Provably Safe and Scalable Unmanned Aerial Vehicle Routing: A Case Study in San Francisco and the Bay Area

Mo Chen*, Somil Bansal†, Ken Tanabe‡, and Claire J. Tomlin§

Provably safe and scalable multi-vehicle path planning is an important and urgent problem due to the expected increase of automation in civilian airspace in the near future. Hamilton-Jacobi (HJ) reachability is an ideal tool for analyzing such safety-critical systems and has been successfully applied to several small-scale problems. However, a direct application of HJ reachability to large scale systems is often intractable because of its exponentially-scaling computation complexity with respect to system dimension, also known as the "curse of dimensionality". To overcome this problem, the sequential path planning (SPP) method, which assigns strict priorities to vehicles, was previously proposed; SPP allows multi-vehicle path planning to be done with a linearly-scaling computation complexity. In this work, we demonstrate the potential of SPP algorithm for large-scale systems. In particular, we simulate large-scale multi-vehicle systems in two different urban environments, a city environment and a multi-city environment, and use the SPP algorithm for trajectory planning. SPP is able to efficiently design collision-free trajectories in both environments despite the presence of disturbances in vehicles' dynamics. To ensure a safe transition of vehicles to their destinations, our method automatically allocates space-time reservations to vehicles while accounting for the magnitude of disturbances such as wind in a provably safe way. Our simulation results show an intuitive multi-lane structure in airspace, where the number of lanes and the distance between the lanes depend on the size of disturbances and other problem parameters.

I. Introduction

Due to the recent surge of interest in the use of unmanned aerial systems (UASs) for civil applications such as package delivery, aerial surveillance, disaster response, among many others [1–5], civilian airspace may in the near future contain up to thousands of unmanned aerial vehicles (UAVs), potentially in close proximity of humans, other UAVs, and other important assets. As

*PhD Candidate, Department of Electrical Engineering and Computer Sciences, shared first author
†PhD Student, Department of Electrical Engineering and Computer Sciences, shared first author
‡Toshiba
§Professor, Department of Electrical Engineering and Computer Sciences, Member AIAA

a result, government agencies such as the Federal Aviation Administration (FAA) and National Aeronautics and Space Administration (NASA) of the United States are urgently trying to develop new scalable ways to organize an airspace in which potentially thousands of UAVs can fly together [6, 7].

One essential problem that needs to be addressed for this endeavor to be successful is that of trajectory planning: how a group of vehicles in the same vicinity can reach their destinations while avoiding situations which are considered dangerous, such as collisions. Many previous studies address this problem under different assumptions. In some studies, specific control strategies for the vehicles are assumed, and approaches such as those involving induced velocity obstacles [8–11] and involving virtual potential fields to maintain collision avoidance [12,13] have been used. Methods have also been proposed for real-time trajectory generation [14], for path planning for vehicles with linear dynamics in the presence of obstacles with known motion [15], and for cooperative path planning via waypoints which do not account for vehicle dynamics [16]. Other related work include those which consider only the collision avoidance problem without path planning. These results include those that assume the system has a linear model [17–19], rely on a linearization of the system model [20, 21], assume a simple positional state space [22], and many others [23–25].

However, to make sure that a dense group of UAVs can safely fly in the close vicinity of each other, we need the capability to flexibly plan provably safe and dynamically feasible trajectories without making strong assumptions on the vehicles' dynamics and other vehicles' motion. Moreover, any trajectory planning scheme that addresses collision avoidance must also guarantee both goal satisfaction and safety of UAVs despite disturbances caused by wind and communication faults [7]. Finally, the proposed scheme should scale well with the number of vehicles, as well as result in an intuitive airspace structure for humans to monitor and potentially adjust.

The problem of trajectory planning and collision avoidance under disturbances in safety-critical systems has been well-studied using Hamilton-Jacobi (HJ) reachability analysis, which provides guarantees on goal satisfaction and safety of optimal system trajectories [26–31]. Reachability-based methods are particularly suitable in the context of UAVs because of the hard guarantees that are provided. In reachability analysis, one computes the reach-avoid set, defined as the set of states from which the system can be driven to a target set while satisfying (possibly time-varying) state constraints at all times. A major practical appeal of this approach stems from the availability of modern numerical tools, which can compute various definitions of reachable sets [32–35]. These numerical tools, for example, have been successfully used to solve a variety of differential games, path planning problems, and optimal control problems. Concrete practical applications include aircraft auto-landing [36], automated aerial refueling [37], MPC control of quadrotors [38], and multiplayer reach-avoid games [39]. Despite its power, the approach becomes numerically intractable as the state space dimension increases. In particular, reachable set computations involve solving a HJ partial differential equation (PDE) or variational inequality (VI) on a grid representing

a discretization of the state space, resulting in an *exponential* scaling of computational complexity with respect to the dimensionality of the problem. Therefore, dynamic programming-based approaches such as reachability analysis are not directly suitable for managing the next generation airspace, which is a large-scale system with a high-dimensional joint state space because of the high density of vehicles that needs to be accommodated [7].

To overcome this problem, the Sequential Path Planning (SPP) method was proposed in [40]. Here, vehicles are assigned a strict priority ordering. Higher-priority vehicles plan their paths without taking into account the lower-priority vehicles. Lower-priority vehicles treat higher-priority vehicles as moving obstacles. Under this assumption, time-varying formulations of reachability [29,31] can be used to obtain the optimal and provably safe paths for each vehicle, starting from the highest-priority vehicle. Thus, the curse of dimensionality is overcome for the multi-vehicle path planning problem at the cost of a mild structural assumption, under which the computation complexity scales just *linearly* with the number of vehicles. Intuitively, SPP algorithm allocates a space-time trajectory to each vehicle based on their priorities. The highest-priority vehicle gets to choose any space-time trajectory that does not collide with static obstacles, such as the optimal trajectory. The next vehicle's trajectory must not intersect with the trajectory of the highest-priority vehicle, and so on. Hence two vehicles can either follow same state tarjectory but at different times (referred to as *time-separated* trajectories here on) or follow different state trajectories but at the same time (referred to as *state-separated* trajectories here on), but not both. Finally, they can have different state trajectories at different times (referred to as *state-time separated* trajectories here on). So by design, SPP algorithm ensures that the space-time trajectories of the vehicles do not intersect, and hence a safe transition to destination is guaranteed for all vehicles.

Authors in [41] and [42], respectively, extend SPP to the scenarios where disturbances and adversarial intruders are present in the system, resolving some of the practical challenges associated with the basic SPP algorithm in [40]. The focus of these works, however, have mostly been on the theoretical development of SPP algorithm. Our focus in this work is instead on demonstrating the potential of SPP algorithm as a provably safe trajectory planning algorithm for large-scale systems. In particular, our main contributions in this paper are as follows:

- We simulate large-scale multi-vehicle systems in two different urban environments under the presence of disturbances in vehicles' dynamics. First, the SPP algorithm is used for trajectory planning at a city level and then at a regional level. For city level planning, we consider the city of San Francisco in California, USA, and for regional level planning we consider a part of San Francisco Bay Area in California, USA. The main differences between these two case studies are that the city level planning needs to take into account static physical obstacles such as tall buildings, whereas the origins and destinations are farther apart at the regional level. In both cases, we demonstrate that SPP algorithm is able to design provably-safe trajectories despite the disturbances.

- We demonstrate how different types of space-time trajectories emerge naturally out of SPP algorithm between a given pair of origin and destination for different disturbance conditions and other problem parameters. These emerging behaviors, while being provably safe, are also intuitive and would facilitate human monitoring.

- We also show the reactivity of the control law obtained from SPP algorithm. In other words, we show that the obtained control law is able to effectively counteract the disturbances in real-time without requiring any communication with other vehicles.

The rest of the paper is organized as follows: in Section II, we formally present the SPP problem in the presence of disturbances. In Section III, we present a brief review of time-varying reachability, the basic SPP algorithm [40] in the absence of disturbances, and the Robust Trajectory Tracking (RTT) method [41] to account for disturbances. We also use this algorithm for all our simulations in this paper. Simulation results are in Sections IV and V.

II. Sequential Path Planning Problem

Consider N vehicles (also denoted as *SPP vehicles*) which participate in the SPP process $Q_i, i = 1, \ldots, N$. We assume their dynamics are given by

$$\begin{aligned} \dot{x}_i &= f_i(x_i, u_i, d_i), t \leq t_i^{\text{STA}} \\ u_i &\in \mathcal{U}_i, d_i \in \mathcal{D}_i, i = 1 \ldots, N \end{aligned} \tag{1}$$

where $x_i \in \mathbb{R}^{n_i}$, $u_i \in \mathcal{U}_i$ and $d_i \in \mathcal{D}_i$, respectively, represent the state, control and disturbance experienced by vehicle Q_i. We partition the state x_i into the position component $p_i \in \mathbb{R}^{n_p}$ and the non-position component $h_i \in \mathbb{R}^{n_i - n_p}$: $x_i = (p_i, h_i)$. We will use the sets $\mathbb{U}_i, \mathbb{D}_i$ to respectively denote the set of functions from which the control and disturbance functions $u_i(\cdot), d_i(\cdot)$ are drawn.

Each vehicle Q_i has initial state x_i^0, and aims to reach its target $\mathcal{L}_i$ by some scheduled time of arrival t_i^{STA}. The target in general represents some set of desirable states, for example the destination of Q_i. On its way to $\mathcal{L}_i$, Q_i must avoid a set of static obstacles $\mathcal{O}_i^{\text{static}} \subset \mathbb{R}^{n_i}$. The interpretation of $\mathcal{O}_i^{\text{static}}$ could be a tall building or any set of states that are forbidden for each SPP vehicle. In addition to the static obstacles, each vehicle Q_i must also avoid the danger zones with respect to every other vehicle $Q_j, j \neq i$. The danger zones in general can represent any joint configurations between Q_i and Q_j that are considered to be unsafe. We define the danger zone of Q_i with respect to Q_j to be

$$\mathcal{Z}_{ij} = \{(x_i, x_j) : \|p_i - p_j\|_2 \leq R_c\} \tag{2}$$

whose interpretation is that Q_i and Q_j are considered to be in an unsafe configuration when they are within a distance of R_c of each other. In particular, Q_i and Q_j are said to have collided, ifif $(x_i, x_j) \in \mathcal{Z}_{ij}$.

Given the set of SPP vehicles, their targets $\mathcal{L}_i$, the static obstacles $\mathcal{O}_i^{\text{static}}$, and the vehicles' danger zones with respect to each other $\mathcal{Z}_{ij}$, our goal is, for each vehicle Q_i, to synthesize a controller which guarantees that Q_i reaches its target $\mathcal{L}_i$ at or before the scheduled time of arrival t_i^{STA}, while avoiding the static obstacles $\mathcal{O}_i^{\text{static}}$ as well as the danger zones with respect to all other vehicles $\mathcal{Z}_{ij}, j \neq i$. In addition, we would like to obtain the latest departure time t_i^{LDT} such that Q_i can still arrive at $\mathcal{L}_i$ on time.

In general, the above optimal path planning problem must be solved in the joint space of all N SPP vehicles. However, due to the high joint dimensionality, a direct dynamic programming-based solution is intractable. Therefore, authors in [40] proposed to assign a priority to each vehicle, and perform SPP given the assigned priorities. Without loss of generality, let Q_j have a higher priority than Q_i if $j < i$. Under the SPP scheme, higher-priority vehicles can ignore the presence of lower-priority vehicles, and perform path planning without taking into account the lower-priority vehicles' danger zones. A lower-priority vehicle Q_i, on the other hand, must ensure that it does not enter the danger zones of the higher-priority vehicles $Q_j, j < i$; each higher-priority vehicle Q_j induces a set of time-varying obstacles $\mathcal{O}_i^j(t)$, which represents the possible states of Q_i such that a collision between Q_i and Q_j could occur.

It is straight-forward to see that if each vehicle Q_i is able to plan a trajectory that takes it to $\mathcal{L}_i$ while avoiding the static obstacles $\mathcal{O}_i^{\text{static}}$ and the danger zones of *higher-priority vehicles* $Q_j, j < i$, then the set of SPP vehicles $Q_i, i = 1, \ldots, N$ would all be able to reach their targets safely. Under the SPP scheme, path planning can be done sequentially in descending order of vehicle priority in the state space of only a single vehicle. Thus, SPP provides a solution whose complexity scales linearly with the number of vehicles, as opposed to exponentially with a direct application of dynamic programming approaches.

III. Background

In this section, we present the basic SPP algorithm [40] in which disturbances are ignored and perfect information of vehicles positions is assumed. This simplification allows us to clearly present the basic SPP algorithm. However, in presence of disturbances, it is no longer possible to commit to exact trajectories (and hence positions), since the disturbance $d_i(\cdot)$ is *a priori* unknown. Thus, disturbances and incomplete information significantly complicate the SPP scheme. We next present the robust trajectory tracking algorithm [41] that can be used to make basic SPP approach robust to disturbances as well as to an imperfect knowledge of other vehicles' positions. All of these algorithms use time-varying reachability analysis to provide goal satisfaction and safety guarantees; therefore, we start with an overview of time-varying reachability.

A. Time-Varying Reachability Background

We will be using reachability analysis to compute a backward reachable set (BRS) $\mathcal{V}$ given some target set $\mathcal{L}$, time-varying obstacle $\mathcal{G}(t)$, and the Hamiltonian function H which captures the system dynamics as well as the roles of the control and disturbance. The BRS $\mathcal{V}$ in a time interval $[t, t_f]$ will be denoted by

$$\mathcal{V}(t, t_f) \quad \text{(backward reachable set)} \tag{3}$$

Several formulations of reachability are able to account for time-varying obstacles [29, 31] (or state constraints in general). For our application in SPP, we utilize the time-varying formulation in [31], which accounts for the time-varying nature of systems without requiring augmentation of the state space with the time variable. In the formulation in [31], a BRS is computed by solving the following *final value* double-obstacle HJ VI:

$$\begin{aligned} \max \Big\{ &\min\{D_t V(t,x) + H(t,x,\nabla V(t,x)), l(x) - V(t,x)\}, \\ &- g(t,x) - V(t,x) \Big\} = 0, \quad t \le t_f \\ &V(t_f, x) = \max\{l(x), -g(t_f, x)\} \end{aligned} \tag{4}$$

In (4), the function $l(x)$ is the implicit surface function representing the target set $\mathcal{L} = \{x : l(x) \le 0\}$. Similarly, the function $g(t,x)$ is the implicit surface function representing the time-varying obstacles $\mathcal{G}(t) = \{x : g(t,x) \le 0\}$. The BRS $\mathcal{V}(t, t_f)$ is given by

$$\mathcal{V}(t, t_f) = \{x : V(t,x) \le 0\} \tag{5}$$

Some of the reachability computations will not involve an obstacle set $\mathcal{G}(t)$, in which case we can simply set $g(t,x) \equiv \infty$ which effectively means that the outside maximum is ignored in (4).

The Hamiltonian, $H(t, x, \nabla V(t,x))$, depends on the system dynamics, and the role of control and disturbance. Whenever H does not depend explicit on t, we will drop the argument. In addition, the Hamiltonian is an optimization that produces the optimal control $u^*(t,x)$ and optimal disturbance $d^*(t,x)$, once V is determined. For BRSs, whenever the existence of a control ("$\exists u$") or disturbance is sought, the optimization is a minimum over the set of controls or disturbance. Whenever a BRS characterizes the behavior of the system for all controls ("$\forall u$") or disturbances, the optimization is a maximum. We will introduce precise definitions of reachable sets, expressions for the Hamiltonian, expressions for the optimal controls as needed for the many different reachability calculations we use.

B. SPP Without Disturbances

In this section, we give an overview of the basic SPP algorithm assuming that there is no disturbance and intruder affecting the vehicles. Although in practice, such assumptions do not hold, the description of the basic SPP algorithm will introduce the notation needed for describing the subsequent, more realistic versions of SPP. The majority of the content in this section is taken from [40].

Recall that the SPP vehicles $Q_i, i = 1, \ldots, N$, are each assigned a strict priority, with Q_j having a higher priority than Q_i if $j < i$. In the absence of disturbances, we can write the dynamics of the SPP vehicles as

$$\begin{aligned} &\dot{x}_i = f_i(x_i, u_i), t \leq t_i^{\text{STA}} \\ &u_i \in \mathcal{U}_i, \qquad i = 1 \ldots, N \end{aligned} \tag{6}$$

In SPP, each vehicle Q_i plans the path to its target set $\mathcal{L}_i$ while avoiding static obstacles $\mathcal{O}_i^{\text{static}}$ and the obstacles $\mathcal{O}_i^j(t)$ induced by higher-priority vehicles $Q_j, j < i$. Path planning is done sequentially starting from the first vehicle and proceeding in descending priority, $Q_1, Q_2, \ldots, Q_N$ so that each of the path planning problems can be done in the state space of only one vehicle. During its path planning process, Q_i ignores the presence of lower-priority vehicles $Q_k, k > i$, and induces the obstacles $\mathcal{O}_k^i(t)$ for $Q_k, k > i$.

From the perspective of Q_i, each of the higher-priority vehicles $Q_j, j < i$ induces a time-varying obstacle denoted $\mathcal{O}_i^j(t)$ that Q_i needs to avoid. Therefore, each vehicle Q_i must plan its path to $\mathcal{L}_i$ while avoiding the union of all the induced obstacles as well as the static obstacles. Let $\mathcal{G}_i(t)$ be the union of all the obstacles that Q_i must avoid on its way to $\mathcal{L}_i$:

$$\mathcal{G}_i(t) = \mathcal{O}_i^{\text{static}} \cup \bigcup_{j=1}^{i-1} \mathcal{O}_i^j(t) \tag{7}$$

With full position information of higher priority vehicles, the obstacle induced for Q_i by Q_j is simply

$$\mathcal{O}_i^j(t) = \{x_i : \|p_i - p_j(t)\|_2 \leq R_c\} \tag{8}$$

Each higher priority vehicle Q_j plans its path while ignoring Q_i. Since path planning is done sequentially in descending order or priority, the vehicles $Q_j, j < i$ would have planned their paths before Q_i does. Thus, in the absence of disturbances, $p_j(t)$ is *a priori* known, and therefore $\mathcal{O}_i^j(t), j < i$ are known, deterministic moving obstacles, which means that $\mathcal{G}_i(t)$ is also known and deterministic. Therefore, the path planning problem for Q_i can be solved by first computing the

Algorithm 1: SPP algorithm in the absence of disturbances

input : Set of vehicles $Q_i, i = 1, \ldots, N$ in the descending priority order;
Vehicle dynamics (6) and initial states x_i^0;
Vehicle destinations $\mathcal{L}_i$ and static obstacles $\mathcal{O}_i^{\text{static}}$.

output: Provably safe vehicle trajectories to respective destinations for all vehicles;
Goal-satisfaction controller $u^{\text{basic}}(\cdot)$ for all vehicles.

for $i = 1 : N$ **do**
  Trajectory planning for Q_i
  compute the total obstacle set $\mathcal{G}_i(t)$ given by (18). If $i = 1$, $\mathcal{G}_i(t) = \mathcal{O}_i^{\text{static}}\ \forall t$;
  compute the BRS $\mathcal{V}_i^{\text{basic}}(t, t_i^{\text{STA}})$ defined in (9).
  Trajectory and controller of Q_i
  compute the optimal controller $u_i^{\text{basic}}(\cdot)$ given by (11);
  determine the trajectory $x_i(\cdot)$ using vehicle dynamics (6) and the control $u_i^{\text{basic}}(\cdot)$;
  output the trajectory and optimal controller for Q_i.
  Induced obstacles by Q_i
  given the trajectory $x_i(\cdot)$, compute the induced obstacles $\mathcal{O}_k^i(t)$ given by (8) for all $k > i$.

BRS $\mathcal{V}_i^{\text{basic}}(t, t_i^{\text{STA}})$, defined as follows:

$$\begin{aligned}\mathcal{V}_i^{\text{basic}}(t, t_i^{\text{STA}}) =& \{y : \exists u_i(\cdot) \in \mathbb{U}_i, x_i(\cdot) \text{ satisfies (6)},\\ & \forall s \in [t, t_i^{\text{STA}}], x_i(s) \notin \mathcal{G}_i(s),\\ & \exists s \in [t, t_i^{\text{STA}}], x_i(s) \in \mathcal{L}_i, x_i(t) = y\}\end{aligned} \tag{9}$$

The BRS $\mathcal{V}(t, t_i^{\text{STA}})$ can be obtained by solving (4) with $\mathcal{L} = \mathcal{L}_i$, $\mathcal{G}(t) = \mathcal{G}_i(t)$, and the Hamiltonian

$$H_i^{\text{basic}}(x_i, \lambda) = \min_{u_i \in \mathcal{U}_i} \lambda \cdot f_i(x_i, u_i) \tag{10}$$

The optimal control for reaching $\mathcal{L}_i$ while avoiding $\mathcal{G}_i(t)$ is then given by

$$u_i^{\text{basic}}(t, x_i) = \arg\min_{u_i \in \mathcal{U}_i} \lambda \cdot f_i(x_i, u_i) \tag{11}$$

from which the trajectory $x_i(\cdot)$ can be computed by integrating the system dynamics, which in this case are given by (6). In addition, the latest departure time t_i^{LDT} can be obtained from the BRS $\mathcal{V}(t, t_i^{\text{STA}})$ as $t_i^{\text{LDT}} = \arg\sup_t \{x_i^0 \in \mathcal{V}(t, t_i^{\text{STA}})\}$. In summary, the basic SPP algorithm is given as follows:

C. Robust Trajectory Tracking (RTT)

In the basic SPP algorithm, lower priority vehicles know the trajectories of all higher priority vehicles. The region that a lower priority vehicle needs to avoid is thus simply given by the danger zones around these trajectories; however, disturbances and incomplete information significantly

complicate the SPP scheme. Committing to exact trajectories is no longer possible, since the disturbance $d_i(\cdot)$ is *a priori* unknown. Thus, the induced obstacles $\mathcal{O}_i^j(t)$ are no longer just the danger zones centered around positions. In this section, we provide an overview of the RTT algorithm that can overcome these issues. For simplicity of explanation, we will assume that no static obstacles exist, but method can be generalized even when static obstacles do exist. The material in this section is taken partially from [41]. Note that other algorithms have been developed in [41] to account for the disturbances, we use RTT algorithm for the simulations in this paper and only present RTT algorithm here. Interested readers are referred to [41] for the other algorithms.

Even though it is impossible to commit to and track an exact trajectory in the presence of disturbances, it may still be possible to instead *robustly* track a feasible *nominal* trajectory with a bounded error at all times. If this can be done, then the tracking error bound can be used to determine the induced obstacles. Here, computation is done in two phases: the *planning phase* and the *disturbance rejection phase*.

In the planning phase, a nominal trajectory $x_{r,j}(\cdot)$ is computed that is feasible in the absence of disturbances. This planning is done for a reduced control set $\mathcal{U}^p \subset \mathcal{U}$, as some margin is needed to reject unexpected disturbances while tracking the nominal trajectory.

In the disturbance rejection phase, we compute a bound on the tracking error, independently of the nominal trajectory. To compute this error bound, we find a robust controlled-invariant set in the joint state space of the vehicle and a tracking reference that may "maneuver" arbitrarily in the presence of an unknown bounded disturbance. Taking a worst-case approach, the tracking reference can be viewed as a virtual evader vehicle that is optimally avoiding the actual vehicle to enlarge the tracking error. We therefore can model trajectory tracking as a pursuit-evasion game in which the actual vehicle is playing against the coordinated worst-case action of the virtual vehicle and the disturbance.

Let x_j and $x_{r,j}$ denote the states of the actual vehicle Q_j and the virtual evader, respectively, and define the tracking error $e_j = x_j - x_{r,j}$. When the error dynamics are independent of the absolute state as in (12) (and also (7) in [27]), we can obtain error dynamics of the form

$$\begin{aligned} \dot{e_j} &= f_{e_j}(e_j, u_j, u_{r,j}, d_j), \\ u_j &\in \mathcal{U}_j, u_{r,j} \in \mathcal{U}_j^p, d_j \in \mathcal{D}_j, \quad t \leq 0 \end{aligned} \tag{12}$$

To obtain bounds on the tracking error, we first conservatively estimate the error bound around any reference state $x_{r,j}$, denoted $\mathcal{E}_j$:

$$\mathcal{E}_j = \{e_j : \|p_{e_j}\|_2 \leq R_{\text{EB}}\}, \tag{13}$$

where p_{e_j} denotes the position coordinates of e_j and R_{EB} is a design parameter. We next solve a reachability problem with its complement $\mathcal{E}_j^c$, the set of tracking errors violating the error bound, as the target in the space of the error dynamics. From $\mathcal{E}_j^c$, we compute the following BRS:

$$\begin{aligned}\mathcal{V}_j^{\text{EB}}(t,0) =& \{y : \forall u_j(\cdot) \in \mathbb{U}_j, \exists u_{r,j}(\cdot) \in \mathbb{U}_j^p, \exists d_j(\cdot) \in \mathbb{D}_i, \\ & e_j(\cdot) \text{ satisfies (12)}, e_j(t) = y, \\ & \exists s \in [t,0], e_j(s) \in \mathcal{E}_j^c\},\end{aligned} \tag{14}$$

where the Hamiltonian to compute the BRS is given by:

$$H_j^{\text{EB}}(e_j, \lambda) = \max_{u_j \in \mathcal{U}_j} \min_{u_r \in \mathcal{U}_j^p, d_j \in \mathcal{D}_j} \lambda \cdot f_{e_j}(e_j, u_j, u_{r,j}, d_j). \tag{15}$$

Letting $t \to -\infty$, we obtain the infinite-horizon control-invariant set $\Omega_j := \lim_{t \to -\infty} \left(\mathcal{V}_j^{\text{EB}}(t,0)\right)^c$. If Ω_j is nonempty, then the tracking error e_j at flight time is guaranteed to remain within $\Omega_j \subseteq \mathcal{E}_j$ provided that the vehicle starts inside Ω_j and subsequently applies the feedback control law

$$\kappa_j(e_j) = \arg\max_{u_j \in \mathcal{U}_j} \min_{u_r \in \mathcal{U}_j^p, d_j \in \mathcal{D}_j} \lambda \cdot f_{e_j}(e_j, u_j, u_{r,j}, d_j). \tag{16}$$

The induced obstacles by each higher-priority vehicle Q_j can thus be obtained by:

$$\begin{aligned}\mathcal{O}_i^j(t) &= \{x_i : \exists y \in \mathcal{P}_j(t), \|p_i - y\|_2 \le R_c\} \\ \mathcal{P}_j(t) &= \{p_j : \exists h_j, (p_j, h_j) \in \mathcal{M}_j(t)\} \\ \mathcal{M}_j(t) &= \Omega_j + x_{r,j}(t),\end{aligned} \tag{17}$$

where the "+" in (17) denotes the Minkowski sum[a]. Intuitively, if Q_j is tracking $x_{r,j}(t)$, then it will remain within the error bound Ω_j around $x_{r,j}(t)$ $\forall t$. This is precisely the set $\mathcal{P}_j(t)$. The induced obstacles can then be obtained by augmenting a danger zone around this set. Finally, we can obtain the total obstacle set $\mathcal{G}_i(t)$ using:

$$\mathcal{G}_i(t) = \mathcal{O}_i^{\text{static}} \cup \bigcup_{j=1}^{i-1} \mathcal{O}_i^j(t) \tag{18}$$

Since each vehicle Q_j, $j < i$, can only be guaranteed to stay within Ω_j, we must make sure during the path planning of Q_i that at any given time, the error bounds of Q_i and Q_j, Ω_i and Ω_j, do not intersect. This can be done by augmenting the total obstacle set by Ω_i:

$$\tilde{\mathcal{G}}_i(t) = \mathcal{G}_i(t) + \Omega_i. \tag{19}$$

Finally, given Ω_i, we can guarantee that Q_i will reach its target $\mathcal{L}_i$ if $\Omega_i \subseteq \mathcal{L}_i$; thus, in the path planning phase, we modify $\mathcal{L}_i$ to be $\tilde{\mathcal{L}}_i := \{x_i : \Omega_i + x_i \subseteq \mathcal{L}_i\}$, and compute a BRS, with the control authority $\mathcal{U}_i^p$, that contains the initial state of the vehicle. Mathematically,

[a]The Minkowski sum of sets A and B is the set of all points that are the sum of any point in A and B.

$$\begin{aligned}\mathcal{V}_i^{\text{rtt}}(t, t_i^{\text{STA}}) =& \{y : \exists u_i(\cdot) \in \mathbb{U}_i^p, x_i(\cdot) \text{ satisfies (6)}, \\ & \forall s \in [t, t_i^{\text{STA}}], x_i(s) \notin \tilde{\mathcal{G}}_i(t), \\ & \exists s \in [t, t_i^{\text{STA}}], x_i(s) \in \tilde{\mathcal{L}}_i, x_i(t) = y\}\end{aligned} \tag{20}$$

The BRS $\mathcal{V}_i^{\text{rtt}}(t, t_i^{\text{STA}})$ can be obtained by solving (4) using the Hamiltonian:

$$H_i^{\text{rtt}}(x_i, \lambda) = \min_{u_i \in \mathcal{U}_i^p} \lambda \cdot f_i(x_i, u_i) \tag{21}$$

The corresponding optimal control for reaching $\tilde{\mathcal{L}}_i$ is given by:

$$u_i^{\text{rtt}}(t) = \arg\min_{u_i \in \mathcal{U}_i^p} \lambda \cdot f_i(x_i, u_i). \tag{22}$$

The nominal trajectory $x_{r,i}(\cdot)$ can thus be obtained by using vehicle dynamics in the absence of disturbances (given by (6)) with the optimal control $u_i^{\text{rtt}}(\cdot)$ given by (22). From the resulting nominal trajectory $x_{r,i}(\cdot)$, the overall control policy to reach $\mathcal{L}_i$ can be obtained via (16). The robust trajectory tracking method can be summarized as follows:

IV. City Environment Simulation

We now illustrate SPP algorithm using a 50-vehicle UAV system where UAVs are flying through a city environment. This setup can be representative of many UAV applications, such as package delivery, aerial surveillance, etc.

A. Setup

We grid the City of San Francisco (SF) in California, USA, and use it as our position space, as shown in Figure 1.

Algorithm 2: Robust trajectory tracking algorithm (SPP algorithm in the presence of disturbances)

input : Set of vehicles $Q_i, i = 1, \ldots, N$ in the descending priority order;
Vehicle dynamics (1) and initial states x_i^0;
Vehicle destinations $\mathcal{L}_i$ and static obstacles $\mathcal{O}_i^{\text{static}}$.

output: The nominal trajectories to respective destinations for all vehicles;
Goal-satisfaction controller for all vehicles.

for $i = 1 : N$ **do**
- **Computation of tracking error bound for** Q_i
- obtain the error dynamics given by (12);
- decide on a reduced control authority $\mathcal{U}_i^p$ for the planning phase, and choose a parameter R_{EB} to conservatively bound the tracking error;
- compute the BRS $\mathcal{V}_i^{\text{EB}}(t, 0)$ using (14);
- compute the Ω_i using the converged BRS $\mathcal{V}_i^{\text{EB}}$;
- **if** $\Omega_i \neq \emptyset$ **then**
  - the error bound on the tracking error is given by Ω_i;
- **else**
  - recompute the tracking error bound using a larger R_{EB};
- **Computation of obstacles for** Q_i
- determine the total obstacle set $\mathcal{G}_i(t)$, given in (18). In the case $i = 1$, $\mathcal{G}_i(t) = \mathcal{O}_i^{\text{static}}\ \forall t$;
- using Ω_i, determine the augmented obstacle set $\tilde{\mathcal{G}}_i(t)$, given in (19).
- **Trajectory planning for** Q_i
- compute the reduced target set $\tilde{\mathcal{L}}_i$;
- compute the BRS $\mathcal{V}_i^{\text{rtt}}(t, t_i^{\text{STA}})$ defined in (20).
- **Trajectory and controller of** Q_i
- compute the nominal controller $u_i^{\text{rtt}}(\cdot)$ given by (22);
- determine the nominal trajectory $x_{r,i}(\cdot)$ using vehicle dynamics (6) and the control $u_i^{\text{rtt}}(\cdot)$;
- the overall goal satisfaction controller for Q_i is given by (16);
- output the nominal trajectory and the optimal tracking controller for Q_i.
- **Induced obstacles by** Q_i
- given the trajectory $x_{r,i}(\cdot)$ and the tracking error bound Ω_i, compute the induced obstacles $\mathcal{O}_k^i(t)$ given by (17) for all $k > i$.

Each box in Figure 1 represents a 1 km^2 area of SF. The origin point for the vehicles is denoted by the blue star. Four different areas in the city are chosen as the destinations for the vehicles. Mathematically, the target sets $\mathcal{L}_i$ of the vehicles are circles of radius r in the position space, i.e. each vehicle is trying to reach some desired set of positions. In terms of the state space x_i, the target sets are defined as

$$\mathcal{L}_i = \{x_i : \|p_i - c_i\|_2 \leq r\} \tag{23}$$

where c_i are centers of the target circles. In this simulation, we use $r = 100$ m. The four targets are represented by four circles in Figure 1. The destination of each vehicle is chosen randomly from these four destinations. Finally, some areas in downtown SF and the city hall are used as representative static obstacles for the SPP vehicles, denoted by black contours in Figure 1.

For this simulation, we use the following dynamics for each vehicle:

$$\begin{aligned} \dot{p}_{x,i} &= v_i \cos\theta_i + d_{x,i} \\ \dot{p}_{y,i} &= v_i \sin\theta_i + d_{y,i} \\ \dot{\theta}_i &= \omega_i, \\ \underline{v} \leq v_i \leq \bar{v},\ &|\omega_i| \leq \bar{\omega}, \\ \|(d_{x,i}, d_{y,i})\|_2 &\leq d_r \end{aligned} \tag{24}$$

where $x_i = (p_{x,i}, p_{y,i}, \theta_i)$ is the state of vehicle Q_i, $p_i = (p_{x,i}, p_{y,i})$ is the position, θ_i is the heading, and $d = (d_{x,i}, d_{y,i})$ represents Q_i's disturbances, for example wind, that affect its position evolution. The control of Q_i is $u_i = (v_i, \omega_i)$, where v_i is the speed of Q_i and ω_i is the turn rate; both controls have a lower and upper bound. To make our simulations as close as possible to real scenarios, we choose velocity and turn-rate bounds as $\underline{v} = 0$ m/s, $\bar{v} = 25$ m/s, $\bar{\omega} = 2$ rad/s, aligned with the modern UAV specifications [43, 44]. The disturbance bounds are chosen to be either $d_r = 6$ m/s or $d_r = 11$ m/s. These conditions correspond to *moderate breeze* and *strong breeze* respectively on the Beaufort scale [45]. The scheduled times of arrival t_i^{STA} for all vehicles are chosen to be all 0 for a high UAV density condition. For medium and low density conditions, we separated the t_i^{STA} for the vehicles by 5 s and 10 s respectively, with the latest t_i^{STA} being 0. Note that we have used same dynamics and input bounds across all vehicles for clarity of illustration; however, SPP can easily handle more general systems of the form in which the vehicles have different control bounds, t_i^{STA} and dynamics.

The goal of the vehicles is to reach their destinations while avoiding a collision with the other vehicles or the static obstacles. The joint state space of this 50-vehicle system is 150-dimensional (150D), making the joint path planning and collision avoidance problem intractable for direct analysis. Therefore, we assign a priority ordering to vehicles and solve the path planning problem sequentially. For this simulation, we assign a random priority order to fifty vehicles and use RTT algorithm to compute the trajectories of the vehicles.

B. Results for 6 m/s wind and no arrival time separation

As per Algorithm 2, we start with Q_1 and sequentially compute the optimal control policy κ_j and the latest departure time t_j^{LDT} for each vehicle. To compute the error bound $\mathcal{E}_j$ on the tracking error of vehicle j, we choose $R_{\text{EB}} = 5$ m and use the reduced control authority $\mathcal{U}_j^p = \{(v_{r,j}, \omega_{r,j}) : 11 \text{ m/s} \leq v_{r,j} \leq 13 \text{ m/s}, |\omega_{r,j}| \leq 1.2 \text{ rad/s}\}$. Given dynamics in (24), the error dynamics between Q_j and the virtual evader are given by [27]:

$$\begin{aligned} \dot{e}_{x,j} &= v_j \cos(e_{\theta,j}) - v_{r,j} + \omega_{r,j} e_{y,j} + d_{x,j} \\ \dot{e}_{y,j} &= v_{r,j} \sin(e_{\theta,j}) - \omega_{r,j} e_{x,j} + d_{y,j} \\ \dot{e}_{\theta,j} &= \omega_j - \omega_{r,j}, \end{aligned} \tag{25}$$

where $e_j = (e_{x,j}, e_{y,j}, e_{\theta,j})$ is the tracking error in the three states of Q_j. Given relative dynamics, we compute the BRS $\mathcal{V}^{\text{EB}}$ using (14) and evaluate the infinite-horizon control invariant set Ω_j. For all the BRS computations in this simulation, we use Level Set Toolbox [35]. In presence of moderate winds, the obtained tracking error bound is 5 m. This means that given any trajectory (which is a sequence of states over time) of vehicle, winds can at most cause a deviation of 5 m from this trajectory at all times. Consequently, the vehicle will be within a distance of 5 m from the trajectory. Note that since all SPP vehicles have same dynamics, the error bound is also same for all vehicles.

The optimal control policy for Q_j is thus given by $\kappa_j(e_j)$ in (16). However, to compute the relative state e_j, the nominal trajectory $x_{r,j}$ for Q_j is required. Using Ω_j, we compute the obstacles induced by the higher-priority vehicles for Q_j and the obstacle induced by Q_j for the lower-priority vehicles. These obstacles are given by (19) and (17) respectively. Note that since disturbance directly impacts the computation of tracking error bound, these obstacles also grow as disturbance magnitude increases. We will illustrate the effect of disturbance magnitude on the trajectories of vehicles in IV-C.

The nominal trajectory can now be obtained by computing $\mathcal{V}_j^{\text{rtt}}$ in (20) and executing the corresponding control policy u_j^{rtt} in (22), starting from the initial state x_j^0. Finally, the latest departure time t_j^{LDT} is given by $\arg\sup_t x_j^0 \in \mathcal{V}_j^{\text{rtt}}(t, t_j^{\text{STA}})$. It is important to note that since the BRS $\mathcal{V}_j^{\text{rtt}}$ is computed backwards starting from the scheduled time of arrival t_j^{STA}, (a) the latest departure time t_j^{LDT} directly depends on and varies with t_j^{STA} and (b) it directly impacts the obstacles that Q_j needs to avoid in its path towards its destination. We will illustrate the effect of the scheduled time of arrival on the trajectories of vehicles also in IV-C.

The resulting trajectories of vehicles for $d_r = 6$ m/s and $t_j^{\text{STA}} = 0 \ \forall j$ at different times are shown in Figure 2. As evident from the figures, the vehicles remain clear of all the static obstacles (the black contours) and make steady progress towards reaching their destinations. The vehicles whose destinations are relatively closer need less time to travel to their destinations and depart

later. Note that the shown trajectories are simulated under uniformly random disturbance (i.e., for every vehicle, the disturbance is uniformly sampled from a circle of radius $d_r = 6$ m/s at each time step), but the SPP algorithm guarantees safety and reactivity despite the worst case disturbance, as we show later in this section.

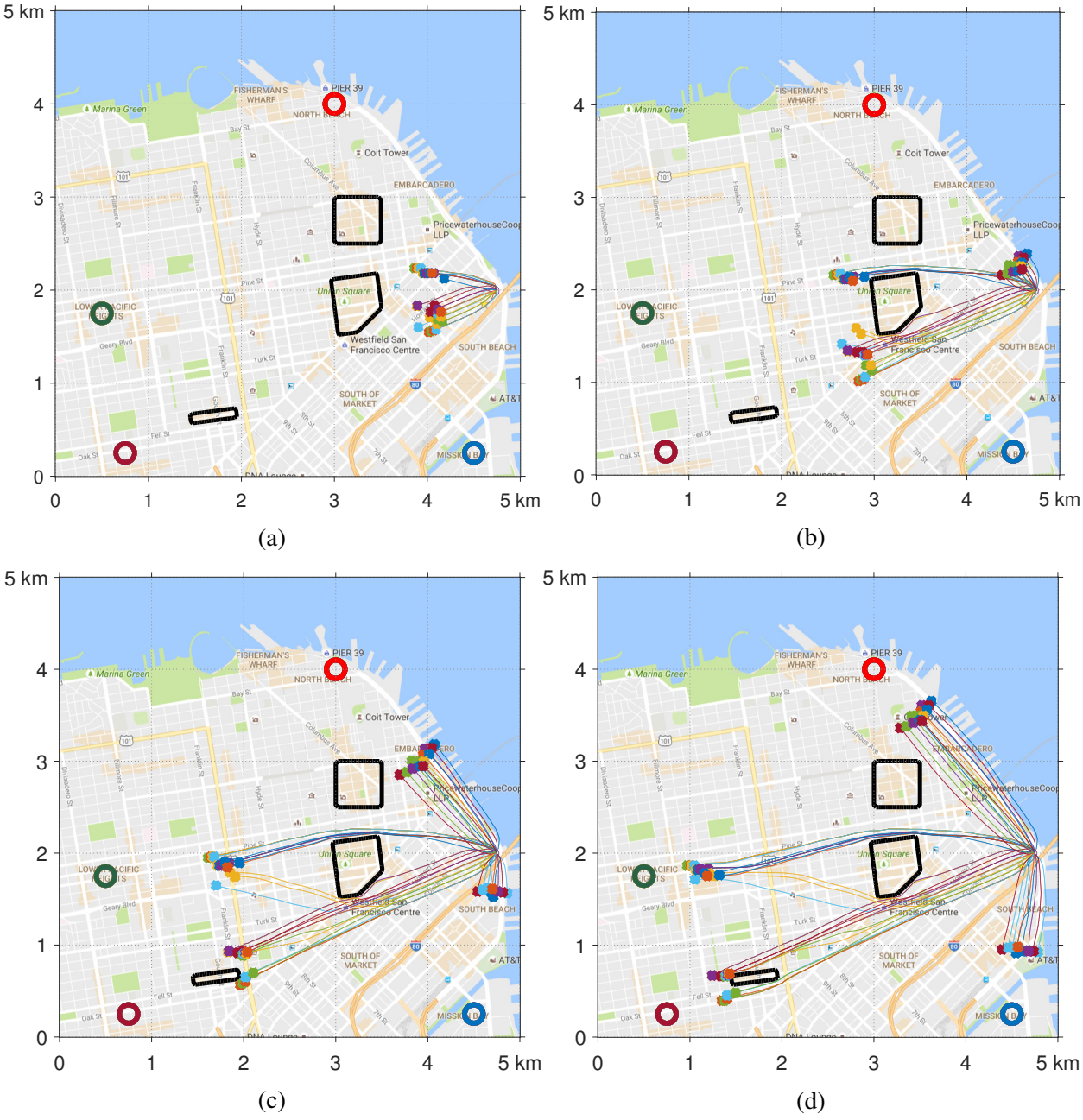


Figure 2. Snapshots of vehicle trajectories at different times for uniform disturbance with $d_r = 6$ m/s. The vehicles remain clear of all static obstacles despite the disturbance in the dynamics.

The full trajectories of vehicles are shown in Figure 7a. All vehicles reach their respective destinations. A zoomed-in version of Figure 7a near the red-target (Figure 3) illustrates that vehicles are also outside each other's danger zones (circles around the vehicles) as required. It is interesting

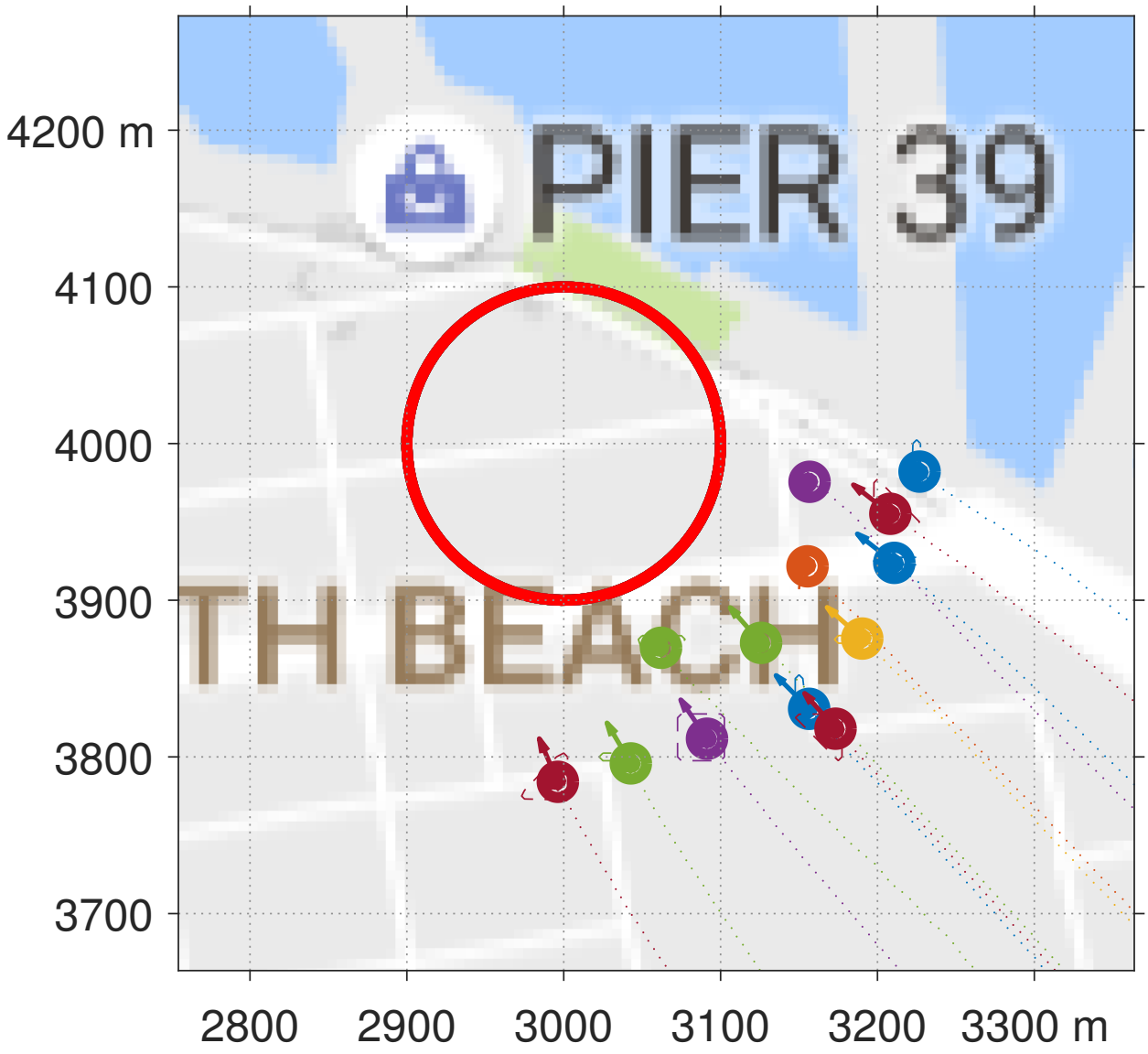
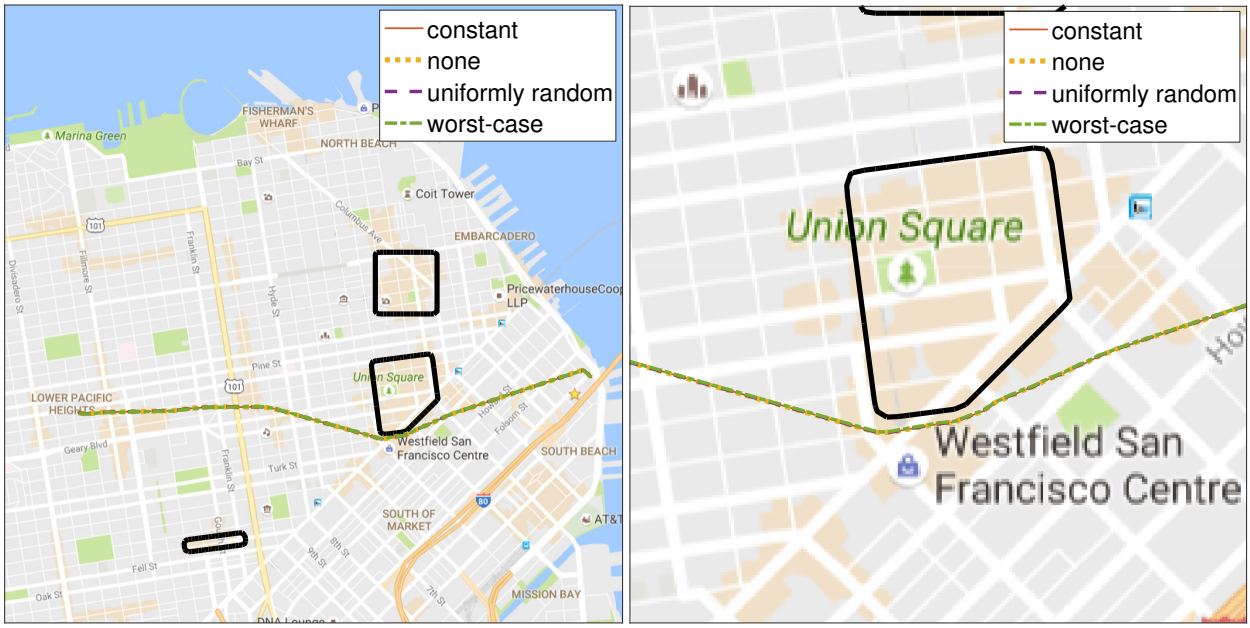


Figure 3. Zoomed-in version of vehicle trajectories near the red target in Figure 7a. The SPP algorithm ensures that the vehicles are outside each other's danger zones (circles around the vehicles).

to note that the vehicles going to the same destination take different trajectories. This is because all vehicles have same scheduled time of arrival, and hence the lower-priority vehicles do not have the flexibility to wait for the higher-priority vehicles. In order to ensure that they reach their destinations on time, they take alternative trajectories to their destinations, forming different "traffic lanes". Thus, the vehicles' trajectories obtained in this case are predominately *state-separated* trajectories, i.e., they follow different state trajectories but at the same time.

Although the plotted trajectories in Figure 7a are for a particular realization of the (uniformly random) disturbances, the SPP algorithm guarantees obstacle avoidance regardless of the realized disturbance. To illustrate this, we plot the trajectories of a particular vehicle ($Q_1 7$) for three different disturbance realizations – uniform, worst-case, and constant wind from the west direction – in

Figure 4. One can see that the trajectories are nearly indistinguishable even in the zoomed-in plot on the right.



(a) (b)

Figure 4. Trajectories of Q_{17} under different types of bounded disturbances – constant wind from the west, no wind, uniformly random wind, and worse-case wind. The right plot zooms in on the beginning of the left plot.

Figure 5 shows the minimum distance between Q_{17} and other vehicles. One can see that this distance is never below 10 m, which is minimum required separation between the vehicles. The separation distance is small in the beginning because the vehicles need to depart as soon as possible while maintaining this minimum required separation.

Figure 6 shows the optimal control action in terms of the turn rate and the linear speed of each vehicle, demonstrating that the tracking control law reacts to tracking error due to disturbances in an intuitive way. Both subplots show the control law when the vehicle is facing east. Figure 6a indicates, for example, that generally when the tracking error is such that the vehicle is too far to the right compared to the nominal position, the optimal tracking control is to turn left. Figure 6b indicates, for example, that when the vehicle is too far ahead of the nominal position, it should slow down. The optimal turn rate and linear speed controls ensure that the vehicle never deviates from the nominal trajectory for more than 5 m.

The average trajectory computation time per vehicle is 2 s using a CUDA implementation of the level set toolbox on a desktop computer with a Core i7 5820K processor and two GeForce GTX Titan X graphics processing units. Note that all this computation is done offline and the resulting optimal policy $\kappa_j(e_j)$ is obtained as a lookup table. In real time, neither any computation nor any communication between vehicles is required. Only a lookup table query is required, and this can be performed very quickly in real time. This illustrates the capability of SPP as a provably safe

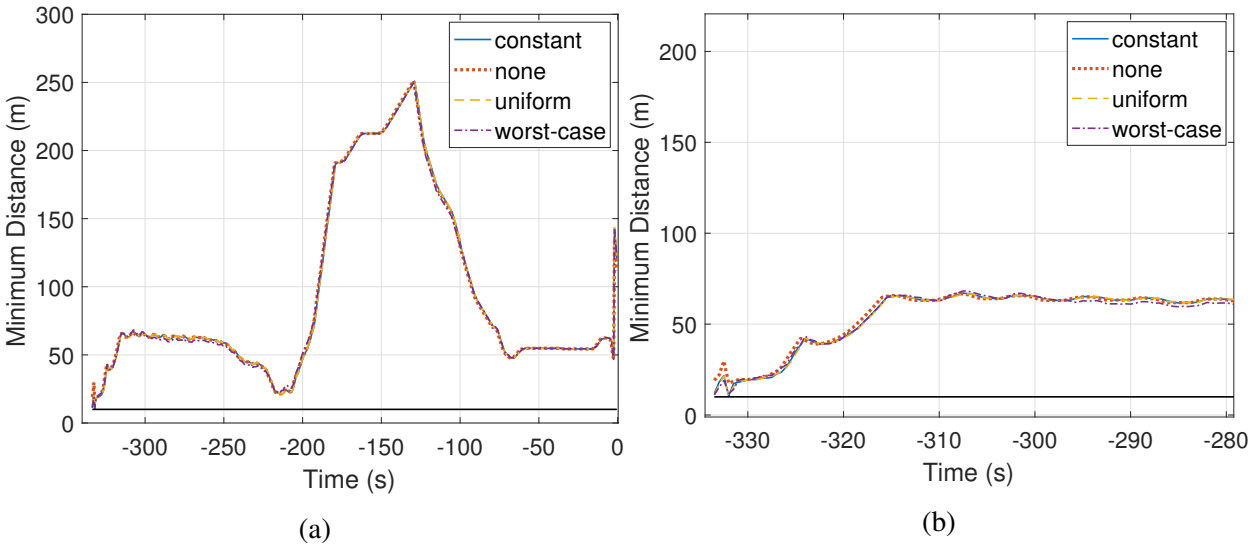
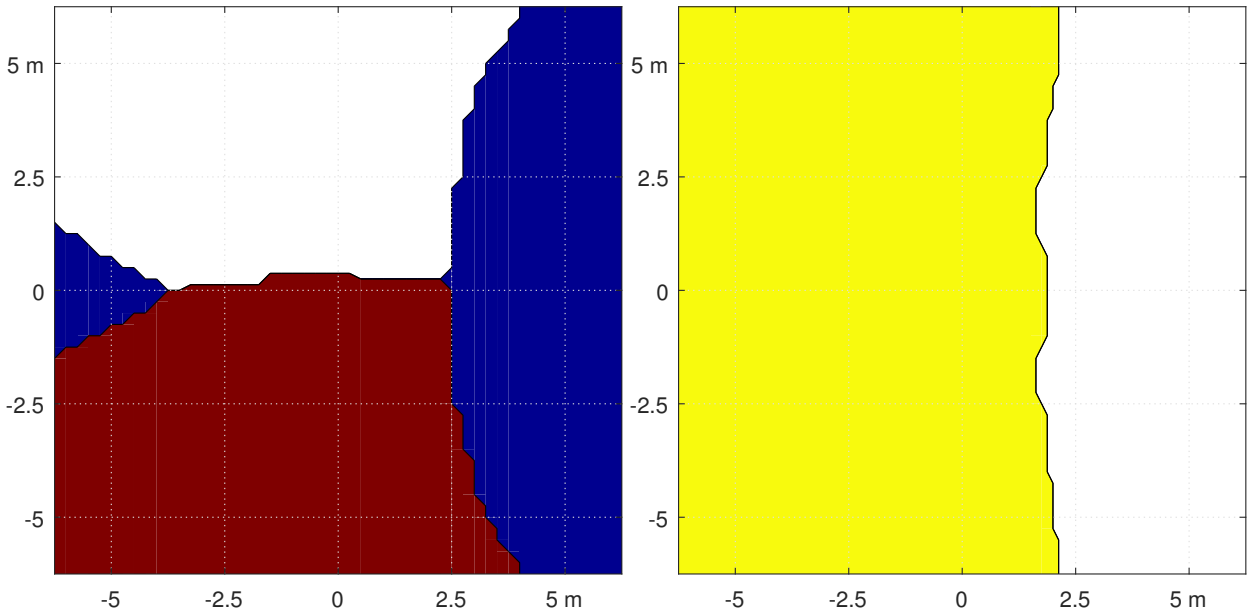


Figure 5. Minimum distance to other vehicles for Q_{17} under different types of bounded disturbances – constant wind from the west, no wind, uniformly random wind, and worse-case wind. The right plot zooms in on the beginning of the left plot.



(a) Turn rate control as a function of tracking error. White: turn right; red: turn left; blue: go straight.

(b) Linear speed control as a function of tracking error. Yellow: high speed; white: low speed.

Figure 6. Optimal tracking control law for each vehicle for a heading of zero degrees (facing towards the right).

path planning algorithm for large multi-vehicle systems. Without CUDA, the computation time per vehicle is 33 s using a C++ implementation of the level set toolbox, and 200 s using a Matlab implementation.

C. Effect of Disturbance and Scheduled Time of Arrival

In this section, we illustrate how the disturbance bound d_r in (24) and the relative t^{STA}s of vehicles affect the vehicle trajectories. For this purpose, we simulate the SPP algorithm for four additional scenarios:

- Case-0: $d_r = 6$ m/s, $t_i^{\text{STA}} = 0 \; \forall i$
- Case-1: $d_r = 11$ m/s, $t_i^{\text{STA}} = 0 \; \forall i$
- Case-2: $d_r = 6$ m/s, $t_i^{\text{STA}} = 5(i-1) \; \forall i$
- Case-3: $d_r = 11$ m/s, $t_i^{\text{STA}} = 5(i-1) \; \forall i$
- Case-4: $d_r = 11$ m/s, $t_i^{\text{STA}} = 10(i-1) \; \forall i$

The interpretation $t_i^{\text{STA}} = 5(i-1)$ is that the scheduled time of arrival of any two consecutive vehicles is separated by 5 s, which represents a medium vehicle density scenario; a separation of 10 s represents a low vehicle density scenario. $d_r = 6$ m/s and $d_r = 11$ m/s correspond to the moderate breeze and strong breeze respectively on Beaufort wind force scale [45].

Intuitively, as d_r increases, it is harder for a vehicle to closely track a particular nominal trajectory, which results in a higher tracking error bound. As mentioned previously, with a 6 m/s wind speed, the tracking error bound is 5 m; however, with an 11 m/s wind speed, the tracking error bound becomes 35 m. Thus, the vehicles need to be separated more from each other in space to ensure that they do not enter each other's danger zones. This is also evident from comparing the results corresponding to Case-0 (Fig. 7a) and Case-1 (Fig. 7b). As the disturbance magnitude increases from $d_r = 6$ m/s (moderate breeze) to $d_r = 11$ m/s (strong breeze), the vehicles' trajectories get farther apart from each other. Since t^{STA} is same for all vehicles, the vehicles trajectories are still predominately *state-separated* trajectories.

We next compare Case-0 and Case-2. The difference between these two cases is that vehicles have a 5 second separation in their schedule times of arrival in Case-2. When vehicles Q_i and Q_j ($j > i$) have same scheduled time of arrival and are going to the same destination, they are constrained to travel at the same time to make sure they reach the destination by the designated t^{STA}. However, since Q_i is high-priority, it gets access to the optimal trajectory (in terms of the total time of travel to destination) and Q_j has to settle for a relatively sub-optimal trajectory. Thus, all vehicles going to a particular destination take different trajectories creating a "band" of trajectories between the origin and the destination, as shown in Figure 7a; the high-priority vehicles take a relatively straight path between the origin and the destination whereas the low-priority vehicles take a (relatively sub-optimal) curved path. If we think of an air highway between the origin and the destination, then vehicles take different lanes of that highway to reach the destination in Case-0. Thus, the trajectories of vehicles in this case are *state-separated*. However, when $t_j^{\text{STA}} > t_i^{\text{STA}}$,

then Q_j is not bound to travel at the same time as Q_i; it can wait for Q_i to depart and take a shorter path later on. Thus, vehicles travel in a single (optimal) lane in this case, as shown in Figure 7c. In other words, they take the same trajectory to the destination, but at different times. Thus, the trajectories of vehicles in this case are *time-separated*.

Note that the exact number of lanes depends on *both* the disturbance and separation of scheduled times of arrival. As the disturbance increases, vehicles need to be separated more from each other to ensure safety. A larger arrival time difference between vehicles is also able to ensure this separation even if the vehicles were to take the same lane. As shown in Figure 7d, a difference of 5 s in the t^{STA}'s is not sufficient to achieve a single lane behavior for stronger 11 m/s wind conditions. However, the number of lanes is significantly less than that in Case-1 (Fig. 7b). Finally, a separation of 10 s in t^{STA}'s ensure that we get the single lane behavior even in the presence of 11 m/s winds, leading to *time-separated* trajectories. Videos of the simulations can be found at https://youtu.be/1ocaBGZqSAE.

Overall, the relative magnitude of disturbance and scheduled times of arrival separation determines the number of lanes and type of trajectories that emerge out of the SPP algorithm. For a fixed disturbance magnitude, as the separation in the scheduled times of arrival of vehicles increases, the number of lanes between a pair of origin and destination decreases, and more and more trajectories become time-separated. On the other hand, for a fixed separation in the scheduled times of arrival of vehicles, as the disturbance magnitude increases, the number of lanes between a pair of origin and destination increases, and more and more trajectories become state-separated.

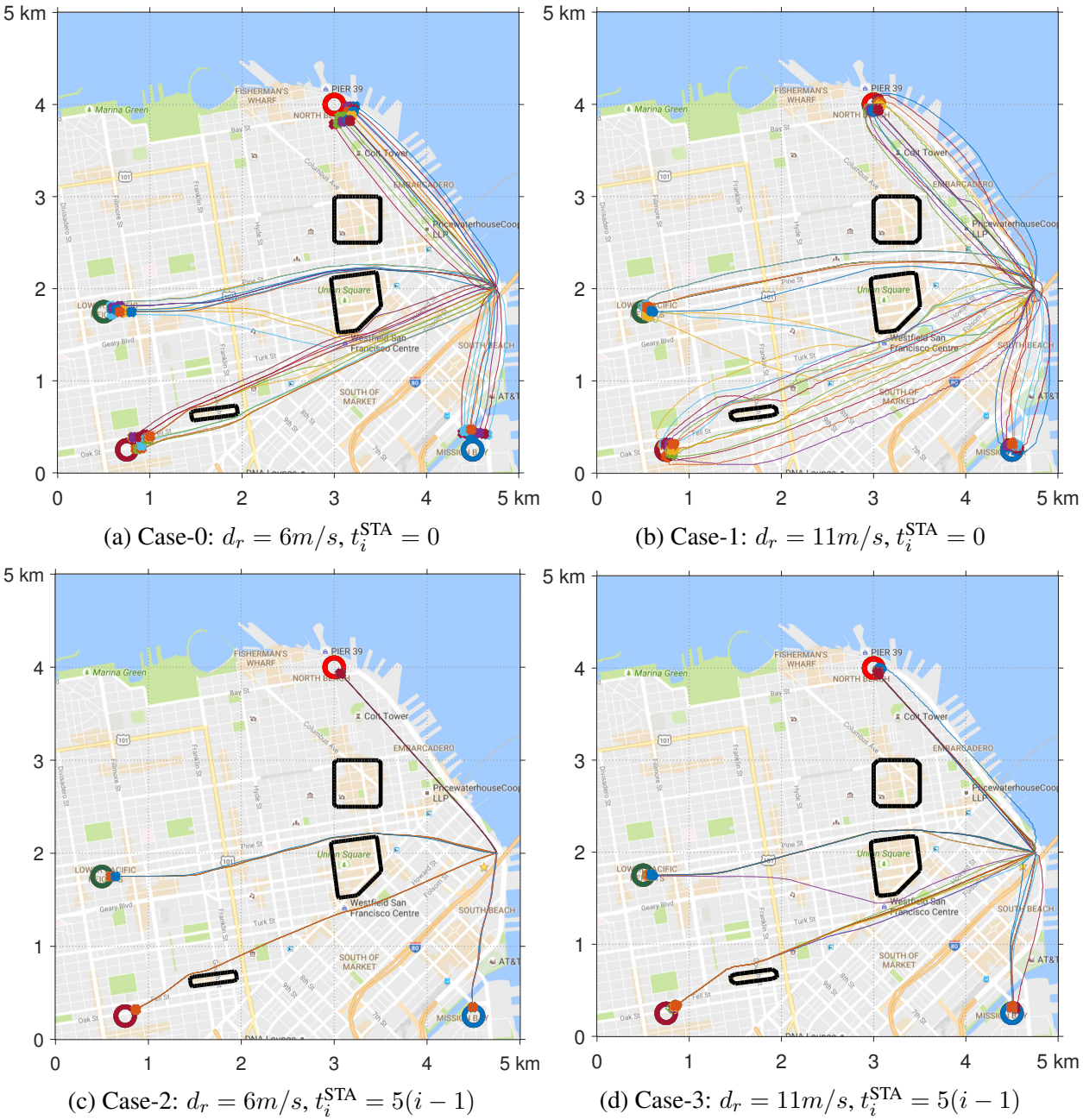
V. Multi-city Environment Simulation

We next use SPP algorithm to design trajectories for a 200-vehicle UAV system where UAVs are flying through a multi-city region.

A. Setup

We grid the San Francisco Bay Area in California, US and use it as our state space, as shown in Figure 9. We consider the UAVs flying to and from four cities: Richmond, Berkeley, Oakland, and San Francisco. The blue region in Fig. 9 represents bay. This environment is different from the city environment in Section IV in that now the UAVs need to fly for longer distances and through a high-density vehicle environment with strong winds, but have very few static obstacles like tall buildings.

Each box in Figure 9 represents a 25 km^2 area. The vehicles are flying to and from the four cities indicated by the four circles. The origin and the destination of each vehicle is chosen randomly from these four cities. The vehicle dynamics are given by (24). We choose velocity and turn-rate bounds as $\underline{v} = 0$ m/s, $\bar{v} = 25$ m/s, $\bar{\omega} = 2$ rad/s. The disturbance bound is chosen as



(a) Case-0: $d_r = 6m/s$, $t_i^{\text{STA}} = 0$

(b) Case-1: $d_r = 11m/s$, $t_i^{\text{STA}} = 0$

(c) Case-2: $d_r = 6m/s$, $t_i^{\text{STA}} = 5(i-1)$

(d) Case-3: $d_r = 11m/s$, $t_i^{\text{STA}} = 5(i-1)$

Figure 7. Effect of the disturbance magnitude and the scheduled times of arrival on vehicle trajectories. All trajectories are simulated under uniformly random disturbance. The relative separation in the scheduled times of arrival of vehicles determines the number of lanes between a pair of origin and destination, and more and more tarjectories become time-separated as this relative separation increases. The disturbance magnitude determines the relative separation between different lanes, and more and more tarjectories become state-separated as the disturbance increases.

$d_r = 11$ m/s, which corresponds to *strong breeze* on Beaufort wind force scale [45]. The scheduled time of arrival t^{STA} for vehicles are chosen as $5(i-1)$ s.

The goal of the vehicles is to reach their destinations while avoiding a collision with the other vehicles. The joint state space of this 200-vehicle system is 600-dimensional, making the joint path

planning and collision avoidance problem intractable for direct analysis. Therefore, we assign a priority order to vehicles and solve the path planning problem sequentially.

B. Results

The trajectory planning for the vehicles is done using RTT algorithm, similar to that in Section IV-B. The resulting trajectories of vehicles are shown in Figure 10a. Once again, the vehicles remain clear of all other vehicles and reach their respective destinations. Given the separation between the scheduled times of arrival, the trajectories are predominately *time-separated*, with roughly two lanes for each pair of cities (one for going from city A to city B and another for from city B to city A). A high-density of vehicles is achieved in the center since the 4 paths are intersecting in the center (Richmond-Oakland, Oakland-Richmond, Berkeley-San Francisco, San Francisco-Berkeley), but the SPP algorithm ensures safety despite this high-density, as shown in the zoomed-in version of center at an intermediate time when a large number of vehicles are passing through the central region (Figure 10b).

Finally, we simulate the system for the case where $t_i^{\text{STA}} = 0 \; \forall i$. As evident from Figure 11, we get multiple lanes between each pair of cities in this case and trajectories become predominately state-separated, as we expect based on the discussion in Section IV-C.

The average computation time per vehicle is 4 minutes using a CUDA implementation of the Level Set Toolbox on a desktop computer with a Core i7 5820K processor and two GeForce GTX Titan X graphics processing units. The computation time is much longer than in the previous simulation in SF because of the larger space over which planning is done. Once again all the computation is done offline and only a lookup table query is required in real-time, which can be performed very efficiently. This simulation illustrates the scalability and the potential of deploying the SPP algorithm for provably safe path planning for large multi-vehicle systems.

VI. Conclusion

Provably safe multi-vehicle path planning in an important problem that needs to be addressed to ensure that vehicles can fly in close proximity of each other. Recently, the SPP algorithm was proposed for multi-vehicle path planning problem that scales linearly with the number of vehicles. We illustrate the full potential of the algorithm by using it for large-scale multi-vehicle path planning problems under different flying conditions. We demonstrate how different types of space-time trajectories emerge naturally out of the algorithm for different disturbance conditions and other problem parameters. The reactivity of the obtained controller is also demonstrated under different wind conditions.

Acknowledgements

This research is supported by ONR under the Embedded Humans MURI (N00014-16-1-2206).

References

[1] Tice, B. P., "Unmanned Aerial Vehicles – The Force Multiplier of the 1990s," *Airpower Journal*, 1991.

[2] DeBusk, W., "Unmanned Aerial Vehicle Systems for Disaster Relief: Tornado Alley," *AIAA Infotech@Aerospace 2010*, American Institute of Aeronautics and Astronautics, Apr. 2010, pp. 2010–3506.

[3] Amazon.com, Inc., "Amazon Prime Air," 2016.

[4] AUVSI News, "UAS Aid in South Carolina Tornado Investigation," 2016.

[5] BBC Technology, "Google plans drone delivery service for 2017," 2016.

[6] Joint Planning and Development Office, "Unmanned Aircraft Systems (UAS) Comprehensive Plan," Tech. rep., Federal Aviation Administration, 2014.

[7] Prevot, T., Rios, J., Kopardekar, P., Robinson III, J. E., Johnson, M., and Jung, J., "UAS Traffic Management (UTM) Concept of Operations to Safely Enable Low Altitude Flight Operations," *16th AIAA Aviation Technology, Integration, and Operations Conference*, American Institute of Aeronautics and Astronautics, Jun. 2016, pp. 1–16.

[8] Fiorini, P. and Shiller, Z., "Motion Planning in Dynamic Environments Using Velocity Obstacles," *The International Journal of Robotics Research*, Vol. 17, No. 7, Jul. 1998, pp. 760–772.

[9] Chasparis, G. and Shamma, J., "Linear-programming-based multi-vehicle path planning with adversaries," *American Control Conference*, Jun. 2005, pp. 1072–1077.

[10] van den Berg, J., Ming Lin, and Manocha, D., "Reciprocal Velocity Obstacles for real-time multi-agent navigation," *International Conference on Robotics and Automation*, May 2008, pp. 1928–1935.

[11] Wu, A. and How, J. P., "Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles," *Autonomous Robots*, Vol. 32, No. 3, Apr. 2012, pp. 227–242.

[12] Olfati-Saber, R. and Murray, R. M., "DISTRIBUTED COOPERATIVE CONTROL OF MULTIPLE VEHICLE FORMATIONS USING STRUCTURAL POTENTIAL FUNCTIONS," *IFAC Proceedings Volumes*, Vol. 35, No. 1, 2002, pp. 495–500.

[13] Chuang, Y. L., Huang, Y. R., D'Orsogna, M. R., and Bertozzi, A. L., "Multi-vehicle flocking: Scalability of cooperative control algorithms using pairwise potentials," *International Conference on Robotics and Automation*, Apr. 2007, pp. 2292–2299.

[14] Feng-Li Lian and Murray, R., "Real-time trajectory generation for the cooperative path planning of multi-vehicle systems," *Conference on Decision and Control*, Vol. 4, Dec. 2002, pp. 3766–3769.

[15] Ahmadzadeh, A., Motee, N., Jadbabaie, A., and Pappas, G., "Multi-vehicle path planning in dynamically changing environments," *International Conference on Robotics and Automation*, May 2009, pp. 2449–2454.

[16] Bellingham, J., Tillerson, M., Alighanbari, M., and How, J., "Cooperative path planning for multiple UAVs in dynamic and uncertain environments," *Conference on Decision and Control*, Vol. 3, Dec. 2002, pp. 2816–2822.

[17] Beard, R. and McLain, T., "Multiple UAV cooperative search under collision avoidance and limited range communication constraints," *Conference on Decision and Control*, Vol. 1, 2003, pp. 25–30.

[18] Schouwenaars, T. and Feron, E., "Decentralized Cooperative Trajectory Planning of Multiple Aircraft with Hard Safety Guarantees," *Guidance, Navigation, and Control Conference and Exhibit*, Aug. 2004, pp. 2004–5141.

[19] Stipanovic, D. M., Hokayem, P. F., Spong, M. W., and Siljak, D. D., "Cooperative Avoidance Control for Multiagent Systems," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 129, No. 5, 2007, pp. 699.

[20] Massink, M. and De Francesco, N., "Modelling free flight with collision avoidance," *International Conference on Engineering of Complex Computer Systems*, 2001, pp. 270–279.

[21] Althoff, M. and Dolan, J. M., "Set-based computation of vehicle behaviors for the online verification of autonomous vehicles," *International Conference on Intelligent Transportation Systems*, Oct. 2011, pp. 1162–1167.

[22] Lin, Y. and Saripalli, S., "Collision avoidance for UAVs using reachable sets," *International Conference on Unmanned Aircraft Systems*, Jun. 2015, pp. 226–235.

[23] Lalish, E., Morgansen, K. A., and Tsukamaki, T., "Decentralized reactive collision avoidance for multiple unicycle-type vehicles," *American Control Conference*, Jun. 2008, pp. 5055–5061.

[24] Hoffmann, G. M. and Tomlin, C. J., "Decentralized cooperative collision avoidance for acceleration constrained vehicles," *Conference on Decision and Control*, 2008, pp. 4357–4363.

[25] Chen, M., Shih, C.-Y., and Tomlin, C. J., "Multi-Vehicle Collision Avoidance via Hamilton-Jacobi Reachability and Mixed Integer Programming," *Conference on Decision and Control (to appear)*, 2016.

[26] Barron, E. N., "Differential games with maximum cost," *Nonlinear Analysis*, Vol. 14, No. 11, Jun. 1990, pp. 971–989.

[27] Mitchell, I., Bayen, A., and Tomlin, C., "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *Transactions on Automatic Control*, Vol. 50, No. 7, Jul. 2005, pp. 947–957.

[28] Bokanowski, O., Forcadel, N., and Zidani, H., "Reachability and Minimal Times for State Constrained Nonlinear Problems without Any Controllability Assumption," *Journal on Control and Optimization*, Vol. 48, No. 7, Jan. 2010, pp. 4292–4316.

[29] Bokanowski, O. and Zidani, H., "MINIMAL TIME PROBLEMS WITH MOVING TARGETS AND OBSTACLES," *IFAC Proceedings Volumes*, Vol. 44, No. 1, Jan. 2011, pp. 2589–2593.

[30] Margellos, K. and Lygeros, J., "HamiltonJacobi Formulation for ReachAvoid Differential Games," *Transactions on Automatic Control*, Vol. 56, No. 8, Aug. 2011, pp. 1849–1861.

[31] Fisac, J. F., , M., Tomlin, C. J., and Sastry, S. S., "Reach-avoid problems with time-varying dynamics, targets and constraints," *International Conference on Hybrid Systems Computation and Control*, 2015, pp. 11–20.

[32] Sethian, J. A., "A fast marching level set method for monotonically advancing fronts." *National Academy of Sciences*, Vol. 93, No. 4, Feb. 1996, pp. 1591–1595.

[33] Osher, S. and Fedkiw, R., *Level Set Methods and Dynamic Implicit Surfaces*, Springer-Verlag, 2006.

[34] Mitchell, I. M., *Application of Level Set Methods to Control and Reachability Problems in Continuous and Hybrid Systems*, Ph.D. thesis, Stanford University, 2002.

[35] "A toolbox of level set methods," Tech. rep., 2007.

[36] Bayen, A. M., Mitchell, I. M., Osihi, M. K., and Tomlin, C. J., "Aircraft Autolander Safety Analysis Through Optimal Control-Based Reach Set Computation," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 1, Jan. 2007, pp. 68–77.

[37] Ding, J., Sprinkle, J., Sastry, S. S., and Tomlin, C. J., "Reachability calculations for automated aerial refueling," *Conference on Decision and Control*, 2008, pp. 3706–3712.

[38] Bouffard, P., *On-board Model Predictive Control of a Quadrotor Helicopter: Design, Implementation, and Experiments*, Master's thesis, University of California, Berkeley, 2012.

[39] Huang, H., Ding, J., Zhang, W., and Tomlin, C. J., "A differential game approach to planning in adversarial scenarios: A case study on capture-the-flag," *International Conference on Robotics and Automation*, 2011, pp. 1451–1456.

[40] Chen, M., Fisac, J. F., Sastry, S., and Tomlin, C. J., "Safe sequential path planning of multi-vehicle systems via double-obstacle Hamilton-Jacobi-Isaacs variational inequality," *European Control Conference*, Jul. 2015, pp. 3304–3309.

[41] Bansal, S., Chen, M., Fisac, J. F., and Tomlin, C. J., "Safe Sequential Path Planning of Multi-Vehicle Systems Under Presence of Disturbances and Imperfect Information," *American Control Conference (To appear)*, 2017.

[42] Chen, M., Bansal, S., Fisac, J. F., and Tomlin, C. J., "Robust Sequential Path Planning Under Disturbances and Adversarial Intruder," *arXiv preprint arXiv:1611.08364*, 2016.

[43] 3D Robotics, "Solo Specs: Just the facts," 2015, https://news.3dr.com/solo-specs-just-the-facts-14480cb55722#.w7057q926 [retrieved May 9, 2017].

[44] New Atlas, "Amazon Prime Air," http://newatlas.com/amazon-new-delivery-drones-us-faa-approval/36957/ [retrieved May 9, 2017].

[45] Wikipedia, "Beaufort scale," https://en.wikipedia.org/wiki/Beaufort_scale#Modern_scale [retrieved May 9, 2017].

Figure 8. Vehicle trajectories for Case-4: $d_r = 11m/s$, $t_i^{\text{STA}} = 10(i-1)$**. Since different vehicles have different scheduled times of arrival, there is a single lane between every origin-destination pair.**

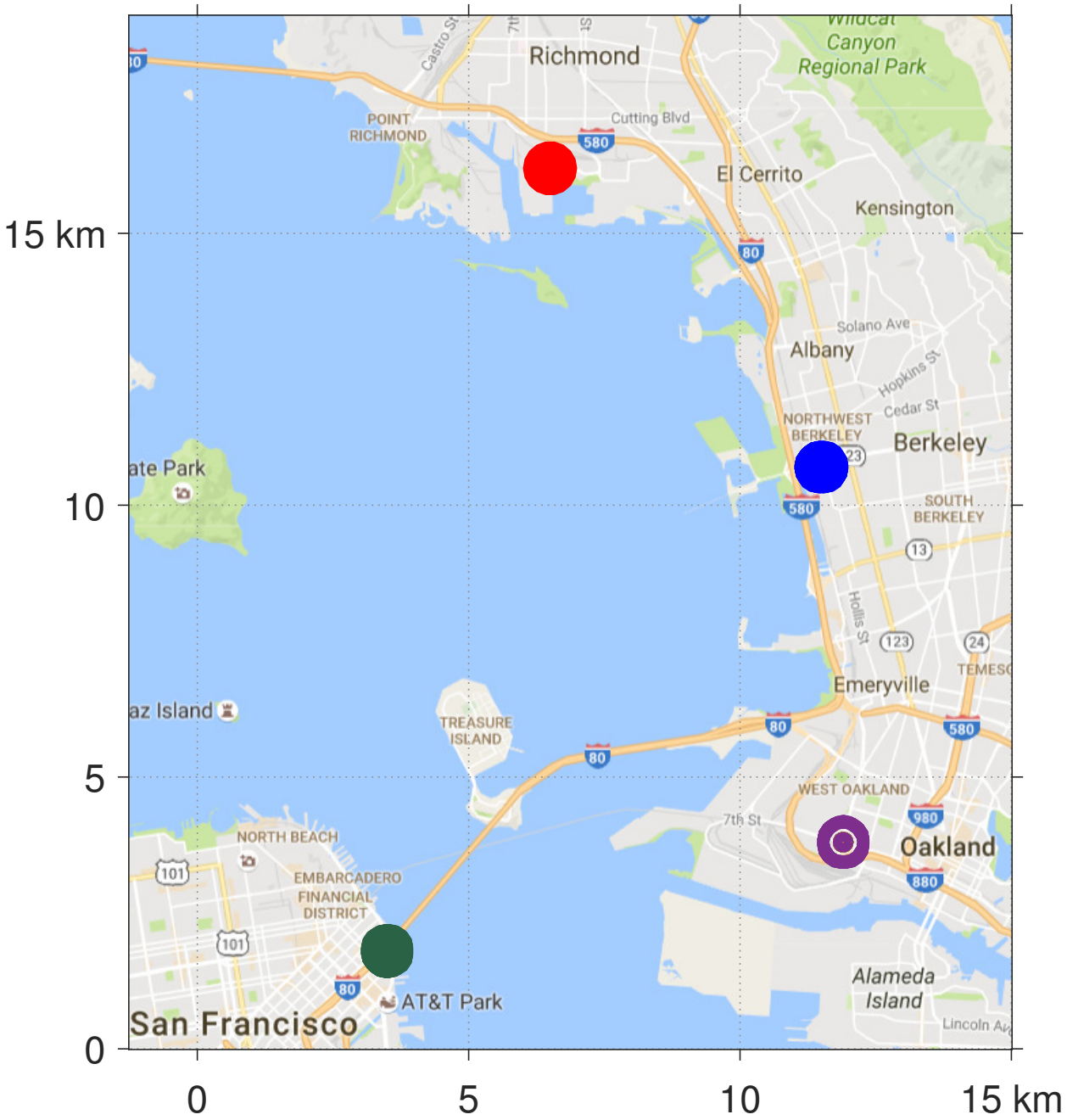


Figure 9. Multi-city simulation setup. A $300km^2$ area of San Francisco Bay Area is used as the state-space for vehicles. SPP vehicles fly to and from the four cities indicated by the four circles. The simulations are performed under the strong winds condition with $d_r = 11m/s$.

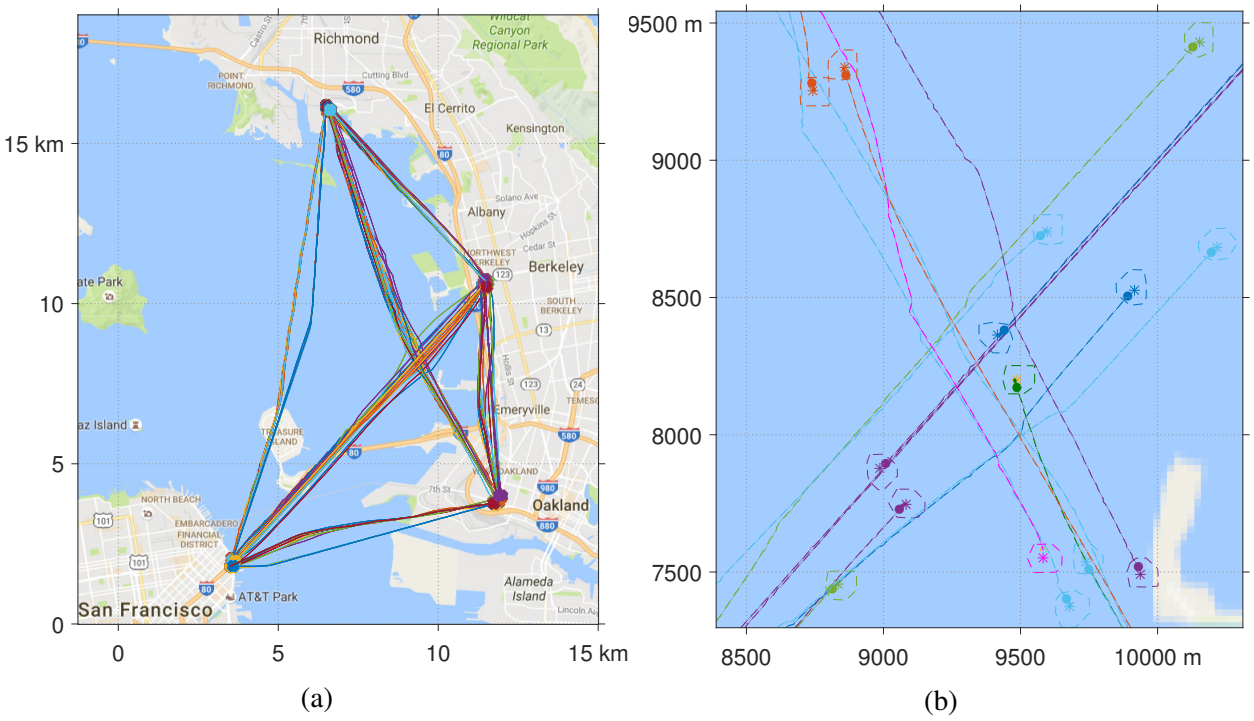


Figure 10. (a) Trajectories obtained from the SPP algorithm for the multi-city simulation with $d_r = 11m/s$, $t_i^{\text{STA}} = 5(i-1)$. (b) Zoomed-in version of the central area. A high density of vehicles is achieved at the center because of the intersection of several trajectories; however, the SPP algorithm still ensures that vehicles do not enter each other's danger zones and reach their destinations.

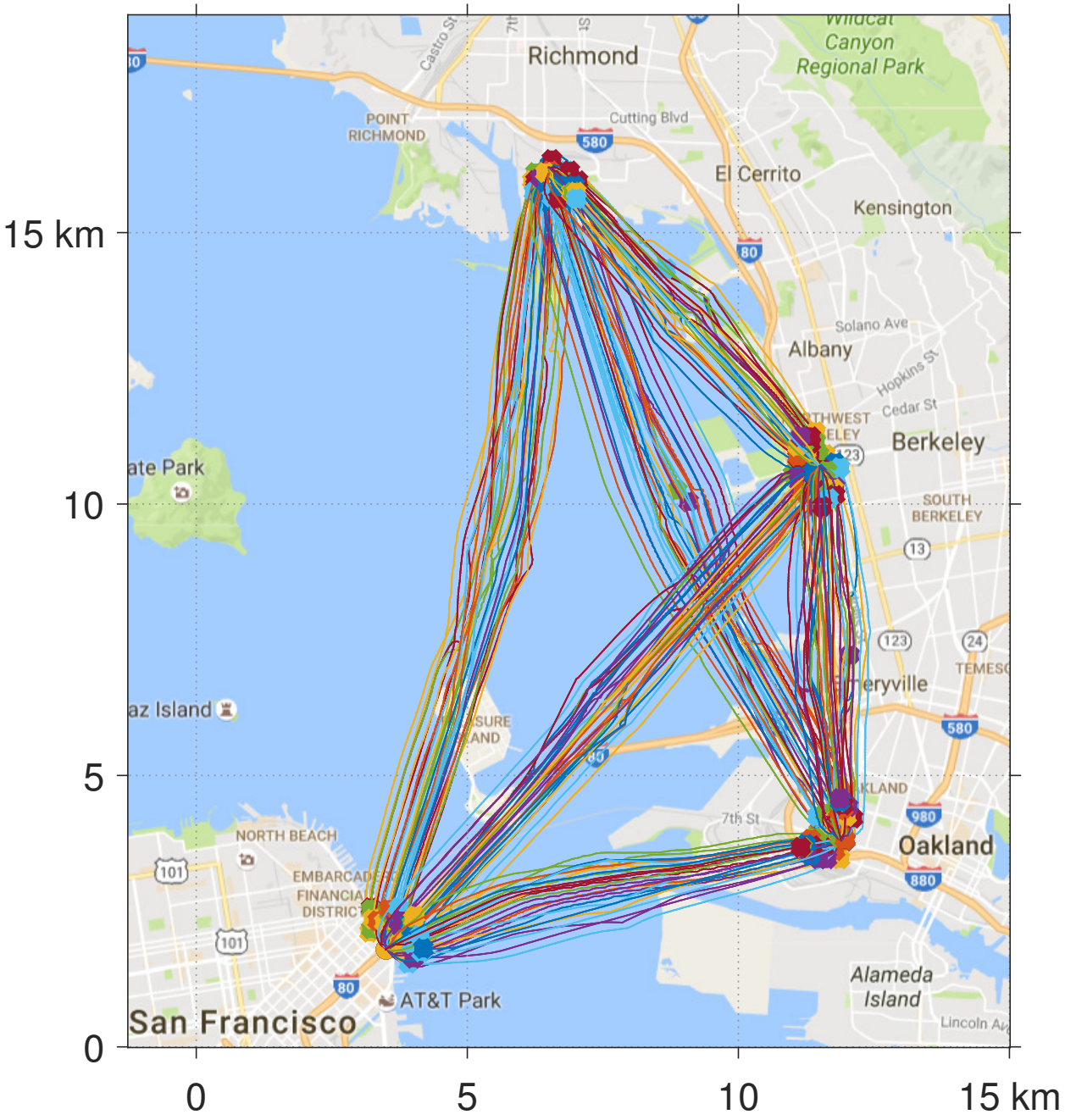


Figure 11. Vehicle trajectories for $d_r = 11m/s$, $t_i^{\text{STA}} = 0$. Since different vehicles have same scheduled times of arrival, a multiple-lane behavior is observed between every pair of cities.