid, for example, when the goal is unreachable due to incorrect rendering of
 96 the surrounding obstacles or when it covers a large part of the image (e.g., when only a wall is
 97 visible). We also exclude episodes where the goal does not meet the criteria of the task, for example,
 98 when all possible alternative goals for an episode fall on the path to the original goal. Finally, we
 99 exclude scenes with rendering issues such as *i)* large areas of missing geometry and *ii)* areas that are
 100 visually traversable but not reachable in the simulator. This leads to a removal of 6, 16, 13, and 9
 101 episodes, respectively, for the Imitate, Alt-Goal, Shortcut, and Reverse tasks.

102 A3 Additional Results

103 A3.1 Quantitative Analyses

104 **Comparing different GNM models:** In the main paper, we used the GNM model representing an
 105 image-relative approach, which we trained on the same simulator data as that used for our proposed
 106 object-relative method ObjectReact. Here, we include results for the GNM model that was trained
 107 on a variety of real-world outdoor scenes and embodiments, using the official checkpoint provided
 108 by the original authors. As observed in Table A1, both the GNM models – sim and real world –
 109 perform similarly; this shows that the performance gap on the challenging tasks is not attributed to
 110 the training environment or its diversity in terms of embodiment, but instead on the type of world
 111 representation (image- versus object-level) and the subgoal conditioning type (image pairs versus
 112 WayObject costmap).

113 A3.2 Qualitative Results and Analyses

114 We provide details of our real-world and simulated experiments, with reference to the accompanying
 115 supplementary videos. We tested navigation capabilities on the Unitree Go1 robot dog [9] on a
 116 variety of tasks and environmental conditions. In the following, we discuss many successful trials
 117 that show that object-relative navigation is a promising avenue for further research. We also discuss
 118 failure modes to provide a better understanding of the key limitations, which we plan to address in
 119 future work.

Table A1: Comparing against different (image-relative) GNM models trained on real world and simulator data.

| Method | Train Env | Imitate | | Alt Goal | | Shortcut | | Reverse | |
|--------------------|-----------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| | | SPL | SSPL | SPL | SSPL | SPL | SSPL | SPL | SSPL |
| GNM | Sim | 57.58 | 66.15 | 2.17 | 13.54 | 7.69 | 18.40 | 11.60 | 23.59 |
| GNM | Real | 48.48 | 56.18 | 6.52 | 17.08 | 9.62 | 24.05 | 3.33 | 8.54 |
| ObjectReact (Ours) | Sim | 59.08 | 64.62 | 21.74 | 27.40 | 23.08 | 39.56 | 26.67 | 36.69 |

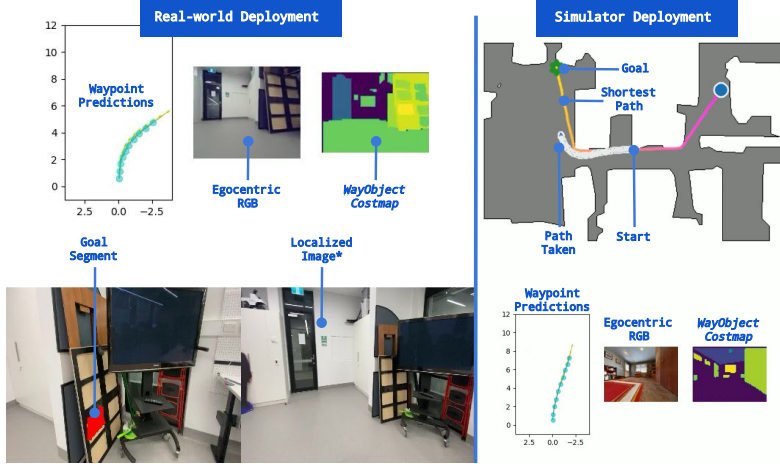


Figure A1: **Examples of demonstration videos.** Real-world demonstration video example (left) and simulator video example (right). * The localized image is the closest match found in the map (see Section A2.1 for details).

120 A3.2.1 Simulator Deployment

121 We have included several example videos of successful runs for our ObjectReact agent deployed in
 122 the simulator. These runs do not use any ground truth object instance information: FastSAM is used
 123 for segmentation and SuperPoint-LightGlue for matching. We have additionally included a deploy-
 124 ment run that uses ground-truth instances, *ground_truth_perception_imitate_sim.mp4*, to highlight the
 125 gap in perception quality that ObjectReact must overcome during deployment using the inference-
 126 based perception pipeline. We also include several failed episodes that demonstrate failure to avoid
 127 objects (e.g. *imitate_failure_sim.mp4*), and perception failures when facing away from the goal or in
 128 a position that diverges from the mapping run (e.g. *reverse_failure_sim.mp4*). Another failure mode
 129 is inability to match the goal segment in the map. This is more common for Alt Goal and Reverse
 130 tasks, where the goal may be on the periphery of the mapping perspective.

131 A3.2.2 Real-World Deployment

132 Here we provide a qualitative overview of the deployment of ObjectReact on the Unitree Go1
 133 quadruped robot under various conditions. For each trial, we include the video we used to gen-
 134 erate the map as well as the robot’s egocentric observations during autonomous deployment. See
 135 Figure A1 for details on the components of these videos.

136 **Cross-embodiment generalisation:** In these trials, we examined cross-embodiment generalization
 137 between mapping and execution in the real world. We used maps generated from videos taken
 138 using a *phone camera*, and deployed ObjectReact using these maps on the quadruped robot. We
 139 have included videos of a number of successful cross-embodiment demonstrations (those labeled
 140 *CrossEmbodiment* and several others). This generalization capability could be useful in future for
 141 taking advantage of multiple different sources of mapping data when a robot is navigating in a new
 142 or changing environment.

143 **Alternative tasks:** Here we examine the Alt Goal and Shortcut tasks. In *AltGoal_humanoidRobot*,
144 we demonstrate successful navigation towards an object that was only visible in the periphery of the
145 mapping run. In *Shortcut_cutout*, we show that ObjectReact is able to reach the goal along a direct
146 path, even when the mapping run followed a much longer, winding path to the goal.

147 **Different environmental conditions:** We also investigated whether ObjectReact is robust to envi-
148 ronmental changes between mapping and execution – an expected advantage attributed to the use of
149 open-set, zero-shot perception models. For *LowLight* trials, we generated the map with full lighting
150 but deployed the robot under low lighting conditions. For *DayNight*, we generated the map under
151 natural daylight conditions, but deployed during the evening. In both cases, ObjectReact was able
152 to successfully generalize due to its WayObject Costmap representation that is largely invariant to
153 these changes. In trials marked *Obstacles*, we also show that ObjectReact can also adapt to obstacles
154 that were not present during the mapping run.

155 A3.2.3 Limitations

156 In our video demonstrations, we present a number of trials that show that our ObjectReact controller
157 is able to generalize to real-world settings. However, there are several limitations of our approach
158 that make real-world deployment challenging. Here, we outline these, enumerate common failure
159 modes, and discuss our plans for overcoming these limitations in future.

160 **Training-deployment gap:** While our WayObject Costmap observations mitigate many problems
161 of *visual* generalization to the real world, there are several other challenges our approach must over-
162 come when deployed on a real robot. First, camera parameters differ between the simulator training
163 and real-world deployment. Second, the use of FastSAM + LightGlue in deployment for percep-
164 tion leads to different kinds of perception failures in comparison to those observed during training
165 (which uses ground-truth simulator perception with data augmentation based on perturbation of path
166 lengths). Finally, as shortest path demonstrations are used during training, cloning this behaviour
167 results in an agent that prefers to closely corner around objects, leaving little room for error; also,
168 the demonstrations go directly to the goal, lacking examples of taking slightly longer routes around
169 obstacles.

170 **Failure modes:** The above issues result in several common failure modes during real-world opera-
171 tion. These included **a)** collisions, **b)** perception failures, and **c)** deviating from the mapping area.
172 Because the ObjectReact policy is trained to take the shortest path to the goal, it often leaves little
173 margin for error when navigating past obstacles, sometimes resulting in collisions during execution
174 (see *failure_close_to_obstacle.mp4*). Perception failures can occur when the perception pipeline fails
175 to match important segments (like the goal or an obstacle), or matches a segment that connects to
176 many other objects, resulting in WayObject Costmaps that have large, low-cost regions that are dif-
177 ficult to differentiate. During deployment, the agent can also deviate from the mapping area due to
178 perception failures etc., but recovery from this state is not explicitly demonstrated in the training
179 data (see *failure_diverged_from_map_matched_wall.mp4*).

180 **Future work:** We hope to address the above limitations in future by **a)** incorporating real-world
181 training data, **b)** using a demonstration path-finding algorithm that leaves a larger margin for obsta-
182 cles where possible, **c)** training with the inferred perception pipeline to minimize sim-to-real gap,
183 and **d)** further improving the perception pipeline.

184 References

- 185 [1] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. Gnm: A general navigation model to
186 drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*,
187 pages 7226–7233. IEEE, 2023.
- 188 [2] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland,
189 L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár,

- 190 and C. Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint*
 191 *arXiv:2408.00714*, 2024. URL <https://arxiv.org/abs/2408.00714>.
- 192 [3] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell,
 193 P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervi-
 194 sion. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- 195 [4] P. Lindenberger, P-E. Sarlin, and M. Pollefeys. LightGlue: Local Feature Matching at Light
 196 Speed. In *ICCV*, 2023.
- 197 [5] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao. Depth anything: Unleashing the
 198 power of large-scale unlabeled data. In *CVPR*, 2024.
- 199 [6] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang. Fast segment anything,
 200 2023.
- 201 [7] W. Cai, S. Huang, G. Cheng, Y. Long, P. Gao, C. Sun, and H. Dong. Bridging zero-shot ob-
 202 ject navigation and foundation models through pixel-guided navigation skill. In *2024 IEEE*
 203 *International Conference on Robotics and Automation (ICRA)*, pages 5228–5234. IEEE, 2024.
- 204 [8] S. Garg, K. Rana, M. Hosseinzadeh, L. Mares, N. Suenderhauf, F. Dayoub, and I. Reid. Robo-
 205 hop: Segment-based topological map representation for open-world visual navigation. In *2024*
 206 *International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.
- 207 [9] Unitree go1 product page. <https://www.unitree.com/go1>. Accessed: 2025-05-01.