Robot Kinematic Constraints

- The kinematic constraints on a robot come from the combination of constraints from its wheels
- Castor, Swedish, and spherical wheels impose no constraints
- Therefore we consider only constraints from fixed and steerable standard wheels
- Consider a robot with $N_f$ fixed and $N_s$ steerable wheels ($N = N_f + N_s$ number of standard wheels)
- Incorporate wheel information into vectors
  - $B_f$: orientation ($\beta$ angles) of fixed wheels
  - $B_s(t)$: orientation of steerable wheels at time $t$
  - $\Phi_f(t)$: wheel roll angle of fixed wheels at time $t$
  - $\Phi_s(t)$: wheel roll angle of steerable wheels at time $t$
- Let,$$
\Phi(t) = \begin{bmatrix}
\Phi_f(t) \\
\Phi_s(t)
\end{bmatrix}
$$

We can then combine the rolling constraints for all standard wheels,
$$
\begin{bmatrix}
\sin(\alpha_1 + \beta_1) & -\cos(\alpha_1 + \beta_1) & (-l_1)\cos(\beta_1) \\
\vdots & \vdots & \vdots \\
\sin(\alpha_N + \beta_N) & -\cos(\alpha_N + \beta_N) & (-l_N)\cos(\beta_N)
\end{bmatrix}R(\theta)\dot{\xi}_I =
\begin{bmatrix}
r_1 & \ldots & 0 \\
\vdots & \vdots & \vdots \\
0 & \ldots & r_N
\end{bmatrix}\Phi
$$

We denote the big matrix on the left $J_1(B_s)$; the matrix on the right is $J_2$

$J_1(B_s)R(\theta)\dot{\xi}_I = J_2\Phi$

$J_1(B_s)$ is $N \times 3$ with the components from fixed wheels in the top $N_f$ rows and those from steerable wheels below

$J_1(B_s) = \begin{bmatrix}
J_{1f} \\
J_{1s}(B_s)
\end{bmatrix}$

It is a function of $B_s$ ($B_f$ is a constant vector)

$J_2$ is an $N \times N$ diagonal matrix
We do the same sort of combination for the sliding constraints,
\[
\begin{bmatrix}
\cos(\alpha_1 + \beta_1) & \sin(\alpha_1 + \beta_1) & l_1 \sin(\beta_1) \\
\vdots & \ddots & \vdots \\
\cos(\alpha_N + \beta_N) & \sin(\alpha_N + \beta_N) & l_N \sin(\beta_N)
\end{bmatrix}
R(\theta) \dot{\xi}_I = 0
\]
We denote the matrix on the left \( C_1(B_s) \); On the R.H.S. is the zero vector. \( C_1(B_s) \) is \( N \times 3 \) with the components from fixed wheels in the top \( N_f \) rows and those from steerable wheels below
\[
C_1(B_s) = \begin{bmatrix} C_{1f} \\ C_{1s}(B_s) \end{bmatrix}
\]
We can combine our two big equations (summarized on board) into one,
\[
\begin{bmatrix} J_1(B_s) \\ C_1(B_s) \end{bmatrix} R(\theta) \dot{\xi}_I = \begin{bmatrix} J_2 \Phi \\ 0 \end{bmatrix}
\]
Assume that the two wheels are equidistant from \( P \); Let \( l = l_r = l_l \)
\[
J_{1f} = \begin{bmatrix} \sin(\alpha_r + \beta_r) & -\cos(\alpha_r + \beta_r) & (-l) \cos(\beta_r) \\
\sin(\alpha_l + \beta_l) & -\cos(\alpha_l + \beta_l) & (-l) \cos(\beta_l) \\
1 & 0 & l \\
1 & 0 & -l
\end{bmatrix}
\]
\[
J_2 = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}
\]
\[
C_{1f} = \begin{bmatrix} \cos(\alpha_r + \beta_r) & \sin(\alpha_r + \beta_r) & l \sin(\beta_r) \\
\cos(\alpha_l + \beta_l) & \sin(\alpha_l + \beta_l) & l \sin(\beta_l) \\
0 & 1 & 0 \\
0 & 1 & 0
\end{bmatrix}
\]
Our overall equation is,
\[
\begin{bmatrix} 1 & 0 & l \\ 1 & 0 & -l \\ 0 & 1 & 0 \\
0 & 1 & 0 \end{bmatrix} R(\theta) \dot{\xi}_I = \begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_l \\ 0 \end{bmatrix}
\]
We solve for \( \dot{\xi}_R = R(\theta) \dot{\xi}_I \)

COMP 4766/6778 (MUN) Kinematics of Wheeled Robots: Part 2 January 30, 2009 5 / 40

Example: A Differential-Drive Robot

- Our D-D robot has two FSW’s, plus a Castor wheel for stability (plays no part in the analysis)
- The parameters of the two FSW’s are as follows:
  - Right wheel: \( \alpha_r = -\frac{\pi}{2} \), \( \beta_r = \pi \) (+ve spin should cause movement in + X_R direction)
  - Left wheel: \( \alpha_l = \frac{\pi}{2} \), \( \beta_l = 0 \)
- We now use these parameters to compute \( J_1(B_s) \) and \( C_1(B_s) \)
- We have fixed wheels, therefore \( J_1(B_s) = J_{1f} \) and \( C_1(B_s) = C_{1f} \)

COMP 4766/6778 (MUN) Kinematics of Wheeled Robots: Part 2 January 30, 2009 6 / 40

compass on board
As an alternative to the procedure above, we can observe that the fourth row of the matrix is redundant and can be removed to yield a square matrix:

\[
\begin{bmatrix}
1 & 0 & l \\
1 & 0 & -l \\
0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
r \\
\dot{\phi}_r \\
\dot{\phi}_l
\end{bmatrix}
= r
\begin{bmatrix}
\dot{\phi}_r \\
\dot{\phi}_l \\
0
\end{bmatrix}
\]

we can invert the matrices on the left to obtain the forward kinematic model,

\[
\dot{\xi} = r \mathbf{R}(\theta)^{-1}
\begin{bmatrix}
1 & 0 & l \\
1 & 0 & -l \\
0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
\dot{\phi}_r \\
\dot{\phi}_l \\
0
\end{bmatrix}
= \ldots
\]

Consider how the robot moves for \(\dot{\phi}_l = \dot{\phi}_r\) and \(\dot{\phi}_l = -\dot{\phi}_r\).

\[
\begin{bmatrix}
0 & 1 & 0 \\
-\sqrt{2}/2 & \sqrt{2}/2 & \sqrt{2}/2 \\
-1 & 0 & -1 \\
-\sqrt{2}/2 & -\sqrt{2}/2 & \sqrt{2}/2
\end{bmatrix}
\begin{bmatrix}
r \\
\dot{\phi}_1 \\
\dot{\phi}_2 \\
0
\end{bmatrix}
= r
\begin{bmatrix}
\dot{\phi}_1 \\
\dot{\phi}_2 \\
0
\end{bmatrix}
\]

Notice that this matrix cannot be inverted; Nor are there any duplicate equations that can be thrown away; We apply Gauss-Jordan elimination to determine both the solution, and the condition on the existence of the solution:

\[
\begin{bmatrix}
0 & 1 & 0 \\
-\sqrt{2}/2 & \sqrt{2}/2 & \sqrt{2}/2 \\
-1 & 0 & -1 \\
-\sqrt{2}/2 & -\sqrt{2}/2 & \sqrt{2}/2
\end{bmatrix}
\begin{bmatrix}
r \dot{\phi}_1 \\
r \dot{\phi}_2 \\
0 \\
0
\end{bmatrix}
\rightarrow \text{row exchanges and combinations}
\]

\[
\begin{bmatrix}
1 & 0 & 0 & -r \dot{\phi}_1/2 \\
0 & 1 & 0 & r \dot{\phi}_1/2 \\
0 & 0 & 1 & r \dot{\phi}_1/2
\end{bmatrix}
\]

Yielding the following:

\[
\dot{x}_R = -r \dot{\phi}_1/2, \quad \dot{y}_R = r \dot{\phi}_1, \quad \dot{\theta} = r \dot{\phi}_1/2,
\]

\[
\dot{\phi}_1 = \frac{\sqrt{2}}{2} \dot{\phi}_2
\]

This last equation is a condition on the existence of solutions; Unlike a differential drive robot, the two wheel speeds here cannot be set arbitrarily.

**Example: A Turning Bicycle**

A bicycle with its front wheel locked in a left turn:

\[\mathbf{J}_{1f} = \begin{bmatrix}
\sin(\alpha_1 + \beta_1) & -\cos(\alpha_1 + \beta_1) & (-h_1) \cos(\beta_1) \\
\sin(\alpha_2 + \beta_2) & -\cos(\alpha_2 + \beta_2) & (-h_2) \cos(\beta_2)
\end{bmatrix} = \begin{bmatrix}
0 & 1 & 0 \\
-\sqrt{2}/2 & \sqrt{2}/2 & \sqrt{2}/2
\end{bmatrix}\]

\[\mathbf{J}_{2} = \begin{bmatrix}
r & 0 \\
0 & r
\end{bmatrix}\]

\[\mathbf{C}_{1f} = \begin{bmatrix}
\cos(\alpha_1 + \beta_1) & \sin(\alpha_1 + \beta_1) & h_1 \sin(\beta_1) \\
\cos(\alpha_2 + \beta_2) & \sin(\alpha_2 + \beta_2) & h_2 \sin(\beta_2)
\end{bmatrix} = \begin{bmatrix}
-1 & 0 & -1 \\
-\sqrt{2}/2 & -\sqrt{2}/2 & \sqrt{2}/2
\end{bmatrix}\]

After a number of steps, we arrive at,

\[
\begin{bmatrix}
r \dot{\phi}_1 \\
\dot{\phi}_2 \\
0
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
r \dot{\phi}_1 \\
\dot{\phi}_2 \\
0
\end{bmatrix} + \begin{bmatrix}
0 \\
-\sqrt{2}/2 & \sqrt{2}/2 & \sqrt{2}/2 \\
-1 & 0 & -1 \\
-\sqrt{2}/2 & -\sqrt{2}/2 & \sqrt{2}/2
\end{bmatrix}
\begin{bmatrix}
r \dot{\phi}_1 \\
\dot{\phi}_2 \\
0
\end{bmatrix}
\]

Yielding the following:

\[
\dot{x}_R = -r \dot{\phi}_1/2, \quad \dot{y}_R = r \dot{\phi}_1, \quad \dot{\theta} = r \dot{\phi}_1/2,
\]

\[
\dot{\phi}_1 = \frac{\sqrt{2}}{2} \dot{\phi}_2
\]
Using the Forward Kinematic Equation

- Localization:
  - One use for the forward kinematic equation is to allow a robot’s current pose to be tracked (localization).
  - Let us say we know $\xi_i(t)$ for the previous time step and we wish to determine the pose for current time $t'$.
  - From the motors’ optical encoders we can get an estimate of the wheels’ current roll speeds: $\phi_1, \phi_2, \ldots$.
  - Using the forward kinematic equation we obtain: $\dot{\xi}_R$.
  - We can obtain $\dot{\xi}_I$ using our current estimate for $\theta$.
  - We apply a first-order Taylor series expansion:
    $$\xi_i(t') = \xi_i(t) + (t' - t)\dot{\xi}_i(t) + \cdots$$
    (The “…” represents higher-order terms that we don’t bother to include in a first-order approximation)
  - This equation can be applied iteratively to localize the robot over time. However, it will certainly drift as time passes.

- A robot’s instantaneous velocity must be tangential to the circle centred at its ICR.
  - Recall the translatory component of the turning bicycle’s velocity:
    $$x_R = -r\dot{\phi}/2, \quad y_R = r\dot{\phi}_1$$
  - For any given $r$ and $\dot{\phi}_1$ the velocity vector (heavy vector below) will be orthogonal to the line connecting $P$ and ICR.

Manoeuvrability: Degree of Mobility

- Some robots are more manoeuvrable than others.
  - Intuitively, we can see that the differential drive robot is more manoeuvrable than the turning bicycle.
- A robot’s degree of mobility is defined by its sliding constraints.
- Sliding constraints can be visualized by drawing a zero motion line through the wheel’s axis, perpendicular to the wheel plane.
- The intersection of all zero motion lines defines the instantaneous centre of rotation (ICR).

ICR’s for Various Wheel Configurations

- Left: For a differential-drive the two zero-motion lines are coincident; Thus, the ICR is constrained only to lie somewhere on that line.
- Centre: For an Ackerman configuration (e.g. a car) the two rear wheels give only one zero-motion line; To prevent slipping, the two front wheels must be steered such that their zero motion lines intersect the rear line at a common point.
- Right: A degenerate configuration; There is no ICR; If there is no slipping, there is also no movement.
We can formally characterize a robot’s degree of mobility.

Consider a differential-drive robot; it has two wheels but only one independent sliding constraint.

To determine the degree of mobility we count the number of independent sliding constraints.

The matrix $C_1(B_s)$ gives the sliding constraints.

The rank of this matrix is the number of independent constraints.

- $rank = number$ of independent rows or columns (smaller of the two).

Examples:
- Differential-drive:
  
  $C_1(B_s) = C_{1f} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$

  $rank [C_1(B_s)] = 1$

- Turning bicycle:

  $C_1(B_s) = C_{1f} = \begin{bmatrix} -1 & 0 & -1 \\ -\sqrt{2}/2 & -\sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$

  $rank [C_1(B_s)] = 2$

The maximum rank of an $N \times 3$ matrix is $\text{___}$.

We define a robot’s **degree of mobility** as follows,

$$\delta_m = 3 - rank \ [C_1(B_s)]$$

- Differential-drive: $\delta_m = 2$
- Turning bicycle: $\delta_m = 1$
- Robot with all omnidirectional wheels: $\delta_m = 3$

Determining $\delta_m$ is an important part of determining how manoeuvrable a robot is; however, the fact that some wheels are steerable should also be considered...

We define a robot’s **degree of steerability**, $\delta_s$, as the number of independently steerable wheels that yield a valid ICR.

- A normal bicycle: $\delta_s = 1$
- A car: $\delta_s = 1$ (cannot independently steer both front wheels)
- The maximum $\delta_s$ is 2: Once two wheels define the ICR, the choice of the third is not independent.

We define a robot’s **degree of manoeuvrability** as follows,

$$\delta_M = \delta_m + \delta_s$$
The term holonomic robot refers to a robot with either...
- ...constraints which can be expressed purely in terms of position variables — e.g. the following locked-wheel bicycle

\[ y = 0 \text{ and } \theta = 0 \]

...or no constraints on its motion: an omnidirectional robot

Nonholonomic robots are subject to nonholonomic constraints (i.e. sliding constraints)

Omnidirectional robots have \( \delta_M = 3 \) and exhibit the best possible manoeuvrability; However, the omnidirectional wheels required for such robots (i.e. Swedish, Castor, or spherical) exhibit a number of drawbacks:
- increased complexity and expense
- reduced accuracy for dead reckoning
- reduced ground clearance for powered versions
- standard wheels can passively counteract lateral forces; more efficient and stable for high-speed turns

What is the difference between an omnidirectional robot, and a non-omnidirectional robot such as the Two-Steer? Both have \( \delta_M = 3 \)?

The difference is that it takes time for a steered robot to steer its wheels to the appropriate positions; Consider this omnidirectional robot following an ‘L’ shaped trajectory

Now consider the trajectory of the Two-Steer

During time intervals 1-2 and 3-4 the robot was doing nothing but steering its wheels; The omnidirectional robot could transition between segments of the trajectory without any delay

Both robots take the same path but the trajectories (path + time dimension) differ
Motion Control: Trajectory Following

- One simple means of controlling the motion of a robot is to decompose its path into a sequence of elementary motions.
- Elementary motions may include lines and segments of circles which any robot with $\delta_M \geq 2$ can execute.

The robot’s trajectory can be planned completely in advance without using any sensors → Open-loop control

- Alternatively, information from the sensors can be used to update the plan → Closed-loop control

Open-loop systems cannot correct for disturbances. In our example, the final temperature would deviate from the desired temperature.

- e.g. A toaster is an open-loop system. It cannot correct for the thickness of the bread or whether it is whole wheat or white.

Open-Loop System:

The input (a.k.a. reference) is first converted by the input transducer to the form required by the controller. The process or plant carries out the core function of the system (e.g. the furnace in a heating system, motors in a robot). The output (a.k.a. the controlled variable) differs from its desired value because of the two disturbances.

- e.g. Open-loop heating system: The controller is an electronic amplifier and disturbance 1 is noise in the amplifier’s output. Disturbance 2 might be variations in temperature due to the furnace itself.

Closed-Loop System:

In a closed-loop system there is an output transducer or sensor which converts the output into the form used by the controller. e.g. Position can be converted to an electrical signal by a potentiometer.

- The first summing junction subtracts the output signal from the input. This is the error signal.
Closed-loop systems compensate for disturbances through **feedback**. If the actuating signal is zero then the output is correct and the plant does not need to be driven. Otherwise, the actuating signal describes how different the output is from what it should be. This drives the plant to correct this difference.

While open-loop systems fail to correct for disturbances or changes in the environment, they will tend to be simpler and cheaper than closed-loop systems. Thus, there is a trade-off to consider between them.

Closed-Loop Control

- Consider the problem of driving a differential-drive robot to goal position \( \mathbf{g}_I = [g_I x, g_I y]^T \), expressed in the global reference frame.
- (Later we will consider the problem of arriving at the goal position with a particular orientation.)
- We need to determine how to set the robot’s forward speed \( \nu(t) \) and rotational speed \( \omega(t) \).
- For a differential-drive robot we have the following,
  \[
  \nu(t) = \dot{x}_R = \frac{r(\dot{\phi}_r + \dot{\phi}_l)}{2}
  \]
  \[
  \omega(t) = \dot{\theta} = \frac{r(\dot{\phi}_r - \dot{\phi}_l)}{2l}
  \]

...Some Details

- If we can obtain \( \mathbf{g}_R \) then we can apply some control function \( f \) to get \( \nu(t) \), the forward velocity component, and \( \omega(t) \), the angular velocity component.
  \[
  \begin{bmatrix}
  \nu(t) \\
  \omega(t)
  \end{bmatrix} = f(\mathbf{g}_R)
  \]
- The control function should drive the robot such that,
  \[
  \lim_{t \to \infty} \mathbf{g}_R(t) = [0, 0]^T
  \]
  which just means that the robot will eventually reach the goal.

- We are given \( \mathbf{g}_I \); How do we determine \( \mathbf{g}_R \)?
- **COVERED ON BOARD**
Two-step Controller

- We break the problem into two steps:
  - Turn to face the goal
  - Move towards goal
- First, it is convenient to express $\mathbf{g}_R$ using polar coordinates $[\rho \ \alpha]^T$

The two steps can now be specified:
- Minimize $\alpha$
- Minimize $\rho$

There are problems with this controller:
- If the first step fails, the second will also fail
- It is difficult to choose appropriate values for the parameters: $k_\alpha, \epsilon_\alpha, k_\rho, \epsilon_\rho$
  - Smaller thresholds require high-precision localization and actuation (if too small, goal is never reached)
  - Larger thresholds reduce accuracy
- Splitting the motion into two distinct phases is inefficient; We can save time by moving forwards while turning

The Controller: Two States

parameters: $k_\alpha, \epsilon_\alpha, k_\rho, \epsilon_\rho$

- \[
\begin{bmatrix}
  v(t) \\
  \omega(t)
\end{bmatrix} = \begin{bmatrix}
  0 \\
  k_\alpha \ \text{sign}(\alpha)
\end{bmatrix}
\]
  Switch to state 2 if $|\alpha| < \epsilon_\alpha$

- \[
\begin{bmatrix}
  v(t) \\
  \omega(t)
\end{bmatrix} = \begin{bmatrix}
  k_\rho \\
  0
\end{bmatrix}
\]
  End if $\rho < \epsilon_\rho$

Note: angle $\alpha$ must be in $[-\pi, \pi]$

Smooth Controller 1

- We try to minimize the quantities $\mathbf{g}_{Rx}, \mathbf{g}_{Ry}$, but now we minimize both simultaneously
- Consider the following control law
  \[
  v(t) = k_v \mathbf{g}_{Rx}
  \]
  \[
  \omega(t) = k_\omega \mathbf{g}_{Ry}
  \]
  where the $k$ parameters are positive and $\alpha \in [-\pi, \pi]$

- The robot drives forward until $\mathbf{g}_{Rx} = 0$
- If $\mathbf{g}_{Ry}$ is positive, robot will turn CCW to face the goal; If negative it will turn CW.
- DEMO
Assume we now wish to drive the robot to a desired pose. Pose means \((x, y)\) position and orientation \(\theta\).

We are given the goal pose \(g_I = [g_{Ix} \ g_{Iy} \ g_{I\theta}]^T\), expressed in the inertial reference frame.

We need the goal pose in the robot reference frame

\[ g_R = R(\theta) (g_I - \xi_I) \]

(We should use the 3 \(\times\) 3 rotation matrix here.)

Again it will be useful to express the goal pose using polar coordinates

The final orientation is given by \(g_{R\theta}\).

We require a controller that minimizes \(\alpha\), \(\rho\), and \(g_{R\theta}\) simultaneously.

Consider the following control law

\[
\begin{align*}
v(t) &= k_\rho \rho \\
\omega(t) &= k_\alpha \alpha - k_\theta g_{R\theta}
\end{align*}
\]

where the following conditions hold:

- \(\alpha \in [-\pi, \pi]\)
- all of the \(k\) parameters are positive
- \(k_\theta < k_\alpha\)

The robot drives forward until \(\rho = 0\).

If \(\alpha\) is positive, robot will turn CCW to minimize it.

\(k_\theta < k_\alpha\) so \(\theta\) does not have much influence until \(\alpha\) becomes small; At this point the robot will be driven to turn away from the goal; This increases \(\alpha\) so the robot will turn towards the goal again, only now \(g_{R\theta}\) will be reduced.

If \(\alpha \in (-\frac{\pi}{2}, \frac{\pi}{2})\) then the robot will approach the goal directly (although its trajectory will be curved).

If \(\alpha \in (-\pi, -\frac{\pi}{2}] \cup [\frac{\pi}{2}, \pi]\) then the robot will first have to turn around before approaching the goal; We can detect this situation and modify the control law so that the robot backs up to the goal position, without turning around.

\[
\begin{align*}
v(t) &= -k_\rho \rho \\
\omega(t) &= -k_\alpha (\alpha - \pi) - k_\theta g_{R\theta}
\end{align*}
\]

(Here the angle \((\alpha - \pi)\) must be in \([-\pi, \pi]\))