Stochastic Ensemble Simulation Motion Planning in Stochastic Dynamic Environments

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

***Abstract*— Motion planning in stochastic dynamic environments is difficult due to the need for constant plan adjustment caused by the uncertainty of the environment. Many important motion planning problems however, including flight coordination human interacting robots and autonomous vehicles that require an algorithm to predict obstacle motion and plan safely. In this paper, we propose Stochastic Ensemble Simulation (SES)-based planning, a novel framework to efficiently predict and produce safe trajectories in the presence of stochastic obstacles. The stochastic obstacles can be introduced in several ways including stochastic motion or position/speed uncertainty. SES-based planning works by first predicting an obstacle's future position offline through an ensemble of Monte Carlo simulations. This simulates the stochastic obstacle dynamics and store the simulation results in temporal snapshots of predicted positions. An online planner then uses this output to identify a predicted collision-free direct path to the goal. If a direct path toward the goal is not expected to be collision-free, a more expensive, but flexible tree-based planner is used. Our experiments show SES-based planning outperforms other methods that have high planning success rate in environments with 900 stochastically moving obstacles. Furthermore, our method plans trajectories with an 80% success rate for a 7 DOF robot in an environment with 250 stochastic moving obstacles and 50 obstacles with speed/position uncertainty. This complex problem is currently beyond the capability of several comparison methods.**

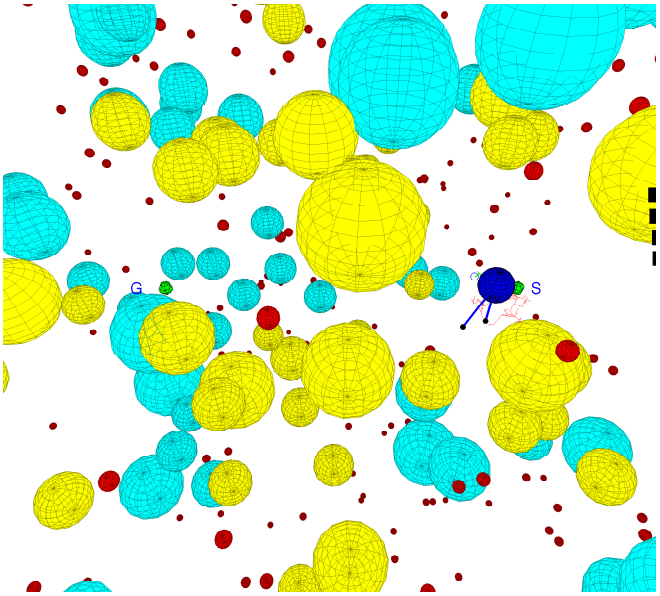


Fig. 1: 7 DOF robot in a 3D environment with 300 stochastic obstacles: The 200 red spheres are stochastically moving obstacles with linear trajectories. The cyan/yellow spheres (50 each) are random walk/noisy sensor obstacles. The blue robot must navigate from the right green sphere (S) to the left one (G).

I. Deviation from the IROS Paper

This report is modified from the IROS paper and I highlighted the key differences here: 1) Every section has slight modification in phrasing, including the abstract (you have to read carefully to find those, they are minor but in my opinion, make it more readable). 2) The related work section more detailed. 3) A paragraph about ICS is added to the preliminaries section. 4) Comparison in implementation between SES and SR is added to the methods section. 4) A brief explanation of how the global path is generated is added to the methods section. 5) The conclusion section is replaced by future work.

II. Introduction

Motion planning in dynamic environments is critical in several applications such as flight coordination, autonomous vehicles and human interacting robots. Typical approaches plan in these environments by dynamic adjustments that account for moving obstacles. Performing these adjustments often proves computationally expensive. In addition, several methods incur an increased computational expense due to the complexity of the robot shape and/or degrees of freedom (DOF) [1] [2].

Velocity obstacle-based planning [3] computes velocity obstacles, i.e., the set of possible robot velocities that lead to collisions while considering both the obstacle and robot dynamics. It has been very successful in the presence of multiple moving obstacles, and techniques such as linear programming can be used to speed up computation [2]. However, it can be intractable in cases where the geometry of the robot and obstacles are complex or if the robot has high DOF. There is also a large extra cost if obstacles move in a stochastic manner [4].

Several other approaches apply sampling-based methods in dynamic environments [5], [6], [7], [8]. These methods update plans dynamically, however updates only consider the current environment state but do not predict positional changes. Other methods consider a joint state-time space for sampling-based planning [9], [10]. By considering time as an extra dimension, the future motions of the obstacles can be fully considered, thus enabling the planner to avoid regions of inevitable collision. However, these methods do

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

not consider stochastic obstacles.

In this paper, we propose Stochastic Ensemble Simulation (SES)-based planning, a novel method for approximating the motions of stochastically moving obstacles in the workspace and then combining these approximations with motion planning in the robot's configuration space. This combination of methods allows for planning, even with high DOF robots, in the presence of stochastically moving obstacles and/or stochastic uncertainty in obstacle position. The method works by first predicting stochastic obstacle motions and/or positions through Monte Carlo (MC) simulations. Multiple MC runs are used to produce an ensemble of simulated predictions capturing potential future positions of a single obstacle. While the full predictions capture the underlying stochastic obstacle dynamics, predictions are stored as temporal snapshots taken during the simulation of an ensemble of MC runs. By incorporating these snapshots into SES-based planning, the likelihood of collision with a stochastic obstacle can be quickly predicted. The choice of planning method is general, and we designed and demonstrated SES used by a dynamic tree-based planner in configuration space that is guided by a precomputed path free of collision with static obstacles.

The main contributions of SES include: 1) The stochastic ensemble simulation which is an inexpensive method to predict the likelihood of future states in a dynamic environment. It can capture a wide range of environmental uncertainty such as stochastically moving obstacles and modeled sensor uncertainty. 2) A SES-based planning method which utilizes the results from SES combined with a tree-based planner in a novel way to generate a safe trajectory in the robot's configuration space. 3) SES separates motion planning from dynamic environment prediction, allowing for a wide array of planners to be used for SES-based planning. 4) Evaluation of our method against the current state-of-the-art planning methods including Path-Guided APF-SR [11], ORCA [2] and Gaussian APF [12].

We tested SES planning in three different environments with various obstacle types and robot dynamics. Tests include holonomic and unicycle robots in 2D environments with 300 to 900 moving obstacles and "bug trap" and narrow corridor static obstacles. Also, we tested a 7 DOF robot in a crowded 3D environment with 300 moving obstacles with stochastic dynamics and modeled sensor uncertainty (Figure 1). Our results show SES planning has a higher planning success rate than comparison methods even for a high DOF robot.

III. Related Work

Velocity obstacles [3] are widely used for path planning with moving obstacles. It is simple to implement and suitable for fast online planning in environments with hundreds of moving obstacles. ORCA (Optimal Reciprocal Collision Avoidance) [2] extended this idea to multi-agent path planning and the velocity obstacle calculation was accelerated by linear programming. This allows it to be efficient enough to be implemented by mainstream video games. Additional velocity obstacle formulation incorporated arcing motion obstacles, allowing an algorithm to handle both linear and arcing motion obstacles [13]. Frameworks have also been proposed for planning with stochastically moving obstacles by iterating through all possible robot velocities in order to approximate the true velocity obstacle and find the optimal robot velocity [4]. The main drawback of velocity obstacle based collision avoidance is that the cost of velocity obstacle calculation becomes prohibitively high for obstacles with complex geometry and robots with high DOF. In addition, if obstacles move stochastically, the velocity space needs to be discretized and highly efficient tools such as linear programming can no longer be used to speed up the computation. This results in a running time inversely proportional to the fourth power of the velocity resolution [4].

Sampling-based methods have been used for motion planning in dynamic environments. These methods can easily handle complex robot geometry due to the use of sampling. Methods utilizing a lazy-Probabilistic Roadmap (PRM) to accommodate moving obstacles have also been used [5], [6]. Additional researchers have proposed a Rapidly-exploring Random Tree (RRT) like local tree-based planner guided by a precomputed global PRM [8]. The method presented in [7] pre-builds a RRT from the goal and extends to most parts of the environment. The tree is modified at runtime whenever obstacle changes are detected. All these methods plan according to the current environment and do not take the future of the environment into account. This results in the robot being blind to potential collisions and regions of inevitable collision.

State-time space was introduced by [14]. It also introduced an A* based method in 1D to avoid two obstacles. State-time space allows sampling-based planners to incorporate information about the future environment [9]. An RRT with τ-safety criterion, which ensures the path is collision-free for at least τ seconds in the future, was used to navigate robots around moving doors has been used [10]. The moving obstacles, however, are entirely deterministic in these methods. These methods assume a deterministic function for the algorithm to query the exact location of obstacles in the future.

Stochastic Reachability (SR) analysis has been used for collision avoidance. Probabilistic SR sets [15] describe the set of states in which a collision is guaranteed with a certain probability. In [16], the obstacles are modeled as random sets that evolve over time, whereas the desired target set is known and stationary. The probability of collision is computed using dynamic programming for each robot starting configuration and iterates backwards in time for all possible control input. The computation cost is thus exponential in robot's DOF. Recently, SR based methods have been developed for path planning with many stochastic moving obstacles: The method presented by [17] computes the SR-set offline and weights the PRM edges by querying the probability of collision from the set. Similarly, the methods in [1] and [11] compute the SR-set offline but use the probability of collision as a repulsive potential in an Artificial Potential Field (APF)-based method. The latter two methods are real-time capable and achieved very high success rates in highly

dynamic, cluttered and stochastic environments. However, these methods are only suitable for robots with low DOF since the cost of dynamic programming is exponential in robot's DOF.

Other APF-based methods predict stochastic obstacle motion through stochastic transfer kernels [18], [19]. These methods use the probability of obstacle occupation as repulsive APFs. They require explicit calculation of stochastic transfer kernels, i.e., how the probability of obstacle occupation changes in time under the stochastic obstacle dynamics. The calculation can be difficult and therefore limits the stochastic obstacle dynamics to a Gaussian-biased random walk [18] or empirically determined heuristics [19]. Our method bypasses this difficulty by an efficient ensemble of MC simulations, thus allowing a wide range of stochastic obstacle dynamics. In addition, the repulsive APFs in these methods are time-averaged for all snapshots in the future, thus blocking an unnecessarily large amount of workspace.

IV. Preliminaries

The Degree of Freedom (DOF) of a robot is defined as the number of parameters required to completely describe it's configuration. An example of these parameters are the positions, link angles and link displacements of the robot. The space of all possible configuration parameters (feasible or infeasible) is called configuration space (C-space) [20]. Any physical robot and obstacle exist in the 3D universe (workspace), however the C-space can be high dimensional for robots with many linkages (e.g., high DOF). This high dimensional C-space can be classified as C-free, i.e., robot configurations not in collision, or C-obstacle, i.e., robot configuration in collision. A robot configuration (a point in C-space) in a given environment can be classified by collision detection between the robot and obstacles in the workspace. Thus, motion planning consists of identifying a trajectory in C-free from the start to the goal configuration.

Motion planning in dynamic environments involves obstacles moving in workspace according to certain deterministic or stochastic dynamics f. The future configuration (position, orientation, etc) of an obstacle can only be predicted exactly in the former case. The stochastic obstacle dynamics can be intrinsic to the obstacle, such as a linearly moving obstacle (the direction of f is fixed) with speed ($|f|$) randomly sampled from a distribution every T_{sample} second, or a random walking obstacle with fixed $|f|$ but the direction is randomly sampled every T_{sample} second. Sensor error may also introduce stochastic obstacle dynamics. For example, an obstacle may move in a fixed direction and speed, but the measured obstacle speed/position f could demonstrate noise, e.g., a value from a Gaussian distribution centered around the true speed/position.

Inevitable Collision States (ICS) refers to configurations in C-space that lead to collision with probability one. This can be caused by the robot's dynamic constraints and obstacle dynamics, e.g., slow robot cannot avoid a giant approaching trash compactor. A motion planner in dynamic environments should avoid guiding the robot into ICS as well as configurations in collision since the robot will be in collision with certainty in the future.

V. Methods

SES-based planning has two components: the offline SES component and the online planning component. First, SES uses MC simulations to produce an ensemble of potential future configurations of a stochastic obstacle. The ensemble is then used by the online planning in order to quickly find the likelihood of collision between the robot and the obstacle in the robot's configuration space. Then, we show SES integrated with an online planning method that plans in a limited region around the robot's current position through the use of tree-based planning and a precomputed path for guidance to the goal. By separating stochastic workspace changes from configuration space motion planning, SES provides several benefits: 1) A wide range of complex, stochastic planning environment changes such as stochastically moving obstacles and sensor error can be approximately predicted with little cost. 2) planning is general and any efficient planning method can be employed to use the SES output for collision prediction. 3) SES planning generates trajectories with much higher success rate than current state of the art methods.

A. Stochastic Ensemble Simulation (SES)

SES produces predictions of future positions of stochastically moving obstacles through an ensemble of MC runs as shown in Algorithm 1. Since each MC run produces a single obstacle trajectory prediction, an ensemble of MC runs is required. Each run starts with a single obstacle placed at the origin (line 2). As the MC runs progress, divergence between each obstacle position prediction will occur according to the stochastic dynamics described by $updateEnviornmentDynamics$ (line 5-6). The simulation time step δ in a run should be small enough to capture the system dynamics. The simulated obstacle position predictions are stored as temporal snapshots every planning time step Δ (line 7). After all MC runs have finished evolving to the simulation time horizon $\mathcal{T}$, the ensemble of runs stored in each of the temporal snapshots are processed.

One option for processing the ensemble of runs for a single obstacle is to create an approximate map of obstacle occupation likelihoods. This can be done by defining an indicator variable P for each grid point in the workspace and each MC run i in the ensemble:

$$P(x,y,i) = \begin{cases} 1 & \text{If } (x,y) \text{ is occupied by obstacles} \\ & \text{in the } i^{th} \text{ MC run.} \\ 0 & \text{Otherwise.} \end{cases}$$

The approximate probability of obstacle occupation is therefore an average over all MC runs:

$$P_{occupation}(x,y) = \frac{1}{N}\sum_{i=1}^{N} P(x,y,i) \tag{1}$$

A large variety of stochastic obstacles can be captured efficiently by SES through $updateEnviornmentDynamics$

Algorithm 1 SES

Input: Current obstacle configuration $x(0)$, Simulation time horizon $\mathcal{T}$, Simulation time step δ, Size of Ensemble N, Planning time resolution Δ

Output: Future obstacle configuration snapshots: $x_E \equiv \{x_E(0), x_E(\Delta), x_E(2\Delta), ..., x_E(\mathcal{T})\ \}$

---

for $i = 0$; $i < N$; i++ **do**
  $\bar{x} = x(0) = x_i(0)$
  $k = 1$
  for $t = 0$; $t <= \mathcal{T}$; $t = t + \delta$ **do**
    $f_i = updateObstacleDynamics(\bar{x}, t)$
    $\bar{x} = \bar{x} + \Delta f_i$
    if $t = k\Delta$ **then**
      $x_i(k) = \bar{x}$
      k++
    end if
  end for
end for
for $m = 0$; $m <= \mathcal{T}/\ \Delta$; m++ **do**
  $x_E(m\Delta) = processEnsemble(x_1(m), ...x_N(m))$
end for

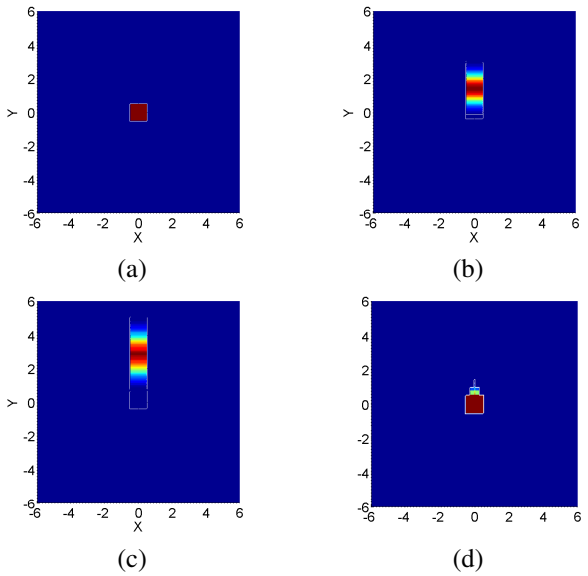


Fig. 2: Predictions of obstacle occupation (red=high likelihood, blue=low/no likelihood) for a square obstacle traveling in a linear trajectory with stochastic speeds as specified in Section VI. (a-c) SES ensemble (500 runs) at time snapshots (a) $t = 0s$, (b) $t = 4s$, (c) $t = 8s$. (d) An SR probability of collision considering the holonomic robot dynamics with speed 0.36 unit/s.

(several were covered in Section IV). An ensemble of 500 MC runs from SES is shown in Figure 2(a)-(c) for moving obstacle in a linear trajectory with a stochastic speed.

It is worth mentioning that SES is much faster than the stochastic reachability (SR) analysis [15] since it approximates the future position distribution of obstacles and does not iterate through all possible robot actions. The map shown in figure 2(a)-2(c) takes 21 seconds to generate. A SR analysis (shown in figure 2(d)) of identical grid resolution for a holonomic robot finishes in 3992 seconds. In addition, SES is more flexible and easier to implement due to the Monte Carlo simulation nature. Modifications to $updateEnviornmentDynamics$ can easily capture hybrid obstacle dynamics such as an obstacle dynamics that switches between line and arc motion with a time varying probability.

B. SES-based Planning

Algorithm 3 growGoalTree

Input: Future workspace snapshots: x_E, Robot Current State x_r, Temporary Goal $tempGoal$, Planning Time Step Δ

Output: $growFullTree$, Tree: τ

$conf = x_r$; $k = 1$; $\tau = null$;
$growFullTree = false$
while $dist(conf, tempGoal) > 0.5$ **do**
  $conf = conf + \Delta \cdot getActionToGoal(conf, tempGoal)$
  $collProb = getCollisionProb(conf, x_E, k\Delta)$
  if $collProb <= Pr_{accept}$ **then**
    $\tau.addToTree(conf)$
    k++
  else
    $growFullTree = true$
    $return$
  end if
end while

SES provides an occupation likelihood in the workspace for a single obstacle. This could be used directly as a likelihood of collision in order to identify paths with a higher likelihood of safety, e.g., [21] in a lazy-PRM manner. However, the choice of planning method can be critical to planning success. First, since the stochastic motions become more divergent over longer timescales, e.g., less predictive, it is helpful to plan the robot's motions for a short time frame

Algorithm 2 SES-based planning

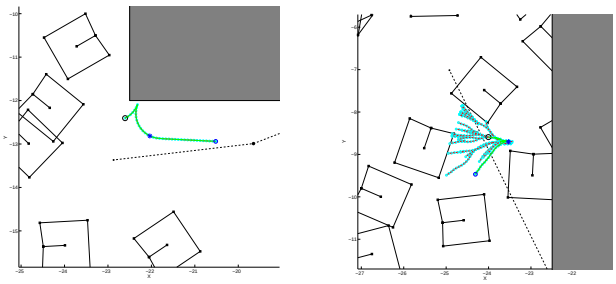
Input: Future obstacle configuration snapshots: $x_E \equiv \{x(0), x(\Delta), x(2\Delta), ..., x(\mathcal{T})\}$, Guidance Path $\mathcal{P}_\mathcal{G}$, Robot Current State x_r, Planning Time step Δ

1: **for** $t = 0$; $t < maxTime$; $t = t + \Delta_{exe}$ **do**
2: **if** $reGrowTree == true$ **then**
3: $Tree : \tau = null$
4: $t_{lastPlan} = t$
5: $tempGoal = getPathGuidanceGoal(x_r, \mathcal{P}_\mathcal{G})$
6: $(growFullTree, \tau) = growGoalTree(x_r, tempGoal, x_E)$
7: **if** $growFullTree == true$ **then**
8: $\tau = growFullTree(x_r, tempGoal, \tau, x_E)$
9: **end if**
10: $\mathcal{P} = getPathFromTree(\tau)$
11: **end if**
12: $x_r = x_r + \Delta_{exe} \cdot getAction(\mathcal{P}, t - t_{lastPlan})$
13: **if** $nodeReached == true$ **then**
14: $reGrowTree = checkFutureNodes(\mathcal{P}, x_r, t, tempGoal)$
15: **end if**
16: **end for**

and continuously update the plan as obstacles are moving. Second, it is helpful to plan according to a time step Δ equal to the one used to generate the snapshots for maximum obstacle position accuracy.

In order to address these requirements, our SES-based planner utilizes a tree-based planning method with planning time step Δ (equal to the snapshots time step) in Algorithm 2. Each node in the tree corresponds to a particular snapshot, and can therefore be checked for potential collision. This allows SES-based motion planning to operate in C-space while checking for collision of the robot and the obstacles in the workspace. In order to guide the local trees following the robot to the goal, we use a precomputed global path from the start to the goal that is free of collision with static obstacles. This global path can be generated by a variety of method. We chose a simple uniform sampling PRM in our experiments. The SES planning starts by querying the global path to find a temporary goal (nodes of PRM generated path) based on the current robot state (Algorithm 2 line 5).

SES-based planning then attempts to grow a tree *directly* toward the temporary goal in $growGoalTree$ (Algorithm 3 and shown in Figure 3(a)). This subroutine finds the direct action toward the temporary goal and evolves a tentative configuration $conf$ toward it. Each new iteration of $conf$ (line 4), with its unique corresponding time $k\Delta$, ($k \in \mathbb{N}$) in the future, is checked for potential collision by $getCollisionProb$, which uses the temporal snapshots of the predicted obstacle occupation given by SES (line 5). The returned collision likelihood, $collProb$, is the sum of $P_{occupation}$ (from (1) in the obstacle's relative coordinate) considering obstacles within range $d_{horizon}$. If $collProb$ is



(a) Goal Tree (b) Full Tree

Fig. 3: Tree building with SES predictions for a unicycle robot (gray shaded=static obstacle, square outlines=stochastic moving obstacles in linear trajectory, green/blue=tree built from robot's current position, an asterisk). (a) A goal tree plans a path directly toward the goal if SES predicts the path has no likelihood of collision, otherwise (b) a full tree is grown. Note that the RRT does not extend in front of the obstacle left of the robot due to future predictions given by SES.

smaller than some threshold Pr_{accept}, $conf$ is added to the tree and the process continues until the temporary goal is reached. Otherwise, potential collisions are detected along the direct path and the subroutine returns with a flag to grow a full tree.

In the case that a direct path leads to potential collisions, a tree-based planning algorithm such as RRT [22] or EST [23] can be used for $growFullTree$ (shown in figure 3(b)). The ordinary collision checking is replaced by the same $getCollisionProb$ function from line 5 of Algorithm 3 in order to utilize the predictions given by SES. If the likelihood of collision is above Pr_{accept}, this tentative node is discarded, otherwise, the node is added to the tree and the likelihood of collision is stored.

To extract a safe path from the tree while progressing toward $tempGoal$, the $getPathFromTree$ subroutine computes a weight for each node i:

$$w(i) = \epsilon \cdot distanceToTempGoal(i) + \frac{collProb(i)_{accu}}{N(i)_{traversed}}, \quad (2)$$

where $collProb(i)_{accu}$ is the accumulative likelihood of collision from the root to node i; $N(i)_{traversed}$ is the number of nodes away from the root and ϵ is the greediness parameter. The subroutine picks the path with the lowest w and checks if

$$N(i)_{traversed} >= N_{safety} \quad (3)$$

is satisfied in order to ensure the τ-safety [24] criterion, i.e., this path is safe for at least $\tau = N_{safety}\Delta$ seconds in the future. Provided τ is big enough, this strategy greatly reduces the probability of a chosen path leading to an inevitable collision state since the robot has at least τ seconds to replan a safe trajectory at any time. A path is rejected if (3) is not satisfied and the node with next lowest w is checked. If all

nodes are eliminated, the algorithm chooses the longest path in the tree with $collProb(i)_{accu} = 0$.

Finally, the robot then executes the extracted path (line 12). Every time a node is reached, $checkFutureNodes$ does the same $getCollisionProbability$ query for the next N_{safety} nodes in the path. If any node has a likelihood of collision larger than Pr_{accept} or the remaining path is shorter than N_{safety}, the current tree is discarded and a new tree will be grown. This compensates for the approximate nature of SES by constantly incorporating new information about the current workspace, and allows the collision likelihood of a node to come from more than one snapshot as illustrated in Figure 4.

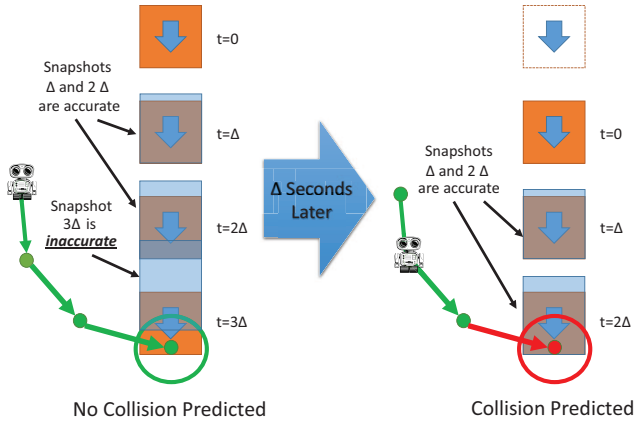


Fig. 4: A robot progressing along a path (green) with a linear stochastic obstacle (orange) predicted to be collision free (left). As he reaches the next node on his path (right, Δ seconds later), he updates the obstacle position and the SES likelihoods, therefore predicting a collision and indicating that replanning is needed.

VI. Experiments

To demonstrate the SES planning algorithm, we tested three different robots in three different environments with stochastically moving obstacles. To maintain a constant density of moving obstacles, we restrict the simulation environment of the robot and moving obstacles to a circular or a spherical radius 50. When an obstacle reaches the boundary, it is transported to the antipodal position with velocity vector unchanged. All methods, other than ORCA, were implemented in MATLAB. A C++ implementation of ORCA was downloaded and modified for a single robot with multiple obstacles from [25]. All experiments were run on a single core of an AMD FX-8320 at 3.5GHz with 16GB RAM. Experiments are repeated 100 times (10 times per bin, based on 10 initial guidance paths) in order to compute the average and standard deviation of success rate and path length.

A. Experiment 1: 2D Open Environment

Experiment 1 is designed to compare SES planning with Path-Guided APF-SR (PG-APF-SR) [11], Gaussian APF method [12] (abbreviated as Gaussian) with $\mathcal{N}(0, 0.15^2)$ and ORCA [2]. The environment is 2D and has 300 to 900 stochastically moving obstacles (no static obstacles).

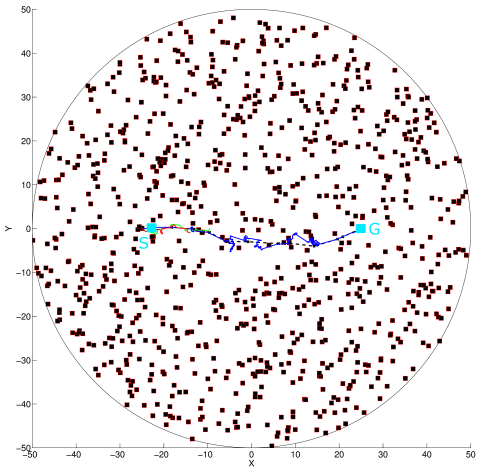
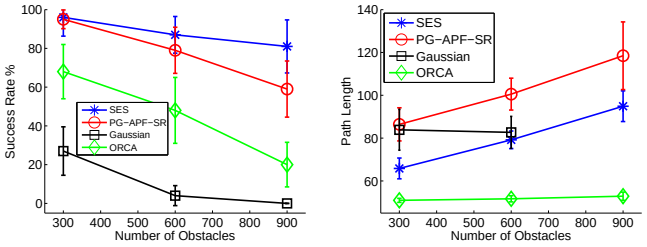


Fig. 5: 2D open environment with 900 stochastically moving obstacles (black squares). The robot starts at S (start) and must traverse to G (goal). The blue line is the trajectory of SES planning. Path-Guided APF-SR (red line) and Gaussian APF (green line) failed to reach the goal due to collision. The black dashed curve is the guidance path.

Setup: *Obstacles:* 300, 600 or 900 stochastically moving square obstacles with unit width. They move in a fixed direction but their speed is stochastically-sampled every T_{sample}=1 second. The set of possible speeds is $\{0.1, 0.2, 0.5, 0.7\}$ unit/s with probability $\{0.3, 0.2, 0.3, 0.2\}$. *Robot:* The robot is a holonomic point robot with a maximum speed of 0.36 unit/s. It starts at $(-25, 0)$ and the goal is at $(25, 0)$. PG-APF-SR and Gaussian have a goal bias of 0.01, and this environment (shown in figure 5), is identical with [11]. Ten guidance paths are generated by ten PRM roadmaps created offline with 1000 nodes and edges selected by connecting nodes to their 10 nearest neighbor connections. *SES parameters:* The simulation time horizon $\mathcal{T}$ in Algorithm 1 was set to 8 seconds, the simulation time step $\delta = 0.01$s, and the ensemble size $N = 500$. The planning time step Δ for SES planning was 0.2s. RRTs with a goal bias 0.05 and 1500 maximum $getCollisionProb$ queries per tree was used in $growFullTree$. We set $N_{safety} = 10$ to provide a 2 second τ-safety. The acceptance threshold Pr_{accept} was set to 1% and the greediness parameter $\epsilon = 0.001$. The robot is only aware of obstacles within range $d_{horizon} = 5.7$.

Results: Figure 6a shows the success rate of SES planning exceeds comparison methods. The high success rate was maintained even in the 900 obstacle extremely crowded environment (81%, which exceeds comparison methods by 32%). Gaussian and ORCA do not consider the stochastic changes of the environment and thus result in a low success rate (0 and 20%). In addition, the path length (shown in Figure 6b) of SES planning is consistently lower than Gaussian and PG-APF-SR. This is because SES's tree-based planner

does not have a local minima problem, unlike the APF methods. Therefore, it was not stringently held to guidance path following [11].



(a) Success rate (b) Path length

Fig. 6: Performance comparison between SES planning, Path-Guided APF-SR, Gaussian APF method, and ORCA. The success rate of Gaussian with 900 obstacles was lower than 1%, and therefore the path length was not calculated.

Table I shows the average running time per planning step for all methods are in the same order of magnitude. Given the planning time step Δ=0.2s for SES planning, our method is online-capable. SES planning achieved this because it only grows a tree when a potential collision is detected or the robot has traversed near the end of a tree. In addition, it attempts to grow a goal tree *directly* toward the goal if possible, and only grow the expensive full RRT if necessary. Our experiment in the 900 obstacle environment utilized 244±205 tree growth calls per trajectory and 27% of them are the inexpensive $growGoalTree$.

# of Obstacles	SES	PG-APF-SR	Gaussian	ORCA
300	6.0±26.4	4.6±1.8	4.3±1.6	5.1±0.1
600	15.8±44.9	7.5±1.8	7.6±1.3	11.0±0.4
900	44.6±77.7	10.1±1.3	9.3±1.4	16.0±1.1

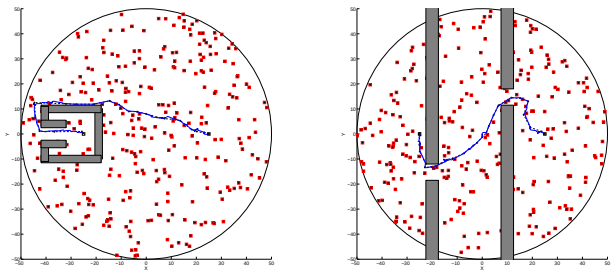
TABLE I: Average running time per planning step in a 2D free environment in milliseconds

B. Experiment 2: Unicycle Robot in Environment with Static Obstacles

Experiment 2 involves an unicycle robot in a 2D environment with either bug trap or narrow corridor static obstacles in order to demonstrate the ability for SES to handle nonholonomic robot dynamics and the integration with guidance paths.

Setup: *Obstacles:* A bug trap environment (Figure 7a), with an escape opening 5 times the width of moving obstacles and a narrow corridor environment (Figure 7b) with openings of the same size were tested. 300 moving obstacles (described in Experiment 1) are also present. *Robot:* The robot has unicycle dynamics with a maximum turn rate of $\pi/5$ per second and the maximum speed is the same as Experiment 1. *SES parameters:* The same as Experiment 1. All static obstacles have the stochastic dynamics $f = 0$ since they are stationary.

Results: Table II shows that the more restrictive nonholonomic robot dynamics and complex static terrain do not impact the success rate severely for SES planning. This is expected since tree-based methods were originally developed to tackle nonholonomic and kinodynamic planning problems [22]. Methods without any prediction such as Gaussian have a significantly lower success rate.



(a) Bug trap (b) Narrow corridor

Fig. 7: Unicycle robot in environments with (a) bug trap and (b) narrow corridor static obstacles. 300 stochastically moving obstacles are shown as black squares with red outline. The black line (under) is the guidance path and the blue line (over) is the actual path take by the robot with SES planning

Environment	SES	PG-APF-SR	Gaussian
Narrow Corridor	89±8%	88±9%	7±1%
Bug Trap	85±11%	79±18%	2±0%

TABLE II: Success rate comparison of unicycle robots in environments with static obstacles.

C. Experiment 3: 7 DOF Robot in a crowded 3D environment

Experiment 3 demonstrates the algorithm's ability to plan for high DOF robots in the presence of stochastically moving obstacles and modeled sensor uncertainty (which is currently beyond the capability of SR or velocity obstacle based methods). A 7 DOF holonomic robot must navigate through a crowded 3D environment with 250 stochastically moving obstacles and 50 moving obstacles with position/speed uncertainty.

Setup: *Obstacles:* The radius 50 spherical environment contains 300 moving spherical obstacles. 200 of these obstacles have a radius of 0.5 and exhibit linear motion with speed sampled every 1 second from the set $\{3.125, 12.5\}$ with probability $\{0.5, 0.5\}$ (up to 4 times faster than the robot). 50 random walk obstacles of radius 3.5 have a speed of 1 but their direction is selected randomly every T_{sample}=1 second. The last 50 obstacles have a radius of 3.5 and move with a constant speed (1 unit/s) and direction. To simulate sensor uncertainty, however, the robot samples the position and speed of these obstacles by a Gaussian distribution of 10% standard deviation around their true position and speed. *Robot:* The robot consists of 2 side spheres of radius 0.4 connected to a center sphere of radius 2.65. The length of connections are 3 times the radius of the center sphere. The robot is holonomic with a maximum speed of 3.125

and the linkages can rotate freely in any direction. The robot therefore has 7 DOF in the configuration space (7 parameters for x,y,z coordinates for the center sphere, polar and azimuthal angles of two side spheres). We compare SES with a naive APF method [26] with a potential cutoff of 4 units. *SES parameters:* The same as Experiment 1, except the robot is aware of obstacles within range $d_{horizon} = 50$.

Results: As shown in table III, SES planning is able to plan trajectories of higher success rate (about 30% higher) than the comparison APF method, with less average running time per planning step. This indicates SES planning is able to generate safe trajectories even for for a high DOF robot in a crowded dynamic environment, with stochastic obstacles (motions and postion/speeds).

	SES	Naive APF
Success Rate	80±17%	53±11%
Running Time	147 ±90 ms	160 ±15 ms

TABLE III: Performance comparison for a 7 DOF robot in an environment with 300 stochastic obstacles.

D. Discussion

The 2D environments in Experiments 1 and 2 clearly show that SES planning has a very high success rate and is capable of online planning despite the large number of stochastically moving obstacles. In all cases, SES planning has a higher planning success rate, and the running time per planning step is in the same order of magnitude of Path-Guided APF-SR, Gaussian APF and ORCA. The 3D environment shows the method can successfully find trajectories for a high DOF robot in the presence of stochastically moving obstacles and obstacles with uncertain position/speed, while having a runtime cost comparable to a low cost APF method. Also, it was able to achieve a higher success rate on the high-dimensional problem, a problem Path-Guided APF-SR and ORCA could not address.

SES-based planning reduces the number of expensive $growFullTree$ calls by planning directly toward the goal ($growGoalTree$) when such a path is safe. Table IV shows this happens 26% of the time, even in the crowded 900 obstacle environment. In the 900 obstacles environment, the robot traversed the path in 110±16.8 seconds while calling $growFullTree$ 189±157 times. Therefore, each $growFullTree$ call has a real-time budget of 582 ms on average (110 s/189 calls), an abundance compared to the 44.6 ms average running time per planning step. In addition, since SES-based planning does not restrict to a specific tree-based planner, techniques to speed-up tree-based methods for real-time purposes such as [27] and [28] can be used in order to meet real-time constraints. We also tried using EST [23] instead of RRT and the resulting success rate and path lengths were mostly within one standard deviation as shown in Table V.

SES is an approximate method to predict stochastic environments. Empirical analysis shows that the success rate is not highly sensitive to the size of the ensemble (which determines the quality of the approximation). Table VI shows that the success rates are within one standard deviation for a wide range of ensemble sizes. This is likely because the $checkFutureNode$ subroutine compensated for the poor approximation quality using methods described in section V.

# of Obstacles	growGoalTree Ratio	Total grow tree calls
300	67.8±21.4%	107±136
600	46.0±24.8%	142±143
900	26.3±23.3%	257±205

TABLE IV: The ratio of growGoalTree calls and total number of grow tree calls (full+goal). The setup is SES in Experiment 1.

# of obstacles	RRT Success %	RRT PL	EST Success %	EST PL
300	94±10%	67±5	94±5%	71±4
600	87±9%	79±4	80±13%	91±6
900	81±14%	95±7	81±14%	88±8

TABLE V: Performance comparison between RRT and EST as the tree-based planner. (PL=Path Length)

Size of Ensemble	Success Rate
100	$76 \pm 15\%$
500	$81 \pm 14\%$
1000	$80 \pm 12\%$

TABLE VI: Success rate for various SES ensemble sizes. The setup is Experiment 1 with 900 moving obstacles.

VII. Future Work

A. Parameter Investigation

SES planning has several parameter such as Pr_{accept}, ϵ, N_{safety}, Δ and $d_{horizon}$. These parameters are currently hard-coded and set to empirically determined values. However, parameters can be related to each other. For example, $d_{horizon}$ should be larger than $N_{safety} \times \Delta \times$ maximum robot speed, otherwise the robot may not see the obstacle before it's too late. A systematic parameter relationship investigation help reduce the number of tweak-able parameters or give bounds to parameters. In addition, it would be interesting to see if SES is sensitive to certain parameters and how the parameters differ in different environments.

B. Adaptive Node Acceptance Probability

Although SES planning is designed to choose the path that balance progression toward the goal and path safety with equation 1, our choice of setting the node acceptance probability $Pr_{accept} = 0.01$ in our experiments limited the exploration of escaping paths with finite collision probability. This bias the algorithm to explore regions with very low probability of collision efficiently as samples with collision likelihood larger than 0.01 are discarded. However, there may exist scenarios where all escape routes have probability of collision larger than 0.01. Such escape path will never be found with Pr_{accept} set to a low value such as 0.01. On the other hand, if Pr_{accept} is set to a high value such as 0.5,

the algorithm may spend a large portion of time building a tree with high likelihood of collision, reducing the safety of generated path (regions with very low or zero likelihood of collision is less explored). Therefore, an adaptive node acceptance probability could explore regions of potential collision when the tree length is short by increasing the node acceptance probability, and otherwise focus on exploring regions with no probability of collision.

C. Reduce SES Query Cost

shrink wrap and total prunning We mentioned one option in Section V for $processEnsemble$. The approximate map of obstacle occupation likelihood approach allows the planner to take an escaping path with non-zero collision probability. However, in an environment with fewer moving (hence plenty of escape path with zero collision probability) but more obstacles types (i.e., many distinct obstacle dynamics and shapes), the obstacle occupation likelihood map approach may take a long time to precompute all obstacle types and requires lots of memory at runtime. A possible alternative is, after the ensemble simulation generated the position for each simulated obstacle at each temporal snapshot, a 3D polygon is generated to "shrink wrap" every simulated obstacle in the ensemble. This can be done offline by 3D graphics software such as Blender. This polygon represents the region of non-zero obstacle occupation probability, and the $getCollisionProb$ subroutine becomes a simple CD query between the robot and the shrink wrapping polygon. This reduces the cost of SES planning query if the robot has a complex geometry and must be described by a large number of control points.

D. Obstacle-Obstacle and Obstacle-Robot Interaction

The insensitivity of SES planning to ensemble size may enables the possibility of running SES *online* in order to plan in environments with stochastic obstacle-obstacle and obstacle-robot interaction. This is extremely challenging since the dimensionality of the configuration space for obstacles grows exponentially with the number of interacting obstacles, thus rendering reachability and velocity obstacle based approach infeasible. Our method, with the sampling-based nature (for both simulating and planning) and the insensitivity to ensemble size may be a viable approach.

REFERENCES

[1] 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)

[2] van den Berg, J., Guy, S.J., Lin, M.C., Manocha, D.: Reciprocal n-body collision avoidance. In: INTERNATIONAL SYMPOSIUM ON ROBOTICS RESEARCH. (2009)

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

[4] B.Kluge: Recursive agent modeling with probabilistic velocity obstacles for mobile robot navigation among humans. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS). (2003) 376–380

[5] Jaillet, L., Simeon, T.: A PRM-based motion planner for dynamically changing environments. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS). (2004)

[6] Kallman, M., Mataric, M.: Motion planning using dynamic roadmaps. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (2004) 4399–4404

[7] M.Otte, E.Frazzoli: Rrt-x: Real-time motion planning/replanning for environments with unpredictable obstacles. In: Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR). (2014)

[8] Rodriguez, S., Lien, J.M., Amato, N.M.: A framework for planning motion in uncertain environments. Technical Report TR06-010, Parasol Lab, Dept. of Computer Science, Texas A&M University (Sep 2006)

[9] Hsu, D., Kindel, R., Latombe, J.C., Rock, S.: Randomized kinodynamic motion planning with moving obstacles. Int. J. Robot. Res. **21**(3) (March 2002) 233–255

[10] Frazzoli, E., Dahleh, M.A., Feron, E.: Real-time motion planning for agile autonomous vehicles. In: American Control Conference, 2001. Proceedings of the 2001. Volume 1., IEEE (2001) 43–49

[11] Chiang, H.T., Malone, N., Lesser, K., Oishi, M., Tapia, L.: Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (2015) To appear

[12] Massari, M., Giardini, G., Bernelli-Zazzera, F.: Autonomous navigation system for planetary exploration rover based on artificial potential fields. In: Dynamics and Cont. of Systems and Structures in Space (DCSSS). (2004) 153–162

[13] Large, F., Sckhavat, S., Shiller, Z., Laugier, C.: Using non-linear velocity obstacles to plan motions in a dynamic environment. In: Control, Automation, Robotics and Vision. (2002) 734–739

[14] Fraichard, T.: Trajectory planning in a dynamic workspace: a'state-time space'approach. Advanced Robotics **13**(1) (1998) 75–94

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

[16] Summers, S., Kamgarpour, M., Lygeros, J., Tomlin, C.: A stochastic reach-avoid problem with random obstacles. In: Proc. Int. Conf. Hybrid Sys.: Comp. and Cont. (HSCC). (2011) 251–260

[17] Malone, N., Lesser, K., Oishi, M., Tapia, L.: Stochastic reachability based motion planning for multiple moving obstacle avoidance. In: Hybrid Systems: Computation and Control, HSCC (2014) 51–60

[18] Nam, Y.S., Lee, B.H., Kim, M.S.: View-time based moving obstacle avoidance using stochastic prediction of obstacle motion. In: Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on. Volume 2., IEEE (1996) 1081–1086

[19] Tadokoro, S., Hayashi, M., Manabe, Y., Nakami, Y., Takamori, T.: On motion planning of mobile robots which coexist and cooperate with human. In: Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on. Volume 2., IEEE (1995) 518–523

[20] Lozano-Pérez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Communications of the ACM **22**(10) (October 1979) 560–570

[21] Malone, N., Manavi, K., Wood, J., Tapia, L.: Construction and use of roadmaps that incorporate workspace modeling errors. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS). (2013) 1264–1271

[22] LaValle, S.M., Kuffner, J.J.: Rapidly-exploring random trees: Progress and prospects. In: New Directions in Algorithmic and Computational Robotics. A. K. Peters (2001) 293–308 book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Hanover, NH, 2000.

[23] Hsu, D., Latombe, J.C., Motwani, R.: Path planning in expansive configuration spaces. Int. J. Comput. Geom. & Appl. (1999) 495–517

[24] E. Frazzoli, M.A. Dahleh, E.F.: Real-time motion planning for agile autonomous vehicles. In: American Control Conference. (2001) 43–49

[25] 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/`.

[26] Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (1985) 500–505

[27] Barraquand, J., Latombe, J.C.: Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. Algorithmica **10**(2-4) (1993) 121–155

[28] Bekris, K.E., Kavraki, L.E.: Greedy but safe replanning under kinodynamic constraints. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (2007) 704–710