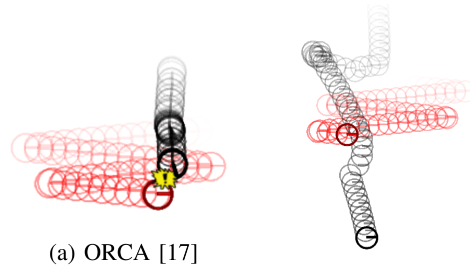
2019 American Control Conference (ACC)
Philadelphia, PA, USA, July 10-12, 2019

# Toward Verifiable Real-Time Obstacle Motion Prediction for Dynamic Collision Avoidance

Vince Kurtz and Hai Lin

*Abstract*—**Next generation Unmanned Aerial Vehicles (UAVs) must reliably avoid moving obstacles. Existing dynamic collision avoidance methods are effective where obstacle trajectories are linear or known, but such restrictions are not accurate to many real-world UAV applications. We propose an efficient method of predicting an obstacle's motion based only on recent observations, via online training of an LSTM neural network. Given such predictions, we define a Nonlinear Probabilistic Velocity Obstacle (NPVO), which can be used select a velocity that is collision free with a given probability. We take a step towards formal verification of our approach, using statistical model checking to approximate the probability that our system will mispredict an obstacle's motion. Given such a probability, we prove upper bounds on the probability of collision in multi-agent and reciprocal collision avoidance scenarios. Furthermore, we demonstrate in simulation that our method avoids collisions where state-of-the-art methods fail.**



(a) ORCA [17]

(b) Ours

Fig. 1: An obstacle (red) moves back and forth while slowly drifting downwards. Existing methods fail to avoid such an obstacle, while our system accurately predicts its motion and avoids a collision.

## I. Introduction and Related Work

To enable future UAV applications like disaster response, infrastructure inspection, and package delivery, autonomous UAVs must avoid collisions with moving obstacles. The motion of moving obstacles may not be known a priori, and they will not reliably continue at their current velocity. However there is often some structure to each obstacle's movement such that we might predict future motion by observing past behavior. A simple example is shown in Figure 1.

We propose a novel algorithm to predict the motion of moving obstacles via online training of an LSTM Recurrent Neural Network (RNN) [6], using dropout to obtain uncertainty estimates over these predictions [3, 4]. To our knowledge, this is the first work proposing an online obstacle motion prediction system for collision avoidance without a priori environmental knowledge or extensive offline training.

Hug et al [9] show that LSTM neural networks can predict pedestrian paths from extensive training data. They note that LSTM is well-suited to time-varying patterns with both short and long-term dependencies. We extend this work by learning patterns of obstacle motion online rather than from a large training set. Furthermore, we use dropout sampling to approximate a probability distribution such that the variance of this distribution can be used as an uncertainty estimate [11].

Given probabilistic predictions of obstacle movement, we propose an uncertainty-aware multi-agent dynamic collision avoidance algorithm based on Nonlinear Probabilistic Velocity Obstacles (NPVO), a novel extension of existing velocity obstacle notions. These include Probabilistic Velocity Obstacles, which provide an uncertainty-aware policy in static environments [2], and Nonlinear Velocity Obstacles [15], which can guarantee collision avoidance for obstacles moving along known trajectories.

The state-of-the-art "Optimal Reciprocal Collision Avoidance" (ORCA) algorithm uses reciprocal velocity obstacles to control multiple agents in unstructured environments [17]. This approach is popular due to its ease of implementation and guarantees that a collision-free trajectory will be found for $\tau$ time, if one is available. However, ORCA assumes that all obstacles in the workspace are either static or operating according to the same policy. Violations of this assumption can lead to catastrophic behavior, as shown in Figure 1a. We demonstrate in simulation that our NPVO approach is able to avoid such obstacles.

Finally, we take an important step towards rigorous verification of our framework. Existing results for formal verification of systems based on techniques like LSTM are highly limited [1, 7, 12], but formal guarantees are of vital importance for safety-critical applications like collision avoidance. We propose a novel statistical model checking formulation to approximate the probability that an obstacle will remain within certain bounds. Along the way, we demonstrate that predictions generated by our algorithm are robust to perception uncertainty in the form of additive Gaussian noise.

Given the probability that obstacles will remain within the bounds of our predictions (which we approximate with statistical model checking), we prove several important safety

The partial support of the National Science Foundation (Grant No. ECCS-1253488, IIS-1724070, CNS-1830335) and of the Army Research Laboratory (Grant No. W911NF- 17-1-0072) is gratefully acknowledged

Vince Kurtz and Hai Lin are with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 46556 USA. vkurtz@nd.edu, hlin1@nd.edu

properties of our algorithms. Namely, we prove bounds on the probability of collision for one agent avoiding $N$ obstacles and of any collision occurring between $N$ agents using our proposed approach for reciprocal collision avoidance.

## II. Problem Formulation

Consider a robotic agent $A$ with single-integrator dynamics and perfect actuation. That is, the position of agent $A$ at instant $k$, $\mathbf{p}_k^A$, is governed by

$$\mathbf{p}_{k+1}^A = \mathbf{p}_k^A + \mathbf{v}_k \Delta t \tag{1}$$

where $\mathbf{v}$ is a velocity (control input), and $\Delta t$ is a sampling period. Assume there exists some desired velocity $\mathbf{v}_{des}$, provided a priori or by a higher-level planner.

Suppose that agent $A$ operates in a workspace with $N$ dynamic obstacles $B_1, B_2, \ldots, B_N$. Each $B_i$ moves according to an unknown and possibly time-varying policy:

$$\mathbf{p}_{k+1}^{B_i} = f_i(\mathbf{p}_k^{B_i}, k, \mathbf{u}_k^{B_i}) \tag{2}$$

where $\mathbf{p}_k^{B_i}$ denotes the position of obstacle $B_i$ at instant $k$, and $\mathbf{u}_k^{B_i}$ denotes some unknown control input. We assume perfect observability of the position of all $B_i$ for the last $n$ timesteps. That is, at instant $n$,

$$\mathcal{O} = \{\mathbf{p}_0^{B_i}, \mathbf{p}_1^{B_i}, ..., \mathbf{p}_n^{B_i}\}$$

is known to agent $A$.

Finally, assume that there exists some safe distance $r_s$ such that for any obstacle $B_i$, if

$$\|\mathbf{p}_k^A - \mathbf{p}_k^{B_i}\|_{\mathcal{L}_2} \geq r_s,$$

then $A$ and will not collide with $B_i$ at instant $k$.

The multi-agent dynamic collision avoidance problem can then be stated as follows:

**Problem 1.** *Multi-Agent Dynamic Collision Avoidance*

Given a preferred velocity $\mathbf{v}_{des}$ and observations $\mathcal{O} = \{\mathbf{p}_0^{B_i}, \mathbf{p}_1^{B_i}, ..., \mathbf{p}_n^{B_i}\}_{i=1}^N$, find a safe velocity $\mathbf{v}_{safe}$ that minimizes the probability of collision for the next $m$ timesteps,

$$\mathbb{P}(\|\mathbf{p}_{n+k}^A - \mathbf{p}_{n+k}^{B_i}\| < r_s \text{ for any } k \in [1, m], i \in [1, N]). \tag{3}$$

## III. Online Prediction of Obstacle Motion

To solve Problem 1, we first propose an algorithm to predict the motion of moving obstacles based on past observations. Our algorithm provides predictions as *probability distributions* over future obstacle positions, which will later allow us to estimate and minimize the probability of collision (3). In this section, we consider the motion of a single obstacle $B_i$. For simplicity, we denote this obstacle's position at instant $k$ as $\mathbf{p}_k^{B_i} = \mathbf{p}_k$. To predict the motion of several obstacles, separate instances of the algorithms described in this section can be used.

Given a set of observations $\mathcal{O} = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$, we approximate equation 2 by

$$\mathbf{p}_{n+1}^{B_i} = g(\mathcal{O}) \tag{4}$$

and approximate $g(\mathcal{O})$ online using an LSTM network [6].

First, we calculate a training set of known inputs and outputs. Observations are transformed from positions to changes in position, so that the input to the network is invariant to shifts in the workspace. That is, we calculate $\hat{\mathcal{O}} = \{\Delta\mathbf{p}_1, \Delta\mathbf{p}_2, ..., \Delta\mathbf{p}_n\}$ where $\Delta\mathbf{p}_k = \mathbf{p}_k - \mathbf{p}_{k-1}$. We then consider the first $k$ changes in position as an input datapoint and the subsequent $m$ changes in position to be the corresponding output. To model perception uncertainty, we perturb each $\Delta\mathbf{p}$ with zero-mean Gaussian noise with variance $\sigma^2$. Changing $\sigma^2$ will allow us to verify the robustness of our system to perception uncertainty (see Section V). The complete training dataset at timestep $n$, $d_n = (\mathbf{X}, \mathbf{Y})$, is then specified as follows:

$$\mathbf{X} = \{\mathbf{x}_k\}_{k=1}^{n-m}, \quad \mathbf{Y} = \{\mathbf{y}_k\}_{k=1}^{n-m}$$
$$\mathbf{x}_k = \{\Delta\mathbf{p}_i + \epsilon_i\}_{i=1}^k, \quad \epsilon_i \sim G(0, \sigma^2)$$
$$\mathbf{y}_k = \{\Delta\mathbf{p}_i + \epsilon_i\}_{i=k+1}^{k+m}, \quad \epsilon_i \sim G(0, \sigma^2).$$

For a given input $\mathbf{x}_k$, we denote the output of the neural network parameterized by weights $\mathcal{W}$ as $\hat{\mathbf{y}}_k = \mathcal{NN}(\mathbf{x}_k; \mathcal{W})$. We then find weights to minimize minimize the cost function $C_k(\mathcal{W}) = L_\delta[\mathbf{y}_k - \hat{\mathbf{y}}_k]$ using a fixed number of iterations ($N_{iter}$) of the stochastic gradient descent algorithm Adam [13]. $L_\delta[\cdot]$ indicates the Huber norm [8] with parameter $\delta$. This process is summarized in Algorithm 1.

**Algorithm 1** TrainNetworkOnline

```
procedure TrainNetwork({Δp_1, ..., Δp_n})
    for k = [1, n − m] do
        x_k = {Δp_i + ε_i}_{i=1}^{k},    ε_i ∼ G(0, σ^2)
        y_k = {Δp_i + ε_i}_{i=k+1}^{k+m},    ε_i ∼ G(0, σ^2)
        ŷ_k = NN(x_k; W)
    W* = arg min Σ_k C_k(W)
    return NN(·; W*)
```

The second part of our online learning approach is prediction, which is outlined in Algorithm 2. We consider the whole history of observed position changes, perturbed by perception noise, as input to a neural network with weights $\mathcal{W}$:

$$\mathbf{x} = \{\Delta\mathbf{p}_i + \epsilon_i\}_{i=1}^n \quad \epsilon_i \sim G(0, \sigma^2).$$

Applying dropout as per [4], we can treat each output of the network as a prediction of changes in position for the next $m$ timesteps:

$$\{\Delta\hat{\mathbf{p}}_{n+1}, \Delta\hat{\mathbf{p}}_{n+2}, \ldots, \Delta\hat{\mathbf{p}}_{n+m}\} = \mathcal{NN}(\mathbf{x}).$$

With repeated application of dropout, we can construct a set of predictions

$$\hat{Y} = \{\Delta\hat{\mathbf{p}}_{n+1}^i, \Delta\hat{\mathbf{p}}_{n+2}^i, \ldots, \Delta\hat{\mathbf{p}}_{n+m}^i\}_{i=1}^{N_s}.$$

Note that $\hat{Y}$ contains $N_s$ predictions for each timestep, $\{\Delta\hat{\mathbf{p}}_{n+k}^i\}_{i=1}^{N_s}$. We consider these predictions to be samples from an underlying multivariate Gaussian distribution $G(\mu_k, \Sigma_k)$. This distribution represents the probability that a certain motion, $\Delta\mathbf{p}_k$, will be taken by the obstacle at instant $k$.

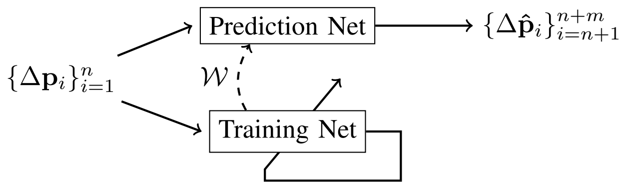



Fig. 2: Two copies of the LSTM network are used in parallel for online prediction.

Given sample predictions $\{\Delta\hat{\mathbf{p}}_{n+k}^i\}_{i=1}^{N_s}$, we calculate the maximum likelihood estimates of $\mu_k$ and $\Sigma_k$. This allows us to estimate the position of the obstacle at instant $n+k$ as

$$\hat{\mathbf{p}}_{n+k} = \sum_{i=n+1}^{n+k} \mu_i + \mathbf{p}_n.$$

We can then write a new set of distributions that reflect the position of the obstacle in the future. Writing the position of the obstacle at time $n+k$ as a random variable $\mathbf{P}_{n+k}$, we have

$$\mathbf{P}_{n+k} \sim G(\hat{\mathbf{p}}_{n+k}, \Sigma_k).$$

Then, given some threshold $\gamma$, we construct ellipsoids $e_k$ in position space such that

$$\mathbb{P}(\mathbf{P}_{n+k} \in e_k) \geq \gamma \quad \forall k \in [1, m].$$

These ellipsoids $\{e_k\}_{k=1}^m$ will be used later to construct an NPVO (see Section IV, Algorithm 3).

**Algorithm 2** PredictObstacleMotion

**procedure** PREDICTMOTION($\{\Delta\mathbf{p}_i\}_{i=1}^n, \mathcal{NN}, \gamma$)
  $\mathbf{x} = \{\Delta\mathbf{p}_i + \epsilon_i\}_{i=1}^n, \quad \epsilon_i \sim G(0, \sigma)$
  $\hat{Y} = \{\}$, $ellipoids = \{\}$
  **for** $j = [1, N_s]$ **do**
    $\{\Delta\hat{\mathbf{p}}_i\}_{i=n+1}^{n+m} = \mathcal{NN}(\mathbf{x})$ ▷ with dropout
    $\hat{Y} \leftarrow \{\Delta\hat{\mathbf{p}}_i\}_{i=n+1}^{n+m}$
  **for** $k = [1, m]$ **do**
    $\mu_k, \Sigma_k = MLE(\{\Delta\mathbf{p}_{n+k}^i\}_{i=1}^{N_s})$
    $\hat{\mathbf{p}}_k = \mathbf{p}_n + \sum_{i=n+1}^k \mu_i$
    $e_k = \{\mathbf{p} \mid \mathbb{P}(\mathbf{p} \in e_k) \geq \gamma\}$
    $ellipsoids \leftarrow e_k$
  **return** ellipsoids

To achieve real-time online training and prediction, we use the multithreading approach shown in Figure 2. The reason for this is that Algorithm 1 is not guaranteed to terminate within $\Delta t$ time, in which case it is important to be able to still make predictions. Two identical copies of the network, one for training and one for prediction, are created. Algorithm 1 determines weights of the training network. These weights are then copied to the prediction network and used in Algorithm 2. The only difference between the two networks is the dropout mask used: the training network always uses different masks at each timestep for regularization, while the prediction network uses the same mask at each timestep to approximate a Bayesian network [4].

## IV. Nonlinear Probabilistic Velocity Obstacles

In this section, we shown how the predictions generated in Section III can be used for collision avoidance. In doing so, we propose a new velocity obstacle concept, the NPVO. The traditional velocity obstacle is defined as follows:

**Definition IV.1.** *Velocity Obstacle. [17]* The Velocity Obstacle for agent $A$ induced by agent $B$ for time window $\tau$ is the set of all velocities of $A$ that will result in a collision between $A$ and $B$ within $\tau$ time:

$$VO_{A|B}^{\tau} = \{\mathbf{v} \mid \exists t \in [0, \tau] \;::\; \mathbf{v}t \in D(\mathbf{p}_B - \mathbf{p}_A, r_s)\}$$

where $D(\mathbf{p}, r)$ denotes a disk of radius $r$ centered at position $\mathbf{p}$, $\mathbf{p}_A$ ($\mathbf{p}_B$) is the position of agent $A$ ($B$), and $r_s$ is the minimum safe distance between agents.

Our NPVO is a straightforward extension of this notion, under the assumption that we can obtain some probabilistic estimate of the future position of a moving obstacle.

**Definition IV.2.** *Nonlinear Probabilistic Velocity Obstacle.* Assume the position of obstacle $B$ at timestep $k$ is estimated by the random variable $\mathbf{P}_k^B \sim \mathcal{F}_k$. The NPVO for agent $A$ induced by obstacle $B$ for $m$ timesteps with probability $\gamma$ is then given by

$$NPVO_{A|B}^{m,\gamma} =$$
$$\{\mathbf{v} \mid \exists k \in [0, m] \;::\; \mathbb{P}(\mathbf{v}k \in D(\mathbf{p}^A - \mathbf{P}_k^B, r_s)) > \gamma\}$$

where $D(\mathbf{p}, r)$ denotes a disk of radius $r$ centered at position $\mathbf{p}$, $\mathbf{p}^A$ is the current position of agent $A$, and $r_s$ is the minimum safe distance between agents.

Given a nonlinear probabilistic velocity obstacle and a desired velocity, we can specify a safe velocity via the constrained optimization problem

$$\arg\min(\|\mathbf{v}_{des} - \mathbf{v}_{safe}\|_{\mathcal{L}_2}^2),$$
$$\mathbf{v}_{safe} \notin NPVO_{A|B}^{\gamma,m},$$

where $\mathbf{v}_{des}$ is a desired velocity. $\mathbf{v}_{safe}$ is guaranteed to avoid a collision for the next $m$ timesteps with a given probability, as long as there exists a dynamically feasible $\mathbf{v} \notin NPVO_{A|B}^{\gamma,m}$.

The NPVO concept can be easily applied to achieve collision avoidance with the prediction system outlined in Section III, as shown in Algorithm 3.

**Algorithm 3** NPVO Collision Avoidance

**procedure** COLLAVOID($\mathbf{p}_n$, $\{\Delta\mathbf{p}_i\}_{i=1}^n$, $\gamma$, $\mathbf{v}_{des}$, $goal$)
  **while** $goal$ not reached **do**
    $\mathcal{NN} = $ TRAINNETWORK($\{\Delta\mathbf{p}_1, ..., \Delta\mathbf{p}_n\}$)
    $\{e_k\}_{k=1}^m = $ PREDICTMOTION($\{\Delta\mathbf{p}_i\}_{i=1}^n, \mathcal{NN}, \gamma$)
    $NPVO_{A|B}^{\gamma,m} =$
    $\{\mathbf{v} \mid \exists k \in [1, m] \;::\; \mathbf{v}k + r \in e_k \forall r \text{ s.t.} |r| \leq r_{safe}\}$
    $\mathbf{v}_{safe} = \arg\min(\|\mathbf{v}_{des} - \mathbf{v}_{safe}\|^2)$
      s.t. $\mathbf{v}_{safe} \notin NPVO_{A|B}^{\gamma,m}$
    apply $\mathbf{v}_{safe}$

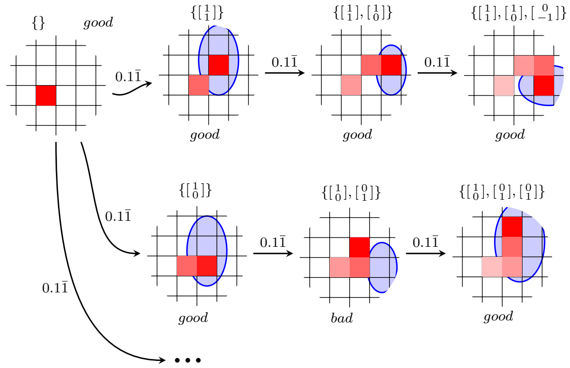



Fig. 3: An example Markov Chain used to verify our prediction system. Red shaded squares indicate the path of a (hypothetical) obstacle. The state $s = \{\Delta \mathbf{p}_1, ..., \Delta \mathbf{p}_n\}$ is indicated above each node. Transitions are drawn uniformly from a 9-cell grid (not all are shown). Predictions are indicated by blue ellipses: a state is *good* if the latest position is within the last prediction.

The notion of an NPVO can be easily extended to the multi-agent case, allowing us to use our prediction paradigm for multi-agent collision avoidance.

**Definition IV.3.** *Multi-Agent NPVO*

Assume the positions of $N$ obstacles $\{B_i\}_{i=1}^N$ at timestep $k$ can be estimated by the random variables $\mathbf{P}_{B_i} \sim \mathcal{F}_k^i$. The NPVO for agent $A$ induced by $\{B_i\}_{i=1}^N$ for $m$ timesteps with probability $\gamma$ is then given by

$$NPVO^{m,\gamma}_{A|\{B_1,B_2,...,B_N\}} =$$

$$\{\mathbf{v} \mid \exists k \in [0,m] \;\; :: \;\; \mathbb{P}(\mathbf{v}k \in D(\mathbf{p}^A - \mathbf{P}_k^{B_i}, r_s)) \geq \gamma \quad \text{for any } i \in [1,N]\}$$

where $D(\mathbf{p}, r)$ denotes a disk of radius $r$ centered at position $\mathbf{p}$, $\mathbf{p}^A$ is current the position of agent $A$, and $r_s$ is the minimum safe distance between $A$ and any agent $B_i$.

## V. Statistical Model Checking

Rigorously verifying that systems based on neural networks satisfy certain specifications, even probabilistically, is a difficult and open problem [1, 7, 12]. In safety-critical applications like dynamic collision avoidance, however, it is important to provide some guarantees of safety and performance. In this section, we demonstrate a novel means of providing approximate probabilistic guarantees for our prediction system.

While it might be tempting to reason about theoretical guarantees directly from the probability distribution estimated from dropout, it is well established that dropout provides only a lower bound on model uncertainty [3, 11]. Therefore, we use *statistical model checking* to obtain more rigorous results. Specifically, we are interested in estimating the probability that the an obstacle $B$ will remain within prediction ellipsoids $e_k^B$ throughout in the next $m$ timesteps:

$$\mathbb{P}(\mathbf{p}^B_{n+k} \in e_k^B \;\forall\; k \in [1,m]). \tag{5}$$

As shown in Section VI, estimating this probability will allow us to prove important properties of Algorithm 3..

The general problem of statistical model checking is defined as follows:

**Definition V.1.** Statistical Model Checking

Given a model $\mathcal{M}$ and a property $\phi$, determine if $\mathbb{P}(\mathcal{M} \vDash \phi) \geq \theta$, where $\theta$ is a desired performance threshold.

Typically, $\mathcal{M}$ is a stochastic model such as a Markov Chain (MC), and $\phi$ is encoded in a logic such as Probabilistic Computation Tree Logic (PCTL) [5]. Assuming that $\phi$ can be determined on finite executions of $\mathcal{M}$, we define $B_i \sim Bernoulli(p)$ such that $b_i = 1$ if the $i^{th}$ execution of $\mathcal{M}$ satisfies $\phi$. Model checking is then reduced to choosing between hypothesis $H_0 : p \geq \theta + \delta$ and $H_1 : p < \theta - \delta$ where $\delta > 0$ denotes some indifference region. In this work, we use the Sequential Probability Ratio Test (SPRT) [19, 14], to choose between $H_0$ and $H_1$ with a given strength $(\alpha, \beta)$[1] using minimal samples $m$.

To verify the performance of our prediction system, we first compute the MC abstraction

$$\mathcal{M} = (\mathcal{S}, Init, T, AP, L)$$

where

- $\mathcal{S} \ni s = \{\Delta \mathbf{p}_1, ..., \Delta \mathbf{p}_n\}$, $\Delta \mathbf{p}_i \in \mathcal{P}$, where $\mathcal{P}$ is a set of possible position changes
- $Init = \{\}$
- $T(s, s') = \mathbb{P}(\Delta \mathbf{p}_{n+1} \mid s) = \begin{cases} \frac{1}{|\mathcal{P}|} & \Delta \mathbf{p}_{n+1} \in \mathcal{P} \\ 0 & \text{else} \end{cases}$
- $AP = \{bad, good\}$
- $L(s) = \begin{cases} good & \mathbf{p}_{n-m+k} \in e_k \;\forall k \in [1,m] \\ bad & \text{else} \end{cases}$

and $\mathbf{p}_j = \sum_{i=1}^{j} \Delta \mathbf{p}_i$. An example of this abstraction is shown in Figure 3.

**Remark 1.** Uniform random transitions $T(s, s')$ form a *worst case scenario* for our prediction system, since the prediction system described in Section III relies on structure underlying the timeseries data.

With this in mind, given a sufficiently fine gridding $\mathcal{P}$, we can estimate the value of (5) by checking the PCTL property

$$\phi^* = \mathbb{P}_{\geq \theta}(\Box good) \tag{6}$$

for various values of $\theta$. The highest value of $\theta$ such that $\phi^*$ is satisfied is an upper bound (recalling Remark (1)) of the probability of mispredicting an obstacle's motion.

[1] $\alpha$ and $\beta$ denote the maximum probabilities of Type I and Type II error respectively

| | $\theta = 0.9$ | $\theta = 0.85$ | $\theta = 0.8$ | $\theta = 0.75$ |
|---|---|---|---|---|
| $\sigma^2 = 0$ | UNSAT | UNSAT | SAT | SAT |
| $\sigma^2 = 0.001$ | UNSAT | SAT | SAT | SAT |
| $\sigma^2 = 0.01$ | SAT | SAT | SAT | SAT |
| $\sigma^2 = 0.05$ | SAT | SAT | SAT | SAT |

TABLE I: Satisfaction of Equation (7) for various obstacle threshold probabilities $\theta$ and noise variances $\sigma^2$.

Unfortunately, existing statistical model checking methods do not handle unbounded properties like $\phi^*$ well [14]. Instead, we verify the related PCTL property

$$\phi = \mathbb{P}_{\geq\theta}(\Box^{\leq N} good). \tag{7}$$

For sufficiently large $N$, the largest $\theta$ such that the model $\mathcal{M}$ satisfies $\phi$ serves as an *approximation* of the probability of mispredicting an obstacle's motion.

### A. Example

As an example, we define a 3x3 grid

$$\mathcal{P} = \{\Delta\mathbf{p}\} = \{\begin{bmatrix}\Delta x \in \{-1,0,1\} \\ \Delta y \in \{-1,0,1\}\end{bmatrix}\}.$$

For all trials, we use the SPRT to determine the satisfaction of Equation 7 with $N = 20$, $\alpha = 0.1$, $\beta = 0.1$, and $\delta = 0.05$. The results for select values of $\theta$ are shown in Table I.

We test robustness to perception uncertainty by adjusting the variance, $\sigma^2$, of the noise added to position measurements. The results in Table I for different values of $\sigma$ demonstrate that predictions are not only robust to this sort of additive perception uncertainty, but in fact the addition of noise improves the quality of predictions. Intuitively, this suggests that noisier training data results in higher uncertainty estimates, but at the price of reduced precision. We leave a more thorough characterization of this tradeoff as a topic of future work.

## VI. Provable Correctness

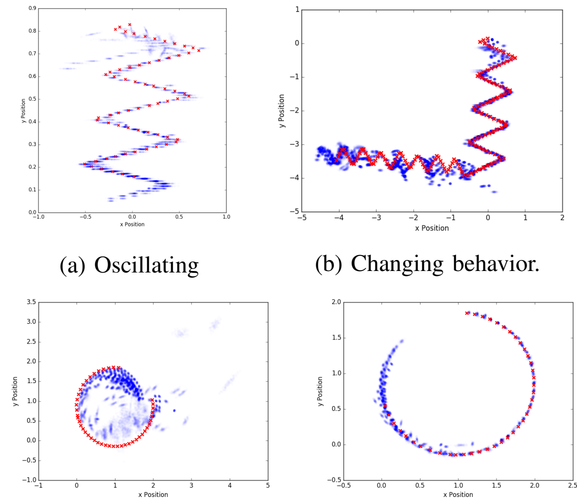
In this section, we show that a known value of Equation (5), which we approximated in the previous section, allows us to derive probabilistic guarantees on the performance of our collision avoidance approach. Specifically, we will show that we can obtain bounds on the probability of collision in multi-agent scenarios as well as for $N$-agent reciprocal collision avoidance. The proofs, omitted here for brevity, are available at `https://arxiv.org/abs/1811.01075`.

First, consider a scenario in which agent $A$ avoids $N$ arbitrarily moving obstacles, which we label $B_1, B_2, \dots, B_N$.

**Theorem 1.** *Multi-Agent Collision Avoidance.*
Assume that:

1) For each obstacle $B_i$, we can guarantee the accuracy of predictions $e^i_k$ in the sense that $\mathbb{P}(\mathbf{p}^{B_i}_{n+k} \in e^{B_i}_k \ \forall\ k \in [1,m]) \geq \theta$.
2) There exists some dynamically feasibly velocity $\mathbf{v}_{safe} \notin NPVO^{\gamma,m}_{A|B_1,B_2,\dots,B_N}$

Then the probability of a collision between $A$ and *any* of the obstacles $B_1, B_2, ...B_N$, $\mathbb{P}(collision)$, is bounded by $1-\theta^N$.



(a) Oscillating  (b) Changing behavior.

(c) Circle: initial  (d) Circle: after $\sim$ 10s

Fig. 4: Predictions of several obstacle motion patterns. Red x's indicate actual obstacle position, blue o's are sample predictions.

Now consider a scenario in which $N$ agents, $A_1, A_2, \dots, A_N$, each avoid the other $N-1$ agents:

**Theorem 2.** *Reciprocal Collision Avoidance.*
Assume that:

1) We can guarantee that $\mathbb{P}(\mathbf{p}^{A_j}_{n+k} \in e^{A_j}_k \ \forall\ k \in [1,m]) \geq \theta$ for any agent $A_i$ relative to any other agent $A_j$ ($i \neq j$).
2) For all agents $A_i$, there exists some velocity $\mathbf{v}^{A_i}_{safe} \notin NPVO^{\gamma,m}_{A_i|\{A_j\}_{i\neq j}}$.

Then the probability that there is *any* collision in the workspace, $\mathbb{P}(collision)$, is bounded by $1-(2\theta-\theta^2)^{\frac{1}{2}(N-1)N}$.

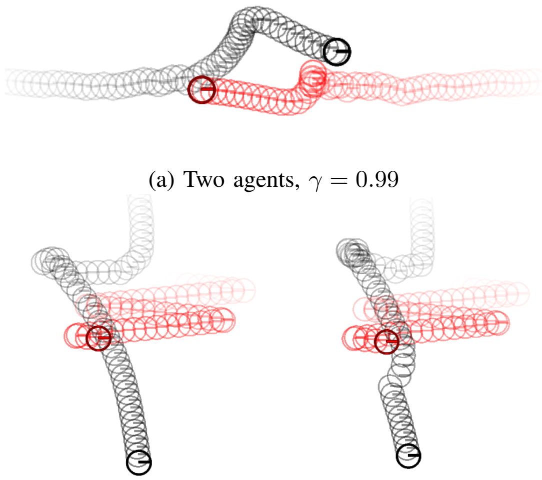
## VII. Simulation

### A. Dynamic Obstacle Motion Prediction

We implemented the motion prediction system described in Section III in Python using Tensorflow, ROS, and the Stage simulator [18]. We use an LSTM network with a hidden layer size of 20, dropout probability $p = 0.9$, $N_{iter} = 100$ and a learning rate of 0.003 to predict 10 steps into the future ($m = 10$). Note that these hyperparameter values are not optimal, and instead serve as a proof of concept of this methodology. The resulting predictions are shown for several patterns of obstacle motion in Figure 4. All predictions were made in real time (2Hz) on a laptop with an Intel i7 processor and 32GB RAM. Code to reproduce these experiments is available at `https://github.com/vincekurtz/rnn_collvoid`.

### B. NPVO Controller

We implemented the NPVO controller described in Section IV in simulation as well, using the *SciPy* [10] minimization toolkit to find $\mathbf{v}_{safe} \notin NPVO_{A|B}$. Results for several illustrative examples are shown in Figures 1 and 5.



(a) Two agents, $\gamma = 0.99$

(b) $\gamma = 0.5$

(c) $\gamma = 0.99$

Fig. 5: A higher threshold $\gamma$ leads to greater deviation from the desired trajectory, but avoids obstacles by a larger margin.

## VIII. Conclusion

We presented a novel method for predicting the motion of dynamic obstacles from past observations. We introduced the NPVO as a generalization of existing velocity obstacle concepts and demonstrated that an NPVO controller with our prediction system avoids moving obstacles that existing collision avoidance algorithms cannot. We used statistical model checking to (approximately) verify that actual motion remains within certain bounds with a given probability. Finally, we proved upper bounds on the probability of collision in multi-agent and reciprocal collision avoidance scenarios.

Future work will focus on characterizing the tradeoff between better uncertainty estimates arsing from additional perception noise and the resulting decreased performance, using Bayesian Optimization [16] to find optimal hyperparameter values, and quantifying the scalability of our approach in scenarios with many obstacles.

## References


[1] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[2] Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier. Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1610–1616. IEEE, 2007.

[3] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[4] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016.

[5] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal aspects of computing*, 6(5):512–535, 1994.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*, pages 3–29. Springer, 2017.

[8] Peter J Huber et al. Robust estimation of a location parameter. *The annals of mathematical statistics*, 35 (1):73–101, 1964.

[9] R. Hug, S. Becker, W. Hübner, and M. Arens. Particle-based pedestrian path prediction using LSTM-MDL models. *ArXiv e-prints*, April 2018.

[10] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/. [Online; accessed ¡today¿].

[11] Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

[12] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: An overview. In *International conference on runtime verification*, pages 122–135. Springer, 2010.

[15] Zvi Shiller, Frederic Large, and Sepanta Sekhavat. Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 3716–3721. IEEE, 2001.

[16] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[17] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[18] Richard Vaughan. Massively multi-robot simulation in stage. *Swarm intelligence*, 2(2-4):189–208, 2008.

[19] Abraham Wald. Sequential tests of statistical hypotheses. *The annals of mathematical statistics*, 16(2):117–186, 1945.