Occlusion-Aware MPC for Guaranteed Safe Robot Navigation with Unseen Dynamic Obstacles

Roya Firoozi[1], Alexandre Mir[2], Gadiel Sznaier Camps[1], Mac Schwager[1]

***Abstract*— For safe navigation in dynamic uncertain environments, robotic systems rely on the perception and prediction of other agents. Particularly, in occluded areas where cameras and LiDAR give no data, the robot must be able to reason about potential movements of invisible dynamic agents. This work presents a provably safe motion planning scheme for real-time navigation in an a priori unmapped environment, where occluded dynamic agents are present. Safety guarantees are provided based on reachability analysis. Forward reachable sets associated with potential occluded agents, such as pedestrians, are computed and incorporated into planning. An iterative optimization-based planner is presented that alternates between two optimizations: Nonlinear Model Predictive Control (NMPC) and collision avoidance. Recursive feasibility of the MPC is guaranteed by introducing a terminal stopping constraint. The effectiveness of the proposed algorithm is demonstrated through simulation studies and hardware experiments with a TurtleBot robot equipped with a LiDAR system. A video of experimental results is available at** `https://youtu.be/OUnkB5Feyuk`.

I. INTRODUCTION

Robotic systems rely on various types of sensors including range-based sensors such as LiDAR or depth cameras to perceive the surrounding environment. However, sensors are susceptible to occlusion. The field of view of the robot's sensors limit its visibility, and dynamic agents can hide undetected in occluded regions. In a dynamic environment, safe trajectory planning for a robot requires predicting the future behavior of other agents present in the scene. However, occluded areas may hide undetected dynamic agents, whose trajectory a robot cannot predict. Therefore, to safely navigate in an a priori unmapped environment among dynamic agents where occlusions are present, a robot must reason about all potential future motions of all potential dynamic agents that might be hidden in the occluded regions.

This work proposes a framework for general robotics settings including ground robots, autonomous cars or aerial robots to safely travel through the occluded environment in real-time, while avoiding both static and dynamic obstacles. Our approach relies on real-time sensor observations to detect occluded region boundaries from which a potential moving agent might emerge at any time in the future. Forward reachability analysis is performed for dynamic obstacles in both visible and occluded areas. The robot

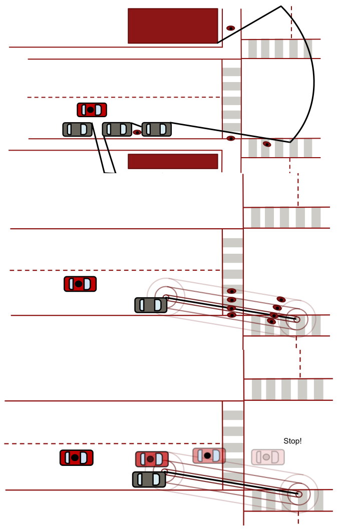

Fig. 1: The ego vehicle (red) equipped with a range-based sensor scans the visible regions. **Top:** Pedestrians (small red circles) who are hidden in an occlusion are not detected by the sensor. **Middle:** Nested capsules representing the forward reachable sets for all pedestrians potentially hidden in the occlusion are computed. **Bottom:** The vehicle optimizes its trajectory while avoiding the nested capsules along its prediction horizon, coming to a stop at the end of the horizon.

optimizes a trajectory in an MPC fashion which attempts to reach its goal position as fast as possible while it avoids collision with the forward reachable sets that are computed based on the last updated LiDAR scan of the environment. Fig. 1 illustrates an example of our approach in an autonomous driving scenario. The top figure shows that the ego vehicle (red) uses its range sensor (LiDAR or depth camera) to scan the visible regions (visible region boundary is shown as solid black line). Dynamic agents or pedestrians (shown as small red circles) who are outside the visible region are not detected by sensors. To reason about the movement of potential pedestrians in the occluded region, the robot assumes their worst case behavior, which is moving with maximum speed in any direction from the occlusion boundary. The middle figure shows capsules which are the pre-computed reachable sets resulting from this worst case assumption on the undetected agents in the occlusion. In the bottom figure, MPC prediction steps are illustrated, where the vehicle avoids colliding with the growing reachable capsules over the prediction horizon. Also, we find that requiring the robot to come to a stop at the end of its prediction horizon ensures recursive feasibility for the MPC, although, due to the MPC re-solve at each time step, the vehicle does not actually come to a stop during normal execution. Planning

[1] Department of Aeronautics & Astronautics, Stanford University `rfiroozi, gsznaier, schwager@stanford.edu`

[2] Mechanical Engineering Department, ETH Zurich `alexmir@student.ethz.ch`

This work was supported in part by ONR grant N00014-18-1-2830. Toyota Research Institute provided funds to support this work. The first author was supported on an ASEE eFellows fellowship.

to stop can be seen as a guaranteed safe contingency plan in the unlikely event that the worst-case prediction comes true.

The contributions of the proposed approach are summarized below:

- Line-of-Sight of LiDAR at the boundary of occluded region is used to determine worst-case location of all potential occluded agents over a time horizon.
- Reachability analysis is provided for collision avoidance with both visible and hidden dynamic agents in the environment.
- An efficient iterative optimization is proposed for fast online computation.
- A formal guarantee for recursive feasibility (which ensures collision avoidance) of the proposed MPC scheme is provided.

The paper is organized as follows: Sec. II reviews the related work. Sec. III provides the required notations. Sec. IV describes the problem definition. Sec. V provides recursive feasibility guarantees for the proposed MPC scheme. Sec. VI presents our proposed iterative optimization algorithm. Sec. VII presents the applications and numerical results based on simulation studies and real world experiments. Sec. VIII makes concluding remarks and discusses the future work.

II. Related work

Hamilton-Jabobian (HJ) reachability is a standard method to analyze safety, but it is computationally expensive. In particular, for real-time applications such as robot navigation, to address this challenge and reduce the online computation burden, the authors in [1] propose precomputation of HJ reachable sets offline and storing them in a look-up table to be used online. In [2], the authors present fast and safe tracking approach (FaSTrack) in which slow offline computation provides safety guarantees for a faster online planner. In [3], the authors propose to extend FaSTrack approach to a meta-planning scheme in which a refined offline computation enables safe switching between different online planners. However, these approaches are restricted to static environments and are not applicable to dynamic environments with moving obstacles. In [4], the authors present a pursuer-evader game theoretic framework for occlusion-aware driving (in dynamic environment) with safety guarantees using reachability, and simulation studies are provided by assuming linear dynamics (double integrator) for the vehicle model.

Joint active perception and planning helps the robot to gather additional information about the occluded regions and gain visibility in a partially occluded environment. The authors in [5], propose a perception-aware MPC framework for quadrotors that simultaneously optimizes planning and perception objectives. In [6], an MPC framework is coupled with a visual pipeline. A deep optical flow dynamics presented as a combination of optical flow and robot dynamics is used to predict the movement of points of interest. The proposed Pixel MPC algorithm controls the robot to accomplish a high-speed task while maintaining visibility of the important features. In [7] a geometric heuristic is proposed to define a visibility measure around a corner. The perception objective (visibility) is added to the MPC multi-objective cost to navigate the occluded corner while maximize visibility. In our proposed approach visibility maximization is not incorporated explicitly. However, by including a minimum safety distance to the avoid set, the robot indirectly maximizes its visibility to minimize the occluded area.

Other works including [8] and [9], These works minimize the risk of collision but they do not provide formal safety guarantees. In [10], MPC planning is proposed in which the objective is to maximize visibility, but formal safety guarantees or recursive feasibility of MPC is not provided. Compared to these online navigation approaches, our proposed MPC provides safety guarantees by online computation of reachable sets.

In [11], an inference and motion planning scheme is proposed to guarantee safety with respect to hypothetical hidden agents. Reachable sets are constructed by taking an occupancy map and enlarging the occluded boundaries. Compared to this work, our approach does not make any assumption regarding the intention of hidden agents. Our proposed framework is dynamic-agnostic with respect to hidden agents and assumes the hidden agents may move in any arbitrary direction.

III. Notation

The required definitions and notations are provided below.
MPC Scheme: MPC is useful for online motion planing among dynamic obstacles because it is able to re-plan according to the newly available sensory data. MPC relies on the receding-horizon principle. At each time step it solves a constrained optimization problem and obtains a sequence of optimal control inputs that minimize a desired cost function l, while considering dynamic, state, and input constraints, over a fixed time horizon. Then, the controller applies, in closed-loop, the first control-input solution. At the next time step, the procedure is repeated. In this paper, $\mathbf{u}_{0:N-1}$ and $\mathbf{z}_{0:N}$ indicate the input and state values along the entire planning horizon N, predicted based on the measurements at time t. For example $[\mathbf{z}_{0|t}\ \mathbf{z}_{1|t},...,\mathbf{z}_{N|t}]$ represents the entire state trajectory along the horizon N predicted at time t (referred to as the open-loop trajectory). Throughout this paper τ represents the absolute time, t is the MPC execution time and k is the open-loop prediction step for a given time t. The static obstacles are denoted as $\mathcal{O}$ and dynamic agents' reachable sets are represented as Ω. So the unsafe set is defined as union of these two sets $\Omega \cup \mathcal{O}$. The safe set $\mathcal{Z}^s$ is represented as the complement of the unsafe set $\mathcal{Z}^s = (\Omega \cup \mathcal{O})^c$. Superscript c denotes the complement of the set.
Occluded Area: As the robot navigates in the environment, the occluded regions are identified using a LiDAR or depth camera sensor. The boundary of the occluded area can be used to perform reachability analysis for hidden occluded agents. Similarly forward reachability analysis can be performed for visible agents, in case that no prediction knowledge is assumed regarding their future actions.

Forward Reachable Sets: The potential locations of dynamic agents (visible or invisible) in the environment are captured using forward reachability analysis. In general, a forward reachable set Ω_N is defined as the set of states that are reachable after N time steps by a dynamical system $z_{k+1} = f(z_k, u_k)$ from a given set of initial states Ω_0 by applying an admissible sequence control input $u_k \in \mathcal{U}$, specifically $\Omega_N = \{z_N : \exists u_k \in \mathcal{U}\ \forall k = 0, \ldots, N-1, \quad \text{s.t. } z_{k+1} = f(z_k, u_k), z_0 \in \Omega_0\}$. It follows from this definition that[1]

$$\Omega'_k \subset \Omega_k \implies \Omega'_N \subset \Omega_N \text{ for all } N \geq k. \tag{1}$$

IV. Problem Formulation

Safe navigation in dynamic occluded environment can be formulated as an NMPC optimization problem that computes collision-free trajectories in real-time. The MPC optimization problem is formulated as follows

$$\begin{aligned}
\min_{\mathbf{u}_{t:t+N-1}} \quad & \sum_{k=0}^{N} l(\mathbf{z}_{k|t}, \mathbf{u}_{k|t}) && \text{(2a)}\\
\text{subject to} \quad & \mathbf{z}_{k+1|t} = f(\mathbf{z}_{k|t}, \mathbf{u}_{k|t}), && \text{(2b)}\\
& \mathbf{z}_{0|t} = \mathbf{z}(t), && \text{(2c)}\\
& \mathbf{z}_{k|t} \in \mathcal{Z},\ \mathbf{u}_{k|t} \in \mathcal{U}, && \text{(2d)}\\
& \|\mathbf{z}_{k|t} - \mathbf{z}_{k-1|t}\| g_t(\mathbf{z}_{k|t}) \leq 0, && \text{(2e)}\\
& \mathbf{z}_{N|t} = \mathbf{z}_{N-1|t}, && \text{(2f)}\\
& \text{and } k \in \{0, 1, 2, .., N\},
\end{aligned}$$

where $\mathbf{u}_{t:t+N-1} = [u_{0|t}, ..., u_{N-1|t}]$ denotes the sequence of control inputs over the MPC planning horizon N. The robot state and input variables $\mathbf{z}_{k|t}$ and $\mathbf{u}_{k|t}$ at step k are predicted at time t. The function $f(\cdot)$ in (2b) represents the nonlinear (dynamic or kinematic) model of the robot, which is discretized using Euler discretization. $\mathcal{Z}, \mathcal{U}$ are the state and input feasible sets, respectively. These sets represent state and actuator limitations. Constraint (2e) is the complementarity constraint in which $g_t(\mathbf{z}_{k|t}) \leq 0$ represents all the collision avoidance constraints. The time-varying safe set $\mathcal{Z}^s_{k|t}$ is defined as $\mathcal{Z}^s_{k|t} = \{g_t(\mathbf{z}_{k|t}) \leq 0,\ \forall \mathbf{z}_{k|t}\}$. The complementarity constraint (2e) formulation allows for $g_t(\mathbf{z}_{k|t})$ to be greater than 0 (the collision avoidance constraint is violated) as long as $\mathbf{z}_{k|t} = \mathbf{z}_{k-1|t}$ (the robot is stopped). As an example consider a scenario where a pedestrian hits a stopped car, the collision avoidance constraint is violated but since the car is stopped, we consider the plan safe. The idea is that if the pedestrian hits a stopped car, likely there will not be major injury, and the pedestrian bears the fault in the collision, not the car. Constraint (2f) is the terminal constraint that is required to guarantee recursive feasibility. Details are discussed in the next section. The collision avoidance constraint $g_t(\mathbf{z}_{k|t}) \leq 0$ includes static and dynamic constraint as follows:

$$\mathcal{C}(\mathbf{z}_{k|t}) \cap \mathcal{O}^b = \emptyset,\ b \in \mathcal{B}, \tag{3a}$$
$$\mathcal{C}(\mathbf{z}_{k|t}) \cap \Omega^r_{k|t} = \emptyset,\ r \in \mathcal{S}. \tag{3b}$$

Constraints (3a) are collision-avoidance constraints between the robot $\mathcal{C}$ (modeled as a circle) and the static environment denoted by set $\mathcal{B}$ indexed by b. Constraints (3b) represent the collision-avoidance constraints between the robot $\mathcal{C}$ and hidden and visible dynamic obstacles in the scene denoted by the set $\mathcal{S}$ indexed by r.

V. Recursive Feasibility Guarantees

Assumption 1: We assume the robot's sensor at each time t returns a set $\bar{\Omega}^r_{0|t}$ that contains agent r at time t. If r is not occluded, this set describes the region implied by the sensor measurement (e.g., a disk with some error radius). If r is occluded, this set is the occlusion region itself. If a sensor measurement is not taken at time t, $\bar{\Omega}^r_{0|t}$ is the entire admissible space for the agent.

Before we prove the main results of recursive feasibility, we state and prove a lemma concerning the evolution of the safe set for the robot.

Lemma 1: The predicted safe set at absolute time τ is non-decreasing as t increases, that is,

$$\mathcal{Z}^s_{(\tau-t_1)|t_1} \subset \mathcal{Z}^s_{(\tau-t_2)|t_2} \quad \forall t_1 \leq t_2 \leq \tau.$$

Proof: We only have to show that the forward reachable set of dynamic agents is non-increasing,

$$\Omega^r_{(\tau-t_2)|t_2} \subset \Omega^r_{(\tau-t_1)|t_1} \quad \forall t_1 \leq t_2 \leq \tau \quad \forall r \in \mathcal{S},$$

since these are the only dynamic components in $\mathcal{Z}^s$. Consider a forward reachable set $\Omega^r_{(\tau-t)|t}$ predicted for absolute time τ based on an initial set $\Omega^r_{0|t}$. Now consider a sensor measurement at time $t+1$ that localizes the agent to a set $\bar{\Omega}^r_{0|(t+1)}$ based on Assumption 1. We know that the agent lies both in the set $\bar{\Omega}^r_{0|(t+1)}$ from the sensor, and in the one-step reachable set $\Omega^r_{1|t}$ computed from the previous time step. Therefore, at time step $(t+1)$ the agent is in $\Omega^r_{0|(t+1)} = \bar{\Omega}^r_{0|(t+1)} \cap \Omega^r_{1|t} \subset \Omega^r_{1|t}$. By the nested property of reachable sets in (1) it follows that $\Omega^r_{(\tau-(t+1))|(t+1)} \subset \Omega^r_{(\tau-t)|t}$ for all $t \leq \tau$. Applying mathematical induction on t proves the desired result. ■

As an illustration of the concept formalized in Lemma 1, Fig. 2 depicts the forward reachable sets as circular shapes. The black circles represent the forward reachable sets at time step $t = 0$ and the red circles show the forward reachable sets at the next time step $t = 1$. By comparing the sets in absolute time $\tau = t + k$, the reachable set at prediction step $k = 1$ at execution time step $t = 1$ (shown in solid red) is always contained in the reachable set at prediction step $k = 2$ at execution time step $t = 0$ (shown in solid gray), so $\Omega^r_{\tau|t=1} \subset \Omega^r_{\tau|t=0}$. Therefore, the safe set $\mathcal{Z}^s_{\tau|t=0}$ at time $t = 0$ is always contained in the safe set at time step $t = 1$. At time $t = 0$, the safe set is $\mathcal{Z}^s_{\tau|t=0} = (\Omega^r_{\tau|t=0} \cup \mathcal{O}^b)^c$ and at

[1]Proof: Consider $z_k \in \Omega'_k$, a sequence of inputs $(u_k, \ldots, u_{N-1})$, and the resulting reachable point $z_N \in \Omega'_N$. Since $\Omega'_k \subset \Omega_k$, we know $z_k \in \Omega_k$. Since z_N is reachable through the same set of control inputs, $z_N \in \Omega_N$. This is true for any $z_N \in \Omega'_N$, so $\Omega'_N \subset \Omega_N$.

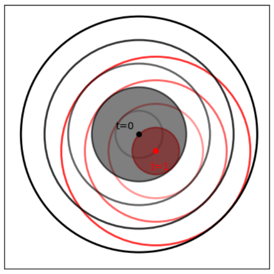

Fig. 2: Forward reachable sets of unsafe region Ω are shown for two consecutive time steps $t = 0$ and $t = 1$ in the simple case of a single pedestrian based on a top speed bound.

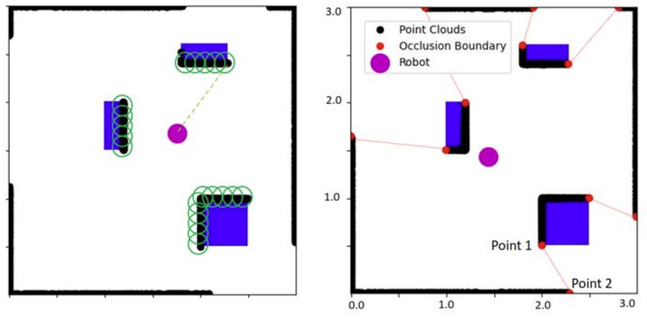

Fig. 3: **Left:** LiDAR-based collision avoidance: Blue areas are obstacles; Black dots are point clouds; Green circles are centered at down-sampled point-clouds. **Right:** Boundary of occlusion is detected based on a jump in range values of two consecutive points 1 and 2.

time $t = 1$, the safe set is $\mathcal{Z}^s_{\tau|t=1} = (\Omega^r_{\tau|t=0} \cap \Omega^r_{\tau|t=1} \cup \mathcal{O}^b)^c$. So $\mathcal{Z}^s$ is always non-decreasing.

We now state our main theoretical result.

Theorem 2: If a feasible solution exists for the NMPC optimization (2) at time $t = 0$, a feasible solution exists for all $t > 0$, that is, the proposed NMPC framework is recursively feasible.

Proof: Consider the NMPC problem (2) is solved at time $t = 0$. Assume feasibility of z_0 and let $[u^*_{0|0}, u^*_{1|0}, ..., u^*_{N-1|0}]$ be the optimal feasible control sequence computed at z_0 and $[z_0, z_{1|0}, ..., z_{N|0}]$ be the corresponding feasible state trajectory (which are assumed to exist at $t = 0$). Apply $u^*_{0|0}$ and let the system evolve to $z_1 = f(z_0, u^*_{0|0})$. At z_1 consider the control sequence $[u^*_{1|0}, u^*_{2|0}, ..., u^*_{N-1|0}, u^*_{N-1|0}]$ for the consecutive MPC problem at time $t = 1$. Since the time-varying state safe set $\mathcal{Z}^s$ is non-decreasing (according to Lemma 1) and the input constraints do not change, the state and input constraints are satisfied for all times except for potentially at state $z_{N|1}$. However, applying $u_{N-1|1} = u^*_{N-1|0}$ at the last step keeps the state at the previous state $z_{N|1} = z_{N-1|1}$. At time $t = 1$ in case the collision-avoidance constraint is violated at the last prediction time step $g(\mathbf{z}_{k=N|t=1}) > 0$, the fact that the robot is stopped $z_{N|1} = z_{N-1|1}$ ensures the robot will not violate the complementarity constraint at the last step, so the solution remains feasible. Applying this reasoning in a mathematical induction for all subsequent time steps yields the result stated in the theorem. ■

While the result applies generally to all problems that fall under our assumptions, in the rest of this paper we consider a specific setup in which the sensor is a LiDAR sensor, and the agents are assumed to be able to move in any direction with a known speed bound, leading to a simple geometric computation of reachable sets. We develop a practical algorithm based on (2) for this setup.

VI. Iterative Optimization Scheme

A. NMPC Optimization

The following optimization represents the nonlinear MPC scheme. The collision-avoidance with static obstacles in the environment $\mathcal{O}^b$ represented in (3a) are performed based on point-cloud data received from LiDAR. (Other sensors and methods can be used for collision detection and avoidance as well). Figure 3 (left) depicts LiDAR-based collision avoidance scheme, where black dots are point clouds and blue areas are obstacles. Point-clouds are down sampled and green circles are centered at the sampled point clouds. Circles' radius should be selected to ensure full coverage of the visible portion of obstacle. The safe distance between the robot (magenta circle) with each green circle are incorporated as constraints in the NMPC optimization

$$\begin{aligned} \min_{\mathbf{u}_{t:t+N-1}} \quad & (2a) && (4a)\\ \text{subject to} \quad & (2b), (2c), (2d), (3a),\ (2e),\ (2f) && (4b)\\ & \text{dist}(\mathcal{C}(\mathbf{z}_{k|t}), \bar{\mathbf{z}}_{\text{proj}k|t}) \geq d_{\text{safe}},\ r \in \mathcal{S}, && (4c)\\ & \text{and } k \in \{0, 1, 2, .., N\}, \end{aligned}$$

where $\bar{\mathbf{z}}^r_{\text{proj}k|t}$ denotes the projected state trajectory computed by solving collision avoidance optimization (The bar notation means the value is known). More details are included in the next section. The computed minimum distance is then constrained to be larger than a safe predefined minimum allowable distance d_{safe}. (d_{safe} is a design parameter and should be determined based on the uncertainty quantification of physical models and stochastic measurement errors.)

B. Collision Avoidance Optimization

Occlusion detection: To detect the boundary of occlusion, the consecutive range values obtained from LiDAR are compared and the ones with a difference larger than a threshold are detected as occlusion boundary. For example, in Figure 3 (right), there is a large jump between the range values of Point 1 and Point 2 (red points), so the LiDAR line-of-sight that passes through those two points indicates a boundary of occlusion. All the other red points in the figure are computed is the same way and their line-of-sight indicate an occlusion boundary.

Reachable set construction: The reachable set is where target agents (pedestrians) might present. The initial set for constructing the reachable set is a line segment which is the boundary of occlusion detected by LiDAR. To consider the worst-case, the pedestrian is assumed to start its movement

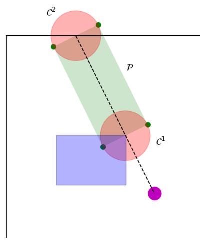

Fig. 4: The capsule set is constructed by polytope and circles.

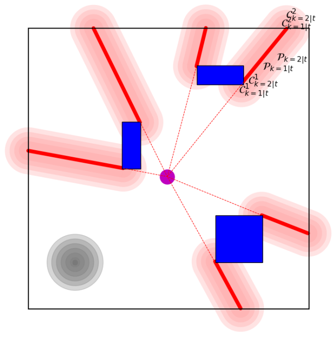

Fig. 5: Capsule sets are enlarged along the MPC prediction horizon based on the speed upper-bound v_{target} assumed for invisible agent. The gray circles illustrate forward reachable sets for visible target. The red dashed lines represent LiDAR rays.

Algorithm 1 Capsule Set Construction Algorithm

```
1: According to LiDAR scan of the environment, the jump in the values of two consecutive range measurements indicates the boundary of an occluded region.
2: The circles $\mathcal{C}^1$ and $\mathcal{C}^2$ with radius $d_{\text{target}}$ centered at the end points of line-of-sight are created and the vertices of the associated polytope are determined according to the circles' radius.
3: Convex hull of polytope vertices is obtained to determine H-representation of polytope, necessary for collision avoidance optimization.
```

from the occlusion boundary. No prior knowledge of the agents' dynamic model is assumed and the only assumption is that the target maximum speed v_{target} is known. This upper-bound on speed can be specified according to the environment such as driving or in-door settings.

Capsules are described as the union of two circles $\mathcal{C}^1$, $\mathcal{C}^2$ and a rectangle $\mathcal{P}$ as shown in Figure 4. Figure 5 shows the forward reachable sets that are nested capsules, which grow over time along the MPC horizon. The reachable sets are enlarged at each time step k along the MPC planing horizon. The enlarged capsule set is computed by increasing the radius of the two circles $\mathcal{C}^1$ and $\mathcal{C}^2$ by distance $d_{\text{target}} = \frac{v_{\text{target}}}{\Delta t}$, which is the maximum distance that a pedestrian can travel in a time step Δt. The corresponding polytopic set is constructed as shown in Figure 4 based on the convex hull of the 4 points (green dots) obtained from circle enlargement. The capsule set construction procedure is summarized in Algorithm 1.

Minimum distance from the capsule set: The robot shape is approximated as a circle with radius r_{robot} and the collision avoidance between the robot and the dynamic agent such as pedestrian is formulated based on the minimum distance between the circle (robot) and the capsules (the pedestrian forward reachable sets). To compute the minimum distance $\text{dist}(\mathcal{C}_{\text{robot}}, \Omega)$, first the robot's center of mass z is projected on to the capsule $d_{\text{proj}} = \min\{d_{\mathcal{C}^1}, d_{\mathcal{C}^2}, d_{\mathcal{P}}\}$, where $d_{\mathcal{C}^1}, d_{\mathcal{C}^2}, d_{\mathcal{P}}$ are distances to the circle $\mathcal{C}^1$ and circle $\mathcal{C}^2$ and the polytope $\mathcal{P}$, respectively. Then the robot radius is subtracted as $\text{dist}(\mathcal{C}_{\text{robot}}(\mathbf{z}), \Omega) = d_{\text{proj}} - r_{\text{robot}}$. Distance to the circles $d_{\mathcal{C}^1}$ and $d_{\mathcal{C}^2}$ are computed by calculating the distance to the center of circles and subtracting the radius of circle. Distance to the polytope $\mathcal{P}$ is computed by solving the optimization problem: $d_{\mathcal{P}} = \min_{\mathbf{y}}\{\|\bar{\mathbf{z}} - \mathbf{y}\|_2 | \mathbf{A}\mathbf{y} \leq \mathbf{b}\}$, where $\bar{\mathbf{z}}$ is the robot position, which is known (the bar notation means the value is kept as fixed). The optimal solution of the above optimization problem is denoted as $\mathbf{y}^*$. After computing d_{proj}, the corresponding projected point $\mathbf{z}_{\text{proj}}$ on the capsule set is computed (the projection of robot on the capsule set) by

$$\mathbf{z}_{\text{proj}} = \begin{cases} \mathbf{y}^* & \text{if } d_{\text{proj}} = d_{\mathcal{P}} \\ (1 - \text{ratio}_1)\bar{\mathbf{z}} + \text{ratio}_1 \mathcal{C}^1_{\text{center}} & \text{if } d_{\text{proj}} = d_{\mathcal{C}_1} \\ (1 - \text{ratio}_2)\bar{\mathbf{z}} + \text{ratio}_2 \mathcal{C}^2_{\text{center}} & \text{if } d_{\text{proj}} = d_{\mathcal{C}^2}, \end{cases} \quad (5)$$

where $\text{ratio}_1 = (d_{\mathcal{C}^1} - r_{\mathcal{C}^1})/d_{\mathcal{C}^1}$ and $\text{ratio}_2 = (d_{\mathcal{C}^2} - r_{\mathcal{C}^2})/d_{\mathcal{C}^2}$, and $r_{\mathcal{C}^1}$ and $r_{\mathcal{C}^2}$ are radius of the Circle 1 and Circle 2, respectively. In (5), the first equation is projection of robot on polytope and the second and third equations represent the projection of the robot on Circle 1 and Circle 2, respectively. The projection of the robot on the capsule set $\mathbf{z}_{\text{proj}}$ is computed by solving (5). This projection step is solved for all the robot predicted states along the MPC horizon $\bar{\mathbf{z}}_{k|t}$ $\forall k \in \{1, ..., N\}$ (open-loop state trajectory) and for each boundary of occlusion $r \in \mathcal{S}$, in parallel. The projected open-loop state trajectory $\bar{\mathbf{z}}^r_{\text{proj}k|t}$ is computed and is incorporated into the constraint (4c) as a fixed known value. To summarize, the optimization (4) and optimization (5) alternate. In the NMPC (4), the projected states $\bar{\mathbf{z}}^r_{\text{proj}k|t}$ are kept fixed and decision variables of optimization are the robot states along the MPC horizon $\mathbf{z}_{k|t}$. In collision avoidance optimization (5), the robot states are kept fixed $\bar{\mathbf{z}}_{k|t}$ and the projected states $\mathbf{z}_{\text{proj}k|t}$ are computed.
Also, to use open-loop projected states in the NMPC in constraint (4c), we need to rely on the projected states from the previous iteration $(t - 1)$ of NMPC solutions (open-loop state trajectory). So, before solving (5), the robot state trajectory $\bar{\mathbf{z}}_{k|t}$ is shifted one step forward in time and the last step is interpolated as $[\bar{\mathbf{z}}(2), ..., \bar{\mathbf{z}}(N), \bar{\mathbf{z}}(\text{Intp})]$.

Note that the optimization problem for projecting the robot on polytope and finding $d_{\mathcal{P}}$ has a lower-bound on minimum distance which is zero, so in case the point $\mathbf{z}$ is already inside the polytopic set $\mathcal{P}$ the minimum distance is calculated as zero. One alternative solution is to formulate the collision avoidance with capsule sets exactly the same as collision avoidance formulation for static obstacles, in which circles are sampled on the occlusion boundary and the robot distance to these circles are incorporated into the NMPC

Algorithm 2 Iterative Optimization Algorithm

```
1: Initialize state trajectory $[\mathbf{z}(1), ..., \mathbf{z}(N)]$
2: for $t = 0, 1, ..., \infty$ do
3:     Receive LiDAR scan and compute the capsule sets: $[\Omega^r(1), ..., \Omega^r(N)]$ for each $r \in \mathcal{S}$
4:     Compute the shifted state $[\bar{\mathbf{z}}(2), ..., \bar{\mathbf{z}}(N), \bar{\mathbf{z}}(\text{Intp})]$.
5:     Solve Problem (5) for the shifted state trajectory for each $r \in \mathcal{S}$ in parallel to find projected robot states on the capsule sets $[\mathbf{z}_{\text{proj}}(2), ..., \mathbf{z}_{\text{proj}}(N), \mathbf{z}_{\text{proj}}(\text{Intp})]$ for each prediction step of the open-loop trajectory.
6:     Solve NMPC Problem (4)
7:     Apply $\mathbf{u}_{\text{MPC}}$ to move forward.
8: end for
```

problem directly as collision avoidance constraint. Using this alternative approach no iterative optimization scheme is required. However, incorporating the constraints directly into the NMPC can slightly affect the computation time.

C. Iterative Optimization Algorithm

Problem (5) is itself an optimization problem, so imposing it as a constraint of Problem (2) yields a bilevel optimization which is computationally intensive. To overcome this challenge, by relying on the ability of MPC to generate predicted open-loop trajectories, we devise an alternative optimization algorithm in which we alternate between the two optimizations: NMPC optimization (4) and collision avoidance optimization (5), as detailed in Algorithm 2. In step ① of Algorithm 2, the sequence of state trajectory is initialized. In step ③, new LiDAR measurement is received and the reachable sets are computed accordingly. In step ④, the associated open-loop state trajectory is shifted forward one step in time and the last step is copied or interpolated. In step ⑤, the collision avoidance optimization is solved in parallel for each boundary of occluded area. In step ⑥ NMPC problem (4) is solved and a sequence of open-loop actions is computed. In step ⑦, the first control input $\mathbf{u}_{\text{MPC}}$ is applied and the procedure is repeated for the next time step.

VII. Simulations and Hardware Experiments

To validate the effectiveness of the proposed approach, both simulation studies and real-world experiments have been performed. The NMPC optimization is modeled using CasADi [12] in python. For the experiment, a TurtleBot robot equipped with Velodyne LiDAR (Vlp-16 Puck Lidar Sensor 360 Degree) is used. OptiTrack motion capture system is used for robot localization. The MPC is ran on a Lenovo ThinkPad laptop with ubuntu 20.04 OS with 3.00 GHz Intel CPU Core i7 with 32 GB of RAM.

The Robot dynamic is modeled by a nonlinear kinematic unicycle model $\dot{x} = v\cos(\psi)$, $\dot{y} = v\sin(\psi)$, $\dot{\psi} = \delta$, where the state vector is $\mathbf{z} = [x, y, \psi]^\top$ (x, y, and ψ are the longitudinal position, the lateral position, and the heading angle, respectively). The input vector is $\mathbf{u} = [v, \delta]^\top$ (v and δ are the velocity and the steering angle, respectively). Using Euler discretization, the unicycle model is discretized with sampling time Δt as $x(t+1) = x(t) + \Delta t\ v(t)\cos(\psi(t))$, $y(t+1) = y(t) + \Delta t\ v(t)\sin(\psi(t))$, $\psi(t+1) = \psi(t) + \Delta t\ \delta(t)$. The cost is defined as $l(\mathbf{z}, \mathbf{u}) = \sum_{k=t}^{t+N} \|(\mathbf{z}_{k|t} - \mathbf{z}_{\text{Goal}})\|^2_{Q_\mathbf{z}} + \sum_{k=t}^{t+N-1} (\|(\mathbf{u}_{k|t})\|^2_{Q_\mathbf{u}} + \|(\Delta\mathbf{u}_{k|t})\|^2_{Q_{\Delta\mathbf{u}}})$, where $\Delta\mathbf{u}$ penalizes changes in the input rate. $Q_\mathbf{z} \succeq 0$, $Q_\mathbf{u}, Q_{\Delta\mathbf{u}} \succ 0$ are weighting matrices, $\mathbf{z}_{\text{Goal}}$ is the goal location that robot should reach. Throughout the simulations the sampling time Δt is 0.1s. Simulation studies are performed for two different scenarios including navigating a corner and navigating a complex environment with multiple obstacles. In both scenarios the proposed Occlusion-Aware planner (O-A MPC) is compared against baseline MPC which is agnostic towards occlusions and possible presence of invisible agents.

Figure 6 illustrates corner navigation scenario. The closed-loop simulation for both planners (O-A MPC and baseline MPC) are shown. In this simulation, minimum allowable

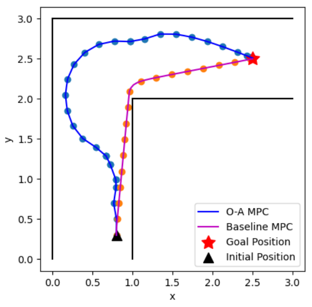

Fig. 6: Corner navigation simulation.

distance d_{safe} is $0.5m$, the velocity input is bounded within 0 to $2m/s$. The steering input is bounded within $\pm\pi rad$. The robot shape is defined by a circular shape with radius $r_{\text{robot}} = 0.2m$. Boundaries of the track shown are defined as the limits on the position states x and y. The width of the track is $1m$. The robot's initial condition $\mathbf{z}_0$ is $[x_0, y_0, \psi_0]$ = $[0.8m, 0.3m, \frac{\pi}{2}rad]$. Two goal positions $\mathbf{z_{goal_1}}$ as $[x = 1, y = 2.5]$ and $\mathbf{z_{goal_2}}$ as $[x = 2.5, y = 2.5]$ are specified. The MPC prediction horizon N is 10. A potential dynamic agent hidden behind the occluded area is assumed to be moving with speed $v_{\text{ped}} = 0.5m/s$. As seen, baseline MPC (red) is agnostic towards occlusion boundary and navigates the corner without considering collision with any possible potential invisible agent that might emerge from the occluded region. Conversely, O-A MPC planner (blue) slightly turns to the left as it approaches the corner to take a wider turn before turning completely to the right to avoid collision with any potential invisible agent.

Figure 7 (left figure) shows navigation in an environment with multiple obstacles which lead to various occluded regions. Closed-loop simulation for baseline MPC and O-A MPC with different speeds assumed for invisible agents are compared. As seen, the baseline MPC (blue) is the less conservative planner which is close to the obstacles, but the O-A MPC with $v_{\text{ped}} = 0.3m/s$ considers collision avoidance with possible invisible agent and keeps larger distance to the obstacles. Also, as the presumed speed upper-bound for invisible target agent increases to $v_{\text{ped}} = 0.5m/s$

Computation Time (Sec)			
Approach	Average	Max	Std
Baseline MPC	0.023	0.030	0.004
NMPC Optimization	0.033	0.054	0.006
CA Optimization	0.004	0.007	0.001
O-A MPC (Total)	0.037	0.061	0.007

TABLE I: Computation time statistics over 100 iterations is compared for baseline MPC and O-A MPC.

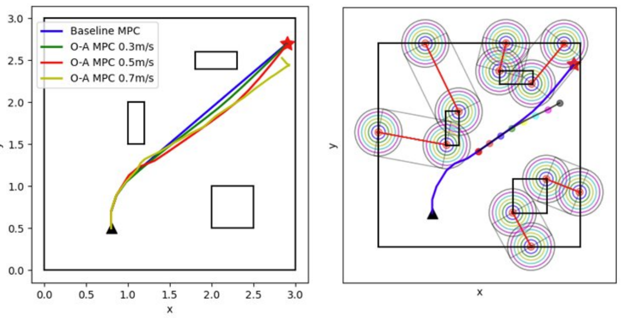

Fig. 7: **Left:** Navigation in environment with multiple obstacles. The plots compare robot executed trajectories for various speed upper-bounds assumed for the target agent. **Right:** Open-loop MPC trajectory is shown for time step $t = 7s$. Capsule sets corresponding to each predicted step of MPC are shown in the same color.

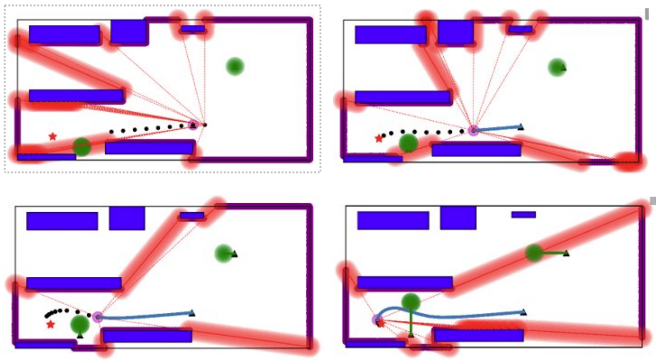

Fig. 8: **Top Left:** One of the pedestrians (green circles in the top right) is moving to the left and is visible to the robot. The other pedestrian (green circle in the bottom left region) is intiallly in the occluded region and she/he starts to move towards the top. **Top Right:** At the moment that pedestrian crosses occlusion boundary, she/he is visible to the robot and the forward reachable sets (green) are computed to avoid collision with visible target. **Bottom Left:** The robot moves towards its goal while avoiding collision with visible/hidden agents, (open-loop trajectories are shown in black). **Bottom Right:** Finally, the robot reaches its goal and its executed trajectory is illustrated (blue).

and $v_{\text{ped}} = 0.7m/s$ values (shown in red and light green, respectively), the robot's behavior is more conservative and it keeps larger distance towards the occluded regions, as expected. Figure 7 (right figure) shows the enlargement of capsule sets along the MPC horizon. The blue trajectory shows the closed-loop plan from the initial condition $\mathbf{z}_0 = [x_0, y_0, \psi_0] = [0.8m, 0.3m, \frac{\pi}{2}rad]$ to the goal position $\mathbf{z}_{\mathbf{goal}}$ as $[x = 2.9, y = 2.8]$. The open-loop trajectory at time step $t = 7s$ is shown as colored dots, where each color represents one step of prediction. For example red dot shows $\mathbf{z}_{k=0|t=7}$, light red is $\mathbf{z}_{k=1|t=7}$, purple is $\mathbf{z}_{k=2|t=7}$, green is $\mathbf{z}_{k=3|t=7}$, yellow is $\mathbf{z}_{k=4|t=7}$, light blue is $\mathbf{z}_{k=5|t=7}$, pink is $\mathbf{z}_{k=6|t=7}$ and so on (only portion of horizon steps are illustrated in the figure for clarity). The corresponding capsule sets are shown in the same color. As seen, at each horizon step k, the robot satisfies minimum distance requirement to capsule sets. Computation times for baseline MPC and O-A MPC for the scenario of Figure 7 are reported in Table I for over 100 iterations. As seen, the average of computation time for O-A MPC is larger than baseline MPC, due to extra constraints associated with reachable sets on occluded boundaries, but O-A MPC is still suitable for real-time implementation.

Figure 8 shows snapshots of closed-loop simulation of LiDAR and O-A MPC for a scenario in which a robot (magenta circle) navigates in an environment with multiple static (blue regions) and dynamic obstacles or pedestrians (green circles). Top left figure shows the initial condition in which one pedestrian is visible to the robot and the other is initially hidden in occluded region. Point clouds from LiDAR simulation are shown in purple. Boundary of occlusion and capsule reachable sets are shown in red. As the robot navigates the environment to reach its goal (red star), the occlusion boundary changes as the LiDAR scan is updated. As soon as the pedestrian crosses the boundary of occlusion (top right figure), the O-A planner creates the pedestrian's reachable sets (nested green circles) and avoid collision with visible agents (green reachable sets) and capsules (red reachable sets) corresponding to occluded area. The bottom left figure shows the open-loop trajectory (black dots) as the robot move towards its goal. The bottom right figure shows the entire executed trajectory in which the robot reaches its goal while avoiding collision with hidden/visible pedestrians. Note that visible targets can create dynamic occlusion in the environment. This type of occlusion can be handled in a similar way. However, in this simulation the occlusion caused by moving visible pedestrian is considered negligible due to the small size of pedestrian.

Figure 9, shows experimental results that compares baseline MPC with O-A MPC for robot navigation in different environments, a corner, square-shaped obstacle and triangular-shaped obstacle. For all these experiments, the environment is unknown to the robot, so it relies on its LiDAR to perceive the environment in real-time. LiDAR point cloud data (purple) define the environment boundaries and obstacles. However, they act differently on how they navigate the occluded regions. Blue dots represent the executed plan starting from black small triangle and reaching the goal position (red star). Gray arrows show the heading of robot at each time step. Both MPC planners are operating in 10 Hz with horizon $N = 10$. The left column shows corner scenario in which O-A MPC takes a wider turn to navigate the corner to avoid collision with possible invisible agents, but baseline MPC which is agnostic towards occlusion regions navigates the corner without considering collision avoidance with occluded dynamic agents. The middle and right columns confirm the same results for navigation in environments with square-shaped and triangular-shaped obstacles.

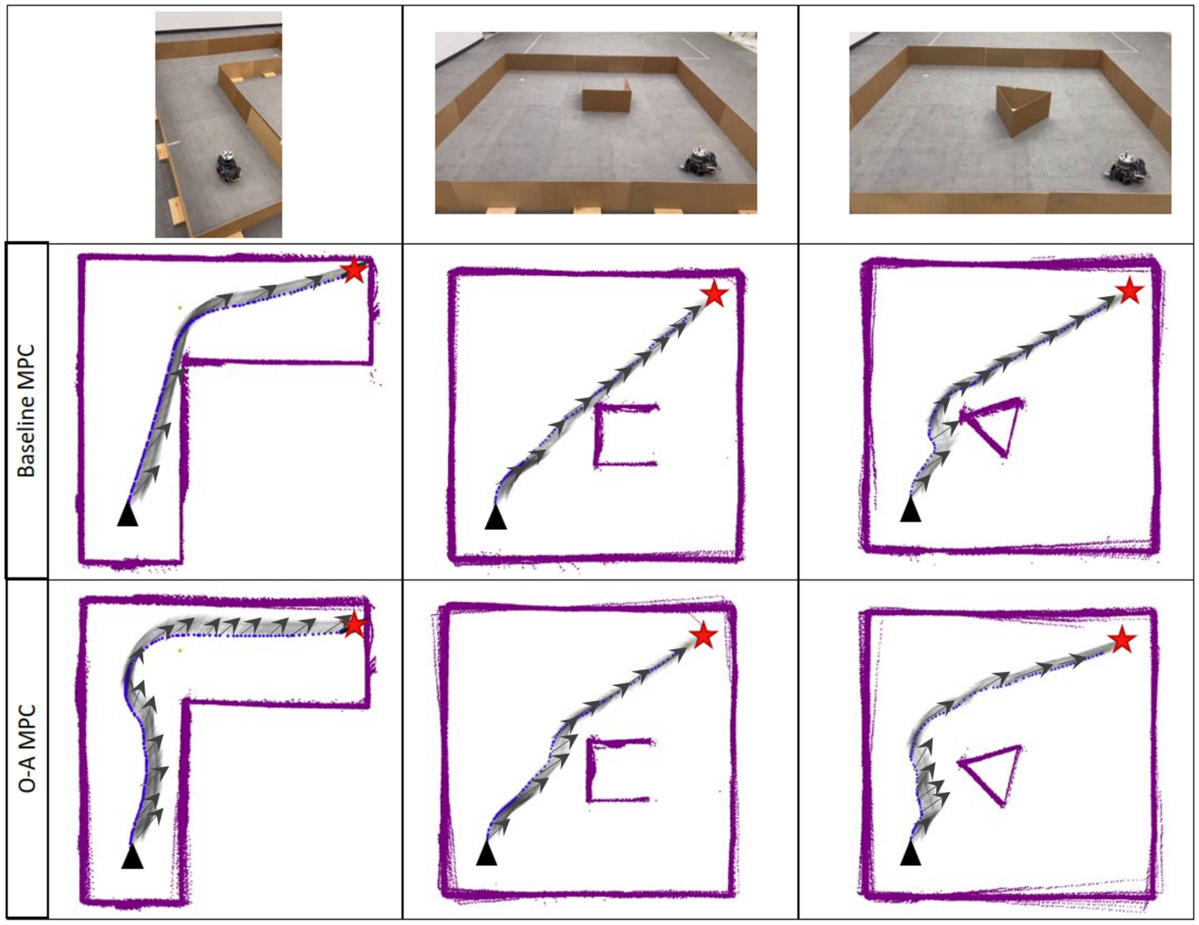

Fig. 9: Experimental Results: Gray arrows show the robot's heading at each time step; Blue dots show the executed plan; LiDAR point clouds are shown in purple. **Top Row**: Different environments (corner, square obstacle, triangle obstacle (from left to right). **Second Row**: Baseline MPC is occlusion-agnostic and navigates the occluded regions by taking tight turns without considering collision avoidance with potential invisible dynamic agents. **Third Row**: Occlusion-Aware MPC planner takes wide turns in occluded regions.

VIII. Conclusion and Future Work

This work proposed a novel perception-aware real-time planning framework for safe navigation of robotics systems in an a priori unknown dynamic environment where occlusions exist. The presented NMPC strategies provide safety guarantees and computational efficiency. For future work, the proposed algorithm can be extended to be used for safe navigation in 3-dimensional spaces for aerial robot applications.

References

[1] Jerry Ding, Eugene Li, Haomiao Huang, and Claire Tomlin. Reachability-based synthesis of feedback policies for motion planning under bounded disturbances. pages 2160–2165, 05 2011.

[2] Sylvia L. Herbert, Mo Chen, SooJean Han, Somil Bansal, Jaime F. Fisac, and Claire J. Tomlin. Fastrack: A modular framework for fast and guaranteed safe motion planning. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1517–1522, 2017.

[3] David Fridovich-Keil, Sylvia L. Herbert, Jaime F. Fisac, Sampada Deglurkar, and Claire J. Tomlin. Planning, fast and slow: A framework for adaptive real-time safe trajectory planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 387–394, 2018.

[4] Zixu Zhang and Jaime F Fisac. Safe Occlusion-Aware Autonomous Driving via Game-Theoretic Active Perception. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[5] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. Pampc: Perception-aware model predictive control for quadrotors. 04 2018.

[6] Keuntaek Lee, Jason Gibson, and Evangelos A. Theodorou. Aggressive perception-aware navigation using deep optical flow dynamics and pixelmpc, 2020.

[7] Jacob Higgins and Nicola Bezzo. Negotiating visibility for safe autonomous navigation in occluding and uncertain environments. *IEEE Robotics and Automation Letters*, 6(3):4409–4416, 2021.

[8] Stefan Hoermann, Felix Kunz, Dominik Nuss, Stephan Renter, and Klaus Dietmayer. Entering crossroads with blind corners. a safe strategy for autonomous vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 727–732, 2017.

[9] Minchul Lee, Kichun Jo, and Myoungho Sunwoo. Collision risk assessment for possible collision vehicle in occluded area based on precise map. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2017.

[10] Hans Andersen, Wilko Schwarting, Felix Naser, You Hong Eng, Marcelo H. Ang, Daniela Rus, and Javier Alonso-Mora. Trajectory optimization for autonomous overtaking with visibility maximization. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2017.

[11] Yannik Nager, Andrea Censi, and Emilio Frazzoli. What lies in the shadows? safe and computation-aware motion planning for autonomous vehicles using intent-aware dynamic shadow regions. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5800–5806, 2019.

[12] Joel Andersson, Joris Gillis, and Moritz Diehl. User documentation for casadi v3.2 - sourceforge, Sep 2017.