Supplementary Material

A. Verifying Threshold Optimality

We evaluate the effectiveness of our threshold-determining method for instance re-identification (Sec. 4.1). We extract a subset of 400 episodes belonging to the Train split and ensure scene and object diversity by sampling from 20 different scenes, 40 object instances per scene, and 1 episode per object. We then perform a search for the best detection threshold by running our complete `Mod-IIN` system with varying detection thresholds. In Fig. 7, we see the results of this search with success rate plotted against threshold value. The threshold determined via our maximal F-measure method coincides directly with the optimal threshold in downstream performance.

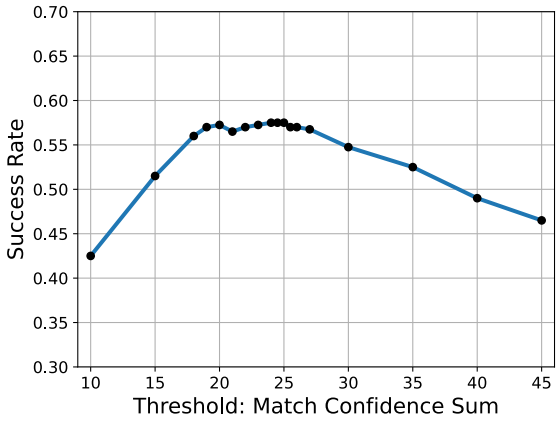

Figure 7: Empirical performance of instance Re-ID thresholds on a subset of the Train split.

B. Re-ID Thresholds of Alternate Methods

In Fig. 9, we show precision-recall curves and F-measure curves for each goal instance re-identification method we experiment with in the main paper: *ResNet*, *CLIP*, *Keypoint-Match*, and *Keypoint-Conf*. Both keypoint-based methods demonstrate significantly higher PR AUC and maximal F-measure scores than *ResNet* and *CLIP*. While maximal F-measure is very close between *Keypoint-Match* and *Keypoint-Conf* (0.962 *vs*. 0.972), this difference leads to a downstream InstanceImageNav performance difference of 0.022 `SR` (Tab. 2). Notably, the rank order of maximal F-measure aligns with the rank order of downstream performance.

C. *Crop-Projected* Goal Localization Details

Our primary goal localization method, *Detic-Projected*, involves producing an instance segmentation mask of the goal instance to determine which feature correspondences to project to world coordinates. We ablate this instance mask using the *Crop-Projected* method — instead of an instance-specific mask, *Crop-Projected* takes a static central crop of the goal image that is the same for all episodes. Matched keypoints lying inside this mask are projected to the goal map channel. We find that taking a crop of the central 1/3 of the goal image extending down to a height of 1/8 to be effective. We visualize two examples of this mask in Fig. 8.

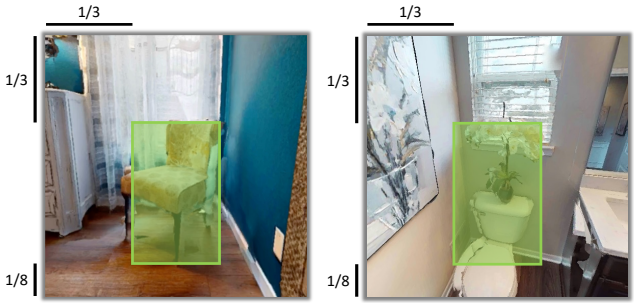

Figure 8: Examples of the static central crop applied to goal images in the Val split. While the crop reasonably frames the chair (left), the plant (right) is much smaller than the crop.

D. Implementation Details and Resources

We implement our models and vectorized mapping in PyTorch and base our implementation off Home-Robot[3] and SemExp ObjectNav[4] repositories. Model variations using ResNet employ a ResNet50 pre-trained on ImageNet-1K. Model variations using CLIP use the *ViT-B/32* weights. We use the pretrained SuperGlue model[5] with indoor-trained weights from ScanNet. For the Detic model, we use the official repository[6] with the default 21K-trained model. We apply a custom vocabulary of { *chair, sofa, bed, toilet, potted_plant, tv_monitor* }. For our real-world experiments, we deploy our system on a Stretch V1 by Hello Robot equipped with the default Intel RealSense D435i RGBD camera.

We compute the empirical run-time of our complete `Mod-IIN` method as evaluated in simulation against the Val split. Using 1 RTX 6000 GPU and 5 CPU cores, evaluation took 20 hours and averaged 6.3 FPS for combined environment simulation and model inference. For hardware experiments ran on Hello Robot Stretch, we offload model inference using ROS networking.

E. Real-World Qualitative Example

Fig. 1 shows a qualitative example of our agent's performance in the real world. In this example, the agent is operating in Env A (a furnished office space) and is given a picture of a chair. The agent traverses a hallway past a couch

[3] github.com/facebookresearch/home-robot
[4] github.com/devendrachaplot/Object-Goal-Navigation
[5] github.com/magicleap/SuperGluePretrainedNetwork
[6] github.com/facebookresearch/Detic

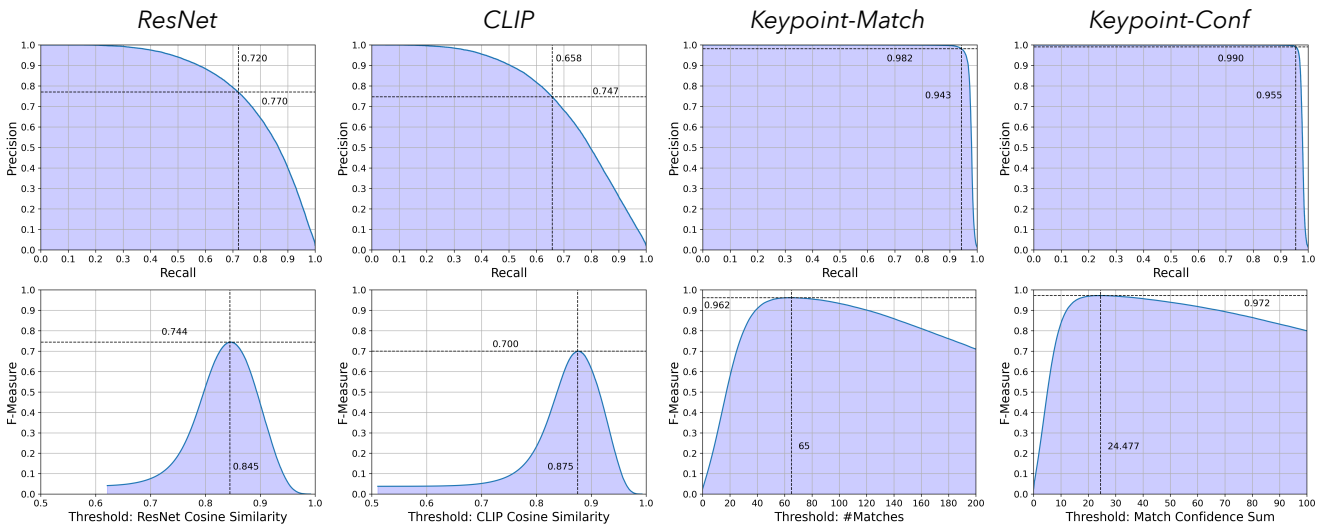

Figure 9: Precision-recall curves (top) and F-measure curves (bottom) for each goal instance Re-ID method.

(Steps 1, 97), enters the room with the chair (Step 164), re-identifies and localizes the chair (Step 229), and successfully navigates to it (Step 237). This example shows our method's robustness to the domain gap between sim and real – RGBD observations on Stretch contain real-world noise and poses are estimated using Hector SLAM on ROS[7]. The image goal is taken from a mobile phone camera and from a pose that affords an intuitive view of the goal instance according to the user that issues the task. With no reliance on robot hardware, this demonstrates the ease of goal specification in InstanceImageNav and the robustness of our method.

F. Complete Real-World Results

Tab. 3 shows the description and results for each episode performed by `Mod-INN` in the real world. The episode IDs in Tab. 3 correspond to the IDs of videos on the project page.

Episode ID	Environment	Goal ID	**Success?**
1	A	Chair_0	True
2	A	Plant_0	True
3	A	Couch_0	False
4	A	Couch_0	True
5	A	Couch_0	True
6	A	Couch_0	True
7	B	Plant_1	True
8	B	Bed_0	True

Table 3: Description of episodes performed by `Mod-INN` in the real-world with results.

[7] wiki.ros.org/hector_slam