

## 441 A Background: Conformal Prediction

442 We provide a brief background on conformal prediction (CP) [5, 46], which will be our primary  
 443 tool for performing rigorous uncertainty quantification for perception. Suppose we have  $N$  i.i.d. (or  
 444 exchangeable) samples  $U_1, \dots, U_N$  of a scalar random variable  $U$ . We can compute the threshold,  
 445  $\hat{q}_{1-\epsilon}$ , such that the next sample,  $U_{\text{test}}$ , has the following guarantee:

$$\mathbb{P}[U_{\text{test}} \leq \hat{q}_{1-\epsilon}] \geq 1 - \epsilon, \quad \hat{q}_{1-\epsilon} = \begin{cases} U_{(\lceil (N+1)(1-\epsilon) \rceil)} & \text{if } \lceil (N+1)(1-\epsilon) \rceil \leq N, \\ \infty & \text{otherwise,} \end{cases} \quad (6)$$

446 where  $U_{(1)} \leq U_{(2)} \leq \dots \leq U_{(N)}$  are the order statistics (sorted values) of the  $N$  samples  
 447  $U_1, \dots, U_N$ . In the CP literature,  $U$  is known as the non-conformity score and it is a measure  
 448 of the (in)correctness of a model. The above guarantee (6) is *marginal*, i.e., (6) holds over the sam-  
 449 pling of both the calibration dataset  $U_1, \dots, U_N$  and the test variable  $U_{\text{test}}$ . Hence, we will need to  
 450 generate a fresh set of i.i.d. calibration data  $\bar{U}_1, \dots, \bar{U}_N$  for the guarantee to hold for a new sample  
 451  $\bar{U}_{\text{test}}$ . However, in practice, one typically only has access to a single dataset of examples; inferences  
 452 from this dataset must be used for all future predictions on test examples.

453 In this work, we consider the dataset-conditional guarantee [37] that does not require us to generate  
 454  $N$  new samples for every new test prediction. The following bound holds with probability  $1 - \delta$   
 455 over the sampling of the calibration dataset:

$$\mathbb{P}[U_{\text{test}} \leq \hat{q}_{1-\epsilon}|U_1, \dots, U_N] \geq \text{Beta}_{N+1-v,v}^{-1}(\delta), \quad v := \lfloor (N+1)\hat{\epsilon} \rfloor, \quad (7)$$

456 where,  $\text{Beta}_{N+1-v,v}^{-1}(\delta)$  is the  $\delta$ -quantile of the Beta distribution with parameters  $N+1-v$  and  $v$ ,  
 457 and we can choose  $\hat{\epsilon}$  to achieve the desired  $1 - \epsilon$  coverage.

## 458 B Proof of Proposition 1

459 As seen in Appendix A, conformal prediction gives us the following *dataset-conditional* guarantee  
 460 on a new sample of the nonconformity score  $U_{\text{test}}$  corresponding to a test environment  $E_{\text{test}}$ . With  
 461 probability  $1 - \delta$  over the sampling of  $U_1, \dots, U_N$ ,

$$\mathbb{P}[U_{\text{test}} \leq \hat{q}_{1-\epsilon}|U_1, \dots, U_N] \geq \text{Beta}_{N+1-v,v}^{-1}(\delta).$$

462 We can rewrite the event  $U_{\text{test}} \leq \hat{q}_{1-\epsilon}$  as:

$$\begin{aligned} & \{U_{\text{test}} \leq \hat{q}_{1-\epsilon}\} \\ &= \left\{ \hat{q}_{1-\epsilon} \geq \min_{q_{\text{test}}} q_{\text{test}} | A_{\text{test}} \subseteq B_{s,\text{test}} + \Delta_{q_{\text{test}}}, \forall s \in \mathcal{S} \right\} \\ &= \left\{ A_{\text{test}} \subseteq B_{s,\text{test}} + \Delta_{\hat{q}_{1-\epsilon}}, \forall s \in \mathcal{S} \right\} \\ &= \{A_{\text{test}} \subseteq \bar{B}_{s,\text{test}}, \forall s \in \mathcal{S}\} \\ &= \left\{ \bar{C}_{E_{\text{test}}}(\bar{\phi}) = 0 \right\}, \end{aligned}$$

463 which gives us the desired result (4).

## 464 C Implementation with a limited field-of-view

465 A natural question that arises after following the calibration procedure described above is: what  
 466 happens if the robot is not able to observe all objects in the environment from all states? This may  
 467 happen due to a limited sensing capability or because some parts of the environment are occluded  
 468 from view. We address this issue in our calibration procedure implementation by only taking into  
 469 account perception errors for objects that are within the field-of-view of the robot in a given state,  
 470 and masking any ground-truth bounding boxes that are not visible to the robot, i.e.,  $A$  (which now  
 471 depends on state  $s$ ) is the union of all the ground-truth bounding boxes of the *visible* objects. Hence,  
 472 the perception system correctness assurance stated above holds for all objects within the field-of-  
 473 view of the robot at any given state. The presence of possibly occluded obstacles are dealt with by a  
 474 safe planner, which we describe next.

## D Planner implementation details

For our simulation and hardware experiments, we use the safe planner proposed in [16] due to its approximate optimality and ease of implementation. The safety filter in this case is an inevitable collision set (ICS) constraint [17], where the robot is forbidden to enter any state that will eventually result in collision no matter what control actions are taken. Within the known free space  $\bar{\mathcal{X}}_t^{\text{free}}$ , the robot plans using the fast marching tree algorithm (FMT\*) [47] with dynamics [48]. If the goal is not visible within  $\bar{\mathcal{X}}_t^{\text{free}}$ , the robot plans to an intermediate goal on the boundary of its free space. The intermediate goals are chosen based on the cost-to-come from current robot state to the intermediate goal, and the distance-to-go from the intermediate goal to the actual goal. The robot replans whenever it receives a sensor update and an updated  $\bar{\mathcal{X}}_{t+1}^{\text{free}}$  from its non-deterministic filter, and accounts for ICS constraints [49] in-between sensor updates.

## E Proof of Proposition 2:

As shown in Proposition 1, the misdetection rate of the calibrated perception system  $\bar{\phi}$  is  $\epsilon$ -bounded on environments drawn from  $\mathcal{D}$  at each time step  $t$ , where the robot is at state  $s_t$ . In other words, the predicted occupied space  $\hat{\mathcal{X}}_t^{\text{occ}}$  at each time step contains the true obstacles  $A$  with high probability across environments. Conversely, the predicted free space  $\hat{\mathcal{X}}_t^{\text{free}}$  at each time step does not contain the true obstacles  $A$  with high probability across environments. If we consider a safety-relevant misdetection cost at time step  $t$ :

$$\hat{C}_E^{\text{safe}}(\bar{\phi}, s_t) = \begin{cases} 1 & \text{if } A \subseteq \hat{\mathcal{X}}_t^{\text{free}} \text{ (unsafe),} \\ 0 & \text{otherwise,} \end{cases} \quad (8)$$

then the misdetection rate over the set of states should be  $\epsilon$ -bounded across environments by Proposition 1:

$$\mathbb{E}_{E \sim \mathcal{D}_E} \max_{t \in [0, T]} \hat{C}_E^{\text{safe}}(\bar{\phi}, s_t) \leq \epsilon. \quad (9)$$

Because the expectation in Equation (9) is over the set of environments, the following statement holds in any new environment (with probability  $1 - \delta$  over the calibration dataset of environments),

$$\mathbb{Pr} \left\{ \max_{t \in [0, T]} \hat{C}_E^{\text{safe}}(\bar{\phi}, s_t) = 0 \right\} \geq 1 - \epsilon. \quad (10)$$

Given  $m_t = \{\bar{\mathcal{X}}^{\text{free}}, \bar{\mathcal{X}}^{\text{occ}}, \bar{\mathcal{X}}^{\text{unknown}}\}$ , a safe planner never drives the robot outside of the free space. Therefore, the safe planner guarantees  $C_E^{\text{safe}}(\pi^{\bar{\phi}}) \leq \hat{C}_E^{\text{safe}}(\bar{\phi})$ .

$$\mathbb{Pr} \left\{ C_E^{\text{safe}}(\pi^{\bar{\phi}}) = 0 \right\} \geq 1 - \epsilon. \quad (11)$$

## F Extensions

In this section, we outline a few extensions to the basic technical approach described in Sections 3 and 4: (i) fine-tuning a pre-trained perception model, (ii) incorporating sensor and dynamics uncertainty, and (iii) calibrating perception modules beyond bounding box prediction.

### F.1 Fine-Tuning a Pre-Trained Perception Model

In Section 3, we assumed access to a pre-trained perception model  $\phi$  that outputs bounding boxes. The conformal prediction-based uncertainty quantification procedure then uses the calibration dataset  $D = \{E_1, \dots, E_N\}$  of environments to produce a calibrated perception system  $\bar{\phi}$  which lightly processes the outputs of  $\phi$  by inflating the predicted bounding boxes. In practice, it may also be useful to *fine-tune*  $\phi$  for our target deployment environments before performing uncertainty quantification.

This can be achieved using *split conformal prediction* [38], where one splits the overall dataset  $D$  into  $D = D_{\text{tune}} \cup D_{\text{cal}}$ . If the perception model takes the form of a neural network  $\phi_w$  parameterized

by weights  $w$ , we can use  $D_{\text{tune}}$  to fine-tune  $w$  (or the weights of a residual network). We can then utilize  $D_{\text{cal}}$  in order to perform the CP-based calibration as described in Section 3. As we demonstrate in Section 5, this additional fine-tuning step before calibration can reduce the conservatism of outputs and improve end-to-end success rates.

The typical choice of loss function for training a bounding box predictor is the *generalized intersection-over-union (gIoU) loss* [50]. This is a differentiable version of the IoU loss: given a ground-truth bounding box  $A$  and a predicted box  $B$ , one computes  $L(A, B) := |A \cap B|/|A \cup B|$ . However, while this loss is popular in computer vision, it is not suitable for robot navigation. In particular, the IoU loss is *symmetric*: it does not distinguish between the ground-truth and predicted bounding box and thus does not encourage the predicted box to *contain* the ground-truth box. We propose a modification to the gIoU loss in Appendix F.1.1, which encourages that the predicted bounding box encloses the ground-truth box while also ensuring that the predicted box is not too large. Similar to the gIoU loss, this loss is (almost-everywhere) differentiable and scale invariant. We utilize this loss for fine-tuning in our experiments (Section 5). However, one could use any other method for finetuning not limited to training a simple neural network with gIoU loss [51].

### F.1.1 Loss Function for Fine-Tuning

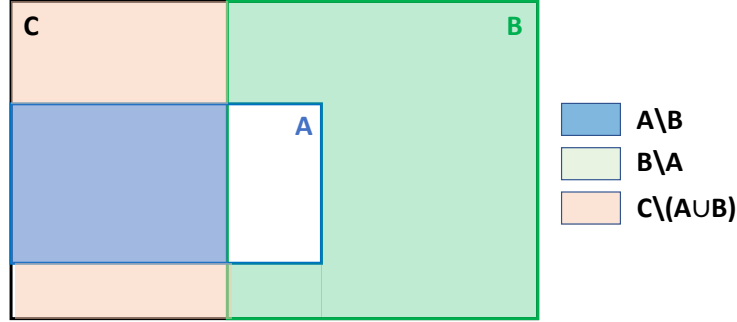


Figure 8: Visualization of different terms in the loss function for a single object setting.

We use a (almost-everywhere) differentiable loss function for training. The loss function seeks to ensure that the predicted shape (e.g., bounding box) encloses the ground truth shape while also ensuring that the predicted shape is not too large.

Let’s consider the simplest setting wherein we have one object in the scene and we are making a single prediction. In this case,  $A$  denotes the (convex) ground-truth shape and  $B$  denotes the (convex) predicted shape. Let  $C$  denote the convex hull of  $A$  and  $B$ . Our loss function is a weighted combination of three terms,

$$L := w_1 l_1 + w_2 l_2 + w_3 l_3 = w_1 \frac{|A \setminus B|}{|A|} + w_2 \frac{|B \setminus A|}{|B|} + w_3 \frac{|C \setminus (A \cup B)|}{|C|}.$$

The first term is the most important; it tries to ensure that  $B$  encloses  $A$ . The second term tries to make sure that  $B$  is not much larger than it needs to be, see Figure 8. The first and second terms are sufficient if  $A$  and  $B$  are overlapping. However, if they do not overlap, there is no gradient information provided by the first two terms. Following [50], we introduce a third loss term in order to provide gradient information when the shapes do not intersect. The loss terms  $l_1, l_2, l_3$  are each bounded within  $[0, 1]$ . Hence, if we choose  $w_1, w_2, w_3$  such that  $\sum_i w_i = 1$ , then the overall loss is also bounded within  $[0, 1]$ . Now let’s consider the setting wherein,  $A$  denotes the union of multiple ground-truth bounding boxes (say we have  $m$  objects in the scene) and  $B$  is the union of all the predicted bounding boxes (we predict  $n$  boxes). We consider all the individual bounding box predictions  $B_i, \forall i \in \{1, \dots, n\}$  and associate the closest *visible* ground-truth bounding box  $A_i$  to each prediction. Now we can define  $C_i$  as the convex hull of  $A_i$  and  $B_i$  and the resulting loss

function,  $L_i$ ,

$$L_i := w_1 \frac{|A_i \setminus B_i|}{|A_i|} + w_2 \frac{|B_i \setminus A_i|}{|B_i|} + w_3 \frac{|C_i \setminus (A_i \cup B_i)|}{|C_i|}.$$

Hence, the overall loss is,

$$L = \frac{1}{n} \sum_{i=1}^n L_i.$$

Please refer to [50, Appendix 4.3] for instructions on how to compute the loss analytically for axis-aligned bounding boxes.

### F.1.2 Simulation Results - Effect of finetuning dataset size

Upon collecting a calibration dataset of  $\sim 400$  environments, as described in the experiment setup in Section 5, we may choose to use a smaller subset of the calibration dataset to further finetune the pre-trained perception model to perform better in the types of environments we are interested in deploying the robot in. We consider the effect of different dataset split sizes for finetuning and then calibration. Using a larger set of environments for finetuning  $|D_{\text{tune}}|$  may result in a better tuned model, but will leave fewer environments for calibration,  $|D_{\text{cal}}|$ , resulting in a more conservative  $\hat{e}$  and  $\hat{q}_{1-\epsilon}$  that satisfies the dataset-conditional guarantee (7), and vice versa. This trade-off is seen in Table 1, where we observe the best performance when we have an equal split between finetuning and calibration.

	Split size ( $ D_{\text{tune}}  +  D_{\text{cal}} $ )	$\hat{q}_{0.85}$ (in m)	Collision	Misdetection	Goal Reached
562	100 + 300	0.68	0%	1%	89%
	200 + 200	<b>0.64</b>	0%	<b>1%</b>	<b>94%</b>
	300 + 100	0.93	0%	2%	76%

Table 1: A comparison of the effect of various partition sizes for finetuning and calibration for PWC.

### F.2 Sensor Errors and Dynamics Uncertainty

In Section 2, we modeled the robot’s sensor as a deterministic mapping  $\sigma : \mathcal{S} \times \mathcal{E} \rightarrow \mathcal{O}$ , which provides observations from a particular state in a given environment. This formulation allows us to also incorporate sensor errors. Specifically, any errors or randomness in the sensor can be formally included as part of the environment  $E \in \mathcal{E}$ . Thus, in addition to sampling environmental variables such as obstacle locations, geometries, etc., each environment  $E$  also samples random variables that prescribe sensor errors from each state  $s \in \mathcal{S}$  in the environment. This way of modeling sensor errors allows  $\sigma$  to be deterministic (since all sources of randomness are included in  $E$ ), allows the sensor errors to be dependent on the relative pose of the robot relative to obstacles (e.g., modeling the fact that depth estimates are often further from ground-truth depth values as distance increases), and also allows us to model correlations in sensor errors from different locations (e.g., capturing the fact that sensor errors from nearby robot locations can be highly correlated). Modeling time-varying sensor errors (i.e., different sensor errors from the robot state at different times) is not as immediate, but could potentially be incorporated by augmenting the state space  $\mathcal{S}$  to include the time-step.

In addition to errors in sensing, one can also account for uncertainty in the dynamics of the robot by using a robust planner (see [15] for an overview). In the experiments described in Section 6, we incorporate uncertainty by generating plans that prevent the robot from entering the inevitable collision set (cf. Section 4) even with bounded uncertainty in the dynamics.

### F.3 Calibration with General Occupancy Prediction Models

Section 3 introduced the CP-based calibration procedure in the context of bounding box prediction. However, the theoretical formulation in Section 3 is applicable to more general occupancy prediction models; the key requirement is the presence of a scalar quantity that monotonically grows the size of the predicted occupied space (e.g., the inflation parameter  $q$  for bounding boxes in Section 3). This allows one to define the non-conformity score  $U_i$  for an environment  $E_i$  as in (2) to be the smallest

scalar such that the inflated predicted occupied space contains the ground-truth obstacles (for all robot locations). Hence, we can calibrate the outputs of any perception system that predicts an occupied set or performs *occupancy prediction* more generally, i.e., assigns a (heuristic) occupancy confidence to each point in the space. Possibilities for the latter include scene completion networks [45] or deep signed-distance function representations [52]. A threshold on this confidence acts as the scalar parameter that monotonically controls the size of the predicted occupied space. The conformal prediction procedure from Section 3 can then be used to find a confidence threshold such that predicted occupied space contains the true occupied space (with probability  $1 - \epsilon$  in a new environment).

## G Calibration and planning

We collect a calibration dataset of 400 environments wherein we randomly place 1 – 5 chairs from the diverse 3D-Front dataset [10] in a 8 m  $\times$  8 m room. In this 8 m  $\times$  8 m space, we use a fixed set of 2000 sampled configurations for the sampling-based motion planner and use the same set of samples for the calibration procedure. We construct the calibration dataset in simulation using CAD models of *real* furniture pieces from the 3D-Front dataset [10], which contains a highly diverse array of industrial CAD models developed by professional designers to ensure that the performance of the perception system remains the same in its simulation and hardware implementation. Similarly, we collect an additional fine-tuning dataset  $D_{\text{tune}}$  consisting of 100 environments. These environments include ones with occlusions of the goal and objects in the scene.

### G.1 Metrics for experiments

We simulate the dynamics of the Unitree Go1 quadruped robot and task the robot with navigating to a goal location that is  $\sim 7$  m away from the initial location of the robot. The robot camera has a field of view of  $70^\circ$  and a visibility range of  $[1, 5]$  m. With an allowable misdetection rate of  $\epsilon = 0.15$ , we obtain  $\hat{q}_{0.85} = 0.75$  m for PwC,  $\hat{q}_{0.85} = 0.65$  m for PwC-fine-tuned, and  $\hat{q}_{0.85} = 0.05$  m for CP-avg. through calibration. The planner replans and obtains a new sensor observation to update the filter every 0.5 s or less (if the previous plan is already completed).

We utilize the following metrics for our simulation experiments: a trial is counted as a collision if the robot collides with an obstacle and we count a misdetection for a trial if the free space predicted by the planner has any intersection with the ground-truth bounding boxes of the obstacles. We say that the goal has been reached in a given trial if the robot is able to navigate to within 1 m around the goal in less than 140 s. We also record the average path length for trials in which the goal is reached.

### G.2 Results: Effects of closed-loop distribution shift on misdetections

To illustrate the effect of closed-loop distribution shifts on misdetections, we used exactly the same setup described above to obtain the simulation results in Figure 4. We changed the planner cost to have a different weighting on the cost-to-go. For one setting, we chose a weight  $w = 1$  on the cost-to-go, which is the same as the weighting on the cost-to-come. In another setting, we chose a weight  $w = 10$  on the cost-to-go, and hence a  $10\times$  more emphasis on the cost-to-go compared to the cost-to-come. Table 2 shows the KL-divergence between the states visited by the planner and the sampling distribution of states for calibration as a measure of the closed-loop distribution shift. Increasing closed-loop shifts lead to higher misdetections. One can see that a simple change in the planner parameters can lead to potentially large changes in the safety rates for CP-avg. The closed-loop shift we may see in practice is unknown apriori. Hence, it is difficult to make any statements on the planner safety in closed-loop despite using CP for calibration of the perception system. PwC, on the other hand, is robust to the closed-loop

Method	Collision	Mis-detection	KL-divergence
CP-avg. ( $w = 1$ )	14%	54%	2.09
CP-avg. ( $w = 10$ )	2%	64%	2.72
PwC ( $w = 1$ )	0%	0%	1.48
PwC ( $w = 10$ )	0%	2%	2.04

Table 2: A comparison of the effect of changing the planner parameters on CP-avg. and PwC.

shifts and can still satisfy the misdetection and safety assurance regardless of the planner parameters used.

## H Hardware

### H.1 Hardware and Environmental Setup

We represent the robot’s state as  $s_t = [x, y, v_x, v_y]^T$  where  $x$  and  $y$  are its position in the environment and  $v_x$  and  $v_y$  are the respective velocities. For each trial, the robot is initialized around position  $[4, 0]$ m (with the origin set to bottom left corner of the room) and has a time horizon of 60 seconds to reach the goal within a 1m radius. The robot replans every 1s in a receding horizon manner using the safe planner described in Section 4. The goals are varied every 10 environments and include positions  $[2, 7]$ m,  $[7, 0]$ m, and  $[6, 7]$ m.

**Hardware.** We use the Unitree Go1 quadruped robot with fully onboard sensing and computation. The robot is equipped with a ZED 2i RGB-D camera and a ZED Box computer attached to the base of the robot as shown in the top row of Figure 7. The Zed 2i provides the Go1 with point cloud observations with a  $70^\circ$  field of view and a visibility range of  $[1, 5]$ m. The Zed 2i also uses vision-inertial odometry to provide accurate positional state estimates in the environment. The Zed Box includes an 8-core ARM processor and a 16GB Orin NX GPU. This allows us to process the point cloud observations in order to produce bounding boxes using the pre-trained 3DETR model [19]. The bounding boxes are aggregated over time to update the estimated free, occupied, and unknown spaces as described in Section 4. The safe planner described in Section 4 is used to output Cartesian velocity commands bounded at a speed of 0.8m/s; these commands are sent from the Zed Box over UDP to the Go1’s processor. The average planning time on the ZED Box across trials is approximately 0.5s. The dynamics of the Go1 are estimated using MATLAB’s System Identification Toolbox [53] and are provided in Appendix H.2.

**Environments.** We test the robot in 30 different environments, consisting of various chair configurations and geometries in an 8 m  $\times$  8 m room. Configurations range from random, occluded goal, occluded chairs, clustered chairs, and narrow paths (approximately 1.8m in width leaving 0.4m of available freespace for PWC to find). Each environment has between 4 and 8 chairs present. See Appendix H.3 and H.4 for the unseen chairs used in testing and the environment configurations respectively. We use a Vicon motion capture system to log the ground-truth placement and bounding boxes of the chairs for each environment.

### H.2 System Identification

To perform system identification of the Unitree Go1 quadruped robot, we collected trajectories using a Vicon motion capture system. We then used MATLAB’s system identification toolbox [53]. Specifically, we provided an initial linear ODE grey box model guess and then used prediction error minimization (PEM) for refinement. The resulting system is shown in (12) where  $x$  and  $y$  describe the positional state of the robot in the environment,  $v_x$  and  $v_y$  describe the respective velocities, and  $u_x$  and  $u_y$  describe the respective commanded velocities.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -2.5170 & 0.1353 \\ 0 & 0 & -0.5197 & -3.9680 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 2.3350 & 0 \\ 0 & 4.6510 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} \quad (12)$$

### H.3 Chair Test Dataset

Our test dataset of chairs for the experiments conducted in Section 6 included 8 chairs with diverse sizes and geometries unseen in training and calibration for the perception system. Test chairs are shown below in Figure 9.

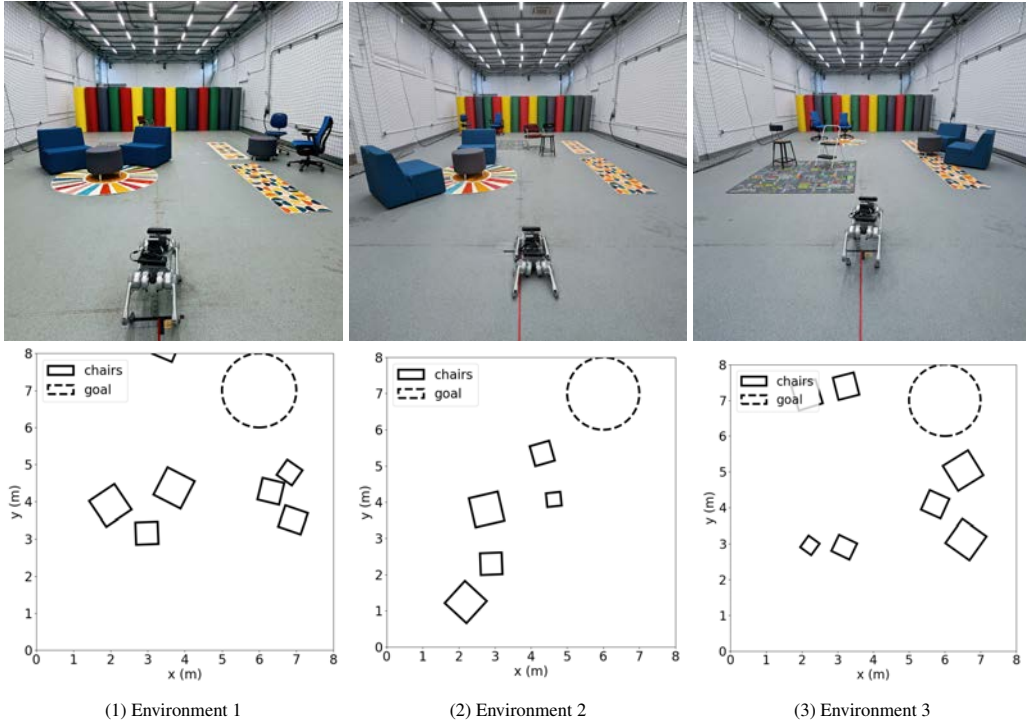


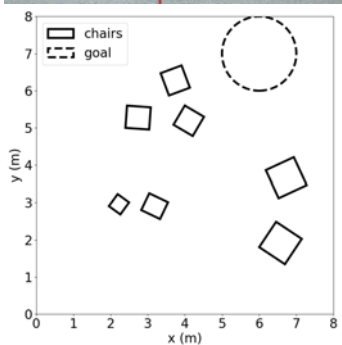
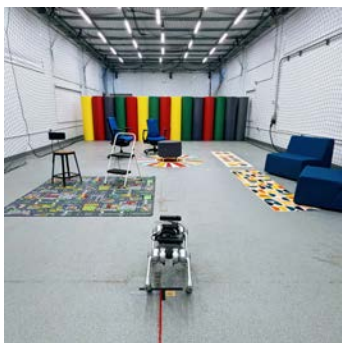


Figure 9: New, unseen test chairs used in hardware experiments.

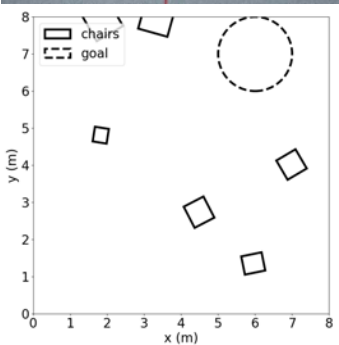
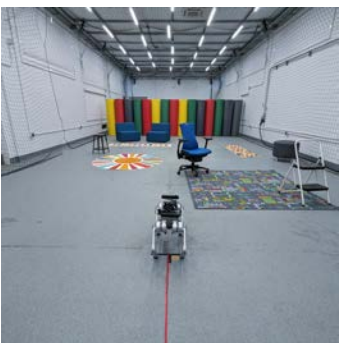
#### 676 H.4 Environments

677 As described in Section 6, the robot was tested in 30 unique environments with varying furniture  
 678 configurations and goals. The following 30 figures show an image of each configuration, accompa-  
 679 nished by a bird's-eye map of the obstacle and goal locations.

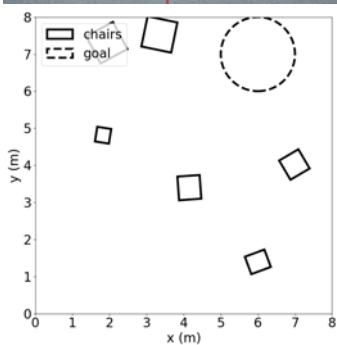
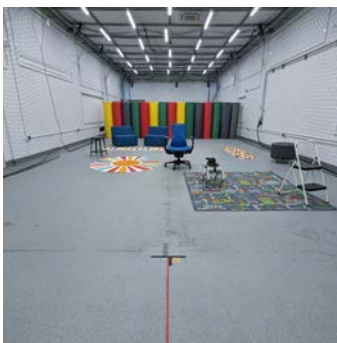




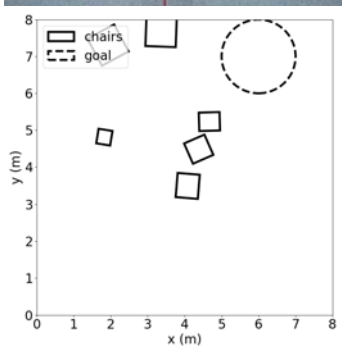
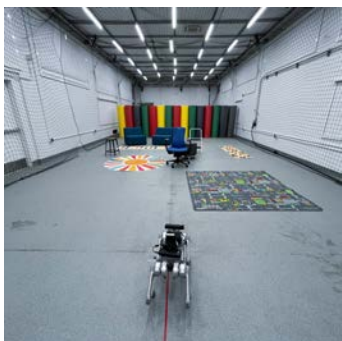
(4) Environment 4



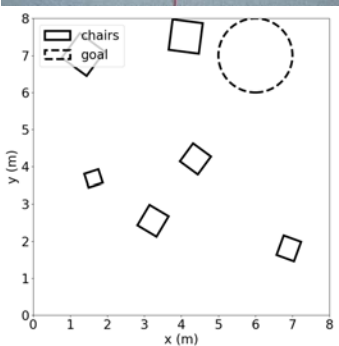
(5) Environment 5



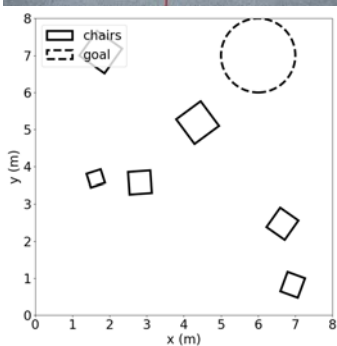
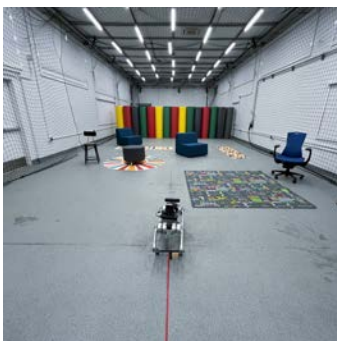
(6) Environment 6



(7) Environment 7

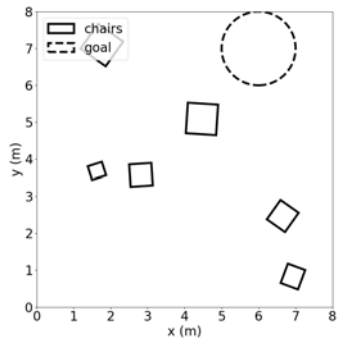
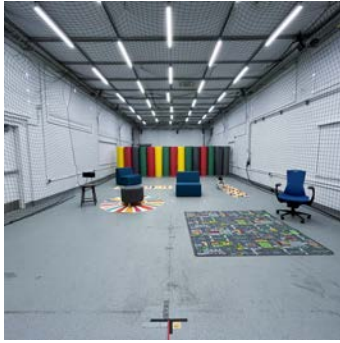


(8) Environment 8

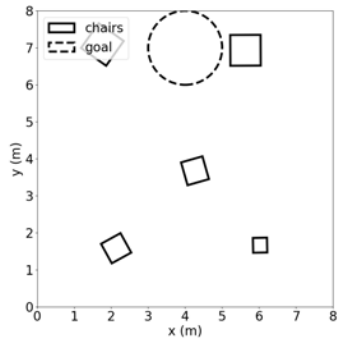


(9) Environment 9

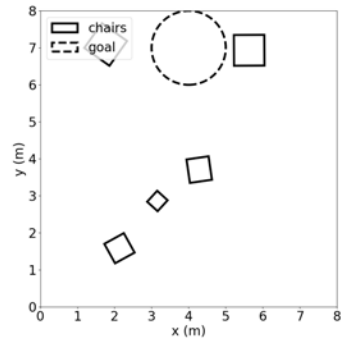
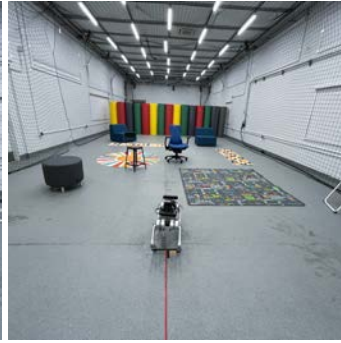




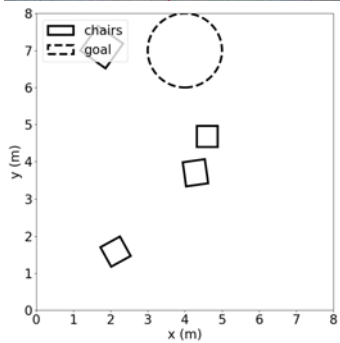
(10) Environment 10



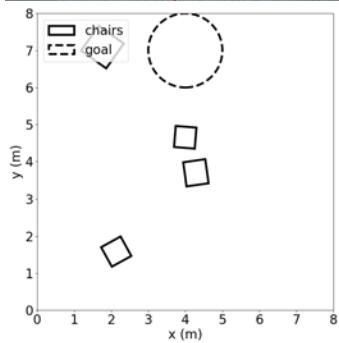
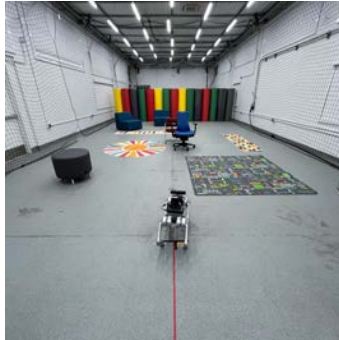
(11) Environment 11



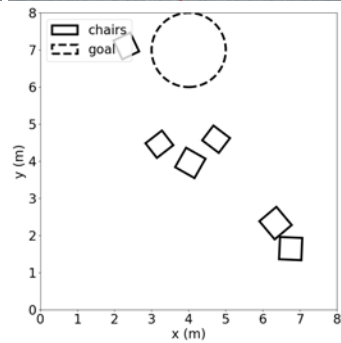
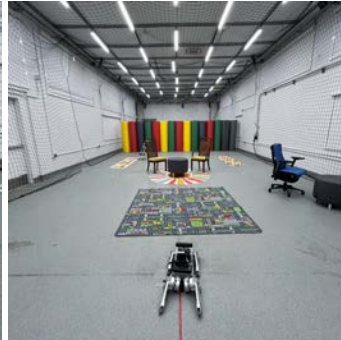
(12) Environment 12



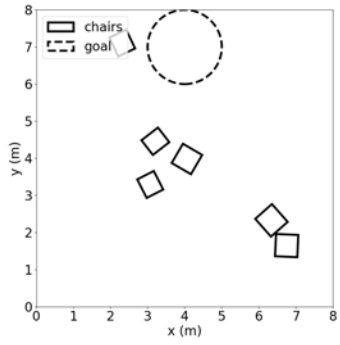
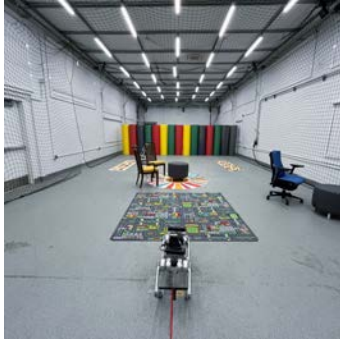
(13) Environment 13



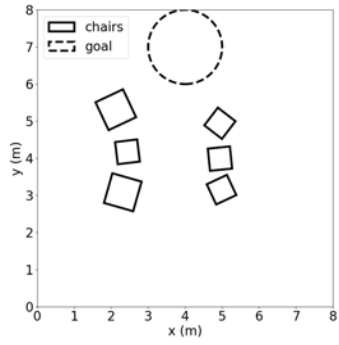
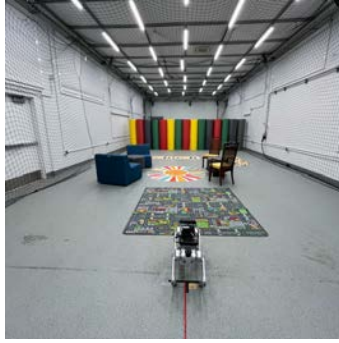
(14) Environment 14



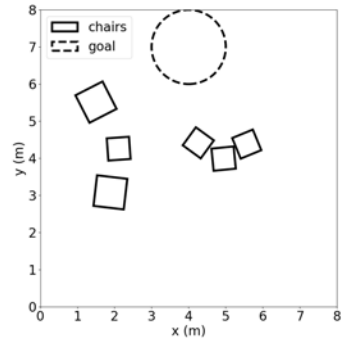
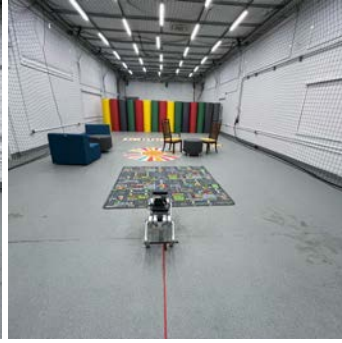
(15) Environment 15



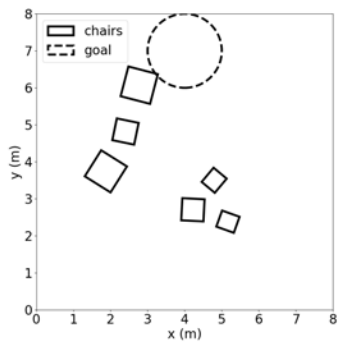
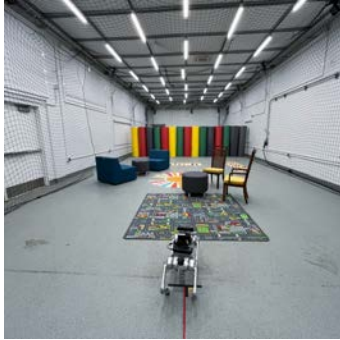
(16) Environment 16



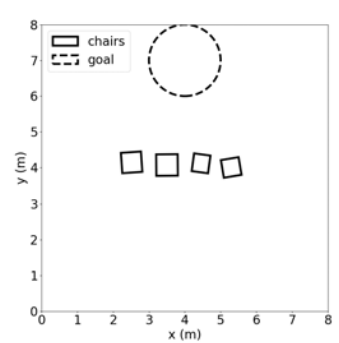
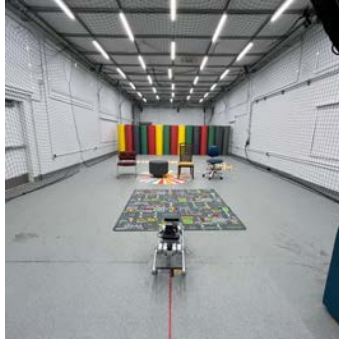
(17) Environment 17



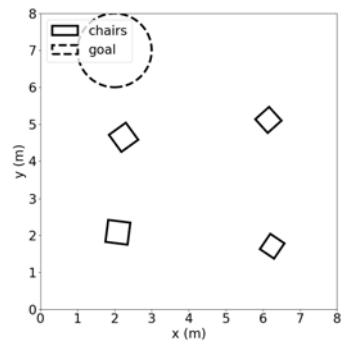
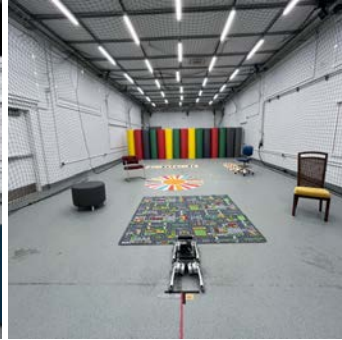
(18) Environment 18



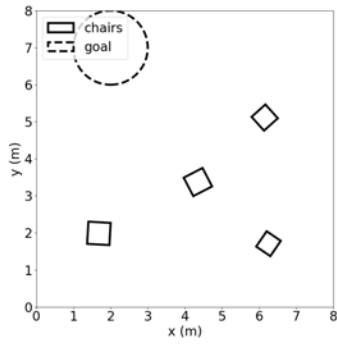
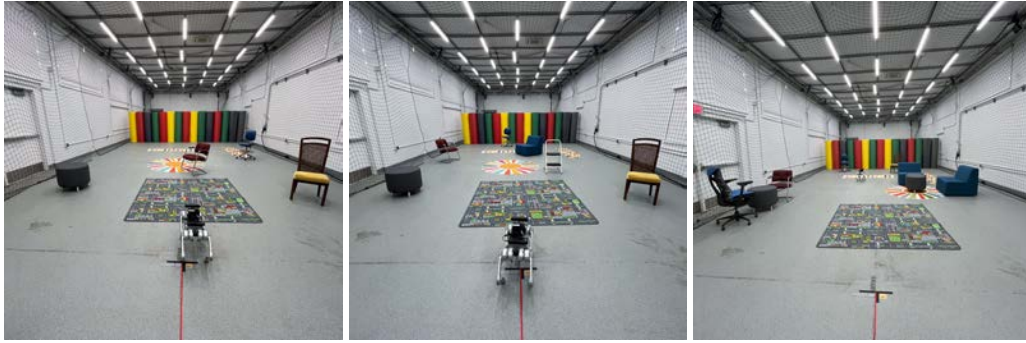
(19) Environment 19



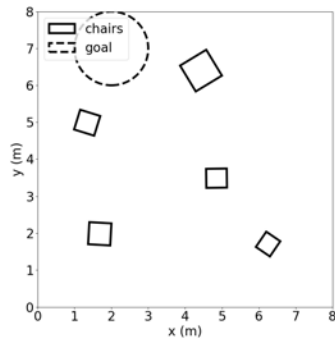
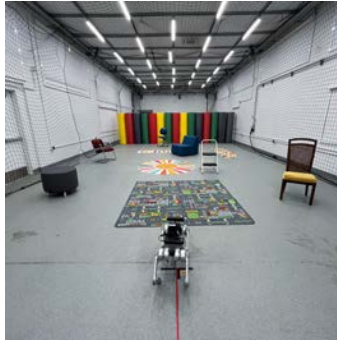
(20) Environment 20



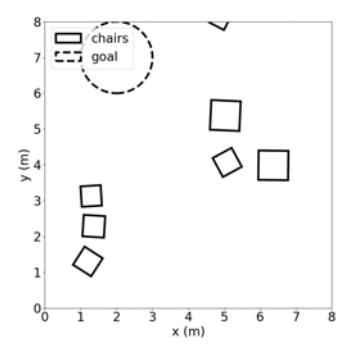
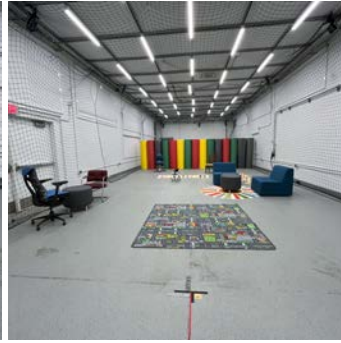
(21) Environment 21



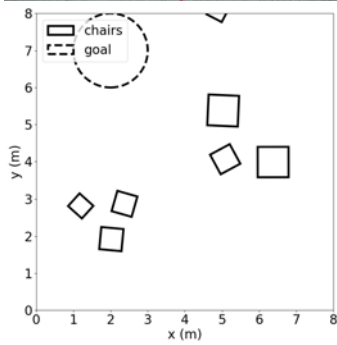
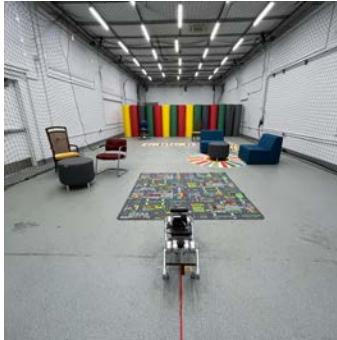
(22) Environment 22



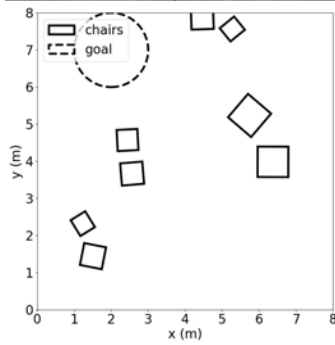
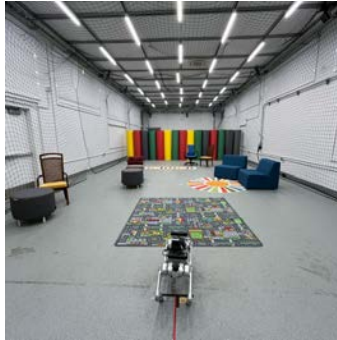
(23) Environment 23



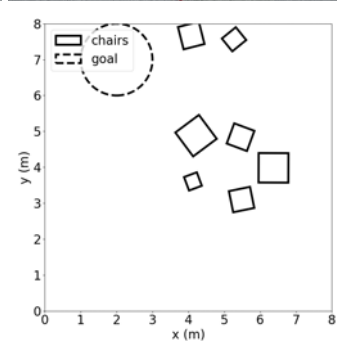
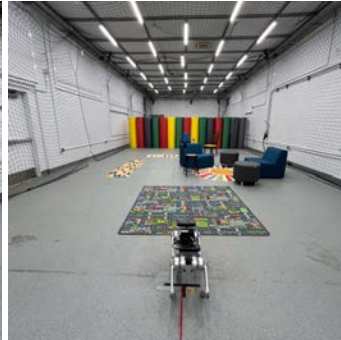
(24) Environment 24



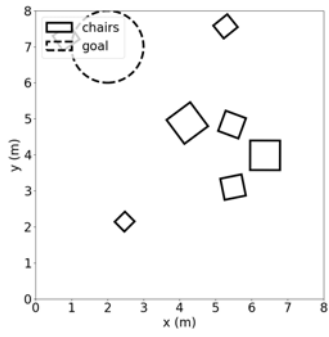
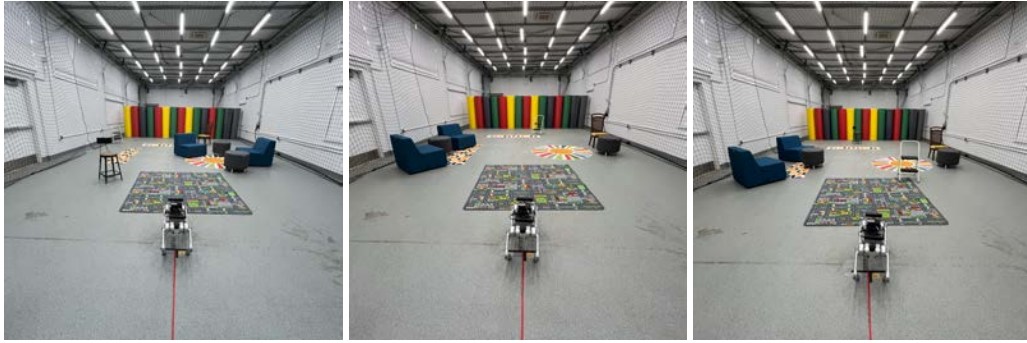
(25) Environment 25



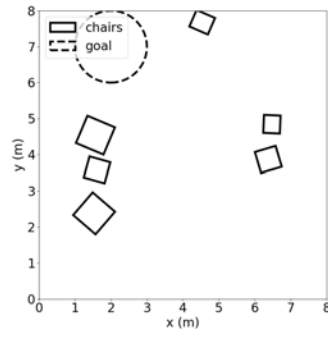
(26) Environment 26



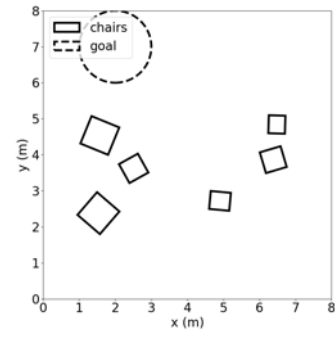
(27) Environment 27



(28) Environment 28



(29) Environment 29



(30) Environment 30