Proceedings of the ASME 2019
Dynamic Systems and Control Conference
DSCC2019
October 8-11, 2019, Park City, Utah, USA

DSCC2019-9208

MOTION PLANNING FOR INDUSTRIAL MOBILE ROBOTS WITH CLOSED-LOOP STABILITY ENHANCED PREDICTION

Jessica Leu
Mechanical Systems Control Lab
Department of Mechanical Engineering
University of California
Berkeley, CA 94720 USA
Email: jess.leu24@berkeley.edu

Masayoshi Tomizuka
Mechanical Systems Control Lab
Department of Mechanical Engineering
University of California
Berkeley, CA 94720 USA
Email: tomizuka@berkeley.edu

ABSTRACT

Real-time, safe, and stable motion planning in co-robot systems involving dynamic human robot interaction (HRI) remains challenging due to the time varying nature of the problem. One of the biggest challenges is to guarantee closed-loop stability of the planning algorithm in dynamic environments. Typically, this can be addressed if there exists a perfect predictor that precisely predicts the future motions of the obstacles. Unfortunately, a perfect predictor is not possible to achieve. In HRI environments in this paper, human workers and other robots are the obstacles to the ego robot. We discuss necessary conditions for the closed-loop stability of a planning problem using the framework of model predictive control (MPC). It is concluded that the predictor needs to be able to detect the obstacles' movement mode change within a time delay allowance and the MPC needs to have a sufficient prediction horizon and a proper cost function. These allow MPC to have an uncertainty tolerance for closed-loop stability, and still avoid collision when the obstacles' movement is not within the tolerance. Also, the closed-loop performance is investigated using a notion of M-convergence, which guarantees finite local convergence (at least M steps ahead) of the open-loop trajectories toward the closed-loop trajectory. With this notion, we verify the performance of the proposed MPC with stability enhanced prediction through simulations and experiments. With the proposed method, the robot can better deal with dynamic environments and the closed-loop cost is reduced.

INTRODUCTION

The development of intelligent industrial robots is craving for a safe, reactive, and stable motion planning in dynamic environments. For example, an automated guided vehicle (AGV) in co-robot systems involving dynamic human robot interaction (HRI), needs to decide in real time how to bypass multiple human workers and a set of obstacles in the environment in order to approach its target efficiently.

In literature, motion planning algorithms usually fall into three categories: graph-search based algorithm, sampling-based algorithm, and optimization-based algorithm. Among these algorithms, this paper focuses on model predictive control (MPC) [1], an optimization-based algorithm. MPC has been widely adopted both in academic research and in industrial applications. The fact that MPC observes the environment every time before solving the optimization problem allows the system to adjust and re-plan according to the changes of the environment. Its ability to handle input and state constraints also makes it popular in addressing motion planning problems. A typical MPC approach involves a sequence of receding horizon optimization. The sequential nature of such MPC formulation introduces a closed-loop with respect to the performance index for optimization. The closed-loop stability problem is a serious issue when the environment, including motions of obstacles, is dynamic. If the MPC problem is convex and has time-invariant constraints, the stability of the closed loop system can be guaranteed by

modifying prediction horizon, adding terminal cost, adding terminal equality constraints, or using terminal set constraints instead [2]. These methods can guarantee that the calculated commands and the resulting states are bounded in the presence of bounded disturbances. One common application of motion planning MPC problems is autonomous vehicles [3], where stability can be guaranteed by setting stability boundary for the control system, i.e., stability is quantified at several vehicle speeds. Another common application is industrial robots [4], where stability can be guaranteed by using Lyapunov-like functions and the terminal-state controller when the state space is convex.

However, two natures of motion planning in dynamic environments have contaminated the stability properties of MPC. First, the existence of obstacles in the environment introduces non-convex state constraints, which result in time varying and non-convex MPC problems. This makes closed-loop stability of MPC hard to analyze. The second problem is that the constraints in the optimization problem change continuously through out time due to the changes in the environment. The fact that optimization problems are sensitive to the constraints makes the accuracy of the prediction extremely crucial because the prediction determines the state constraints of the problem. Commonly seen assumptions in MPC, such as obstacles are moving in constant velocity or acceleration, may not suit the environment well and cause the open-loop trajectories at each time step to vary largely. This might result in dynamically unreasonable closed-loop trajectories. Dynamically unreasonable closed-loop trajectory will cause the robot to execute violent or zigzagging movements that are harmful to the robot's motors and also scare other workers in the environment. Fortunately, with some assumptions, it is possible to formulate the uncertainties in the MPC problem in a proper way. Some works have been focusing on dealing with uncertainties in MPC problems [5, 6]. In these papers, systems under persistent disturbance can be controlled by robust MPC. However, a more common scenario in motion planning problems is to have uncertainties in the environment, e.g., environment that has obstacles moving at varying speed. Therefore, a probabilistic model of the obstacles' future movement is needed. In [7], stochastic MPC deals with probability constraints to handle uncertainties in the environment. To achieve even higher efficiency, another approach is to directly predict the object's future motion. In [8,9], the authors focus on industrial HRI environment, where the uncertainties in the environment are usually caused by human workers. It is clear that although the workers do not explicitly reveal their intention, it is still possible to predict their future motion based on past observations [10, 11]. In [12], a Bayesian filter is used for motion prediction given observations in the past and the state transition model.

The contributions of this paper are to discuss the conditions to enable closed-loop stability and to provide an example of a motion planning method with a closed-loop stability enhanced predictor in a MPC framework. Conditions for the predictor and the MPC are identified and can guarantee closed-loop stability when the environment satisfies some assumptions, and collision avoidance even though the assumptions are not satisfied. With the conditions, we proposed an example of MPC with stability enhanced prediction, which utilizes the past and current observations of the environment to predict a worker's intention and construct the state space for the optimization problem at each MPC time step. It is assured that the proposed method is stable theoretically in the sense of Lyapunov in a set of common factory scenarios that satisfies the assumptions. The proposed method can also better deal with environment uncertainties which addresses the limitation of MPC without stability enhanced prediction, i.e., MPC that only uses current observations and always assumes constant velocity or constant acceleration. To show that owing to a better knowledge of the dynamic environment, the proposed method performs better than MPC without stability enhanced prediction, a new notion of open-loop prediction convergence property called M-convergence is used, which serves as another indicator of the closed-loop performance. The open-loop and closed-loop cost of the proposed method are also compared with the MPC without stability enhanced prediction to verify the improvement. Simulation studies are performed to test the performance and finally, experiments are carried to verify the proposed method (video is publicly available at `jessicaleu24.github.io/DSCC2019.html`).

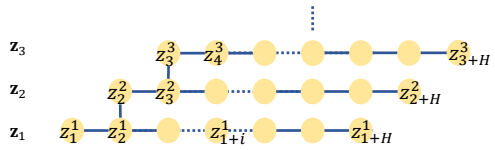


FIGURE 1: The execution structure of MPC.

PROBLEM FORMULATION

Traditional Non-convex MPC and Notations

In MPC (FIGURE 1), at each time step t, a future trajectory will be planned by solving an optimization problem. This trajectory is denoted as $\mathbf{z}_k := [z_k, z_{k+1}, z_{k+2}, \cdots, z_{k+H}]$ where the state, z_k, is called the k^{th} action location. H is the prediction horizon. In this paper, $z_k = [x(k), y(k)]^\intercal$ contains the x and y coordinate of the robot's location in 2-dimensional Cartesian space at time step $t = k$. Note that z_k corresponds to the current position, denoted as $z_0(t = k) = (x, y)|_{t=k}$. The trajectory will then be executed and the robot will go to z_{k+1}, the $k + 1^{th}$ action location. The sampling time between two time steps is a constant, denoted as Δt. After reaching the next action location, the robot will again solve the optimization problem and plan a new trajectory and repeat the process. To avoid confusion, the current time step can

also be marked as superscript in some cases , e.g., z^k_{k+1} means the planned action location z_{k+1} at time step $t = k$.

At time step $t = k$, given the current state, the following optimization needs to be solved to obtain $\mathbf{z}_k$,

$$\min_{\mathbf{z}_k} J(\mathbf{z}_k), \tag{1}$$

$$s.t. \quad \mathbf{z}_k \in \Gamma^k_m(z_0(k), \mathbf{O}^k_m). \tag{2}$$

This optimization is performed at every MPC time step k as time evolves. There are two assumptions in this formulation:

Assumption 1 (Cost). *The cost function, containing both the stage cost and the terminal cost, is convex, regular, and time-invariant, and has the following form*

$$J(\mathbf{z}_k) = C_1\|\mathbf{D}\mathbf{z}_k - \mathbf{z}_{ref}\|_2^2 + C_2\|\mathbf{V}\mathbf{z}_k - \mathbf{v}_{ref}\|_2^2 + C_3\|\mathbf{A}\mathbf{z}_k\|_2^2. \tag{3}$$

Here, C_1 is the coefficient of the first term, which penalizes the robot's deviation from a reference line so that the robot output trajectory is not too irregular. Matrix $\mathbf{D}$ is a projection matrix that extract all y's from $\mathbf{z}_k$. C_2 is the coefficient of the second term, which penalizes the speed profile of the planned trajectory with regard to a constant speed so that the robot will be time efficient. $\mathbf{V}\mathbf{z}_k$ is the velocity vector. Here, the speed reference is set to be a constant speed going along the positive x-axis. C_3 is the coefficient of the third term, which penalizes the acceleration of the output trajectory so that the motion will be smooth. $\mathbf{A}\mathbf{z}_k$ is the acceleration vector.

Assumption 2 (Constraint). *The state constraint Γ is non-convex and its complement, $\mathbf{O}$, is a collection of disjoint convex sets.*

Based on the current observation, assuming that the obstacles are moving in constant speed, the obstacle region, $\mathbf{O}^k_m$, can be obtained. The set of state constraints, Γ^k_m, satisfies Assumption 2. The current state, $z_0(k)$, is measured and assigned to the first entry of $\mathbf{z}_k$. The final trajectory is $[z^k_k, z^{k+1}_{k+1}, \ldots]$, which is equivalent to $[z^{k-1}_k, z^k_{k+1}, \ldots]$ (FIGURE 1) if the tracking controller is perfect.

The Convex Feasible Set Algorithm

Because the problem we are solving has non-convex state constraints, the problem is a non-convex MPC problem. Here we solve the optimization problem at each MPC time step using the convex feasible set algorithm (CFS) [13], which iteratively solves a sequence of sub-problems of the original non-convex problem using convex constraints, e.g., convex feasible set. The CFS algorithm in the MPC structure uses the previous solution as a reference to construct the convex feasible set. The convex feasible set for a reference point z^r is computed as $\mathscr{F}(z^r) = \{z : A(z^r)z \le b(z^r)\}$ where $A(z^r)$ is a matrix and $b(z^r)$ is a column vector. Assuming perfect prediction, at time step $k+1$, the reference is set as $\mathbf{z}^r_{k+1} = [z^k_{k+1}, z^k_{k+2}, \ldots, z^k_{k+H}, z^*_{k+H+1}]$ where

$$\begin{aligned} z^*_{k+H+1} = \arg\min_{z_{k+H+1}} &\|z_{k+H+1}\|^2_Q + \|z^k_{k+H} - z_{k+H+1}\|^2_R \\ &+ \|z^k_{k+H-1} - 2z^k_{k+H} + z_{k+H+1}\|^2_P. \end{aligned} \tag{4}$$

If $z^*_{k+H+1} \in \Gamma$, then the optimal solution is $\mathbf{z}^o_{k+1} = \mathbf{z}^r_{k+1}$. If $z^*_{k+H+1} \notin \Gamma$, denote the feasible solution as

$$\begin{aligned} \bar{z}_{k+H+1} = \arg\min_{z_{k+H+1}\in\Gamma} &\|z_{k+H+1}\|^2_Q + \|z^k_{k+H} - z_{k+H+1}\|^2_R \\ &+ \|z^k_{k+H-1} - 2z^k_{k+H} + z_{k+H+1}\|^2_P. \end{aligned} \tag{5}$$

Moreover, denote
$\mathbf{z}^u_{k+1} := [z^k_{k+1}, z^k_{k+2}, \ldots, z^k_{k+H}, \bar{z}_{k+H+1}]$.

Because of the construction above, the optimal solution is $\mathbf{z}^o_{k+1}$ at step $k+1$ satisfies that

$$\mathbf{z}^o_{k+1} = \arg\min_{z_{k+i}\in\mathscr{F}(z^o_{k+i})} J(\mathbf{z}_{k+1}). \tag{6}$$

Note that $J(\mathbf{z}^r_{k+1}) \le J(\mathbf{z}^o_{k+1}) \le J(\mathbf{z}^u_{k+1})$, therefore, the cost of the trajectory is always bounded. The executed trajectory is from those $\mathbf{z}^o_{k+1}$ for different k.

Closed-loop Stability of MPC-based Planning Problems

As discussed previously, uncertainties in the environment are usually caused by human workers or other robots. Here, we identify the necessary conditions that enable closed-loop stability. First, the scenario where the worker's motion is predictable needs to be identified. Assuming that the worker's movements can be captured by a predictor with finite modes, $m \in \{m_1, m_2, \ldots, m_d, m_{up}\}$, where m_i, $i = 1, \ldots, d$ are the modes where the worker's movements are predictable and m_{up} captures all the other unpredictable movements. An assumption is made in the predictable modes:

Assumption 3 (Obstacle motion). *Assuming that the acceleration and velocity of the obstacle are bounded, the velocity, v_{obs}, of the obstacle during m_i mode is also bounded, i.e., $\|v_{obs}\| \le v^u_{m_i}$.*

Therefore, we can define the predictable mode as the following:

Definition 1 (Predictable mode). *During predictable mode m_i, there exists an obstacle region $\mathbf{O}^k_i$ that covers the obstacle*

movements for the coming T_{ci} time duration, and thus, the movements in this mode are predictable. Here, T_{ci} is called the time delay allowance.

With the Definition 1, we can define closed-loop stability of MPC in time varying environments when the obstacle's movement mode is predictable as the following:

Definition 2 (Closed-loop stability of MPC). *Let $J^*(k)$ be the optimal cost at time step k. MPC is stable in closed-loop if the optimal cost function is a Lyapunov function, i.e., $0 \leq J^*(k) \leq J^*(k-1)$ for all k during predictable mode m_i.*

Denote k_s to be the time when the obstacle movement switches in between different modes, and k_{ps} to be the time when the predictor detects that change. Denote T_d to be the time delay between k_s and k_{ps}, i.e., $T_d = k_{ps} - k_s$. With Assumption 3, the first condition for the overall system is:

Condition 1 (Condition for the predictor). *The time delay, T_d, needs to satisfy $T_d < T_{ci}$ for all $i = 1, \ldots, d$.*

This is important for safety reason especially when the worker is switching from predictable modes to unpredictable mode. This condition guarantees that the obstacle region can cover the worker's movements even though the predictor hasn't detected the mode change. Therefore, the motion planner can still plan for a solution that avoids collision. With the above assumptions and condition, we have the first main result:

Main result 1 (Convergence of the state constraints). *A series of obstacle region, $\mathbf{O}_i^k$, can be chosen by the predictor and $\mathbf{O}_i^k$ satisfies the relationship $\mathbf{O}_i^k \subseteq \mathbf{O}_i^{k-1}$ during predictable mode m_i. The set of state constraints, Γ_i^k, is the complement of $\mathbf{O}_i^k$ such that $\mathbf{z}_k \in \Gamma_i^k$ and $\Gamma_i^{k-1} \subseteq \Gamma_i^k$ during mode m_i.*

The MPC problem during mode m_i is:

$$\mathbf{z}_k^o = \arg\min_{\mathbf{z}_k} J(\mathbf{z}_k), \tag{7}$$

$$s.t. \quad \mathbf{z}_k \in \Gamma_i^k(z_0(k), \mathbf{O}_i^k). \tag{8}$$

Here, Γ_i^k also satisfies Assumption 2. Denote $J^*(k) = J(\mathbf{z}_k^o)$.

A condition on the MPC controller is also needed to guarantee a sufficient knowledge of the obstacle size in the planning problem:

Condition 2 (Prediction horizon and terminal cost). *Both the prediction horizon H and the penalty on the terminal cost are sufficient that during mode m_i, the open-loop problem always solves for a solution that can bring the robot to completely pass by the obstacle.*

With Assumption 1, the cost function is time-invariant and quadratic. With Condition 2 and Main result 1, the number of future way points affected by the obstacle is non-increasing. Therefore, $\Gamma_i^{k-1} \subseteq \Gamma_i^k$ implies that the open-loop planning problem is finding a solution in a larger and larger space throughout time with regard to a time-invariant cost function during mode m_i.

Main result 2 (Closed-loop stability of MPC). *The cost of the optimal open-loop trajectory follows $0 \leq J^*(k) \leq J^*(k-1)$ when Assumption 1 to 3 and Condition 1 and 2 are satisfied during mode m_i. Therefore, the proposed MPC with stability enhanced prediction is stable in the sense of Lyapunov during mode m_i before encountering a new obstacle or the worker begins to leave the obstacle region, i.e., m switches back to m_{up}.*

Regarding the closed-loop stability, the main results may seem to be relatively straight forward to obtain under the assumptions. However, it is important that with Condition 1, the proposed method can react to situations when the condition of a Lyapunov function is violated, i.e., collision is avoided and thus, safety is guaranteed at all times.

A Predictor Example

The following example presents a preliminary closed-loop stability enhanced predictor, which provides needed information to enhance closed-loop stability. First, the scenario where the workers' motion is predictable needs to be identified. In a factory setting, human workers and mobile robots usually move toward a place and stop to complete some tasks. This means they are going to stay in an area for an amount of time which is likely to be long enough for the ego robot to consider them as static objects with margins. The robot can then plan a path to go around them so that the shared-space is better utilized. We divided the worker's movement into two mode, moving mode and working (stop) mode. Denote the mode as $m \in \{moving, working\}$. Here *moving* is the unpredictable mode and *working* is the predictable mode. Therefore, it is important to know whether the worker is going to stop, i.e., switches to predictable mode. To do such prediction, denote the past position of the obstacle at time $t = k$ to be $z_{obs}(k)$, the past path of the obstacle is $\mathbf{h}_{obs}^k = [z_{obs}(0), z_{obs}(1), \ldots, z_{obs}(k)]^\top$. With $\mathbf{h}_{obs}^k$, denote the probability for the obstacle to continue moving and to stop to be $p_{go}(k)$ and $p_{stop}(k)$, respectively ($p_{go}(k) + p_{stop}(k) = 1$). Let r_1 and r_2 to be the update rates ($r_1 > 1$ and $0 < r_2 < 1$) and b to be the bias term. Notice that T_d is determined by the choices of these constants. We have the probability-update algorithm as shown in Algorithm 1.

Although the worker/robot has stopped, it is still possible that the he/she will have some movements within the working area (i.e., obstacle region). Therefore, the ego robot needs to know potentially how big the area is by observing the movements of the worker/robot. A naive Bayesian filtering-based method is presented to track the obstacle (e.g., other moving robot or human worker) and give a safety margin according to the past movements that the robot should keep away from the obstacle.

Each past position has an associated weight

Algorithm 1 The probability-update algorithm.

procedure MOVEPRED($\mathbf{h}^k_{obs}, p_{go}(k-1), p_{stop}(k-1)$)
  Acc $\leftarrow getAcc(\mathbf{h}^k_{obs})$
  endVel $\leftarrow PredictEndVel(Acc)$
  if abs(endVel) $\geq$ threshold **then**
    $w_{go} \leftarrow r_1 p_{go}(k-1)$
    $w_{stop} \leftarrow r_2 p_{stop}(k-1) + b$
  else
    $w_{go} \leftarrow r_2 p_{go}(k-1) + b$
    $w_{stop} \leftarrow r_1 p_{stop}(k-1)$
  $p_{go}(k), p_{stop}(k) \leftarrow normalize(w_{go}, w_{stop})$
  return $p_{go}(k), p_{stop}(k)$

$p(z_{obs}(k))$, and the weight vector for $\mathbf{h}^k_{obs}$ is $\mathbf{p}^k_{obs} = [p(z_{obs}(0)), p(z_{obs}(1)), \ldots, p(z_{obs}(k))]^\top$. In each MPC time step, $\mathbf{p}^k_{obs}$ is updated:

$$\begin{aligned} \hat{\mathbf{p}}^k_{obs} &= \left[\left[\gamma \mathbf{p}^{k-1}_{obs} \right]^\top 1 \right]^\top, \\ \mathbf{p}^k_{obs} &= \frac{\hat{\mathbf{p}}^k_{obs}}{\left\| \hat{\mathbf{p}}^k_{obs} \right\|_1}, \end{aligned} \tag{9}$$

where γ is the discount constant ($0 < \gamma < 1$). Let $\mathbf{h'}^k_{obs}$ and $\mathbf{p'}^k_{obs}$ be the last 10 elements of $\mathbf{h}^k_{obs}$ and $\mathbf{p}^k_{obs}$, respectively. To determine the obstacle region in the motion planning problem at each MPC time step, denote the uncertainty index as $\sigma(k) = std(\mathbf{p'}^k_{obs} \circ \mathbf{h'}^k_{obs})$, where $\circ$ denotes the entry-wise multiplication. A path that has bigger position change for the last several time steps will result in larger $\sigma(k)$. This can serve as an indicator of position uncertainty in the future under the assumption that the future movement of the obstacle is related to the current and past movements.

State Space Formulation and Stability

As denoted previously, k_{ps} is the time when the predictor detects that the worker/robot has switched to working mode ($m = working$), i.e., $p_{go}(k_{ps}) \leq c p_{stop}(k_{ps})$ and $0 \leq c \leq 1$. In the following, we construct the state space during working mode ($k > k_{ps}$) with the information presented in the previous section. Denote the high-dimensional obstacle region in the trajectory space at each MPC time step as $\mathbf{O}^k_w(\sigma(k), \mathbf{h'}^k_{obs})$. Consider the scenario where there is an obstacle coming into the scene and staying within a working area (obstacle region) while moving back and forth to complete a task in the area, stability analysis with regard to the open-loop cost function can be done with some assumptions.

With Assumption 3, it can be shown that $\sigma(k)$ has an upper bound, $\sigma(k) \leq O(t^2 v^u_{working})$, where t is the number of time steps used to calculate $\sigma(k)$ and $t = 10$ in our case. Therefore, the motion is predictable and it is possible to find $\mathbf{O}^k_w$ that can cover the obstacle movements as stated in Definition 1. With a proper choice of the update rates, the first condition holds and Main result 1 can be applied.

Again, the set of state constraints, Γ^k_w, is the complement of $\mathbf{O}^k_w$ and satisfies Assumption 2. The MPC problem during mode *working* is:

$$\mathbf{z}^o_k = \arg\min_{\mathbf{z}_k} J(\mathbf{z}_k), \tag{10}$$

$$s.t. \quad \mathbf{z}_k \in \Gamma^k_w(z_0(k), \mathbf{O}^k_w). \tag{11}$$

Denote $J^*(k) = J(\mathbf{z}^o_k)$.

The cost of the optimal open-loop trajectory follows $0 \leq J^*(k) \leq J^*(k-1)$ when Assumption 1 to 3 and Condition 1 and 2 are satisfied. Therefore, the proposed MPC with stability enhanced prediction is stable in the sense of Lyapunov for all k during mode *working*.

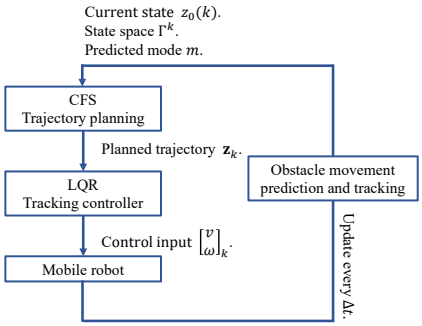


FIGURE 2: The overall system control design.

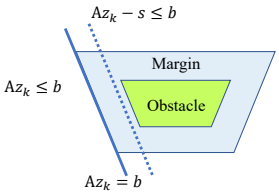


FIGURE 3: Illustration of the slack variable.

Overall System

The overall system is shown in FIGURE 2. Although the planned trajectory is feasible, tracking error will occur in real world experiment, causing the state becoming infeasible. This results in a violent motion because the control command according to the new planning result at the next time step will immediately pull the robot out of the infeasible area. In order to avoid

such problem, we introduce $\mathbf{S}^k = [s^k_{k+1}, s^k_{k+2}, \ldots, s^k_{k+H}]$, the slack variable vector. Introducing slack variables allows the states to violate the original constraints, which are the margin boundaries of the obstacles (FIGURE 3). However these slack variables are also added to the cost function so that the violation is penalized [14]. The new problem is shown in the following:

$$\min_{\mathbf{z}_k} \; J(\mathbf{z}_k) + \|\mathbf{S}^k\|_2^2, \tag{12}$$

$$s.t. \quad \mathbf{z}_k \in \Gamma^k, \tag{13}$$

where Γ^k is:

$$\Gamma^k = \begin{cases} \Gamma^k_w(z_0(k), \mathbf{O}^k_w, \mathbf{S}^k), & \text{if } m = working \\ \Gamma^k_m(z_0(k), \mathbf{O}^k_m, \mathbf{S}^k), & \text{otherwise} \end{cases}. \tag{14}$$

An iterative LQR (ILQR) controller [15] is used for a better tracking performance, which outputs the control commands $[v, \omega]^\top_k$. The overall system goes through a process as shown in Algorithm 2 at ever MPC time step.

Algorithm 2 The overall algorithm.

```
while MPC do
    GetInformation()
    Predictor()
    if p_go(k_w) ≤ cp_stop(k_w) then
        m = working
    else
        m = moving
    z_k ← CFS(z_0(k), Γ^k, m)
    [v, ω]^⊤_k ← ILQR(z_k)
    ego robot executes the control commands
```

M-Convergence for Analyzing Closed-loop Performance

In this paper, we propose a new notion, M-convergence, to analyzing transient of the planned open-loop trajectory from one planning instance to the next.

Definition 3 (M-convergence). *We say that an action location, z_k, is M-converging if for $k-M < t \leq k$, the last M predictions for z_k satisfy: (i) $\|z^t_k - z^{t-1}_k\| \leq \|z^{t-1}_k - z^{t-2}_k\|$, or (ii) $\left|\|z^t_k - z^{t-1}_k\| - \|z^{t-1}_k - z^{t-2}_k\|\right| \leq \delta$, when $\|z^t_k - z^{t-1}_k\| \leq \varepsilon$, or both. Here δ and ε are small thresholds.*

In FIGURE 4, the pink-orange-color circles are the action location z_k planned at different time steps that are tested for M-convergence of z_k. The conditions imply that the planned state z_k would have smaller and smaller change ((i)) or sufficiently small change ((ii)) on the predicted action location between consecutive time steps after time step $k-M$. This notion allows us to examine the local convergence property of the open-loop trajectories to the closed-loop trajectory at each action location.

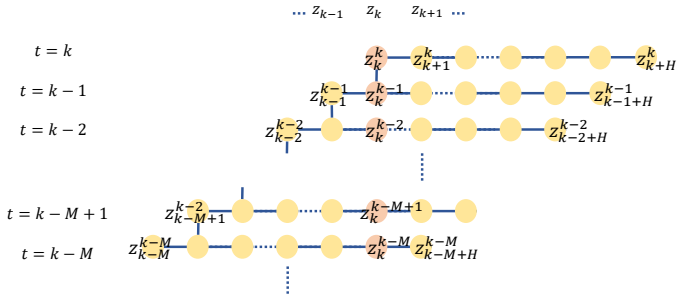


FIGURE 4: Illustration of M-convergence.

Having such property benefits the robot performance. M-convergence guarantees that the action location will not change much, and therefore, guarantees smoothness of the output trajectory. If M is sufficiently large, the robot system will not experience sudden changes. Moreover, since the planned trajectory is usually tracked by a low-level tracking controller, the larger the M is, the smoother the control command would be. Therefore, M-convergence is chosen to be one of the metrics to evaluate the closed-loop performance.

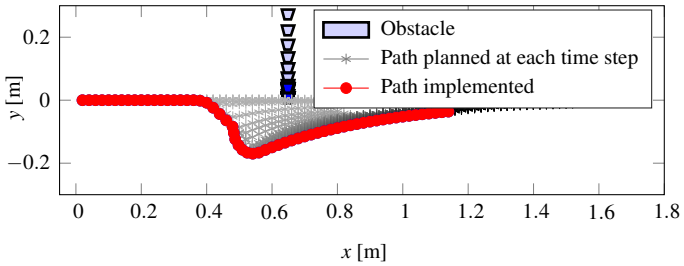
SIMULATION RESULT AND DISCUSSION

The simulation scenario in this work is similar to that of mobile robots operating in factories. To test our algorithm and see the improvement of the proposed method, a common industrial scenario is considered. The goal for the robot is to move along a line, $y = 0$, in the positive x-axis direction, while maintaining a constant speed which also points along the positive x-axis.

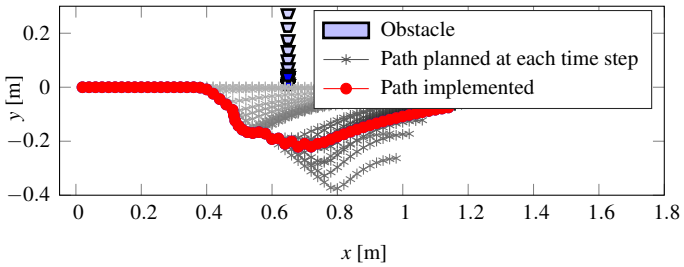
Result of Scenario with Shared-space Working Area

To show that the proposed method can better deal with dynamic environments, we design a scenario which is common in human-robot shared-space environments. In the beginning, the human worker walks (moving mode) in constant acceleration towards and stops around the line $y = 0$ in front of the ego robot. Once the worker stops, he/she starts to work in the area and moves back and forth in a small range (working mode). To successfully pass the working area, the robot needs to be able to determine the size of the working area and not be overreacting to the worker's small but inconsistent movements.

Simulation results are shown in FIGURE 5a and FIG-



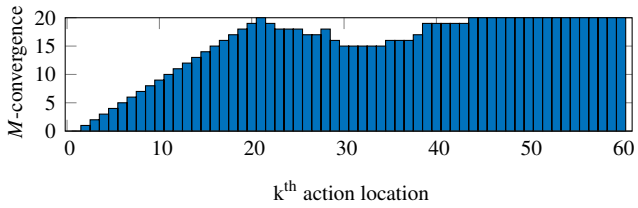
(a) Simulation result of the proposed MPC with stability enhanced prediction.



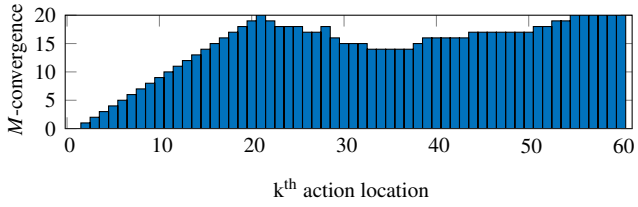
(b) Simulation result of MPC without stability enhanced prediction.

FIGURE 5: Scenario that has an oscillating moving obstacle.

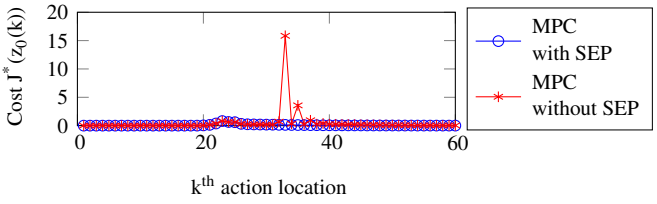
URE 5b, where the blue blocks represent the worker's positions and the gray star-lines represent the planned trajectories at each time step (darker colors represents time steps that are closer to current). Here, $k_{ps} = 31$. We can see that the proposed method has a overall smooth closed-loop trajectory, while MPC without stability enhanced prediction experience oscillation on the open-loop prediction due to the worker's small movements in the working mode. Correspondingly, FIGURE 6a and FIGURE 6b also show that the proposed method recovers faster in M-convergence after the worker switches to working mode. In FIGURE 6c we can see that having the knowledge of the size of the working area, the proposed method can plan safely around the worker while not being overreacting to the small movements of the worker. On the other hand, MPC without stability enhanced prediction reacts strongly to the speed change and results in open-loop trajectories with high cost. In FIGURE 6d we can see that the cost with the proposed method is the same as MPC without stability enhanced prediction in the first 12 steps but is noticeably smaller afterwords. This is because the proposed method can better utilize the share-space therefore has better overall performance.



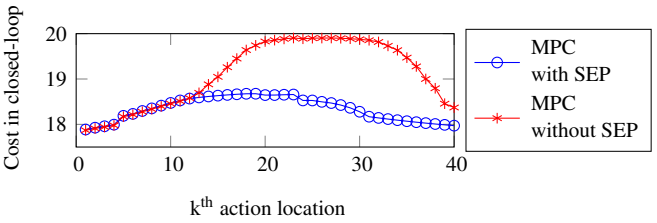
(a) M-convergence for simulation result of the proposed MPC with stability enhanced prediction.



(b) M-convergence for simulation result of MPC without stability enhanced prediction.



(c) Comparison for open-loop cost of MPC with/without stability enhanced prediction (SEP).



(d) Comparison for closed-loop cost of MPC with/without stability enhanced prediction (SEP).

FIGURE 6: Comparison between MPC without stability enhanced prediction and the proposed MPC with stability enhanced prediction.

Discussion

The proposed method is also tested in other similar scenarios with different factors, e.g, worker's acceleration, range of movement during working mode. It is shown that the ego robot can always complete its task without colliding with the obstacle. However, it is noticed that tuning the predictor sometimes improves the performance, this indicates that a better predictor is crucial in the improvement of the proposed MPC with stability enhanced prediction. The predictor introduced in this paper is only a preliminary version and can be replaced by more advanced ones in the future.

EXPERIMENTAL RESULT

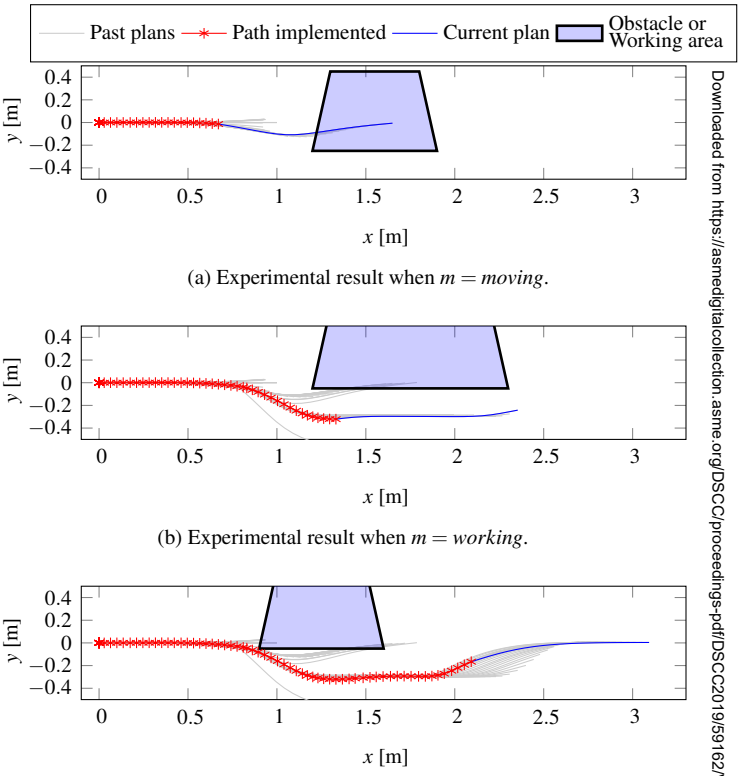
To verify the performance, the proposed MPC (with $H = 20$) controller with stability enhanced prediction is tested on TurtleBot3, which is developed by ROBOTIS. TurtleBot3 is a ROS standard platform robot. The MPC controller is running in MATLAB on a separate laptop with an 2.8GHz Intel Core i7-7700HQ.

Similar to the simulation scenario, we have the ego robot running the proposed MPC with stability enhanced prediction and another remote-controlled robot (obstacle robot) representing the worker in the first scenario. The experimental result is shown in FIGURE 9a, where we can see the robot successfully avoids the obstacle and merges back to the original route without overreacting to the other robot's movements in the working area.

In the second scenario (FIGURE 9b), a human worker, having a higher priority, i.e., doesn't need to yield to the robot, comes toward to the table to perform some tasks and leaves the working area toward the opposite direction of the robot after finishing the tasks. The robot running the proposed MPC with stability enhanced prediction avoids the human worker while tracking a straight line. From FIGURE 7a, we can see that at the beginning the robot detects the worker's movement and the prediction was assuming constant speed. After the worker switches to working mode, the robot predicts the working area and plans accordingly (FIGURE 7b). Finally, after the worker left the working area and starts to walk away from the robot, the robot detects that movement, switches m back to $m = moving$, and starts to merge back to the line $y = 0$ prior to the prediction when m is still at the working mode. FIGURE 7c shows that the gray lines turn earlier and earlier comparing to the blue line in FIGURE 7b as the worker leaves.

The third scenario is similar to the second scenario at the beginning, however, the worker leaves from the other side of the table. Therefore, the worker walks along the same direction as the robot for a short period of time after leaving the working area. This experiment shows that the proposed method can react to situations when the condition of a Lyapunov function is violated, i.e., avoiding collision when the predictor is switching back to unpredictable mode. From FIGURE 8a, we can see that at the beginning the robot does the same thing as in scenario 2 shown in FIGURE 7b. After the worker switches back to moving mode, the robot detects that movement, switches m back to $m = moving$, and can successfully re-plan and continue to avoid collision (FIGURE 8b) instead of merging back to $y = 0$ as planned previously (FIGURE 8a). We can also see that because the working area is sufficient to capture the obstacle movement when the predictor is detecting the change, which fulfills Condition 1, the path implemented is smooth even though there is a change in the worker's mode. Finally, after the worker starts to leave the scene along the positive y direction, the robot starts to merge back to the line $y = 0$ (FIGURE 8c).

The experimental results show that with the proposed method, the robot can deal with time varying environment and efficiently utilize the shared working space.



(a) Experimental result when $m = moving$.

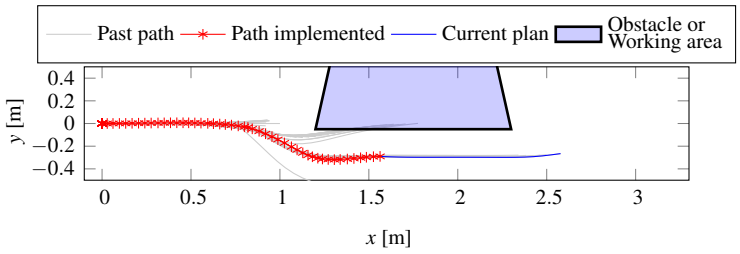
(b) Experimental result when $m = working$.

(c) Experimental result when m is switched back to $m = moving$ (obstacle is moving away from the robot).

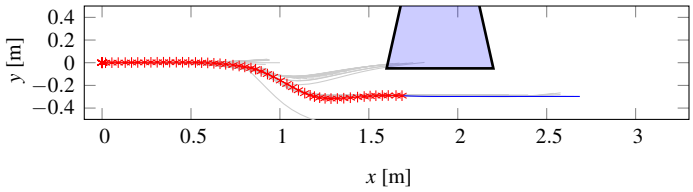
FIGURE 7: Path planned and implemented in the second scenario.

CONCLUSION

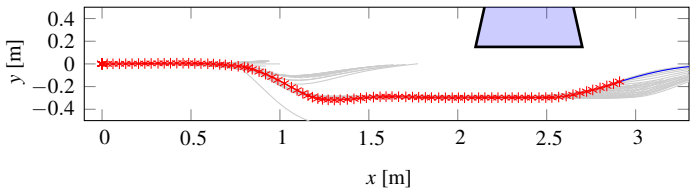
This paper discussed the conditions to enable closed-loop stability of a motion planning MPC that handles common time varying scenarios in co-robot systems involving dynamic HRI. In order to guarantee such property, under the listed assumptions, the predictor needed to be able to detect the workers' movement mode change within a time delay allowance and the MPC needed to have a sufficient prediction horizon and a proper cost function. An example of a MPC with a closed-loop stability enhanced predictor was presented. Satisfying the conditions, the proposed MPC with stability enhanced prediction observed the environment and constructed proper state constraints for the optimization problem. This guaranteed the robot to have closed-loop stability theoretically when all assumptions were satisfied. The conditions also allowed the proposed method to avoid collision even though the environment did not satisfy all the assumptions. To evaluate the proposed method's performance in practice, a new notion, M-convergence, was used and simulation results showed



(a) Experimental result when $m = working$.



(b) Experimental result right after m is switched back to $m = moving$.



(c) Experimental result when the worker leaves the scene.

FIGURE 8: Path planned and implemented in the third scenario.

that a robot with the proposed method was capable of dealing with dynamic environments, had a better convergence property, and thus, resulted in a smooth closed-loop trajectory. It was also shown that the cost was reduced sufficiently for both open-loop cost and closed-loop cost. Finally, experiments on Turtlebot3 showed that the proposed method performed well in time varying environments. In the future, we will improve the predictor so that the proposed MPC with stability enhanced prediction can deal with more complicated environments efficiently.

ACKNOWLEDGMENT

Thanks go to Liting Sun, Changliu Liu, Rachel Lim, and Jieming Li for the help in editing the paper and in conducting the experiments. This work was supported by the National Science Foundation under Grant No.1734109. Any opinion, finding, and conclusion expressed in this paper are those of the authors and do not necessarily reflect those of the National Science Foundation.

REFERENCES

[1] Rawlings, J. B., 1999. "Tutorial: Model predictive control technology". In American Control Conference, 1999. Proceedings of the 1999, Vol. 1, IEEE, pp. 662–676.

[2] Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O., 2000. "Constrained model predictive control: Stability and optimality". *Automatica,* **36**(6), pp. 789–814.

[3] Borrelli, F., Falcone, P., Keviczky, T., Asgari, J., and Hrovat, D., 2005. "Mpc-based approach to active steering for autonomous vehicle systems". *International Journal of Vehicle Autonomous Systems,* **3**(2-4), pp. 265–291.

[4] Gu, D., and Hu, H., 2005. "A stabilizing receding horizon regulator for nonholonomic mobile robots". *IEEE Transactions on Robotics,* **21**(5), pp. 1022–1028.

[5] Chisci, L., Rossiter, J. A., and Zappa, G., 2001. "Systems with persistent disturbances: predictive control with restricted constraints". *Automatica,* **37**(7), pp. 1019–1028.

[6] Kuwata, Y., Richards, A., Schouwenaars, T., and How, J. P., 2007. "Distributed robust receding horizon control for multivehicle guidance". *IEEE Transactions on Control Systems Technology,* **15**(4), pp. 627–641.

[7] Luders, B., Kothari, M., and How, J., 2010. "Chance constrained rrt for probabilistic robustness to environmental uncertainty". In AIAA guidance, navigation, and control conference, p. 8160.

[8] Goodrich, M. A., Schultz, A. C., et al., 2008. "Human–robot interaction: a survey". *Foundations and Trends® in Human–Computer Interaction,* **1**(3), pp. 203–275.

[9] Kruse, T., Pandey, A. K., Alami, R., and Kirsch, A., 2013. "Human-aware robot navigation: A survey". *Robotics and Autonomous Systems,* **61**(12), pp. 1726–1743.

[10] Cheng, Y., Liu, C., Zhao, W., and Tomizuka, M., 2018. "Human motion prediction using adaptable neural networks". In arXiv:1810.00781.

[11] Bütepage, J., Kjellström, H., and Kragic, D., 2017. "Anticipating many futures: Online human motion prediction and synthesis for human-robot collaboration". *arXiv preprint arXiv:1702.08212*.

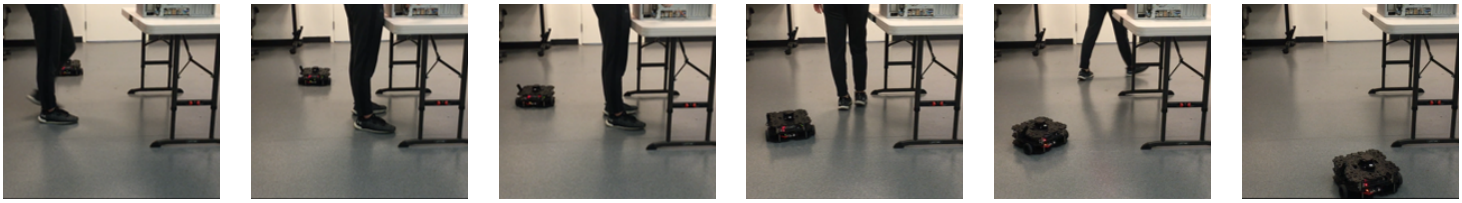
[12] Kumar, P., Perrollaz, M., Lefevre, S., and Laugier, C., 2013. "Learning-based approach for online lane change intention prediction". In Intelligent Vehicles Symposium (IV), 2013 IEEE, IEEE, pp. 797–802.

[13] Liu, C., Lin, C.-Y., and Tomizuka, M., 2018. "The convex feasible set algorithm for real time optimization in motion planning". *SIAM Journal on Control and Optimization,* **56**(4), pp. 2712–2733.

[14] Chen, J., Liu, C., and Tomizuka, M., 2018. "Foad: Fast optimization-based autonomous driving motion planner". In 2018 Annual American Control Conference (ACC), IEEE, pp. 4725–4732.

[15] Tassa, Y., Erez, T., and Todorov, E., 2012. "Synthesis and stabilization of complex behaviors through online trajectory optimization". In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, IEEE, pp. 4906–4913.

(a) Experimental result with mobile robot passing by another robot.



(b) Experimental result with mobile robot passing by a human worker that walks away from the robot after leaving the working area.

(c) Experimental result with mobile robot passing by a human worker that walks along the same direction as the robot for a short period of time after leaving the working area.

FIGURE 9: Experimental result of mobile robot running the proposed MPC with stability enhanced prediction.