Autonomous and Mobile Robotics
Prof. Giuseppe Oriolo

Wheeled Mobile Robots
Motion Control: Introduction and Trajectory Tracking
motion control

- A desired motion is assigned for the WMR, and the associated nominal inputs have been computed.
- To execute the desired motion, we need feedback control because the application of nominal inputs in open-loop would lead to very poor performance.
- Dynamic models are generally used in robotics to compute commands at the generalized force level.
- Kinematic models are used to design WMR feedback laws because (1) dynamic terms can be canceled via feedback and (2) wheel actuators are equipped with low-level PID loops that accept velocities as reference.
• **actual** control scheme

![Actual control scheme diagram](image)

- reference motion
- error
- high-level control
- velocity commands
- PID
- actuators
- robot (dyn model)
- actual motion
- actual velocities
- (localization)

• **equivalent** control scheme (for design)

![Equivalent control scheme diagram](image)

- reference motion
- error
- high-level control
- velocity commands
- robot (kin model)
- actual motion
- (localization)
motion control problems

- trajectory tracking (predictable transients)
- posture regulation (no prior planning)

w.l.o.g. we consider a unicycle in the following

\[
\begin{pmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{pmatrix} =
\begin{pmatrix}
\cos \theta \\
\sin \theta \\
0
\end{pmatrix} v +
\begin{pmatrix}
0 \\
0 \\
1
\end{pmatrix} \omega
\]
trajectory tracking: state error feedback

- the unicycle must track a Cartesian desired trajectory \((x_d(t), y_d(t))\) that is admissible, i.e., there exist \(v_d\) and \(\omega_d\) such that

\[
\begin{align*}
\dot{x}_d &= v_d \cos \theta_d \\
\dot{y}_d &= v_d \sin \theta_d \\
\dot{\theta}_d &= \omega_d
\end{align*}
\]

- thanks to flatness, from \((x_d(t), y_d(t))\) we can compute

\[
\begin{align*}
\theta_d(t) &= \text{Atan2}(\dot{y}_d(t), \dot{x}_d(t)) + k\pi \quad k = 0, 1 \\
v_d(t) &= \pm \sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)} \\
\omega_d(t) &= \frac{\dot{y}_d(t) \dot{x}_d(t) - \ddot{x}_d(t) \dot{y}_d(t)}{\dot{x}_d^2(t) + \dot{y}_d^2(t)}
\end{align*}
\]
• the desired state trajectory can be used to compute the **state error**, from which the **feedback action** is generated; whereas the nominal input can be used as a **feedforward term**

• the resulting block scheme is
• rather than using directly the state error \( q_d - q \), use its rotated version defined as

\[
e = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{pmatrix}
\]

\((e_1, e_2)\) is \(e_p\) (previous figure) in a frame rotated by \(\theta\)

• the error dynamics is nonlinear and time-varying

\[
\dot{e}_1 = v_d \cos e_3 - v + e_2 \omega \\
\dot{e}_2 = v_d \sin e_3 - e_1 \omega \\
\dot{e}_3 = \omega_d - \omega
\]
via approximate linearization

- A simple approach for stabilizing the error dynamics is to use its linearization around the reference trajectory (indirect Lyapunov method $\Rightarrow$ local results)

- First, to make the reference trajectory an unforced equilibrium for the error dynamics

\[
\begin{align*}
\dot{e}_1 &= \nu_d \cos e_3 - \nu + e_2 \omega \\
\dot{e}_2 &= \nu_d \sin e_3 - e_1 \omega \\
\dot{e}_3 &= \omega_d - \omega
\end{align*}
\]

use the following (invertible) input transformation

\[
\begin{align*}
u_1 &= \nu_d \cos e_3 - \nu \\
u_2 &= \omega_d - \omega
\end{align*}
\]
\[ \begin{align*}
\dot{e}_1 &= \omega_d e_2 + u_1 - e_2 u_2 \\
\dot{e}_2 &= -\omega_d e_1 + v_d \sin e_3 + e_1 u_2 \\
\dot{e}_3 &= u_2
\end{align*} \]

that is, \( \dot{e} = \varphi(e, u, t) \) with \( \varphi(0, 0, t) = 0 \)

\[ \varphi(e, u, t) = \begin{pmatrix} \omega_d e_2 \\ -\omega_d e_1 + v_d \sin e_3 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \]

- drift term nonlinear, time-varying
- input term nonlinear, linear in \( u \)
• since

\[ \varphi(e, u, t) \approx \varphi(0, 0, t) + \left. \frac{\partial \varphi}{\partial e} \right|_{e=0} e + \left. \frac{\partial \varphi}{\partial u} \right|_{e=0} u \]

the linear approximation of the error dynamics is

\[ \dot{e} = \varphi(e, u, t) \approx \left. \frac{\partial \varphi}{\partial e} \right|_{e=0} e + \left. \frac{\partial \varphi}{\partial u} \right|_{e=0} u = A(t)e + Bu \]

• one easily finds

\[ \frac{\partial \varphi}{\partial e} = \begin{pmatrix} 0 & \omega_d - u_2 & 0 \\ -\omega_d + u_2 & 0 & v_d \cos e_3 \\ 0 & 0 & 0 \end{pmatrix} \quad \frac{\partial \varphi}{\partial u} = \begin{pmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{pmatrix} \]
• hence, the linearized error dynamics around the reference trajectory is written as

\[
\dot{e} = \begin{pmatrix}
0 & \omega_d & 0 \\
-\omega_d & 0 & v_d \\
0 & 0 & 0
\end{pmatrix} e + \begin{pmatrix}
1 & 0 \\
0 & 0 \\
0 & 1
\end{pmatrix} \begin{pmatrix}
u_1 \\
u_2
\end{pmatrix}
\]

• now define the linear feedback

\[
u = K e = \begin{pmatrix}
-k_1 & 0 & 0 \\
0 & -k_2 & -k_3
\end{pmatrix} \begin{pmatrix}
e_1 \\
e_2 \\
e_3
\end{pmatrix}
\]

• the closed-loop error dynamics is still time-varying!

\[
\dot{e} = (A(t) + BK)e = A_{cl}(t)e = \begin{pmatrix}
-k_1 & \omega_d & 0 \\
-\omega_d & 0 & v_d \\
0 & -k_2 & -k_3
\end{pmatrix} e
\]
letting

\[ k_1 = k_3 = 2\zeta a \quad k_2 = \frac{a^2 - \omega_d^2}{v_d} \]

with \( a > 0, \zeta \in (0,1) \), the characteristic polynomial of \( A(t) \) becomes time-invariant and Hurwitz

\[ p(\lambda) = (\lambda + 2\zeta a)(\lambda^2 + 2\zeta a\lambda + a^2) \]

- real
- negative eigenvalue

- pair of complex
- eigenvalues with negative real part

**caveat:** this does **not** guarantee asymptotic stability, unless \( v_d \) and \( \omega_d \) are constant (rectilinear and circular trajectories); even in this case, asymptotic stability of the unicycle is **not global** (indirect Lyapunov method)
• the actual velocity inputs $v, \omega$ are obtained plugging the feedbacks $u_1, u_2$ in the input transformation

• note: $(v, \omega) \rightarrow (v_d, \omega_d)$ as $e \rightarrow 0$ (pure feedforward)

• note: $k_2 \rightarrow \infty$ as $v_d \rightarrow 0$, hence this controller can only be used with persistent Cartesian trajectories (stops are not allowed)

• global stability is guaranteed by a nonlinear version

$$u_1 = -k_1(v_d, \omega_d) e_1$$
$$u_2 = -k_2 v_d \frac{\sin e_3}{e_3} e_2 - k_3(v_d, \omega_d) e_3$$

if $k_1, k_3$ bounded, positive, with bounded derivatives
• the final block scheme for trajectory tracking via state error feedback and approximate linearization is

![Block diagram](image)

- based on state error
- needs $v_d, \omega_d$
- needs $\theta$ also for error rotation + input transformation
trajectory tracking: output error feedback

• another approach: develop the feedback action from the output (Cartesian) error only, without computing a desired state trajectory, while the feedforward term is the velocity along the reference trajectory

• the resulting block scheme is
exact i/o linearization: brush-up

• consider a driftless nonlinear system

\[
\dot{x} = G(x)u \\
y = h(x)
\]

\[x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m, \quad y \in \mathbb{R}^m\]

• being

\[
\dot{y} = \frac{\partial h}{\partial x} \dot{x} = \frac{\partial h}{\partial x} G(x)u = T(x)u
\]

if the \( m \times m \) decoupling matrix \( T \) is invertible we set

\[u = T^{-1}(x)v \quad \text{obtaining} \quad \dot{y} = T(x)T^{-1}(x)v = v\]

i.e., an exactly linear map between the inputs and the (time derivative of) the outputs
• in pictures

\[ u = T^{-1}(x)v \]

under the action of the linearizing feedback
\[ u = T^{-1}(x)v \]

the system behaves as

...a simple (vector) integrator from \( v \) to \( y \)

• given a reference output \( y_d(t) \), the dynamics of the output error \( e = y_d - y \) is
\[ \dot{e} = \dot{y}_d - \dot{y} = \dot{y}_d - v \]

• let \( v = \dot{y}_d + Ke \) (feedforward + proportional feedback) to obtain
\[ \dot{e} = -Ke \]

i.e., global exponential stability provided that the eigenvalues of \( K \) are in the rhp
trajectory tracking via i/o linearization

• let us adopt the exact i/o linearization approach to design a Cartesian trajectory tracking controller for the unicycle

• however, in this case the decoupling matrix turns out to be singular

\[
\begin{pmatrix}
\dot{x} \\
\dot{y}
\end{pmatrix} = 
\begin{pmatrix}
\cos \theta & 0 \\
\sin \theta & 0
\end{pmatrix}
\begin{pmatrix}
v \\
\omega
\end{pmatrix}
\]

as a consequence, exact input-output linearization is not possible for the output \((x, y)\)
• solution: change slightly the output so that the new input-output map is invertible and exact linearization becomes possible

• displace the output from the contact point of the wheel to point $B$ along the sagittal axis

\begin{align*}
y_1 &= x + b \cos \theta \\
y_2 &= y + b \sin \theta
\end{align*}
• differentiating wrt time

\[
\begin{pmatrix}
\dot{y}_1 \\
\dot{y}_2
\end{pmatrix} = \begin{pmatrix}
cos \theta & -b \sin \theta \\
\sin \theta & b \cos \theta
\end{pmatrix} \begin{pmatrix}
u \\
\omega
\end{pmatrix} = T(\theta) \begin{pmatrix}
u \\
\omega
\end{pmatrix}
\]

Determinant $= b$

• if $b \neq 0$, we may set

\[
\begin{pmatrix}
u \\
\omega
\end{pmatrix} = T^{-1}(\theta) \begin{pmatrix}
u_1 \\
u_2
\end{pmatrix} = \begin{pmatrix}
cos \theta & \sin \theta \\
-sin \theta/b & \cos \theta/b
\end{pmatrix} \begin{pmatrix}
u_1 \\
u_2
\end{pmatrix}
\]

obtaining

\[
\begin{align*}
\dot{y}_1 &= u_1 \\
\dot{y}_2 &= u_2 \\
\dot{\theta} &= \frac{u_2 \cos \theta - u_1 \sin \theta}{b}
\end{align*}
\]
• achieve global exponential convergence of \( y_1, y_2 \) to the desired trajectory letting

\[
\begin{align*}
u_1 &= \dot{y}_{1d} + k_1(y_{1d} - y_1) \\
u_2 &= \dot{y}_{2d} + k_2(y_{2d} - y_2)
\end{align*}
\]

with \( k_1, k_2 > 0 \)

• \( \theta \) is not controlled with this scheme, which is based on output error feedback (compare with the previous)

• the desired trajectory for \( B \) can be arbitrary; in particular, square corners may be included
• the final block scheme for trajectory tracking via output error feedback + input-output linearization is

![Block diagram](image)

- based on output error
- needs $\dot{p}_d$
- needs $x, y, \theta$ for output reconstruction and $\theta$ also for input transformation
simulations

tracking a circle via approximate linearization
simulations

tracking an 8-figure via nonlinear feedback
tracking a square via i/o linearization

\[ b = 0.75 \Rightarrow \text{the unicycle rounds the corners} \]
tracking a square via i/o linearization

\[ b = 0.2 \Rightarrow \text{accurate tracking but velocities increase} \]