

# Does learning when to look pay for home service robots? A preregistered test of adaptive perception scheduling in closed-loop navigation

MD ABU SAYEB RAYHAN

[sayeb.cse@gmail.com](mailto:sayeb.cse@gmail.com)

Comilla University

Hustagna Hiya

Jahangirnagar University

Ajoy Shutra Dhar

Comilla University

---

## Research Article

**Keywords:** Service robots, Mobile robot navigation, Adaptive perception, Embedded computing, Object detection, Closed-loop evaluation, Preregistration

**Posted Date:** October 13th, 2026

**DOI:** <https://doi.org/10.21203/rs.3.rs-11335280/v1>

**License:**   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

**Additional Declarations:** No competing interests reported.

---

Does learning when to look pay for home service  
robots? A preregistered test of adaptive perception  
scheduling in closed-loop navigation

Md Abu Sayeb Rayhan<sup>1\*</sup>, Hustagna Hiya<sup>2</sup> and Ajoy Shutra Dhar<sup>1</sup>

<sup>1\*</sup>Department of Computer Science and Engineering, Comilla  
University, Cumilla, Bangladesh.

<sup>2</sup>Institute of Information Technology, Jahangirnagar University, Savar,  
Dhaka, Bangladesh.

\*Corresponding author(s). E-mail(s): [sayeb.cse@gmail.com](mailto:sayeb.cse@gmail.com);

Contributing authors: [20243660751hustagna@juniv.edu](mailto:20243660751hustagna@juniv.edu);

[ajoy.cc.75@gmail.com](mailto:ajoy.cc.75@gmail.com);

**Abstract**

A household service robot has to watch for objects on the floor while it moves, and on a small, battery-powered computer the object detector is often the most expensive part of that job. Adaptive perception, which runs the detector only when and where it seems necessary, is a popular way of reducing this cost, but it is rarely evaluated in a closed loop where a missed detection can leave a task unfinished or cause a collision. We tested it in 25 photorealistic home scans that were held out from all training, with obstacles that reach the planner only through the detector. The evaluated controller learns when to run the detector and adds hand-designed rules for where to look and which model size to use, a safety shield and an experience memory. The task, its validity gates and the success criterion were fixed in a preregistration, and the comparison against twelve baselines was run once (120 paired episodes per policy). The controller cut computation by 85% relative to running the heavy detector on every frame, but it reached the goal in 55.0% of episodes, against 79.2% for that baseline, and collided in 8.3%, against 0.8% for the light detector run on every frame, so it did not meet the criterion. For designers of home service robots the message is practical: running the cheapest adequate detector on every frame is a strong default,

and on the measured processor it is also the only detector tier that completes within the 0.1 s control period. An adaptive scheduler should be adopted only after it beats that default in closed-loop tests.

**Keywords:** Service robots, Mobile robot navigation, Adaptive perception, Embedded computing, Object detection, Closed-loop evaluation, Preregistration

## 1 Introduction

Cleaning, delivery and assistive robots, and research platforms for domestic mobile manipulation such as the Human Support Robot [1], share a basic requirement: they must cross rooms in which furniture is moved and objects are left lying around, without hitting anything and without stopping for no reason. Most of them rely on a camera and a neural object detector for this, and they run it on an embedded computer whose power budget also has to cover localisation, planning and interaction with people. Computation and actuation draw on the same battery, and the computing workload measurably shapes a mobile robot’s energy use [2]; on small autonomous vehicles, the onboard computer can even be what limits the safe speed [3]. The usual design nonetheless runs the detector on every frame, on the whole image and with one model, even when nothing in view has changed. It is therefore tempting to make perception adaptive.

There is good reason to expect savings. Video analytics systems skip, crop or reconfigure inference with little loss of accuracy [4–8], and dynamic neural networks scale their computation to the input [9–14]. These results, however, come from fixed datasets, where a skipped frame only lowers recall. A service robot pays a different price. It may stall in front of an object it has misjudged, give up on its task, or bump into something, and what happens depends on its next moves. We could not find a study that measures whether the promised savings survive in this closed loop under conditions in which a negative answer would also be taken seriously.

This paper provides one. We evaluate a perception controller that learns *when* to run its detector from inexpensive signals, and combines this with hand-designed rules for *where* in the image to look and *which* model size to use, a safety shield that bounds how long the robot may travel without looking, and an experience memory. The robot navigates photorealistic scans of real homes [15] in the Habitat simulator [16]. Because the outcome of such a comparison can easily be shaped after the fact, we fixed the task, the conditions under which the task counts as informative, and the success criterion in a preregistration before running the controller [17], and we ran the main comparison only once.

The answer is negative, and we think it is useful precisely for that reason. The controller saves computation, but it completes fewer navigation tasks and collides more often than a robot that simply runs its cheapest detector on every frame. This paper makes three contributions to service robotics:

- a benchmark of home navigation in which perception, rather than a ground-truth map, decides whether the robot reaches its goal, together with validity gates that show, before any method is run, whether a task can tell compute policies apart (Sections 3 and 5);
- a preregistered comparison of a learned perception scheduler with twelve baselines in 25 held-out homes, which shows that the scheduler does not pay off (Section 7);
- practical guidance on how to choose and test compute-saving perception for a service robot, drawn from these results (Section 8).

## 2 Related work

**Adaptive video analytics.** Glimpse [4] sends a frame for recognition only when it differs enough from the last processed frame, and NoScope [5] screens frames with difference detectors and small specialised models. Chameleon [6] retunes a video pipeline over time, Reducto [7] filters frames on the camera with adaptive low-level features,

ApproxDet [8] adjusts a detector to content and resource contention on mobile devices, and deep feature flow [18] propagates features from sparse key frames. These systems usually assume a fixed camera and optimise accuracy against cost; nothing in them moves or acts.

**Dynamic neural inference.** Spatially adaptive computation [10], glance-and-focus classification [11], block skipping [12, 13] and early exits [14] all adapt computation to the input, as surveyed by Han et al. [9]. They decide per image, on fixed datasets and without a closed loop.

**Adaptive perception on robots.** Recent work carries these ideas onto robotic platforms. Adaptive-Cloud switches between detector backbones of different widths according to scene complexity and cuts the computation of light detection and ranging (LiDAR) 3D object detection by 41% with a small loss in accuracy [19]. Agile3D uses a learned controller to choose among execution branches of a 3D detector on an embedded graphics processing unit (GPU) under resource contention [20]. On small planetary rovers, a lightweight saliency map cuts the computational cost of detecting a lander by about 97% [21]. These studies measure savings as detection accuracy against cost on recorded data. A second line of work allocates sensing rather than computation: it points a sensor with a limited field of view where collision risk is highest [22], or makes a planner react to the uncertainty of its perception model [23]. Neither line tests whether skipping or cheapening detections pays off when the robot’s task success depends on every frame it does not process.

**Perception and safe navigation.** Late perception limits how fast a robot can move safely [24], and streaming perception evaluates a detector’s accuracy and latency jointly, because the world changes while a frame is being processed [25]. A local planner such as the dynamic window approach [26] running on an occupancy grid [27], or on a dynamic cost map as in recent work on an accompanying cane-type robot [28], cannot avoid an obstacle that is missing from its map.

**Evaluation practice.** Navigation agents are normally judged by success and path efficiency [29], and it has been argued that replicable robotics research needs protocols that are fixed in advance [30]. Preregistration, common in other empirical fields, limits the flexibility of analyses chosen after the data are seen [17, 31]. Simulation is the usual test bed for embodied navigation; its results predict real-world performance only when the simulator is set up with care [32], and modular pipelines with learned perception, like the one studied here, have transferred to real homes better than end-to-end policies [33].

**Research gap.** To our knowledge, no earlier work has tested the visual compute schedule of a service robot inside a closed navigation loop, with cost weighed against task success and safety, under a protocol that fixes the comparison before the method is run. Table 1 summarises how the closest lines of work differ from this study.

**Table 1** How this study differs from the closest lines of work

| Line of work                           | Adaptive compute | Closed loop | Preregistered |
|----------------------------------------|------------------|-------------|---------------|
| Adaptive video analytics [4, 5, 7, 8]  | yes              | no          | no            |
| Dynamic neural inference [9, 10, 14]   | yes              | no          | no            |
| Adaptive robot perception [19–21]      | yes              | no          | no            |
| Sensing allocation for safety [22, 23] | no               | yes         | no            |
| This study                             | yes              | yes         | yes           |

Adaptive compute: the method changes how much detector computation is spent per frame. Closed loop: the method is judged by the success and safety of a robot whose motion depends on its output. Preregistered: the task, the comparison and the success criterion were fixed in advance.

### 3 A benchmark in which perception matters

In a standard embodied-AI simulator the navigation mesh doubles as ground truth, so a robot never runs into anything, whatever its perception does. We therefore place household objects as obstacles that are left out of the navigation mesh. The local

planner sees them only through a robot-centred occupancy grid built from detections. The grid also holds the static floor plan, which mirrors a home robot that has mapped the house once and knows its walls, but not what has been left on the floor since; floor plans are reliable enough to anchor a robot’s map in practice [34], whereas objects on the floor are not in any plan. A missed or weak detection can now cause a collision, or leave the robot stuck behind a wrong belief. Layouts in which the robot cannot pass around the objects are rejected, and an episode is kept only if an ideal detector can complete it, so that every failure can be traced to perception.

**Metrics.** An episode succeeds if the robot reaches the goal within a step budget proportional to the start–goal geodesic distance; it fails on a collision or a timeout. We report success rate, collision rate, success weighted by path length (SPL) [29], detector calls per episode and computation per episode in giga floating-point operations (GFLOPs). Each detector call is charged at its measured cost, including region-of-interest (ROI) crops at their own input size, and the controller’s own computation is added to the total.

**Validity gates.** A task can only reveal differences between compute policies if it is neither too easy nor too hard. Using baselines only, G1 (ceiling) requires the heavy detector on every frame to succeed in at least 90% of episodes with at most 5% collisions. G2 requires the light detector on every frame, and G3 a policy that processes only every eighth frame, to fail in at least 20% of episodes, so that both the choice of model and the choice of timing have consequences. Section 5 gives the definitions we used.

## 4 Evaluated method

Figure 1 shows the loop. At each control step the controller decides whether to run the detector (WHEN), on which tiles of the image (WHERE) and with which model size (WHAT). A safety shield can override the decision and an experience memory can

adjust it. Detections update a tracker, the tracker updates the occupancy grid, and a dynamic-window planner chooses the next velocity command.

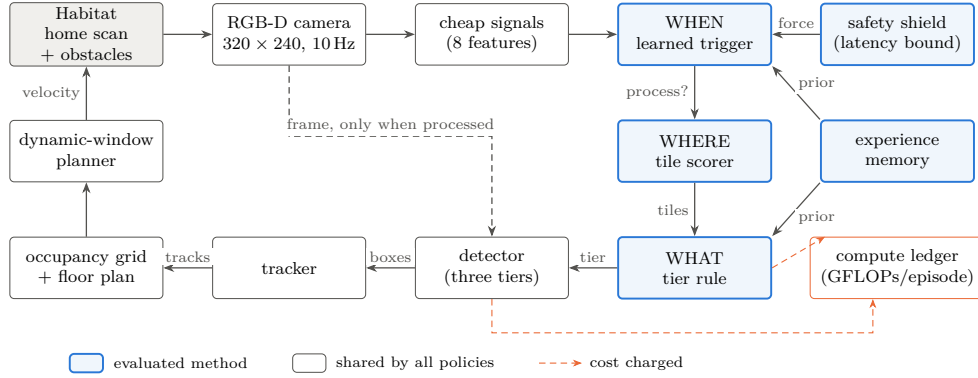


**Fig. 1** The closed perception-action loop of the evaluated method. Blue boxes are the parts of the evaluated method; white boxes are shared by all policies. Obstacles are absent from the navigation mesh, so the planner avoids only what the detector has put into the occupancy grid. The dashed black arrow carries the camera frame to the detector only when the controller decides to process it. Orange dashed arrows show that every detector call and the controller’s own computation are charged to a compute ledger.

**WHEN: a learned trigger.** Eight cheap signals are computed on every frame and charged to the controller: frame-difference energy and edge-histogram distance on a thumbnail, translational and rotational ego-motion, the time since the last detection, how stale the current belief is, and the minimum range and time-to-collision in a forward depth corridor. A two-layer network  $f_\theta$  with 16 hidden units turns these features  $x_t$  into the probability  $p_t = \sigma(f_\theta(x_t))$  that the current belief is *missing* a relevant obstacle, and the detector runs when  $p_t \geq 0.5$ . The network is trained offline on frames from training homes. Each frame is labelled by an oracle that knows whether processing it would have changed a planning-relevant part of the belief, and the loss is a class-weighted binary cross-entropy. We also train a variant that predicts information gain, the quantity most filtering methods optimise, and test it as a separate trigger.

**WHERE: tile selection.** The image is split into  $4 \times 3$  tiles, and the detector receives the bounding box of the chosen tiles. Tile  $i$  scores  $s_i = \sigma(\mathbf{w}^\top \phi_i) + \lambda c_i$ , where  $\phi_i$  holds motion energy, projected track confidence, a relevance prior and a bias, and  $c_i$

is the share of the tile covered by the robot’s braking corridor; tiles with  $s_i \geq 0.35$  are kept. The weights are set by hand, because a logistic regression fitted on training data did not beat them on held-out episodes (validation average precision 0.315 against 0.309).

**WHAT: model size.** The three tiers are one Faster R-CNN detector [35] with a ResNet-50 backbone and a feature pyramid network (FPN) [36, 37], run at three input sizes (Table 2). We used a resolution ladder because ladders built from different architectures were not monotone in latency on our central processing unit (CPU); a MobileNet-based detector [38], for example, needed the fewest FLOPs but was the slowest. The tier follows the time-to-collision  $T$ : heavy if  $T < 2.5$  s, medium if  $T < 5$  s, light otherwise.

**Table 2** Detector tiers, measured on the deployment CPU

| Tier   | Resolution       | GFLOPs | Latency (ms) |
|--------|------------------|--------|--------------|
| light  | $256 \times 192$ | 26.8   | 99           |
| medium | $427 \times 320$ | 62.1   | 154          |
| heavy  | $640 \times 480$ | 123.6  | 244          |

GFLOPs and latency are per detector call, measured on the deployment CPU (Apple M1 Pro).

**Safety shield.** Given the nearest range  $d$ , speed  $v$ , control period  $\Delta t$  and stopping margin  $r = 0.4$  m (our robot’s braking distance), a  $(k+1)$ -th skipped frame in a row is allowed only while  $(k+1)v\Delta t \leq \alpha \max(d-r, 0)$ . Otherwise a detection is forced, which bounds the worst-case detection delay by construction.

**Experience memory.** Each entry pairs an appearance descriptor of the scene with the outcome that followed. An episodic level recalls the nearest neighbours, and a parametric ridge model is the level that could generalise to homes the robot has not seen. The memory may force or suppress a detection and may lower the time-to-collision used by the tier rule; this is safe because the shield bounds the delay

**Table 3** Validity gates measured with baselines only on 24 tuning episodes

| Gate        | Policy       | Metric     | Measured | Required    | Result |
|-------------|--------------|------------|----------|-------------|--------|
| G1 ceiling  | always-heavy | success    | 0.917    | $\geq 0.90$ | pass   |
| G1 ceiling  | always-heavy | collisions | 0.000    | $\leq 0.05$ | pass   |
| G2 tier     | always-light | failures   | 0.292    | $\geq 0.20$ | pass   |
| G3 temporal | fixed skip-8 | failures   | 0.417    | $\geq 0.20$ | pass   |

Failures are episodes that end in a collision or a timeout (Amendment 1).  
 Values are fractions of the 24 episodes.

independently. It is fitted offline from an oracle log, since a robot online only learns the outcome of frames it chose to process.

## 5 Preregistration

We changed the task once, for a reason that has nothing to do with our method. On an earlier, uncertified development benchmark the robot could stop within a single 0.1 s control step, so detection range hardly affected safety and the cheapest detector never collided. We added an acceleration limit ( $1.0 \text{ m/s}^2$ ) and froze a harder task: a top speed of 0.9 m/s, three static household objects per episode scaled by 0.7–1.1, and start–goal distances of 6–18 m. Before the controller was run on this task, we committed a preregistration to the project repository that fixed the task, the gates, the success criterion and what we would report in either case (2 September 2026), following common practice for preregistered studies [31].

On the final backend G1 held, but cheaper or skipped perception mostly produced *failed* episodes, in which the robot stalled behind false or stale tracks, rather than collisions. We therefore adopted an amendment, with baselines only and before any result of our method existed (30 September 2026), that counts failed episodes (collisions or timeouts) in G2 and G3 and keeps the 20% thresholds. On 24 fresh tuning episodes the task is admissible (Table 3); the original, collision-only G2 and G3 measured 0.000 and 0.125.

**Success criterion.** Our method counts as better than the baselines only if (i) its success rate is within 0.05 of the best baseline’s, (ii) its collision rate is within 0.05 of the lowest baseline’s, and (iii) it needs significantly fewer GFLOPs (paired tests with Holm correction across the baseline family,  $\alpha = 0.05$ ). Otherwise the negative result is reported. A procedural gate also required the trained trigger to beat the validation average precision of always looking by more than 0.02 before the main comparison could start. Every number in this paper comes from a run that passed automated certification checks: trained detector, recorded weight hashes, a clean code revision, a monotone cost profile, and learned parts fitted only on training or tuning homes.

## 6 Experimental setup

**Homes and robot.** We use Habitat-Sim 0.3.3 [16, 39] with Habitat-Matterport 3D (HM3D) home scans [15] and Yale-CMU-Berkeley (YCB) household objects [40] as obstacles. The 100 HM3D validation homes were split by scene into 63 training, 12 tuning and 25 test homes: learned parts are fitted on training homes, gates are measured on tuning homes and every comparison runs on test homes. The differential-drive robot carries a  $320 \times 240$  colour and depth (RGB-D) camera ( $79^\circ$  field of view, 0.88 m high), runs at a 0.1 s control period with limits of 0.9 m/s, 1.5 rad/s and  $1.0 \text{ m/s}^2$ , and plans with a dynamic-window planner [26] (3s horizon) on a 0.15 m occupancy grid [27].

**Detectors.** The three tiers (Table 2) were trained on 80,753 renderings from 61 training homes and temperature-calibrated [41] on 8,000 tuning renderings, and detections are kept above a calibrated score of 0.4. Costs come from a profile measured on the deployment CPU, an Apple M1 Pro, which here stands in for an embedded service-robot computer. Because the camera renders  $320 \times 240$  images, the medium and heavy tiers upsample the same picture. Figure 2 shows typical inputs and outputs.

**Learned trigger.** The trigger was trained on 41,625 oracle-labelled frames from 50 training-home episodes (38 for training, 12 for validation; 8.7% positive) with

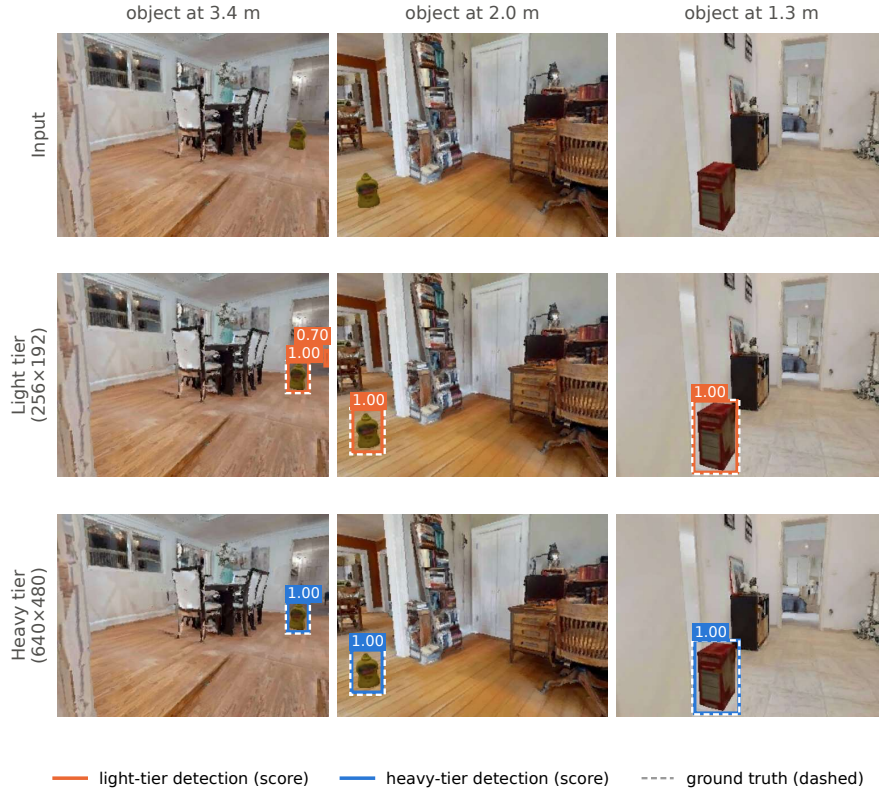


**Fig. 2** Detector inputs and outputs in tuning homes held out from detector training. Top row: camera frames, each with one household object at a different range. Middle and bottom rows: detections of the light and heavy tiers at the operating threshold, labelled with calibrated scores; the dashed white box is the ground truth. The light tier also reports a false detection (score 0.70) on a sliver at the right edge of the first frame.

Adam [42], a learning rate of 0.003, one full-batch update per step, at most 300 steps and early-stopping patience of 40.

**Policies.** The twelve baselines are the heavy and light detectors on every frame (*always-heavy*, *always-light*); random selection of 25% of frames; processing every fourth, sixth or eighth frame; a pixel-difference trigger after Glimpse [4]; an edge-feature trigger after Reducto [7]; a motion-gated trigger; centre and motion-ranked ROI crops; and a confidence cascade. Our method is tested with its parts added one

507 by one: trigger with shield (*ours-T*), plus WHERE (*ours-TS*), plus WHAT (*ours-*  
508 *TSW*), plus memory (*ours-full*, the preregistered method). Running no perception at  
509 all serves as a lower bound.  
510

511 **Protocol and statistics.** All 17 policies run on the same 40 test-home episodes for  
512 each of three seeds, giving 120 paired episodes per policy. A Friedman test [43], which  
513 becomes Cochran’s Q [44] for binary outcomes, comes before the pairwise Wilcoxon  
514 signed-rank tests [45], which use Holm’s correction [46], paired bootstrap 95% confi-  
515 dence intervals (CIs) [47] and Cliff’s  $\delta$  [48]; Kendall’s  $W$  [49] gives the effect size. In a  
516 separate sweep, every trigger is run across its own threshold range (43 settings of 20  
517 episodes, heavy detector, full frame, no shield). The memory study uses 80 episodes  
518 in held-out homes per condition.  
519

520 **Use of AI tools.** The authors used an AI assistant to help write and review the  
521 manuscript. The authors checked all results and text, and take full responsibility for  
522 the work.  
523

## 531 7 Results

### 532 7.1 The preregistered verdict

533 The policies differ on every metric (Friedman test,  $p < 10^{-4}$  for collisions, success,  
534 detector calls, GFLOPs and SPL; Kendall’s  $W$  0.17–0.69). Table 4 and Figure 3 sum-  
535 marise the main comparison. Always-heavy has the best baseline success rate (0.792),  
536 which puts the success bar at 0.742, and always-light the lowest baseline collision rate  
537 (0.008), which puts the collision bar at 0.058. Ours-full succeeds in 0.550 of episodes  
538 and collides in 0.083, so it fails conditions (i) and (ii). It does satisfy condition (iii)  
539 against 11 of the 12 baselines: at 11,609 GFLOPs per episode it is significantly cheaper  
540 than every baseline except the centre crop (10,097), 85% cheaper than always-heavy  
541 and 34% cheaper than always-light.  
542

The gaps are large. Compared with always-light, ours-full’s success rate is lower by 21.7 percentage points (95% CI 10.0 to 33.3;  $p = 0.0004$ ) and its collision rate higher by 7.5 points (3.3 to 12.5;  $p = 0.003$ ); against always-heavy the gaps are 24.2 and 6.7 points, and both are significant. None of the partial variants meets the criterion either. **The preregistered criterion is not met.**

**Table 4** Main comparison in test homes, pooled over three seeds

| Policy                          | Succ. | Coll. | SPL   | Calls | GFLOPs | $p_{\text{Holm}}$  |
|---------------------------------|-------|-------|-------|-------|--------|--------------------|
| <i>Baseline family</i>          |       |       |       |       |        |                    |
| always-heavy                    | 0.792 | 0.017 | 0.737 | 621.6 | 76805  | $< 10^{-4\dagger}$ |
| pixel-difference                | 0.742 | 0.083 | 0.693 | 574.1 | 70939  | $< 10^{-4\dagger}$ |
| edge-feature                    | 0.408 | 0.250 | 0.360 | 223.6 | 27625  | $< 10^{-4\dagger}$ |
| random 25%                      | 0.400 | 0.158 | 0.342 | 217.1 | 26829  | $< 10^{-4\dagger}$ |
| fixed skip-4                    | 0.400 | 0.158 | 0.368 | 209.4 | 25869  | $< 10^{-4\dagger}$ |
| ROI motion                      | 0.450 | 0.150 | 0.391 | 207.3 | 21797  | $< 10^{-4\dagger}$ |
| motion-gated                    | 0.425 | 0.117 | 0.380 | 142.7 | 17627  | $< 10^{-4\dagger}$ |
| fixed skip-6                    | 0.417 | 0.183 | 0.381 | 141.9 | 17528  | $< 10^{-4\dagger}$ |
| always-light                    | 0.767 | 0.008 | 0.712 | 652.8 | 17479  | $< 10^{-4\dagger}$ |
| confidence cascade              | 0.433 | 0.125 | 0.396 | 204.9 | 16677  | $< 10^{-4\dagger}$ |
| fixed skip-8                    | 0.458 | 0.142 | 0.405 | 104.7 | 12934  | $0.023^\dagger$    |
| ROI centre                      | 0.358 | 0.292 | 0.313 | 216.0 | 10097  | 0.183              |
| <i>Ours, axes added in turn</i> |       |       |       |       |        |                    |
| ours-T                          | 0.475 | 0.167 | 0.432 | 317.1 | 39183  | —                  |
| ours-TS                         | 0.475 | 0.125 | 0.433 | 343.0 | 29888  | —                  |
| ours-TSW                        | 0.483 | 0.108 | 0.433 | 333.8 | 12394  | —                  |
| ours-full                       | 0.550 | 0.083 | 0.497 | 291.8 | 11609  | ref.               |
| no perception                   | 0.050 | 0.950 | 0.050 | 0.0   | 0      | —                  |

120 paired episodes per policy. Succ., success rate; Coll., collision rate; calls and GFLOPs are per episode.  $p_{\text{Holm}}$  tests each baseline’s GFLOPs against ours-full (paired Wilcoxon signed-rank test, Holm-corrected over the twelve baselines);  $^\dagger$  marks baselines that use significantly more computation than ours-full.

For a home robot, then, the cheapest detector is good enough as long as it runs on every frame. Always-light succeeds within 2.5 points of always-heavy (not significant), collides no more often, and needs only 23% of its computation. Every policy that processes fewer than half of the frames loses success: the nine baselines in that group

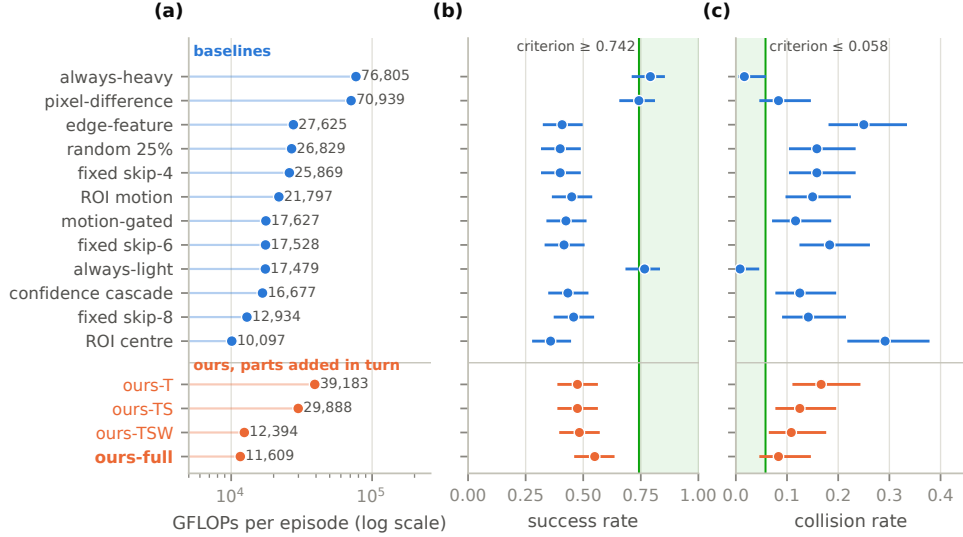


**Fig. 3** Computation, success and collisions of every policy in the main comparison. (a) GFLOPs per episode (logarithmic scale); (b) success rate; (c) collision rate. Dots are means over the 120 paired episodes and bars are Wilson 95% confidence intervals. The green line and shading mark the region the preregistered criterion requires: success at least 0.742 and collisions at most 0.058. Baselines are sorted by computation. Running no perception (success 0.050, collisions 0.950, no computation) is not drawn.

succeed in 35.8–45.8% of episodes. Ours-full has the highest success rate within this group, although this is an exploratory observation outside the criterion.

## 7.2 How the method fails

Figure 4 shows two test episodes in which always-light reaches the goal and ours-full does not. In the first, ours-full stops beside an object after about 600 steps and never moves on; in the second, it collides while detouring around two objects. Across the 40 episodes of seed 0, always-light succeeds where ours-full fails in 12 episodes, and the reverse happens in 5. On the gate episodes, the failures of cheap or skipped perception were stalls behind false or stale tracks rather than contacts. For a service robot this means an unfinished task more often than damage, and it is the main reason why task success, not only collisions, must be measured.

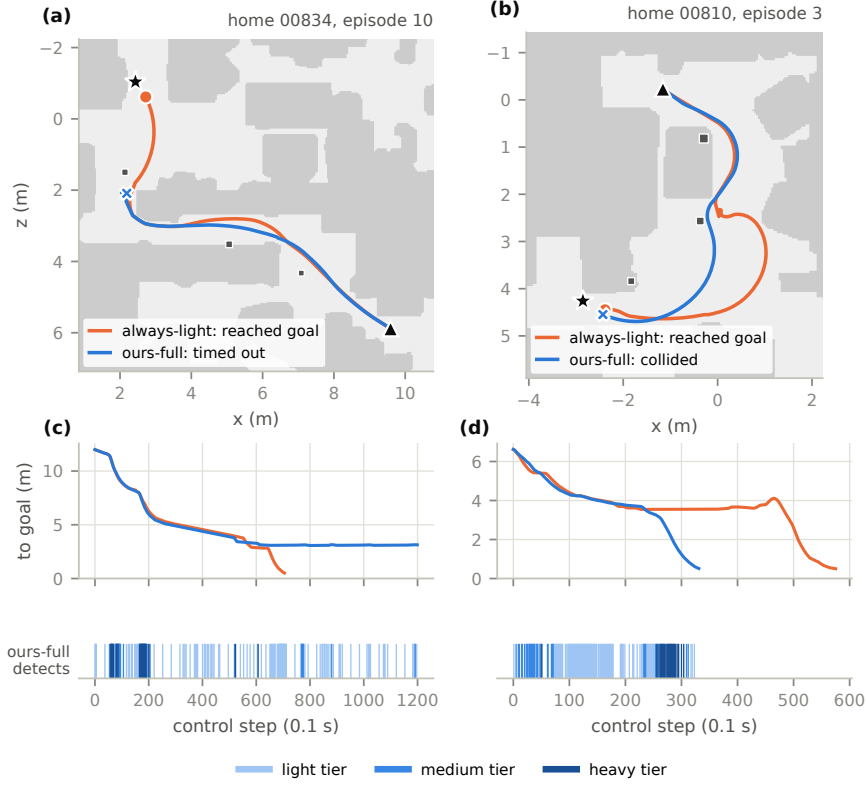


**Fig. 4** Two test episodes in which always-light reaches the goal and ours-full does not. (a) A timeout and (b) a collision, both from seed 0. Top: trajectories on the floor plan, objects in dark grey, start  $\blacktriangle$ , goal  $\star$ . Middle: distance to the goal. Bottom: steps at which ours-full ran its detector, by tier. In seed 0, always-light succeeds where ours-full fails in 12 of 40 episodes and the reverse occurs in 5.

### 7.3 Sweeping the WHEN axis

Comparing methods at a single operating point tends to flatter adaptive methods, so every trigger was swept across its threshold (Figure 5). The learned trigger never reaches the frontier of detector calls against collisions. At its most permissive threshold it makes as many calls as always-heavy (547 against 548), yet collides in 25% of episodes instead of none and succeeds in 55% instead of 90%. At about 90 calls per episode, processing every eighth frame collides in 15% of episodes, compared with 55% for the learned trigger at 95 calls, so simple fixed-rate skipping is the stronger choice at equal budget. The information-gain trigger never asked for a detection at

any threshold (95% collisions), even though it ranked validation frames well (average precision 0.653 against 0.358 for always looking); we have not yet found out whether its target or its calibration is to blame, and we draw no conclusion from it.

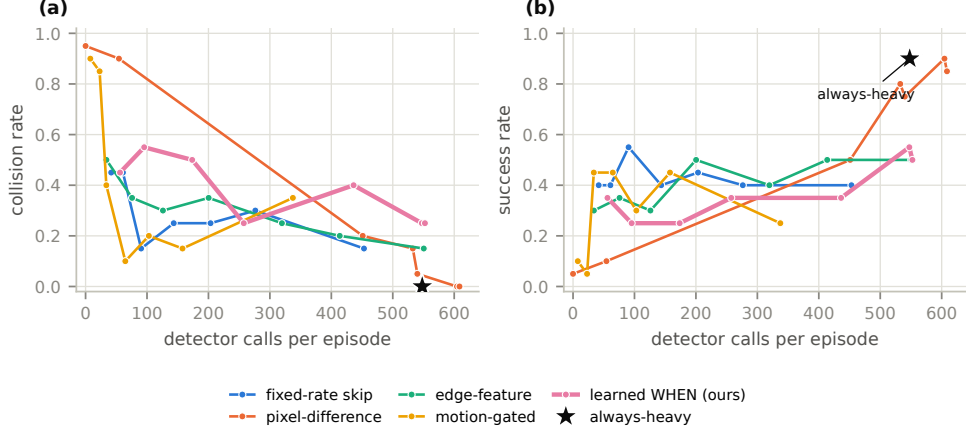


**Fig. 5** Each trigger swept across its own threshold. (a) Collision rate and (b) success rate against detector calls, 20 episodes per setting, heavy detector on the full frame, without the shield. The information-gain trigger made no detector calls at any threshold and is not drawn.

## 7.4 Training, ablation and controller overhead

The trigger passed its gate (validation average precision 0.207 against 0.088 for always looking) but did not converge (Figure 6): training hit the 300-step limit while validation average precision was still rising (0.198 at step 250, best 0.207 afterwards). The information-gain variant behaved the same way. The budget was fixed before the main comparison, so a longer schedule remains untested.

Most of the computational saving comes from the WHAT rule rather than from the learned trigger (Table 4, Figure 7). Ours-T processes about half of the frames with the heavy detector (39,183 GFLOPs); adding WHERE lowers this to 29,888, adding the tier rule to 12,394, since most frames are then handled by the light or medium tier, and adding memory to 11,609. Success stays between 47.5% and 55.0%, while

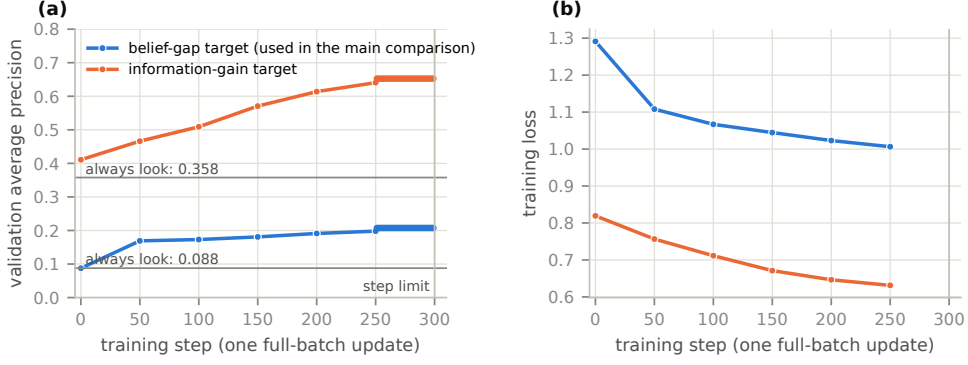


**Fig. 6** Training of the two trigger variants. (a) Validation average precision and (b) training loss, logged every 50 steps. The thick segment marks the best validation score, reached after step 250; grey lines mark the score of always looking.

collisions fall as parts are added. The controller itself costs at most 0.008 GFLOPs per episode, below  $10^{-4}\%$  of the detector cost of any perceiving policy.

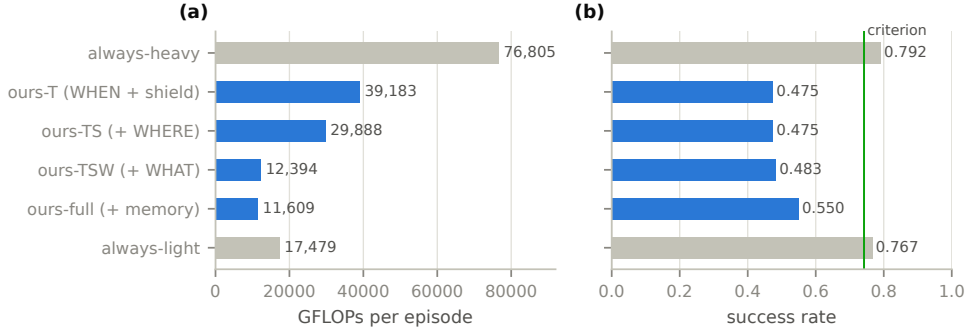


**Fig. 7** The evaluated method with its parts added in turn. (a) Computation and (b) success rate, shown between the two every-frame baselines. The green line in (b) is the success bar of the preregistered criterion.

## 7.5 Does compute need transfer between homes?

A robot that remembers where careful perception was needed could save computation in new homes, provided that compute need depends on how a place looks. We tested this on 80 episodes per condition in homes the memory had never seen (Table 5, Figure 8). The parametric level cut detector calls from 323.7 to 271.2 per episode, a

16% reduction that is not significant (Holm  $p = 0.119$ ), while success fell from 46.3% to 40.0% and collisions rose from 10.0% to 20.0%. A memory keyed by ground-truth pose, which is deliberately privileged, reached 229.2 calls, a significant 29% reduction. Compute need is thus easier to predict from where the robot is than from what it sees. Over three visits to the same scene (26 episodes per visit) detector calls even rose, from 228.5 to 243.8 and 252.5, so the memory did not get better with experience of a place.

**Table 5** Memory study in held-out homes (80 episodes per condition)

| Memory                | Calls | Collisions | Success | $p_{\text{Holm}}$ (calls) |
|-----------------------|-------|------------|---------|---------------------------|
| none                  | 323.7 | 0.100      | 0.463   | —                         |
| episodic recall       | 310.1 | 0.150      | 0.487   | 0.301                     |
| parametric model      | 271.2 | 0.200      | 0.400   | 0.119                     |
| episodic + parametric | 264.6 | 0.075      | 0.575   | 0.119                     |
| pose-keyed control    | 229.2 | 0.175      | 0.350   | $< 0.001^*$               |

$p_{\text{Holm}}$  compares detector calls with no memory (paired Wilcoxon signed-rank test, Holm-corrected); \* significant at  $\alpha = 0.05$ . The pose-keyed control receives ground-truth position, which the real system never has.

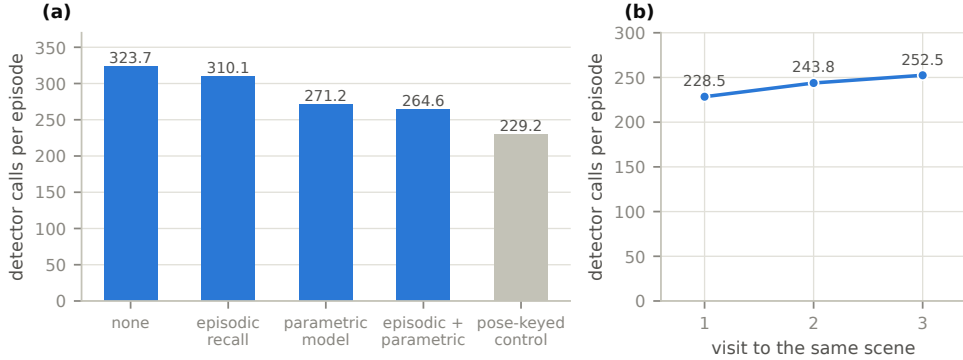


**Fig. 8** Detector calls per episode in the memory study. (a) By memory condition; (b) across three visits to the same scene with memory persisting. The pose-keyed control (grey) receives ground-truth position, which the real system never has.

## 8 Discussion

**Why the light detector on every frame wins.** Static household objects stay in view for many frames before the robot reaches them, so a detector that runs on every frame confirms them early. At the short ranges where collisions are decided, the light and heavy tiers detect about equally well, because the heavy tier only upsamples the same  $320 \times 240$  image. Skipping frames then saves little that the light tier has not already saved, and it leaves wrong tracks uncorrected for longer.

**Implications for the robot’s computing hardware.** The measured profile (Table 2) turns the result into a sizing rule. At the 0.1s control period, the light tier needs 26.8 GFLOPs per frame, about 270 GFLOP/s sustained, and its measured single-call latency (99 ms) just fits within one control period on the deployment CPU. The heavy tier needs 123.6 GFLOPs per frame, about 1240 GFLOP/s, and its latency (244 ms) is about 2.4 times the control period. A robot that runs the light tier on every frame can therefore be built around a much smaller computer than one designed for the heavy tier, and it does not depend on a scheduler to stay within its budget. Recent real-time detectors [50, 51] improve the accuracy obtainable at such a budget, so making the every-frame detector cheaper is a route that deserves at least as much attention as scheduling a more expensive one.

**Practical guidance for service robot designers.** Four points follow from our results. (1) Start from the cheapest adequate detector running on every frame. In our homes it matched the heavy detector’s success at 23% of the cost, and any adaptive scheduler should have to beat it in closed-loop tests before it is deployed. (2) Measure task completion, not only collisions; in homes, poor perception mostly leaves tasks unfinished. (3) Evaluate a scheduler across its whole budget range, since at equal budget plain fixed-rate skipping beat the learned trigger, and good offline accuracy did not predict a closed-loop benefit. (4) Before claiming that a robot carries experience from one home to another, compare against a position-keyed control.

**Why the result can be trusted.** The comparison was planned before it was run: the task, the gates, the success criterion and the reporting commitment were fixed in advance, the main comparison was executed once, and the policies were compared on identical episodes with corrected statistics. After the preregistration we corrected implementation defects found by tests or inspection (ground-truth box offsets, occluded objects labelled as visible, an inverted projection of the WHERE prior, non-deterministic room seeding, crops charged below their measured cost), fixed the data pipeline before detector training, and corrected planner and tracker defects using baselines only. None of these changes was made after our method’s results were seen, and the full log is part of the released materials.

**Limitations.** The study is simulation-only, with one robot and static household objects; with a moving obstacle even the heavy detector collided in our setup, so no dynamic task was admissible. Simulated navigation can overstate real-world performance [32]. Here only contact with the placed objects counts as a collision, and the robot may slide along walls; this applies to all policies alike, but the absolute success rates should not be read as real-world rates. The homes are a partition of the HM3D validation split. The low camera resolution may favour the light tier, the learned trigger did not converge, WHERE used hand-set weights, the braking-corridor prior never became active at the preregistered lookahead, the shield’s margin fits one robot only, detector latency is charged as cost without delaying detections, although it matters for safety in closed loop [25], and the statistics treat episodes rather than homes as the unit of analysis. Tests on a real robot in real homes [33], a higher-resolution camera and latency in the loop are the natural next steps.

## 9 Conclusions

We asked whether learning when to run a detector pays off for a home service robot whose safety depends on what it sees, and we fixed the task, the gates and the success criterion before finding out. It does not. The evaluated controller uses 85% less computation than the heavy detector on every frame and is cheaper than 11 of 12 baselines, yet it finishes fewer navigation tasks and collides more often than the light detector run on every frame. For service robots working in homes, that simple and cheap default is the benchmark any adaptive perception method has to beat. It also eases hardware design: on the measured processor only the light tier completes within one control period, so a robot built around it needs a smaller onboard computer and no scheduler to stay within its budget. The benchmark, gates and evaluation practices described here make the closed-loop test straightforward to run.

## Abbreviations

|        |                                           |
|--------|-------------------------------------------|
| AI     | Artificial intelligence                   |
| CI     | Confidence interval                       |
| CPU    | Central processing unit                   |
| FLOPs  | Floating-point operations                 |
| FPN    | Feature pyramid network                   |
| GPU    | Graphics processing unit                  |
| GFLOPs | Giga ( $10^9$ ) floating-point operations |
| HM3D   | Habitat-Matterport 3D (dataset)           |
| LiDAR  | Light detection and ranging               |
| R-CNN  | Region-based convolutional neural network |
| RGB-D  | Colour (red, green, blue) and depth       |
| ROI    | Region of interest                        |
| SPL    | Success weighted by path length           |
| YCB    | Yale-CMU-Berkeley (object set)            |

## Declarations

**Availability of data and materials.** The code, configuration files, episode definitions, result files and the preregistration with its version history that support the findings of this study are available from the corresponding author on reasonable request. The HM3D scenes [15] are available from Matterport under their terms of use and are not redistributed; the YCB objects [40] are publicly available.

**Competing interests.** The authors declare that they have no competing interests.

**Funding.** The authors did not receive support from any organization for the submitted work.

**Authors' contributions.** MASR conceived and designed the study, developed the software, conducted the experiments and statistical analysis, and wrote the initial

manuscript draft. HH contributed to the analysis, validation, and revision of the manuscript. ASD contributed to the interpretation of the results, resource management, and critical revision of the manuscript. All authors reviewed and approved the final manuscript.

**Acknowledgements.** Not applicable.

## References

- [1] Yamamoto, T., Terada, K., Ochiai, A., Saito, F., Asahara, Y., Murase, K.: Development of Human Support Robot as the research platform of a domestic mobile manipulator. *ROBOMECH Journal* **6**(1), 4 (2019) <https://doi.org/10.1186/s40648-019-0132-3>
- [2] Wu, C., Haghbayan, H., Malik, A., Miele, A., Plosila, J.: Runtime energy-efficient control policy for mobile robots with computing workload and battery awareness. In: 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 16314–16321 (2025). <https://doi.org/10.1109/iros60139.2025.11246814>
- [3] Krishnan, S., Wan, Z., Bhardwaj, K., Jadhav, N., Faust, A., Reddi, V.J.: Roofline model for UAVs: A bottleneck analysis tool for onboard compute characterization of autonomous unmanned aerial vehicles. In: 2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 162–174 (2022). <https://doi.org/10.1109/ispass55109.2022.00023>
- [4] Chen, T.Y.-H., Ravindranath, L., Deng, S., Bahl, P., Balakrishnan, H.: Glimpse: Continuous, real-time object recognition on mobile devices. In: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys) (2015). <https://doi.org/10.1145/2809695.2809711>

- 1059 [5] Kang, D., Emmons, J., Abuzaid, F., Bailis, P., Zaharia, M.: NoScope: Optimizing  
1060 neural network queries over video at scale. *Proceedings of the VLDB Endowment*  
1061 **10**(11), 1586–1597 (2017) <https://doi.org/10.14778/3137628.3137664>  
1062  
1063  
1064 [6] Jiang, J., Ananthanarayanan, G., Bodik, P., Sen, S., Stoica, I.: Chameleon: Scal-  
1065 able adaptation of video analytics. In: *Proceedings of the ACM SIGCOMM*  
1066 *Conference* (2018). <https://doi.org/10.1145/3230543.3230574>  
1067  
1068  
1069 [7] Li, Y., Padmanabhan, A., Zhao, P., Wang, Y., Xu, G.H., Netravali, R.: Reducto:  
1070 On-camera filtering for resource-efficient real-time video analytics. In: *Pro-*  
1071 *ceedings of the ACM SIGCOMM Conference* (2020). [https://doi.org/10.1145/](https://doi.org/10.1145/3387514.3405874)  
1072 [3387514.3405874](https://doi.org/10.1145/3387514.3405874)  
1073  
1074  
1075 [8] Xu, R., Zhang, C.-l., Wang, P., Lee, J., Mitra, S., Chaterji, S., Li, Y., Bagchi,  
1076 S.: ApproxDet: Content and contention-aware approximate object detection for  
1077 mobiles. In: *Proceedings of the 18th ACM Conference on Embedded Networked*  
1078 *Sensor Systems (SenSys)* (2020). <https://doi.org/10.1145/3384419.3431159>  
1079  
1080  
1081 [9] Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y.: Dynamic neural net-  
1082 works: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*  
1083 **44**(11), 7436–7456 (2022) <https://doi.org/10.1109/TPAMI.2021.3117837>  
1084  
1085  
1086 [10] Figurnov, M., Collins, M.D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., Salakhut-  
1087 dinov, R.: Spatially adaptive computation time for residual networks. In: *Pro-*  
1088 *ceedings of the IEEE Conference on Computer Vision and Pattern Recognition*  
1089 *(CVPR)* (2017). <https://doi.org/10.1109/cvpr.2017.194>  
1090  
1091  
1092 [11] Wang, Y., Lv, K., Huang, R., Song, S., Yang, L., Huang, G.: Glance and focus:  
1093 a dynamic approach to reducing spatial redundancy in image classification. In:  
1094 *Advances in Neural Information Processing Systems (NeurIPS)* (2020). <https://doi.org/10.1109/neurips.2020.3187000>  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104

[//doi.org/10.48550/arXiv.2010.05300](https://doi.org/10.48550/arXiv.2010.05300) 1105  
1106  
1107  
[12] Wang, X., Yu, F., Dou, Z.-Y., Darrell, T., Gonzalez, J.E.: SkipNet: Learn- 1108  
ing dynamic routing in convolutional networks. In: Proceedings of the Euro- 1109  
pean Conference on Computer Vision (ECCV) (2018). [https://doi.org/10.1007/](https://doi.org/10.1007/978-3-030-01261-8_25) 1110  
[978-3-030-01261-8\\_25](https://doi.org/10.1007/978-3-030-01261-8_25) 1111  
1112  
1113  
1114  
[13] Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L.S., Grauman, K., Feris, 1115  
R.: BlockDrop: Dynamic inference paths in residual networks. In: Proceedings 1116  
of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 1117  
(2018). <https://doi.org/10.1109/cvpr.2018.00919> 1118  
1119  
1120  
1121  
1122  
[14] Teerapittayanon, S., McDanel, B., Kung, H.T.: BranchyNet: Fast inference via 1123  
early exiting from deep neural networks. In: Proceedings of the 23rd International 1124  
Conference on Pattern Recognition (ICPR) (2016). [https://doi.org/10.1109/icpr.](https://doi.org/10.1109/icpr.2016.7900006) 1125  
[2016.7900006](https://doi.org/10.1109/icpr.2016.7900006) 1126  
1127  
1128  
1129  
[15] Ramakrishnan, S.K., Gokaslan, A., Wijmans, E., Maksymets, O., Clegg, A., 1130  
Turner, J., Undersander, E., Galuba, W., Westbury, A., Chang, A.X., Savva, 1131  
M., Zhao, Y., Batra, D.: Habitat-Matterport 3D dataset (HM3D): 1000 large- 1132  
scale 3D environments for embodied AI. In: Advances in Neural Information 1133  
Processing Systems (NeurIPS) Datasets and Benchmarks Track (2021). <https://doi.org/10.48550/arXiv.2109.08238> 1134  
1135  
1136  
1137  
1138  
1139  
1140  
[16] Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, 1141  
J., Liu, J., Koltun, V., Malik, J., Parikh, D., Batra, D.: Habitat: A platform 1142  
for embodied AI research. In: Proceedings of the IEEE/CVF International Con- 1143  
ference on Computer Vision (ICCV) (2019). [https://doi.org/10.1109/iccv.2019.](https://doi.org/10.1109/iccv.2019.00943) 1144  
[00943](https://doi.org/10.1109/iccv.2019.00943) 1145  
1146  
1147  
1148  
1149  
1150

- 1151 [17] Nosek, B.A., Ebersole, C.R., DeHaven, A.C., Mellor, D.T.: The preregistration  
1152 revolution. *Proceedings of the National Academy of Sciences* **115**(11), 2600–2606  
1153 (2018) <https://doi.org/10.1073/pnas.1708274114>  
1154  
1155  
1156  
1157 [18] Zhu, X., Xiong, Y., Dai, J., Yuan, L., Wei, Y.: Deep feature flow for video recog-  
1158 nition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern*  
1159 *Recognition (CVPR)* (2017). <https://doi.org/10.1109/cvpr.2017.441>  
1160  
1161  
1162 [19] Mohammad, M.S., Kamal, U., Mukhopadhyay, S.: Adaptive-Cloud: Dynamic  
1163 computation control for 3D object detection from LiDAR point clouds. *IEEE*  
1164 *Robotics and Automation Letters* **10**(6), 6512–6519 (2025) [https://doi.org/10.](https://doi.org/10.1109/lra.2025.3568566)  
1165  
1166  
1167 [1109/lra.2025.3568566](https://doi.org/10.1109/lra.2025.3568566)  
1168  
1169  
1170 [20] Wang, P., Liu, Z., Bagchi, S., Xu, R., Bagchi, S., Li, Y., Chaterji, S.: Agile3D:  
1171 Adaptive contention- and content-aware 3D object detection for embedded GPUs.  
1172 In: *Proceedings of the 23rd Annual International Conference on Mobile Systems,*  
1173 *Applications and Services*, pp. 528–541 (2025). [https://doi.org/10.1145/3711875.](https://doi.org/10.1145/3711875.3729147)  
1174  
1175  
1176 [3729147](https://doi.org/10.1145/3711875.3729147)  
1177  
1178  
1179 [21] Sakamoto, K., Minegishi, R., Maeda, T., Kunii, Y.: Efficient lunar lander detec-  
1180 tion using a lightweight saliency map for small planetary rovers. *ROBOMECH*  
1181 *Journal* **12**(1), 10 (2025) <https://doi.org/10.1186/s40648-025-00300-x>  
1182  
1183  
1184  
1185 [22] Bena, R.M., Zhao, C., Nguyen, Q.: Safety-aware perception for autonomous colli-  
1186 sion avoidance in dynamic environments. *IEEE Robotics and Automation Letters*  
1187 **8**(12), 7962–7969 (2023) <https://doi.org/10.1109/lra.2023.3322345>  
1188  
1189  
1190  
1191 [23] Pohland, S., Tomlin, C.: Competency-aware planning for probabilistically safe  
1192 navigation under perception uncertainty. In: *2025 IEEE/RSJ International Con-*  
1193 *ference on Intelligent Robots and Systems (IROS)*, pp. 15291–15298 (2025).  
1194  
1195  
1196

|                                                                                                                                                                   |      |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| <a href="https://doi.org/10.1109/iros60139.2025.11246073">https://doi.org/10.1109/iros60139.2025.11246073</a>                                                     | 1197 |
|                                                                                                                                                                   | 1198 |
| [24] Falanga, D., Kim, S., Scaramuzza, D.: How fast is too fast? The role of perception                                                                           | 1199 |
| latency in high-speed sense and avoid. IEEE Robotics and Automation Letters                                                                                       | 1200 |
| 4(2), 1884–1891 (2019) <a href="https://doi.org/10.1109/lra.2019.2898117">https://doi.org/10.1109/lra.2019.2898117</a>                                            | 1201 |
|                                                                                                                                                                   | 1202 |
|                                                                                                                                                                   | 1203 |
|                                                                                                                                                                   | 1204 |
| [25] Yang, J., Liu, S., Li, Z., Li, X., Sun, J.: Real-time object detection for streaming                                                                         | 1205 |
| perception. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern                                                                                           | 1206 |
| Recognition (CVPR), pp. 5375–5385 (2022). <a href="https://doi.org/10.1109/cvpr52688.2022.00531">https://doi.org/10.1109/cvpr52688.</a>                           | 1207 |
|                                                                                                                                                                   | 1208 |
|                                                                                                                                                                   | 1209 |
|                                                                                                                                                                   | 1210 |
|                                                                                                                                                                   | 1211 |
|                                                                                                                                                                   | 1212 |
| [26] Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision                                                                                    | 1213 |
| avoidance. IEEE Robotics & Automation Magazine 4(1), 23–33 (1997) <a href="https://doi.org/10.1109/100.580977">https://doi.org/10.1109/100.580977</a>             | 1214 |
|                                                                                                                                                                   | 1215 |
|                                                                                                                                                                   | 1216 |
|                                                                                                                                                                   | 1217 |
|                                                                                                                                                                   | 1218 |
| [27] Elfes, A.: Using occupancy grids for mobile robot perception and navigation.                                                                                 | 1219 |
| Computer 22(6), 46–57 (1989) <a href="https://doi.org/10.1109/2.30720">https://doi.org/10.1109/2.30720</a>                                                        | 1220 |
|                                                                                                                                                                   | 1221 |
|                                                                                                                                                                   | 1222 |
| [28] Miyachi, Y., Kato, K., Hasegawa, Y.: Accompaniment and collision avoidance                                                                                   | 1223 |
| for a cane-type robot using dynamic cost maps and MPC. ROBOMECH Journal                                                                                           | 1224 |
| 13(1), 17 (2026) <a href="https://doi.org/10.1186/s40648-026-00345-6">https://doi.org/10.1186/s40648-026-00345-6</a>                                              | 1225 |
|                                                                                                                                                                   | 1226 |
|                                                                                                                                                                   | 1227 |
|                                                                                                                                                                   | 1228 |
| [29] Anderson, P., Chang, A., Chaplot, D.S., Dosovitskiy, A., Gupta, S., Koltun, V.,                                                                              | 1229 |
| Kosecka, J., Malik, J., Mottaghi, R., Savva, M., Zamir, A.R.: On evaluation of                                                                                    | 1230 |
| embodied navigation agents. arXiv preprint arXiv:1807.06757 (2018) <a href="https://doi.org/10.48550/arXiv.1807.06757">https://doi.</a>                           | 1231 |
|                                                                                                                                                                   | 1232 |
|                                                                                                                                                                   | 1233 |
|                                                                                                                                                                   | 1234 |
|                                                                                                                                                                   | 1235 |
| [30] Bonsignorio, F., del Pobil, A.P.: Toward replicable and measurable robotics                                                                                  | 1236 |
| research. IEEE Robotics & Automation Magazine 22(3), 32–35 (2015) <a href="https://doi.org/10.1109/MRA.2015.2452073">https://doi.org/10.1109/MRA.2015.2452073</a> | 1237 |
|                                                                                                                                                                   | 1238 |
|                                                                                                                                                                   | 1239 |
|                                                                                                                                                                   | 1240 |
|                                                                                                                                                                   | 1241 |
|                                                                                                                                                                   | 1242 |

- 1243 [31] Lakens, D., Mesquida, C., Rasti, S., Ditroilo, M.: The benefits of preregistration  
1244 and registered reports. *Evidence-Based Toxicology* **2**(1), 2376046 (2024) <https://doi.org/10.1080/2833373x.2024.2376046>  
1245  
1246  
1247  
1248  
1249 [32] Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., Savva,  
1250 M., Chernova, S., Batra, D.: Sim2Real predictivity: Does evaluation in simulation  
1251 predict real-world performance? *IEEE Robotics and Automation Letters* **5**(4),  
1252 6670–6677 (2020) <https://doi.org/10.1109/lra.2020.3013848>  
1253  
1254  
1255  
1256 [33] Gervet, T., Chintala, S., Batra, D., Malik, J., Chaplot, D.S.: Navigating to  
1257 objects in the real world. *Science Robotics* **8**(79) (2023) [https://doi.org/10.1126/](https://doi.org/10.1126/scirobotics.adf6991)  
1258 [scirobotics.adf6991](https://doi.org/10.1126/scirobotics.adf6991) . Article eadf6991  
1259  
1260  
1261  
1262 [34] Hoshi, M., Hara, Y., Nakamura, S.: Graph-based SLAM using wall detection  
1263 and floor plan constraints without loop closure. *ROBOMECH Journal* **11**(1), 18  
1264 (2024) <https://doi.org/10.1186/s40648-024-00285-z>  
1265  
1266  
1267  
1268 [35] Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object  
1269 detection with region proposal networks. *IEEE Transactions on Pattern Analy-*  
1270 *sis and Machine Intelligence* **39**(6), 1137–1149 (2017) [https://doi.org/10.1109/](https://doi.org/10.1109/tpami.2016.2577031)  
1271 [tpami.2016.2577031](https://doi.org/10.1109/tpami.2016.2577031)  
1272  
1273  
1274  
1275 [36] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recogni-  
1276 tion. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern*  
1277 *Recognition (CVPR)* (2016). <https://doi.org/10.1109/cvpr.2016.90>  
1278  
1279  
1280  
1281 [37] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature  
1282 pyramid networks for object detection. In: *Proceedings of the IEEE Conference*  
1283 *on Computer Vision and Pattern Recognition (CVPR)* (2017). [https://doi.org/](https://doi.org/10.1109/CVPR.2017.106)  
1284 [10.1109/CVPR.2017.106](https://doi.org/10.1109/CVPR.2017.106)  
1285  
1286  
1287  
1288

- [38] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T.,  
Andreetto, M., Adam, H.: MobileNets: Efficient convolutional neural networks  
for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) <https://doi.org/10.48550/arXiv.1704.04861>
- [39] Szot, A., *et al.*: Habitat 2.0: Training home assistants to rearrange their habitat.  
In: Advances in Neural Information Processing Systems (NeurIPS) (2021). <https://doi.org/10.48550/arXiv.2106.14405>
- [40] Calli, B., Singh, A., Walsman, A., Srinivasa, S., Abbeel, P., Dollar, A.M.: The  
YCB object and model set: Towards common benchmarks for manipulation  
research. In: Proceedings of the International Conference on Advanced Robotics  
(ICAR) (2015). <https://doi.org/10.1109/icar.2015.7251504>
- [41] Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neu-  
ral networks. In: Proceedings of the 34th International Conference on Machine  
Learning (ICML) (2017). <https://doi.org/10.48550/arXiv.1706.04599>
- [42] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Interna-  
tional Conference on Learning Representations (ICLR) (2015). <https://doi.org/10.48550/arXiv.1412.6980>
- [43] Friedman, M.: The use of ranks to avoid the assumption of normality implicit in  
the analysis of variance. Journal of the American Statistical Association **32**(200),  
675–701 (1937) <https://doi.org/10.1080/01621459.1937.10503522>
- [44] Cochran, W.G.: The comparison of percentages in matched samples. Biometrika  
**37**(3/4), 256–266 (1950) <https://doi.org/10.2307/2332378>
- [45] Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin  
**1**(6), 80–83 (1945) <https://doi.org/10.2307/3001968>

1335 [46] Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian  
1336 Journal of Statistics **6**(2), 65–70 (1979)  
1337  
1338  
1339 [47] Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall,  
1340 New York (1993). <https://doi.org/10.1201/9780429246593>  
1341  
1342  
1343 [48] Cliff, N.: Dominance statistics: Ordinal analyses to answer ordinal questions. Psy-  
1344 chological Bulletin **114**(3), 494–509 (1993) [https://doi.org/10.1037/0033-2909.](https://doi.org/10.1037/0033-2909.114.3.494)  
1345  
1346 [114.3.494](https://doi.org/10.1037/0033-2909.114.3.494)  
1347  
1348  
1349 [49] Kendall, M.G., Babington Smith, B.: The problem of  $m$  rankings. The Annals  
1350 of Mathematical Statistics **10**(3), 275–287 (1939) [https://doi.org/10.1214/aoms/](https://doi.org/10.1214/aoms/1177732186)  
1351  
1352 [1177732186](https://doi.org/10.1214/aoms/1177732186)  
1353  
1354  
1355 [50] Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., Liu, Y., Chen, J.: DETRs  
1356 beat YOLOs on real-time object detection. In: 2024 IEEE/CVF Conference on  
1357 Computer Vision and Pattern Recognition (CVPR), pp. 16965–16974 (2024).  
1358  
1359 <https://doi.org/10.1109/cvpr52733.2024.01605>  
1360  
1361  
1362 [51] Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., Ding, G.: YOLOv10: Real-  
1363 time end-to-end object detection. In: Advances in Neural Information Processing  
1364 Systems 37, pp. 107984–108011 (2024). <https://doi.org/10.52202/079017-3429>  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380