# Ch. 13 Wheeled Mobile Robots

7.11.23

Kinematic Model: map wheel speeds to robot velocities
Dynamic Model: map wheel torques to robot accelerations

Assumption: Robot with single rigid-body chassis
- config: $T_{sb} \in SE(2)$: representing chassis frame $\{b\}$ relative to fixed space frame $\{s\}$
- $T_{sb}$ can be represented as $q = (\phi, x, y)$
  - velocity of chassis: time derivative $\dot{q} = (\dot{\phi}, \dot{x}, \dot{y})$
- Alternatively: chassis planar twist $V_b = (\omega_{bz}, v_{bx}, v_{by})$ expressed in $\{b\}$

with:

$$V_b = \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \cdot \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \cdot \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix}$$

## 13.1 Types of wheeled Mobile Robots

Two major categories:
- omnidirectional: no equality constraint on chassis velocity $\dot{q}$
- nonholonomic: subject to single Pfaffian constraint $A(q)\dot{q} = 0$
  - velocity constraint can not be integrated to configuration constraint $\Rightarrow$ nonholonomic constraint

### Wheels for omnidirectional mobile robots:
- omniwheels
- mecanum wheels

### Kinematic wheeled mobile robots
- differential drive
- car-like / Ackermann steering
- omnidirectional

## 13.2. Omnidirectional Wheeled Mobile Robots

### Modeling

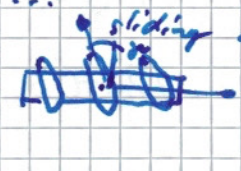
Omnidirectional Robot needs at least 3 wheels to achieve arbitrary velocity $\dot{q} = (\dot{\phi}, \dot{x}, \dot{y})$

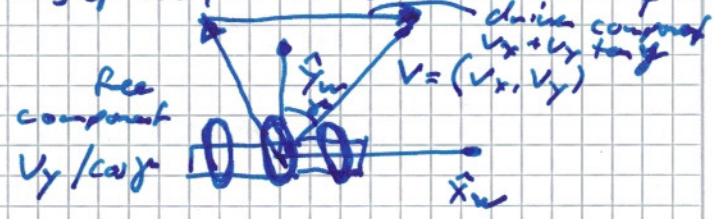
Kinematic modeling:

1.) given chassis velocity $\dot{q}$ what speed must the wheels be driving

2.) given limits of wheel driving speeds, what is the limit on $\dot{q}$

...




$\gamma$ : · omniwheel $\gamma = 0$
· mecanum wheel $\gamma = \pm 45°$

$$\Rightarrow \begin{bmatrix} v_x \\ v_y \end{bmatrix} = v_{drive} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + v_{slide} \begin{bmatrix} -\sin\gamma \\ \cos\gamma \end{bmatrix}$$

"sliding": motion allowed by the passive rollers on circumference of the wheels

$$\Rightarrow \quad v_{drive} = v_x + v_y \tan\gamma$$

$$v_{slide} = v_y / \cos\gamma$$

With: radius of wheel: $r$

· driving angular speed of wheel: $u$

$$u = \frac{v_{drive}}{r} = \frac{1}{r}(v_x + v_y \tan\gamma)$$

From chassis velocity $\dot{q}$ to angular speed $u_i$ for wheel $i$:

$u_i = h_i(\phi) \cdot \dot{q} =$ · center and driving direction: $(\beta_i, x_i, y_i)$ · chassis frame $\{b\}$ is at $q = (\phi, x, y)$

$$\begin{bmatrix} \frac{1}{r_i} & \frac{\tan\gamma_i}{r_i} \end{bmatrix} \begin{bmatrix} \cos\beta_i & \sin\beta_i \\ -\sin\beta_i & \cos\beta_i \end{bmatrix} \begin{bmatrix} -y_i & 1 & 0 \\ x_i & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \dot{q}$$

calculate driving ang. velocity

express the linear velocity in wheel frame $\hat{x}_w$-$\hat{y}_w$

linear velocity at wheel in $\{b\}$

express $\dot{q}$ as $V_b$

12.11.20

$$h_i(\phi) = \frac{1}{r_i \cos\gamma_i} \begin{bmatrix} x_i \sin(\beta_i + \gamma_i) - y_i \cos(\beta_i + \gamma_i) \\ \cos(\beta_i + \gamma_i + \phi) \\ \sin(\beta_i + \gamma_i + \phi) \end{bmatrix}^T$$

omnidirectional Robot with $m \geq 3$ wheels:

$H(\phi) \in \mathbb{R}^{m \times 3}$ : mapping desired chassis velocity $\dot{q}$ to vector of driving speeds $u$

$u \in \mathbb{R}^m$ is constructed by stacking $m$ rows $h_i(\phi)$:

$$u = H(\phi)\,\dot{q} = \begin{bmatrix} h_1(\phi) \\ h_2(\phi) \\ \vdots \\ h_m(\phi) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

with $\dot{q} \in \mathbb{R}^3$, $u \in \mathbb{R}^m$, $H(\phi) \in \mathbb{R}^{m \times 3}$

Mapping between $u$ and body twist $V_b$ (does not depend on chassis orientation $\phi$)

$$u = H(0)\,V_b = \begin{bmatrix} h_1(0) \\ h_2(0) \\ \vdots \\ h_m(0) \end{bmatrix} \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix}$$

$H(0)$ needs to be rank 3

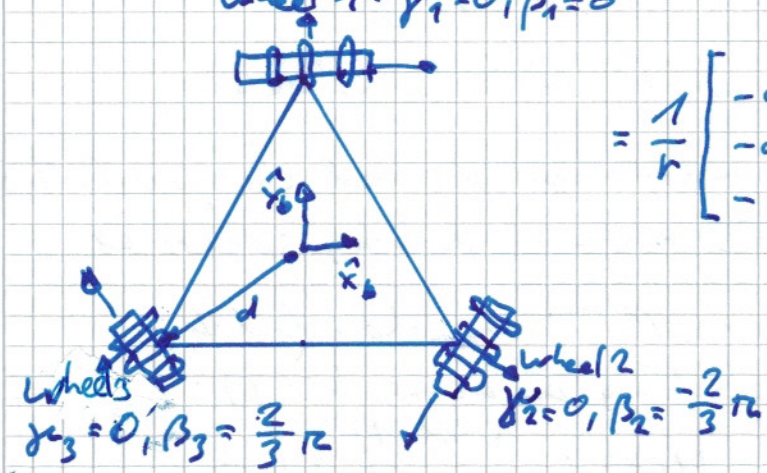
→ need to choose appropriate wheel positions and headings $(\beta_i, x_i, y_i)$

# Examples of kinematic models

## Omni-wheel robot:

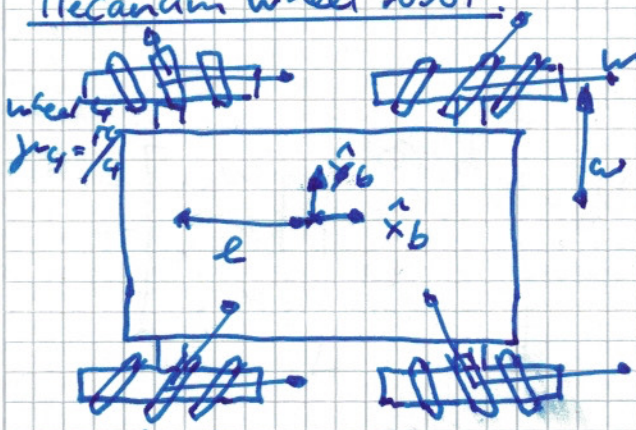
$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = H(0) V_b$$

$$= \frac{1}{r} \begin{bmatrix} -d & 1 & 0 \\ -d & -1/2 & -\sin(\pi/3) \\ -d & -1/2 & \sin(\pi/3) \end{bmatrix} \begin{bmatrix} \omega_{bz} \\ V_{bx} \\ V_{by} \end{bmatrix}$$




r: radius of wheels

## Mecanum wheel robot:




the wheels need to be arranged in this way:

$\gamma_{1,3} = -\pi/4$

$\gamma_{2,4} = \pi/4$

↳ $H(0)$: full rank

r: radius of wheels

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = H(0) V_b = \frac{1}{r} \begin{bmatrix} -l-w & 1 & -1 \\ l+w & 1 & 1 \\ l+w & 1 & -1 \\ -l-w & 1 & 1 \end{bmatrix} \begin{bmatrix} \omega_{bz} \\ V_{bx} \\ V_{by} \end{bmatrix}$$

(c) (a) (b)

For robot to move in:

(a) $+\hat{x}_b$ direction: all wheels drive forward at same speed

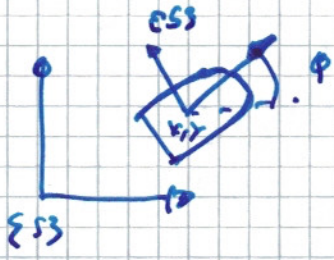
(b) $+\hat{y}_b$ direction: wheels 1 and 3 drive backward, wheels 2 and 4 drive forward } at same speed

(c) rotate in counter-clockwise direction: wheels 1 and 4 drive backward, wheels 2 and 3 drive forward } at same speed

16.11.23

Configuration of chassis of wheeled mobile robot:

$q = (\phi, x, y)$




wheelspeeds $u$:

↳ $u = H(\phi)\,\dot{q}$

Properly constructed omnidirectional robots can achieve any $\dot{q}$ by proper wheel speeds.

↳ motion planning and trajectory generation relatively simple

Control

Apply <u>feedforward plus PI feedback</u> on a planned trajectory $q_d(t)$:

$$\dot{q}(t) = \underbrace{\dot{q}_d(t)}_{\text{desired chassis velocity}} + \underbrace{K_p\,(q_d(t) - q(t))}_{\text{proportional feedback according to configuration error}} + \underbrace{K_i \int_0^t (q_d(t) - q(t))\,dt}_{\text{integral feedback according to configuration error}}$$

↳ commanded chassis velocity

Simpler (but often good enough) is P control:

$$\dot{q}(t) = K_p\,(q_d(t) - q(t))$$

For control we need to estimate current chassis configuration $q$.

- odometry
- sensors: cameras, GPS, laser range finder

↳ calculate wheel speeds $u$ with commanded velocity $\dot{q}(t)$ and kinematic model $H(\phi)$

$$u = H(\phi)\,\dot{q}$$

## Ch. 13.3. Nonholonomic Wheeled Mobile Robots

19.11.23

kPfaffian velocity constraints acting on system with config. $q \in \mathbb{R}^n$

$$A(q)\dot{q} = 0$$

where $A(q) \in \mathbb{R}^{k \times n}$

Instead of specifying k directions in which velocities are not allowed:

write allowable velocities of kinematic system as linear combinations of n-k velocity directions

→ the coefficients of the linear combinations are the controls available for the system

### The Unicicle

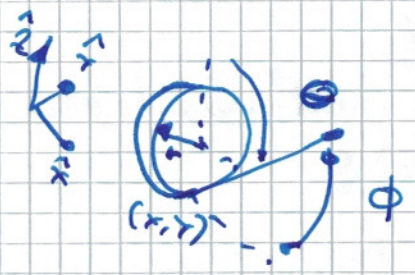
single upright rolling wheel

r : radius

(x,y) : contact point

$\phi$ : heading direction

$\theta$ : rolling angle




config.: $q = (\phi, x, y, \theta)$

config. of chassis: $(\phi, x, y)$

Kinematic equation:

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ r\cos\phi & 0 \\ r\sin\phi & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$= G_1(q)u = g_1(q)u_1 + g_2(q)u_2$$

control inputs: $u = (u_1, u_2)$

with $u_1$ : wheels forward/backward driving speed

$u_2$ : heading direction turning speed

constraints $-u_{1,max} \leq u_1 \leq u_{1,max}$

$-u_{2,max} \leq u_2 \leq u_{2,max}$

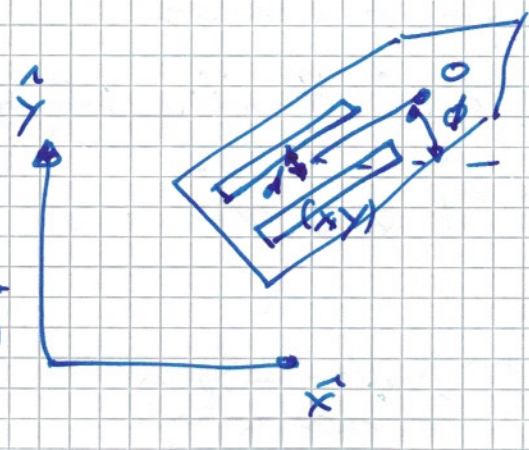
$g_i(q) \in \mathbb{R}^4$ : columns of matrix $G(q)$:
- tangent vector fields / control vector fields / velocity vector fields associated with controls $u_i = 1$ and specified $q$ : $g_i(q)$ : tangent / velocity vector

usually not concerned with rolling angle of wheel $\theta$:

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ r\cos\phi & 0 \\ r\sin\phi & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

## The Differential Drive Robot

$2d$: distance between the 2 drive wheels
$(x, y)$: reference point halfway between wheels
$\theta_L, \theta_R$: rolling angles of wheels
Config: $q = (\phi, x, y, \theta_L, \theta_R)$




Kinematic equations:

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \\ \dot{\theta}_L \\ \dot{\theta}_R \end{bmatrix} = \begin{bmatrix} -r/2d & r/2d \\ \frac{r}{2}\cos\phi & \frac{r}{2}\cos\phi \\ \frac{r}{2}\sin\phi & \frac{r}{2}\sin\phi \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix}$$

$u_L, u_R$ : angular speed of left / right wheel

Usually not concerned with rolling angles of wheels

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -r/2d & r/2d \\ \frac{r}{2}\cos\phi & \frac{r}{2}\cos\phi \\ \frac{r}{2}\sin\phi & \frac{r}{2}\sin\phi \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix}$$

Advantages of diff drive:

- simplicity, high maneuverability

Disadvantages:

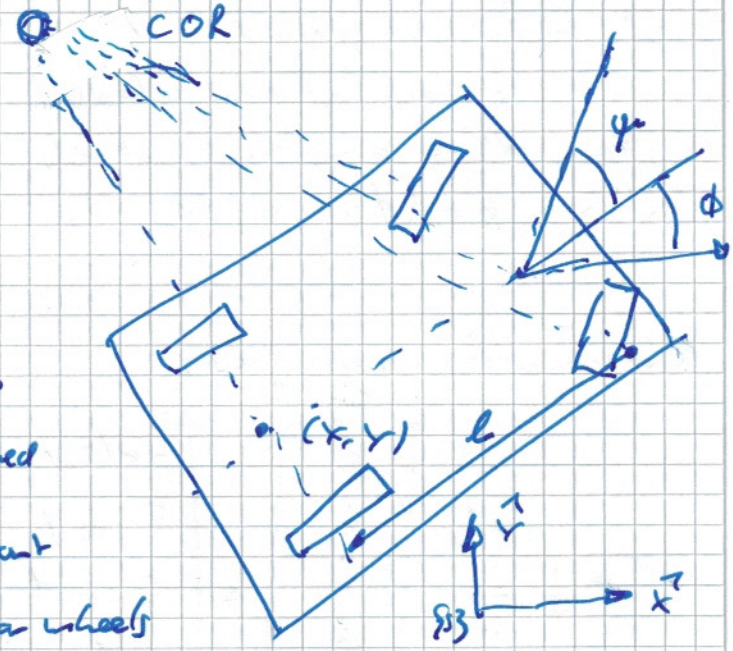
- caster wheels often not appropriate for outdoor use

## Car-Like Robot

Ackermann steering

Config. (ignoring rolling angles of wheels, as often not of interest)

$$q = (\phi, x, y, \psi)$$




where:

$(x, y)$ : location midpoint of rear wheels

$\phi$ : car's heading direction

$\psi$ : steering angle (defined at virtual wheel at midpoint between front wheels

$l$: wheelbase between front/rear wheels

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} (\tan(\psi))/l & 0 \\ \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

controls: - $v$ : forward speed of car (at reference point)

- $\omega$ : angular speed of steering angle

21.11.23

Simplified kinematic for car like robot:

↳ steering controll just angle $\psi$ instead of its rate $\omega$
↳ if steering rate limit $\omega_{max}$ high enough
(that steering angle can be changed nearly instantaneously by low-level controller)
↳ $\psi$ can be eliminated as state variable
→ car's config: $q = (\phi, x, y)$

Use control inputs $(v, \omega)$ where • $v$: car's forward speed
• $\omega$: rate of rotation

↳ can be converted to controls $(v, \psi)$ by:

$$v = v \qquad \psi = \tan^{-1}\left(\frac{l\omega}{v}\right)$$

Simplified kinematics:

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = G(q)u = \begin{bmatrix} 0 & 1 \\ \cos\phi & 0 \\ \sin\phi & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Nonholonomic constraints:

$$\dot{x} = v \cdot \cos\phi$$
$$\dot{y} = v \cdot \sin\phi$$

↳ $A(q)\dot{q} = [0 \ \sin\phi \ -\cos\phi]\,\dot{q}$

$$= \dot{x}\sin\phi - \dot{y}\cos\phi = 0$$

23.11.23

# Canonical Simplified Model for Nonholonomic Mobile Robots

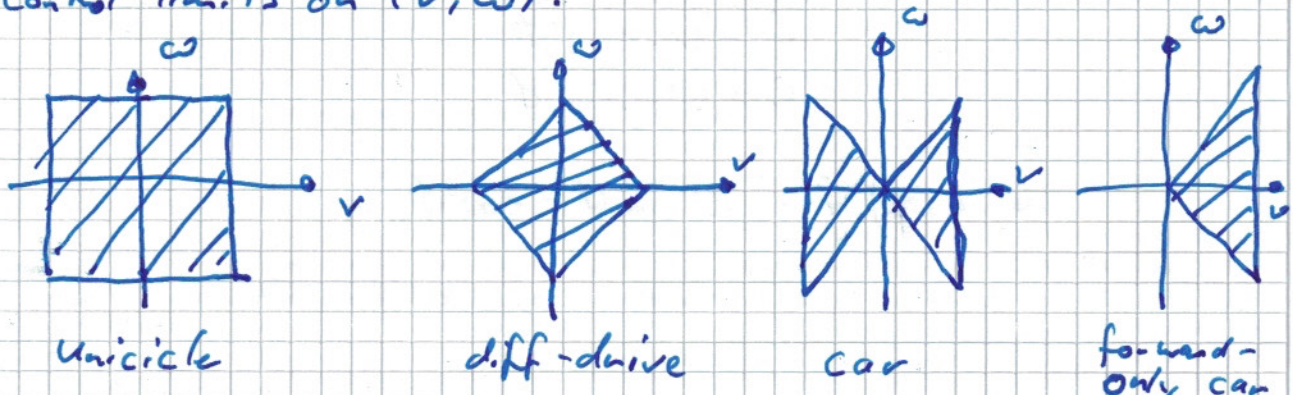
The simplified model for car-like robot can be used for other robots, too.

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = G(q)u = \begin{bmatrix} 0 & 1 \\ \cos\phi & 0 \\ \sin\phi & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

with control transformations:

- Unicicle: $u_1 = \frac{v}{r}$, $u_2 = \omega$
- diff-drive: $u_L = \frac{v - \omega d}{r}$, $u_R = \frac{v + \omega d}{r}$
- car like robot: $v = v$, $\psi = \tan^{-1}\left(\frac{l\omega}{v}\right)$

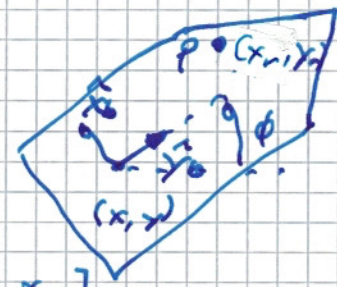
The only difference between these robots is the control limits on $(v, \omega)$:



23.11.23

Control inputs $(v, \omega)$ to directly control linear velocity of a reference point P fixed on the chassis.
→ useful when a sensor is located at P

$P = (x_p, y_p)$ , reference point in world frame
$(x_m, y_m)$ , constant coordinates in chassis frame $\{B\}$




Find controls $(v, \omega)$ needed to achieve desired world-frame motion $(\dot{x}_p, \dot{y}_p)$:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x_m \\ y_m \end{bmatrix}$$

→ differentiating

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} + \dot{\phi} \begin{bmatrix} -\sin\phi & -\cos\phi \\ \cos\phi & -\sin\phi \end{bmatrix} \begin{bmatrix} x_m \\ y_m \end{bmatrix}$$

→ substituting $\omega = \dot{\phi}$ , $(v\cos\phi, v\sin\phi) = (\dot{x}, \dot{y})$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{1}{x_m} \begin{bmatrix} x_m\cos\phi - y_m\sin\phi & x_m\sin\phi + y_m\cos\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix}$$

→ can be read as

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = J^{-1}(q) \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix}$$

where $J(q)$: jacobian relating $(v, \omega)$ to world-frame motion of P

! $J(q)$ is singular if P chosen on the line $x_m = 0$!

Points on line $x_m = 0$ (such as midway point between wheels of a diff-drive robot or between rear wheels of a car) can only move in the heading direction of the vehicle.

24.11.23

## Controllability

Ability to drive a system from one state to another.

Wheeled mobile robot (kinematic):

State: $q = (\phi, x, y)$ (config of chassis)

P-Controller:

$$\dot{q} = K(q_{goal} - q)$$

K acts as spring that pulls robot toward goal.

K: if positive definite
↳ error will decay to zero

If goal is origin ($q_{goal} = 0$)

$$\dot{q} = -Kq$$

this only works because $u = H(\phi)\dot{q}$ has a $\dot{q}$ for any $u$ (u: wheel speeds)
↳ any $\dot{q}$ can be achieved by some wheel speeds $u$

↳ $\dot{q} = -Kq = \underbrace{H^+(\phi)\, u}_{v}$

$$\dot{q} = v$$

$H^+(\phi)$: pseudo-inverse of $H(\phi)$

$v$: nu (greek letter)

More general class of control systems:

$$\dot{x} = Ax + Bv$$

$x \in \mathbb{R}^n$
$v \in \mathbb{R}^m$
$A \in \mathbb{R}^{n \times n}$
$B \in \mathbb{R}^{n \times m}$

Systems of the form $\dot{x} = Ax + Bv$ are linearly controllable if Kalman rank condition is satisfied

$$\text{rank}[B \; AB \; A^2B \ldots A^{n-1}B] = \dim(x) = n$$

↳ this condition ensures that the $m$ controls can act on all $n$ states

If a system is linearly controllable, it's possible to drive it between arbitrary states.

To stabilize origin:

Feedback controller $v = -Kx$

↳ dynamics: $\dot{x} = Ax + B(-Kx) = (A - BK)x$

Stability: chose $K$ such that eigenvalues of $(A - BK)$ all have negative real values

$$\dot{q} = v = \overset{0}{A}q + \overset{I}{B}v$$

25.11.23

## Controllability: nonholonomic Robots

The canonical nonholonomic robot is not a linear control system

$$\dot{q} = G(q)\,u \qquad , G(q) \in \mathbb{R}^{3\times 2}$$

because $G$ depends on configuration $q$

Is there a simple control law to stabilize $q_{desired}$?

Negative example:

Theorem: The system $\dot{q} = G(q)u$, where $\operatorname{rank}(G(0)) < \dim(q)$ cannot be stabilized to the origin by a continuous time-invariant feedback control law.

For canonical nonholonomic robot:

$$\operatorname{rank} G(0) = 2$$
$$\dim(q) = 3$$

$\Rightarrow$ cannot be stabilized to origin using a linear continuous time-invariant feedback control.

Nonholonomic robots are not linearly controllable: use weaker controllability conditions from nonlinear control theory.

$$\dot{q} = G(q)u \qquad q \in \mathbb{R} \qquad u \in U \subset \mathbb{R}^m,\ m < n$$
$$G(q) \in \mathbb{R}^{n\times m}$$

- (global) controllability from $q$
- small-time local controllability from $q$ (STLC)
- small-time local accessability from $q$ (STLA)

„small-time": time $T \to 0$
„local": neighborhood becomes arbitrarily small

28.11.23

## (Global) controllability from $q$:

- any $q_{goal}$ is achievable from $q$
- a control exists that drives robot from $q$ to $q_{goal}$ in finite time

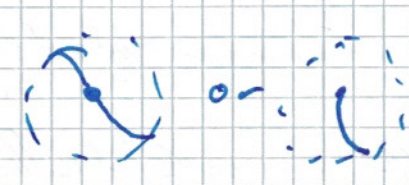
Reachable set:

$q$: in 2d space (as example, real mobile robots: $q \in \mathbb{R}^3$)


to all feasible trajectories from $q$ for time $t \leq T$ must remain inside $W$

Reachable set: $R^W(q, \leq T)$: (or equal)
reachable configs. in time less than $T$ without leaving $W$

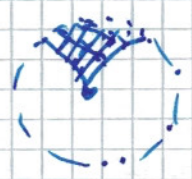
As $W$ and $T$ become arbitrarily small, $R^W(q, \leq T)$ could look like




$\dim(R^W(q, \leq T)) < n$

robot is confined to lower dimensional subset than its config space

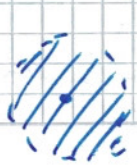
STUCK



$\dim(R^W(q, \leq T)) = n$,
$q \notin \text{in}(R^W(q, \leq T))$

robot can reach a full-dimensional subset of its config space
initial config is on boundary of the reachable set

FREE



$\dim(R^W(q, \leq T)) = n$
$q \in \text{in}(R^W(q, \leq T))$
reachable set is full-dimensional, initial config. is in its interior. $\Rightarrow$ robot can locally move in any direction

1.12.23

## Motion Planning (nonholonomic Robots)

### Obstacle-Free Plane

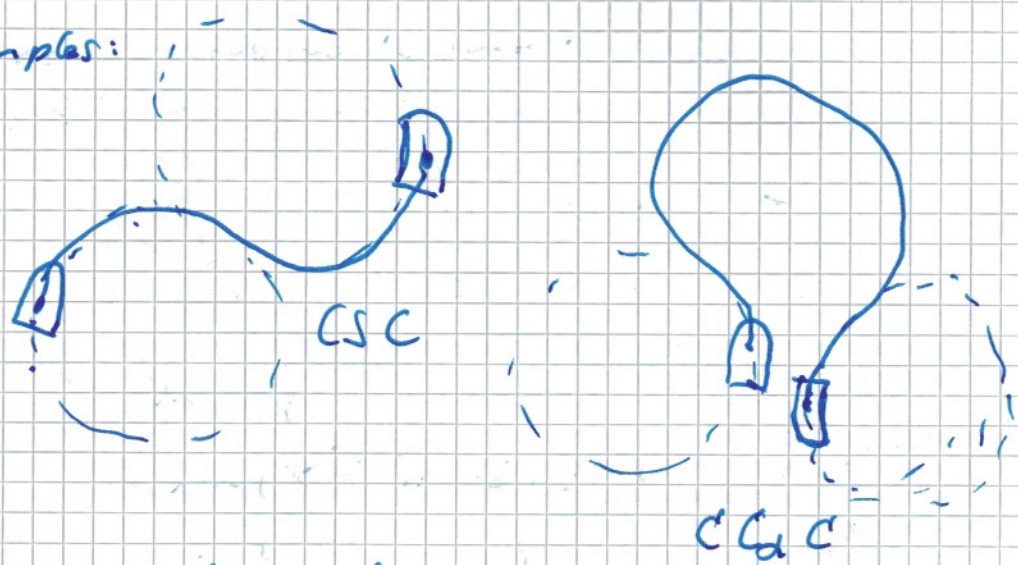
Easy to find feasible motions between two chassis config. $q_0$ and $q_{goal}$.
More interesting to optimize an objective function (shortest path, fastes path...)

### Shortest Paths for Forward-Only Car (Dubins car)

minimize path length from $q_0$ to $q_{goal}$

Theorem: For a forward-only car the shortest paths consist only of arcs of minimum turning radius and straight-line segments.
Circular arc segments: C
straight line segments: S
E.g.
a) sequence CSC
b) sequence $CC_dC$ ($C_d$: circular arc of angle $\geq \pi$)
In any of the C or S segments can be of length zero.

Examples:




Calculate shortest path:
enumerate all possible CSC and $CC_dC$ paths
1. construct 2 minimum-turning radius circles at both $q_0$ and $q_{goal}$

2. Solve for:

a) points where lines (with correct heading direction) (CSC) are tangent to one of the circles at $q_0$ and one of the circle at $q_{goal}$

b) points where a minimum-turning-radius circles (with correct heading direction) is $(CC_\alpha C)$ tangent to one of the circles at $q_0$ and one of circles at $q_{goal}$

↳ shortest path is optimal.
Solution may not be unique

## Shortest Paths for car with reverse gears

2.12.23

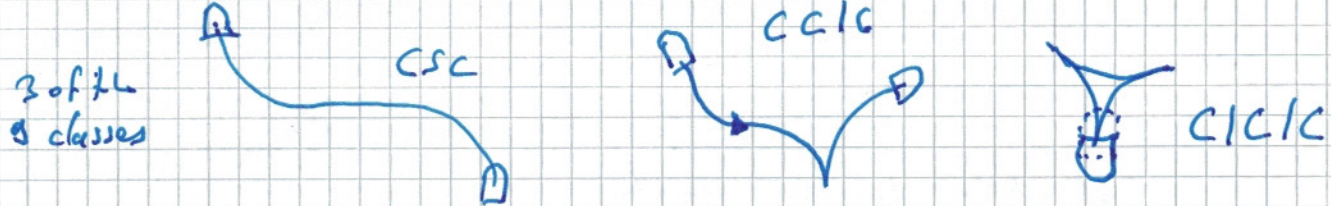
Reads-Shep car
Use only straight-length segments and minimum-turning radius arcs.

Notation:
- C: minimum-turning-radius arc
- $C_\alpha$: arc of angle $\alpha$
- S: straight line
- I: reversal of linear velocity (cusp)

Theorem: For a car with a reverse gear the shortest path between any two configurations is in one of the following nine classes:

| | | |
|---|---|---|
| $C\|C\|C$ | $CC_\alpha\|C$ | $CSC_{\pi/2}\|C$ |
| $CC\|C$ | $C\|C_\alpha C_\alpha C$ | $C\|C_{\pi/2}SC_{\pi/2}\|C$ |
| $C\|CC$ | $C\|C_{\pi/2}SC$ | $CSC$ |

Any of the C or S segments can be zero.




Shortest path: enumerate finite set of possible solution
↳ find shortest
Solution might not be unique.
if it takes zero time to reverse linear velocity: shortest path is also minimum time path.

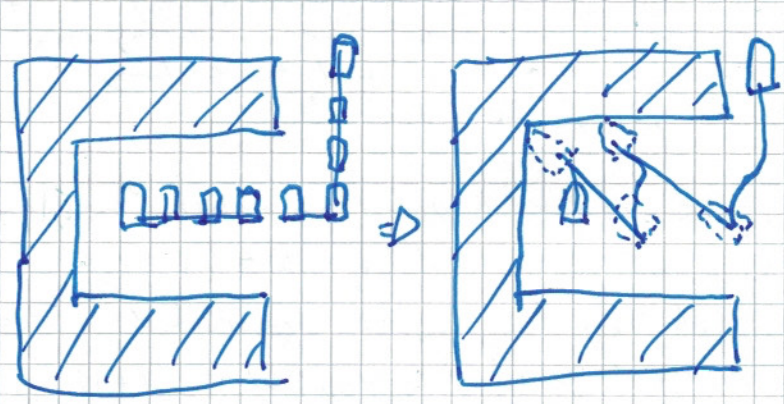
## Minimal-Time Motions for the Diff.-Drive 2.12.23

**Theorem** For a diff.-drive robot, minimum-time motions consist of forward and backward translations (F and B) at max. speed $\pm v_{max}$ and spins in place (R and L) at max. angular speed $\pm \omega_{max}$.
↳ see Book for table with the 40 possible types.

Any reconfiguration of diff.-drive can be achieved by spinning, translating and spinning again, it might not be time-optimal (depending on linear and angular velocities).

## Motion planning with Obstacles

Apply grid-based motion planning method.
Or sampling methods (PRM, RRT)
For any planning method: a reverse gear car can follow the path arbitrarily closely even if it might be slow due to going back and forth multiple times (car parallel parking into a lot)
Alternatively: an initially constraint-free path can be transformed into a fast, feasible path that respects the car constraints (see description in Book)



# Feedback Control

3 types of feedback control problems for the canonical nonholonomic mobile robot with controls $(v, \omega)$

a) stabilization of configuration: Given desired configuration $q_d$ drive error $q_d - q(t)$ to zero as time goes to infinity ($q_d - q(t) = 0$ for $t \to \infty$).
No time-invariant, continuous feedback law in state variables can stabilize config. There exist often (time-varying and discontinuous feedback) laws to accomplish that task

b) trajectory tracking: given desired trajectory $q_d(t)$ drive error $q_d(t) - q(t)$ to zero as time goes to infinity ($q_d(t) - q(t) = 0$ for $t \to \infty$)

c) path tracking: given path $q(s)$, follow geometric path without regard to the time of the motion. More control freedom than trajectory tracking

Trajectory tracking:

Reference trajectory: $q_d(t) = (\phi_d(t), x_d(t), y_d(t))$ for $t \in [0, T]$
Nominal control: $(v_d(t), \omega_d(t)) \in \text{int}(U)$ for $t \in [0, T]$

nominal control in the interior of feasible control set $U$

some control effect is "left over" for correcting small errors

to neither shortest path nor time-optimal trajectory
to both would saturate control

Simple controller: chose reference point on robot (not on axis of the 2 driving wheels)
Desired trajectory $q_d(t)$: represented as desired trajectory of $P$: $(x_{pd}(t), y_{pd}(t))$

Track reference trajectory using P-control: (proportional control law)

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} k_p (x_{pd} - x_p) \\ k_p (y_{pd} - y_p) \end{bmatrix} \quad \text{where } k_p > 0$$

chain-coupling

$\leftarrow$ to convert $(\dot{x}_p, \dot{y}_p)$ to $(v, \omega)$

# Mobile Manipulation

20.12.23

Mobile base with one or more arms.
End-effector motion can be achieved by moving base and arm.
Arm motion is typically more precise than the base.
One approach:

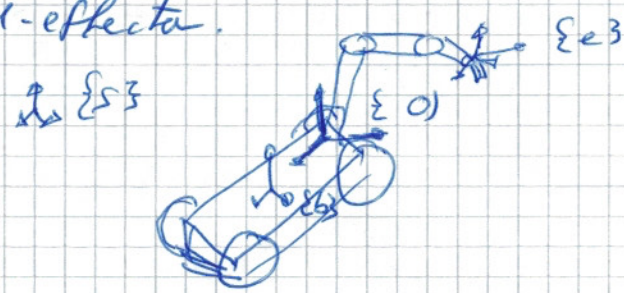
1. drive to a location
2. use arm for manipulation

Some cases contol of wheels and arm joints need to be coordinated:

- if arm has only 5 dof but task requires 6dof
- if more precision or performance is required

End-effector: track trajectory in $SE(3)$

Controller: Jacobian mapping wheel and joint speeds to twist of end-effector.




Frames:
$\{s\}$: space frame
$\{b\}$: reference of mobile base
$\{0\}$: base of robot arm
$\{e\}$: end-effector

$\{b\}$ and $\{0\}$ could be defined as be the same

end-effector config in space frame $\{s\}$:

$$X(q, \theta) = T_{se}(q, \theta)$$

$q$: config of mobile base
$\theta$: arm config.

to calculate transform

$$X(q, \theta) = T_{se}(q, \theta) = T_{sb}(q)\, T_{b0}\, T_{0e}(\theta)$$

- $T_{sb}(q)$: base relative to space frame
- $T_{b0}$: const. offset of base to end-effector base
- $T_{0e}(\theta)$: end-effector frame relative to zero frame

$$X(q,\theta) = T_{se}(q,\theta) = T_{sb}(q)\,T_{b0}\,T_{0e}(\theta)$$

$$T_{sb}(q) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & x \\ \sin\phi & \cos\phi & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ constant height of frame

$T_{0e}$: determined by forward kinematics of arm

$J_{base}(\theta)\,u$: contribution of wheel velocities to end-effector velocity

$J_{arm}(\theta)\,\dot\theta$: contribution of joint velocities to end-effector velocity

$$V_e = J_e(\theta)\begin{bmatrix} u \\ \dot\theta \end{bmatrix} = \underbrace{\begin{bmatrix} J_{base}(\theta) & J_{arm}(\theta) \end{bmatrix}}_{6\times(m+n)} \begin{bmatrix} u \\ \dot\theta \end{bmatrix}$$

$J_b(\theta)$

$V_e$: end-effector twist expressed in end-effector frame

$u$: wheel speeds (m wheel speeds)

$\dot\theta$: joint velocities (n joint velocities)

$J_e(\theta)$: jacobian $6\times(m+n)$ matrix

Recall $J_{base}(\theta)$:

$$V_b = F\dot u, \quad F \in \mathbb{R}^{3\times m}$$

$$F_6 = \begin{bmatrix} 0_{1\times m} \\ F_{3\times m} \\ 0_{2\times m} \end{bmatrix} \in \mathbb{R}^{6\times m}$$

$V_b$: twist of chassis in chassis frame

$F$: $3\times m$ transformation

$V_b$: 6-dim chassis twist in chassis frame

$$\Rightarrow V_{b6} = F_6 u$$

Express twist $V_{b6}$ in end-effector frame

$$[Ad_{T_{es}(\theta)}]\,V_{b6} = [Ad_{T_{0e}^{-1}(\theta)\,T_{b0}^{-1}}]\,\underbrace{V_{b6}}_{F_6 u} =$$

$$\underbrace{[Ad_{T_{0e}^{-1}(\theta)\,T_{b0}^{-1}}]\,F_6}_{J_{base}(\theta)}\,u$$

$\Rightarrow$ full mobile manipulator Jacobian:

$$J_e(\theta) = [\,J_{base}(\theta) \quad \underbrace{J_{arm}(\theta)}_{J_b(\theta)}\,]$$

$J_b(\theta)$: body Jacobian (see chapter 6)

# Mobile Manipulation : Control

21.12.23

Task-space feedforward + PI control

$$V(t) = [Ad_{X^{-1}X_d}]V_d(t) + K_p X_{err}(t) + K_i \int_0^t X_{err}(t)dt$$

$X_{err}$ : twist (expressed in end-effector frame) that takes actual end-effector config to desired config. in unit time

$[X_{err}] = \log(X^{-1}X_d)$

$V(t)$ : commanded twist (commanded by controller) expressed in end-effector frame

To calculate wheel and joint velocities:

$$\begin{bmatrix} u \\ \dot{\theta} \end{bmatrix} = J_e^{\dagger}(\theta)V$$

$J_e^{\dagger}(\theta)$ : pseudo-inverse of $J_e(\theta)$

# Odometry

9.12.23

Estimating chassis config. q from wheel motions.
Essentially integrating wheel velocities.
Available on all mobile robots. Cheap and convenient.
Estimation errors accumulate over time.
to supplement with other position sensors:

- GPS
- visual recognition of landmarks
- ultrasonic range sensing, ultrasonic beacons
- laser range finder
- ...

Odometry generally gives superior results on short time scales. But should be either

- periodically corrected or
- (preferably) integrated with other sensing modalities in an estimation framework (Kalman filter, particle filter, ...)

Assuming:

- each wheel of an omnidirectional robot
- each rear wheel of a diff-drive car

} has an encoder that senses rotation

If wheels are driven by stepper motors
the encoder is not necessary as we know the number of steps we commanded to it (and the angular distance of one step).

Goal: estimate new chassis config $q_{k+1}$ as function of previous config. $q_k$, given change in wheel angles from instant $k$ to instant $k+1$

$\Delta\theta_i$: change in wheel $i$'s driving angle since $\Delta t$ ago

assumption: wheel's angular velocity was constant during time interval $\dot{\theta}_i = \Delta\theta_i / \Delta t$

18.12.13

6-dimensional twist of chassis in chassis frame $\{b\}$:

$$V_{b6} = \begin{bmatrix} \omega_{bx} \\ \omega_{by} \\ \omega_{bz} \\ v_{bx} \\ v_{by} \\ v_{bz} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \omega_{bz} \\ v_{bx} \\ v_{by} \\ 0 \end{bmatrix}}_{\text{chassis motion is planar: 3 components}} = \begin{bmatrix} 0 \\ 0 \\ V_b \\ 0 \end{bmatrix}$$

$V_b$: planar twist

## Odometry process

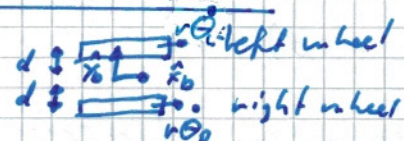
1. Measure wheel displacements $\Delta\theta$ (usually w/ encoders)
2. Assume constant wheel speeds since last reading:
$$\dot\theta = \frac{\Delta\theta}{\Delta t}, \underbrace{\Delta t = 1}_{\text{time units don't matter}} \Rightarrow \dot\theta = \Delta\theta$$
3. Find $V_b = F\dot\theta = F\Delta\theta$
4. Integrate $V_{b6}$ for $\Delta t = 1$ using matrix exponential
$$\underbrace{T_{b_k b_{k+1}}}_{\text{find chassis config at step } k+1 \text{ relative to config at step } k} = e^{[V_{b6}]}$$
5. Express new chassis frame relative to space frame
$$T_{sb_{k+1}} = T_{sb_k} T_{b_k b_{k+1}} \quad \text{(or express as } q_{k+1}\text{)}$$

Most steps are straightforward.

## Details for 3. Find $V_b = F\dot\theta = F\Delta\theta$

- car or diff-drive:

$\{b\}$ midway between the wheels




r: radius of wheels

$$V_b = F\Delta\theta = r \underbrace{\begin{bmatrix} -1/(2d) & 1/(2d) \\ 1/2 & 1/2 \\ 0 & 0 \end{bmatrix}}_{\text{relates wheel increments to chassis twist } V_b} \begin{bmatrix} \Delta\theta_L \\ \Delta\theta_R \end{bmatrix}$$

- omnidirectional robots:

$$\dot\theta = H(0) V_b \Rightarrow V_b = H^+(0)\dot\theta = F\dot\theta = F\Delta\theta$$

$H^+(0)$: pseudoinverse of $H(0)$

Refers to 3. Find $V_b = F\dot{\Theta} = F\Delta\Theta$ (cont.) 18.12.23

- three omniwheel robot:

inverting $H \Rightarrow$

$$V_b = F\Delta\Theta = r\begin{bmatrix} -1/(3d) & -1/(3d) & -1/(3d) \\ 2/3 & -1/3 & -1/3 \\ 0 & -1/(2\sin(\pi/3)) & 1/(2\sin(\pi/3)) \end{bmatrix}\Delta\Theta$$

- four mecanum wheels robot:

$$V_b = F\Delta\Theta = r\begin{bmatrix} -1/(l+w) & 1/(l+w) & 1/(l+w) & -1/(l+w) \\ 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \end{bmatrix}\Delta\Theta$$

with F matrix and wheel increments ($\Delta\Theta$) we get planar chassis twist $V_b$:

↳ $V_{b6} = \begin{bmatrix} 0 \\ 0 \\ V_b \\ 0 \end{bmatrix} \Rightarrow \quad T_{bb_{k+1}} = e^{[V_{b6}]}$

↳ express in space frame

$$T_{sb_{k+1}} = T_{sb_k} T_{b_k b_{k+1}} = T_{sb_k} e^{[V_{b6}]}$$

↳ $q_{k+1}$

equivalently

↳ $\Delta q_b \rightarrow \Delta q \rightarrow q_{k+1} = q_k + \Delta q$