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

Alexander Raistrick, Nilesh Kulkarni, David F. Fouhey
University of Michigan
{alexrais,nileshk,fouhey}@umich.edu

## Abstract

*What does bumping into things in a scene tell you about scene geometry? In this paper, we investigate the idea of learning from collisions. At the heart of our approach is the idea of collision replay, where we use examples of a collision to provide supervision for observations at a past frame. We use collision replay to train convolutional neural networks to predict a distribution over collision time from new images. 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 this approach with an agent that has noisy actuation in a photorealistic simulator.*

## 1. Introduction

Suppose an autonomous agent collides with an obstacle. What information does this collision reveal about the scene? Certainly we know that the current position of the agent on the boundary of occupied space, but it is not obvious what is known about nearby locations. If we consider the observation $k$ steps earlier, and replay this observation, we know there is a $k$-step path to an obstacle, but this does not rule out a shorter path or actuation noise. Nonetheless, we will show how this process of replaying a collision (which we call *Collision Replay*) enables us to convert the local and ambiguous signal of a collision into vision-conditioned information about the broader scene and new scenes via time and the accumulation of data.

At the heart our approach is the observation that we can learn to *model* the distribution of collision times given an observation at a location. Each individual collision is a single sample from the true distribution of collision times. Many are overestimates of the distance function of the scene: for instance, in Figure 1 (top), we may pass by the couch on the right of the scene without hitting it. However, over samples across the training set, we learn to match the distribution of collision times and estimate the likelihood that we will hit something in the next $t$ steps. Given this ability to model the collision time, we can then predict the distribution of times in new scenes (Fig. 1 (bottom)), conditioning on angle and convert that distribution to the distance function.

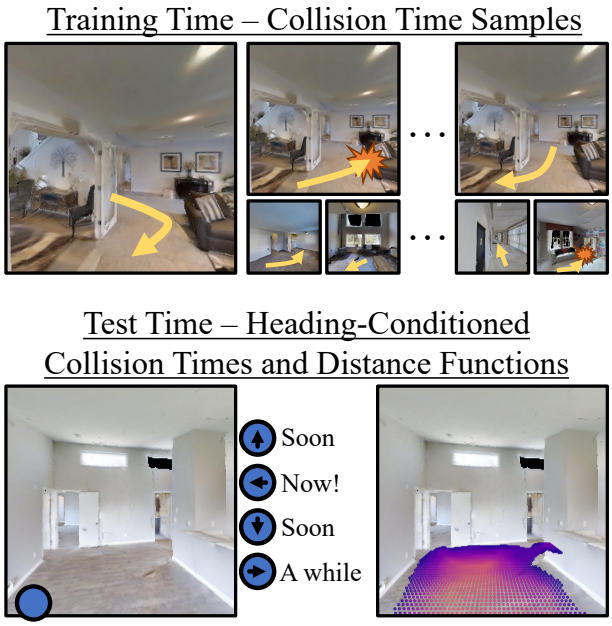



**Figure 1:** At training time, our system receives noisy samples of random walks and collisions. While each individual sample does not reveal the true scene geometry, their collective information lets us learn to predict distributions over collision times from images. In turn, this enables us to predict a distribution over collision times at new locations, and, as we show, distance functions in new scenes.

We see considerable value in learning from collisions. Normally, considerable effort in vision and robotics is spent trying to model and thus avoid collisions [9, 25]. However, collisions serve as clear evidence about the world as a form of simple touch sensing. By combining this with the time-

locked [24] signals of vision and noisy egomotionand the right models, the collision can be converted into the shape of the scene.

We show how to implement this approach in two settings in Section 3. In the first, we predict collision time conditioned on views of other parts of the scene using a PIFu [21]-style architecture. In the second, we predict collision time conditioned on an egocentric view using a standard resnet [10]. In both cases, we learn from collision states and egocentric images collected from an agent executing a random walk in a training scene with noisy actuation [2] and predict an angle-conditioned distribution,

We evaluate our approach in using the Habitat simulator [15] and Gibson Dataset [27] in Section 4. We show that our system that predicts the collision time can accurately capture the distance function of the scene compared to alternatives and that the angle-conditioned distribution over collision times carries information about the local scene topography.

## 2. Related Work

**(DF: We also need to cite Senthil's paper: *BOUNCE AND LEARN: MODELING SCENE DYNAMICS WITH REAL-WORLD BOUNCES* https://arxiv.org/pdf/1904.06827.pdf)**

The goal of this paper is to take a single *previously unseen* image and infer the distribution of steps to collision. We show that this distribution carries information about the shape of the scene in terms of both its distance function and the local topography, and that we can learn this distribution from collisions done by an agent executing a random walk.

Estimating what parts of the scene are occupied and one's distance to them has been long been a goal of vision and robotics, and there are a variety of methods for doing so, each with different assumptions. Collision replay provides a principled model for converting simple observations (paired bumps and ordinary image observations) into a sensor for scene geometry for new images. This ability to infer collision times in new data separates it from a large body of work that aims to build models of an environment over experience, e.g., via SLAM [1, 4, 9, 25]. Similarly, the simplicity of the observations used at training time separates it from approaches that learn to predict floormaps from ground-truth floor plans [12, 20, 23] or RGB-D cameras [3, 8]. In contrast to these, our approach needs to only briefly store frames and register bumps. Our work is therefore most related to work on self-supervised depth or 3D estimation. Our use of collision, an alternate modality, separates us from work that uses visual consistency [7, 26, 29] as does our prediction in invisible regions. Here, we use it as a supervisory signal, unlike the visual echoes work [6], which uses it an input.

The signal that we use, the time to collision, requires few sensors and no human expertise. This is part of a larger 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 [5], grasp [19], identifying when humans will collide with the robot [14], or understand robot terrain [11]. Our work is inspired by this research and applies it by modeling the time to collision with the world (unlike [14], which modeled time to collision with a pedestrian). Of these works, the most similar is [5], which predicts whether a drone will crash in the next $k$ steps. In contrast, we aim to predict a *distribution* over possible times and show that that this distribution conveys information about the scene, in particular the distance function to the scene.

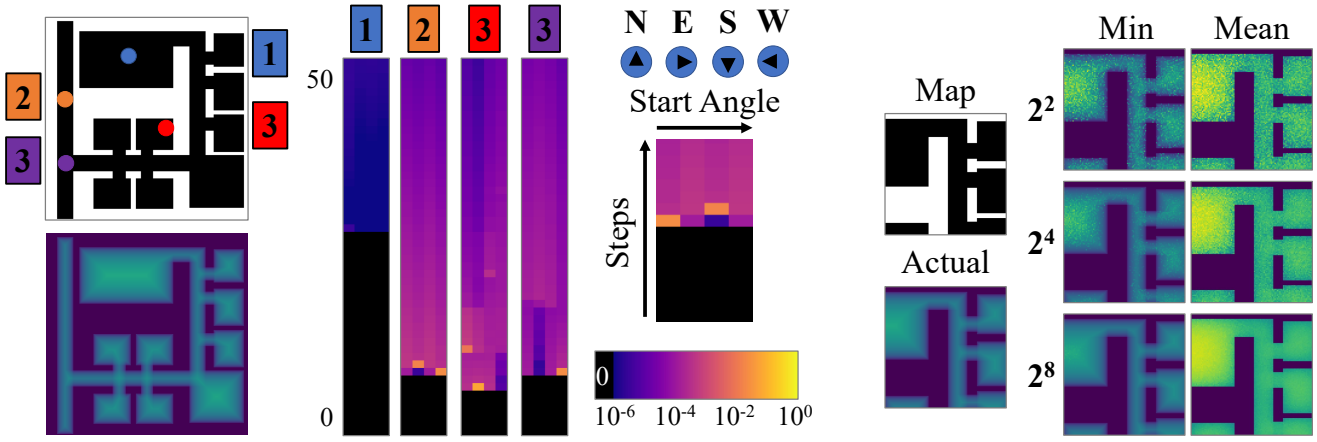
For much of the paper, we make extensive use of implicit representation learning. These implicit functions have become popular for predicting 3D structures [16, 18, 22, 28]. For instance, PIFu [22] learns to predict occupancy and color from images of humans. We use these approaches, although with different inputs and outputs: PIFu conditions each implicit function on image features and a depth value and predict occupancy and color. We condition each implicit function on image features and and an orientation and predict a distribution over time-to-collision if the were agent to start at that location and orientation.

## 3. Method

The heart of our approach is modeling a distribution over time to collision. We show that we can predict this conditioned on angle and visual input with neural networks on new images. Once predicted, these distributions carry information about the local topography and can be converted into distance functions by finding the first non-zero probability. We begin with an illustration of collision-times in a simplified world without inference (Section 3.1), followed by examining what happens if when we try to model this distribution. (Section 3.2). Finally, we we show how to predict these distributions on new data (Section 3.3) for both of our settings.

### 3.1. Preliminaries

We begin with an illustration of collision-times in a simplified, noiseless, discrete 2D grid world shown in Fig 2. In this world, the agent has a position $\mathbf{x}$ and heading $\alpha$ and can rotate in place in either direction and take a single step forward. Let us assume the agent follows a random walk policy in which it chooses randomly between moving forward or rotating. Then, let $t \in \mathbb{N}_0$ denote number of forward steps the agent takes until it next bumps into part of the scene. The distribution $P(t)$ gives the expected forward steps until the next bump, known as the *hitting time* in the study of stochastic processes. This work aims to learn to




(A) Environment (B) Probability Distributions At Points (C) Results By Sample Count

**Figure 2:** A simplified overhead example. In (A), we show an environment from an overhead view and its distance transform. We show collision times conditioned on locations 1-4 and starting angle in (B). Finally, we show a corner of the map (C) and how estimates of the minimum and mean converge according to number of samples.

estimate $P(t)$, conditioned on values representing location $\mathbf{x}$ and heading $\alpha$.

We then condition on location and angle, or $P(t|\mathbf{x}, \alpha)$, which tells us a belief about the steps to bump if we start at a location and heading angle. For instance, in Fig 2, point 2, the agent is in the hallway and likely to bump into the walls it heads out East or West, and is one step closer to the West wall (seen by the peak coming a one unit earlier). If the agent sets out on point 3, going East will almost surely lead to a bump and going west will likely take longer.

Given this distribution, we can readily convert it into a number of other quantities of interest, including the distance function to non-navigable parts of the scene as well as a floor-plan. The distance function at a location $l$ bumps is the first time $t$ which has support across all of the possible heading angles $\alpha$, or

$$\text{DF}(\mathbf{x}) = \underset{t}{\text{argmin}} \text{ s.t. } \max_{\alpha} P(t|\mathbf{x}, \alpha) > 0, \quad (1)$$

where the $\max_\alpha$ is not necessary for if the agent can freely rotate and rotation does not count towards steps.

Throughout, we model the agent's motion as a random walk because this is the simplest policy one can run, requires no skill, and accounts for noise. Naturally, different walk policies give different distributions of collision times: a policy that is rotation heavy is more likely to spin in circles without a collision; a policy that never rotates functions like a laser rangefinder but maintaining perfect heading is difficult without external signals.

### 3.2. Modeling Collision Times

In practice, we are not fortunate enough for the world to give us clean distributions for each $\mathbf{x}$, $\alpha$. Instead, we must instead learn to estimate it via a function $f(\mathbf{x}, \alpha)$ given noisy samples $s$ drawn from $\mathcal{S}$ from an agent in an environment. The noise poses the challenge that some of our samples are wrong. However, even if the samples were noise-free, it takes may samples to converge. This can be seen in Fig 2(C), which is conducted under the generous but *highly* unrealistic assumption of independent randomly sampling a set of random walks per-grid cell: even with 16 independent samples per-pixel, the estimates have not converged.

Our main approach frames the problem as predicting a distribution over a set of $k+1$ categories, $0, \ldots, k-1$ and finally $k$ or more steps. We frame this as a simple classification problem, where $f$ predicts a probability vector $\mathbf{t}$ ($\mathbf{0} \leq \mathbf{t} \leq \mathbf{1}$, $||\mathbf{t}||_1 = 1$). If one minimizes a negative-log-likelihood per-sample $s$, or

$$\mathbb{E}_{s\sim\mathcal{S}} \left[ -\log f(\mathbf{x}, \alpha)_s \right] \quad (2)$$

this is (up to a constant) the KL-divergence between the true distribution $S$ and the prediction $f(\mathbf{x}, \alpha)$: for each sample $s$ from $\mathcal{S}$, there is a corresponding negative-log-likelihood. The network is then incentivized to match $S$ as closely as possible.

Another option is to predict the step count via a regressor $f_R$ that predicts some statistic of $\mathcal{S}$. This minimizes

$$\mathbb{E}_{s\sim\mathcal{S}} \left[ L(s, f_R(\mathbf{x}, \alpha)) \right] \quad (3)$$

for some regression loss $L$. The optimal solution is then the statistic where the loss has its minimum. This minimum is the mean time to collision for MSE and median for L1. The mean, as can be seen in Fig 2, is an over-estimate. Consider, as well, an agent that facing the nearest wall, 1m away: the distance function is 1m irrespective of what is behind the

agent. However, having another wall 1m behind the agent will result in a far lower hitting time compared to a wall 100m behind the agent.

Once we have a function $f$ that gives a distribution for the collision, we may want to decode it into another output space like a distance function or floorplan. Two reasons preclude directly converting a multinomial distribution into a distance function: the output of a softmax function is always positive (i.e., $P(0|t,\alpha) > 0 \; \forall \, t, \alpha$ by definition); even if we adjusted the model, the presence of noise would ensure we always place some chance on undershooting the actual distance. We therefore solve it as $\text{argmin}_t \;\; \text{s.t. } \max_\alpha P(t|\mathbf{x},\alpha) \geq \epsilon$, or the first step where the collision probability exceeds $\epsilon$ where $\epsilon$ is picked on the validation set. Once we have extracted a distance function, we can then recover the floorplan as the super-level set of the distance function (i.e., all points with a distance above a threshold).

### 3.3. Visually Estimating on New Images

Finally, we turn to the challenge of learning to estimating hitting-time distributions on new scenes having trained on finite samples. Modeling the hitting-time distributions on a single scene without generalizing is of limited value: for nearly anything one could keep a count, for instance, finding the distance function by just keeping track of minimum travel time. Thus, instead of conditioning on the location $\mathbf{x}$, we condition on features $\phi(\mathbf{x})$ that can be extracted from the location $\mathbf{x}$ and predict $f(\phi(\mathbf{x}), \alpha)$. The hope is that if $\phi$ is a parameterized and learned function, the process of predicting hitting time distributions on a training set causes the learner to extract meaningful features that generalize well to a new set.

At training time, we have a random walk consisting of locations $(l_0, l_1, \ldots, l_N)$ and observations $(O_0, O_1, \ldots, O_N)$. If there is a collision at time $t$, $l_t$ is labeled as zero steps, $l_{t-1}$ as one step, etc. This yields $t$ training samples of pairs of locations and labels; these are converted to training samples in a setting-dependent fashion.

**Predicting Remote Locations:** Our primary setting is predicting a remote location, where we predict the distribution on **remote location** on the ground plane conditioned on an angle. The result can be converted into a distance function or floorplan for the scene. We predict this distribution using a PIFu [21] architecture, in which a CNN predicts image features used to condition an implicit function for each camera ray that uses depth as an input. In our case, we focus on the ground-plane (which has a unique location of intersection with each camera ray) and so the implicit function is conditioned on heading angle $\alpha$ represented in the egocentric frame. Thus, $\phi(x)$ is the feature map vector predicted at the location.

At training time, collision replay requires unwinding actuation to label each step $i$, and then again to find other steps $j$ where step $i$ is visible. This is because while one is certain of a collision close to the collision time, images at collision times tend to be uninformative and certainly uninformative for collision avoidance. This requires a (potentially noisy) estimate of egomotion from either actuation or perception, but does not require explicit reconstruction. Additionally, it enables the observation of the label from multiple views and even through occlusion.

**Predicting Current Locations:** To show the generality of our method, we experiment with predicting a **current location**, where we predict the collision time given an observation at the current location and a next action. This could be a small image in a low-cost robot, sound [6] or another localized measurement. We use a low-resolution ($32 \times 32$) RGB image to mimic learning from a low-cost sensor or other impoverished stimulus. At training time, we can directly use the label for the given timestep.

**Implementation Details:** *Architectures:* Our remote location network follows the architecture of PIFu [21] using a Resnet-50 [10] backbone. Our current location network uses a Resnet-14 [10] model due to its size. *Training:* We train all remote location networks for 20 epochs. The models for egocentric prediction are trained for 5 epochs due to their smaller image size. All models are optimized with Adam [13], following a cosine schedule with a starting value of 2e-5, a maximum value of 2e-4 after 30% of training has elapsed, and a final value of 1e-5. We apply random horizontal image flips, as well as $N(0, 10\%)$ horizontal image shifts. Points in the remote predictions setting augmented with Gaussian noise within the floor plane, with $\sigma = 3\text{cm}$ *Classification setting:* Throughout, we used a maximum label of $k = 10$ steps for our classification model. Given the action space we used, we found this was sufficient to handle the interesting parts of most scenes. A full description of implementation details appear in the supplemental material.

## 4. Experiments

We now describe a series of experiments that aim to investigate the effectiveness of the proposed approach. These experiments are conducted in a standard simulation environment, including results with actuation noise. After explaining the environmental setup **(Section 4.1),** we introduce a series of experiments. We first evaluate predicting on remote points **(Section 4.2)**, comparing our approach to a variety of alternative architectures and upper bounds. We then analyze what the predicted time-to-collision distributions represent **(Section 4.3)**. Finally, we demonstrate the generality of the idea by applying it to egocentric views **(Section 4.4)** with a different input and action space.

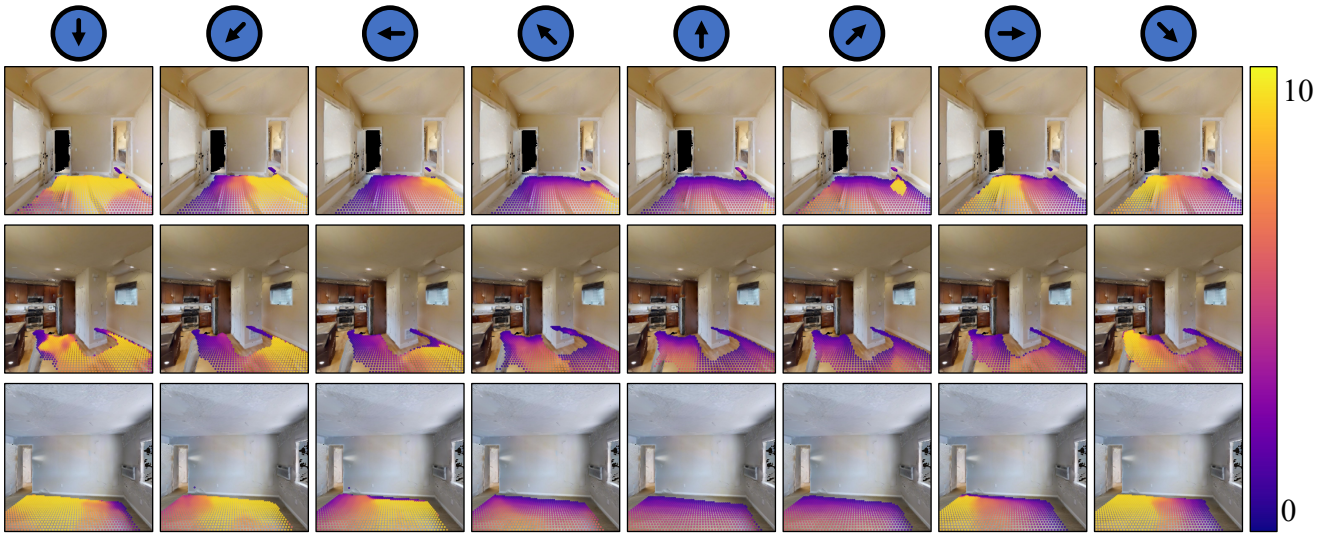



**Figure 3:** Visualizations of the estimated first step with $> \epsilon$ probability of collision for grids of remote points with different heading angles $\alpha$. Points are missing if the method predicted them to be occupied. These results come from a classification model that was trained with noisy walks.

### 4.1. Environment and Dataset

We conduct our experiments in the Habitat simulator [15], using the Gibson Dataset [27]. For each of 360 training environments, we conduct 10 random walks of 500 time steps each, collecting collision state, egocentric image data, and estimated egomotion at each timestep. At test time, we conduct 10 similar 500 step episodes in held out sets of 35 validation environments and 35 test environments.

**Agent and Action Space:** We model the agent as a cylinder with a radius of 18cm, a forwards facing camera 1.5 meters above the ground plane, and a binary sensor that indicates if the agent has collided with the scene. At each timestep, the agent selects one of four actions. Three follow Active Neural Slam [2]: *move forward* (25cm), *turn left* ($-10°$), *turn right* ($10°$). We also give the agent a fourth *turn around* ($180°$) action for use after a collision. When noise is active, we use the noise model from [2], which are a set of Gaussian Mixture Models extracted from a LoCoBot [17] agent; we assume the *turn around* action is distributed like *turn left* 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 $+45°$ respectively. This means the network is not forced to make fine-grained distinctions between angles given a small image while also ensuring the space of possible turn angles reasonably covers the $90°$ field of view of the camera.

**Policy:** Our agent conducts a random walk by selecting an action from the set (*move forward*, *turn left*, *turn right*) with probabilities (60%, 20%, 20%) so long as it did not collide with an obstacle in the previous time step. Upon collision, the agent executes a *turn around* action to prevent getting stuck against obstacles and walls.

### 4.2. Predicting Remote Time-To-Collision

**Qualitative Results:** We first show qualitative results of maps of the collision time in Fig. 3. We observe that the model correctly varies the time-to-collision prediction such that lower values are predicted at points where the current heading value $\alpha$ faces a nearby obstacle, and higher values are predicted where the heading value faces into open areas. The prediction also correctly varies from high to low values in hallways and passageways as the heading angle varies between the orientation of the hallway, and perpendicular to it. Additionally, the model is able to infer some regions where the distance function is nonzero despite that the floor plane itself is occluded by obstacles.

As shown in Fig. 2, we can concisely describe the collision time probabilities in a chart showing $P(t|\alpha)$ for varying $\alpha$. We show a set of these in Figure 5. For each example image, we choose a single point shown as a circular marker in the left image, and evaluate the model at that point. This produces a distribution of possible time-to-collision values, shown as a matrix on the right of each image. As one moves from the left to right of each distribution, we see the distribution of possible times-to-collision starting from a relative heading of $-180°$ at the leftmost column, through to $0°$ at the center column, and finally $180°$ at the rightmost column.

These distributions can be decoded to produce distance functions for the scene as described in 3, which we show


**Figure 4:** Examples of 2D scene distance functions extracted both from (left) the simulator navigation-mesh; (middle) a model trained on noisy random walks; and (right) a model trained on noise-free random walks.

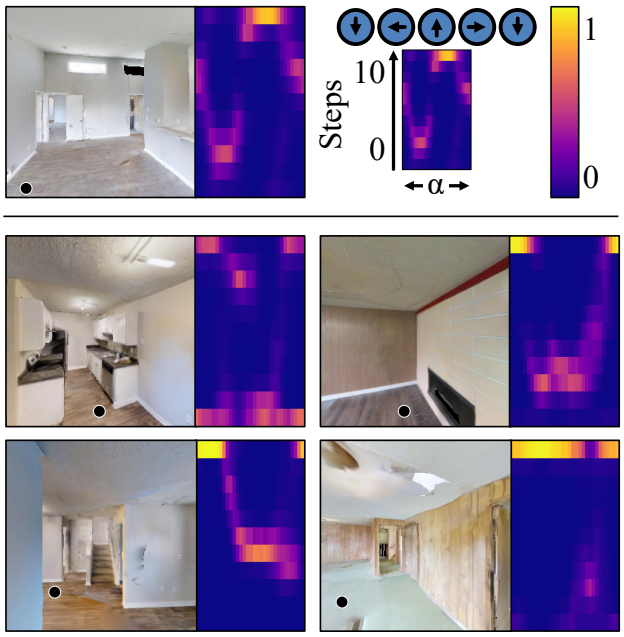



**Figure 5:** Examples of $P(t|\alpha)$ for selected points in the test set. The center column of each heatmap represents the distribution over over times to the next collision with a heading of $0^\circ$ with columns to the left and right representing different angles $\alpha$. In the illustrative example, a collision is likely soon if one goes to the left, while heading to the right is less likely to collide.

in Fig. 6 with both noisy and (for comparison) noise-free training. We see that 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. 3. The distance function attains high values in the middle of open spaces, and smaller values in enclosed hallways and near obstacles.

**Comparisons:** We compare against a number of approaches and ablations. For fair comparison, unless otherwise specified, we use an identical ResNet50 [10] PiFU [21] backbone, and change only the size of the last layer to accommodate different prediction settings. All outputs can be converted into a predicted floorplan and a scene distance function. Methods which produce binary outputs can be made to produce a distance function by applying a distance transform to their predicted floorplans. We train both classification and regression models with and without conditioning on angles.

(*Regression*): We train a network to regress the log of the number of steps to collision with a Smoothed L1 loss. We use the log to account for the natural long-tailed distribution of time to collision labels and the Smoothed-L1 to prevent large values (such as when a long path narrowly misses an obstacle) from dominating gradients.

**(DF: (*Regression-L1*:) We also train a network using a a standard L1 loss to regress the number of steps to collision.)**

**(DF: (*Regression-L2*:) We also train a network using a standard Mean-Squared Error loss (i.e., minimizing the L2 distance) to regress the number of steps to collision)**

(*Free space Classification*): We train a binary classification model which predicts the collision label provided by the agent's collision sensor at each time step visible to the camera, thus learning to predict whether a given pixel is navigable or occupied.

(*Strongly Supervised Encoder-Decoder*): To give a sense of an upper bound of what would be expected with neural net machinery, we train an encoder-decoder network which encodes the agent's camera observation into a 2048-dimensional vector using the same ResNet50 backbone as

**Table 1:** Quantitative results for distance functions and floorplans.

| | Angle | Noise | Distance Function MAE | RMSE | $\leq t$ | Floor IoU |
|---|---|---|---|---|---|---|
| Classification | ✗ | ✓ | 0.16 | 0.31 | 0.77 | **0.49** |
| Regression | ✗ | ✓ | 0.25 | 0.36 | 0.66 | 0.46 |
| Classification | ✓ | ✓ | **0.11** | **0.21** | **0.83** | 0.47 |
| Regression | ✓ | ✓ | 0.16 | 0.25 | 0.77 | **0.49** |
| Regression-L1 | ✓ | ✓ | | | | |
| Regression-L2 | ✓ | ✓ | | | | |
| Classification | ✓ | ✗ | **0.10** | **0.20** | **0.86** | 0.54 |
| Regression | ✓ | ✗ | 0.14 | 0.24 | 0.80 | **0.56** |
| Regression-L1 | ✓ | ✓ | | | | |
| Regression-L2 | ✓ | ✓ | | | | |
| Free-Space | ✗ | ✓ | 0.13 | 0.23 | 0.82 | 0.52 |
| Supervised | - | ✗ | **0.08** | **0.19** | **0.90** | **0.66** |
| Depthmap | - | ✗ | 0.09 | 0.20 | 0.87 | 0.57 |

in other models, which is decoded to predict a 256x256 floorplan for the visible scene. This model is supervised with a Binary Cross-Entropy loss against dense ground truth floor plans extracted from Habitat.

(*Analytical Depthmap Projection*): Our task requires models to predict the distance function and navigability of occluded regions of the scene. To give an upper bound of the performance of a model which only considers the visible portions of the scene, we implement a method which determines freespace by analytically projecting ground truth simulator depth maps onto the floor plane.

**Metrics:** We evaluate each timestep on a $256 \times 256$ grid of points covering a $4\text{m} \times 4\text{m}$ square in front of the agent. We compute both the distance to the navigation mesh and the whether the point is free space. We use these labels to compute multiple metrics for methods that produce a distance function and floorplan. To evaluate distance functions, we use take 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 $t$ ($\% \leq t$), $\frac{1}{N}\sum_i 1(|y_i - \hat{y}_i| < t)$. Throughout, we use $t = 0.25\text{m}$, the average step distance. To evaluate floorplans, we compute the average *Floorplan IoU* and compute the intersection-over-union (i.e., $TP/(FP+TP)$) between predicted and ground truth floors.

**Quantitative Results:** We report results in Table 1 with distance measured in meters and $t$ set to the robot width. In each both noise-free and noisy settings, the approach of modeling with classification produces better distance functions compared to regression approaches; floorplan IoU has a far smaller dynamic range and regression approaches narrowly outperform classification approaches. **(DF: This part is a bit weak. Floorplan estimation is a weird problem in the sense that if you goof up higher distances it doesn't matter.)** Unsurprisingly, training to predict free-space via collision replay outperforms systems that obtain the floorplan estimate by indirect analysis of the distance function. Nonetheless, one can observe that most of the methods trained via collision replay produce results that are close to the performance of methods the strongly supervised approach and using the analytical depthmap. However, we see advantages to rich representation of collision time distributions that go beyond the ability to sense the scene.

**(DF: Comment on overshoot/undershoot. This will be important: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus ac eros quis mauris tempus vehicula. Duis pretium augue sit amet elementum ultrices. Sed quis risus massa. Ut tempor augue metus, non imperdiet metus iaculis vitae. Praesent id finibus metus, sit amet pharetra nibh. Integer varius justo massa, nec venenatis ligula hendrerit ac. Aliquam erat volutpat. Curabitur sed pharetra nisi. )**

### 4.3. Analysis of Learned Distributions

We next analyze what the distributions that the method predicts captures. As shown in Fig. 2, the distribution of hitting times conditioned on angles carries information about the surrounding environment and where one can move. Here, we analyze this in the context of the remote time-to-collision experiment.

In order to do this, we to be able to express similarity between two distributions over collision times. We start with distributions of $P_1(t|\alpha)$ and $P_2(t|\alpha)$ conditioned on two points; we can compare these using any divergence. The immediate difficulty is that $\alpha$ is expressed in the camera view and so a high dissimilarity either indicates a different nearby space or different viewing angle. We therefore use a dissimilarity function that also aligns the angles,

$$\min_{\theta} \text{SymKL}(P_1(t|\alpha), P_2(t|(\alpha+\theta) \bmod 2\pi)), \quad (4)$$

or the symmetrized KL-divergence between the two distributions when they have been best aligned in angle. If the distributions are estimated correctly and represent the same point in space, the relative angle between the views is a minimum for Eqn. 4.

**Qualitative Results:** We show some selected nearest neighbors using Eqn. 4 in Fig. 6. In the top row, we observe that a query point in a door way returns nearest neighbors that are also doorways or hallways with the same distinct distribution of $P(t|\alpha)$. In the second row, a query point in the corner of a room yields the corners of other similarly large and open rooms. Lastly, the third row shows that a query point against a surface yields a similar learned distri-

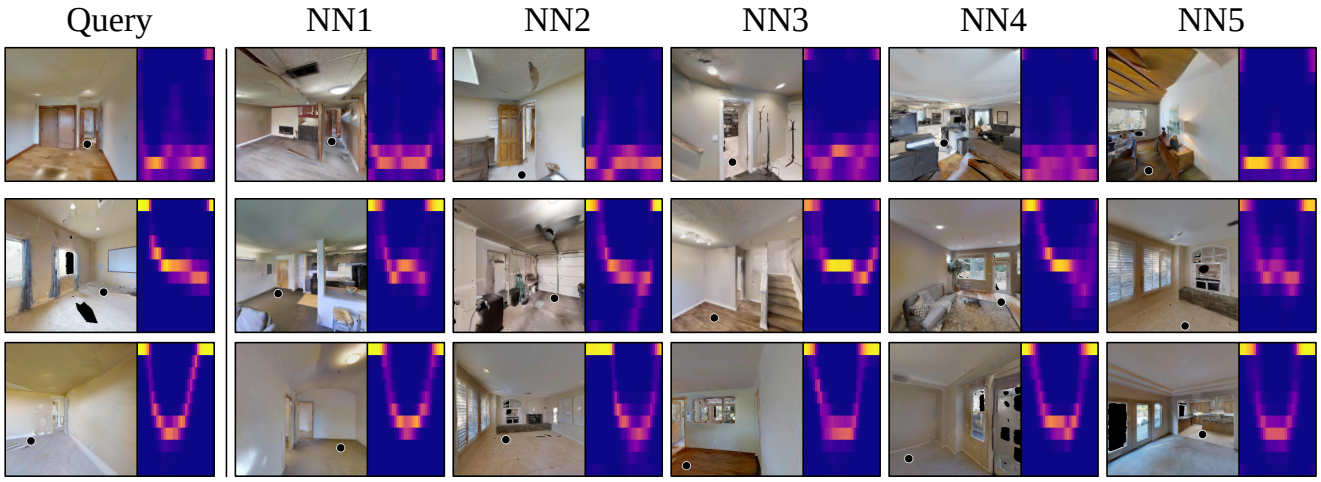



**Figure 6:** Examples of nearest neighbors using Eqn. 4 on predicted hitting time distributions, and selecting only one per-scene. Top: passing through a doorway; Middle: corner of a room; Bottom: near a wall.

bution of time-to-collisions, despite being drawn from different unseen test environments.

**Quantitative Results:** To quantify performance, we manually labeled a subset of the test images with a set of topographic labels. We discarding images that labelers marked as ambiguous or occupied as well as corners, which labelers found difficult to distinguish from other categories. This resulted in 258 images with four labels: Open Space, Hallway, Wall, and Crowded Space. We then used Eqn. 4 as a predictor that two locations share the same location type. We evaluate performance using the area under the receiver operating characteristic (AUROC) where 0.5 indicates chance performance. The distance function achieves an overall score of 0.61 when distinguishing between Open/Hallway/Wall/Crowded. Some pairs are easier to distinguish than others: Crowded-vs-Open and Hallway-vs-Open have AUROCs of 0.75 and 0.68 respectively, while Hallway-vs-Wall has an AUROC of 0.55.

### 4.4. Predicting Egocentric Time-To-Collision

We next test the generality of the approach by testing time to collision given an observation at the current time. Here, we use a small ($32 \times 32$) image as an observation as a stand-in for a variety of other options of senses one could collect at that location (e.g., sound, wifi). Our goal is then to model the collision time conditioned on what one does next (*turn left*, *move forwards*, *turn right*).

**Qualitative Results:** We show qualitative results in Fig. 7. The resulting time-to-collision distribution functions as a crude depthmap: for instance, in Fig. 7(top-left), the distance function suggests that part of the scene that is on the right side is substantially further away than the part that is ahead or to the left. Additionally, the model predicts a bi-modal distribution in 7(bottom-left), correctly mirroring uncertainty as to whether the random walk policy will go through the doorway or collide with it.

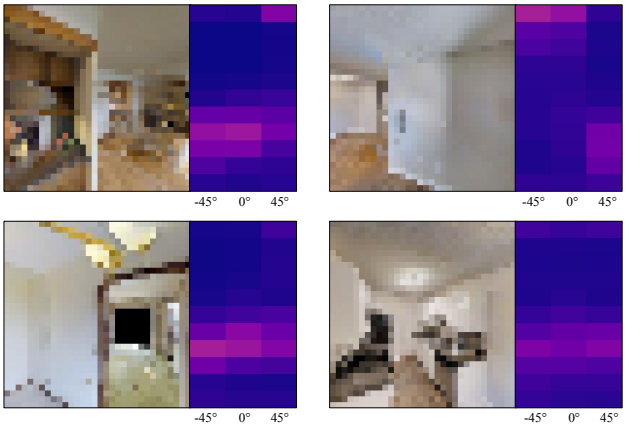



**Figure 7:** Example estimates of $P(t|\alpha)$ for selected low resolution images in the egocentric images test set.

**Table 2:** Quantitative results for egocentric distance function prediction

| | Distance Function | | |
|---|---|---|---|
| | MAE | RMSE | $\% \leq t$ |
| Regression | 0.96 | 1.14 | 0.13 |
| Classification | **0.58** | **0.79** | **0.36** |

**Quantitative Results:** There are no comparable works, and so we compare with variations of our model to see whether trends observed in the remote time-to-collision set-

ting translate to the egocentric one. We evaluate approaches using the same distance function metrics as in the remote prediction case, but on their ability to predict the distance function at the camera position. In the *Regression* method we regress time to collision labels, and multiply by expected forwards motion to obtain a distance function. In the *Classification* method, we apply the same minimum support formulation as in Eqn. 1. Modeling the time via classification produces a better estimate of the true distance compared to regression. In particular, regression (as Fig. 2 would suggest) systematically overestimates, producing an overestimate $\sim 90\%$ of the time compared to $\sim 60\%$ for classification

## 5. Conclusions and Discussion

This paper has introduced the idea of collision-replay which enables translating observations of the scene and bumps into the ability to estimate scene geometry in new scenes. While a simple idea, we show the careful consideration of the modeling of the problem is important. The paper presented the concept in the particular case of an agent navigating in the plane of the floor using images as observations. However, we see a variety of other settings in which this can be applied, ranging from manipulation in occluded regions to converting travel time to scene maps.

**Acknowledgments:** This work was supported by the DARPA Machine Common Sense Program. The authors thank the ghost of JJ Gibson for inspiring them and the Fleetwood Diner for nourishing them.

## References

[1] 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. 2

[2] 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. 2, 5

[3] Devendra Singh Chaplot, Emilio Parisotto, and Ruslan Salakhutdinov. Active neural localization. *arXiv preprint arXiv:1801.08214*, 2018. 2

[4] 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. 2

[5] 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. 2

[6] Ruohan Gao, Changan Chen, Ziad Al-Halab, Carl Schissler, and Kristen Grauman. Visualechoes: Spatial image representation learning through echolocation. In *ECCV*, 2020. 2, 4

[7] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017. 2

[8] 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. 2

[9] R Hartley and A Zisserman. Multiple view geometry in computer vision, cambridge uni. *Pr., Cambridge, UK*, 2000. 1, 2

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 4, 6

[11] Gregory Kahn, Pieter Abbeel, and Sergey Levine. Badgr: An autonomous self-supervised learning-based navigation system. *arXiv preprint arXiv:2002.05700*, 2020. 2

[12] 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. 2

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015. 4

[14] 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. 2

[15] 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. 2, 5

[16] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 2

[17] 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. 5

[18] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 2

[19] 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. 2

[20] Santhosh K Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation. *ECCV 2020*, 2020. 2

[21] 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. 2, 4, 6

[22] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2304–2314, 2019. 2

[23] 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. 2

[24] Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artif. Life*, 11(1–2):13–30, Jan. 2005. 2

[25] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002. 1, 2

[26] 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. 2

[27] 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. 2, 5

[28] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems*, pages 492–502, 2019. 2

[29] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 2