Collision Replay: What Does Bumping Into Things Tell You About Scene Geometry?

Alexander Raistrick
alexrais@umich.edu
Nilesh Kulkarni
nileshk@umich.edu
David F. Fouhey
fouhey@umich.edu

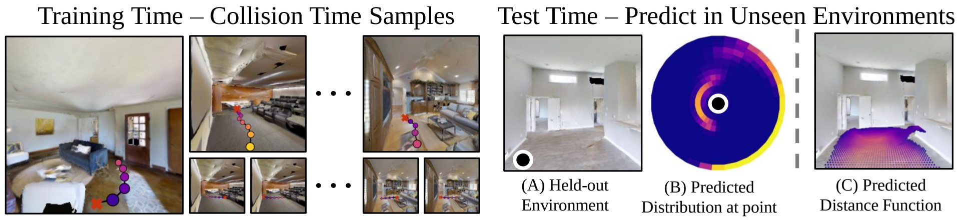
University of Michigan
Ann Arbor, MI

Abstract

What does bumping into things in past scenes tell you about scene geometry in a new scene? In this paper, we investigate the idea of learning from collisions. At the heart of our approach is the idea of collision replay, where after a collision an agent associates the pre-collision observations (such as images or sound collected by the agent) with the time until the next collision. These samples enable training a deep network that can map the pre-collision observations to information about scene geometry. Specifically, we use collision replay to train a model to predict a distribution over collision time from new observations by using supervision from bumps. We learn this distribution conditioned on visual data or echolocation responses. This distribution conveys information about the navigational affordances (e.g., corridors vs open spaces) and, as we show, can be converted into the distance function for the scene geometry. We analyze our approach with a noisily actuated agent in a photorealistic simulator.

1 Introduction

Suppose you bump into something. What does the collision reveal? You know your current position is on the edge of occupied space, but it is not clear what is known about other locations. If you consider where you were k steps earlier, and replay this observation, there must be a k-step path to *some* obstacle, but this does not rule out a shorter path to *another* obstacle. Our goal is to use this replaying of a collision (*Collision Replay*) to convert the local collision to supervision for scene geometry for other modalities like vision or sound. Specifically, we will investigate systems that learn from pairs consisting of a pre-collision observation (an image, or image-like data such as a sound spectrogram) and a corresponding time until collision. These pairs can supervise deep networks that predict time-to-collision information, which can be further converted into information about the geometry of the scene. Our key observation is that one collision in one scene is of limited value, but *multiple collisions* over *multiple training scenes* provide sparse but strong supervision. Once a network is trained to map observations to information about time-to-collision, this information can be used to infer geometric properties in new environments.


Training Time – Collision Time Samples
Test Time – Predict in Unseen Environments

(A) Held-out Environment

(B) Predicted Distribution at point

(C) Predicted Distance Function

Figure 1: **Overall approach. (Left)** At train time, our models learn to predict time-to-collision from samples of collisions across many scenes. We show training trajectories (dots) that end in a collision at a red x. The remaining time to collision is represented as the color and serves as a learning target. **(Right)** (A) At test time, the learned model receives an observation. (B) As output, our model produces a distribution over times to collision at a query point on the floor in a given direction. We show this distribution, conditioned on image features at the query point and heading direction. We plot it as a function of heading direction (radius: distance; angle: angle; color: probability of collision). This distribution can be predicted for each point on the floorplane. (C) One can obtain a distance function on the floor by finding the minimum distance per-point.

We operationalize this idea in Sec. 3 with a method that can predict distributions over time-to-collision from image observations. The key to unlocking this supervision is a modeling of a *distribution* over times to collision (often studied in stochastic processes [10]). The modeling of a distribution enables the model to account for the chance that a training-time collision resulted from a suboptimally short path. We demonstrate our approach on three image-like observations: normal resolution RGB images, low-resolution RGB images, and binaural echo-response spectrograms (which can be treated as an image).

Our embracing of the learning signal of collisions separates our work from the considerable effort usually spent in vision and robotics trying to *avoid* collisions. Approaches range from building detailed geometric maps [15, 32, 34], to using these signals to learn depth estimation models [8, 13, 30, 41], to learning navigation systems [7, 14, 40] to everything in the middle. Our work complements this literature by showing how an agent could self-supervise a range-like sensor via cheap-to-sense collisions. Further, while many of our experiments produce geometric information from images (where there are many other approaches), our approach can be used for general modalities, which we demonstrate with audio spectrograms.

We test our approach in Section 4 using simulated [24] indoor environments [3, 39] with realistic actuation noise. We focus on estimating the scene's distance function on the floor or from an egocentric view. Here, modeling the full distribution outperforms other approaches, and despite sparse supervision, the performance nearly reaches strongly supervised methods. Our experiments furthermore illustrate advantages of collision replay. First, collisions require a buffer of observations and noisy action estimates rather than an explicit scene reconstruction. As Sec. 4.4 shows, this enables collisions to be used as supervision for observations from low resolution images and spectrograms where reconstruction may not be feasible. Second, the distribution provides a good descriptor of the navigational affordances, which Sec. 4.3 shows contains information that distinguishes different parts of rooms.

2 Related Work

Our goal is to take a single *unseen* image and infer the distribution of steps to collision. This distribution is informative about scene shape and local topography, and we show that we can learn this distribution from collisions of a randomly-walking agent.

Estimating occupancy and distance to obstacles has long been a goal of vision and robotics. Collision replay provides a principled method to derive sparse noisy training signal from random walks. Models resulting from collision replay are able to predict these quantities from a single image observation of an unseen environment, which separates it from the large body of work that aims to build models of a specific environment over experience, e.g., via SLAM [2, 9, 15, 34] or bug algorithms [25] that use cheap sensors and loop closure. Instead, this puts our work closer to the literature in inferring the spatial layout [21, 42] of a new scene. In this area, the use of collisions separates it from work that predicts floor plans from strong supervision like ground-truth floor plans [18, 29, 33] or RGB-D cameras [4, 14]. This lightweight supervision puts us most closely to work on self-supervised depth or 3D estimation, such as work using visual consistency [13, 35, 36, 38, 41]. Bumps offer an alternate source of supervision that does not depend on 3D ray geometry and therefore (as we show) can be used with different modalities, such as sound [12] and low-resolution images.

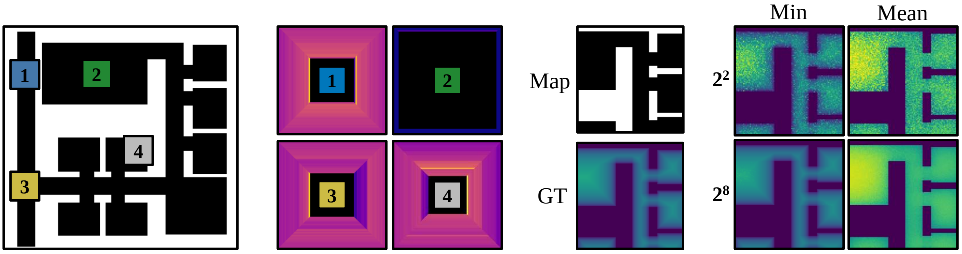
Our supervisory signal, time to collision, requires a few sensors and no human expertise. This is part of a trend in autonomous systems, where learning-based systems are trained on large-scale, noisy datasets of robots performing tasks often with a randomized policy. This has been successfully applied to learning to fly by crashing[11] or with a proximity sensor [20], grasping [27], poking [1], identifying when humans will collide with the robot [23], understanding surfaces through bounce trajectories [28], understanding robot terrain [17], and understanding parts of the image as navigable by labeling regions travelled by the robot as floor [37]. Our work is inspired by this research and applies it by modeling the time to collision with the objects in the world (unlike [23], which modeled collision with a pedestrian). The most similar to us is LFC [11] and [20]. These aim which predicts whether a drone will crash in the next k steps from egocentric views. Our work differs by: using a random, not targeted, policy, and by outputting an angle- and location- conditioned distribution of steps to collision as opposed to a single probability of collision in a fixed time. Finally, we also predict on remote scene points rather than just egocentric views.

3 Method

The heart of our approach is modeling the distribution over time to collision conditioned on an observation and heading angle. We introduce our formulation (Section 3.1) before describing the particular implementations for scenes and egocentric views (Section 3.2).

3.1 Formulation

We begin with an illustration of collision-times in a simple noiseless 2D grid world shown in Fig 2 (although note that the agent does not have access to an overhead view but instead its position and whether it has bumped into something). The agent is at a position $\mathbf{x}$ and has heading α and can rotate in place left/right or take a single step forward. Let us assume the agent follows a random walk policy. For simplicity in Figure 2, we will assume the agent selects one of forward, rotate left, or rotate right with probability $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. In practice, agents may execute other policies that are subsumed under this framework, for instance, a forwards-only policy that will rapidly find the distance to the nearest surface in the forward direction. We see thinking of the agent as on a random walk as useful for accounting for noise: practically, actuation noise may convert a forwards-only policy into a random walk



(A) Environment

(B) Angle-Conditioned Distributions at Points

(C) Results By Sample Count

Figure 2: A toy grid-cell environment shown in overhead view, with free-space shown in black *(A)*. We use this grid-world environment to illustrate the general idea of learning from collisions and how the distribution over collisions carries information about nearby surfaces. We sample random walks at each grid cell for each heading α. The distribution of hitting-times *(B, in log-scale)* reflects the environment. Estimates of the scene distance function *(C, showing one corner)* converge quickly.

with a heavy bias towards forward motion. The random walk formulation can account for the resulting samples occasionally being inaccurate due to an elongated path. Similarly, an agent may accomplish other objectives that move it over the scene, which may be well-approximated by a random walk. Finally, a walk requires no particular skill, compared e.g., to searching each direction, which requires being able to reset the robot to a previous state.

If we let $T \in \{0,1,\dots\}$ be a random variable for the number of forward steps the agent takes until it bumps into part of the scene. The distribution over the number of forward steps until the next bump is the *hitting time* in the study of stochastic processes and Markov chains [10]. Our work aims to learn this distribution conditioned on location $\mathbf{x}$ and heading α. If we condition on location and angle, the probability mass function over T, or $P_T(t|\mathbf{x},\alpha)$, gives the likelihood for the number of steps to a collision starting at a location and heading angle. For instance, in Fig 2, point 1, the agent is likely to bump into the walls if it heads East or West. If the agent sets out at 3 it collide sooner going west versus east.

One can convert this distribution into other quantities of interest, including the distance function to obstacles as well as a floor-plan. The distance function at location $\mathbf{x}$ is the first time t with support at any of heading angles α, or $\mathrm{DF}(\mathbf{x}) = \min_t t$ s.t. $\max_\alpha P_T(t|\mathbf{x},\alpha) > 0$. The $\max_\alpha$ is not necessary if the agent can rotate and rotation does not count as a step. This distance can be converted to a floorplan by thresholding.

In practice, we would like to be able to infer distributions over hitting times in *new* scenes. We therefore learn a function $f(\phi(\mathbf{x}),\alpha)$ that characterizes $P_T(t|\mathbf{x},\alpha)$ as a distribution. $\phi(\mathbf{x})$ are image features at location $\mathbf{x}$ and α is the camera-relative heading angle. We frame the problem as predicting a multinomial distribution over $k+1$ categories: $0,\dots,k-1$ and finally k or more steps. In other words $f(\phi(\mathbf{x}),\alpha)$ is a discrete distribution with $k+1$ entries. If f is learnable, one can train it to match the predictions of f to $P_T(t|\mathbf{x},\alpha)$ by minimizing the expected negative-log-likelihood over the distribution, or $\mathbb{E}_{s\sim P_T(t|\mathbf{x},\alpha)}[-\log f(\phi(\mathbf{x}),\alpha)_s]$. Typically, networks estimate a distribution with something like a softmax function that makes all values non-zero. Thus in practice, we take the first time t where the cumulative probability exceeds a hyperparameter ε, or the minimum t such that $\max_a(\sum_{i=0}^{t} f(\phi(\mathbf{x}),\alpha)_i) \geq \varepsilon$. An alternate approach is predicting a scalar, e.g., via the MSE (which would minimize the expected time to collision).

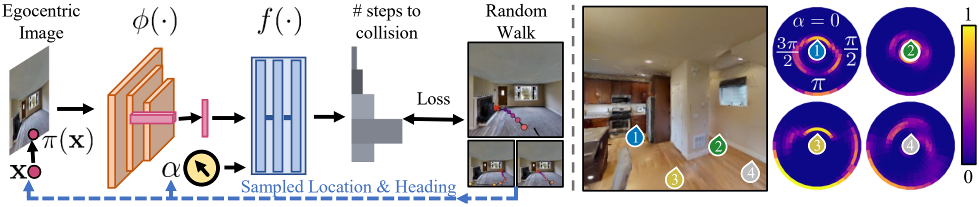


Figure 3: **Overview of training and inference**: Left: Our approach takes a point, $\mathbf{x}$, on a random walk and predicts the distribution of times to collision conditioned on features $\phi(\mathbf{x})$ and the heading angle α. The prediction is supervised by the labels generated for the sampled point using collision replay. Right: We run our predictor for $\alpha \in [0, 2\pi)$ and visualize the distribution over steps to collision for specific points (1, 2, 3, 4) on the ground plane

Distributions like $P_T(t|\mathbf{x}, \alpha)$ are well-studied: in 1D, our setting is the classic Gambler's Ruin [10] problem for which many results are known. A full description is in the supplement since the formulae are unwieldy, but we summarize two salient results. First, the *expected* time (predicted by the MSE) overestimates the distance by a policy-dependent factor; second, approximately short paths (e.g., a few steps longer) are surprisingly likely.

3.2 Learning to Predict Collision Times

Our core idea is that one can derive supervision from sequences containing observations (e.g., images) and collisions, and learn a mapping from the observation to the time until the next collision. At train time, our agent collects many sequences. If a collision occurs at step k, a past step j (with $j<k$) can be labeled with the time to collision $k-j$. In the egocentric case, this entails training a deep network using observation j as input and the value $k-j$ as label. For remote points, we reason about a point on the floor, and predict the value for the point at j as seen from a past location i (with $i<j<k$). In other words, to an agent, remote prediction entails learning that another location is $k-j$ steps away from collision.

Predicting Remote Locations: Our primary setting is predicting the time to collision for a remote location on the ground-plane. This distribution can be converted into a distance function or floorplan for the scene. We predict the distribution using a PIFu [31]-like architecture shown in Fig. 3, in which a CNN predicts image features used to condition an implicit function for each camera ray. In our case, we condition on a heading angle α and projected depth represented in the egocentric frame (instead of only projected depth as in [31]). Thus, $\phi(\mathbf{x})$ is the feature map vector predicted at the location.

At train time, collision replay unwinds the agent's actuations to provide supervision at previous steps. When predicting remote locations, we unwind twice: given three steps i, j, k with $i < j < k$ and a collision at k. We know that step j is $(k-j)$ steps from a collision, so we can project its location into the image at step i and give that pixel the label $(k-j)$. The reasoning is done assuming that the intended egomotion was executed without actuation noise, which is incorrect but simply propagates the noise to the estimate of P_T.

Predicting Current Locations: To show the generality of our method, we experiment with predicting the time-to-collision distribution for the agent's **current location**, conditioned on the agent's next action and current observation. This could be a small image in a low-cost robot, sound, [12] or another localized measurement like WiFi signals. We demonstrate this approach using input from two example sensors:first, a low-resolution (32×32) RGB image, designed to simulate the variety of possible impoverished stimuli found on low cost

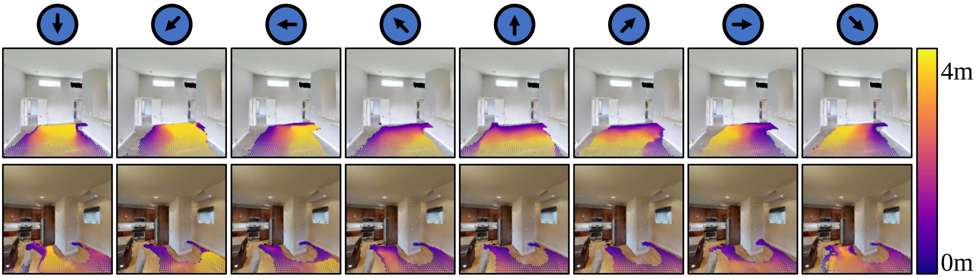


Figure 4: Visualizations of the first step with $> \varepsilon$ probability of collision for remote points with heading angles α shown as arrows. Results are predicted by a model trained with noisy random walks. Points are missing if they were predicted as occupied.

robots, and second, a binaural sound spectrogram, to simulate echolocation. In the remote prediction case, the predictive model has the agent's heading angle, letting it make heading-specific predictions. In the egocentric case, we mirror this by giving the model the agent's next action, which enables similar action-conditioned predictions (i.e., "if I turn right").

Implementation Details: Complete details appear in the supplement, but briefly: both networks are Resnet-18 [16], and the remote prediction network uses a FPN [22] followed by a PIFu [31]-style encoder-decoder. All models are optimized with Adam [19]. Throughout, we set a maximum label of $k = 10$ and clamp the labels; thus class 10 is 10 or more.

Simplifying Modeling Assumptions Made for Simulation: Our experiments test the idea in simulation, and we therefore make a few simplifying modeling assumptions that may not hold in a real-world deployment. First, we assume the agent actually collides. One might handle scenarios in which real collisions are not safe by replacing collision with being close (e.g., $<$ 10cm) to an obstacle, measured with a proximity sensor. This approach has been successfully used by [20] in a similar setting. Second, our agent's policy at training time is random while useful robots usually do useful things rather than randomly walk. We use random policies to show that our system does not depend on a *specialized* policy, and to tie our work with theory on random walks [10]. Different policies change P_T instead of precluding learning. In fact, goal-directed policies may lead to more nearly-short paths.

4 Experiments

We now describe experiments that test the proposed approach, done in simulations with actuation noise (§ 4.1). We compare our approach with alternate ones and upper bounds (§ 4.2) and then analyze the predicted time-to-collision distributions in (§ 4.3). We conclude by using collision-replay with egocentric views and sound (§ 4.4), showing generality.

4.1 Environment and Dataset

Our experiments use data collected from the Habitat simulator [24] and Gibson [39], except for Table 2, which uses Matterport3D [3] to support sound simulation [6]. We use environment splits provided by the datasets for train/val/test. In each environment, we perform 10 random walks of 500 steps, starting at a random location. At each time-step, we collect collision state, sensor data (egocentric images or sound spectrograms), and ego-motion.

Agent and Action Space: Our agent is a cylinder with radius 18cm and a binary collision

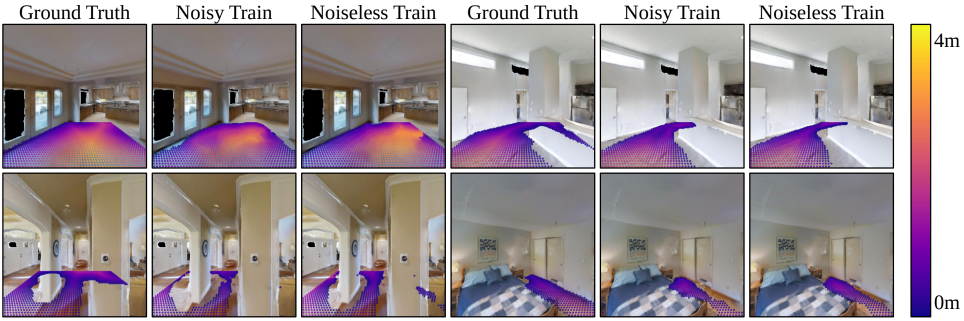


Figure 5: Examples of 2D scene distance functions from (left) the simulator; (middle) a model trained on noisy walks; and (right) a model trained on noise-free walks.

sensor. At each timestep, the agent selects one of four actions like Active Neural Slam [5]: *move forward* (25cm), *turn left* (-10°), *turn right* (10°). We also give the agent a *turn around* (180°) action to use after a collision. We use the noise model from [5], which is a set of Gaussian Mixture Models extracted from a LoCoBot [26] agent; we assume the *turn around* action has distribution like *turn left/right* but with a mean rotation of 180. When collecting data for egocentric prediction experiments, we increase the mean rotation of the turn left and turn right actions to -45° and $+45^\circ$ respectively. This means the network is not tasked with making fine-grained distinctions between angles given a small image while also ensuring the space of possible turn angles covers the 90° field of view of the camera.

Policy: Our agent does a random walk by selecting from the set (*move forward*, *turn left*, *turn right*) with probabilities (60%, 20%, 20%). On collision, the agent executes a *turn around* action to avoid getting stuck at obstacles and walls. As explained in 3.1, this random walk policy is chosen for its simplicity, and the method does not depend on this policy.

4.2 Predicting Remote Time-To-Collision

Qualitative Results: In Fig. 4 we show maps of collision times for specific heading angles for all points within a 4 m x 4 m grid on the floor in front of the camera. We observe that the model correctly varies the time-to-collision such that lower values are predicted at points where the current heading value α faces towards an obstacle, and higher values are predicted where the heading value faces into open areas. Fig. 5 shows the distance function from our angle conditioned outputs with both noisy and (for comparison) noise-free egomotion training. The model produces a distance function considering the closest object in any direction to each point in the scene, rather than the distance in one direction of interest as in Fig. 4. The distance function has high values in the middle of open spaces, and smaller ones in enclosed spaces or near obstacles. Our model can generate the distribution of steps to collision conditioned on the input heading angle α.

Comparisons: For fairness, unless otherwise specified, we use an identical ResNet-18 [16] and PIFu [31] backbone as described in Section 3.2, changing only the size of the outputs to accommodate different targets. Methods that produce only a floorplan are converted using a distance transform. We train multiple variants of the methods, and for each variant report: *(Angle)* whether it does (✓)/does not (✗) consider the heading angle; *(Noise)* whether the training data comes with (✓)/without (✗) actuation noise. Results are shown in Table 1 for: (*Regression-L1*): We use smoothed L1 loss to regress a scalar steps to collision value for each point. We found L1 loss performs best for our task out of all regression losses we

Table 1: **Remote Prediction** Results using the metrics described in § 4.2. Modeling distributions (Classification) outperforms other baselines and is on par with methods with more information (Supervised/Predicted Depthmap).

			Distance Function			Floor
	Angle	Noise	MAE	RMSE	% $\leq \delta$	IoU
Classification (Ours)	✗	✓	0.16	0.31	0.77	0.49
Regression-L1	✗	✓	0.25	0.38	0.66	0.47
Classification	✓	✓	**0.11**	**0.21**	**0.83**	0.47
Regression-L1	✓	✓	0.13	0.24	0.79	0.45
Free-Space	✗	✓	0.13	0.23	0.82	**0.52**
Classification (Ours)	✓	✗	**0.10**	**0.20**	**0.86**	**0.54**
Regression-L1	✓	✗	0.11	0.22	0.83	0.49
Supervised	-	✗	**0.08**	**0.19**	**0.90**	**0.66**
Simulated Depthmap	-	✗	0.09	0.20	0.87	0.57
Predicted [30] Depthmap	-	✗	0.16	0.32	0.75	0.27

tried. Additional comparisons to L2 loss are in the supplement. (*Free-Space*): We train a binary classifier for colliding and non-colliding points. (*Supervised Encoder-Decoder (Supervised)*): We train a model to predict floor plans using dense ground truth floor-plans collected from the environments. This baseline is an upper-bound for our method because it uses more supervision than our method. (*Depthmap Projection (Sim Depthmap)*): We use simulator depth to predict free-space for visible regions in the scene by analytically projecting the depth maps onto the floor plane. *(Predicted Depthmap)*: We also report results using (*MiDaS* [30]) to predict depth and convert it to free-space by projection.

Metrics: We evaluate each timestep on a 64^2 grid of points covering a 4m × 4m square in front of the agent. We compute the distance to the navigation mesh and the occupancy and compute multiple metrics. To evaluate distance functions, we compare the ground-truth y_i and prediction $\hat{y}_i$ and report: mean absolute error (MAE), $\frac{1}{N}\sum_i |y_i - \hat{y}_i|$; root-mean squared error (RMSE), $\sqrt{\frac{1}{N}\sum_i (y_i - \hat{y}_i)^2}$; and percent within a threshold (δ) $\frac{1}{N}\sum_i 1(|y_i - \hat{y}_i| < \delta)$. We set $\delta = 0.25$m, the average step distance. We evaluate floorplans with *Floorplan IoU*, the intersection-over-union between predicted and ground truth floorplans.

Quantitative Results: Results in Table 1 compares out performance against baselines. We observe that despite significantly sparser supervision, methods trained via collision replay produce on results on par with the strongly supervised approaches (using RGB input) and near analytically projecting the Simulator depth (using RGBD input). Our method also outperforms predicted depth (MiDaS [30]). Regression-L1 directly predicts the number of steps and is usually an over-prediction. Most approaches do well on floor-plan IoU as its computation does not require modeling a distance function. Unsurprisingly, training to predict free-space via collision replay outperforms systems that obtain the floorplan estimate by indirect analysis of the distance function. The process of randomly walking around occasionally gives supervision for points behind walls or obstacles. We tested whether collision-replay models can predict these points by evaluating MAE on the subset of test points that are both occluded by an obstacle (computed via ground-truth depth maps) and freespace (≈22% of the data). Our collision-replay-based methods do well: classification, regression, and free-space-predicting methods obtain MAEs of 0.22/0.24/.23 respectively. These approach do

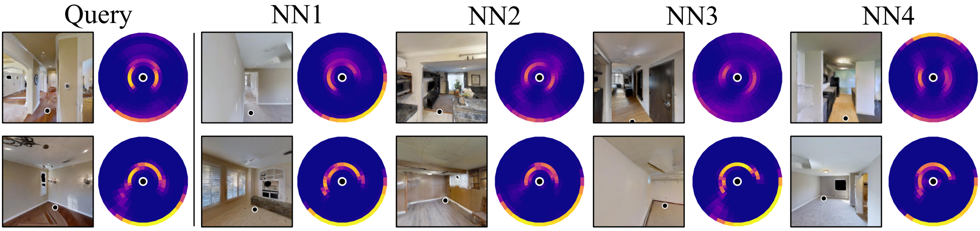


Figure 6: **Nearest neighbor results:** Examples of nearest neighbors using JSD on predicted distributions, and selecting only one per-scene. Top: a doorway; Bottom: room corner.

substantially better than chance, or training-set mean (MAE: 0.32).

4.3 Analysis of Learned Distributions

Qualitatively, we find that the angle-conditioned steps to collision distributions correlate with underlying scene geometry: the distribution in a hallway is different from an empty room. We thus analyze such information in distributions. Given a point in an image we aim to evaluate its similarity to other points in terms of its steps to collision distribution. We start with distributions $P_1(t|\mathbf{x}_1,\alpha_1)$ and $P_2(t|\mathbf{x}_2,\alpha_2)$ conditioned on two points $\mathbf{x}_1$ and $\mathbf{x}_2$ and compute dissimilarity using Jensen-Shannon divergence (JSD) by aligning the relative angle between them. Mathematically, this is $\min_\theta \mathrm{JSD}(P_1(t|\mathbf{x}_1,\alpha), P_2(t|\mathbf{x}_2,(\alpha+\theta \bmod 2\pi)))$.

Qualitative Results: We find nearest neighbors for points using JSD. In the top row of Fig 6, a query point in a hallway returns nearest neighbors that are also doorways or hallways with the similar distributions of $P(t|\mathbf{x},\alpha)$. Similarly, in the second row, a query point in a corner yields the corners of other similarly large and open rooms.

Quantitative Results: We quantify this trend by annotating a subset of test set points with one of 5 classes: Open Space, Wall, Hallway, Concave Corner, or Crowded Space. Many points fall between or outside the categories, so we evaluate on the examples in which 7 out of 9 annotators agreed. This results in a dataset of 1149 points with consensus. We evaluate how well JSD predicts that two locations share the same class, measuring performance with AUROC (0.5 is chance performance). Our method obtains an AUROC of 0.72 distinguishing all 5 labels. Some pairs of labels are easier to distinguish than others: Open-vs-Crowded Space has an AUROC of 0.87 and Wall-vs-Corner has an AUROC of 0.55.

4.4 Predicting Egocentric Time-To-Collisions

Finally, we use collision replay to provide supervision for egocentric observations. We further demonstrate generality with two input modalities: small (32×32) images, and binaural echolocation spectrograms. Given one such input, we predict collision time conditioned on what one does next (*turn left*, *move forwards*, *turn right*).

Qualitative Results: We show qualitative results for the egocentric vision task in Fig. 7. Spectrogram results are in the supplement. The predicted steps-to-collision distribution yields a rough depthmap: in Fig. 7(left), the distribution suggests that the right side of the scene is further away than ahead or to the left. Additionally, the model predicts a bi-modal distribution in Fig. 7(left,right), revealing uncertainty about whether the random walk policy would successfully navigate through the scene.

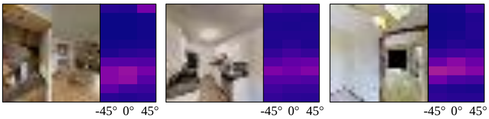


Figure 7: **Egocentric test predictions on 32x32 images:** We show three images and corresponding predicted distributions; the center column is no rotation; left/right show turning ±45°.

Quantitative Results (Visual Input): We first validated that the trend in Table 1, that classification outperforms L1 regression, still holds with egocentric image input. We found classification achieved MAE, RMSE and $\% \leq t$ scores of 0.58, 0.79 and 0.36, whereas L1 Regression achieved 0.96, 1.14 and 0.20. Additionally, we observed that regression produces systematic overestimates (90% of cases vs 60% for classification).

We also constructed a comparison to Learning to Fly by Crashing (LFC) [11]. The learning task used in LFC is a special case of our approach, as it models the task as binary classification by predicting probability of collision within k time steps for a fixed k. We compared our approach by converting its output to predict probability of collision in k time steps, and observe that on accuracy we outperform for all discretizations of k ($k = 8$: 0.66-vs-0.62 and $k = 2$: 0.86-vs-0.85), due to multi-task learning. Comparisons are difficult since LFC aims to produce a policy as opposed to scene structure, which only applies in the egocentric setting. We cannot compare LFC on our metrics as there is no principled way of converting the LFC probabilities to distance functions or floor-plans. By linearly scaling its output, we obtain reasonable results from LFC with large k (RMSE for $k = 8$ is 0.84) and fare worse for small k(RMSE for $k = 2$ is 1.46). Full experiments are in the supplement.

Quantitative Results (Echo-location): In the echolocation setting we compare a model with sound input to the best egocentric vision model. We train both on Matterport3D [3], in order to collect echo spectrograms from Soundspaces [6], which does not currently support Gibson. The vision model is identical to the classification model in Table 1; the sound model replaces the backbone with an Audio-CNN provided by the Soundspaces baseline code. We compare them in Table 2.

Table 2: Egocentric predictions on the Matterport3D [3] dataset with different inputs (visual and audio).

Input	Distance Function		
	MAE	RMSE	$\% \leq t$
Visual	0.64	0.86	0.33
Audio	**0.58**	**0.79**	**0.36**

Sound simulation uses Room Impulse Response (RIR) files that are pre-computed for discrete locations. We apply Gaussian RBF interpolation between the provided RIRs, hence producing an approximate echo-response for any continuous agent location.

Table 2 shows that echolocation outperforms the 32x32 image input. We attribute this to the rich information in audio spectrograms - the return time of emitted sound is related to the underlying depth map of the scene [12], so it provides richer signal than color data. While the sound responses are directional, sound bounces encode information about space around the agent. This is useful as the distance function depends on objects not in camera's FOV.

Discussion. This paper has introduced Collision Replay, which enables using bumps to learn to estimate scene geometry in new scenes. We showcase two applications of our method using sound and vision as input, but see potential in a variety of other settings, ranging from manipulation in occluded regions to converting travel time to scene maps.

Acknowledgements. This work was supported by the DARPA Machine Common Sense Program. NK was supported by TRI. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity

References

[1] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *arXiv preprint arXiv:1606.07419*, 2016.

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[3] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[4] Devendra Singh Chaplot, Emilio Parisotto, and Ruslan Salakhutdinov. Active neural localization. *arXiv preprint arXiv:1801.08214*, 2018.

[5] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations (ICLR)*, 2020.

[6] Changan Chen, Unnat Jain, Carl Schissler, Sebastia Vicenc Amengual Gari, Ziad Al-Halah, Vamsi Krishna Ithapu, Philip Robinson, and Kristen Grauman. Soundspaces: Audio-visual navigaton in 3d environments. In *ECCV*, 2020.

[7] Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*, 2018.

[8] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[9] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. 3-d mapping with an rgb-d camera. *IEEE transactions on robotics*, 30(1):177–187, 2013.

[10] William Feller. *An Introduction to Probability Theory and Its Applications*. Wiley and Sons, New York, 1950.

[11] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3948–3955. IEEE, 2017.

[12] Ruohan Gao, Changan Chen, Ziad Al-Halab, Carl Schissler, and Kristen Grauman. Visualechoes: Spatial image representation learning through echolocation. In *ECCV*, 2020.

[13] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[14] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2017.

[15] R Hartley and A Zisserman. Multiple view geometry in computer vision, cambridge uni. *Pr., Cambridge, UK*, 2000.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[17] Gregory Kahn, Pieter Abbeel, and Sergey Levine. Badgr: An autonomous self-supervised learning-based navigation system. *arXiv preprint arXiv:2002.05700*, 2020.

[18] Kapil Katyal, Katie Popek, Chris Paxton, Phil Burlina, and Gregory D Hager. Uncertainty-aware occupancy map prediction using generative networks for robot navigation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5453–5459. IEEE, 2019.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015.

[20] Kevin Lamers, Sjoerd Tijmons, Christophe De Wagter, and Guido de Croon. Self-supervised monocular distance learning on a lightweight micro air vehicle. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1779–1784. IEEE, 2016.

[21] Chen-Yu Lee, Vijay Badrinarayanan, Tomasz Malisiewicz, and Andrew Rabinovich. Roomnet: End-to-end room layout estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4865–4874, 2017.

[22] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[23] Aashi Manglik, Xinshuo Weng, Eshed Ohn-Bar, and Kris M Kitani. Forecasting time-to-collision from monocular video: Feasibility, dataset, and challenges. *arXiv preprint arXiv:1903.09102*, 2019.

[24] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[25] Kimberly N McGuire, GCHE de Croon, and Karl Tuyls. A comparative study of bug algorithms for robot navigation. *Robotics and Autonomous Systems*, 121:103261, 2019.

[26] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019.

[27] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[28] Senthil Purushwalkam, Abhinav Gupta, Danny M. Kaufman, and Bryan C. Russell. Bounce and learn: Modeling scene dynamics with real-world bounces. *CoRR*, abs/1904.06827, 2019. URL http://arxiv.org/abs/1904.06827.

[29] Santhosh K Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation. *ECCV 2020*, 2020.

[30] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[31] Shunsuke Saito, , Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. *arXiv preprint arXiv:1905.05172*, 2019.

[32] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1):7–42, 2002.

[33] Rakesh Shrestha, Fei-Peng Tian, Wei Feng, Ping Tan, and Richard Vaughan. Learned map prediction for enhanced mobile robot exploration. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1197–1204. IEEE, 2019.

[34] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[35] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] Hsiao-Yu Fish Tung, Hsiao-Wei Tung, Ersin Yumer, and Katerina Fragkiadaki. Self-supervised learning of motion capture. *arXiv preprint arXiv:1712.01337*, 2017.

[37] Iwan Ulrich and Illah Nourbakhsh. Appearance-based obstacle detection with monocular color vision. In *AAAI/IAAI*, pages 866–871, 2000.

[38] Jamie Watson, Michael Firman, Gabriel J Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2162–2171, 2019.

[39] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[40] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018.

[41] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

[42] Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2051–2059, 2018.