Abstract—Consider a robot operating in an uncertain environment with stochastic, dynamic obstacles. Despite the clear benefits for trajectory optimization, it is often hard to keep track of each obstacle at every time step due to sensing and hardware limitations. We introduce the Safety motion planner, a receding-horizon control framework, that simultaneously synthesizes both a trajectory for the robot to follow as well as a sensor selection strategy that prescribes trajectory-relevant obstacles to measure at each time step while respecting the sensing constraints of the robot. We perform the motion planning using sequential quadratic programming, and prescribe obstacles to sense based on a novel connection between the duality information associated with the convex subproblems and the effect of uncertainty reduction through Kalman filter updates. We guarantee safety by ensuring that the probability of the robot colliding with any of the obstacles is below a prescribed threshold at every time step of the planned robot trajectory. We demonstrate the efficacy of the Safety motion planner through software and hardware experiments.

Index Terms—Collision avoidance, motion and path planning, optimization and optimal control.

I. INTRODUCTION

We study the motion planning problem of a robot that operates in an environment populated by dynamic obstacles and has limited sensing capabilities. Despite these limitations, the robot seeks to remain safe by avoiding collisions with these obstacles. Several relevant applications of such a problem include food delivery robots that must reach their destination while avoiding pedestrians [1], self-driving cars that must generate real-time motion plans while remaining cognizant of other road vehicles in addition to pedestrians [2], and industrial robots operating in the vicinity of humans [3]. For the purpose of motion planning, a popular method to predict the motion of the dynamic, uncontrolled obstacles present in the environment is to use stochastic, dynamical models. Such models may be obtained from a combination of data-driven and model-driven approaches [4], [5], [6]. Without real-time measurements, the future obstacle state predictions obtained from these models typically have high variance, requiring overly conservative robot motion plans in order to satisfy the safety requirements. However, reducing uncertainty through measurements of the underlying state of each obstacle at every time step is often impractical due to sensing and hardware limitations. In the case of a food delivery robot, maintaining awareness of all pedestrians at all times can drive up hardware costs, perception-related computational costs, and planning time for the robot, resulting in poor reaction time and performance losses. To strike a balance between these extremes, we propose a motion planning algorithm that simultaneously designs a safe trajectory for the robot to complete its task and allocates its limited available sensors to reduce the uncertainty of the obstacles most relevant to the trajectory.

We measure the relevance of an obstacle in terms of how the performance of the robot is affected by the uncertainty in the obstacle localization. In Fig. 1(a), the task of the drone \( R \), which we can control, is to reach its target state at the yellow star as quickly as possible. The shaded circles represent areas in the environment where an obstacle resides with high probability, as determined by the stochastic model of the obstacle and past measurements. By making a measurement about the underlying state of obstacle \( O_1 \), and thereby reducing the uncertainty in its position, the robot can further reduce the time required to reach the target state, as shown in Fig. 1(b). Clearly, measuring the underlying states of obstacles \( O_2 \) and \( O_3 \) will have no effect on the performance of the drone, and the limited sensing resources available should not be spent on reducing the uncertainty associated with these obstacles.

Our main contribution is a novel receding horizon control-based stochastic motion planner, which we name Safety, that synthesizes a safe trajectory for the robot in a stochastic environment while identifying the obstacles for which sensing resources should be allocated. This allocation is based on the degree to which an obstacle influences the performance of the robot’s future motion plan. We ensure probabilistic safety in
Existing literature has previously studied (Safely ∥·∥ ∈ to denote
[8] be the
N ≜ I
0
[9] II
O
enables a significant improvement in performance.
Reducing the state estimates
be an
[12]
R+1
Safely
[12]
[7]
R
has considered nonlinear robot motion planning in
A
0
likewise considers the relation between sens-
[10]
[17]
 Studied the problem of minimizing the directed
information from the plant output to the control input. However, Tanaka et al. [17] did not consider this problem in the context of obstacle avoidance as we do.
More generally, our work is reminiscent of active perception [18], wherein an agent behaves in such a way as to gain information by controlling what it senses. Recently, researchers have also looked into deep learning to design perceptual control policies that account for attention [19], but enforce safety only by handing over the control to an expert or stopping before the obstacle. Such safety strategies are not feasible for autonomous robots that operate independently of an expert.
The rest of this article is organized as follows. In Section II, we begin by formulating the stochastic motion planning problem of the robot and how it can utilize the measurement of an obstacle state to improve its performance. In Section III, we propose the Safely motion planner using sequential quadratic programming and probabilistic occupancy, and motivate the use of the dual variables in the sensor selection strategy using sensitivity analysis. In Section IV, we provide software and hardware experiments demonstrating the benefit of incorporating the Safely motion planner for obstacle avoidance. Finally, Section V concludes this article.

II. PRELIMINARIES AND PROBLEM FORMULATION

We denote the set of real numbers by \( \mathbb{R} \), the set of \( n \)-dimensional real numbers by \( \mathbb{R}^n \), and the set of all positive real numbers by \( \mathbb{R}^+_n \). In addition, we denote the set of natural numbers by \( \mathbb{N} \), and we use \( \mathbb{N}_{[a,b]} \) to denote a sequence of natural numbers \([a, a+1, \ldots, b]\). Finally, we denote the set of all positive semidefinite matrices of size \( n \times n \) by \( S_{+}^{n} \). Furthermore, let \( O_{m \times n} \) be an \( m \times n \) matrix of zeros and let \( I_n \) be the \( n \)-dimensional identity matrix. For two sets \( A \) and \( B \), we use the operator \( \oplus \) to denote the Minkowski sum \( A \oplus B = \{ a + b | a \in A, b \in B \} \). Throughout the article, we will use \( \| \cdot \| \) to denote the Euclidean norm, and we denote the standard Euclidean ball by \( \text{Ball}(\mu, r) \stackrel{\Delta}{=} \{ x \in \mathbb{R}^n : \| x - \mu \| \leq r \} \).

A. Duality in Optimization

Consider the following general form of a nonlinear optimization problem:

\[
\begin{align*}
\min & \quad c(y) \\
\text{s.t.} & \quad h(y) = 0_{j \times 1} \\
& \quad g(y) \leq 0_{i \times 1}
\end{align*}
\]
where \( y \in \mathbb{R}^l \) is the decision variable, \( c : \mathbb{R}^l \to \mathbb{R} \) is the objective function, \( h : \mathbb{R}^l \to \mathbb{R}^j \) defines the set of \( j \) equality constraints, and \( g : \mathbb{R}^l \to \mathbb{R}^k \) defines the set of \( k \) inequality constraints. Here, we assume that \( c, h, \) and \( g \) are all twice continuously differentiable. The Lagrangian of (1) is formed by transforming it into an unconstrained optimization problem by adding the weighted sum of the constraints to \( c(y) \), which we formally express as
\[
L(y, \lambda, \gamma) = c(y) + \lambda^\top h(y) + \gamma^\top g(y)
\]
where \( \lambda \in \mathbb{R}^j \) and \( \gamma \in \mathbb{R}^k \) are the Lagrange multiplier vectors, also known as the dual variables [20]. From the Lagrangian, one can then form the Lagrange dual problem, given by
\[
\min \quad \hat{c}(\lambda, \gamma)
\]
\[\text{s.t.} \quad \gamma \geq 0_{k+1}
\]
where \( \hat{c}(\lambda, \gamma) \triangleq \inf_y L(y, \lambda, \gamma) \). If (1) contains a strictly feasible point (i.e., Slater’s condition holds) and is convex, then it holds that the optimal solution of (1) is equal to the optimal solution of (3), a property known as strong duality. Finally, note that off-the-shelf convex optimization solvers typically solve (3) in the process of solving (1), thus providing access to the dual variables at no additional computational cost [21].

B. Robot and Environment Model

Consider a robot operating in an environment where it has the discrete-time deterministic dynamics
\[
x[t+1] = f(x[t], u[t])
\]
where \( x[t] \in \mathbb{R}^n \) and \( u[t] \in \mathcal{U} \subset \mathbb{R}^m \) are the state and the input of the robot at time step \( t \), respectively, and \( f : \mathbb{R}^{n+m} \to \mathbb{R}^n \) is a known nonlinear, differentiable function describing the robot’s dynamics. The goal of the robot is to reach within a distance \( \epsilon > 0 \) to a specified target state \( x_g \in \mathbb{R}^n \). In addition, the robot must not collide with any of the \( N_O \in \mathbb{N} \) dynamic obstacles present in the environment, where the \( o \)th obstacle has the known, discrete-time, stochastic, uncontrolled, linear dynamics
\[
x_o[t+1] = A_o x_o[t] + B_o w_o[t]
\]
where \( x_o[t] \in \mathbb{R}^n \) is the obstacle’s state, \( w_o[t] \sim \mathcal{N}(\mu_{w_o}, \Sigma_{w_o}) \) is a Gaussian disturbance with mean \( \mu_{w_o} \in \mathbb{R}^n \) and covariance \( \Sigma_{w_o} \in \mathbb{S}_+^n \) that encodes our known prior information about the motion of the \( o \)th obstacle, and the matrices \( A_o \in \mathbb{R}^{n \times n} \) and \( B_o \in \mathbb{R}^{n \times p} \), respectively. Furthermore, we assume the robot has access to a stochastic prior belief over the initial state of the \( o \)th obstacle, \( x_o[0] \sim \mathcal{N}(\mu_o[0], \Sigma_o[0]) \), where \( \mu_o \in \mathbb{R}^n \) is the mean state estimate and \( \Sigma_o[0] \in \mathbb{S}_+^n \) is the state estimate covariance matrix. Such a model for the obstacle dynamics is common in Kalman filter literature [4].

We assume that if the robot allocates sensor resources to observe obstacle \( o \) at time \( t \), then the robot makes an observation according to the linear sensor model
\[
z_o[t] = H x_o[t] + \nu[k]
\]
where \( z_o[t] \in \mathbb{R}^q \) is the observation made by the robot, \( H \in \mathbb{R}^{q \times n} \), and \( \nu \sim \mathcal{N}(\mu_{\nu}, \Sigma_{\nu}) \) is Gaussian measurement noise with mean \( \mu_{\nu} \in \mathbb{R}^q \) and covariance \( \Sigma_{\nu} \in \mathbb{S}_+^q \). In this article, we assume \( \mu_{\nu} = 0_{q \times 1} \). Given this observation, the robot updates its estimate of the obstacle’s true position \( x_o[t+1] \), characterized by the mean \( \hat{x}_o[t+1] \) and the covariance \( \Sigma_o[t+1] \), according to the standard Kalman filter update [4]
\[
x_o[t+1] = A_o \hat{x}_o[t] + B_o \mu_{w_o}
\]
\[
\Sigma_o[t+1] = A_o \Sigma_o[t] A_o^\top + B_o \Sigma_{w_o} B_o^\top
\]
\[
K_o[t] = \Sigma_o[t+1] H^\top (H \Sigma_o[t+1] H^\top + \Sigma_{\nu})^{-1}
\]
\[
\hat{x}_o[t+1] = \hat{x}_o[t+1] + K_o[t](z_o[t+1] - H \hat{x}_o[t+1])
\]
\[
\Sigma_o[t+1] = (I_n - K_o[t] H) \Sigma_o[t+1]
\]
wherein the present mean and covariance estimate are first propagated through the dynamics (5) in (7a)–(7e) is the Kalman gain, and (7d) and (7e) is the updated estimate after making the observation \( z_o[t+1] \). Note that in the absence of observations, we can recursively express \( \hat{x}_o[t] \), the obstacle state at a future time instant \( t \in \mathbb{N} \), as
\[
x_o[t] \sim \mathcal{N}(\hat{x}_o[t], \Sigma_o[t])
\]
\[
x_o[t] = A_o \hat{x}_o[t-1] + B_o \mu_{w_o}
\]
\[
\Sigma_o[t] = A_o \Sigma_o[t-1] A_o^\top + B_o \Sigma_{w_o} B_o^\top
\]
which again follows from the Kalman filter theory. For ease of notation, we let \( \mu_o[t] \triangleq \hat{x}_o[t] \) and we denote the configuration of the obstacle set at a time step \( t \in \mathbb{N} \) using the concatenated obstacle state vector \( X_{obs}[t] \triangleq [x_1[0], . . . , x_{N_O}[t]]^\top \in \mathbb{R}^{(N_O n)} \). For this concatenated obstacle state, we then define its associated probability measure as \( \mathbb{P}^{X_{obs}[t]} \), where \( X_{obs}[0] \) is the known initial obstacle state vector, and the obstacle state and uncertainty propagate according to (8). Specifically, \( \mathbb{P}^{X_{obs}[t]} \) denotes the probability that the concatenated obstacle state at time \( t \) is \( X_{obs} \), given that the initial concatenated obstacle state is \( X_{obs}[0] \).

C. Optimal Control Problem With Probabilistic Safety

We now discuss the optimal control problem of the robot. At each time step, the robot plans a motion trajectory for the next \( T \) time steps, where \( T \in \mathbb{N} \) is the planning horizon. This planning horizon should be chosen in order to balance the improved performance obtained by planning nonmyopically with the computation time required to do so. We start with the following simplifying assumption.

**Assumption 1:** The shapes of the robot and each obstacle are approximated by symmetric covers of their rigid bodies.

Note that by assuming symmetric covers for their rigid bodies, the following results hold for a robot with rotational motion in addition to translational motion. Let the sets \( \mathcal{R}_{\text{robot}} \) and \( \mathcal{R}_o \) for every \( o \in \mathbb{N}[1, N_O] \) denote the rigid bodies of the robot and the obstacles, respectively. Under Assumption 1, for a given robot trajectory \( \{x[t]\}_{t=0}^{T} \triangleq \{x[0], x[1], . . . , x[T]\} \), the probability of collision with any of the \( N_O \) obstacles over the course of the entire trajectory of the robot is given by the function CollidePr :
\( \mathbb{R}^{(nT)} \to [0, 1] \), where

\[
\text{CollidePr}(x[0], \ldots, x[T]; X_{\text{obs}}[0]) = P_{X_{\text{obs}}[0]} \left\{ \bigcup_{o=1}^{N_O} \bigcup_{t=1}^{T} \{ x_o[t] \in x[t] \oplus B_o \} \right\}
\]

with sets \( B_o \triangleq R_{\text{robot}} \oplus (-R_o) \subset \mathbb{R}^n \) for all \( o \in \mathbb{N}_{[1,N_O]} \), where \( -R_o \triangleq \{-x : x \in R_o\} \). Intuitively, the probability of collision is the probability that, at any time step over the planning horizon of the robot, the rigid body of the robot and the rigid body of any obstacle overlap.

For some performance metric \( J : \mathbb{R}^{(n+m)T} \to \mathbb{R} \) and a user-defined safe operation zone \( S \subseteq \mathbb{R}^n \), we aim to solve the following optimal control problem for the robot:

\[
\begin{align*}
\min_{x[1], \ldots, x[T], u[0], \ldots, u[T-1]} & \quad J(x[1], \ldots, x[T], u[0], \ldots, u[T-1]) \\
\text{s.t.} & \quad u[\cdot] \in \mathcal{U}, x[\cdot] \in S \\
\text{Dynamics} & \quad (4) \text{ and } (5) \\
\text{CollidePr}(x[0], \ldots, x[T]; X_{\text{obs}}[0]) & \leq \alpha
\end{align*}
\]

with decision variables \( x[1], \ldots, x[T] \) and \( u[0], \ldots, u[T-1] \). Here, \( \alpha \in [0, 1] \) is a (small) user-specified upper bound on the acceptable collision probability. The cost function \( J \) is assumed to be twice continuously differentiable and can represent a variety of performance metrics, such as quickly reaching the target state.

**Assumption 2:** For all \( o \in \mathbb{N}_{[1,N_O]} \), \( B_o = \text{Ball}(c_o, r_o) \), for some center \( c_o \in \mathbb{R}^n \) and some radius \( r_o > 0 \).

Assumption 2 is not overly restrictive, since the rigid bodies of the robot and the obstacles typically have finite volume, which result in bounded \( B_o \). Furthermore, we consider a covering hypersphere \( B_o \supseteq R_{\text{robot}} \oplus (-R_o) \) when \( R_{\text{robot}} \oplus (-R_o) \) is not a hypersphere to satisfy Assumption 2, which tightens the probabilistic safety constraint (10d). In order to incorporate future obstacle perception, we iteratively solve (10) through a receding horizon control-based approach [22]. Specifically, at each time step, the robot updates the position estimate of the sensed obstacle, sets up the collision-avoidance constraints using the new estimate, and solves (10) for a new motion plan.

**III. Safely Motion Planner**

In this section, we introduce the Safely motion planner to address Problem 1. We first formulate the safe, observation-free motion planning problem of the robot. We then analyze the utility of observing a particular obstacle through the lens of duality and sensitivity analysis, and subsequently use this analysis to propose a constrained sensor selection strategy. We conclude with a brief discussion of the numerical implementation of Safely.

**A. Nonlinear Stochastic Motion Planning**

In this section, we begin by discussing ellipsoidal outer-approximations of the keep-out sets, whose avoidance guarantees the desired probabilistic safety in (10d). These outer-approximations follow from probabilistic occupancy functions [7], and generalize sufficient conditions discussed in [8] for a conservative enforcement of joint chance constraints (10d). We formalize these connections in Proposition 1.

**Proposition 1:** A sufficient condition to satisfy (10d) is the following collection of constraints:

\[
(x[t] - \mu_o[t])^\top (Q_o^+[t])^{-1} (x[t] - \mu_o[t]) \geq 1
\]

for each obstacle \( o \in \mathbb{N}_{[1,N_O]} \) and each time step \( t \in \mathbb{N}_{[1,T]} \). Here, \( Q_o^+[t] \in \mathbb{R}^{n \times n} \) are positive definite matrices defined by the stochastic dynamics of the corresponding obstacle and a user-specified unit vector \( l_o \in \mathbb{R}^n \) given by

\[
\begin{align}
Q_o^+[t] &= \left( \sqrt{\frac{l_o^\top Q_o[t] l_o + r_o}{\sqrt{l_o^\top Q_o[t] l_o}}} + r_o I_n \right) \\
Q_o[t] &= -2 \log \left( \frac{\alpha}{T N_O \text{Volume(Ball}(c_o, r_o))} \right) \Sigma_o[t].
\end{align}
\]

**Proof:** Using Boole’s inequality [23], we have

\[
\begin{align*}
\mathbb{P}_{X_{\text{obs}}[0]} \left\{ \bigcup_{o=1}^{N_O} \bigcup_{t=1}^{T} \{ x_o[t] \in x[t] \oplus B_o \} \right\} & \leq \sum_{o=1}^{N_O} \sum_{t=1}^{T} \mathbb{P}_{X_{\text{obs}}[0]} \left\{ x_o[t] \in x[t] \oplus B_o \right\} \\
& \leq \frac{\alpha}{T N_O}
\end{align*}
\]

for every \( o \in \mathbb{N}_{[1,N_O]} \) and \( t \in \mathbb{N}_{[1,T]} \) as the sum on the right-hand side of (13) is then upper bounded by \( \alpha \). For the Gaussian obstacles we consider, it was shown in [8, Sec. 3.3] that a conservative approximation to this keep-out set for a given obstacle \( o \) and time step \( t \) can be constructed according to \( \mathcal{E}(\mu_o[t], Q_o[t]) + B_o \), in which the first term denotes an ellipse with mean state \( \mu_o[t] \) and shape \( Q_o[t] \), where \( Q_o[t] \) is given by (12b). Recalling Assumption 2, we note that there does not generally exist a closed-form expression for the Minkowski sum of two ellipsoids. However, we can compute an ellipsoidal outer-approximation of this sum that is tight in a given direction of interest \( l_o \), expressed as \( \mathcal{E}(\mu_o[t], Q_o^+[t]) \), where \( Q_o^+[t] \) is given
by (12a) [24]. Since each ellipsoid defines a conservative approximation of the original avoid-set for each obstacle \( o \) and time \( t \), it follows that the satisfaction of the collection of constraints (11) is sufficient to guarantee (10d) under Assumption 2. □

Remark 1: The use of Boole’s inequality in the preceding proof may introduce conservatism in bounding the collision probability and in the robot’s motion plan. In cases where such conservatism is prohibitive, alternative methods for bounding the collision probability can be considered, as discussed in [25].

Using Proposition 1, we now obtain a tightened reformulation of (10) by replacing the constraint (10d) with the collection of ellipsoidal keep-out constraints (11). Note every feasible solution of this modified problem is feasible for (10) by the result of Proposition 1. Furthermore, it is straightforward to express this modified optimization problem in the general form of a nonlinear program given in (1), for which numerous methods exist to solve it to local optimality. We focus our attention on sequential quadratic programming (SQP)-based approaches [9], wherein a sequence of quadratic programming (QP) subproblems are iteratively solved until convergence to a local optimum is obtained. At each iterate, these subproblems are constructed using a quadratic approximation of the objective function subject to the original constraints linearized about the previous solution iterate. Let \( y^k \) denote the optimal solution of the QP at the \( k \)th iteration. Then, \( y^{k+1} \) is obtained by solving the following QP:

\[
\min \quad \nabla c(y^k)^T (y^{k+1} - y^k) + (y^{k+1} - y^k)^T M^k (y^{k+1} - y^k)
\]

s.t. \( h(y^k) + \nabla h(y^k)^T (y^{k+1} - y^k) = 0_{j \times 1} \)

\( g(y^k) + \nabla g(y^k)^T (y^{k+1} - y^k) \leq 0_{l \times 1} \) (15)

where the decision variable is \( y^{k+1} \) and \( M^k \in \mathbb{R}^{ixi} \) is a positive definite approximation to the Hessian of the Lagrangian \( \nabla^2 L(y^k, \lambda, \gamma) \), in which \( L(y^k, \lambda, \gamma) \) is given by (2). Note that since \( M^k \) is positive definite and all constraints are affine in \( y^{k+1} \), (15) is a convex QP and can be solved efficiently using off-the-shelf solvers such as OSQP [26].

Define the concatenated vectors \( \mathbf{x} = [x[1]^T, \ldots, x[T]^T]^T \) and \( u = [u[0]^T, \ldots, u[T - 1]^T]^T \). Then, applying the optimization problem given in (15) to the optimization problem solved by the Safety motion planner given in (10) with the ellipsoidal keep-out constraints (11), the QP given in (16) shown at the bottom of this page, is solved at every iteration of the SQP procedure, in which (16b) and (16c) are the linearizations of the ellipsoidal keep-out constraints (11) and the robot dynamics (4), respectively, about the previous solution iterate of the SQP procedure. The values of the constants \( C_{obs}^k[t], d_{obs}^k[t], C_{dyn}^k[t], \) and \( d_{dyn}^k[t] \) are provided in the Appendix.

Note that (16d) and (16e) propagate the mean and the covariance of each obstacle over the planning horizon assuming no observations are made, which yields a conservative motion plan compared to the case that the Kalman filter updates (7d) and (7e) are included in the optimization problem. Including these equations in the constraints imposes additional complexity, however, as they are nonlinear and the observations are inherently stochastic. In [16], the stochasticity of these constraints is avoided by assuming that the maximum-likelihood observation is always made at each time step over the planning horizon, which comes at the cost of losing the probabilistic safety guarantees of (16). For this reason, to maintain a tractable optimization problem, we simply assume no observations are made over the planning horizon when solving (16). As will be discussed in more detail in Section III-C, this conservativeness is mitigated due to the fact that we are resolving (16) at each time step of the simulation through a receding horizon control-based approach, with Kalman filter updates taking place in between successive solutions. In what follows, we denote the solution of the SQP procedure by \( \{x_{sup}[t]\}_{t=0}^T \). By construction, \( \{x_{sup}[t]\}_{t=0}^T \) is a dynamically feasible trajectory for the robot that also satisfies the probabilistic safety constraint (10d).

B. Duality-Based Constrained Sensor Scheduling

We now turn our attention to the problem of constrained sensor selection. Intuitively, the robot can dramatically improve its overall performance in future time steps by reducing the uncertainty in the states of the obstacles that adversely affect its current motion. By reducing the uncertainty of these obstacles, the robot is free from adopting conservative trajectories for the sake of safety. However, since the keep-out constraints (11) are enforced via linearized approximations, the impact of the measurement of an obstacle state on \( \{x_{sup}[t]\}_{t=0}^T \) is unclear, as shown in Fig. 2. We propose a refinement of \( \{x_{sup}[t]\}_{t=0}^T \) using the
project-and-linearize approach [8] and show that the influence of a measurement on the obstacle state on the refinement is easily established via sensitivity analysis.

We refine the solution \( \{x_{\text{sqp}}[i]\}_{t=0}^{T} \) by solving the following nonlinear program:

\[
\begin{align*}
\text{min} & \quad J(x[1], \ldots, x[T], u[0], \ldots, u[T-1]) \\
\text{s.t.} & \quad C_{\text{proj}} \left[ x[0]^{T}, \ldots, x[T]^{T} \right]^{T} \leq d_{\text{proj}} \quad (10b), (10c), (16d), (16e). \tag{17b}
\end{align*}
\]

Here, the matrix \( C_{\text{proj}} \in \mathbb{R}^{(TN_{O})x(Tn)} \) and the vector \( d_{\text{proj}} \in \mathbb{R}^{(TN_{O})} \) define a convex feasible set defined by the supporting hyperplanes at the projection of \( \{x_{\text{sqp}}[i]\}_{t=0}^{T} \) onto the keep-out ellipsoids (11), depicted by the green line in Fig. 2. Note that the optimization problem (17) is guaranteed to find a solution, which we will denote as \( \{x_{\text{proj}}[i]\}_{t=0}^{T} \), no worse than \( \{x_{\text{sqp}}[i]\}_{t=0}^{T} \), since \( \{x_{\text{sqp}}[i]\}_{t=0}^{T} \) is feasible for (17).

While the safety constraints in (16) are enforced via a linearization of the nonconvex quadratic constraint (11) at each iteration of the SQP procedure, the safety constraints in (17) are enforced via the supporting hyperplanes of the keep-out ellipsoids (11). Consequently, the effect of shrinking the keep-out sets directly influences \( d_{\text{proj}} \) in contrast to the linearized constraints.

The effect of obstacle state measurement can also be viewed as a relaxation of the nonlinear program (17), wherein \( d_{\text{proj}} \) is replaced by \( d_{\text{proj}} + \delta \) for some slack variable \( \delta \in \mathbb{R}^{(TN_{O})} \) such that \( \delta \geq 0 \). After obtaining a new observation, the maximum eigenvalue of the covariance matrix of the obstacle position shrinks due to the subtraction of the Kalman gain term in (7e). This reduction in the covariance in turn shrinks the keep-out ellipsoids (11) of the obstacle, consequently relaxing the obstacle avoidance constraints in the subsequent solution of the motion-planning problem of the robot.

We now utilize sensitivity analysis, along with a mild assumption, to design a constrained sensor scheduling strategy based on the preceding discussion.

**Lemma 1.** (Sensitivity Analysis [20, Sec. 5.6]): Consider the \( \delta \)-perturbed, convex optimization problem

\[
\begin{align*}
\text{min} & \quad f_{0}(x) \\
\text{s.t.} & \quad x \in \mathcal{X} \quad (18b) \\
& \quad f_{i}(x) \leq \delta_{i} \forall i \in [1,N] \quad (18c)
\end{align*}
\]

with decision variable \( x \in \mathbb{R}^{n} \), convex \( f_{i} \forall i \in [1,N] \), convex set \( \mathcal{X} \), and perturbation vector \( \delta = [\delta_{1}, \ldots, \delta_{N}] \in \mathbb{R}^{N} \). Let \( p^{*}(\delta) : \mathbb{R}^{N} \rightarrow \mathbb{R} \) denote the optimal value of (18). Then, whenever strong duality holds for (18) with \( \delta = 0 \), we have

\[
p^{*}(\delta) \geq p^{*}(0) - (\lambda^{*})^{T} \delta \quad (19)
\]

for every \( \delta \in \mathbb{R}^{N} \), where \( \lambda^{*} \in \mathbb{R}^{N} \) is the optimal dual variable vector corresponding to (18) with \( \delta = 0 \). In addition to strong duality, if \( p^{*} \) is differentiable at \( \delta = 0 \), then relaxing the \( i \)-th constraint \( f_{i}(x) \leq \delta_{i} \), for some \( \delta_{i} > 0 \), approximately decreases the optimal value by \( \lambda^{*}_{i} \delta_{i} \).

**Assumption 3:** Each QP subproblem of (17) satisfies Slater’s condition, i.e., it has a strictly feasible solution.

The validity of Assumption 3 can be easily checked via the positivity of the Chebyshev radius of the polytope corresponding to the feasible set of (17). Recall that the computation of the Chebyshev radius is a linear program [20, Ch. 8]. In all of our simulation and hardware experiments, we found that Assumption 3 was always satisfied. Under Assumption 3, strong duality always holds for each QP subproblem of the SQP procedure for the refined nonlinear program in (17). Furthermore, the optimal solution map \( p^{*} \) for (17) is always differentiable via the Karush-Kuhn-Tucker (KKT) conditions [20, Ch. 5].

**Constrained Sensor Scheduling Strategy:** By Lemma 1 and Assumption 3, the obstacle measurement that corresponds to the greatest impact on the improvement of the performance of the robot should correlate to the obstacle with the largest values for its associated dual variables. Consequently, for each obstacle \( o \in [1,N_{O}] \), we first compute a scalar \( \Lambda_{o} = \sum_{t=1}^{T} \gamma^{t} \lambda_{o,t}^{*} \) using the optimal dual variables. Here, \( \gamma \in (0,1) \) serves as a discount factor that allows the user to balance the present uncertainty in the state of an obstacle with its increasingly uncertain future state position. For example, in a cluttered environment, it may be preferable for the robot to sense nearby obstacles that affect its
trajectory, rather than sensing one far away that affects the end of its motion plan. In this case, decreasing the discount factor drives the robot to sense these nearby obstacles. We subsequently measure the underlying states of the obstacles corresponding to the top $K$ values of $\Lambda_o$. Based on the measurements, we recompute their associated (smaller) keep-out ellipsoids (11) through the Kalman filter update (7a)–(7e). For the obstacles that were not measured, we continue using the original keep-out ellipsoids for motion planning.

Remark 2: A limitation of the proposed sensor selection strategy is its inability to handle a pathological case where one obstacle is unilaterally observed while the ever-growing uncertainty of another obstacle results in an infeasible motion plan for the robot. This scenario is typical of the well-known exploration–exploitation dilemma [27]. Future work will study how to prevent such cases.

C. Numerical Implementation

We summarize the Safely motion planner used to solve (10) in Algorithm 1, which follows a receding horizon control-based approach that iteratively updates the motion plan of the robot at runtime as observations about the obstacles are made. At each iteration of the while loop, the obstacle mean state and uncertainty are first propagated forward in time, conservatively assuming no observations. Note that since we consider linear Gaussian systems for the obstacle dynamics, we have closed-form expressions for all terms in the propagation and the construction step [8]. Recall that constructing $Q_o^+[t]$ in (12a) requires specifying a direction $l_o$. A natural choice for this direction given an obstacle $o$ and time step $t$ is $l_o = (\mu_o[t] - x[t])/(\|\mu_o[t] - x[t]\|)$, i.e., the ellipsoidal outer-approximation $E(\mu_o[t], Q_o^+[t])$ is tight in the direction from the obstacle’s mean estimated position to the position of the robot. Once these parameters are obtained, (16) is then solved to local optimality via SQP, wherein a computationally efficient QP is solved at each iteration. Once this inner while loop terminates, the solution is then refined by a projection-based approach, wherein we can compute the projections used to construct $C_{proj}$ and $d_{proj}$ for the ellipsoidal outer-approximations by solving

$$\min_{x \in \mathbb{R}^n} \|x - x_{\text{sqp}}[t]\|^2$$

s.t. $(x - \mu_o[t])^T(Q_o^+[t])^{-1}(x - \mu_o[t]) \leq 1$

for each $o \in \mathbb{N}_{[1,N_o]}$ and each $t \in \mathbb{N}_{[1,T]}$, which is a convex second-order cone program that can be solved efficiently using off-the-shelf solvers like ECOS [28]. Once the motion plan of the robot is obtained, it subsequently uses the dual variables corresponding to the collision–avoidance constraints to select a subset of the obstacles to sense according to its observation model, using a Kalman filter to then update its estimate of the position of the obstacles given these observations. The agent then progresses to the subsequent time step by transitioning to the next state output in the projection-based solution, setting the obstacle mean state positions and uncertainties to their updated estimates.

### Algorithm 1: Safely Motion Planner.

**Input:** robot dynamics (4), initial state $x[0]$, target state $x_g$, tolerance $\epsilon$, feasible control input $U$, and safe set $S$; obstacle shape $B_o$, dynamics (5), and initial belief $x_o[0] \sim N(\mu_o[0], \Sigma_o[0])$ for each $o \in \mathbb{N}_{[1,N_o]}$; maximum probability $\alpha$, planning horizon $T$, discount factor $\gamma$, observation limit $K$.

**while** $|x[0] - x_o| \geq \epsilon$ **do**

  **Propagate, construct ellipsoids:**

  for $o = 1 \ldots N_o$ do

  for $t = 1 \ldots T$ do

  Compute $\mu_o[t], \Sigma_o[t]$ by (8b),(8c)

  Compute $Q_o[t]$ by (12b)

  Compute $Q^+_o[t]$ by (12a)

  Store $E(\mu_o[t], Q_o^+[t])$

  **Solve SQP:**

  while $|J_{\text{new}} - J_{\text{old}}| \geq \epsilon$ do

  $J_{\text{new}} = \min (16)$

  $x_{\text{sqp}}[t]_{t=0} = \text{argmin}(16)$

  **Projection:**

  Compute $C_{proj}, d_{proj}$

  $x_{pr}[t]_{t=0} = \text{argmin} (17)$

  Compute $\Lambda_o[t]$ for $o \in \mathbb{N}_{[1,N_o]}$

  **Observe and update obstacle uncertainty models:**

  for $o = 1 \ldots N_o$ do

  if $\Lambda_o \in K$ largest then

  Draw observation $x_o[1]$ according to (6)

  Update $x_o[1], \Sigma_o[1]$ according to (7a)–(7e)

  else

  Update $x_o[1], \Sigma_o[1]$ according to (8a)–(8c)

  **Reset:**

  $x[0] = x_{pr[1]}$

  for $o = 1 \ldots N_o$ do

  $\mu_o[0] = \mu_o[1], \Sigma_o[0] = \Sigma_o[1]$

### IV. EXPERIMENTAL VALIDATION

We now consider several numerical experiments demonstrating the Safely motion planner. For all examples, we use a value of $\alpha = 0.01$. Furthermore, note that in all examples we assume either circular or spherical obstacle shapes, satisfying the symmetric cover requirement of Assumption 1.

#### A. Software Experiment Using Linear Robot Dynamics

We consider the environment shown in Fig. 3(a), where the safe set $S$ of the robot is the $6 \times 6 \times 6$ m-black cube centered at the origin. The robot begins in the lower left corner in state $x[0] = [-2.75, -2.75, -2.75]^T$ and must travel to the target state located at $x_g = [2.75, 2.75, 2.75]^T$. We use the quadratic
where \( x[t] \in \mathbb{R}^6, u[t] \in \mathbb{R}^3 \), and
\[
A = \begin{bmatrix}
I_3 & dt \cdot I_3 \\
0_{3 \times 3} & I_3
\end{bmatrix},
B = \begin{bmatrix}
dt^2/2 \cdot I_3 \\
dt \cdot I_3
\end{bmatrix}
\]
in which \( dt \) is the time interval between discrete steps, here set to \( dt = 0.25 \) s. Note that \( x[t] \) is decomposed as \( x[t] = [p[t] \top, v[t] \top] \top \), where \( p[t], v[t] \in \mathbb{R}^3 \) are the position and the velocity components of the state vector, respectively. The parameters for the sensor model (6) are given by
\[
H = I_3, \quad \mu_\nu = [0, 0, 0] \top, \quad \Sigma_\nu = 0.05 I_3.
\]
Furthermore, each obstacle \( o \in \mathbb{N}[1,N_o] \) follows the linear dynamics (5) with
\[
A_o = I_3, \quad B_o = dt \cdot I_3.
\]
The control input \( u[t] = [u_1[t], u_2[t], u_3[t]] \top \) of the robot is constrained to \( U = \{u_1[t], u_2[t], u_3[t] : \|u[t]\|_\infty \leq 0.25 \} \) and the velocity \( v[t] = [v_1[t], v_2[t], v_3[t]] \top \) is constrained to \( V = \{v_1[t], v_2[t], v_3[t] : \|v[t]\|_\infty \leq 0.5 \} \). Finally, the parameters of the Gaussian disturbances for each obstacle are provided in the Appendix. We allow a single underlying obstacle state to be measured at each time step (setting \( K = 1 \)). We use the CASADI Python interface [29] with the OSQP SQP solver [30] to solve (16). Furthermore, since the state dynamics are linear and the cost function is quadratic, the refined problem in (17) reduces to a QP, which we solve using OSQP. The simulation is run on a 1.8-GHz Intel Core i7-8550 U CPU with 16 GB RAM. Fig. 4 displays the resulting trajectory obtained using the Safely motion planner in the case that \( T = 25 \) and \( \gamma = 1 \). In this case, the robot required a total of 61 time steps to reach the target state from its initial position, with the relevant obstacle observed at each time step shown in Fig. 6. After initially observing obstacle \( O_4 \), the robot is most affected by obstacle \( O_1 \) as it attempts to move through the center of the environment, as shown in Fig. 4(b). After briefly observing obstacle \( O_3 \), as shown in Fig. 4(c), the robot repeatedly observes obstacle \( O_5 \) since it resides close to its goal state. By time step 50, as shown in Fig. 4(f), the robot has sufficiently reduced the uncertainty associated with obstacle \( O_3 \) such that it no longer needs to observe it in order to reach its goal state.

Fig. 5 shows how the planning horizon used affects the resulting behavior of the robot in Safely. As discussed in Section II-C, there exists a tradeoff between the computation time required and the efficiency of the resulting motion plan. In this example, although Safely with a planning horizon of \( T = 5 \) time steps requires significantly less computation time per iteration of Safely, its near-myopic planning causes the robot to get trapped by the obstacles, whereas the robot is able to reach its goal state for all other planning horizons. Notably, a planning horizon of \( T = 15 \) is able to obtain a motion plan through Safely as efficient as that of \( T = 25 \) or \( T = 35 \), while requiring less computation time on average.

We also consider how the choice of the discount factor \( \gamma \) affects which obstacle the robot chooses to sense at each time step. We consider three values of \( \gamma \), with the resulting relevant obstacle choices shown in Fig. 6. In this example, we observe

Fig. 3. Initial configurations of the sample environments. (a) Initial configuration of Example 1. (b) Initial configuration of Example 2.
that the value of $\gamma$ only has a small effect on the resulting choice of relevant obstacle as the exact same obstacles are chosen at each time step for both $\gamma = 1$ and $\gamma = 0.75$. Note that when we extract the dual variables from the optimization solver in Safely and subsequently compute their discounted value, we filter out values less than some small $\hat{\epsilon} > 0$ due to numerical imprecision. For this reason, in the case of $\gamma = 0.5$, the discounted dual variable corresponding to obstacle $O_3$ is ignored, and the robot makes no observation. Finally, note that although the robot passes near obstacle $O_3$, it is never deemed the most relevant obstacle and is thus never observed.

**B. Software Experiment Using Linear Dynamics and Varying Observation Strategies**

We again consider that the robot operates in the environment shown in Fig. 3(a) and uses the sensor model given by (21). In this experiment, however, the robot must now avoid colliding with a set of $N_o = 20$ obstacles whose initial positions, disturbance covariance matrices, and spherical shape approximations are randomly instantiated. Note that we assume each obstacle has zero mean disturbance. The process for sampling these parameters for each obstacle is discussed in the Appendix.

We are interested in studying how the proposed duality-based sensor selection strategy compares to several alternative sensor
selection strategies. Specifically, each of these alternative strategies corresponds to the robot as follows.

1) **(Uncertain)** Sensing the obstacle with the largest uncertainty, i.e., at time step \( t \), the robot senses the obstacle \( o^* \) such that \( o^* = \arg \max_{o \in \{1, \ldots, N_o\}} \text{Tr}(\Sigma_o[t]) \).

2) **(Closest)** Sensing the obstacle whose estimated position is closest to the robot’s current position, i.e., at time step \( t \), the robot senses the obstacle \( o^* \) such that \( o^* = \arg \min_{o \in \{1, \ldots, N_o\}} ||x[t] - \hat{x}_o[t]|| \).

3) **(Uniform)** Sensing each obstacle, iteratively switching between them at each time step, i.e., at time step \( t \), the robot senses the obstacle \( o^* \) such that \( o^* = \text{mod}(t, N_o) + 1 \), where \( \text{mod}(\cdot, \cdot) \) is the standard modulo operator.

We consider 100 random instantiations of each of the 20 obstacles, and fix their true trajectories over each iteration of the four sensor selection strategies. We then run Algorithm 1 using either the proposed duality-based sensor selection strategy or one of the three alternative strategies listed above. In each case, we assume that at most one obstacle can be observed at each time step (setting \( K = 1 \)). For all random instantiations and sensor selection strategies, we use a discrete time step of \( dt = 0.25 \), a planning horizon of \( T = 20 \) time steps, and set \( \alpha = 0.01 \). Finally, we constrain the robot’s control input according to \( ||u[t]||_{\infty} \leq 0.5 \) and its velocity according to \( ||v[t]||_{\infty} \leq 0.5 \).

For a fixed obstacle instantiation and sensor selection strategy, we define a **successful run** as a simulation in which the robot is able to reach its goal state without one of the following three conditions occurring.

1) **(Collision)** The robot collides with one of the obstacles, i.e., there exists an \( o \in N_o \) such that \( ||x[0] - \bar{x}_o[0]|| \leq r_o \) at some iteration of Algorithm 1.

2) **(No buffer)** The optimization problem (17) in Algorithm 1 fails to return a solution at least \( T \) time steps in a row, i.e., the robot can no longer execute an action from its last motion plan obtained from the solution of (17).

3) **(Timeout)** The robot fails to reach the goal state in at most \( T = 100 \) iterations of Algorithm 1.

Table I shows the number of successful runs, the number of each type of failure observed, the quantiles of iterations required for completion of successful runs, and the average number of observations made for each successful run for the sensor selection strategies considered. As shown in this table, using either the proposed Safely sensor selection strategy or the “closest” sensing strategy, yielded the greatest number of successful runs out of the strategies considered and never failed due to collision with an obstacle. From the quantile information, a robot using Safely required much fewer iterations to reach the goal without any collisions, as compared to other strategies. Furthermore, Safely required fewer observations as it only devotes sensor resources toward observing an obstacle when that obstacle is relevant to its trajectory, i.e., when its dual variables are nonzero. In all other instances, the sensor selection strategies always make an observation of an obstacle at each time step, even if that obstacle may not be relevant to its current motion plan.

**C. Software Experiment Using Nonlinear Robot Dynamics**

We now apply the Safely motion planner to the case that the robot follows the dynamics of Dubins’ vehicle \([31]\), where

\[
x[t+1] = x[t] + dt \begin{bmatrix} v[t] \cos \theta[t] \\ v[t] \sin \theta[t] \end{bmatrix}
\]

\[
\theta[t+1] = \theta[t] + dt \cdot \omega[t]
\]

in which \( v[t] \in [v_{\text{min}}, v_{\text{max}}] \subset \mathbb{R} \) is the velocity input in m/s, \( \theta[t] \in (-\pi, \pi] \) is the heading angle, \( \omega[t] \in [\omega_{\text{min}}, \omega_{\text{max}}] \subset \mathbb{R} \) is the turning rate in rad/s, and \( dt \) is the discrete time step. We consider these dynamics since they allow us to examine situations where the orientation of the robot affects what it can sense. Specifically, we consider the case that the robot uses a fixed, forward-facing camera with viewing angle \( \theta_v \), as shown in Fig. 7. When Safely identifies a relevant obstacle to sense, the robot must adjust its motion plan to locate this obstacle within its field of view (FOV). Since there is an explicit coupling between sensing and planning, we adjust the objective function in (20) to

\[
\sum_{t=1}^{T} -\beta \gamma^t \langle \mu_o, \mu \rangle - \langle x[t], u_0(t) \rangle + ||x[t] - x_g||^2
\]
where \( u_p[t] = \left[ \cos(\theta[t]), \sin(\theta[t]) \right]^\top \), \( \beta \in \mathbb{R}_+ \) is a weighting parameter, \( \gamma \in (0, 1] \) is a discount factor, and \( \mu_{\nu[t]} \) refers to the mean position at time \( t \) of the current most relevant obstacle. The first term in (24) is the inner product between the heading angle of the robot and the direction vector from the robot to the most relevant obstacle’s mean position, which incentivizes the robot to turn toward the most relevant obstacle. Note that if no obstacle is identified as relevant at a given time step, then the first term in (24) is omitted for that iteration of \( \text{Safely} \). By discounting the first term, the resulting obstacle dual variables should continue to indicate the relevance of an obstacle in regard to the primary task of reaching the target state.

In this example, we consider the environment shown in Fig. 3(b), where the safe set \( S \) of the robot is the \( 6 \times 4 \)-m black square centered at the origin. The robot begins in the lower left in state \( x[0] = [-2.75, -1.00]^\top \) and must travel to the target state located at \( x_g = [1.9, 1.9]^\top \). We use \( dt = 0.25 \) s and \( T = 20 \) time steps, and set \( \beta = 10 \) and \( \gamma = 0.8 \). Finally, we set \( v[t] \in [0.01, 0.25] \) m/s, \( \omega[t] \in [-\frac{\pi}{2}, \frac{\pi}{2}] \) rad/s, and set \( \theta_v = \frac{\pi}{2} \). Furthermore, the obstacles follow the linear dynamics
\[
A_o = I_2, \quad B_o = dt \cdot I_2
\] (25)

with their corresponding parameters provided in the Appendix. Finally, the parameters for the sensor model (6) are given by
\[
H = I_2, \quad \mu_{\nu} = [0, 0]^\top, \quad \Sigma_{\nu} = 0.05I_2.
\] (26)

All remaining parameters are the same as in the previous example. Since the objective and the constraints are nonconvex, we use \texttt{WORHP} to solve both (16) and (17).

As a comparison, we additionally implement a modified version of the belief-space planning (BSP) method proposed in [16], which considers obstacle uncertainty in the motion planning problem but not their relevance. Specifically, we modify the BSP framework by removing the hard end-point constraint and imposing obstacle avoidance by enforcing the ellipsoidal keep-out constraints corresponding to the \( Q^+ \) matrices constructed using the previous iterate’s output from [16, line 9 of Algorithm 1]. We again use \texttt{WORHP} to solve the SQP subproblems in each iteration of the BSP algorithm. Fig. 8 displays the resulting trajectories over the course of each simulation. For \( \text{Safely} \), the robot only allocates sensor resources toward sensing obstacle \( O_2 \) since it travels directly through the shortest path between the initial state and the target state of the robot. After initially turning toward this obstacle at time step in between time step 25 and time step...
Fig. 11. Results of the hardware experiment at several time steps of interest. The upper row shows the robot’s understanding of the environment, where the black marks indicate the trajectory of the robot up to the current time step and the light blue marks indicate the future motion plan obtained from the current iteration. The red marks indicate the realization of the random obstacle trajectories up to the current time step and the ellipsoids indicate the history of the present keep-out regions for guaranteeing safety (Proposition 1), where it is shaded blue if it was observed at that time step. The lower row shows the corresponding physical state, with the red triangle representing the field of view of the robot.

50, as shown in Fig. 8(b) and (c), the robot soon observes it, reducing the uncertainty in its state position. The robot must turn toward this obstacle several more times as it continues to plan to travel above obstacle \( O_2 \). After sufficiently many observations of \( O_2 \), the robot eventually plans to travel beneath it instead, as shown in Fig. 8(d). By time step 100, the robot has an unimpeded trajectory to the target state, as shown in Fig. 8(e).

Without the notion of obstacle relevance, the robot using the BSP method is unable to reach the target state without violating the collision–avoidance constraint. Initially, in Fig. 8(g), the robot’s motion plan seeks to align itself with obstacle \( O_1 \), as doing so provides the greatest uncertainty reduction while still adequately moving toward the target state. Furthermore, note that obstacle \( O_2 \) pushes the robot closer to obstacle \( O_1 \), as shown in Fig. 8(i). Due to the large uncertainty associated with obstacle \( O_1 \), the robot begins to turn toward it in order to make an observation. However, turning toward this obstacle causes the robot to become “pinched” between obstacles \( O_1 \) and \( O_2 \), as shown in Fig. 8(j). Soon afterward, the robot has no safe nominal trajectory and subsequently violates the collision–avoidance constraint. Through this example, it is evident that obstacle relevance cannot simply be captured by the uncertainty in the position of that obstacle.

We also note that since the belief-space planning method requires solving multiples iterations of a more complex SQP at each time step, it requires significantly more computation time. Although our implementation of the BSP method is not optimized for computational efficiency, Fig. 9 highlights the significantly lower computation time per iteration required by \texttt{Safely}.

D. Hardware-Based Experiment Using Velocity Inputs and Turtlebots

We now examine the feasibility of implementing the \texttt{Safely} motion planner in real-time hardware applications. Specifically, we consider the same robot and obstacle dynamics as in the previous example; however, rather than being given the next waypoint to visit, the robot is instead given the velocity and turning rate inputs \( v[0] \) and \( \omega[0] \) output from Algorithm 1. The robot then uses these inputs until the next iteration of \texttt{Safely} completes, and the process then repeats. Thus, each iteration of \texttt{Safely} must terminate sufficiently quickly such that the environment does not change too drastically before the completion of the subsequent iteration.

The environment is displayed in Fig. 10, where \( x[0] = [-2.5, -0.75]^\top \) and \( x_g = [1.9, 1.9]^\top \). The set \( S \) is the 6 \times 4-m rectangle centered at the origin. Again, the obstacle parameters are provided in the Appendix. For the robot, we use a Turtlebot 3 Waffle Pi connected to the robot operating system (ROS), setting \( v[t] \in [0.01, 0.25] \) m/s and \( \omega[t] \in [-\frac{\pi}{2}, \frac{\pi}{2}] \) rad/s. Furthermore, the built-in Raspberry Pi camera allows a viewing angle of \( \theta_v = 1.09 \) rad. For this experiment, we set \( dt = 0.25 \) s and use a planning horizon of \( T = 25 \) time steps. All other parameters are identical to the previous example. Finally, we use a set of Turtlebot 3 Burger Pis for the environment obstacles. To
detect the obstacles at each sensing step in Safely, we map each one to a color and use OpenCV [32] to search the acquired image for these colors. Specifically, we map $O_1$ to yellow, $O_2$ to green, and $O_3$ to pink. If an obstacle is detected through this method, its precise location is provided using a motion capture system, for which the robot then makes a noisy observation of it through (26).

We run the experiment on a 2.6-GHz Intel Core i7-9750H with 64 GB RAM. To reach the target state, the robot required a total of 217 iterations of Safely. Fig. 11 plots the environment of the robot and its planned trajectory at several relevant time steps. At earlier time steps, the uncertainty in the position of obstacle $O_1$ pushes the robot toward the top of $\mathcal{S}$. The robot then turns to observe $O_1$ several times, as shown in the first and second columns of Fig. 11. Afterward, there is a brief period where the robot is able to travel unimpeded to the target state. Eventually, however, the uncertainty associated with obstacle $O_3$ intersects the nominal trajectory of the robot. The robot then turns toward this obstacle, thereby reducing its uncertainty, as shown in the third column of Fig. 11. Afterward, the robot is again able to travel unimpeded to the target state. Fig. 12 displays the computation time per iteration of Safely. Note that all solve times are less than the 0.25-s time interval between discrete steps in (23), indicating that Safely shows promise for online applications with collision-avoidance constraints.

V. CONCLUSION

In this article, we consider the problem of incorporating the relevance of an obstacle into the constrained sensor selection problem of a robot. To this end, we formulate the Safely motion planner, wherein we use information about the dual variables of the linearized constraints corresponding to the “keep-out” ellipsoids of each obstacle. Through a sensitivity analysis, we show that measuring the state of the obstacle with the largest discounted sum of these dual variables leads to the best gain in performance for the robot. We demonstrate the efficacy of the proposed motion planner in both software and hardware experiments, and find that Safely shows promise for real-time hardware implementations.

APPENDIX

A. Constants for Each SQP Iteration

The values of the constants in (16) are given by

\[
C_{obs}^{k}[t] = 2(Q_o^{k})^{-1}(x_0^{k}[t] - \mu_o[t])
\]

\[
d_{obs}^{k}[t] = 1 - (x_0^{k}[t] - \mu_o[t])^T(Q_o^{k})^{-1}(x_0^{k}[t] - \mu_o[t]) + 2(Q_o^{k})^{-1}(x_0^{k}[t] - \mu_o[t])x_0^{k}[t],
\]

\[
C_{dyn}^{k}[t] = \nabla f(x_0^{k}[t], u_0^{k}[t]),
\]

\[
d_{dyn}^{k}[t] = f(x_0^{k}[t], u_0^{k}[t]) - (\nabla f(x_0^{k}[t], u_0^{k}[t]))^T [x_0^{k}[t] \quad u_0^{k}[t]].
\]

1A video of the experiment is available at https://youtu.be/GO8hFzmnb_0.

B. Obstacle Details for the Software Experiment With Linear Robot Dynamics

For the first simulation experiment, the initial positions of the obstacles are given by

\[
x_1[0] = \begin{bmatrix} 0.00 \\ -0.25 \end{bmatrix}, \quad x_2[0] = \begin{bmatrix} 0.50 \\ 0.00 \end{bmatrix}, \quad x_3[0] = \begin{bmatrix} -0.80 \\ 2.25 \end{bmatrix}
\]

\[
x_4[0] = \begin{bmatrix} -1.90 \\ -2.75 \end{bmatrix}, \quad x_5[0] = \begin{bmatrix} 1.70 \\ 2.70 \end{bmatrix}.
\]

Furthermore, the obstacles have Gaussian disturbances $w_o$ with mean vectors

\[
\mu_{w_1} = \begin{bmatrix} -0.10 \\ 0.00 \end{bmatrix}, \quad \mu_{w_2} = \begin{bmatrix} -0.10 \\ 0.00 \end{bmatrix}, \quad \mu_{w_3} = \begin{bmatrix} 0.05 \\ 0.01 \end{bmatrix}
\]

\[
\mu_{w_4} = \begin{bmatrix} 0.10 \\ 0.00 \end{bmatrix}, \quad \mu_{w_5} = \begin{bmatrix} -0.10 \\ -0.03 \end{bmatrix}
\]

and covariance matrices

\[
\Sigma_{w_1} = \begin{bmatrix} 0.02 & 0.002 & 0.002 \\ 0.002 & 0.025 & 0.002 \\ 0.002 & 0.002 & 0.025 \end{bmatrix}, \quad \Sigma_{w_2} = \begin{bmatrix} 0.02 & 0.002 & 0.002 \\ 0.002 & 0.025 & 0.002 \\ 0.002 & 0.002 & 0.025 \end{bmatrix}
\]

\[
\Sigma_{w_3} = \begin{bmatrix} 0.02 & 0.004 & 0.004 \\ 0.004 & 0.004 & 0.004 \\ 0.004 & 0.004 & 0.004 \end{bmatrix}, \quad \Sigma_{w_4} = \begin{bmatrix} 0.12 & 0.004 & 0.004 \\ 0.004 & 0.004 & 0.004 \\ 0.004 & 0.004 & 0.004 \end{bmatrix}
\]

\[
\Sigma_{w_5} = 0.01 \cdot I_3.
\]

C. Obstacle Details for the Software Experiment Using Linear Robot Dynamics and Varying Observation Strategies

To obtain the initial position of each obstacle, we uniformly at random sample a state within the safe set $\mathcal{S}$ of the robot. Furthermore, we do not allow any of the sampled obstacles to have an initial position with a distance less than 1 m to either the robot’s initial or goal state. This ensures that the initial configuration is feasible and that no obstacle prevents the robot from reaching its goal.

Furthermore, we assume that the Gaussian disturbance associated with each obstacle has a mean $\mu_{w_o} = [0, 0, 0]^T$ for each $o = 1, \ldots, 20$ and a covariance $\Sigma_{w_o}$, that is constructed as follows. First, we uniformly at random sample a $3 \times 3$ lower triangular matrix $L_o$ whose elements are bounded between 0 and 0.2. If any diagonal elements of $L_o$ are equal to zero, we instead set them to some small, positive constant $\hat{c} > 0$. We then form $\Sigma_{w_o}$ according to $\Sigma_{w_o} = L_o L_o^T$, which ensures positive definiteness. Finally, we uniformly at random sample the radius $r_o$ of the spherical shape approximation for each obstacle from the interval [0.2,0.4] meters.
D. Obstacle Details for the Software Experiment Using Nonlinear Robot Dynamics

For the second simulation experiment, the initial positions of the obstacles are given by

\[
x_1[0] = \begin{bmatrix} -2.00 \\ 2.00 \end{bmatrix}, \quad x_2[0] = \begin{bmatrix} -0.50 \\ -2.00 \end{bmatrix}, \quad x_3[0] = \begin{bmatrix} 2.75 \\ -1.75 \end{bmatrix},
\]

Furthermore, the obstacles have Gaussian disturbances \( w_0 \) with mean vectors

\[
\mu_{w_1} = \begin{bmatrix} 0.00 \\ 0.00 \end{bmatrix}, \quad \mu_{w_2} = \begin{bmatrix} 0.025 \\ 0.20 \end{bmatrix}, \quad \mu_{w_3} = \begin{bmatrix} -0.1 \\ 0.0 \end{bmatrix}
\]

and covariance matrices

\[
\Sigma_{w_1} = \begin{bmatrix} 0.025 & 0.001 \\ 0.001 & 0.025 \end{bmatrix}, \quad \Sigma_{w_2} = \begin{bmatrix} 0.006 & 0.0015 \\ 0.0015 & 0.008 \end{bmatrix}, \quad \Sigma_{w_3} = \begin{bmatrix} 0.001 & 0.0 \\ 0.0 & 0.001 \end{bmatrix}.
\]

E. Obstacle Details for the Hardware Experiment.

For the hardware experiment, the initial positions of the obstacles are given by

\[
x_1[0] = \begin{bmatrix} -0.75 \\ -2.00 \end{bmatrix}, \quad x_2[0] = \begin{bmatrix} 1.20 \\ 1.80 \end{bmatrix}, \quad x_3[0] = \begin{bmatrix} 2.25 \\ -2.00 \end{bmatrix}.
\]

Furthermore, the obstacles have Gaussian disturbances \( w_0 \) with mean vectors

\[
\mu_{w_1} = \begin{bmatrix} 0.00 \\ 0.20 \end{bmatrix}, \quad \mu_{w_2} = \begin{bmatrix} 0.05 \\ -0.10 \end{bmatrix}, \quad \mu_{w_3} = \begin{bmatrix} -0.06 \\ 0.10 \end{bmatrix}
\]

and covariance matrices

\[
\Sigma_{w_1} = \begin{bmatrix} 0.008 & 0.001 \\ 0.001 & 0.008 \end{bmatrix}, \quad \Sigma_{w_2} = \begin{bmatrix} 0.004 & 0.0015 \\ 0.0015 & 0.005 \end{bmatrix}, \quad \Sigma_{w_3} = \begin{bmatrix} 0.008 & 0.0025 \\ 0.0025 & 0.0125 \end{bmatrix}.
\]

REFERENCES

[1] D. Jennings and M. Figliozzi, “Study of sidewalk autonomous delivery robots and their potential impacts on freight efficiency and travel,” Transp. Res. Rec., vol. 2673, no. 6, pp. 317–326, 2019.
[2] M. Darms, P. Rybski, and C. Urmson, “Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments,” in Proc. IEEE Intell. Veh. Symp., 2008, pp. 1197–1202.
[3] P. Schlosser and C. Ledermann, “Achieving hard-real-time capability for 3D human pose estimation systems,” in Proc. IEEE Int. Conf. Robot. Automat., 2021, pp. 3772–3778.
[4] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, Estimation With Applications to Tracking and Navigation: Theory Algorithms and Software. Hoboken, NJ, USA: Wiley, 2004.
[5] C. E. Rasmussen, “Gaussian processes in machine learning,” in Summer School on Machine Learning, Berlin, Germany: Springer, 2003, pp. 63–71.
[6] G. S. Aoudi, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, “Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns,” Auton. Robots, vol. 35, no. 1, pp. 51–76, 2013.
[7] A. P. Vinod and M. M. Oishi, “Probabilistic occupancy via forward stochastic reachability for Markov jump affine systems,” IEEE Trans. Autom. Control, vol. 66, no. 7, pp. 3068–3083, Jul. 2021.
[8] A. P. Vinod, S. Rice, Y. Mao, M. M. Oishi, and B. Açıkmeşe, “Stochastic motion planning using successive convexification and probabilistic occupancy functions,” in Proc. IEEE Conf. Decis. Control, 2018, pp. 4425–4432.
[9] P. T. Boggs and J. W. Tolle, “Sequential quadratic programming,” Acta Numerica, vol. 4, pp. 1–51, 1995.
[10] N. E. Du Toit and J. W. Burdick, “Robot motion planning in dynamic, uncertain environments,” IEEE Trans. Robot., vol. 28, no. 1, pp. 101–115, Feb. 2012.
[11] S. Ghosh and J. Biswas, “Joint perception and planning for efficient obstacle avoidance using stereo vision,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2017, pp. 1026–1031.
[12] W. Pryor, Y.-C. Lin, and D. Berenson, “Integrated affordance detection and humanoid locomotion planning,” in Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots, 2016, pp. 125–132.
[13] D. Fox, W. Burgard, and S. Thrun, “Markov localization for mobile robots in dynamic environments,” J. Artif. Intell. Res., vol. 11, pp. 391–427, 1999.
[14] L. J. Guibas, D. Hsu, K. Kurniawati, and E. Rehman, “Bounded uncertainty roadmaps for path planning,” in Algorithmic Foundation of Robotics VIII. Berlin, Germany: Springer, 2009, pp. 199–215.
[15] L. Zhang, A. Provok, and S. Bhattacharya, “Pursuer assignment and control strategies in multi-agent pursuit- evasion under uncertainties,” Front. Robot. AI, vol. 8, 2021, Art. no. 691637.
[16] S. Patil, Y. Duan, J. Schulman, K. Goldberg, and P. Abbeel, “Gaussian belief space planning with discontinuities in sensing domains,” in Proc. IEEE Int. Conf. Robot. Automat., 2014, pp. 6483–6490.
[17] T. Tanaka, P. M. Esfahani, and S. K. Mitter, “LQG control with minimum directed information: Semidefinite programming approach,” IEEE Trans. Autom. Control, vol. 63, no. 1, pp. 37–52, Jan. 2018.
[18] R. Bajcsy, “Active perception,” Proc. IEEE, vol. 76, no. 8, pp. 966–1005, Aug. 1988.
[19] K. Lee, G. N. An, V. Zakharov, and E. A. Theodorou, “Perceptual attention-based predictive control,” in Proc. Mach. Learn. Res., 2020, vol. 100, pp. 220–232.
[20] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[21] S. Diamond and S. Boyd, “CVXPY: A Python-embedded modeling language for convex optimization,” J. Mach. Learn. Res., vol. 17, no. 1, pp. 42–60, 2016.
[22] E. F. Camacho and C. B. Alba, Model Predictive Control. Berlin, Germany: Springer, 2013.
[23] J. Pitman, Probability. Berlin, Germany: Springer, 1999.
[24] A. Kurzhanski and I. Vályi, Ellipsoidal Calculus for Estimation and Control. Cheltenham, U.K.: Nelson Thornes, 1997.
[25] A. Patil and T. Tanaka, “Upper and lower bounds for end-to-end risks in stochastic robot navigation,” 2021, arXiv:2110.15879.
[26] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, “OSQP: An operator splitting solver for quadratic programs,” Math. Program. Comput., vol. 12, no. 4, pp. 637–672, 2020.
[27] T. Lattimore and C. Szepesvári, Bandit Algorithms. Cambridge, U.K.: Cambridge Univ. Press, 2020.
[28] A. Domahidi, E. Chu, and S. Boyd, “ECOS: An SOCP solver for embedded systems,” in Proc. Eur. Contr. Conf., 2013, pp. 3071–3076.
[29] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “CasADi: A software framework for nonlinear optimization and optimal control,” Math. Program. Comput., vol. 11, no. 1, pp. 1–36, 2019.
[30] C. Bäckström and D. Wassell, “The ESA NLP solver WORKH,” in Modeling and Optimization in Space Engineering. Berlin, Germany: Springer, 2012, pp. 85–110.
[31] A. Shapiro, D. Dentcheva, and A. Ruszczyszinski, Lectures on Stochastic Programming: Modeling and Theory. Philadelphia, PA, USA: SIAM, 2014.
Michael Hibbard (Student Member, IEEE) received the B.S. degree in engineering mechanics and astronautics from the University of Wisconsin-Madison, Madison, WI, USA, in 2018, and the M.S. degree in aerospace engineering from the University of Texas at Austin, Austin, TX, USA, in 2020.

He joined the Department of Aerospace Engineering, University of Texas at Austin, as a Ph.D. student, in Fall 2018. His research interests include the development of theory and algorithms providing formal guarantees for the mission success of autonomous agents with limited sensing capabilities, especially in the context of outer-space systems.

Abraham P. Vinod (Member, IEEE) received the B.Tech. and M.Tech. degrees in electrical engineering from the Indian Institute of Technology, Madras, Chennai, Tamil Nadu, India, in 2014, and the Ph.D. degree in electrical engineering from the University of New Mexico, Albuquerque, NM, USA, in 2018.

He was a Postdoctoral Fellow at the Oden Institute for Computational Engineering and Sciences, University of Texas at Austin, Austin, TX, USA, from 2019 to 2020. Since 2020, he has been with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, where he is currently a Research Scientist. His research interests include learning, planning, and decision-making under uncertainty for autonomous systems.

Dr. Vinod was the recipient of the Best Student Paper Award at the 2017 ACM Hybrid Systems: Computation and Control Conference, and the IITM Prof. Achim Bopp Prize.

Ufuk Topcu (Senior Member, IEEE) received the Ph.D. degree in mechanical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 2008.

He is currently an Associate Professor with the Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin, Austin, TX, USA, where he holds the W. A. “Tex” Moncrief, Jr. Professorship in Computational Engineering and Sciences I. His research interests include the theoretical and algorithmic aspects of the design and verification of autonomous systems, typically in the intersection of formal methods, reinforcement learning, and control theory.

Jesse Quattrociocchi received the master’s degree in aerospace engineering from the University of Texas at Austin, Austin, TX, USA, in 2019.

He is currently a Research Engineer with the Autonomous Systems Group, University of Texas at Austin. His research interests include the application and integration of synthesis and learning-based decision-making algorithms for both simulated and hardware-based autonomous systems.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.