Runtime SES Planning: Online Motion Planning in Environments with Stochastic Dynamics and Uncertainty

Hao-Tien Chiang[1], Nathanael Rackley [1], Lydia Tapia[1]

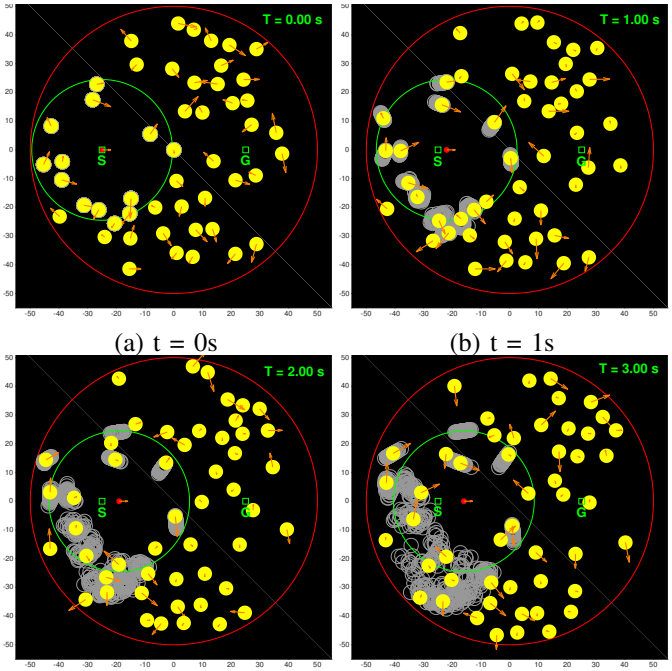
***Abstract*— Motion planning in stochastic dynamic uncertain environments is critical in several applications such as human interacting robots, autonomous vehicles and assistive robots. In order to address these complex applications, several methods have been developed. The most successful methods often predict future obstacle locations in order identify collision-free paths. Since prediction can be computationally expensive, offline computations are commonly used, and simplifications such as the inability to consider the dynamics of interacting obstacles or possible stochastic dynamics are often applied. Online methods can be preferable to simulate potential obstacle interactions, but recent methods have been restricted to Gaussian interaction processes and uncertainty.**

In this paper we present an online motion planning method, Runtime Stochastic Ensemble Simulation (Runtime SES) planning, an inexpensive method for predicting obstacle motion with generic stochastic dynamics while maintaining a high planning success rate despite the potential presence of obstacle position error. Runtime SES planning evaluates the likelihood of collision for any state-time coordinate around the robot by performing Monte Carlo simulations online. This prediction is used to construct a customized Rapidly Exploring Random Tree (RRT) in order to quickly identify paths that avoid obstacles while moving toward a goal. We demonstrate Runtime SES planning in problems that benefit from online predictions, environments with strongly-interacting obstacles with stochastic dynamics and positional error. Through experiments that explore the impact of various parametrizations, robot dynamics and obstacle interaction models, we show that real-time capable planning with a high success rate is achievable in several complex environments.

I. Introduction

Motion planning in environments with a large number of stochastically moving obstacles is critical in applications such as flight coordination [1], human interacting robots [2], assistive robots [3] and autonomous vehicles [4]. Real-world robots often operate among moving obstacles, e.g., pedestrians, animals and vehicles, while utilizing imperfect sensors. These obstacles tend to interact with each other in a stochastic manner. As a result, a motion planner must consider both sensor uncertainty and generic stochastic inter-obstacle dynamics in order to safely navigate through an environment. Due to the cost of simulating obstacle interactions, *online* methods that restrict the computational expense of prediction are often preferred.

Recently published methods that applied stochastic dynamic environments include artificial potential fields biased by stochastic reachable sets (APF-SR) [5] and stochastic ensemble-based (SES) planning [6]. These methods predict stochastic obstacle motion offline using stochastic reachability [7] and Monte Carlo (MC) simulation, respectively. Although these methods were shown to have high success rates, they are restricted by the use of offline computation that restricts the stochastic dynamics of obstacles to either non-interacting or weakly-interacting. This is because of the computational intractability of predicting strongly-interacting obstacle trajectories.



(a) t = 0s (b) t = 1s (c) t = 2s (d) t = 3s

Fig. 1: The Elastic Ricocheting environment with 50 moving obstacles with stochastic dynamics and inter-obstacle interactions where obstacles ricochet off each other elastically. The robot (red disk) starts at S and must navigate to G while avoiding stochastically moving obstacles (yellow disks, arrows indicate obstacle velocity). The green circle shows the robot obstacle detection radius and predicted positions of obstacles are indicated by gray circles. Snapshots at world times $t = 0$s through $t = 3$s and the corresponding predictions made at $t = 0$s are shown in (a-d) respectively.

Online planners that consider strongly-interacting obstacles are common in crowd simulation, human-robot interaction and multi-agent motion planning. Velocity Obstacles (VOs) [8] were developed to avoid collision with other agents based on instantaneous velocities. The Social Force Model (SFM) models crowd behavior using artificial forces [9] and was used to navigate a robot [10], [11]. Despite considering inter-obstacle interactions online, one drawback of these

[1] Computer Science, University of New Mexico, Albuquerque, NM 87131

methods is that they do not have a robust way of handling sensor uncertainty and generic stochastic obstacle interaction dynamics, resulting in lower planning success rates for these types of problems. A recent online method considers both generic stochastic obstacle interaction dynamics and sensor uncertainty [12]. However, the dynamics and sensor uncertainty in that method were limited to Gaussian processes.

In this paper, we propose Runtime Stochastic Ensemble Simulation (Runtime SES) planning, which extends our previous work, SES planning, in order to address more complex stochastic obstacle dynamics and uncertainty through online simulation. Runtime SES planning is a general method that considers *generic* strongly interacting stochastic obstacle dynamics, robot sensor uncertainty and robot dynamics constraints. The method works by alternating between obstacle motion prediction for nearby obstacles and planning at runtime. The obstacle stochastic motion prediction was done by flexible MC simulations to capture sensor uncertainty as well as generic strongly-interacting stochastically moving obstacles. Using the results of this prediction, the likelihood of collision can be estimated for any State-Time (ST) coordinates around the robot. The planning of the method then utilizes these likelihood estimates with a custom RRT which samples and locates nearest neighbors in ST space. To increase safety, the robot executes the path extracted from the tree, and Runtime SES planning checks the future path for likelihood of collision through the incorporation of the new prediction results.

The main contributions of this paper include: 1) The proposed Runtime SES planning is an inexpensive method to predict obstacle motion and estimate likelihood of collision. It can capture a wide range of environmental uncertainty, such as robot sensor error and strongly-interacting generic stochastic obstacle dynamics. 2) A customized RRT implementation that treats state and time on equal footings. We empirically show that it generates straighter paths and increases success rates compared to traditional RRTs. 3) Evaluation of our method against current state of the art planning methods capable of generating real-time solutions in environments with sensor uncertainty and up to 50 stochastically moving, strongly-interacting obstacles (shown in Figure 1). These methods include: SES [6], APF-SR [5], VO [8] and Gaussian APF [13]. The enclosed video submission contains examples of simulations in two different environments.

II. Related Work

Monte Carlo (MC) simulation is a powerful technique to predict stochastic obstacle motion and handling sensor uncertainty. SES-based planning captures stochastic obstacle motion and sensor uncertainty through the use of an *offline* MC simulation [6]. The simulation portion of SES occurs within continuous space and can capture a wide range of stochastic obstacle dynamics. The algorithm queries the offline simulation result at runtime and plans with a tree-based planner in State Time (ST) space to quickly find collision free paths. Rather than running MC simulations in continuous space, [15] simulates stochastic pedestrian motion and pedestrian position uncertainty on a grid. The result of the simulation is fed to an Anytime RRT [16] to generate a collision free path. Instead of simulating obstacles and planning with a tree-based method, [17] generates trajectories for both robot and obstacle independently with MC simulation. It then picks a path for the robot that minimize a cost function combining the probability of collision and distance to goal. The MC simulations employed by these methods capture both sensor uncertainty and stochastic obstacle motion but are limited to obstacles with either non-interacting or weakly-interacting dynamics since the obstacle prediction is done offline.

Agent interactions are crucial for both crowd simulation and multi-agent motion planning. Artificial Potential Field (APF) based methods such as SFM are widely used to simulate crowd motion or navigating a robot around pedestrians [9], [10], [11]. Velocity Obstacles (VO)s [18] are also widely used for crowd simulation and navigation between moving obstacles [8]. Similar to SFM, [19] proposed a human-friendly reactive planner to navigate multiple robots with Gaussian position uncertainty. Despite the fact that these methods consider inter-obstacle interaction, it is difficult to incorporate robot sensor uncertainty (such as obstacle position uncertainty) or obstacle stochastic dynamics into them. In addition, these methods are typically designed for environments with one specific obstacle dynamics such as pedestrians or other agents implementing the same algorithm. In contrast, our method is designed to handle generic obstacle dynamics.

Trautman [12] proposed using a Gaussian processes to approximate inter-obstacle interaction in order to predict obstacles' future position. The prediction result is then integrated with a custom Receding Horizon Controller (RHC) to direct the robot. While this method incorporated inter-obstacle interaction, sensor uncertainty and stochastic obstacle dynamics similar to our proposed method, the error model and the stochastic dynamics in this case are restricted to Gaussian processes whereas Runtime SES planning is able to handle generic processes by means of flexible MC simulations.

III. Methods

Similar to our previous method SES [6], Runtime SES planning also has two components: the obstacle prediction component and the motion planning component. The primary difference between SES and Runtime SES is that the obstacle prediction is done online instead of offline.

Runtime SES planning works by alternating between planning and prediction. The prediction of stochastic obstacle motion was achieved using MC simulations of nearby obstacles in order to find the likelihood of collision at any given ST coordinate near the robot. A variant of RRT utilizes this information in order to plan trajectories with high success rates and short finish time. By alternating online between planning and prediction, Runtime SES planning provides the following benefits: 1) Virtually any complex stochastic

changes to the planning environment such as sensor error and obstacle stochastic dynamics can be approximated with little cost. This includes environments that involve strong inter-obstacle interactions. 2) Resulting trajectories have higher success rates by considering inter-obstacle interaction.

A. Runtime Stochastic Ensemble Simulation (Runtime SES)

Runtime SES predicts the future positions of strongly-interacting stochastically moving obstacles through a MC simulation as shown in Algorithm 1.

Algorithm 1 Runtime SES

Input: Number of nearby obstacles N, Number of MC trials M, Simulation time horizon T_h, Simulation time resolution δ, Planning time resolution Δ

Output: Snapshots of future nearby obstacle i position for each MC trial j: $x_p = \{x_{i,j}(0), x_{i,j}(\Delta), x_{i,j}(2\Delta), ..., x_{i,j}(T_h)\ \}$, $j = 1 \sim M$

1: $k = 1$
2: **for** each MC trial $j = 1 \sim M$ **do**
3: **for** each nearby obstacle $i = 1 \sim N$ **do**
4: $\bar{x_i} = x_{i,j}(0) = getPosFromErrorModel(i)$
5: $\bar{v_i} = v_{i,j}(0) = getVelFromErrorModel(i)$
6: **end for**
7: **end for**
8: **for** each MC trial $j = 1 \sim M$ **do**
9: **for** $t = 0$; $t <= T_h$; $t = t + \delta$ **do**
10: **for** each nearby obstacle $i = 1 \sim N$ **do**
11: $f = computeForces(i, \bar{x}_{1\sim N}, \bar{v}_{1\sim N}, t)$
12: $\bar{v_i} = modifyVel(f, \bar{v_i}, \bar{x}_{1\sim N}, \bar{v}_{1\sim N}, t)$
13: $\bar{x_i} = \bar{x_i} + \Delta \cdot \bar{v_i}$
14: // Record snapshots
15: **if** $t = k\Delta$ **then**
16: $x_{i,j}(k\Delta) = \bar{x_i}$
17: k++
18: **end if**
19: **end for**
20: **end for**
21: **end for**

Runtime SES starts by randomly sampling the position and velocity of nearby obstacles (within distance d from the robot) using a sensor error model for each MC trial (lines 4-5). In $getPosFromErrorModel$ and $getVelFromErrorModel$, the robot reads the position and velocity ($X_{sensorReading}$) of an obstacle i from a simulated sensor which returns a position (X_{sample}) that deviates from the truth using a generic sensor error model. In this paper, we applied no noise, uniform noise (1), constant variance Gaussian (2) noise and distance-dependent variance Gaussian (3) noise position error models:

$$X_{sample} = \begin{cases} X_{sensorReading}+\text{U(-e,e)} & (1) \\ \text{N}(X_{sensorReading}, \sigma) & (2) \\ \text{N}(X_{sensorReading}, \sigma(ar^2)) & (3) \end{cases}$$

where a is a constant and r is the distance between the robot and the obstacle. The distance-dependent variance Gaussian error model is similar to the depth error of structured light depth sensors such as Microsoft Kinect [20].

After sampling the initial position and velocity, Runtime SES conducts a MC simulation consisting of M trials (lines 8-21). For each MC trial, the total force exerted on obstacle i is computed (line 11). The $computeForces$ and $modifyVel$ methods have complete phase space and time information for all nearby obstacles in the simulation and can therefore capture the stochastic dynamics of virtually any moving obstacle, including inter-obstacle interactions and boundary conditions (which is typically not described by forces). We demonstrate two complex stochastic obstacle dynamics with strong inter-obstacle interactions in this paper, the details of which can be found in Section IV. The MC simulation integrates the obstacle trajectory (line 13) with simulation time resolution δ and records the position every Δ seconds (lines 15-18).

The output of Runtime SES consists of a series of snapshots (one series for each MC trial) of future obstacle positions at various time intervals (Δ apart). This approximates the future evolution of nearby stochastically moving obstacles to time horizon T_h. The simulation results of Runtime SES and the actual evolution of obstacles are shown in Figure 1 (predictions made at $t = 0$s and M=50).

B. Runtime SES Motion Planning

Since Runtime SES approximates the evolution of nearby obstacles (the snapshots store various possibilities of obstacle positions at specific times), we can estimate the likelihood of collision in state-time (ST) space coordinates near the robot in the following way:

$$collProb(x_R, t) = \frac{1}{M} \sum_{j=1}^{M} \sum_{i=1}^{N} CD(x_R, x_{i,j}(\bar{t})) \quad (1)$$

where $\bar{t} = round(t/\Delta) * \Delta$, x_R is the robot state and the function CD returns 1 if x_R and $x_{i,j}(\bar{t})$ are in collision and 0 otherwise. Runtime SES planning utilizes a custom Rapidly-exploring Random Tree (RRT) [21] with time step Δ in Algorithm 2. Each node in the tree corresponds to a particular snapshot $k\Delta$, $k \in \mathbb{N}$ and can therefore be checked for potential collision using the $collProb$ function.

In contrast to SES, Runtime SES planning predicts the trajectory of nearby obstacles every $T_{Predict}$ seconds (Algorithm 2, lines 2-5) instead of offline. This allows the predictions to include strongly-interacting stochastic obstacle dynamics.

Like SES, the algorithm first attempts to grow a tree *directly* toward the goal in $growGoalTree$ (line 9, details can be found in [6] section IV.B). If this tree growth results in potential collisions ($collProb \geq P_{acc}$ for any node in the goal tree), a custom RRT is grown from the robot's current position via $growFullTree$ (line 11).

Our custom RRT modifies the random sampling and the nearest neighbor selection of standard RRT. Traditional RRT

Algorithm 2 Runtime SES Motion Planning

Input: Snapshots of future nearby obstacle i position for each MC trial j: $x_p = \{x_{i,j}(0), x_{i,j}(\Delta), x_{i,j}(2\Delta), ..., x_{i,j}(T_h)\ \}$, $j = 1 \sim M$, Robot Current State x_r, Planning Time step Δ, World simulation time step Δ_{world}

1: **for** $t = 0$; $t < maxTime$; $t = t + \Delta_{world}$ **do**
2: **if** $t - t_p > T_{Predict}$ **then**
3: $t_p = t$
4: $RuntimeSES()$ // Algorithm 1
5: **end if**
6: **if** $reGrowTree == true$ **then**
7: $Tree : \mathcal{T} = pruneTree(currentNode)$
8: $t_{lastPlan} = t$
9: $(growFull, \mathcal{T}) = growGoalTree(x_R, x_p, t, t_p)$
10: **if** $growFull == true$ **then**
11: $\mathcal{T} = growFullTree(x_R, t, \mathcal{T}, x_p, t_p)$
12: **end if**
13: $\mathcal{P} = getPathFromTree(\mathcal{T})$
14: **end if**
15: $x_R = x_R + \Delta_{world} \cdot getAction(\mathcal{P}, t - t_{lastPlan})$
16: $reGrowTree = checkFutureNodes(\mathcal{P}, x_R, t, t_p)$
17: **end for**

implementations randomly sample in state space and connect to the nearest neighbor according to a cost metric that considers only the distance in state space. Our custom RRT randomly samples in state space as well as time (ST space):

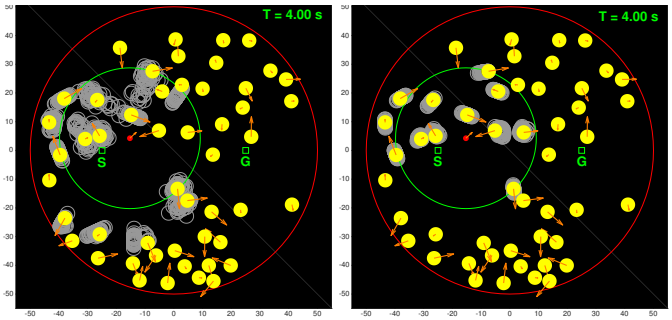
$$\begin{aligned} x_{rand} &= x_R + U(-V_{max}T_h, V_{max}T_h) \\ t_{rand} &= t + U(0, T_h) \end{aligned} \tag{2}$$

where t is the current simulation time and V_{max} is the maximum speed of the robot. In addition, the nearest neighbor metric is defined as:

$$C(x, t') = \|x - x_{rand}\| + |(t' - t_{rand})|V_{max} \tag{3}$$

for a node that is positioned at x and has time t'. It is important to note that although many RRT variants plan in ST space [22], [15], [23], to our best knowledge no RRT variant samples and find nearest neighbors as described above. This custom RRT has the following benefits when combined with Runtime SES planning: 1) The randomly sampled point has a specific ST coordinate and can therefore be checked for potential collisions using the $collProb$ function. In this case, sampled points with excessively high likelihood of collision will be discarded and therefore the resulting tree grows toward the obstacle free regions in ST space. 2) We determined experimentally that our custom RRT performs better in both success rates and finish time than similar methods. Further details can be found in IV-C.

After the local tree is grown (the leaf node of the goal tree reaches T_h or the $growFullTree$ calls $collProb$ for $maxCD$ times), a path that satisfies the τ-safety criterion [23] (at least τ seconds long) and minimizes a weight function that considers both safety and distance to the goal. This path extraction procedure (line 13) is the same as that of SES [6]. Like SES, if no path satisfies τ-safety criterion, the longest path in the tree is used instead.



(a) (b)

Fig. 2: Runtime SES prediction of future obstacle positions for world time $t = 4$s, starting at (a) $t = 1$s and (b) $t = 3$s. The robot (red disk), obstacles (yellow disks) and predicted obstacle positions (gray circles) are shown.

Following the local tree growth, the robot then executes the path (line 15) and every time the robot reaches a node, it checks future nodes (within τ seconds of current time) along the path for potential collision (line 16). This step is crucial for Runtime SES planning in order to find paths with high success rates. As time progresses and the robot executes the path, new predictions of obstacle motion for a given time in the future become increasingly accurate due to a shorter prediction horizon. To illustrate this, Figure 2 shows two predictions for the same given world time $t = 4$s. Figure 2a shows the predicted environment starting at $t = 1$s and Figure 2b at $t = 3$s. The latter has a shorter prediction horizon (prediction time is closer to the given $t = 4$s) and a reduced spread in obstacle position (improved precision). Obstacles to the right of the robot that were not predicted in Figure 2a were predicted in 2b, demonstrating improved accuracy. The method $checkFutureNodes$ integrates information from these new predictions in order to increase path safety.

If $checkFutureNodes$ finds any node with likelihood of collision higher than P_{acc} or the robot traversed near the end of a path, instead of discarding the entire tree as SES does, the algorithm instead prunes nodes in the tree that are not descendants of the current node (line 7) and grows from this pruned tree (lines 8-13). We define traversing near the end of a path as satisfying one of the two following conditions: 1) a path has less than τ seconds left to execute, or 2) the initial extracted path length is shorter than τ seconds and the robot has executed more than half of the path.

IV. Experiments

To demonstrate Runtime SES planning, we tested two different robots in two separate environments consisting of moving obstacles with strongly interacting dynamics. The world simulation time step $\Delta_{world} = 0.01$s. The obstacles and the robot are confined within a circle of radius 50m. An

obstacle ricochets off the world boundary by instantaneously inverting the direction of the perpendicular velocity component (velocity component that points toward the center of the circle). All methods were implemented in C++. The Velocity Obstacle (VO) algorithm was adapted from the RVO2 C++ code base [24] implementation of the Optimal Reciprocal Collision Avoidance (ORCA) algorithm. This algorithm [25] was modified to allow for single-agent collision avoidance, removing the reciprocal aspect of ORCA while maintaining many of ORCA's linear programming optimizations. APF-SR was implemented by mapping the offline computed SR set to the instantaneous position and velocity of obstacles and using this to compute a repulsive potential. All experiments were run on a single core of an Intel i7-3720QM at 2.6GHz with 16GB of RAM. All experiments were repeated 100 times. Uncertainty in success rates due the limited number of experiments is captured using the 99% confidence level derived from the central limit theorem [26].

A. ***Experiment 1:*** *Elastic Ricocheting Environment*

Experiment 1 is designed to compare Runtime SES planning with SES [6], APF-SR [5], Gaussian APF method [13] (abbreviated as Gaussian) with $\mathcal{N}(0,1)$ and VO [8], [25] in an environment with highly stochastic and complex obstacle dynamics. The environment is 2D and has 20 to 50 interacting disk-shaped (radius 2.5m) moving obstacles. Robots were tested with a no position noise and a constant variance Gaussian noise ($\sigma = 0.25$m) model.

Obstacle Dynamics: The obstacle dynamics include intrinsic stochastic motion and deterministic interaction dynamics. An obstacle stochastically samples speed (without changing its heading) every $T_{sample} = 0.1$s. The set of possible speeds is $\{1, 2, 5, 7\}$ m/s with probability $\{0.4, 0.1, 0.2, 0.3\}$, respectively. When two obstacles collide, they bounce off each other elastically (changes occur to both heading and speed). Simulating the system with elastic collisions is difficult because these systems are known to be chaotic. The Lyapunov exponent (which describes how the distance between two nearby trajectories changes with time) of such a system has been shown to be positive [14], and therefore nearby trajectories diverge exponentially with time. Our obstacle dynamics are even more difficult as obstacles stochastically sample speed (rather than drifting between collisions).

Setup: The robot is a holonomic disk robot with radius of $1m$ and a maximum speed of 3m/s (slower than the average speed of obstacles at 3.7m/s). The robot starts at $(-25\text{m}, 0\text{m})$ and its goal is located at $(25\text{m}, 0\text{m})$. The initial environment and evolution with 50 obstacles is shown in Figure 1. The APF-SR and Gaussian algorithms have a goal bias of 0.01. Due to the stochasticity of obstacle motion, we empirically determined that by padding the obstacle radii by 10%, this particular perceived radius for VO yields the highest success rates. The parameters employed by Runtime SES planning are: $T_h = 7$s (time horizon), $T_{Predict} = 0.5$s (prediction interval), $maxCD = 5000$ (maximum number of $collProb$ calls), $\Delta = 0.2$s (planning time resolution), $M = 50$ (number of MC trials), $d = 24.5m$ (robot obstacle detection range) and $P_{Acc} = 0.05$ (probability of acceptance).

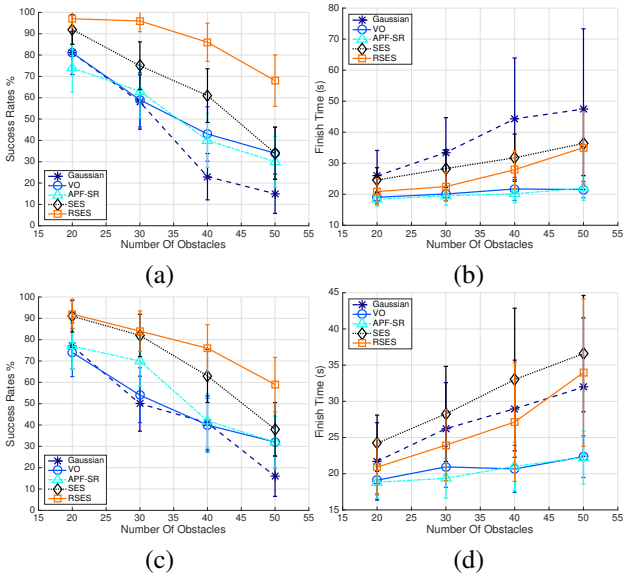


Fig. 3: Success rates and finish time comparison between Runtime SES planning (RSES), APF-SR, VO, SES and Gaussian AFP methods in the Elastic Ricocheting Environment. No position error (a-b) and Gaussian position error with $\sigma = 0.25$m (c-d) are shown.

Results: Figure 3a shows that the success rates of Runtime SES planning exceed those of comparison methods in both the no position noise and constant variance Gaussian model. In the 50 obstacle environment, chaotic obstacle interaction happens often (about 11.1±1.2 inter-obstacle collisions per second). As a result, methods that perform well for non-interacting stochastically moving obstacles such as APF-SR and SES have success rates approaching that of VO instead of clearly outperforming VO as reported in [6]. The finish time of Runtime SES is higher than APF-SR and VO due to the algorithm being sampling-based. Comparing SES and Runtime SES, however, we find that Runtime SES has a shorter finish time due to consideration of inter-obstacle interaction and the employment of our custom RRT that samples in ST space.

No Error					
#Obs.	RSES	SES	APF-SR	Gaussian	VO
20	1.9±.1	.4±.3	.009±.003	.004±.003	.007±.003
30	4.5±5.5	.3±.2	.012±.005	.005±.002	.010±.010
40	6.7±3.3	.4±.4	.016±.006	.005±.002	.010±.005
50	13.3±9.1	.9±.85	.026±.018	.007±.002	.010±.004
Constant variance Gaussian Error					
20	2.7±2.4	.4±.5	.009±.006	.008±.006	.010±.004
30	5.2±7.8	.6±.4	.012±.007	.005±.003	.010±.010
40	8.5±13.4	.7±.6	.017±.006	.004±.003	.012±.006
50	13.3±10.0	.9±.9	.029±.020	.008±.004	.015±.006

TABLE I: Average computation time per planning step in the Elastic Ricocheting environment with no position error (top) and constant variance Gaussian error (bottom). The units are in milliseconds. RSES stands for Runtime SES.

Table I shows that Runtime SES is about 10 times slower than SES and much slower than other methods. However,

Runtime SES finds a path with much higher success rates and remains real-time capable even in the environment with 50 moving obstacles (13.3 ms per planning step). In addition, the presence of position noise does not significantly impact computation time per planning step for all methods.

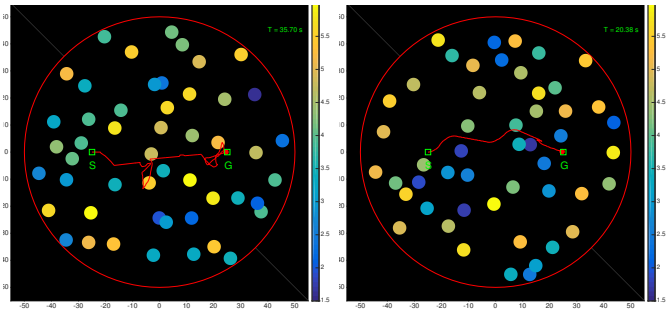
B. *Experiment 2: Electric Charge Environment*

Experiment 2 is designed to demonstrate a strong long-range inter-obstacle interaction that causes the obstacles to change speed and heading at all times. In addition, we demonstrate the ability for Runtime SES to handle nonholonomic constraints with a unicycle robot and various types of obstacle position error models. The combination of position error and deterministic obstacle dynamics with uncertain parameters (electric charge in this case) mimics real-world scenarios as described in [27] and [15].

Obstacle Dynamics: For a given obstacle i, the acceleration is Coulomb-like:

$$a_i = \sum_{i=1, i\neq j}^{N} x_{j,i} \frac{C_i C_j}{\|x_{j,i}\|^2}, \tag{4}$$

where C_i is the charge of obstacle i and $x_{j,i}$ is the vector from the center of obstacle j to obstacle i. The charge of each obstacle is randomly sampled from $U(1.5, 6)$ charge units. The environment and the random charge for each obstacle are shown in Figure 4. Despite the fact that the dynamics of (4) are deterministic, the robot can only observe the charge of a given obstacle with a very noisy error model (a 50% uniform error model is used).



(a) (b)

Fig. 4: Electric Charge environment and example paths (red curves) taken by (a) the holonomic robot and (b) the unicycle robot. The robot is the red disk and the color of the obstacle represents the amount of charge.

Setup: We tested both holonomic (with the same dynamics as in Experiment 1) and unicycle robots. The unicycle robot has a maximum turn rate of $\pi/5$ rad/s and the same 3m/s maximum speed. In addition to charge uncertainty, we also tested three obstacle position error models: Uniform with range $e = 0.5$m (20% of obstacle radius), constant variance Gaussian with $\sigma = 0.5$m and distance-dependent variance Gaussian with $\sigma = 0.005r^2$. The parameters used in Runtime SES are the same as Experiment 1 except for $T_{Predict}$ = 2s. APF-SR and SES were not included in this experiment due to the nature of the stochastic dynamics (which are caused by uncertainty in parameters of obstacle interaction dynamics), which are very difficult to compute offline.

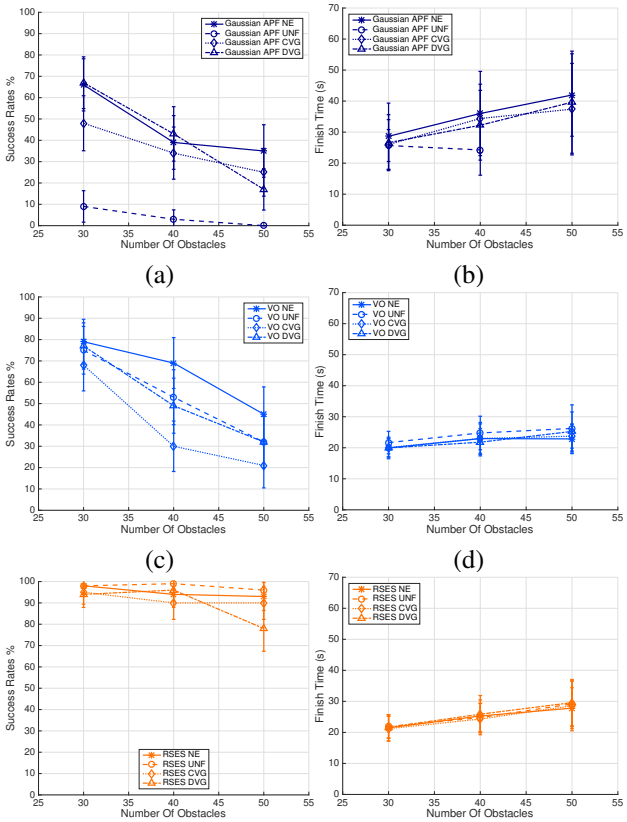
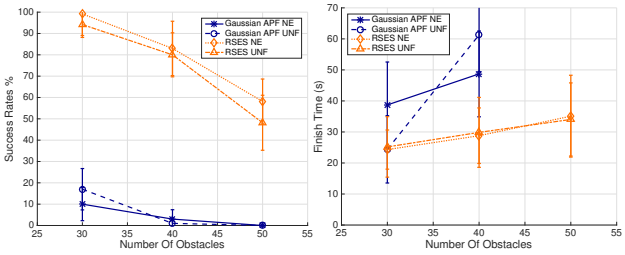


Fig. 5: Holonomic robot success rates and finish time comparison between Gaussian APF (a-b), VO (c-d) and Runtime SES planning (RSES)(e-f) methods in the Electric Charge environment. Three error position models were tested and compared to the no error (NE) model: Uniform (UNF), Constant Variance Gaussian (CVG) and Distance-dependent Variance Gaussian (DVG). The missing finish time data for Gaussian is due to success rates being lower than 1%.

Results: Figure 5a shows that reactive methods such as Gaussian have low success rates and high finish times due to high obstacle speed and density. VO computes velocity obstacles based on the instantaneous velocity and inaccurate positions to avoid collision. It does not take into account that obstacles accelerate as a result of inter-obstacle interaction. This results in low success rates comparable to Experiment 1 despite the system dynamics being deterministic (shown in Figure 5c). In contrast, Figure 5e shows Runtime SES planning is able to predict obstacle motion despite large uncertainties in both charge and position, resulting in very high success rates in the holonomic case, even in the difficult 50 obstacles environment. In addition, Figure 5a, c and e show that various position error models severely impact the success rates of Gaussian and VO while the success rates of Runtime SES remain similar. This indicates Runtime SES is more robust against obstacle position uncertainties. Figure 6 shows that the more constrained unicycle robot dynamics yield lower success rates for Runtime SES for both no error and uniform position error models. This is likely because the

Region of Inevitable Collision (RIC) [28] can be very large due to the unbound obstacle speed and the unicycle robot is less agile in moving away from RIC. The maximum speed of heavily charged obstacles can often reach 9-12m/s, 3 to 4 times faster than the robot. Figure 5b, d and f indicate the finish time of VO is similar to Runtime SES. This indicates the constant acceleration and position sensor uncertainty give rise to suboptimal velocity obstacle calculations.



(a) (b)

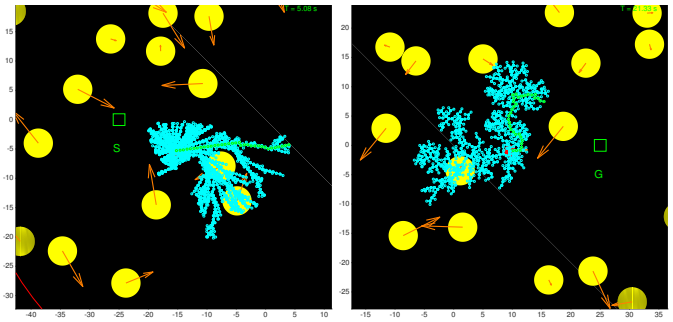
Fig. 6: Unicycle robot success rates and finish time comparisons between Gaussian APF and Runtime SES planning methods in the Electric Charge environment. The Uniform (UNF) error position models were tested and compared to the no error (NE). The missing finish time data for Gaussian is due to success rates being lower than 1%.

C. *Discussion*

Experiment 1 clearly shows that Runtime SES planning has very high success rates and is real-time capable despite the presence of a large number of moving obstacles with chaotic motion, obstacle position error and stochastic interaction dynamics. In all cases, Runtime SES has higher success rates than comparison methods. Despite the runtime obstacle predictions operating slower than comparison methods, these predictions offer significant gains in success rates and shorter finish times while remaining real-time capable. Experiment 2 demonstrates that Runtime SES can successfully plan for dynamically constrained robots in the presence of strong long range inter-obstacle interactions with large obstacle position uncertainty. This is a problem that very recently published methods such as APF-SR and SES could not address. Experiments 1 and 2 have drastically different obstacle stochastic dynamics, position error models and robot dynamics, but Runtime SES is planning in both environments with only the modification of the $T_{Predict}$ parameter.

Figure 7a shows Runtime SES with a typical example of our custom RRT that samples in ST space while Figure 7b shows Runtime SES with a traditional RRT. Table II indicates our custom RRT generates, in general, straighter paths without using expensive additional re-wiring and parent search procedures like RRT* [29]. This allows the algorithm to avoid obstacles more efficiently. As a result, it is more likely to successfully reach the goal.

Table III shows the performance comparison between the $pruneTree$ technique used in Algorithm 2 and discarding tree employed in SES. By pruning the tree instead of discarding, we observe the robot tends to maintain a better path stability, i.e., the robot tends to follow a similar path



(a) (b)

Fig. 7: (a) Custom RRT and (b) traditional RRT examples. The green curve is the selected path and the red disk is the robot. In general, (a) generates a straighter path compared to (b).

Tree Type	Success Rates	Finish Time (s)	Comp. Time (ms)
Custom RRT	84±9%	27.3±6.1	9.1±6.4
Traditional RRT	75±11%	34.0±11.6	10.0±8.8

TABLE II: Performance comparison between custom RRT and traditional RRT as used in Runtime SES. The environment is identical with Experiment 1 and has 40 moving obstacles.

after the $reGrowTree$ flag is raised and a new round of tree growth started. This allows the robot to efficiently avoid obstacles and increase success rates.

Technique	Success Rates	Finish Time (s)	Comp. Time (ms)
Prune Tree	84±9	27.3±6.1	9.1±6.4
Discard Tree	71±12	27.4±6.6	7.9±4.2

TABLE III: The impact of pruning and discarding trees. The environment is Experiment 1 and has 40 moving obstacles.

Runtime SES predicts obstacle stochastic motion online every $T_{predict}$ seconds. Empirical analysis (Table IV) shows that larger $T_{Predict}$ values are detrimental to success rates but frequent predictions do not incur much computational time overhead. This is supported by the fact that Runtime SES has a finite obstacle detection radius d, and with a larger prediction interval of $T_{Predict}$, there is a greater potential that unpredicted obstacles may enter this radius. Running the prediction over shorter time intervals allows for the integration of new obstacle information and more accurate obstacle position estimates.

$T_{Predict}$ (s)	Success Rates	Finish Time (s)	Comp. Time (ms)
0.1	90±8	25.8±5.7	11.4±3.8
0.5	84±9	27.3±6.1	9.1±6.4
1	74±11	30.0±8.4	8.9±11.7
2	62±12	30.6±7.9	14.1±29.8

TABLE IV: The impact of $T_{Predict}$. The environment is identical with Experiment 1 and has 40 moving obstacles with no noise.

Table V indicates that, unlike $T_{Predict}$, the success rates are within one standard deviation for a wide range of MC trials despite the presence of constant variance Gaussian obstacle position noise with $\sigma = 0.25$m. This is likely because the $checkFutureNode$ subroutine compensated for the poor approximation quality.

# of MC Trials	Success Rates	Comp. Time (ms)
25	70±12	5.7±4.2
50	76±11	11.5±9.7
100	82±10	27.7±6.8
200	74±11	32.1±32.7
400	78±11	55.9±57.8

TABLE V: The impact of the number of MC trials. The environment is identical with Experiment 1 and has 40 moving obstacles and constant variance Gaussian position error model.

V. Conclusion and Future Work

In this paper, we proposed Runtime SES planning, a novel motion planning algorithm for environments with sensor uncertainty and obstacles with strongly-interacting stochastic dynamics. We demonstrated that our algorithm is highly successful in these types of environments by alternating between obstacle prediction via MC simulations and planning with a RRT that treats state and time in equal footings. Our experiments also showed that Runtime SES planning generates trajectories with higher success rates than comparison methods in all cases and real-time capability.

In the future, we will explore Runtime SES planning applied to robots with high degree of freedom and obstacles with complex geometry. Due to the use of fast MC simulations and a tree-based planning method, we expect planning for high degree of freedom robots to be straightforward with reduced geometry models to capture complex obstacles. Also, comparisons to other planning methods that consider obstacle interaction such as [11] and [12], and pedestrian navigation are planned. Lastly, parameter sensitivity and behavior analysis of our custom RRT could reveal insights regarding its improved performance over traditional RRT.

VI. Acknowledgments

We thank Jur Van Den Berg for feedbacks on modifying RVO2, Steven Cutlip for the implementation of SR sets and John Baxter for many discussions. This work is partially supported by National Science Foundation Grant IIS-1528047.

References

[1] Glover, W., Lygeros, J.: A stochastic hybrid model for air traffic control simulation. In: Hybrid Systems: Computation and Control. Springer (2004) 372–386

[2] Kruse, T., Pandey, A.K., Alami, R., Kirsch, A.: Human-aware robot navigation: A survey. Robotics and Autonomous Systems **61**(12) (2013) 1726–1743

[3] Roy, N., Gordon, G., Thrun, S.: Planning under uncertainty for reliable health care robotics. In: Field and Service Robotics, Springer (2003) 417–426

[4] Kolski, S., Ferguson, D., Stachniss, C., Siegwart, R.Y., Siegwart, R.Y., Siegwart, R.Y.: Autonomous driving in dynamic environments. ETH-Zürich (2006)

[5] Chiang, H.T., Malone, N., Lesser, K., Oishi, M., Tapia, L.: Aggressive moving obstacle avoidance using a stochastic reachable set based potential field. In: Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR). (2014)

[6] Chiang, H.T., Rackley, N., Tapia, L.: Stochastic ensemble simulation motion planning in stochastic dynamic environments. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE (2015) 3836–3843

[7] Abate, A., Prandini, M., Lygeros, J., Sastry, S.: Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. Automatica (2008) 2724–2734

[8] van den Berg, J., Patil, S., Sewall, J., Manocha, D., Lin, M.: Interactive navigation of multiple agents in crowded environments. In: Proceedings of the 2008 symposium on Interactive 3D graphics and games, ACM (2008) 139–147

[9] Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. Physical review E **51**(5) (1995) 4282

[10] Ferrer, G., Garrell, A., Sanfeliu, A.: Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE (2013) 1688–1694

[11] Ferrer, G., Sanfeliu, A.: Proactive kinodynamic planning using the extended social force model and human motion prediction in urban environments. In: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, IEEE (2014) 1730–1735

[12] Trautman, P., Krause, A.: Unfreezing the robot: Navigation in dense, interacting crowds. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, IEEE (2010) 797–803

[13] Massari, M., Giardini, G., Bernelli-Zazzera, F.: Autonomous navigation system for planetary exploration rover based on artificial potential fields. In: Proceedings of Dynamics and Control of Systems and Structures in Space (DCSSS) 6th Conference. (2004)

[14] Dellago, C., Posch, H.: Lyapunov exponents of systems with elastic hard collisions. Physical Review E **52**(3) (1995) 2401

[15] Yen, H.C., Huang, H.P., Chung, S.Y.: Goal-directed pedestrian model for long-term motion prediction with application to robot motion planning. In: Advanced robotics and Its Social Impacts, 2008. ARSO 2008. IEEE Workshop on, IEEE (2008) 1–6

[16] Ferguson, D., Stentz, A.: Anytime rrts. In: Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, IEEE (2006) 5369–5375

[17] Althoff, D., Wollherr, D., Buss, M.: Safety assessment of trajectories for navigation in uncertain and dynamic environments. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE (2011) 5407–5412

[18] Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. The International Journal of Robotics Research **17**(7) (1998) 760–772

[19] Guzzi, J., Giusti, A., Gambardella, L.M., Theraulaz, G., Di Caro, G.A.: Human-friendly robot navigation in dynamic environments. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE (2013) 423–430

[20] Khoshelham, K., Elberink, S.O.: Accuracy and resolution of kinect depth data for indoor mapping applications. Sensors **12**(2) (2012) 1437–1454

[21] LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. **20**(5) (May 2001) 378–400

[22] Tsai, Y.C., Huang, H.P.: Motion planning of a dual-arm mobile robot in the configuration-time space. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, IEEE (2009) 2458–2463

[23] Frazzoli, E., Dahleh, M.A., Feron, E.: Real-time motion planning for agile autonomous vehicles. Journal of Guidance, Control, and Dynamics **25**(1) (2002) 116–129

[24] van den Berg, J., Guy, S.J., Snape, J., Lin, M.C., Manocha, D.: Rvo2 library: Reciprocal collision avoidance for real-time multi-agent simulation `http://gamma.cs.unc.edu/RVO2/`.

[25] Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics Research: The 14th International Symposium ISRR. Springer (2011) 3–19

[26] Owen, A.B.: Monte Carlo theory, methods and examples. (2013)

[27] Ferrer Mínguez, G., et al.: Social robot navigation in urban dynamic environments. (2015)

[28] Chan, N., Kuffner, J., Zucker, M.: Improved motion planning speed and safety using regions of inevitable collision. In: 17th CISM-IFToMM symposium on robot design, dynamics, and control. (2008) 103–114

[29] Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research **30**(7) (2011) 846–894