Navigation In GPS-Denied Environments Using Approximate Dynamic Programming

Vu Anh Huynh
Department of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, vuhuynh@mit.edu

Controlling a mobile vehicle to navigate in GPS-denied environments introduces a challenging partially observable control problem with complex constraints. This report presents a combination of various suboptimal control schemes such as open loop feedback control (OLFC), certainty equivalent control (CEC), model predictive control (MPC), and using expected values of estimates as full states to address the above problem. First, we review the connection between several rollout algorithms in dynamic programming and MPC. Second, the review of a recent fast and online method for MPC is provided. Finally, we present simulation results to demonstrate that a combination of the mentioned schemes can provide good suboptimal control policies to solve the above constrained navigation problem.

1. Introduction

In GPS-denied environments such as indoor buildings, a mobile robot does not have perfect knowledge of its position, and it has the risk of colliding with obstacles while following waypoints from a motion planner. In most cases, the dynamics of a system is nonlinear except some special vehicles such as quad-helicopters, whose dynamics can be approximately linear. Even when a system is linear, navigating a robot in these environments imposes an optimization problem with continuous state and control constraints as well as fast dynamics. In particular, the stability of the controller that executes a motion plan in feasible regions is significant for the purpose of safety. Current works in this field often eliminate constraints by introducing a large penalty in the cost function to avoid undesirable situations. Alternatively, there is a low-level obstacle avoidance module that takes a higher control priority to rescue a vehicle from collisions in emergency cases. However, tuning the weight of the penalty component in the cost function or using low-level avoidance module does not directly and actively enforce the state of the system in its feasible regions, and thus collisions still remain an issue to improve.

In this project, we would like to explore a combination of open loop feedback control (OLFC), certainty equivalent control (CEC), and model predictive control (MPC) as well as using estimated expected states as full states to navigate in GPS-denied environments. MPC with full state information was motivated to achieve stability in the advent of nonlinearities and constraints in linear quadratic control framework. The connection of MPC and rollout algorithm in dynamic programming framework suggests that a combination of the mentioned suboptimal schemes can be used to reduce the risk of collisions in our problem of interest. In this work, we assume that the map of an environment is given. First, given a partially observable environment, the motion planner can strategically pick up waypoints that provide valuable information for an estimator. The motion planner also provides a set of state and control constraints while traveling between waypoints. Second, with good estimates of its state, the robot can strive for a collision-free trajectory using MPC.

Figure 1 shows an example of the typical map of an area in which the vehicle may operate. Black objects are obstacles, and red dots are landmarks, or features. The area contains rich information for the vehicle such as corners of buildings, walls, doors, and windows. These objects are called landmarks in a feature-based map. The green star is a starting location, and the red star is a
destination. A typical mission consists of waypoints that are sampled near features, and the vehicle follows these waypoints to reach the destination.

![Feature-based map](image_url)

**Figure 1** An example of a feature-based map.

The report structure is organized as follows. In Section 2, we briefly review key results of the paper "Dynamic Programming and Suboptimal Control: A survey from ADP to MPC" (2). The author provided conceptual connections among popular suboptimal control schemes from the perspective of policy iteration based on a suboptimal/heuristic policy. These schemes include methods in approximate dynamic programming and the popular practical MPC scheme in the control community. The author also unified these schemes to introduce a new scheme that generates restricted structure policies, but the review of this scheme is out of the scope of this report.

In Section 3, we present the main contribution from the paper "Fast Model Predictive Control Using Online Optimization" (4), which speeds up the computation for a single MPC control by the order of 100 times compared to a generic optimizer. The method exploits the particular structure of the MPC problem and devises customized fixes in the primal barrier method. Thus, this algorithm is suitable for applications with fast dynamics.

In Section 4, we formulate the dynamics of a simplified vehicle and sensors as well as the extended Kalman filter. We then provide the simulation of 2D navigation using a combination of suboptimal schemes with given missions consisting of waypoints and bounding box constraints for state and control from the motion planner.

Lastly, in Section 5, we conclude the report with discussion on the current approach to solve the problem of navigation with imperfect state information and suggest some directions for further investigation.
2. Survey of suboptimal control schemes

We consider common settings in the basic dynamic programming (DP) framework. For each suboptimal control scheme, if there are different technical assumptions to obtain some desirable properties, we will explicitly mention those assumptions.

2.1. Limited look ahead policies and rollout algorithm

We consider a discrete-time dynamic system:

\[ x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, 2... \tag{1} \]

where \( x_k \) belongs to some set \( X \), \( u_k \) takes value from a set \( U_k(x_k) \), \( w_k \) is a random disturbance, and \( f_k \) is a given function. Disturbance \( w_k \) may depend on current \( x_k, u_k \) but not prior disturbances. The cost at \( k \)th time index is intractable. The idea of one-step limited lookahead policy is to replace the true cost \( J_k(x_k) \) with some approximation \( \bar{J}_k(x_k) \) and we set the value \( \bar{u}_k \) that attains the above minimization.

In the perfect state information case with horizon \( N \) with a terminal cost \( g_N(x_N) \), we choose the control \( u_k = \mu_k(x_k) \in U_k(x_k) \), and \( \pi = \{ \mu_0, \mu_1, ..., \mu_{N-1} \} \) forms policy laws for \( N \) time periods. The cost-to-go of policy \( \pi \) from time \( k \) onwards and its corresponding optimal cost are denoted by

\[ J_k^\pi(x_k) = E \left\{ g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, \mu_i(x_i), w_i) \right\}, \tag{2} \]

\[ J_k(x_k) = \min_{\pi \in \Pi} J_k^\pi(x_k). \tag{3} \]

The recursion equation of DP algorithm is:

\[ J_k(x_k) = \min_{u_k \in U_k(x_k)} E \left\{ g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right\}, \tag{4} \]

and we set the value \( \mu_k(x_k) \) to be \( u_k \) that attains the above minimization.

For problem with complex set \( X \) and \( U_k(x_k) \), solving Equation (4) for every possible state is intractable. The idea of one-step limited lookahead policy is to replace the true cost \( J_{k+1}(f_k(x_k, u_k, w_k)) \) by some approximation \( \bar{J}_{k+1}(f_k(x_k, u_k, w_k)) \) and set \( \bar{J}_N(x_N) = J_N(x_N) \) in the minimization of the right hand side in Equation (4):

\[ \min_{u_k \in U_k(x_k)} E \left\{ g_k(x_k, u_k, w_k) + \bar{J}_{k+1}(f_k(x_k, u_k, w_k)) \right\}. \tag{5} \]

Note that in the above calculation, at time index \( k \), we assume that we can estimate the cost at time index \( k+1 \) without using the DP recursion to compute the true cost. In other words, the limited look ahead policy truncates the time horizon to one step in this case. Furthermore, the constraint set \( U_k \) can also be replaced by \( \bar{U}_k \subset U_k \). The attained policy \( \bar{\pi} = \{ \bar{\mu}_0, \bar{\mu}_1, ..., \bar{\mu}_{N-1} \} \) is therefore called one-step lookahead policy.

Depending on how we select and compute the function \( \bar{J}_{k+1} \) in the above equation, we have broad suboptimal schemes. In this review, we focus on the implicit cost-to-go approximation where the cost of \( \bar{J}_{k+1} \) at \( f_k(x_k, u_k, w_k) \) is computed online. This category includes rollout, OLFC and MPC as we will review below.

The main result on error bounds of one-step lookahead policy is stated without proof in the following proposition.
Problems to perfect state information with new state is the conditional distribution and previous control can be satisfied in many special cases such as rollout algorithm.

2.2. Open loop feedback control

In OLFC scheme, we consider problems of imperfect state information. We will argue that OLFC is a special case of rollout algorithm. Control \( u_k \) at stage \( k \) is determined based on the information vector:

\[
I_k = (u_0, z_1, u_1, ..., u_{k-1}, z_k),
\]

where at all \( k \), \( z_k \) is an observation at stage \( k \). Observation \( z_k \) may depend on current state \( x_k \) and previous control \( u_{k-1} \). After \( u_k \) is chosen, a cost \( g_k(x_k, u_k, w_k) \) is incurred. We can reduce the problem to perfect state information with new state is the conditional distribution \( P_{x_k|I_k} \).

Since it would be intractable to carry out DP algorithm if we consider extra observations in the future, the idea of OLFC is to ignore the availability of extra information and use only current information \( I_k \) to determine \( u_k \) as follows:

**Proposition 1.** Assume that for all \( x_k \) and \( k \), we have

\[
\hat{J}_k(x_k) = \min_{u_k \in U_k(x_k)} E\left\{ g_k(x_k, u_k, w_k) + \hat{J}_{k+1}(f_k(x_k, u_k, w_k)) \right\} \leq \tilde{J}_k(x_k).
\]

Then the cost-to-go functions \( \tilde{J}_k \) of a one-step lookahead policy that uses \( \tilde{J}_k \) and \( U_k(x_k) \subset U_k(x_k) \) satisfy for all \( x_k \) and \( k \):

\[
\tilde{J}_k(x_k) \leq \hat{J}_k(x_k) \leq \tilde{J}_k(x_k).
\]

Equation (7) thus claims that the cost of the one-step lookahead is no worse than estimated cost. In practice, the one-step lookahead policy makes much improvement from its estimated counterpart. Fortunately, such an assumption can be satisfied in many special cases such as rollout algorithm.

Rollout algorithm uses the the cost-to-go of a suboptimal/heuristic policy, called a base policy, as values of \( \tilde{J}_k \). In particular, let a base policy of the rollout algorithm be \( \pi^b = \{\mu^b_0, \mu^b_1, ..., \mu^b_{N-1}\} \), and suppose that \( \mu^b_k(x_k) \in U_k(x_k) \) for all \( k \), we defined \( \tilde{J}_k(x_k) = J^b_k(x_k) \). We then can verify:

\[
\begin{align*}
\tilde{J}_k(x_k) &= E\left\{ g_k(x_k, \mu^b_k(x_k), w_k) + \hat{J}_{k+1}(f_k(x_k, \mu^b_k(x_k), w_k)) \right\} \\
&\geq \min_{u_k \in U_k(x_k)} E\left\{ g_k(x_k, u_k, w_k) + \hat{J}_{k+1}(f_k(x_k, u_k, w_k)) \right\},
\end{align*}
\]

by the definition of the minimization operator. Thus, the assumption (6) holds in rollout algorithm. Furthermore, we can notice that the obtained rollout policy \( \tilde{\pi} \) is indeed the improved policy obtained by a single policy improvement step from the base policy. It turns out that there is a close connection between OLFC, MPC and rollout algorithm in the sense that OLFC, MPC can be viewed as rollout algorithms as we will discuss in the next subsections.

In very special cases, \( \hat{J}_k(x_k) \) can be expressed by a closed-form expression. In most cases, closed-form expressions are not ready, and we can use approximate structure, which has rich connections with machine learning methods such as neural networks, kernel-based learning, and Gaussian regression, to compute \( \tilde{J}_k(x_k) \) offline. The other approach is to compute \( \hat{J}_k(x_k) \) by solving an online optimization problem. Alternatively, when the system is deterministic, we can evaluate exactly \( \hat{J}_k(x_k) \) using a single online simulation. When the system is stochastic, we can evaluate approximately \( \hat{J}_k(x_k) \) using multiple online simulations.

**2.2. Open loop feedback control**

In OLFC scheme, we consider problems of imperfect state information. We will argue that OLFC is a special case of rollout algorithm. Control \( u_k \) at stage \( k \) is determined based on the information vector:

\[
I_k = (u_0, z_1, u_1, ..., u_{k-1}, z_k),
\]

where at all \( k \), \( z_k \) is an observation at stage \( k \). Observation \( z_k \) may depend on current state \( x_k \) and previous control \( u_{k-1} \). After \( u_k \) is chosen, a cost \( g_k(x_k, u_k, w_k) \) is incurred. We can reduce the problem to perfect state information with new state is the conditional distribution \( P_{x_k|I_k} \).

Since it would be intractable to carry out DP algorithm if we consider extra observations in the future, the idea of OLFC is to ignore the availability of extra information and use only current information \( I_k \) to determine \( u_k \) as follows:
1. At time index $k$, information vector $I_k$ is collected and the conditional distribution $P_{x_k|I_k}$ is computed.

2. Find a control sequence $\{\bar{u}_k, \bar{u}_{k+1}, \ldots, \bar{u}_{N-1}\}$ that minimizes the open-loop problem:

$$J_k^* = \min_{u_k, \ldots, u_{N-1}} E \left\{ g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i, w_i) | I_k \right\},$$

subject to constraints:

$$x_{i+1} = f_i(x_i, u_i, w_i) \in X, \quad u_i \in U, \quad i = k, k+1, \ldots, N-1.$$  

3. Apply the control input $\bar{u}_k$ and discard the rest of controls:

$$\bar{\mu}_k(I_k) = \bar{u}_k.$$  

The online optimization problem in Equations (9 - 10) does not take into account of future measurements. The obtained policy $\bar{\pi} = \{ \bar{\mu}_0, \bar{\mu}_1, \ldots, \bar{\mu}_{N-1} \}$ is called OLFC policy. We have the following proposition that ensures the cost improvement property of the OLFC scheme.

**Proposition 2.** Let $J_k^\bar{\pi}$ be the cost of OLFC policy from time period $k$ to $N$, and $J_k^*$ be the cost of the open-loop policy in Equations (9 - 10), we have:

$$J_k^\bar{\pi} \leq J_k^*.$$  

The proof from first principles for the above proposition can be found in (1). There is a simpler argument from the perspective of cost improvement of rollout algorithm. We can view the original problem as a full state information problem with new state $P_{x_{k+1}|I_{k+1}}$ at time index $k$. Thus, we can view the optimal open-loop policy at state $P_{x_{k+1}|I_{k+1}}$ as a base policy, and control $\bar{\mu}_k(I_k) = \bar{u}_k$ is obtained from rollout algorithm corresponding to this base policy. By Proposition 1, we can infer that the cost of the OLFC policy is better than the corresponding optimal open-loop policy.

### 2.3. Model predictive control

Model predictive control (MPC) was motivated to handle nonlinearities and constraints in the linear quadratic Gaussian framework. In this type of control, an optimal control computation is used in conjunction with a rolling horizon of length $m$ to obtain a suboptimal policy but stable closed-loop system. There are cases where MPC can be viewed as a type of rollout algorithms. First, if the problem is deterministic, there are correspondences between MPC and rollout algorithm properties. Second, if disturbances belong to bounded sets, MPC can be casted as a minimax problem, and similar correspondences can be achieved. Since for the problem of robot navigation, disturbances are unbounded, we will focus on the case of deterministic MPC in this review. A deterministic problem is often simplified from a stochastic problem using certainty equivalence as we will show in Section 3. Note that, since in MPC, we will solve a subproblem of shorter horizon $m$ for each time period, there would be not much difference between finite horizon and infinite horizon in the original problem. Therefore, we will consider the value of horizon $N$ to be very large up to $\infty$, and we can stop the control process when the state satisfies stopping conditions. Furthermore, in the following analysis, the system is assumed to be time-invariant.

We consider the deterministic time-invariant fully observable system. The state transition function is

$$x_{k+1} = f(x_k, u_k), \quad k = 0, 1, 2, \ldots$$

and the stage cost $g(x_k, u_k)$ is assumed to be nonnegative for all $x_k, u_k$. At each time period, state and control satisfy constraints $x_k \in X, u_k \in U(x_k)$. When the system is at the origin, it can stay
there at no cost with zero control. We would like to construct a stationary feedback control law \( \mu(x) \) that drives the state \( x_k \) from any initial state \( x_0 \in X \) to zero, and at the same time \( \mu(x_k) \) approaches zero. We require that the trajectory induced by the closed-loop system \( x_{k+1} = f(x_k, \mu(x_k)) \) satisfies state and control constraints.

To enforce the stability requirement, sum of stage costs over an infinite horizon must be finite. This condition is expressed as follows

\[
\sum_{k=0}^{\infty} g(x_k, \mu(x_k)) \leq \infty. \tag{14}
\]

This condition implies that \( x_k \to 0 \) and \( u_k \to 0 \) as \( g(x_k, u_k) \) is nonnegative. The following proposition provides the sufficient condition to satisfy this requirement.

**Proposition 3.** If the system can stay at zero with no cost and zero control, i.e. \( f(0,0) = 0 \) and \( g(0,0) = 0 \), and there exists a positive integer \( m \) such that for every initial state \( x_0 \in X \), there is a sequence of controls \( u_k, k = 0, 1, 2, \ldots, m-1 \) that drives \( x_m \) to zero, while keeping \( x_1, \ldots, x_{m-1} \in X \)
and satisfying \( u_0 \in U(x_0), \ldots, u_{m-1} \in U(x_{m-1}) \), then there exists a policy \( \tilde{\mu} \) that satisfies the condition in Equation (14). In particular, this control policy \( \tilde{\mu} \), called MPC policy, is constructed as follows:

1. At each stage \( k \), we solve an \( m \)-stage deterministic optimal control with the same stage cost

\[
\tilde{J}(x_k) = \min_{u_k, u_{k+1}, \ldots, u_{k+m-1}} \sum_{i=k}^{k+m-1} g(x_i, u_i), \tag{15}
\]

subject to

\[
x_i \in X, \quad u_i \in U(x_i), \quad i = k, k+1, \ldots, k+m-1, \tag{16}
\]

and the terminal constraint \( x_{k+m} = 0 \).

2. Let \( \{ \tilde{u}_k, \tilde{u}_{k+1}, \ldots, \tilde{u}_{k+m-1} \} \) attains the optimal value of the above optimization problem, MPC applies at stage \( k \) the control \( \tilde{u}_k \) and ignores other controls in this sequence:

\[
\tilde{\mu}(x_k) = \tilde{u}_k. \tag{17}
\]

**Proof of Proposition 3.** Using the MPC algorithm as described in the proposition, let \( x_0, u_0, x_1, u_1, \ldots \) be the state and control sequence generated in this control process:

\[
u_k = \tilde{\mu}(x_k), \quad x_{k+1} = f(x_k, \tilde{\mu}(x_k)), \quad k = 0, 1, 2, 3, \ldots \tag{18}
\]

As defined in the proposition, \( \tilde{J}(x) \) is the optimal cost of the \( m \)-stage problem from state \( x \). Let us define a similar optimal cost \( \tilde{J}(x) \) of the \( (m-1) \)-stage problem from state \( x \):

\[
\tilde{J}(x) = \min_{u_0, u_1, \ldots, u_{m-2}} \sum_{i=0}^{m-2} g(x_i, u_i), \tag{19}
\]

subject to

\[
x_i \in X, \quad u_i \in U(x_i), \quad i = 0, 1, \ldots, m-2, \tag{20}
\]

and the terminal constraint \( x_{m-1} = 0 \). Starting from the same state \( x \), having less one stage to drive the state to 0 does not decrease the accumulated cost, thus for all \( x \in X \):

\[
\tilde{J}(x) \leq \tilde{J}(x). \tag{21}
\]
From the definition of $\hat{J}$ and $\tilde{J}$ and the above inequality, it can be seen that

$$\hat{J}(x_k) = \min_{u \in \mathcal{U}(x_k)} \left[ g(x_k, u) + \tilde{J}(f(x_k, u)) \right] = g(x_k, u_k) + \tilde{J}(f(x_k, u_k)) \leq \tilde{J}(x_k). \quad (22)$$

Using the inequality in (21) again at state $f(x_k, u_k)$, we obtain

$$\hat{J}(x_k) = g(x_k, u_k) + \tilde{J}(f(x_k, u_k)) \geq g(x_k, u_k) + \hat{J}(f(x_k, u_k)). \quad (23)$$

Applying the preceding inequality for all $k$ in $[0, K]$, we have

$$\hat{J}(x_{K+1}) + \sum_{k=0}^{K} g(x_k, u_k) \leq \hat{J}(x_0). \quad (24)$$

Since $\hat{J}(x_{K+1})$ is nonnegative as stage cost is nonnegative, taking $K \to \infty$, we obtain the stability condition

$$\sum_{k=0}^{\infty} g(x_k, u_k) \leq \hat{J}(x_0) \leq \infty. \quad (25)$$

More importantly, by comparing the inequality in (22) with the assumption in Proposition 1, we can see that $\hat{J}$ is the implicit one-step lookahead function used by the MPC scheme. Furthermore, we can view the policy that drives state to 0 after $m - 1$ steps while maintaining state and control in feasible regions as a base policy, and thus MPC can be viewed as a rollout algorithm. This is an important conceptual connection between the stability property of MPC scheme and cost improvement property of rollout algorithms.

It is worth noticing that the terminal condition to have state at zero after $m$ stages in the subproblem at each time period can be relaxed. One possible variant is to have a large penalty for not being at zero after $m$ stages, and we will use this variant in Section 4.

3. Fast and online MPC solver

So far, we have seen that at each time period in MPC, we have to solve a constrained optimization problem. In general, obtaining a closed-form solution for this optimization is rare. Thus, we retreat to using numerical solution. In this section, we present a method to obtain online numerical solution quickly. During this discussion, we keep notations consistent with notations in (4). We assume the dynamics is linear and time-invariant:

$$x_{t+1} = Ax_t + Bu_t + w_t, \quad t = 0, 1, 2, \ldots \tag{26}$$

where $x_t \in \mathbb{R}^n, u_t \in \mathbb{R}^m, w_t \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}$. Disturbance $w_t$ are independent and identical with mean $\bar{w} = E\{w_t\}$. In (4), the authors considered a quadratic stage cost:

$$s(x_t, u_t) = \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix}, \quad (27)$$

and the following average cost objective:

$$J = \lim_{T \to \infty} \frac{1}{T} E \left\{ \sum_{t=0}^{T-1} s(x_t, u_t) \right\}. \quad (28)$$
In these equations, $Q = Q^T \in \mathbb{R}^{n \times n}$, $S \in \mathbb{R}^{n \times m}$, $R = R^T \in \mathbb{R}^{m \times m}$, and we assume $\begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \succeq 0$. There are also constraints imposed on state and control

$$F_1 x_t + F_2 u_t \leq f,$$

where $F_1 \in \mathbb{R}^{lxn}$, $F_2 \in \mathbb{R}^{lxm}$, and $f \in \mathbb{R}^l$. The stochastic problem is to find a control policy satisfies the state and control in Inequality (29) and minimize the objective function $J$ in Equation (28).

Before continuing with defining a subproblem at each time period, we briefly mention the use of a suboptimal scheme called certainty equivalent control (CEC) that converts a stochastic problem to a deterministic problem.

### 3.1. Certainty equivalent control

The principle of certainty equivalence certifies that the result of a stochastic optimization is the same as the result of the corresponding deterministic problem where disturbances are replaced by their means. The certainty equivalent controller (CEC) is a suboptimal control scheme that is motivated by non-constrained linear quadratic control theory where the certainty equivalence principle holds. However, this principle does not hold for a general stochastic problem.

Under CEC, for a general constrained problem, at each stage, the control is expected to be close to the optimal value if uncertain quantities are fixed at some typical value such as means. The advantage of CEC is that it replaces the normal DP algorithm in Equation (4) with a simplified deterministic problem that demands much less computation.

The disadvantage of CEC is that an obtained control policy can be strictly suboptimal. The cost improvement property of CEC policy holds, for example, when stage costs and final cost are concave with respect to state variable. When stage costs and terminal cost are convex, the cost improvement property does not hold. In fact, there are examples showing that, for example, in a one-stage problem, the optimal open-loop controller and the OLFC are optimal, but CEC may be strictly suboptimal.

Despite this disadvantage, CEC is popular in practice. In the case of MPC, by substituting disturbances with their means, we can have a deterministic problem with cost improvement and stability properties as we analyzed in Section 2.3. However, this does not mean that cost improvement and stability properties are guaranteed in the original stochastic problem.

### 3.2. Exploiting MPC structure

Using the CEC scheme, we replace disturbances with their means, at each time period, we solve a subproblem with horizon length $T$:

$$\min_{u_t, u_{t+1}, \ldots, u_{t+T-1}} \frac{1}{T} \sum_{\tau = t}^{t+T-1} s(x_{\tau}, u_{\tau})$$

subject to

$$F_1 x_{\tau} + F_2 u_{\tau} \leq f,$$

$$x_{\tau+1} = Ax_{\tau} + Bu_{\tau} + \bar{w},$$

$$0 = Ax_{t+T-1} + Bu_{t+T-1} + \bar{w},$$
where decision variables are $x_{t+1}, ..., x_{t+T-1}$, $u_t, ..., u_{t+T-1}$, and problem data are $x_t, A, B, Q, S, R, F_1, F_2, f, \bar{w}$. Note that, if we ignore the constant factor $\frac{1}{T}$, the subproblem is consistent with what we discussed in Section 2.3.

By rewriting the above optimization problem in a more compact form using matrices so that we can recognize the special structure of the problem. Let us consider a new control variable

$$z = (u_t, x_{t+1}, u_{t+1}, ..., x_{t+T-1}, u_{t+T-1}) \in \mathbb{R}^{Tm+(n-1)}.$$  \hspace{1cm} (34)

The problem is now expressed as a quadratic programming (QP)

$$\min z^T Hz + g^T z$$  \hspace{1cm} (35)

subject to

$$Pz \leq h, \quad Cz = b,$$  \hspace{1cm} (36)

where

$$H = \begin{bmatrix} R & 0 & 0 & ... & 0 & 0 \\ 0 & Q & S & ... & 0 & 0 \\ 0 & S^T & R & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & Q & S \\ 0 & 0 & 0 & ... & S^T & R \end{bmatrix}, \quad g = \begin{bmatrix} 2S^T x_t \\ 0 \\ 0 \end{bmatrix},$$  \hspace{1cm} (37)

$$P = \begin{bmatrix} F_1 & 0 & 0 & ... & 0 & 0 \\ 0 & F_1 & F_2 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & F_1 & F_2 \end{bmatrix}, \quad h = \begin{bmatrix} f - F_1 x_t \\ f \end{bmatrix},$$  \hspace{1cm} (38)

$$C = \begin{bmatrix} -B & I & 0 & 0 & ... & 0 & 0 \\ 0 & -A & -B & I & ... & 0 & 0 \\ 0 & 0 & 0 & -A & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & ... & I & 0 \\ 0 & 0 & 0 & 0 & ... & -A & -B \end{bmatrix}, \quad b = \begin{bmatrix} Ax_t + \bar{w} \\ \bar{w} \\ \bar{w} \end{bmatrix}.$$  \hspace{1cm} (39)

We can use generic optimization solvers such as SDPT3 (3) to solve the above QP exactly. However, since the purpose of solving this QP is to find a suboptimal solution to the original control problem, we can obtain approximate solution to the QP instead. In the following discussion, we present the primal barrier method to solve the QP and possible fixes that reduces the number of iterations to speed up the computation.

3.2.1. Primal barrier method The primal barrier method to solve the QP in (35 - 36) replaces the inequalities with a barrier term in the objective function of the following approximate convex problem

$$\min z^T Hz + g^T z + \kappa \phi(z)$$  \hspace{1cm} (40)
subject to
\[ Cz = b, \]
where \( \phi(z) = \sum_{i=0}^{iT} -\log(h_i - p_i^Tz) \), and \( p_i^T, ..., p_i^T \) are rows of \( P \). The parameter \( \kappa \) is a barrier coefficient. As \( \kappa \) approaches 0, the solution of this approximate problem converges to the solution of the QP.

In a basic primal barrier method, we solve a sequence of problems of the form (40 - 41) by Newton’s method starting from the previously computed points, with a decreasing sequence of values of \( \kappa \). During this process, \( \kappa \) is typically reduced by a factor of 10.

Alternatively, for a given value of \( \kappa \), we can solve the problem using infeasible start Newton method, in which the algorithm is initialized with a point \( z^0 \) that is strictly satisfy \( Pz^0 \leq h \), but need not satisfy \( Cz^0 = 0 \). In other words, \( z^0 \) may be infeasible. This approach will reveal a special structure of the MPC problem. We associate a dual variable \( v \in \mathbb{R}^{Tn} \) with the equality constraint \( Cz = b \). The optimality conditions for (40 - 41) are
\[
\begin{align*}
\quad r_d &= 2Hz + \kappa PTd + C^tv = 0, \\
\quad r_d &= Cz - b = 0,
\end{align*}
\]
where \( d_i = \frac{1}{h_i - p_i^Tz} \). The term \( \kappa PTd \) is the gradient of \( \kappa \phi(z) \). We now carry out a search until the values of \( r_d, r_p \) are small enough. In each search step, values of \( z \) and \( v \) are updated with \( z := z + s\delta z \), and \( v := v + s\delta v \), \( s \) is a step size, by solving the linear equations
\[
\begin{bmatrix}
2H + \kappa P^T \text{diag}(d)^2 P C^T & \delta z \\
C & 0
\end{bmatrix} = \begin{bmatrix}
-\delta v \\
r_d \end{bmatrix},
\]
We note that the above linear system has a block diagonal structure. Thus, by exploiting this structure, we can use block elimination to reduce the running time from \( O(T^3(2n + m)^3) \) to \( O(T(m + n)^3) \). This result means that the complexity of MPC is linear rather than cubic in the problem horizon at each time period. Further details of the infeasible start Newton and block elimination methods can be found in (4).

3.2.2. Fixed \( \kappa \) Since the solution of the induced QP needs not to be accurate, the authors of (4) proposed to use a fixed value of \( \kappa \) instead of reducing value of \( \kappa \) to zero as in a general QP solver. Furthermore, a fixed value of \( \kappa \) enables the efficiency of using warm start to compute a control in the next time period as we will discuss below.

3.2.3. Fixed iteration limit In addition to fixing \( \kappa \), another variant is to limit the number of iteration instead of waiting to obtained small values of \( r_d, r_p \). The maximum number of iteration \( K^{\text{max}} \) could be set between 3 and 10. Note that when the iteration stops due to this limitation, the solution \( z \) of the QP may not even feasible. In other words, the solution \( z \) may not satisfy the \( Cz = b \), i.e. the system dynamics; however, it does satisfies the current state and control constraints, i.e. \( F_1x_t + F_2u_t \leq f \).

3.2.4. Warm start The idea of warm start is to initialize the primal barrier method with the previous computed QP solution. In particular, let us suppose that at time period \( t - 1 \) the computed trajectory is
\[
\tilde{z} = (\tilde{u}_{t-1}, \tilde{x}_{t-1}, ..., \tilde{x}_{t+T-2}, \tilde{u}_{t+T-2}).
\]
We initialize the primal barrier method at time period \( t \) with
\[
z^{\text{init}} = (\tilde{u}_t, \tilde{x}_{t+1}, ..., \tilde{x}_{t+T-2}, \tilde{u}_{t+T-2}, 0, 0).
\]
As reported in (4), this customized solution using above techniques can compute a control action on the order of 100 times faster than other solvers, enabling MPC to be carried out at 200Hz for a reasonable large problem with 12 states, 3 controls, horizon of 30 time steps.
4. Using approximate dynamic programming to navigate a vehicle

In this section, we return to the problem of navigating in imperfect state information environments. We will present the formulation of the problem in the DP framework. We will keep the formulation as general as possible, and numerical example of parameters will be provided in Section 4.5. We also present how to use the extended Kalman estimator and a combination of CEC, OLFC, MPC to design a controller for the system.

4.1. System formulation

We consider a simplified vehicle with discrete time-invariant linear dynamics in 2D environment

\[ x_{k+1} = Ax_k + Bu_k + w_k, \quad k = 0, 1, 2, 3... \]

where \( x_k \in \mathbb{R}^n \), \( u_k \in \mathbb{R}^m \), and \( w_k \in \mathbb{R}^n \) are i.i.d. disturbances with multi-normal distribution. If the time is discretized into unit time interval \( \Delta \), we have

\[ A \in \mathbb{R}^{n \times n}, \quad B \in \mathbb{R}^{n \times m}, \quad w_k \sim N(0, \Delta \Omega^w), \quad \Omega^w \in \mathbb{R}^{n \times n} \succ 0. \]

The sensor dynamics of the vehicle is expressed as

\[ z_k = h(x_k) + v_k, \quad k = 0, 1, 2, 3... \]

where \( z_k \in \mathbb{R}^p \), and \( v_k \) are i.i.d. sensor’s noises with distribution \( N(0, \Omega^v) \), \( \Omega^v \in \mathbb{R}^{p \times p} \succ 0 \). The vehicle receives sensor measurements by sensing features in the surrounding environment. A possible sensor function \( h \) provides approximate distances from the location of the vehicle to detected features.

We assume that the motion planner is able to plan an optimal trajectory that provides the most informative sensor measurements for the vehicle. While traversing this path, the vehicle will receive valuable measurements to improve the estimate of its current position. Let us consider the most informative sensor measurements for the vehicle. While traversing this path, the vehicle will receive sensor measurements.

We plan to reach the waypoint

\[ V \]

initial position and visits each waypoint in that order. For each waypoint \( d_i \), the vehicle has an initial belief \( P_{x_0|I_0} = N(\hat{x}_0, \Lambda_0) \), and there are constraints on the minimum and maximum values of state and control that the vehicle can take to visit the waypoint

\[ x_{\text{min}} \leq x_k \leq x_{\text{max}}, \quad k = 0, 1, 2,... \]

\[ u_{\text{min}} \leq u_k \leq u_{\text{max}}, \quad k = 0, 1, 2,... \]

We plan to reach the waypoint \( d_i \) in \( N \) time steps, where \( N \) is very large, up to \( \infty \), so that it is possible for the vehicle to travel with the above constraint on control. We also assume that there is at least one proper policy to visit a small neighborhood around the waypoint \( d_i \). Let \( x_d \) be a state that enforces the condition that the vehicle is at \( d_i \). The termination cost, and stage cost are

\[ g_d(x_N) = (x_N - x_d)^T Q_d(x_N - x_d), \quad Q_d \succ 0, \]

\[ g(x_k, u_k, w_k) = u_k^T R u_k, \quad R \succeq 0. \]

Note that we only impose penalty on control in the above equations. We would like to minimize the following objective function

\[ J^*(I_0) = \min_{\pi \in \Pi} \lim_{N \to \infty} E \left\{ g_d(x_N) + \sum_{i=k}^{N-1} g(x_i, \mu(x_i), w_i) | I_0 \right\}, \]

subject to

\[ x_{\text{min}} \leq x_k \leq x_{\text{max}}, \quad k = 0, 1, 2,... \]

\[ u_{\text{min}} \leq \mu(x_k) \leq u_{\text{max}}, \quad k = 0, 1, 2,... \]
4.2. Estimator design

Since the sensor function is possibly nonlinear, we use the extended Kalman filter to update the location estimate \( \hat{N}(\hat{x}_k, \Lambda_k) \) at time period \( k \), where \( \hat{x}_k = E[x_k|I_k] \), and \( \Lambda_k = Cov[x_k|I_k] \). Given the current estimate \( \hat{N}(\hat{x}_k, \Lambda_k) \), we would like to find \( \hat{N}(\hat{x}_{k+1}, \Lambda_{k+1}) \) after executing control \( u_k \) and receive measurement \( z_{k+1} \).

- **Predictive step:**
  \[
  \hat{x}_{k+1} = A\hat{x}_k + Bu_k, \\
  \Lambda_{k+1} = A\Lambda_k A^T + \Delta\Omega_w.
  \]

- **Update step:**
  \[
  H_{k+1} = \frac{dh}{dx}|_{\hat{x}_{k+1}}, \\
  K_k = \Lambda_{k+1}H_{k+1}^T(H_{k+1}\Lambda_{k+1}H_{k+1}^T + \Omega^u)^{-1}, \\
  \hat{x}_{k+1} = \hat{x}_{k+1} + K_k(z_{k+1} - h(\hat{x}_{k+1})), \\
  \Lambda_{k+1} = \Lambda_{k+1} - K_kH_{k+1}\Lambda_{k+1}.
  \]

4.3. Controller design

We design a suboptimal controller that uses a combination of OLFC, CEC and MPC. First, as in OLFC, at each time period, we use only current information \( I_k \) and ignore future measurements. Second, as in CEC, we substitute system disturbances by their means. Third, as in MPC, a control at a time period is obtained by solving a subproblem with a fixed horizon length \( T \). We use a variant of MPC by adding a termination cost. In particular, we compute controls as follows:

1. At time index \( k \), information vector \( I_k \) is collected by adding previous control \( u_{k-1} \) and current observation \( z_k \). The conditional EKF estimate is computed \( \hat{N}(\hat{x}_k, \Lambda_k) \) as in Section 4.2. We use the mean \( \hat{x}_k \) as the true state.

2. Replace disturbances \( w_k, w_{k+1}, \ldots \) with their mean \( \bar{w} = 0 \) and initialize \( x_k := \hat{x}_k \). We then find a control sequence \( \{\bar{u}_k, \bar{u}_{k+1}, \ldots, \bar{u}_{k+T-1}\} \) that minimizes:
  \[
  J_k^* = \min_{u_k, \ldots, u_{N+T-1}} \left\{ g_d(x_{k+T}) + \sum_{i=k}^{k+T-1} g_i(x_i, u_i, \bar{w}) \right\},
  \]
  subject to constraints:
  \[
  x_{i+1} = Ax_i + Bu_i + \bar{w}, \quad i = k, k + 1, \ldots, k + T - 1 \\
  x_{\min} \leq x_i \leq x_{\max}, \quad i = k, k + 1, \ldots, k + T \\
  u_{\min} \leq u_i \leq u_{\max}, \quad i = k, k + 1, \ldots, k + T
  \]

3. Apply the control input \( \bar{u}_k \) and discard the rest of controls:
  \[
  \bar{\mu}(I_k) = \bar{u}_k.
  \]

We can see that the above controller design resembles its counterparts in Sections 2.2 and 2.3. To compute controls, we use the fast solver as described in Section 3 to solve the problem in Equations (63-66).

4.4. Executing navigation mission

We now present the algorithm that navigates the vehicle following a set of waypoints from the motion planner as shown in Algorithm 1.
Algorithm 1 Navigation algorithm

1: procedure NAVIGATE($\hat{x}_0, \Lambda_0, WP = \{d_0, ..., d_V\}, \text{Constraint} - \text{Set}$)
2: for $d_i \leftarrow d_0, ..., d_V$ do
3: $x_d$ encapsulates the waypoint $d_i$ to visit
4: extract $x_{\text{min}}, x_{\text{max}}, u_{\text{min}}, u_{\text{max}}$ from $\text{Constraint} - \text{Set}$ to reach $d_i$
5: $k \leftarrow 0$
6: while $g_d(\hat{x}_k)$ is not small enough do
7: $u_k \leftarrow$ Solve a subproblem defined by Equations (63-66)
8: Execute $u_k$, Observe measurement $z_{k+1}$
9: Compute new estimate $N(\hat{x}_{k+1}, \Lambda_{k+1})$
10: $k \leftarrow k + 1$
11: end while
12: Initialize $\hat{x}_0 \leftarrow \hat{x}_k, \Lambda_0 \leftarrow \Lambda_k$ to visit the next waypoint
13: end for
14: end procedure

Table 1 An example of settings in an environment

<table>
<thead>
<tr>
<th>Features $\in \mathbb{R}^2$</th>
<th>Radius</th>
<th>Obstacle $(x_{bl} \in \mathbb{R}^2, x_{tr} \in \mathbb{R}^2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(1, 8)$</td>
<td>1.5</td>
<td>$(4, 6), (6, 10)$</td>
</tr>
<tr>
<td>$(3.2, 5.1)$</td>
<td>1.7</td>
<td>$(4, 0), (6, 4)$</td>
</tr>
<tr>
<td>$(7, 5)$</td>
<td>1</td>
<td>$(1.7), (2.8)$</td>
</tr>
<tr>
<td>$(8, 7)$</td>
<td>1.5</td>
<td>$(2.4), (3.5)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$(8, 6), (9, 9)$</td>
</tr>
</tbody>
</table>

4.5. Simulation results

In this section, we provide simulation results to demonstrate the performance of the algorithm. The state of system $x_k \in \mathbb{R}^2$ represents vehicle position, and control $u_k \in \mathbb{R}^2$ is velocity. We consider the following numerical data for the system described in Section 4:

$$\Delta = 0.1s, \quad A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Delta, \quad \Omega_w = \begin{bmatrix} 0.005 & 0 \\ 0 & 0.005 \end{bmatrix}, \quad \omega = 0.03,$$ (68)

$$z_k \in \mathbb{R}, \quad h_i(x) = \| x - \text{pos}_{\text{feature}_i} \|, \quad \Omega^v = 0.03,$$ (69)

where $\text{pos}_{\text{feature}_i}$ is the position of some detected $i^{th}$ feature around the vehicle. The vehicle only detects a feature if the vehicle is within a range of visibility of the feature. We assume that when the vehicle detects a feature, it can identify the feature position and thus $h_i$. The parameters to define cost and MPC horizon are

$$Q_d = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad T = 10.$$ (70)

Table 1 lists all features with their radius of visibility and obstacles in the environment. Obstacles have rectangular shape defined by bottom left $x_{bl}$ and top right $x_{tr}$ coordinates. Table 2 shows a mission consisting of four waypoints and the corresponding constraints on position and velocity of the vehicle. The initial belief of the vehicle has mean and covariance

$$\hat{x}_0 = [0.2, 9.3]^T, \quad \Lambda_0 = \begin{bmatrix} 0.03 & 0 \\ 0 & 0.03 \end{bmatrix}.$$ (71)
Table 2  Waypoints and constraints

<table>
<thead>
<tr>
<th>Waypoint</th>
<th>$x_{min}$</th>
<th>$x_{max}$</th>
<th>$u_{min}$</th>
<th>$u_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2.3, 8.3)</td>
<td>(0, 8)</td>
<td>(3.8, 10)</td>
<td>-1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>(3.1, 5.5)</td>
<td>(2, 5.2)</td>
<td>(4, 10)</td>
<td>-1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>(7.5, 8.5)</td>
<td>(3, 4)</td>
<td>(10, 6)</td>
<td>-1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Figure 2  Trajectory of the mission visiting waypoints to reach the destination.

Figure 3  Control profile during the trajectory.

Figure 2 shows the executed trajectory to achieve the mission defined in Table 2. In this figure, features are marked by red dots, and the ranges of visibility are represented by dash circles. The
The real trajectory is plotted in a green curve. The estimated trajectory is represented in a pink curve, and the covariance are represented by ellipses around this estimated trajectory. As we can see, when the vehicle was out of ranges of visibility, the uncertainty about its position increased, and when the vehicle received useful measurements from sensor, it improved the estimate of its position. Using good position estimates, the vehicle was able to navigate without colliding with obstacles. In Figure 3, we can verify that the vehicle traveled in feasible state domain using feasible controls. To compute a control, the simulation took 0.2 ms, which enables controls to be updated at 5000 Hz. In this mission, we compute the expected cost to complete the mission by averaging multiple simulations, and the approximated value for the expected cost in this case is 83.5.

We also tested the system when system dynamics has substantial noises, for example $\Omega^w = \begin{bmatrix} 0.03 & 0 \\ 0 & 0.03 \end{bmatrix}$. As shown in Figure 4, the trajectory can become very poor, and the vehicle still risks colliding with obstacles. In such cases, emergency recovery module on the vehicle is still useful to save the vehicle from being crashed. However, under mild environments, when vehicle can maintain good estimates of its position, possibly by carrying more sensors, the proposed controller in this report performs well with direct consideration of contraints in the MPC scheme.

![Figure 4](image)

**Figure 4** Trajectory of the mission when disturbances are substantial.

### 5. Conclusions

In conclusion, we have reviewed a wide range of suboptimal control schemes in approximate dynamic programming. These schemes, one-steps lookahead, rollout, OLFC and MPC, start with a suboptimal or heuristic policy and then perform policy improvement step to get a better policy. Under different assumptions, these schemes have a nice cost improvement property. For MPC, this suboptimal control scheme considers constraints directly, and its stability property in certain circumstances has conceptual connection with the cost improvement property. We have also reviewed a fast MPC solver that exploits the special structure of the quadratic programming induced by the MPC scheme. This solver reduces as many iterations as possible to obtain a feasible control at each time period.
We then use a combination of the OLFC, MPC, CEC and a good estimator to design a controller for navigating a vehicle in an imperfect state environment with explicit constraints on state and control. It is desirable to obtain the cost improvement property for the proposed controller. However, since our cost is convex, CEC does not have the improvement property. Moreover, in our case, the mean of the estimate is not a sufficient statistics for the hidden state. Thus, a combination of these suboptimal schemes does not preserve the cost improvement property, and collisions may happen when disturbances are substantial. Nevertheless, in mild environments, when the vehicle is able to maintain good estimates of its position, the effect of CEC and mean estimate replacement are less unfavorable, the cost improvement property is relevant due to the MPC scheme.

We conclude with a brief discussion on the possible direction to extend this work to nonlinear systems with non-quadratic cost. In such cases, at each time period, by applying the OLFC and MPC, we can linearize the system dynamics, and approximate cost function with a quadratic function around a nominal trajectory. Furthermore, using the current position estimate, we can form a control constraint set such that, for example, we can guarantee that with 95%, the next position is still in feasible domains. We thus can compute a control after possible projections on control constraint sets. The main idea of this proposed approach is that we actively restrict possible controls to reduce the possibility of collision. In this situation, we possibly need to devise a new method to solve subproblems at each time period.

References