Model Predictive Control based on Trajectory-Tracking Error Model

By: Walid Shaker

0.1 Development of the Kinematic Trajectory-Tracking Error Model

The trajectory tracking problem can be defined as controlling a robot to follow a reference trajectory $q_{ref}(t) = [x_{ref}(t), y_{ref}(t), \theta_{ref}(t)]^T, t \in [0, T]$ [1]. The posture of the error is not given in the global coordinate system, but rather as an error in the local coordinate system of the robot that is aligned with the driving mechanism. This error is expressed as the deviation between a virtual reference robot and the actual robot as depicted in Fig. 1. The obtained errors are as follows: e_x that gives the error in the direction of driving, e_y that gives the error in the perpendicular direction, and e_θ that gives the error in the orientation. These errors are illustrated in Fig. 1.

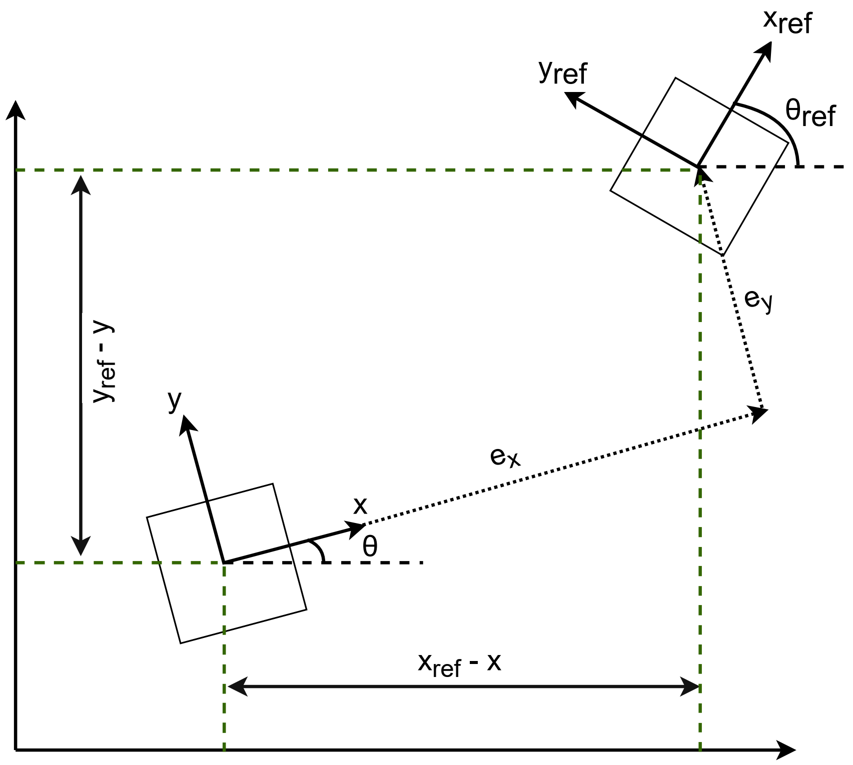

Figure 1: Posture error illustration in local coordinates.

The posture error $e(t) = [e_x(t), e_y(t), e_\theta(t)]^T$ is determined using the actual posture

$q(t) = [x(t), y(t), \theta(t)]^T$ of the real robot and the reference posture $q_{ref}(t) = [x_{ref}(t), y_{ref}(t), \theta_{ref}(t)]^T$ of the virtual reference robot.

where

$$\begin{aligned} e_x(t) &= (x_{ref}(t) - x(t))\cos(\theta(t)) + (y_{ref}(t) - y(t))\sin(\theta(t)) \\ e_y(t) &= -(x_{ref}(t) - x(t))\sin(\theta(t)) + (y_{ref}(t) - y(t))\cos(\theta(t)) \\ e_\theta(t) &= \theta_{ref}(t) - \theta(t) \end{aligned} \tag{1}$$

This can be represented as follows:

$$\begin{bmatrix} e_x(t) \\ e_y(t) \\ e_\theta(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta(t)) & \sin(\theta(t)) & 0 \\ -\sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} (q_{ref}(t) - q(t)) = R(t)\tilde{q}(t) \tag{2}$$

Recall the dynamics of the differential drive system:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3}$$

So,

$$\dot{q} = \begin{bmatrix} \cos(\theta(t))v(t) \\ \sin(\theta(t))v(t) \\ \omega(t) \end{bmatrix} \tag{4}$$

Taking the time derivative of Eq. 2 and using Eq. 4 yields

$$\begin{aligned} \dot{e} &= R\dot{\tilde{q}} + \dot{R}\tilde{q} \\ \dot{e} &= \begin{bmatrix} \cos(\theta)[\cos(\theta_{ref})v_{ref} - \cos(\theta)v] + \sin(\theta)[\sin(\theta_{ref})v_{ref} - \sin(\theta)v] \\ -\sin(\theta)\dot{\theta}(x_{ref} - x) + \cos(\theta)\dot{\theta}(y_{ref} - y) \\ -\sin(\theta)[\cos(\theta_{ref})v_{ref} - \cos(\theta)v] + \cos(\theta)[\sin(\theta_{ref})v_{ref} - \sin(\theta)v] \\ -\cos(\theta)\dot{\theta}(x_{ref} - x) - \sin(\theta)\dot{\theta}(y_{ref} - y) \\ \omega_{ref} - \omega \end{bmatrix} \\ \dot{e} &= \begin{bmatrix} v_{ref}\cos(e_\theta) - v + e_y\omega \\ v_{ref}\sin(e_\theta) - e_x\omega \\ \omega_{ref} - \omega \end{bmatrix} \end{aligned} \tag{5}$$

Therefore, the posture error model can be written as follows:

$$\begin{bmatrix} \dot{e_x} \\ \dot{e_y} \\ \dot{e_\theta} \end{bmatrix} = \begin{bmatrix} \cos(e_\theta) & 0 \\ \sin(e_\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} + \begin{bmatrix} -1 & e_y \\ 0 & -e_x \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{6}$$

where v_{ref} and ω_{ref} are the linear and the angular reference velocities, which will be defined later. The input $u = [v, \omega]^T$ is to be commanded by the controller. Very often the control u is decomposed as

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v_{ff} + v_{fb} \\ \omega_{ff} + \omega_{fb} \end{bmatrix} \tag{7}$$

The role of the feedforward components v_{ff} and ω_{ff} are to keep the robot following the trajectory assuming zero tracking error. To achieve this, set

$$v_{ff} = v_{ref}\cos(e_\theta), \omega_{ff} = \omega_{ref} \tag{8}$$

where $v_{ref}\cos(e_\theta)$ is modulated with the orientation error that originates from the output. On the other hand, when the orientation error is driven to 0, $v_{ref}\cos(e_\theta)$ becomes a "true" feedforward.

Consequently, Eq. 7 becomes

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v_{ref}\cos(e_\theta) + v_{fb} \\ \omega_{ref} + \omega_{fb} \end{bmatrix} \tag{9}$$

Substituting Eq. 9 in tracking-error model Eq. 6

$$\begin{gathered} \begin{bmatrix} \dot{e_x} \\ \dot{e_y} \\ \dot{e_\theta} \end{bmatrix} = \begin{bmatrix} \cos(e_\theta)v_{ref} - v_{ref}\cos(e_\theta) - v_{fb} + \omega_{ref}e_y + \omega_{fb}e_y \\ \sin(e_\theta)v_{ref} - \omega_{ref}e_x + \omega_{fb}e_x \\ \omega_{ref} - \omega_{ref} - \omega_{fb} \end{bmatrix} \\ \begin{bmatrix} \dot{e_x} \\ \dot{e_y} \\ \dot{e_\theta} \end{bmatrix} = \begin{bmatrix} -v_{fb} + \omega_{ref}e_y + \omega_{fb}e_y \\ \sin(e_\theta)v_{ref} - \omega_{ref}e_x + \omega_{fb}e_x \\ -\omega_{fb} \end{bmatrix} \end{gathered} \tag{10}$$

0.2 Error Model Linearization

The error model in Eq. 10 is nonlinear. In this section this model will be linearized to enable the use of a linear controller. The linearization has to take place around some equilibrium point. Here the obvious choice is the zero error ($e_x = e_y = 0$, $e_\theta = 0$). This point is an equilibrium point of Eq. 10 if both feedback velocities are also 0 ($v_{fb} = 0$, $\omega_{fb} = 0$) [2]. Linearizing Eq. 10 around the zero-error yields the following:

$$\begin{bmatrix} \dot{e_x} \\ \dot{e_y} \\ \dot{e_\theta} \end{bmatrix} = \begin{bmatrix} 0 & \omega_{ref} & 0 \\ -\omega_{ref} & 0 & v_{ref} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v_{fb} \\ \omega_{fb} \end{bmatrix} \tag{11}$$

which is a linear time-varying system due to $v_{ref}(t)$ and $\omega_{ref}(t)$ being time dependent. The system given by Eq. 11 is a state-space representation of a dynamic error-system where all the states (errors in this case) are accessible.

0.3 Feedforward Velocities

For a given time t and a given desired reference trajectory, the orientation $\theta_{ref}(t)$ and command $u_{ref}(t) = [v_{ref}(t), \omega_{ref}(t)]^T$ needed to maintain perfect trajectory tracking are given by

$$v_{ref}(t) = (-1)^b \sqrt{\dot{x}_{ref}^2(t) + \dot{y}_{ref}^2(t)} \tag{12}$$

$$\theta_{ref}(t) = atan2(\dot{y}_{ref}(t), (\dot{x}_{ref}(t)) + b\pi \tag{13}$$

where b defines the motion direction with $b = 0$ moving the robot forwards and $b = 1$ moving it backwards.

Since $\theta_{ref}(t) = \arctan(\frac{\dot{y}_{ref}(t)}{(\dot{x}_{ref}(t)})$, let $u = \frac{\dot{y}_{ref}(t)}{(\dot{x}_{ref}(t)}$ yields $\tan(\theta_{ref}(t)) = u$

Taking the time derivative of $\tan(\theta_{ref}(t)) = u$:

$\sec^2(\theta_{ref}(t))\dot{\theta}_{ref}(t) = \dot{u}$

Therefore,

$$\dot{\theta}_{ref}(t) = \omega_{ref}(t) = \frac{\dot{u}}{\sec^2(\theta_{ref}(t))} = \frac{\dot{u}}{\tan^2(\theta_{ref}(t)) + 1} = \frac{\dot{u}}{u^2 + 1} \tag{14}$$

So, $\omega_{ref}(t)$ can be given by

$$\omega_{ref}(t) = \frac{\dot{x}_{ref}(t)\ddot{y}_{ref}(t) - \dot{y}_{ref}(t)\ddot{x}_{ref}(t)}{\sqrt{\dot{x}_{ref}^2(t) + \dot{y}_{ref}^2(t)}} \tag{15}$$

0.4 Model Predictive Control

MPC uses a model of the plant to make predictions about the future plant output behavior. It also uses an optimizer, which ensures that the predicted future plant output tracks the desired reference. It minimizes the difference between the robot future tracking error and the reference error with defined desired dynamics. The control problem is given by the linear tracking error dynamics model Eq. 11, which in shorter notation reads

$$\dot{e} = A_c(t)e + B_c u_{fb} \tag{16}$$

where $A_c(t)$ and B_c are matrices of the continuous state-space model and e is the tracking error defined in local robot coordinates, defined by transformation Eq. 2 and shown in Fig. 1.

Since our controllers will be implemented on physical robots, we need a discrete time model. Using a simple Euler approximation, we get

$$e(k+1) = A(k)e(k) + B(k)u_{fb}(k) \tag{17}$$

where $A(k) \in R^n \times R^n, n$ is the number of the state variables and $B \in R^n \times R^m$, and m is the number of input variables. The discrete matrices $A(k)$ and B can be obtained as follows:

$$\begin{aligned}\boldsymbol{A}(k) &= \boldsymbol{I} + A_{\mathrm{c}}(t)T_{\mathrm{s}} \\ \boldsymbol{B} &= \boldsymbol{B}_{\mathrm{c}}T_{\mathrm{s}}\end{aligned} \tag{18}$$

which is sufficient approximation for a short sampling time T_s. So,

$$\begin{aligned} A(k) &= \begin{bmatrix} 1 & \omega_{ref}(k)T_s & 0 \\ -\omega_{ref}(k)T_s & 1 & v_{ref}(k)T_s \\ 0 & 0 & 1 \end{bmatrix}, \\ B &= \begin{bmatrix} -T_s & 0 \\ 0 & 0 \\ 0 & -T_s \end{bmatrix} \end{aligned} \tag{19}$$

The main idea of MPC is to compute optimal control actions that minimize the objective function given in prediction horizon interval h. The objective function is a quadratic cost function:

$$J\left(\boldsymbol{u}_{\mathrm{fb}}, k\right) = \sum_{i=1}^{h} \boldsymbol{\epsilon}^T(k,i)\boldsymbol{Q}\boldsymbol{\epsilon}(k,i) + \boldsymbol{u}_{\mathrm{fb}}^T(k+i-1)\boldsymbol{R}\boldsymbol{u}_{\mathrm{fb}}(k+i-1) \tag{20}$$

It is similar to LQR cost function with the only difference being that the optimization is over the finite horizon N rather than the infinite horizon. It consists of a future reference tracking error $e_r(k+i)$, predicted tracking error $e(k+i \mid k)$, difference between the two errors $\epsilon(k,i) = e_r(k+i) - e(k+i \mid k)$, and future actions $\boldsymbol{u}_{\mathrm{fb}}(k+i-1)$, where i denotes the ith step-ahead prediction ($i = 1, \ldots, h$). $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the weighting matrices.

For prediction of the state $\boldsymbol{e}(k+i \mid k)$ the error model Eq. 17 is applied as follows:

$$\begin{aligned} \boldsymbol{e}(k+1 \mid k) &= A(k)e(k) + Bu_{\mathrm{fb}}(k) \\ \boldsymbol{e}(k+2 \mid k) &= A(k+1)e(k+1 \mid k) + Bu_{\mathrm{fb}}(k+1) \\ &\vdots \\ e(k+i \mid k) &= A(k+i-1)\boldsymbol{e}(k+i-1 \mid k) + Bu_{\mathrm{fb}}(k+i-1) \\ &\vdots \\ \boldsymbol{e}(k+h \mid k) &= A(k+h-1)\boldsymbol{e}(k+h-1 \mid k) + Bu_{\mathrm{fb}}(k+h-1) \end{aligned} \tag{21}$$

Predictions $\boldsymbol{e}(k+i \mid k)$ in Eq. 21 are rearranged so that they are dependent on current error $\boldsymbol{e}(k)$, current and future inputs $\boldsymbol{u}_{\mathrm{fb}}(k+i-1)$, and matrices $\boldsymbol{A}(k+i-1)$ and $\boldsymbol{B}$. The model-output prediction at the time instant h can then be written as

$$\begin{aligned} \boldsymbol{e}(k+h \mid k) =& \Pi_{j=1}^{h-1} A(k+j)\boldsymbol{e}(k) + \\ &+ \sum_{i=1}^{h} \left(\Pi_{j=i}^{h-1} A(k+j)\right) \boldsymbol{B}u_{\mathrm{fb}}(k+i-1) + \boldsymbol{B}u_{\mathrm{fb}}(k+h-1) \end{aligned} \tag{22}$$

The future reference error ($\boldsymbol{c}_r(k+i)$) defines how the tracking error should decrease when the robot is not on the trajectory. Let us define the future reference error to decrease exponentially from the current tracking error $e(k)$, as follows:

$$e_r(k+i) = A_r^i e(k) \tag{23}$$

for $i = 1, \ldots, h$. The dynamics of the reference error is defined by the reference model matrix $\boldsymbol{A}_r$.

According to Eqs. 21, 22, the robot-tracking prediction-error vector is defined as follows:

$$\boldsymbol{E}^*(k) = \left[e(k+1 \mid k)^T e(k+2 \mid k)^T \ldots e(k+h \mid k)^T\right]^T \tag{24}$$

where $\boldsymbol{E}^*$ is given for the whole interval of the observation (h) where the control vector is

$$\boldsymbol{U}_{\mathrm{fb}}(k) = \left[\boldsymbol{u}_{\mathrm{fb}}^T(k)\boldsymbol{u}_{\mathrm{fb}}^T(k+1)\ldots\boldsymbol{u}_{\mathrm{fb}}^T(k+h-1)\right]^T \tag{25}$$

and

$$\boldsymbol{\Lambda}(k,i) = \Pi_{j=i}^{h-1} A(k+j)$$

The robot-tracking prediction-error vector can be written in compact form:

$$\boldsymbol{E}^*(k) = F(k)e(k) + G(k)U_{\mathrm{fb}}(k) \tag{26}$$

where

$$\boldsymbol{F}^*(k) = \left[\boldsymbol{A}(k)\boldsymbol{A}(k+1)A(k)\ldots\boldsymbol{\Lambda}(k,0)\right]^T \tag{27}$$

and

$$\boldsymbol{G}(k) = \begin{bmatrix} \boldsymbol{B} & 0 & \cdots & 0 \\ \boldsymbol{A}(k+1)\boldsymbol{B} & \boldsymbol{B} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Lambda}(k,1)\boldsymbol{B} & \boldsymbol{\Lambda}(k,2)\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix} \tag{28}$$

where dimensions of $\boldsymbol{F}(k)$ and $\boldsymbol{G}(k)$ are $(n \cdot h \times n)$ and $(n \cdot h \times m \cdot h)$, respectively. The robot reference-tracking-error vector is

$$\boldsymbol{E}_r^*(k) = \left[\boldsymbol{e}_r(k+1)^T \boldsymbol{e}_r(k+2)^T \ldots \boldsymbol{e}_r(k+h)^T\right]^T$$

which in compact form is computed as

$$\boldsymbol{E}_r^*(k) = \boldsymbol{F}_r e(k) \tag{29}$$

where

$$\boldsymbol{F}_r = \left[\begin{array}{ccc} A_r A_r A_r^2 & \cdots & A_r A_r^h \end{array}\right]^T \tag{30}$$

where $\boldsymbol{F}_r$ is the $(n \cdot h \times n)$ matrix. The optimal control vector Eq. 25 is obtained by optimization of Eq. 20, which can be done numerically or analytically. The analytical solution is derived in the following.

The objective function Eq. 20 defined in matrix form reads.

$$J\left(\boldsymbol{U}_{\mathrm{fb}}\right) = \left(\boldsymbol{E}_r^* - \boldsymbol{E}^*\right)^T \overline{\boldsymbol{Q}}\left(\boldsymbol{E}_r^* - \boldsymbol{E}^*\right) + \boldsymbol{U}_{\mathrm{fb}}^T \overline{\boldsymbol{R}} \boldsymbol{U}_{\mathrm{fb}} \tag{31}$$

The minimum of Eq. 31 is expressed as

$$\frac{\partial J}{\partial \boldsymbol{U}_{\mathrm{fb}}} = -2\overline{\mathbf{Q}}\boldsymbol{G}^T \boldsymbol{E}_r^* + 2\boldsymbol{G}^T \overline{\mathbf{Q}} \boldsymbol{E}^* + 2\overline{\boldsymbol{R}}\boldsymbol{U}_{\mathrm{fb}} = 0 \tag{32}$$

and the solution for the optimal control vector is obtained as

$$\boldsymbol{U}_{\text{fb}}(k) = \left(\boldsymbol{G}^T \overline{\mathbf{Q}} \boldsymbol{G} + \overline{\boldsymbol{R}}\right)^{-1} \boldsymbol{G}^T \overline{\mathbf{Q}} \left(\boldsymbol{F}_r - \boldsymbol{F}\right) \boldsymbol{e}(k) \tag{33}$$

where the weighting matrices are as follows:

$$\overline{\boldsymbol{Q}} = \begin{bmatrix} \boldsymbol{Q} & 0 & \cdots & 0 \\ 0 & \boldsymbol{Q} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{Q} \end{bmatrix} \tag{34}$$

and

$$\overline{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{R} & 0 & \cdots & 0 \\ 0 & \boldsymbol{R} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{R} \end{bmatrix} \tag{35}$$

Solution of Eq. 33 contains control vectors $u_{\text{fb}}^T(k+i-1)$ for the whole interval of prediction ($i = 1, \ldots, h$). To apply feedback control action in the current time instant k only the first vector $u_{\text{fb}}^T(k)$ (first m rows of $\boldsymbol{U}_{\text{fb}}(k)$) is applied to the robot. The solution is obtained analytically; therefore, it enables fast real-time implementations that may not be possible if numeric optimization of Eq. 20 is used. The MPC controller is defined by a choice of horizon N and weighting matrices Q and R. We take N as large as possible given computational limitations and then tune Q and R to achieve the best performance.

REFERENCES

[1] G. Song, "Implementation and comparison of tracking algorithms for differential drive robots," Ph.D. dissertation, Boston University, 2021.

[2] G. Klancar, A. Zdesar, S. Blazic, and I. Skrjanc, *Wheeled mobile robotics: from fundamentals towards autonomous systems.* Butterworth-Heinemann, 2017.