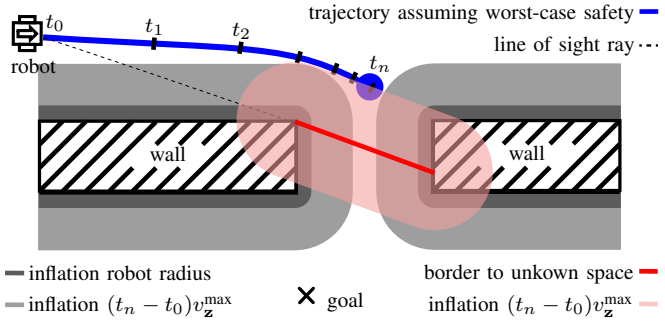
Just-in-Time Emergency Trajectories: A Formulation Towards Safety in Autonomous Navigation

George Todoran[1], Markus Bader[1]

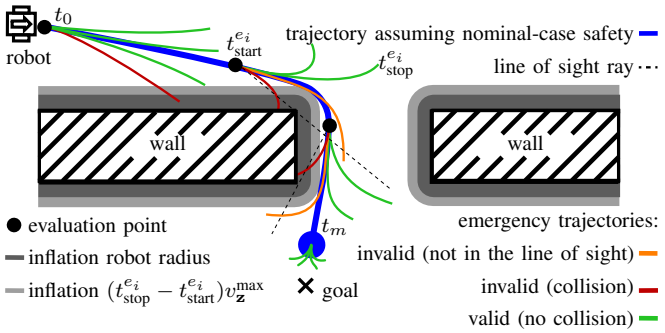
***Abstract*— Emergency trajectories enable one to move faster through an environment while still moving safely. Having an emergency trajectory within an observed vacant space makes it possible to safely navigate through unknown territory or through a door without slowing down. Emergency trajectories allow for safe navigation of a vehicle into a safe system state, e.g. a stop, in the event of recognition of an obstacle. This work formally proves the benefit of using emergency trajectories to generate safe and faster motion controls as compared to vehicle operation without such trajectories. Furthermore, this work also presents a working integration of this formalism into a vehicle's low level control system in a Moving Horizon Trajectory Planner (MHTP) with an update rate of 10Hz. Using an MHTP along with a dynamic model of the environment and the proposed constraints, the system is able to derive emergency trajectory candidates which fulfill our safety requirements. This distinguishes the approach from that of others, which re-plans discrete paths that are then followed by the vehicle's local control system. This approach was implemented on a differential-drive mobile agent and tested using non-static environment assumptions. Simulated and real-robot experimental results illustrate the quality of our approach.**

INTRODUCTION

In order to understand the approach to safe navigation presented here , one has to understand the difference between planning by considering only the worst case scenario and by being aware of a nominal expectation scenario. Fig. 1 shows trajectories for a worst case scenario in (a) and, in (b), a nominal planned trajectory for a vehicle and safety trajectories to select if an unforeseen, non-nominal event occurs. In (a) one can guarantee safety by assuring that the planned trajectory lies within visible, free space and that the vehicle's planned trajectory ends in a safe state, e.g a stop, at time t_n. Therefore we have to inflate all obstacles and the boundaries to unknown space by the distance an obstacle can move within time $t_m - t_0$ at velocity $v_{\mathbf{z}}^{\max}$. This velocity relates to the worst case random walk velocity assumed in an environment $\mathbf{z}$. In (b) we can guarantee safety by computing a nominal trajectory through (not necessarily visible) free space as long as there is a valid emergency trajectory into visible, free space for non-nominal events (moving obstacles). This also means that we only need to inflate the obstacles in an environment by the distance an obstacle can travel within the time it takes to trigger and complete such an emergency trajectory $t_{\text{start}}^{e_i} - t_{\text{end}}^{e_i}$. Because



(a) Worst case scenario: obstacles and shadow borders [1] need to be inflated by the planning horizon $t_n - t_0$, multiplied by the worst case random-walk speed $v_{\mathbf{z}}^{\max}$.



(b) Nominal scenario: Obstacles need to be inflated by the time required to trigger and finish an emergency trajectory $t_{\text{end}}^{e_i} - t_{\text{start}}^{e_i}$ multiplied by the worst case random-walk velocity $v_{\mathbf{z}}^{\max}$.

Fig. 1: Difference between planning under worst case assumption and nominal assumption on passing a doorway. The vehicle in (b) is able to maximize its travel speed while still moving safely and therefore faster than the vehicle in (a).

it's not necessary to plan for a stop within visible, free space, we are able to maximize travel speed by following optimized nominal trajectories.

It goes without saying that much research has been done regarding autonomous driving and mobile robotics navigation [2], [3], [4]. However, the industrial market, such as the logistics industry, still prefers Automated Guided Vehicles (AGVs) over autonomous vehicles due to their simplicity and, more importantly, due to their guaranteed behavior [5]. However, AGV systems are only economical in predictable, structured environments: work spaces where only trained humans are present. Issues with safety certification can be

*The research leading to these results has received funding from the Austrian Research Promotion Agency (FFG) according to grant agreement 855409 (AutonomousFleet) and 854865 (TransportBuddy)

[1] George Todoran and Markus Bader are with the Institute of Computer Engineering, Vienna University of Technology, Vienna 1040, Austria first.lastname@tuwien.ac.at

resolved by using devices such as a special laser range scanner directly connected to the wheel encoders and the braking/motor control system. If an obstacle appears within a predefined safety area of such a safety device, the vehicle will slow down or stop on the currently planned trajectory. An intelligent safety sensor is of course able to scale and transform safety areas according to the vehicle's current speed and angular velocity; nevertheless, in the event of a safety violation, the system will slow the vehicle down on the given trajectory or trigger a stop/hold. This behavior thus reduces the vehicle's speed near obstacles. For example, a door can only be passed if the safety area fits through the doorway; if not, the vehicle's speed must be reduced to decrease the safety area. The approach addressed here differs: our notion of safety relates to the vehicle's capability to deviate from its planned trajectory towards a safe location. Therefore, our system is able to navigate at higher speeds through narrow passages, as it ensures the existence of emergency trajectory candidates along the entire planned trajectory. The challenge of the approach proposed here lies in that fact the system has to derive not only a trajectory which guides the vehicle to a desired destination (nominal planned trajectory) but also a bundle of emergency trajectory candidates along the evaluation points of a nominal trajectory. Such emergency trajectory candidates have to be safe and valid. Safety and validity require that the emergency trajectory candidate be fully within the line of sight of the sensor readings as well as collision-free, as shown in Fig. 1.

In this approach, we use MHTP as a basis for implementing a system which ensures our notion of safety, enabling navigation of a real differential-drive robot. Therefore, we used simulated and real environments with a Pioneer robot to obtain qualitative and quantitative results.

This paper is organized as follows: Section II defines the agent-environment model as well as additional related concepts. Making use of those, Section III formulates two conditions that ensure safety. A practical modelling process and implementation of such constraints is presented in Section IV. Section V presents simulated as well as real-robot experiments. Conclusions are drawn in Section VI.

I. STATE OF THE ART

Offline computational methods for finding feasible navigation paths in dynamic environments have been proposed in many articles. In such approaches, a trajectory is calculated before motion begins [6]. Even though such approaches scale well with respect to planned trajectories' lengths, we believe they are not particularly suited to agents that are expected to navigate quickly, efficiently and in environments that do not have accurate models (unstructured). This motivates us to approach the problem from a receding-horizon perspective: one in which system dynamics is explicitly taken into account.

The paradigm behind MHTP is that by having knowledge of an agent model, one can predict sufficiently accurate outcomes of different commands over a larger *horizon* into the future. This results in the controller's capability of maintaining the satisfiability of the imposed constraints. It also results in an *optimized* trajectory with respect to some user-definable costs. This process is performed periodically and, at each iteration, the agent applies the initial controls of the planned trajectory.

A closely-related approach to MHTP is Model Predictive Control (MPC), the difference being that MPC provides an element of feedback through its repeated iteration, while the MHTP generally assumes an underlying feedback controller in the system. A thorough introduction of MPC in the context of autonomous navigation is given in [3].

Although sampling techniques such as the Dynamic Window Approach exist for searching for a solution in the context of MHTP, we focus on an approach in which the planned trajectory is optimized. This motivates the the oretical formulation of the system as a dynamic optimization problem [7], in practice being discretized and solved using a non-linear optimization solver. In our previous work [4], we developed an efficient way of implementing various optimization problems surrounding autonomous navigation.

It is well known that for moving-horizon controllers, even though a solution exists for iteration k, a solution is not necessarily guaranteed at iteration $k+1$ [8]. This fact motivates the search for a formulation that, when applied as an inequality constraint in an optimization problem, would always guarantee a solution. In his work, Schouwenaars made use of the concept of feasible invariant states: agent states that are guaranteed to be indefinitely collision-free. For example, a state in which a robot stops can be considered such a state. However, some platforms (such as fixed-wing UAVs) cannot maintain such a state. In a similar fashion, [9] makes use of the dual concept of feasible invariant states: inevitable collision states. In [10], Schouwenaars has extended the concept of the feasible invariant state to allow for periodic sequences of states (such as loiter maneuvers by a fixed-wing flying vehicle). Other approaches propose explicitly computing agent states' feasible invariant sets. Although computationally inexpensive during navigation, such approaches have the great limitation that they require accurate a priori knowledge of the environment (requiring most of the cases for it to be static).

In contrast to the aforementioned, our formulation does not assume that the environment is static. Also, most works assume that the environment is known or that the known area is simply a circular space centered at the agent. We directly approach this by taking into account the intrinsic method of visual sensing: the line of sight. Moreover, we propose a formulation that addresses the typical problem encountered when taking into account environment dynamics: the uncertainty of the environment state increases quickly during forward prediction, constraining the planning horizon of the trajectory generator.

The approach presented in [1] identifies structures within the field of view which could occlude dynamic obstacles yet unaccounted-for and hence critical to avoidance of collisions. This information is used to adapt a vehicle's velocity profile along a predefined path, with the extension of stochastic

adaptation of the path towards areas with fewer occlusions in order to maximize the vehicle's speed. Our approach differs from the aforementioned on two major points: first, by formally describing a nominal system behavior to adapt the vehicle's motion through maximization of not only the speed but also of the set of possible emergency trajectories. Note that such emergency trajectories can have arbitrary shapes, whereas in [1], slowing down to a stop is assumed in straight or circular paths. Second, the trajectory computed in [1] is actually a path with a velocity profile which needs to be followed by a local vehicle controller and adapted in the case of an event. The work presented here performs online re-planning and integrates everything into a trajectory planner utilizing MHTP.

II. THEORETICAL DEFINITION OF AGENT-ENVIRONMENT MODELS

Navigation is a task firmly tied to the concept of an environment, more specifically vacant space: locations within the pose space of an agent that are collision-free. As we want to be able to have a closer look at the environment subsystem, we choose to split the system into agent ($\mathbf{x}$) and environment ($\mathbf{z}$). To start, let us define the dynamics of the system in a continuous formulation. Keep in mind, however, that the following analysis can be easily transformed into a discrete system. We thus have

$$\dot{\mathbf{x}}(t) = \hat{\mathbf{f}}_{\mathbf{x}}(\mathbf{x}(t), \mathbf{u}(t)),\ \mathbf{x}(0) = \hat{\mathbf{x}}_0 \tag{1a}$$

$$\dot{\mathbf{z}}(t) = \hat{\mathbf{f}}_{\mathbf{z}}(\mathbf{z}(t), \mathbf{x}(t), t) + \mathbf{w}_{\mathbf{z}}(t),\ \mathbf{z}(0) = \hat{\mathbf{z}}_0 + \mathbf{w}_{\mathbf{z}_0}, \tag{1b}$$

with the (approximate) state transition function $\hat{\mathbf{f}}$, agent control input $\mathbf{u}$ as well as process noise $\mathbf{w}$. In order to simplify our analysis, we neglect agent process noise ($\mathbf{w}_{\mathbf{x}}(t) \ll \mathbf{w}_{\mathbf{z}}(t)$). Furthermore, $\hat{\mathbf{f}}_{\mathbf{x}}$ is independent of $\mathbf{z}$, as we do not take into consideration cases in which the environment actively modifies the state of an agent (e.g. we exclude situations in which an agent is moved or pushed by the environment).

A. Observers and simulators

As the paradigm of moving horizon trajectory generators (controllers) correlates to optimization or sampling *into the future*, it is convenient to define a short-hand notation of the agent and the environment ODE solutions.

$$\begin{aligned} &traj_{\mathbf{x}}(\mathbf{x}_0, t_0, \mathbf{u}, t_1) = \mathbf{x}(t_1) \text{ s.t. } \mathbf{x}(t_0) = \hat{\mathbf{x}}_0 \text{ and} \\ &\dot{\mathbf{x}} = \hat{\mathbf{f}}_{\mathbf{x}}(\mathbf{x}(t), \mathbf{u}(t)),\ \forall t \in [t_0, t_1], \end{aligned} \tag{2}$$

$$\begin{aligned} &traj_{\mathbf{z}}(\mathbf{z}_0, t_0, \mathbf{x}, t_1, \mathbf{w}_{\mathbf{z}}) = \mathbf{z}(t_1) \text{ s.t. } \mathbf{z}(t_0) = \hat{\mathbf{z}}_0 \text{ and} \\ &\dot{\mathbf{z}} = \hat{\mathbf{f}}_{\mathbf{z}}(\mathbf{z}(t), \mathbf{x}(t), t) + \mathbf{w}_{\mathbf{z}}(t),\ t \in [t_0, t_1]. \end{aligned} \tag{3}$$

In practice, $traj_{\mathbf{x}}$ and $traj_{\mathbf{z}}$ are computed using various types of ordinary differential equation solvers, ranging from the simple Euler method up to generic fixed-step Runge-Kutta or adaptive methods.

Even though in practice only the nominal simulation of a system is performed (i.e. $\mathbf{w} = \mathbf{0}$), we allow our environment simulator to take into account process noise, a formulation that will allow us later on to define a boundary of the guaranteed free space.

B. Agent-internal dynamic constraints

In order for the generated agent trajectory to be *feasible*, we have to satisfy a set of (equality $\mathbf{g}$, and inequality $\mathbf{h}$) constraints inherently present in the agent model. They are present typically due internal (unmodeled) dynamics as well as emerging from physical actuator constraints. We summarize them as follows by defining the boolean predicate C_{dyn}

$$\begin{aligned} C_{dyn}(\mathbf{x}) : \textbf{true} \text{ if } &\ \mathbf{g}_{dyn}(\mathbf{x}) = 0, \\ &\ \mathbf{h}_{dyn}(\mathbf{x}) \leq 0 \\ \textbf{false} \quad &\ otherwise \end{aligned} \tag{4}$$

Note that the constraints in (4) are independent of time and environment. Otherwise, additional constraints can be defined, as follows, along with other generally environment-dependent constraints.

C. Agent environment-dependent constraints

In a nutshell, the general constraint that we would like to impose is that of the agent always being in a valid state, i.e. its state is collision-free. Thus, we define $\mathcal{F}(\mathbf{z}(t))$ as the collision-free set of agent states with respect to the environment $\mathbf{z}(t)$ at time t. With this definition in mind, the constraint that we want to impose is

$$\mathbf{x}(t) \in \mathcal{F}(\mathbf{z}(t)), \quad \forall t \in [t_0, \infty). \tag{5}$$

Although succinct and general, this formulation has some major drawbacks: for one, it requires knowledge of the true environment state $\mathbf{z}(t),\ \forall t \in [t_0, \infty)$. In practice, this is rarely the case even for current time $t \to t_0$. Also, without any additional assumptions regarding the environment, there is no guarantee that there is a solution to (5).

Next, we will define several concepts that allow us to realistically and feasibly evaluate (5). They also constitute the main additional assumptions that we have to make regarding the nature of the environment $\mathbf{z}(t)$.

Assuming noise in the environment model

Depending on the applicability domain, environments can be arbitrarily complex; the design of qualitative environment models is a rich field for research on its own. However, all such models possess inaccuracies, especially when predicting states in the future. Nevertheless, we want to make use of their nominal prediction quality. Thus, we define the *nominal environment state* as follows:

$$traj_{\mathbf{z}}^{\mathcal{N}}(\mathbf{z}_0, t_0, \mathbf{x}, t_1) = traj_{\mathbf{z}}(\mathbf{z}_0, t_0, \mathbf{x}, t_1, \mathbf{0}). \tag{6}$$

Note that this set would coincide with the true free-space set if our environment model is perfect $\mathbf{w}_{\mathbf{z}}(\cdot) = \mathbf{0}$. However, otherwise, generally $\mathcal{F}(traj_{\mathbf{z}}^{\mathcal{N}}(\mathbf{z}_0, t_0, \mathbf{x}, t)) \not\subseteq \mathcal{F}(\mathbf{z}(t))$. In order to circumvent this problem, we require that our environment simulator also provide a function $C_{\mathbf{w}_{\mathbf{z}}}$ that enforces the bounds of the environment model initial state error and process noise. With this, we can define the *guaranteed free*

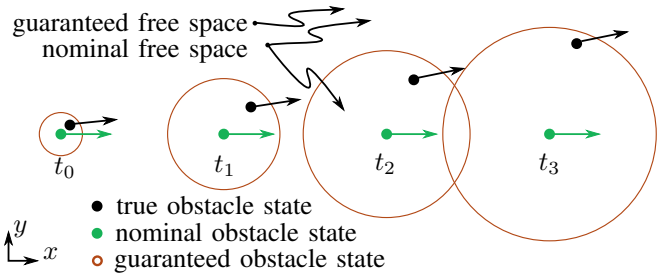


Fig. 2: Nominal versus guaranteed free space for multiple future simulated time-steps

space environment state as

$$traj^{\mathcal{G}}_{\mathbf{z}}(\hat{\mathbf{z}}_0, t_0, \mathbf{x}, t_1) = \mathbf{z} \text{ s.t.}$$
$$\mathcal{F}(\mathbf{z}) = \bigcap_{\mathbf{z}_0, \mathbf{w}_\mathbf{z}} \mathcal{F}(traj_{\mathbf{z}}(\mathbf{z}_0, t_0, \mathbf{x}, t_1, \mathbf{w}_\mathbf{z}))$$
$$C_{\mathbf{w}_\mathbf{z}}(\hat{\mathbf{z}}_0, \mathbf{z}_0, t_0, t, \mathbf{w}_\mathbf{z}(t)) \leq \mathbf{0}, \ \forall t \in [t_0, t_1]. \quad (7)$$

Eq. (7) can be interpreted as follows: given allowed variations of the initial state and process noise, the function provides the environment state having the free space set valid in all variations, i. e. the guaranteed free space. Of course, in practice, efficient computation of $traj^{\mathcal{G}}_{\mathbf{z}}$ would make use of simple noise models and stochastics. However, this would lead to relatively conservative bounds on the environment model error, which would result in $\mathcal{F}(traj^{\mathcal{G}}_{\mathbf{z}})$ greatly shrinking even after short periods of simulation. Fig. 2 illustrates the dual of the free space set (namely the obstacle set) for a nominal vs. guaranteed simulation.

Reducing the constraint evaluation for an infinite amount of time

The general idea is to assume that a set of agent safe states that are free and time-invariant with respect to the environment exists. This implies that if the agent enters into such a state at an arbitrary time, it can always remain in this set and be guaranteed to be free of collisions. Similarly to [8], we refer to such a set as *feasible invariant* $\mathcal{FI}$

$$\mathcal{FI}(\mathbf{z}_0, t_0) = \{\mathbf{x}_s \mid \exists \mathbf{u}(\cdot) \text{ s.t. } \mathbf{x}(t) = traj_{\mathbf{x}}(\mathbf{x}_s, t_0, \mathbf{u}, t),$$
$$\mathbf{z}(t) = traj^{\mathcal{G}}_{\mathbf{z}}(\mathbf{z}_0, t_0, \mathbf{x}, t),$$
$$C_{dyn}(\mathbf{x}(t)) = \mathbf{true},$$
$$\mathbf{x}(t) \in \mathcal{F}(\mathbf{z}(t)), \forall t > t_0\} \quad (8)$$

This formulation is useful, as for many environment models, computing a subset of the true feasible invariant set is reasonably straight-forward. One such example is to assume the stationary state of the agent (for example, zero velocity for a differential drive) for every free pose point with respect to a static map. This would apply, however, some constraints to the dynamics of the environment, namely that no obstacle should collide in the future with the agent, due to its stopping at an arbitrary vacant location on the static map.

We invite the reader to consider the aforementioned statement the other way around: by defining an environment's feasible invariant set, one arbitrarily constrains the environment type; however, as we will see in the next section, that would imply the existence of a solution to the moving horizon trajectory planning.

An additional notation that we would like to consider is the trajectory bundle which can move the agent from a state $\mathbf{x}_0$ to a feasible invariant state $\mathcal{TFI}$

$$\mathcal{TFI}(\mathbf{x_0}, \mathbf{z_0}, t_0) =$$
$$\{\mathbf{x} \mid \exists \mathbf{u}(\cdot), t_1 \text{ s.t. } \mathbf{x}(t) = traj_{\mathbf{x}}(\mathbf{x}_0, t_0, \mathbf{u}, t),$$
$$\mathbf{z}(t) = traj^{\mathcal{G}}_{\mathbf{z}}(\mathbf{z}_0, t_0, \mathbf{x}, t),$$
$$C_{dyn}(\mathbf{x}(t)) = \mathbf{true},$$
$$\mathbf{x}(t) \in \mathcal{F}(\mathbf{z}(t)),$$
$$\mathbf{x}(t_1) \in \mathcal{FI}(\mathbf{z}(t_1)), \ \forall t \in [t_0, t_1]\}. \quad (9)$$

III. CONSTRAINTS ENSURING SAFETY

In this section, we will make use of the definitions and assumptions from the previous section in order to guarantee the safety of a control law devised by an MHTP. We make the following assumptions regarding the MHTP:

- A1: The search for a control input $\mathbf{u}(\cdot)$ is performed at discrete time-steps at temporal intervals T_s.
- A2: MHTP computes a solution in zero time.
- A3: The agent model is exact.
- A4: The low-level controller precisely executes the command $\mathbf{u}_k(t)$ at time t, with $\mathbf{u}_k(\cdot)$ representing the input commands computed at time t_{0_k} that result in a valid trajectory.

Note that by assuming A3, it is easy to drop the assumption A2 and assume that MHTP computes a solution in at most T_s time. However, we require A2 for the sake of a cleaner formulation.

Under these assumptions, we would like to define a set of constraints so that if a control input $\mathbf{u}_0(\cdot)$ exists that satisfies them at the initial iteration of the MHTP (t_{0_0}), a control input $\mathbf{u}_k(\cdot)$ exists that satisfies them $\forall k > 0$. This allows us to provide a safety-guaranteeing constraint for moving horizon trajectory planning for finite horizon ($t_1 < \infty$) and accounting for environment process noise ($\mathbf{w}_\mathbf{z}(\cdot) \neq \mathbf{0}$), as follows:

Theorem III.1. *Worst-Case Safety Constraints*

Given are an agent and environment system of the form (1), as well as guaranteed bounds of the environment initial state error and process noise enforced through the constraint $C_{\mathbf{w}_\mathbf{z}}$. Moreover, the agent is controlled using an MHTP that satisfies A1-A4. If

$$\mathcal{TFI}(\mathbf{x}_{0_k}, \mathbf{z}_{0_k}, t_{0_k}) \neq \emptyset \quad (10)$$

at the initial MHTP iteration ($k = 0$) for the initial conditions $\mathbf{x}_{0_0}, \hat{\mathbf{z}}_{0_0}, t_{0_0}$, a pair $\mathbf{u}_k(\cdot), t_{1_k}$ exists that satisfies (10) $\forall k \geq 0$.

Proof. By construction, $\mathcal{TFI}$ represents the set of agent trajectories starting from a state consistent with the initial agent state $\mathbf{x}_{0_0}$ that are guaranteed to be collision-free and ending in a feasible invariant set. That implies that for any iteration thereafter, there exists at least the valid trajectory

that was computed in the initial iteration, which continues to be valid and collision-free. □

Although this formulation is safe, it possesses one drawback: it requires guaranteed simulation of the environment until the terminal state at time t_{1_k}. As discussed above, due to conservative bounds in environment model uncertainty, this formulation leads to considerable shrinkage of free space with respect to the guaranteed simulation in comparison with the free space of the real environment state. This would imply that in order to avoid a collision, t_{1_k} must be constrained by the "inflation" of the obstacle's predicted state. Conversely, this formulation makes no use of the quality of our nominal environment simulator, but rather of the guaranteed simulator. This motivates an alternative formulation of the safety constraints:

Theorem III.2. *Nominal-Case Safety Constraints*

Given are an agent and environment system of the form (1), as well as guaranteed bounds of the environment initial state error and process noise, enforced through the constraint $C_{\mathbf{w_z}}$. Moreover, the agent is controlled using a MHTP that satisfies A1-A4. If a solution $\mathbf{u}_0(\cdot), t_{1_0}$ exists that satisfies at the initial MHTP iteration ($k = 0$) for the initial conditions $\mathbf{x}_{0_0}, \hat{\mathbf{z}}_{0_0}, t_{0_0}$, a pair $\mathbf{u}_k(\cdot)$, t_{1_k} exists that satisfies (11) $\forall k > 0$.

$$
\begin{aligned}
&\mathbf{x}(t) = traj_{\mathbf{x}}(\mathbf{x}_{0_k}, t_{0_k}, \mathbf{u}_k, t),\\
&\mathbf{z}_{t-T_s} = traj^{\mathcal{N}}_{\mathbf{z}}(\hat{\mathbf{z}}_{0_k}, t_{0_k}, \mathbf{x}, t - T_s)\\
&\bar{\mathbf{z}}(t) = traj^{\mathcal{G}}_{\mathbf{z}}(\mathbf{z}_{t-T_s}, t - T_s, \mathbf{x}, t)\\
&\mathcal{TFI}(\mathbf{x}(t), \bar{\mathbf{z}}(t), t) \neq \emptyset,\ \forall t \in [t_{0_k} + T_s, t_1)
\end{aligned} \tag{11}
$$

Proof. We assume that (11) holds at an iteration $k-1$ starting at time $t_{0_{k-1}} = t_{0_k} - T_s$. This implies that a trajectory induced by $\mathbf{u}_{k-1}(t),\ t \in [t_{0_k} - T_s, t_{0_k})$ exists and $\mathbf{u}'_{k-1}(t),\ t \in [t_{0_k}, t'_1]$ such that the agent is guaranteed to be safe and enters into a feasible invariant state at t'_1. The reason this is always the case is the fact that when evaluating (11) at iteration $k-1$, for $t = t_{0_{k-1}} + T_s = t_{0_k}$, the nominal simulation is performed for $\Delta t = 0$, i.e. $\mathbf{z}_{t-T_s} = \hat{\mathbf{z}}_0$. This implies that the initial environment state evaluated in $\mathcal{TFI}$ is guaranteed. Thus, the MHTP at iteration k will find at least $\mathbf{u}_k(t) = \mathbf{u}'_{k-1}(t)$, which continues to satisfy (11). □

This formulation can be interpreted as follows: we assume that our nominal simulator is perfect and constrains our motion accordingly. However, if our nominal simulator deviates from the real environment in future iterations, we will always have a (guaranteed) safe trajectory that we can follow. An illustration of simulations performed in (11) on a temporal scale is provided in Fig. 3.

IV. A PRACTICAL NON-TRIVIAL EXAMPLE

Now we would like to go through the modelling and definition of the navigation task using a practical example. Note that some assumptions and models regarding the agent and the environment are similar to the ones shown in Fig. 1b.

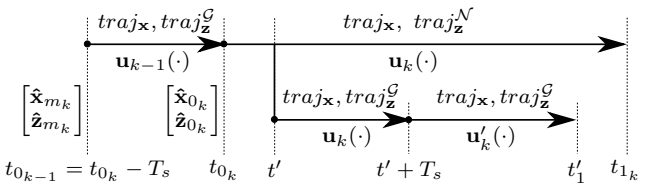


Fig. 3: Timeline of agent and environment simulations when enforcing constraint (11). Note that when using the assumption A2, the computation cycle starts and ends at t_{0_k} with measurements $\begin{bmatrix}\hat{\mathbf{x}}_{0_k} & \hat{\mathbf{z}}_{0_k}\end{bmatrix}^{\mathrm{T}}$. The top timeline represents the nominal trajectory of the agent and its environment. The bottom timeline represents an emergency trajectory that starts at t'. Note that for the initial duration T_s of the emergency trajectory, it evolves as the nominal trajectory using $\mathbf{u}_k(\cdot)$, however with a guaranteed environment simulation. Beyond this temporal point, the trajectory alters according to $\mathbf{u}'_k(\cdot)$ towards a feasible invariant state. As mentioned, A2 can be easily dropped by performing a guaranteed simulation with the control inputs from the previous cycle. In this case, the beginning of the computation cycle is at $t_{0_{k-1}}$ with measurements $\begin{bmatrix}\hat{\mathbf{x}}_{m_k} & \hat{\mathbf{z}}_{m_k}\end{bmatrix}^{\mathrm{T}}$.

We define the agent using a differential-drive model

$$\mathbf{x} = \begin{bmatrix} x & y & \theta & v & \omega \end{bmatrix}^{\mathrm{T}} \tag{12}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} v\cos(\theta) & v\sin(\theta) & \omega & u_v & u_\omega \end{bmatrix}^{\mathrm{T}}. \tag{13}$$

This motion model is subject to various kinematic constraints such as wheel maximum velocity, maximum acceleration, maximum angular velocity, as well as maximum lateral acceleration. Regarding the environment, we assume that:

- the agent kinematic sub-state $\begin{bmatrix} v & \omega \end{bmatrix}^{\mathrm{T}} = \mathbf{0}$ is a candidate for a feasible invariant state.
- if the agent reaches a free-space state with the candidate feasible invariant state, it is in a feasible invariant state.
- the observed initial state of the environment is error-free for the visible region of the on-board sensors.

Note that the last assumption regarding the visible region of the on-board sensors is typically neglected in other navigation approaches. However, we consider it important especially for navigation in narrow or cluttered environments at moderate to high speeds. Without any additional information, we can formulate the nominal initial state of the environment

$$\begin{aligned}\mathbf{z_0} = \{\mathbf{p}^i \mid \forall i, &\text{ an obstacle is sensed at } \mathbf{p}^i \text{ or}\\ &\mathbf{p}^i_0 \text{ is an obstacle in the static map}\},\end{aligned} \tag{14}$$

i. e. the set of all expected obstacle points $\mathbf{p}^i$ (in 2D space) relating to an occupied location. Next, we have to define a dynamic model of the environment. For simplicity's sake, we will assume that the obstacles are subject to a bounded random-walk model (with the maximum random-walk velocity $v^{max}_{\mathbf{z}}$). This implies that our nominal simulator is

$$\mathbf{z}(t) = \{\mathbf{p}^i \mid \mathbf{p}^i \in \mathbf{z_0}\} \tag{15}$$

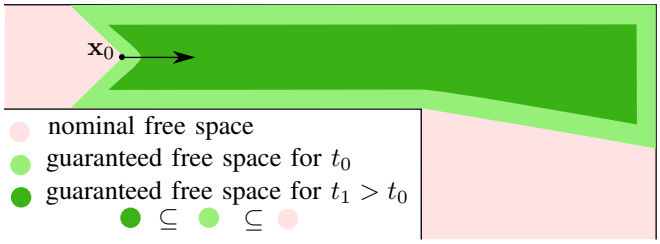


Fig. 4: Nominal and guaranteed free space with respect to the initial agent state, taking into account sensor visibility and a bounded random-walk environment model.

and the guaranteed simulator is

$$\begin{aligned}\mathbf{z}(t) = \{\mathbf{p}^i \mid \forall i,\ \exists \mathbf{p}^j \in \mathbf{z}_0 \text{ s.t.} \\ (||\mathbf{p}^i - \mathbf{p}^j|| \leq v_{\mathbf{z}}^{max}(t-t_0) \wedge \\ \forall \theta,\ \mathbf{x}(t) \neq \begin{bmatrix}(\mathbf{p}^i)^T & \theta & 0 & 0\end{bmatrix}^{\mathrm{T}}) \vee \\ \mathbf{p}^i \text{ not visible from } \mathbf{x}(t_0)\}. \end{aligned} \quad (16)$$

An illustration of the resulting free space from the two simulators is presented in Fig. 4. Exploiting the fact that the environment nominal state is static, one can make use of precomputation for various spatial functions on a discretized 2D grid at the beginning of each MHTP cycle. One such function is the Euclidean Distance Transform [11] (edt)

$$d_{edt}(\mathbf{x}) = \min\{\ ||\mathbf{x} - \mathbf{p}||_2 \mid \forall \mathbf{p} \in \mathbf{z_0}\}. \quad (17)$$

In practice, a discretized solution of d_{edt} is computed on a 2D grid. Thus, one has to perform (bi-linear) interpolation in order to use this function in a continuous formulation. Note, however, that in the random-walk model, the guaranteed minimum distance from a point $\mathbf{p}$ to the closest obstacle point is $d_{obst}^{min}(\mathbf{x}, \Delta t = t - t_0) = max(0, d_{edt}(\mathbf{x}) - v_{\mathbf{z}}^{max}\Delta t)$.

Additionally, we require a measure to evaluate whether a point $\mathbf{p}$ will be visible from a future agent location $\mathbf{x}$. Thus, we define the function viz as

$$viz(\mathbf{x}, \mathbf{x}_1) = \min\{\ d_{obst}^{min}(\mathbf{x}', t - t_0) \mid \forall \mathbf{x}' \in [\mathbf{x}, \mathbf{x}_1]\}, \quad (18)$$

with time t of state $\mathbf{x}$ and the line segment $[\mathbf{x}, \mathbf{x}_1]$ between the positions of $\mathbf{x}$ and the endpoint $\mathbf{x}_1$. Note that this function evaluates to 0 only if the endpoint $\mathbf{x}_1$ is not visible from location $\mathbf{x}(t)$ at time t. For efficient implementation, one could make use of the edt to skip evaluation points when obstacles are far away from the evaluated line segment.

Now, let us have a closer look at the $\mathcal{TFI}$ in order to efficiently evaluate it. Exploiting the fact that a feasible invariant state can be located at any free point, one can argue that it is sufficient to sample a sub-set of $\mathcal{TFI}$ for which the duration of the trajectories is small. Such emergency trajectories are those which quickly move the agent from a state $\mathbf{x}_0$ into a feasible invariant state. As the trajectory of the agent depends solely on its input, it is motivated to precompute such trajectory candidates for discretized values of initial kinematic state $\begin{bmatrix}v & \omega\end{bmatrix}^{\mathrm{T}}$ in an obstacle-free environment, with the pose of the agent coinciding with the origin. Thus, in order to evaluate whether such a trajectory exists $\in \mathcal{TFI}$, one just has to perform a rigid transformation of the trajectory and then make use of the previously-defined d_{obst}^{min} and viz.

Next, we refer to $\mathbf{x_{tf}}_i^j$ as the sampled point $1 \leq j \leq m$ along the precomputed trajectory $1 \leq i \leq n$ after the rigid transformation in the coordinate frame $\mathbf{x}(t_0)$. Finally, our function evaluating whether any trajectory towards a feasible invariant state exists is

$$\frac{1}{n}\sum_{i=1}^{n} \sqrt[m]{\prod_{j=1}^{m} d_{obst}^{min}(\mathbf{x_{tf}}_i^j, t) viz(\mathbf{x}(t), \mathbf{x_{tf}}_i^j)} \geq 0. \quad (19)$$

Note that this function has the value of zero if at least one point along every trajectory i is inside an obstacle or not visible from $\mathbf{x}(t)$. Moreover, this function is continuous in $[x_0, y_0, \theta_0]$. Using bi-linear interpolation in the dimensions v_0 and w_0 of the precomputed trajectories' sets, the function is continuous in all of the agent state dimensions. Thus, it can be used to enforce the safety constraints while solving a gradient-based optimization problem.

An illustration of the planned nominal trajectory and the overlay of the sampled 'emergency' trajectories and their validity is provided in Fig. 1.

V. EXPERIMENTAL RESULTS

After an introduction to implementation specifics, we would like to present results from two scenarios: a cornering maneuver using simulation and navigation through narrow passages using a real robot.

The implementation of the work presented here was done in C++, making use of GazeboSim as a simulation environment and Robotics Operating System (ROS). We developed an MPC library that allows for convenient definition and formulation of various cost functions and constraints. Moreover, this library uses multi-lattice evaluation (equal time, equal distance, parametric function knots, etc.). Details regarding the specifics of the MPC library can be found in our previous work [4]. The MPC typically runs stably at a frequency of 10 Hz on a single core of an Intel i7 processor.

As discussed in Section IV, especially when evaluating nominal safety constraints, it is vital to make use of precomputed lookup tables. Because of this, we chose to split the computation of various functions over discretized grids from the general MPC library. Thus, we use a second core for computing Euclidean distance transforms, functions based on fast-marching methods as well as gradients thereof in every cycle. In this module, we make extensive use of the *GridMap* library [12].

The agent used for (simulated and real-world) testing is a Pioneer P3DX. In the experiments presented here, we limit the agent wheel angular velocity to $10rad/s$ (corresponding to a maximum linear velocity of $1m/s$), wheel angular acceleration to $3rad/s$, and maximum lateral acceleration to $0.5m/s^2$.

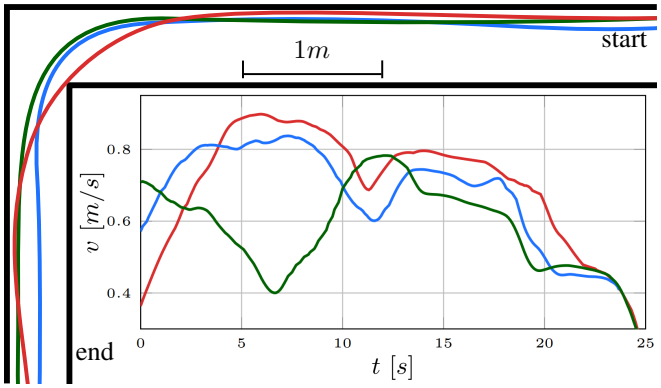


Fig. 5: Evaluation of different safety constraints during a cornering maneuver in a static (partially) unknown environment. *Top-Left*: Trajectories executed using 0 end velocity constraint (*red*), nominal safety constraints (*blue*), and guaranteed safety constraints (*green*). *Bottom*: Velocity profile of respective trajectories.

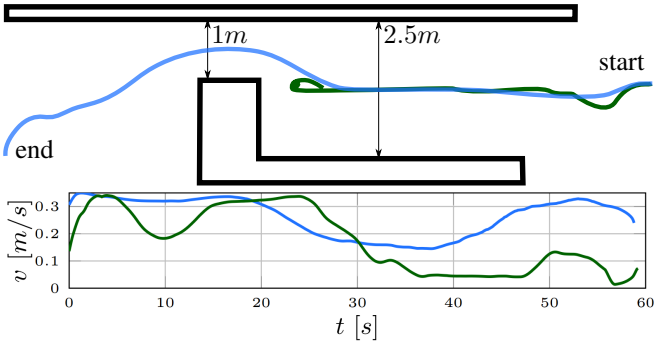


Fig. 6: Navigation under uncertainty. *Top*: Trajectories executed using the nominal safety constraints (*blue*) and guaranteed safety constraints (*green*). *Bottom*: Velocity profile of the respective trajectories

A. Cornering maneuver

In this experiment, we compare resulting trajectories when undertaking a cornering maneuver in a confined space. We assume that the environment is static but (partially) unknown. We perform the experiment using three different formulations:

- F1: guaranteed-safe constraints
- F2: nominally-safe constraints
- F3: linear velocity end-state constraint.

Keep in mind that when the environment model is static *and known*, F3 guarantees safety. However, in this scenario, it is possible that an obstacle is present just around a corner, a situation in which F3 results in a collision. Fig. 5 illustrates the three different trajectories as well as the agent linear velocity during the maneuver. As expected, F3 has the highest velocity profile and the trajectory resembles a time-optimal maneuver: the radius of the turn is maximized because the lateral acceleration constraint is active. Note, however, that the F2 velocity profile closely resembles that of F3. The trajectory performs a motion that increases visibility around the turn, a fact to be expected based on the problem formulation. The velocity profile of F1 is considerably different. The reason for this is that, according to the constraint formulation, the predicted trajectory always has to remain within the visible area. This induces a strong deceleration before the turn, hence the difference in velocity profile.

B. Traversing narrow areas

We evaluate motions resulting from F1 and F2 under the random-walk model in narrow spaces. We set the maximum random-walk velocity to $0.5m/s$ and want to evaluate agent trajectories and velocities in corridors (doorways) of $1-2m$. Of course, in such scenarios where the free-space width is small relative to the random-walk maximum velocity, a better model of the environment would be desirable (e.g. a constant velocity model). Nevertheless, this scenario would allow us to validate expected behaviours. Fig. 6 presents the resulting trajectories and velocity profiles.

In the case of F2, the trajectory is as expected: in order to maximize velocity, the agent should maximize its distance from obstacles (thus being allowed to decelerate towards a feasible invariant state for a longer period of time). Even though the trajectory closely resembles the Voronoi path, note that this behaviour is implicit through the formulated constraints and environment model. The velocity profile is also as expected: in wider sections, the agent has a higher velocity.

F1, however, fails to reach the goal. The reason for this is that due to aggressive inflation of the environment throughout the forward simulation, the planned trajectory is very short: in this experiment, it has an average of $0.1m$ (in comparison with that of F2, which averaged at $1.7m$). This is principally because even though we make use of a non-trivial cost-function in our optimization problem (a discretized solution of the Eikonal equation), the non-linear solver is not guaranteed to find a global minimum for the problem.

VI. CONCLUSIONS

In this paper, we addressed the problem of guaranteeing safety in the context of autonomous navigation using MHTP. We started from general models of the agent and the environment (allowing the environment to be dynamic and assuming inexact knowledge regarding its model). From this, we defined assumptions that have to be made and provided two formulations (in the form of constraints) that, if satisfied once, will continue to be satisfied indefinitely, thus ensuring safety for the navigation task. As previously discussed, although more computationally expensive, the proposed nominal safety constraint allows for planning trajectories for arbitrary horizons even if the uncertainty of the environment prediction is non-negligible.

We have thus provided a relatively simple example of putting theory into practice. The devised model was then

subjected to simulated and real-robot experiments, most of which validate our expectations regarding their quality and/or behavior.

In practice, the nominal safety constraint possesses one drawback: the existence of trajectories towards a feasible invariant state has to be investigated for all the states along the trajectory. Future work is expected in the context of efficient evaluation of such trajectories (for example by parallelizing their evaluation). Moreover, the technique presented here of precomputing trajectory candidates towards feasible invariant states is not generally applicable to cases in which regions in the environment are explicitly excluded from the feasible invariant set. One example could be an expert explicitly not allowing agents to reside in certain regions of the environment. Another problematic situation would be in instances in which the trajectories towards feasible invariant states do not tend to have a short duration.

Last but not least, extensive testing in cluttered human environments is planned in order to evaluate the proposed approach using more complex dynamic environment models.

REFERENCES

[1] K. M. Krishna, R. Alami, and T. Simeon, "Safe proactive plans and their execution," *Robotics and Autonomous Systems*, vol. 54, no. 3, pp. 244 – 255, 2006.

[2] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window Approach to Collision Avoidance," *Robotics Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23–33, March 1997.

[3] T. Howard, "Adaptive model-predictive motion planning for navigation in complex environments," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2009.

[4] G. Todoran and M. Bader, "Expressive Navigation and Local Path-Planning of Independent Steering Autonomous Systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)*, Daejeon, Korea, October 2016, pp. 4742–4749.

[5] M. Bader, A. Richtsfeld, M. Suchi, G. Todoran, W. Holl, W. Kastner, and M. Vincze, "Balancing Centralized Control with Vehicle Autonomy in AGV Systems," in *Proceedings 11th International Conference on Autonomic and Autonomous Systems (ICAS)*, vol. 11, May 2015, pp. 37–43.

[6] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5628–5635.

[7] B. Chachuat, *Nonlinear and Dynamic Optimization: From Theory to Practice - IC-32: Spring Term 2009*, ser. Polycopiés de l'EPFL. EPFL, 2009.

[8] T. Schouwenaars, E. Feron, and J. How, "Safe receding horizon path planning for autonomous vehicles," January 2002.

[9] L. Martinez-Gomez and T. Fraichard, "Collision avoidance in dynamic environments: An ics-based solution and its comparative evaluation," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, ser. ICRA'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 2251–2256.

[10] T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees," in *American Control Conference, 2004. Proceedings of the 2004*, vol. 6. IEEE, 2004, pp. 5576–5581.

[11] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, "2d euclidean distance transform algorithms: A comparative survey," *ACM Comput. Surv.*, vol. 40, no. 1, pp. 2:1–2:44, Feb. 2008.

[12] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, 2016, ch. 5.