# *Safe Visual Navigation via Deep Learning and Novelty Detection*

# Safe Visual Navigation via Deep Learning and Novelty Detection

Charles Richter and Nicholas Roy
Massachusetts Institute of Technology
Cambridge, MA, USA

***Abstract*—Robots that use learned perceptual models in the real world must be able to safely handle cases where they are forced to make decisions in scenarios that are unlike any of their training examples. However, state-of-the-art deep learning methods are known to produce erratic or unsafe predictions when faced with novel inputs. Furthermore, recent ensemble, bootstrap and dropout methods for quantifying neural network uncertainty may not efficiently provide accurate uncertainty estimates when queried with inputs that are very different from their training data. Rather than unconditionally trusting the predictions of a neural network for unpredictable real-world data, we use an autoencoder to recognize when a query is novel, and revert to a safe prior behavior. With this capability, we can deploy an autonomous deep learning system in arbitrary environments, without concern for whether it has received the appropriate training. We demonstrate our method with a vision-guided robot that can leverage its deep neural network to navigate 50% faster than a safe baseline policy in familiar types of environments, while reverting to the prior behavior in novel environments so that it can safely collect additional training data and continually improve. A video illustrating our approach is available at: http://groups.csail.mit.edu/rrg/videos/safe_visual_navigation.**

## I. Introduction

In many ways, cameras are ideal sensors for mobile robot applications since they provide rich contextual information that is required for many different types of navigation and decision making problems. They are much smaller, lighter and cheaper than LIDAR, and work in a wide variety of indoor and outdoor lighting conditions, unlike most infrared RGBD sensors. However, cameras pose substantial computational challenges to extract information in usable forms. For example, the range of dense, accurate depth information from visual SLAM may be limited (like RGBD) to only a few meters. Therefore, it is reasonable to assume that mobile robots will be able to build geometric maps from images, but that those maps will be very limited in range. Range limitations, in turn, restrict the speed at which a robot can travel, and impede its ability to anticipate more distant structures in the environment.

Nevertheless, camera images contain information about the context and structure of the world far beyond the limited range of accurate geometric inference. For instance, if a robot observes a straight empty hallway ahead, it might reasonably learn from experience and visual appearance that it can safely travel at high speed down the hallway, even if it cannot infer the exact geometry of the entire length of the hallway. If we can extract this type of appearance-based information by other means, we can improve navigation performance.

Deep learning has been shown to be well suited to extracting useful information for navigation from raw sensor data such as camera images [13, 27]. However, unlike visual SLAM algorithms that infer geometry using domain-invariant principles, neural networks adapt their representation to domain-specific training datasets. Therefore, they may make unreliable predictions when queried with inputs far from the distribution of the training data [4]. Yet, we would still like to deploy robots using neural networks "in the wild," where safety is critical but real-world data will almost certainly differ from the training dataset, violating the common i.i.d. assumption in machine learning. Moreover, conventional neural networks provide no principled measure of how certain they are, or how well-trained they are to make a given prediction.

Recognizing this limitation, some work has attempted to quantify uncertainty with Bayesian neural networks [20, 5], by training ensembles of networks on re-sampled versions of the training dataset [26, 17, 8, 18], or by using dropout to train an implicit family of models [9]. While these methods are equipped to model the variance *within* the training data, there has yet to emerge an established technique for reliably producing appropriate uncertainty estimates for regions of input space very far from the training data. Recent techniques appear to be sensitive to particular choices of network architecture and residual randomness from initialization [10, 25], and unlike Bayesian techniques such as Gaussian processes, these methods do not predictably converge to a prior mean and variance far from the training data. Furthermore, these methods require tens or hundreds of network evaluations to approximate the output distribution, compromising runtime efficiency. Given these limitations, we believe that determining the trustworthiness of a neural network prediction may be more effectively addressed as a novelty-detection problem.

In this paper, we demonstrate safe, high-speed visual navigation of an autonomous mobile robot, guided by a neural network in environments that may or may not resemble the training environment(s). We use a conventional feedforward neural network to predict collisions based on images observed by the robot, and we use an autoencoder to judge whether those images are similar enough to the training data for the resulting neural network predictions to be trusted. In fact, Pomerleau [28] utilized a similar form of novelty detection when using a neural network to control an autonomous vehicle, which we extend in several ways. First, if our novelty detector classifies a planning scenario as novel, we revert to

a safe prior behavior, which enables our system to operate autonomously in any environment, regardless of whether it has received appropriate training. Second, we continuously collect training data and label it in a self-supervised manner so that we can periodically re-train our models to improve, and expand the set of familiar environment types, rather than needing human demonstrations or manually-designed visual simulators for training in every environment type.

In the next section, we will motivate and present the problem formulation of our robot navigation scenario, which builds upon our prior work in Richter et al. [29]. Then, we will describe our method of learning to predict collisions from camera images, and our approach to novelty detection. Finally, we will present results including novelty detection and navigation performance in several domains. Our simulation results show that our learning method improves performance significantly, in multiple environment types, while remaining collision-free in all trials throughout the learning process. Our experimental results demonstrate the effectiveness of our approach in a real navigation scenario on a high-speed RC car.

## II. Visual Navigation Problem Formulation

We aim to solve the problem of navigation to a goal location incurring minimum expected cost in an unknown map. We define the cost of navigation to be equal to the time spent en route from start to goal, plus a fixed penalty on any collisions that may occur along the way. We assume that we will be building a geometric map online using SLAM because it is helpful for practical reasons such as measuring progress toward the goal. But, for the reasons discussed in Section I, we also assume that our ability to accurately infer dense environment geometry through our sensors is limited to 5 m or less, which is characteristic of many monocular, stereo and RGBD methods. The Microsoft Kinect sensor, for instance, has an advertised range of only 4 m [1]. Since a short perceptual horizon limits the available stopping distance of the robot, the map itself is insufficient for high-speed navigation. As has been shown previously, using learned models for image-based guidance can significantly extend the effective perceptual range and improve navigation performance [13, 34].

Let $m$ denote the true, hidden, stationary map of the environment, and $\hat{m}_t$ be a partial estimate of the map built from a SLAM process. The map estimate $\hat{m}_t$ is computed from the complete history of geometric sensor measurements $z_0, \ldots, z_T$ up to time $T$. The geometric measurements $z_t$ may come from a geometric sensor such as an RGBD camera or from processing stereo or monocular camera images (or even from a simulator during training time). We refer to the raw camera image at time $t$ as $i_t$. We consider $z_t$ and $i_t$ to be separate measurements, although in certain types of visual SLAM systems, $i_t$ could be processed to produce a geometric measurement $z_t$.

Since the map is initially unknown, it is impossible to meaningfully plan a detailed complete path from the start location to the goal. Therefore, we adopt a receding-horizon replanning approach that selects a local action, $a_t$, of some length, that avoids local obstacles that appear in the map estimate $\hat{m}$, avoids future collisions in expectation, and makes progress toward the goal. The optimal action at time $t$ is chosen according to the following optimization problem:

$$a_t^* = \underset{a_t \in \mathcal{A}}{\text{argmin}} \; J_a(a_t) + J_c \cdot p(c|\hat{m}_t, i_t, a_t) + h(a_t, \hat{m}_t) \quad (1)$$

$$\text{s.t.} \;\; g(\hat{m}_t, a_t) = 0. \quad (2)$$

In this control law, $a_t \in \mathcal{A}$ is a dynamically feasible action, $J_a(a_t)$ returns the time duration of action $a_t$, and $h(a_t, \hat{m}_t)$ is a heuristic function that estimates the remaining time to navigate from the end of action $a_t$ to the goal. The second term is an expected collision penalty in which $p(c|\hat{m}_t, i_t, a_t)$ is the probability that a collision will inevitably occur *beyond the end* of action $a_t$, given the current map estimate and image.

Finally, $g(\hat{m}_t, a_t)$ is a collision checking function that returns 1 if action $a_t$ intersects an observed obstacle in the map estimate $\hat{m}_t$, and returns 0 if the action lies entirely within known free space or unobserved space. Note that a conventional collision-checking routine would also classify actions intersecting unknown regions of the map as being in collision as well, in order to formally guarantee safety. However, we consider potential collisions beyond the mapped regions of $\hat{m}_t$ to be the responsibility of the collision prediction term, not the constraint. This formulation will allow us to plan actions whose stopping distance exceeds the available known free space in cases where we trust the collision prediction—a topic that we will describe in detail below.

As stated, this receding-horizon control formulation is straightforward, with one exception: The collision probability $p(c|\hat{m}_t, i_t, a_t)$ is an unknown quantity. In a conventional MDP or POMDP scenario, this probability distribution might be derived from the distribution over environments given in the problem definition. However, since we do not know the distribution over the real-world environments in which we hope to plan, we will approximate the distribution $p(c|\hat{m}_t, i_t, a_t)$ by learning a function from data.

In this work, we employ a neural network to predict collision probability. Therefore, we may be tempted to simply model the collision probability term in equation (1) as:

$$p(c|\hat{m}_t, i_t, a_t) = f_c(c|i_t, a_t), \quad (3)$$

where $f_c(c|i_t, a_t)$ is implemented as a neural network that takes an image and action as input. Since we would like to predict collisions based on images, not geometric maps, we have dropped the dependence on $\hat{m}_t$.

However, as we have noted, neural networks may make erratic predictions when queried with inputs unlike their training distribution, so we must be able to detect when an input is novel and revert to a default collision probability estimate that is known to be safe, though perhaps conservative. To do so, we introduce a function $f_n(i_t)$, which returns 1 if image $i_t$ is novel (i.e., unlike the distribution of training images), and returns 0 otherwise. We then model the collision probability

distribution as follows:

$$p(c|\hat{m}_t, i_t, a_t) = \begin{cases} f_c(c|i_t, a_t) & \text{if } f_n(i_t) = 0 \\ f_p(c|\hat{m}_t, a_t) & \text{if } f_n(i_t) = 1. \end{cases} \quad (4)$$

In this equation, the function $f_c(c|i_t, a_t)$ is a neural network trained to predict collisions, as in equation (3), and $f_p(c|\hat{m}_t, a_t)$ represents a prior estimate of collision probability based on the geometric map estimate $\hat{m}_t$ that can encourage safe and sensible behavior when the robot is faced with a novel-appearing environment for which $f_c$ is untrained. A reasonable prior might be to limit speed such that the robot can stop within the known free space of map $\hat{m}_t$ if need be, and since our geometric sensing range is limited, this speed may be rather slow. In completely novel-appearing environments, the robot will navigate according to the behavior suggested by the prior $f_p$, whereas in environments of familiar appearance, the robot will trust its learned collision prediction model $f_c$.

In the next section, we will describe how to represent and train the function $f_c(c|i_t, a_t)$ as a feedforward neural network and how to model the prior collision probability estimate, $f_p(c|\hat{m}_t, a_t)$. Then, we will describe the method we use for building $f_n(i_t)$ to detect novel images.

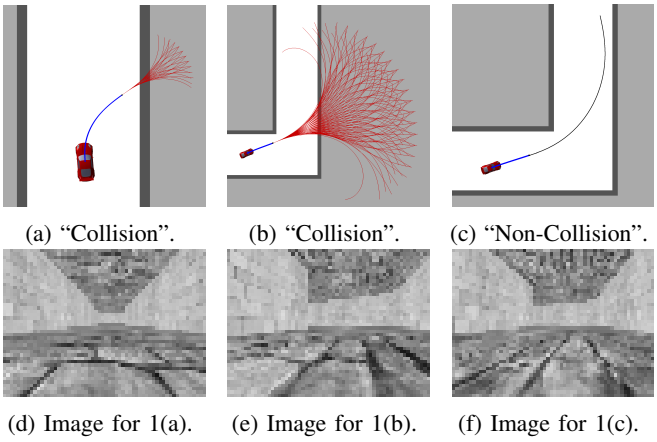
## III. Learning to Predict Collisions

As we have observed in Section I, there are several advantages to predicting collision from images based on training experience, rather than relying solely on a geometric map. First, images contain information about the environment that extends well beyond the range to which detailed geometry can be reliably inferred. And second, a prediction based on training data can account for characteristics of the environment that are not explicitly visible in the image but are implied by the visual appearance, such as free space around a blind corner. Similar to our previous work in Richter et al. [29], our objective in training a model to predict collision probabilities is to capture these advantages implicitly through training examples. We approach this problem as a probabilistic binary classification problem: Given a camera image, and some choice of action, what is the probability that a future collision is inevitable beyond the end of the chosen action? Therefore, we must devise a method for building a training dataset that provides labeled examples of image-action pairs, with a binary response indicating whether a collision was inevitable or not.

### A. Labeling Data

To produce a labeled training dataset offline, we assume a collection of raw data of the following form: $\mathcal{D}_{\text{raw}} = \{\hat{m}, (q_0, i_0), (q_1, i_1), (q_2, i_2), \ldots, (q_N, i_N)\}$, where $\hat{m}$ is a (perhaps incomplete) geometric map of an environment as produced by our SLAM system having traversed the environment, $q_i$ is a robot configuration within that map, $i_i$ is an image taken from configuration $q_i$, and $N$ is the number of data points.

Our strategy for labeling this dataset will be to cycle through each image-configuration pair, randomly select an action $a_i$ that could have been taken from $q_i$, and use the geometric map $\hat{m}$ to determine whether that action would have resulted in a collision at some point in the future. By associating $i_i$ and $a_i$ with the resulting label, the learned model will be able to predict (without access to the future measurements that contributed to map $\hat{m}$) that next time it encounters a similar image-action pair, a similar outcome will be likely.



(a) "Collision". (b) "Collision". (c) "Non-Collision".

(d) Image for 1(a). (e) Image for 1(b). (f) Image for 1(c).

Fig. 1: Examples of the offline data labeling procedure, with the associated image for each scenario.

We illustrate several example labeling scenarios in Figure 1. In Figure 1(a), the robot is in the middle of the hallway, and the randomly selected action, illustrated in blue, steers toward the right wall at low speed. To determine whether a collision is inevitable after executing this action, we test a range of possible actions that would bring the robot to a stop under maximum deceleration with a variety of steering strategies. In Figure 1(a), all of these so-called "emergency-stop" maneuvers intersect the right wall, and are therefore colored red. Hence, there exists no feasible action that can avoid collision after executing the blue action. Figure 1(d) illustrates the robot's view from this configuration. This image, paired with the blue action, would be assigned a label of "collision".

Figure 1(b) illustrates another scenario in which the robot is approaching a turn at a high speed, with Figure 1(e) illustrating the view from this scenario. Again, all emergency-stop maneuvers are doomed to collide with the far wall, regardless of steering angle. Hence, in this case it is speed that makes collision inevitable. However, Figures 1(c) and 1(f) illustrate a scenario in which the robot is approaching the corner, this time more slowly and farther from the turn. In this case, an emergency-stop maneuver is feasible, and is illustrated in black. This scenario would be labeled "non-collision".

The distribution of "collision" and "non-collision" labels is intended to capture the relationship between the image observed by the robot, the geometry of the environment, and the steering and braking capabilities of the robot with respect to that image and environment geometry. After training on a dataset constructed in this manner, our learned model of collision probability will be able to safely guide a robot through an environment like its training examples.

We also observe that we need not limit ourselves to a single action and label for a given image, and since real images are relatively expensive to obtain, it is very beneficial to reuse each

image to produce numerous labeled data points. Therefore, we pair each image with 50 different actions and their associated labels, resulting in a 50x increase in the amount of available data for the collision predictor network to use for training. We assume that for every image in our dataset, we will have a sufficiently dense sampling of labeled image-action pairs associated with that image to train our collision predictor.

### B. Neural Network

We model collision probability using a fully connected feedforward network, with an image and action as input. We represent this input as a vector concatenating the grayscale pixel intensities of the image with the $x, y$-position of the end point of the action in the robot's body-fixed coordinate frame, and the speed at the end of the action. Therefore, our network input is: $x = [i^0, i^1, i^2, \ldots, i^K, a_x, a_y, a_v]$, where $i^k \in [0, 1]$ is the intensity of the $k^{\text{th}}$ pixel in image $i$, and $a_x$, $a_y$, and $a_v$ represent the terminal position and speed of the action respectively. While it is easy to provide a richer action parameterization, we found no benefit from doing so.

We use three hidden layers (200 nodes per layer) with sigmoid activation functions, and a softmax output layer for probabilistic classification of "collision" vs "non-collision" categories. We train this network using mini-batch stochastic gradient descent to minimize the negative log likelihood of the data labels given the network parameters. We do not use dropout or regularization. While this architecture functioned well for this work, it will be useful to test larger networks and convolutional architectures in future work.

### C. Prior Estimate of Collision Probability

The prior is designed to encourage safety with a minimum of domain knowledge or model-specific information. Our prior predicts that an action carries zero probability of future collision if there exists enough free space ahead of it to come to a stop before intersecting an obstacle or unknown space. If there does not exist enough free space to come to a stop, we assign probability equal to zero for actions that reduce speed, and probability equal to one otherwise. While more sophisticated priors could be designed (or learned), this one serves our purposes and illustrates that the performance of our approach does not hinge on carefully hand-crafted functions.

We define the function $d_{\text{free}}(\hat{m}_t, a_t)$, which returns the length of known free space in map estimate $\hat{m}_t$, along the ray extending from some robot pose along action $a_t$, averaged over a number of equally-spaced poses along the action. We also define the function $d_{\text{stop}}(a_t)$, which returns the stopping distance required by the robot to come to a stop from the speed at the end of action $a_t$. Finally, let $v_0(a_t)$ and $v_F(a_t)$ be the initial and final speeds along action $a_t$, respectively. With these functions, we define our prior as:

$$f_p(c|\hat{m}_t, a_t) = \begin{cases} 0 & \text{if } d_{\text{free}}(\hat{m}_t, a_t) > d_{\text{stop}}(a_t) \\ 0 & \text{if } d_{\text{free}}(\hat{m}_t, a_t) < d_{\text{stop}}(a_t) \text{ and } v_0(a_t) > v_F(a_t) \\ 1 & \text{if } d_{\text{free}}(\hat{m}_t, a_t) < d_{\text{stop}}(a_t) \text{ and } v_0(a_t) < v_F(a_t). \end{cases} \quad (5)$$

## IV. Novelty Detection

To detect novelty, we adopt an autoencoder-based approach proposed by Japkowicz et al. [15], and focus on camera images as the domain of interest, although this general approach is also applicable to a wide variety of complex domains, including acoustic signals [22], network server anomalies [33], data mining [14], document classification [21] and others. Classification of novel data amounts to a one-class classification problem since we assume that we only have access to the set of images that have been observed and collected by our system, and no examples of any other "unfamiliar" images. Therefore, we cannot pose novelty classification as a conventional binary classification problem.

To classify images as novel, we use a feedforward autoencoder with three hidden sigmoid layers and a sigmoid output layer. The autoencoder is trained using mini-batch stochastic gradient descent to reconstruct images by minimizing the sum of squared pixel differences between the input and reconstructed output. Let $i^k \in [0, 1]$ denote the intensity value of the $k^{\text{th}}$ pixel of image $i$. The autoencoder loss function is:

$$L_n(i, \hat{i}) = \frac{1}{K} \sum_{k=1}^{K} (i^k - \hat{i}^k)^2, \quad (6)$$

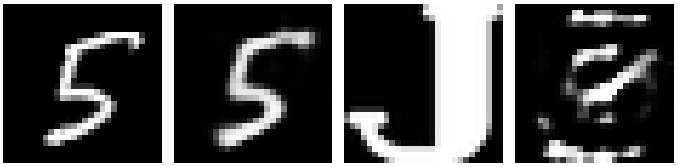
where $\hat{i}$ refers to the output of the autoencoder, i.e., the reconstructed image. The number of pixels in $i$ and $\hat{i}$ is $K$.

This novelty detection approach assumes that the autoencoder network will learn a compressed representation capturing the essential characteristics of its training dataset such that it will reconstruct similar inputs with low reconstruction error. On the other hand, when the autoencoder is presented with a novel input—that is, an input unlike any of its training data—we expect the reconstruction error to be large. Therefore, to classify novel images, we need only to set a threshold and determine whether the reconstruction error for a given image falls above or below that threshold.

Similar to Japkowicz et al. [15], we might consider setting the threshold as some function of the largest reconstruction error of any single training data point. However, since this value would depend only on a single outlier, we adopt a method that we have found to produce more robust and consistent results, which is to compute the empirical CDF of the distribution of losses in the training dataset and select some high percentile, such as the 99$^{\text{th}}$ percentile, to be the threshold. To distinguish between two visually distinct categories of images, usually the autoencoder will separate their distributions of reconstruction error clearly, so the exact value of the threshold is not critical.

### A. MNIST Examples

To illustrate the basic function of the autoencoder novelty-detection method, we trained an autoencoder on the MNIST digit recognition dataset using three hidden sigmoid layers with 50 nodes per layer. Figures 2(a) and 2(b) show an example test input drawn from the MNIST dataset, and its corresponding output, respectively. The reconstruction error for this example is $L_n = 6.9 \times 10^{-3}$, which falls below



(a) Input. (b) Output. (c) Input. (d) Output.

Fig. 2: (a)-(b) Input and reconstruction (output) for an image of a '5' digit, drawn from the training dataset. (c)-(d) Input and reconstruction for an image of a 'J' character, drawn from the notMNIST dataset, exhibiting high reconstruction error, indicating that this input image is considered novel.

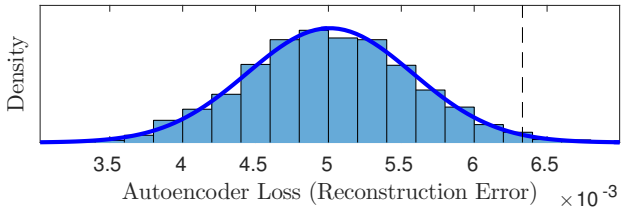


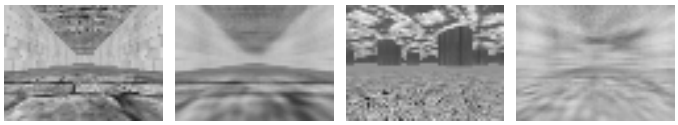

Fig. 3: Autoencoder loss on a training dataset of simulated hallway images with novelty threshold (vertical dashed line).

our computed threshold of $L_n = 1.5 \times 10^{-2}$. On the other hand, Figures 2(c) and 2(d) illustrate an input drawn from the notMNIST dataset [2] of letters from non-standard fonts, and its corresponding output, respectively. In this case, it is clear that the reconstruction does not resemble the input, and that is reflected in the high reconstruction error of $L_n = 3.2 \times 10^{-1}$, which is well above the threshold for novelty. Rather than using a discriminative prediction of class membership from a neural network on this input, we should recognize that we do not have the necessary training data and instead revert to a prior such as a uniform distribution over class labels.

### B. Robot Navigation Examples

To evaluate this novelty detection approach on a visual navigation domain, we trained an autoencoder on a set of simulated camera images from a hallway-type environment and tested it on simulated images of hallways (the training distribution) as well as forests of cylindrical obstacles (the novel test distribution). Again, we used three hidden layers with sigmoid activations, but since these images are more complex than MNIST, we increased the hidden layer size to 200 nodes per layer. For computational efficiency, we use very low resolution 60×44 images in this work, so the input and output layer have dimension 2640. Figure 3 shows a histogram of reconstruction errors for images in the hallway training dataset, along with the PDF of a normal distribution computed from the sample mean and variance of the reconstruction errors. The vertical dashed line illustrates the threshold set at the 99$^{\text{th}}$ percentile of the CDF of this distribution.

Figure 4 illustrates two examples of novelty detection. Figure 4(a) illustrates an input image of a simulated hallway drawn from the distribution of images used to train the autoencoder and 4(b) shows the corresponding reconstruction



(a) Input. (b) Output. (c) Input. (d) Output.

Fig. 4: (a) A familiar input image of a simulated hallway, drawn from the training distribution, and (b) the associated accurate autoencoder reconstruction. (c) A novel input image of a simulated "cylinder forest," *not* drawn from the training distribution, and (d) the resulting inaccurate reconstruction.

with a reconstruction error of $L_n = 4.5 \times 10^{-3}$, which is lower than the threshold of $6.3 \times 10^{-3}$, resulting in a classification of "not novel". On the other hand, Figure 4(c) illustrates an input image of a simulated forest of cylindrical obstacles *not* drawn from the training distribution and 4(d) shows the corresponding reconstruction, which has a reconstruction error of $L_n = 4.4 \times 10^{-2}$, which is much greater than the threshold, resulting in a classification of this image as "novel". It is visually apparent that the reconstruction in Figure 4(b) is much more accurate than that in Figure 4(d), indicating that the "cylinder-forest" image does not follow the same encoding that the autoencoder learned for the hallway images.

### C. Comments on Autoencoder-Based Novelty Detection

The feedforward autoencoder method works well for highly structured data like MNIST digits and robot navigation environments, where we have many similar examples that lie on a fairly low-dimensional manifold of images. However, this method might not be expected to work well for highly varied datasets of unique images like CIFAR-10. Such datasets may require larger convolutional autoencoders, or other techniques to capture the true structure of the data rather than learning a trivial pixel-copy representation. As an alternative to autoencoder-based novelty detection, we might also consider kernel density estimation [4], or density estimation in a feature space learned with a large CNN, as in the content-based image retrieval application proposed by Krizhevsky et al. [16]. However, non-parametric methods would be very computationally expensive since they would require querying a high-dimensional dataset for each prediction.

We also note that our autoencoder is not guaranteed to become familiar with images of a certain appearance at the same rate as the collision predictor network. An image may be classified as "not novel" while the collision predictor is still insufficiently trained for that image. We have tried to prevent this type of failure by matching the size and architecture of the hidden layers of the two networks and have not experienced these types of failures. However, we could also merge the two networks so that they learn a single feature representation both for prediction and novelty classification [28].

While the autoencoder approach has been effective and efficient for our purposes, we believe that novelty detection and uncertainty estimation for neural networks are still open problems, and that other methods may be useful as well.

## V. Results

We tested our approach both in simulation and in experiments using a small autonomous RC car. In both cases, we initialized the system with an empty dataset and a randomly initialized collision prediction network and autoencoder. We initialized the autoencoder threshold to zero so that all images would at first appear novel. We then repeatedly deployed the robot to drive to a specified goal location, collecting images during each run, labeling them, and re-training the collision predictor and autoencoder networks with each addition of data to the growing dataset. Video of our results is available at: http://groups.csail.mit.edu/rrg/videos/safe_visual_navigation.

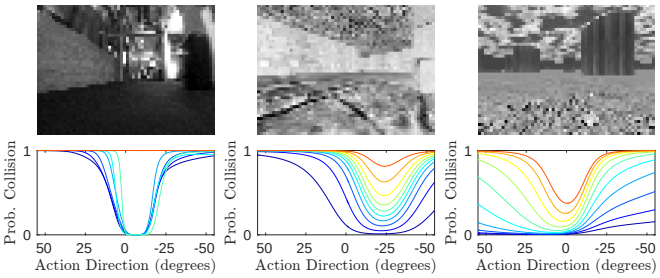
In all simulation trials, we used a $110^\circ$ field-of-view camera model and a 5 m range for geometric sensing. In our experimental trials, we used two different sensor configurations. In one configuration, we used monocular visual-inertial odometry (VIO) based on GTSAM for state estimation, and an Xtion RGBD camera for depth estimation. In the other configuration, we used a Hokuyo UTM-30LX planar LIDAR to perform scan-matching for state estimation, and for geometric perception (artificially limited to a $110^\circ$ field-of-view and 5 m range). In both experimental configurations, we used a Point Grey Chameleon camera with a $110^\circ$ horizontal field-of-view as input to our collision-prediction and novelty-detection models.

While this work is targeted at camera-only perception, the drift and error in VIO prevented us from achieving maximum performance and reliability in the LIDAR-free configuration. Therefore, we used the LIDAR to test the merits of our approach under the idealized circumstances of highly accurate state estimation. We emphasize that in all tests, regardless of the sensor used for state estimation and mapping, collision prediction was performed only on camera images, combined with the action description, as described above.

### A. Visualizing Neural Network Predictions

Figure 5 shows the predictions of collision probability on one real camera image from an experimental run and two simulated images. In each case, the networks were trained with a large dataset of relevant images, so novelty is not a factor. The graph below each image illustrates the collision probability as a function of the steering direction of the action (i.e., atan2$(a_y, a_x)$), shown on the horizontal axis, and the speed at the end of the action, indicated by color. Speeds vary from 1 m/s (blue) to 10 m/s (red), illustrating that faster actions carry a greater probability of resulting in future collision.

Figure 5(a) illustrates a view roughly aligned with the axis of a real hallway, and therefore the learned model assigns low probability of collision to actions aligned with that axis. Figure 5(b) illustrates the robot in a right-hand turn in a simulated hallway, so the probability model assigns lowest probability of collision to actions that bear $25^\circ$ to the right. Finally, Figure 5(c) illustrates the robot in a simulated forest of obstacles. Since there is a long gap between obstacles down the center of the image, the predicted collision probability is lowest at this point, but rises sharply for turns to the right, due to an obstacle in the right foreground of the image.




(a) Real hallway. (b) Simulated hallway. (c) Simulated forest.

Fig. 5: Illustration of predicted collision probabilities. Each image is illustrated with the learned collision probability model below it, with the horizontal axis representing action direction and color representing speed, from 1 m/s to 10 m/s.

### B. Simulation Results

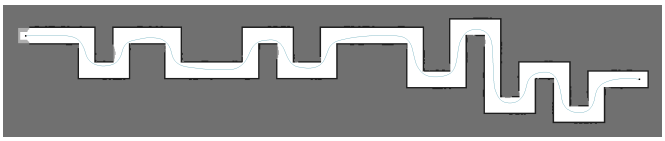
We tested our system in simulation with a run through each of eight random hallway-type environments followed by eight random "cylinder-forest"-type environments, re-training after each run. Examples of both environment types are shown in Figure 6. Each run contributed about 1250 images to the dataset, so that after 16 trials, the system had collected 20000 images in total. For reporting performance statistics, we performed 10 separate testing runs for each amount of training data. All simulated training and testing trials were completed without collision, demonstrating the safety of our approach.

Figure 7 illustrates the performance of our simulated system as a function of the number of training images observed. The first eight data points (blue) represent hallway environments, while the second eight (red) represent "cylinder-forest" environments. As the system gained more training data from hallway-type environments, it began to trust the learned model of collision probability on a greater fraction of observed images, resulting in faster average speed of navigation. As the novelty fraction decreased monotonically, the average speed of navigation increased. In total, the average speed increased 42% from 4.90 m/s to 6.97 m/s in the hallway environments.

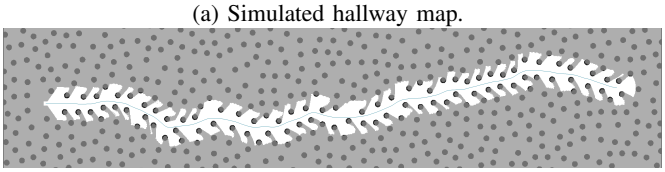
After completing trials in the hallway environments, we tested our system in "cylinder-forest" environments, carrying over the complete dataset and trained models from the final hallway test. Initially, all images were again classified as novel since the forest environment type is visually distinct from the hallway type. However, as our system accumulated images from the forest environments, the same pattern emerged: Monotonic decrease in novelty with corresponding increase in speed. In forest environments, the speed increased 50% from 5.84 m/s to 8.80 m/s. The reason forest navigation was faster than hallway navigation, both before and after training, is that there is on average more free space ahead of the robot in forest environments and no sharp turns are required.

### C. Experimental Results using LIDAR

We performed a total of 80 experimental trials using a small autonomous RC car in much the same way that we performed our simulation trials, except that instead of re-training after



(a) Simulated hallway map.



(b) Simulated forest map.

Fig. 6: Maps of hallway (top) and "cylinder-forest" (bottom) environments built by range-limited SLAM during navigation.

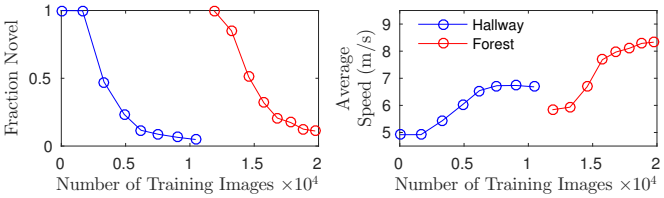
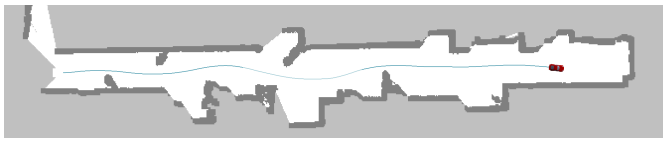



Fig. 7: Simulation results. Perceived novelty of images encountered online, and average navigation speed, as a function of the number of training images. Blue indicates hallway environments and red indicates "cylinder-forest" environments.

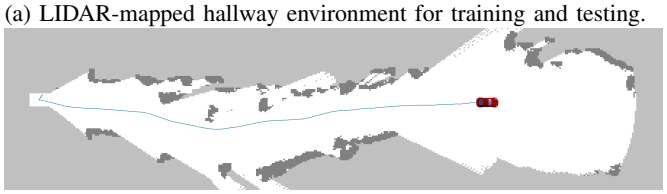
each run, we instead re-trained after every 10 runs through the environment since the experimental environment was smaller and yielded fewer images per run. A generated map of the experimental environment is shown in Figure 8(a) and an image from this environment is shown in Figure 10(a).

We observed essentially the same qualitative results in our experiments as we did in simulation. As the system collected more training images, the fraction of images considered novel decreased, resulting in a corresponding increase in speed of navigation, as illustrated in Figure 9. We observed an average speed increase of 25%, from 3.38 m/s to 4.24 m/s. However, since the experimental environment was small compared to the simulation environment, a substantial portion of the navigation time was spent accelerating from the start and decelerating at the goal. Therefore, we also provide the average maximum speeds observed for each amount of training data, where there was a much more substantial improvement of 55%, from 3.79 m/s to 5.90 m/s. This improvement in maximum speed implies that if the experimental environment were larger, we would observe a greater improvement in average speed as well.

Figure 11 illustrates the autoencoder reconstruction error for three categories of data after training on nearly 9000 training images from numerous runs through the experimental environment depicted in Figure 8(a) and 10(a). The blue histogram and PDF indicates the reconstruction error of the training images, from which the threshold (dashed line) was computed. The red histogram and PDF represents the reconstruction error of approximately 1000 images collected during testing runs in the same environment, but not used for training the autoencoder. While the reconstruction error of these images is



(a) LIDAR-mapped hallway environment for training and testing.



(b) Xtion-mapped novel environment used for LIDAR-free testing.

Fig. 8: Maps of experimental environments.

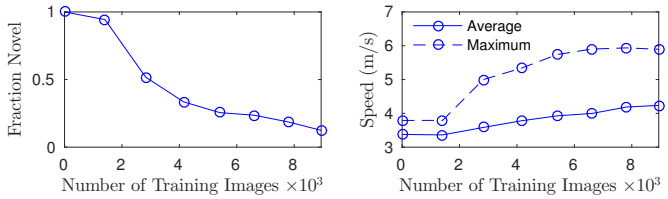
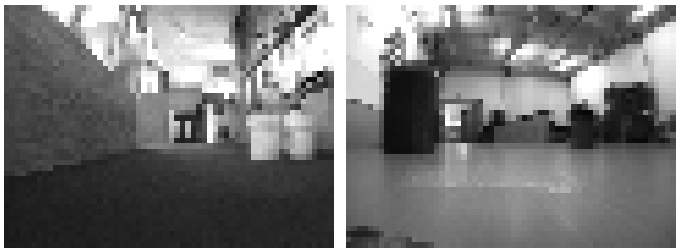



Fig. 9: Experimental results for 80 experimental trials. Each data point represents the average over 10 trials with a given model between re-training episodes.

not as low as the training images, most fall below the novelty threshold. Finally, the black histogram and PDF illustrate the reconstruction error of a truly novel environment—a well-lit lab area with obstacles (Figures 8(b) and 10(b)). Visually, this environment is quite different from the training environment, and that is reflected in the much greater reconstruction error, with all images falling above the novelty threshold.

### *D. Experimental Results using Monocular VIO and RGBD*

We performed experiments on our autonomous RC car using a LIDAR-free sensor configuration in order to demonstrate steps toward a completely vision-based learning and navigation pipeline. However, as noted previously, inaccuracies in the monocular VIO state estimate made this sensor configuration unsuitable for demonstrating the entire training and testing procedure. Nevertheless, we were able to test the LIDAR-free sensor configuration in both environments depicted in Figures 8 and 10 using the collision-prediction and autoencoder



(a) Familiar Environment. (b) Novel Environment.

Fig. 10: Images from (a) the hallway in which training took place, and (b) a novel environment not used in training.

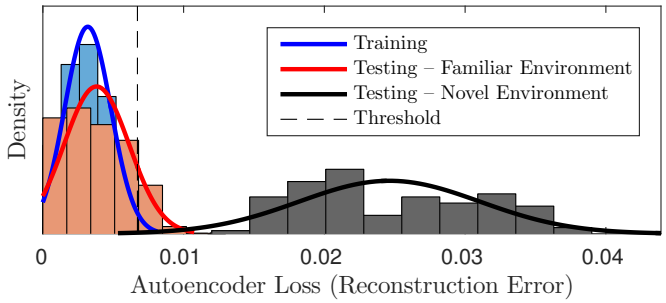



Fig. 11: Histogram of autoencoder loss for experimental data.

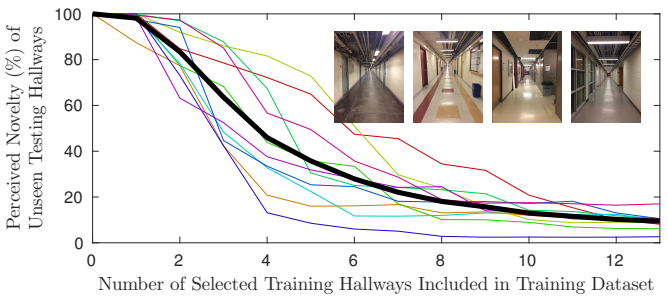



Fig. 12: Perceived novelty of unseen testing hallways as a function of the number of hallways used for training. Four example hallways are pictured in the upper right.

models previously trained with the other sensor configuration.

In the hallway training environment, we achieved a mean speed of 3.26 m/s and a top speed over 5.03 m/s. This result significantly exceeds the maximum speeds achieved when driving in this environment under the prior estimate of collision probability before performing any learning. On the other hand, in the novel environment, for which our model was untrained, the novelty detector correctly identified every image as being unfamiliar. In the novel environment, we achieved a mean speed of 2.49 m/s and a maximum speed of 3.17 m/s.

We emphasize that these results represent an improvement of 31% in mean speed and 59% in maximum speed after learning, for monocular vision-based navigation using a neural network. Naturally, the robot navigated more slowly in the novel environment than it did in the familiar one, but our method enables the robot to navigate in novel environments safely, potentially gathering data and re-training, whereas that would not be feasible without novelty-detection.

### *E. Autoencoder Validation*

To characterize the performance of our autoencoder novelty detection approach on a wider variety of real environments, we tested it on 16 visually distinct hallways at MIT. We collected approximately 5000 images from each hallway and trained autoencoders on the images from between 1 and 13 different hallways, while testing on three other hallways reserved for testing. We then repeated this process for ten different permutations of which hallways were used for training and which were used for testing. The results are shown in Figure 12, with each colored line representing a particular permutation, and the black line indicating the average.

These results show that after training on only a single hallway, the three test hallways were considered to be novel, indicating that a single hallway does not contain enough variation to capture the characteristics of hallways in general. However, after training on 13 hallways, our models classified 90.6% of test hallway images as "not novel", while correctly classifying 91.5% of test images from an outdoor courtyard and 94% of test images from a parking garage as "novel". These results show that the autoencoder method of novelty detection is able to meaningfully generalize to a family of images with common structure, rather than merely memorizing or overfitting to a particular training dataset. See Richter [30] for additional validation examples and discussion.

## VI. Related Work

There has been substantial work applying neural network learning to image-based navigation. For instance, the DARPA LAGR program featured long-range terrain classification using a neural network, in an similar role to our collision prediction approach [13]. Barnes et al. [3] and Oliveira et al. [24] provide more recent examples of deep learning used to segment image pixels into drivable routes, which can then be used for planning. While we focus on supervised learning of collision probability, there are many other ways to map between images and actions, including end-to-end neural networks [6, 19], affordance-based representations [7], reinforcement learning using either simulated or real images [23, 31], and classification of paths from manually collected data [11]. While it would be impossible to survey the vast literature on visual navigation here, we note that these examples and many others do not explicitly address uncertainty about the learned model.

In our prior work, we considered model uncertainty in collision prediction, but focused on features of geometric maps, rather than images and neural networks [29]. Grimmett et al. [12] suggest that if a classification error is to be made in robotic decision making, it should be made with high reported uncertainty so that the robot can avoid consequences of a wrong decision. Sofman et al. [32] use online novelty detection in this way, to avoid scenarios for which their robot is untrained. In contrast, our use of novelty detection allows the robot to proceed safely under the prior, gather its own training data, and continually improve.

## VII. Conclusion

In this work, we have demonstrated a safe, self-supervised method of learning a visual collision prediction model online. We have shown that by using an autoencoder as a measure of uncertainty in our collision prediction network, we can transition intelligently between the high performance of the learned model and the safe, conservative performance of a simple prior, depending on whether the system has been trained on the relevant data. In future work, it will be valuable to train and test on a much wider variety of environments, using large convolutional network architectures, to scale up our approach and examine the generalization of our models.

## References


[1] Microsoft kinect sensor. https://msdn.microsoft.com/en-us/library/hh438998.aspx. Accessed: 2016-12-29.

[2] notMNIST dataset. http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html. Accessed: 2017-01-26.

[3] D. Barnes, W. Maddern and I. Posner. Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy. *arXiv:1610.01238*, 2016.

[4] C. M. Bishop. Novelty detection and neural network validation. *IEE Proceedings - Vision, Image and Signal Processing*, 141(4):217–222, 1994.

[5] C. Blundell et al. Weight uncertainty in neural networks. *arXiv:1505.05424*, 2015.

[6] M. Bojarski et al. End to end learning for self-driving cars. *arXiv:1604.07316*, 2016.

[7] C. Chen et al. DeepDriving: Learning affordance for direct perception in autonomous driving. In *Proc. International Conference on Computer Vision (ICCV)*, 2015.

[8] J. Franke and M. H. Neumann. Bootstrapping neural networks. *Neural Computation*, 12(8):1929–1949, 2000.

[9] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proc. International Conference on Machine Learning (ICML)*, 2015.

[10] Y. Gal. Uncertainty in deep learning. Ph.D. Thesis, University of Cambridge, 2016.

[11] A. Giusti et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.

[12] H. Grimmett, R. Triebel, R. Paul and I. Posner. Introspective classification for robot perception. *International Journal of Robotics Research*, 37(7):743–762, 2016.

[13] R. Hadsell et al. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

[14] S. Hawkins et al. Outlier detection using replicator neural networks. In *Proc. International Conference on Data Warehousing and Knowledge Discovery*, 2002.

[15] N. Japkowicz, C. Myers and M. Gluck. A novelty detection approach to classification. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[16] A. Krizhevsky, I. Sutskever and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[17] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems (NIPS)*, 1995.

[18] B. Lakshminarayanan, A. Pritzel and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv:1612.01474*, 2016.

[19] Y. LeCun et al. Off-road obstacle avoidance through end-to-end learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[20] D. J. C. MacKay. Probable networks and plausible predictions—a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505, 1995.

[21] L. Manevitz and M. Yousef. One-class document classification via neural networks. *Neurocomputing*, 70(7-9):1466–1481, 2007.

[22] E. Marchi et al. A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[23] J. Michels, A. Saxena and A. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proc. International Conference on Machine Learning (ICML)*, 2005.

[24] G. L. Oliveira, W. Burgard and T. Brox. Efficient deep models for monocular road segmentation. In *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[25] I. Osband et al. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[26] G. Paass. Assessing and improving neural network predictions by the bootstrap algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, 1993.

[27] D. A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems (NIPS)*, 1989.

[28] D. A. Pomerleau. Input reconstruction reliability estimation. In *Advances in Neural Information Processing Systems (NIPS)*, 1993.

[29] C. Richter, W. Vega-Brown, and N. Roy. Bayesian learning for safe high-speed navigation in unknown environments. In *Proc. International Symposium on Robotics Research (ISRR)*, 2015.

[30] C. Richter. Autonomous navigation in unknown environments using machine learning. Ph.D. Thesis, Massachusetts Institute of Technology, 2017.

[31] F. Sadeghi and S. Levine. $(CAD)^2RL$: Real single-image flight without a single real image. *arXiv:1611.04201*, 2016.

[32] B. Sofman et al. Anytime online novelty and change detection for mobile robots. *Journal of Field Robotics*, 28(4):589–618, 2011.

[33] B. B. Thompson et al. Implicit learning in autoencoder novelty assessment. In *Proc. International Joint Conference on Neural Networks*, 2002.

[34] S. Thrun et al. Stanley: The robot that won the DARPA grand challenge. *Journal of Field Robotics*, 23(9):661-692, 2006.