Introduction to Robotics

Felipe P. Vista IV

Grading

- Attendance 5%

Name (Original Name)	User Email	Join Time	Leave Time	Duration (Minutes)
[redacted]		4/12/2021 9:12	4/12/2021 10:14	62
		4/12/2021 9:12	4/12/2021 9:14	3
		4/12/2021 9:12	4/12/2021 9:14	3
		4/12/2021 9:12	4/12/2021 9:14	3
		4/12/2021 9:12	4/12/2021 9:14	3
		4/12/2021 9:12	4/12/2021 9:14	3
		4/12/2021 9:13	4/12/2021 9:13	1
		4/12/2021 9:13	4/12/2021 9:14	2
		4/12/2021 9:14	4/12/2021 9:14	1
		4/12/2021 9:14	4/12/2021 9:14	1
		4/12/2021 9:14	4/12/2021 10:14	60

Bad ZOOM User Name (Absent)

- Iphone → Not your name
- SiAko 202100001 → Wrong order
- SiAko → Name only
- 202100001 → ID Num only

ZOOM User Name (Present)

- University ID Num_Name
- 202100001 SiAko → GOOD (Present)

Name (Original Name)	User Email	Total Duration (Minutes)
[redacted]		62
		63
		62
		62
		63
		62
		63

Student Responsibilities

- Download/Install ZOOM app for online lecture
  - *Zoom profile must be your **OASIS ID+name** similar to OASIS*
  - *Ex.: **202061234 YourName***
  - ***If you are asked, but no reply, then you'll be out of zoom & mark absent***
- Regularly login, check OLD IEILMS for updates, notifications
  - *https://ieilmsold.jbnu.ac.kr*
  - *Presentations & lecture videos will be uploaded after class*
- Regularly check Kakao Group Chat for class
  - *Everybody must have a Kakao talk account*
  - *Search & add account "**botjok**", introduce yourself and name of class ("Robotics"), then you will be added to the group chat*

Intro To Robotics

ROBOTIC MOTION AND ODOMETRY

Robotic Motion & Odometry

- Distance, velocity & time
- Acceleration as change in velocity
- Segments to continuous motion
- Navigation by Odometry
- Linear Odometry
- Odometry with Turns
- Errors in Odometry
- Wheel Encoders
- Inertial Navigation Systems
- DOF and Num of Actuators
- Relative number of Actuators and DOF
- Holonomic/Non-holonomic Motion

Intro

- Robots in real world moving to specific locations
  - *Engineering constraints how fast to move or turn*
- Review distance, time, velocity, acceleration
  - *Physics of motion usually through calculus (continuous functions)*
  - *Computers (discrete approximations)*
- Odometry (fundamental algo for computing robotic motion)
  - *Errors in direction more significant than errors in distance*
- Wheel encoders *(Improve accuracy of odometry)*
- Inertial navigation (*Sophisticated form of odometry*)
- Degrees of Freedom (DOF)
  - *Discuss relation bet DOF & number of actuators (motors) in robotic systems*
- Degrees of Mobility *(*DOM*)*

Distance, Velocity & Time

- **Ex: Robot w/ constant velocity of 10 cm/s for time of 5 s.**
  - *Recall: Distance for constant velocity is* $s = vt$ → s = *50 cms.*

dist velo time

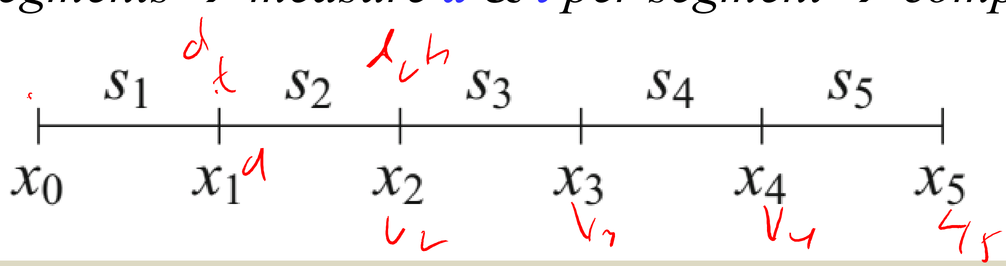
10 cm 5s 50cm

- Robot movement:
  - *power → motors → wheels rotate → robot moves @ certain velocity*
  - *certain power gives certain velocity? Very hard to specify*
    - No two elec/mech components precisely identical (*magnet, wire*, *motor shaft*)
    - Environment affects robot velocity (*friction: too much (mud), too little (ice)*)
    - External forces affect velocity (*power: more (uphill), less (downhill)*)
- Accurate measurements
  - *Short distances (up to several meters): using ruler/tape measure*
  - *Time (hundredths of sec): stopwatch app on phones*
  - *Accurate robot navigation: need to sense objects in environment*

Acceleration as Change in Velocity

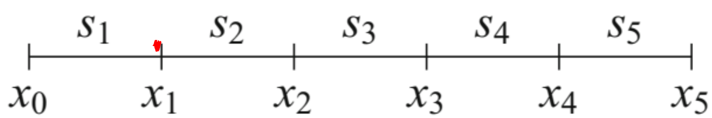
- Exp: Compute moving robot **ave vel** at each marker using ***s = vt***
  - *Longer dist bet markers: Closer to each other*
  - *Shorter dist bet markers: Considerable differences*
- Formula ***s = vt*** assumed constant velocity over entire distance
- Reality, vehicle must change its velocity:
  - *accelerate: from stop to constant velocity*
  - *decelerate: to stop*
- True picture of robot motion
  - *divide into segments → measure **d** & **t** per segment → compute **v***

$v = \frac{s}{t}$



Acceleration as Change in Velocity

- True picture of robot motion
  - *divide into segments → measure **d** & **t** per segment → compute **v*** $= \frac{s}{t}$

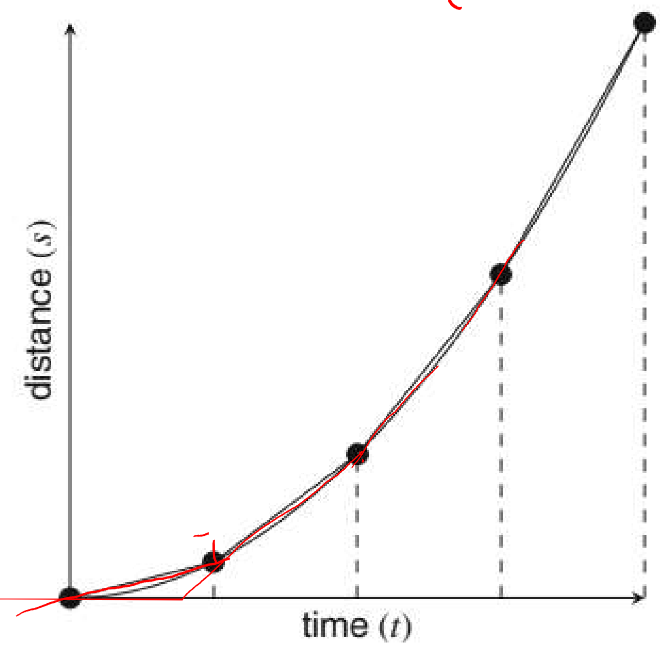


  - Δs_i, *length of* s_i : $\Delta s_i = x_{i+1} - x_i$

    Δt_i, *time to cross* s_i : $\Delta t_i = t_{i+1} - t_i$

    then v_i : $v_i = \Delta s_i / \Delta t_i$

- From accelerating robot graph (a)
  - *Time axis is segmented*
  - *Slopes* $= \Delta s_i / \Delta t_i$
    - which is average velocity in each segment
    - increase over time



(a) Accelerating Robot: Distance increase as square of Time

Acceleration as Change in Velocity

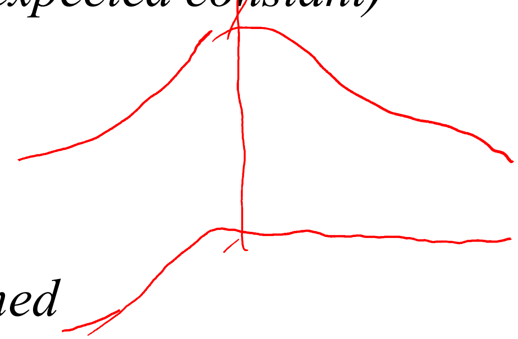
- Definition of "acceleration"
  - *Change in velocity over time:* $a_i = \dfrac{\Delta v_i}{\Delta t_i}$

$\Delta V_i = \dfrac{\Delta S_i}{\Delta t}$

; Velo = change $\dfrac{\text{dist}}{\text{time}}$

acc = chan

- For standard operation of robot
  - *Power (fixed) → force (constant) → acceleration (expected constant) → velocity (increase)*
- But after some time:
  - *acceleration → 0, then velocity → constant*
  - *Since max power possible supplied to wheels reached → to overcome friction & wind resistance*



- What will happen if we increase power setting over time?

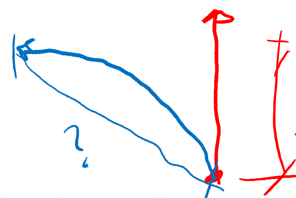
From Segments to Continuous Motion

- Instantaneous velocity as time segments becomes smaller: $v(t) = \dfrac{ds(t)}{dt}.$ $\{v\ d(t) \neq s(t)\}$
- Simultaneously, instantaneous acceleration of robot is: $a(t) = \dfrac{dv(t)}{dt}.$
- While velocity for constant acceleration is: $v(t) = \int a\,dt \Rightarrow a\int dt \Rightarrow at.$
- Then distance is: $s(t) = \int v(t)\,dt \Rightarrow \int a\,t\,dt \Rightarrow \dfrac{at^2}{2}.$

Ex.: Car accelerates from 0 ~ 100 km/h in about 10 s.

- Convert units km/s → m/s: $v_{\max} = 100\,\text{km/h} = \dfrac{(100)(1000)}{(60)(60)}\,\text{m/s} = 27.8\,\text{m/s}.$
- Assuming const acc: $v_{\max} = 27.8 = at = a(10),\ \therefore\ a = 2.78\,\text{m/s}^2.$
- Therefore, distance ***s*** that car moves in 10 s is: $s(10) = \dfrac{at^2}{2} = \dfrac{(2.78)(10)^2}{2} = 139\text{m}.$

Navigation by Odometry



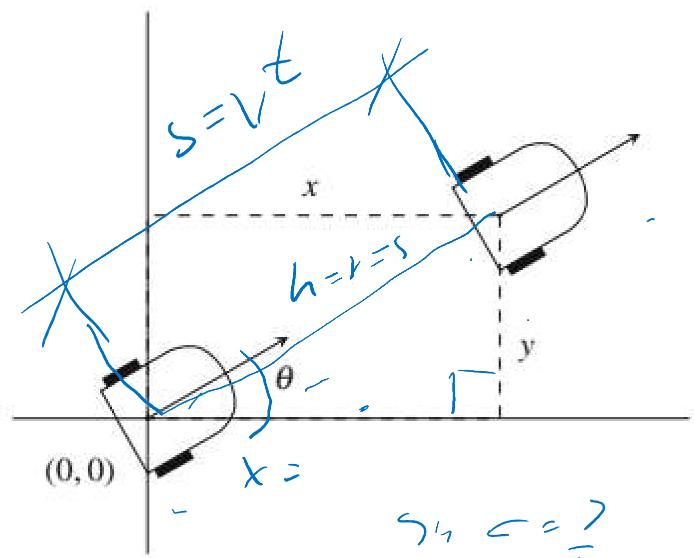
$s = vt$

- Odometry
  - *The measurement of distance*
    - In a car: measures time and speed to calculate distance
  - *Fundamental method used by robots for navigation*
    - From ***s* = *vt***, new position can be computed
    - Computation: easy in 1-D, more complex when making a turn
  - *Form of localization (determine its position in the environment)*
- Disadvantage of Odometry
  - *Indirect measurements (relate motor power/wheel motion to change in pos)*
  - *Can be error prone*
    - Motor speed & wheel rotation relationship can be very non-linear & time varying
    - Wheels slip/skid, affecting the relationship of wheel motion to robot motion
    - Can be improved using **INS** (acceleration & angular velocity)

Linear Odometry

- With $s = vt$ relationship
  - *Robot start at pos (0,0) → moves along x-axis for t secs → new pos (vt,0)*
- Self-driving car can use odometry to determine its position
  - *Not depend only on odometry, but use with sensory data (Ex: when to turn)*
- Moving in 2D (must solve three things)
  - ***position (x, y)***: *relative to a fixed origin*
  - ***heading (θ)***: direction w/c robot is pointing
  - ***pose (x, y, θ)*** : position & heading
- **Ex.: Robot moves**
  - *start at (0, 0) → straight θ; vel v for time t*
  - *new pos(x_2 , y_2)?*

$$x_2 = vt\cos\theta, \quad y_2 = vt\sin\theta.$$



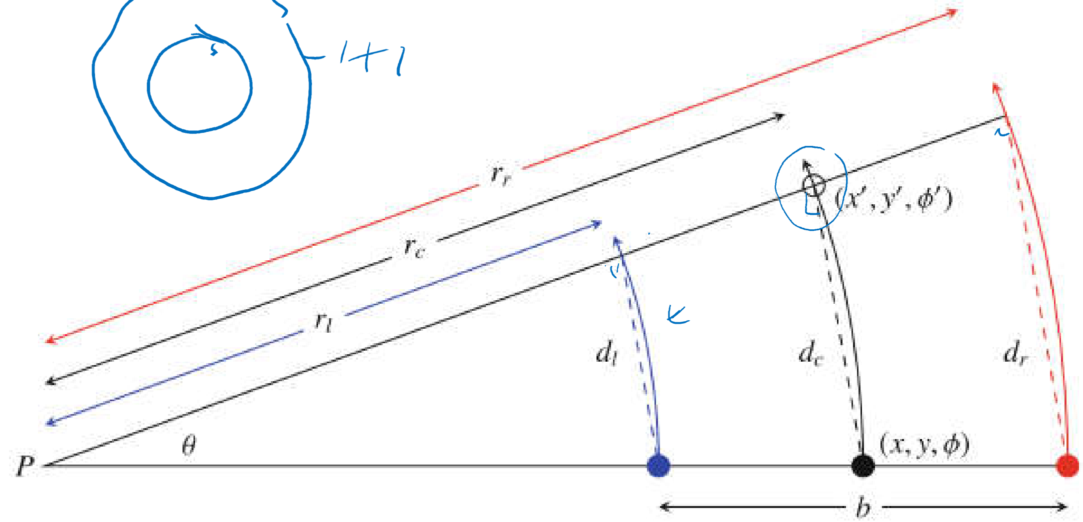
Position and Heading

Odometry with Turns

- **Ex.: Robot turn slightly-left (RightWheel > LeftWheel)**

  - baseline ***b*** : distance bet wheels
  - d_l, d_r, d_c : distances covered by two wheels & center

1+1



Geometry of 2-wheeled robot making left-turn

Get robot new pos & heading?

- We can measure d_l & d_r relating motor power to rotational speed, then multiply w/ time elapsed.

- Or, number of rotations given by wheel encoders:
  - With radius (***R***), rot speeds (ω_l, ω_r) at rev/sec, then after ***t*** secs, the wheel has moved:

$$d_i = 2\pi R\omega_i t, \;\; i = l, r. \qquad \textbf{(5.1)}$$

  - But the task is find new post of robot after wheels has moved these distances!

Odometry with Turns

- **Ex.: Robot turn slightly-left (RightWheel > LeftWheel)**

  - Initial pose ($\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{\phi}$), facing north ($\phi = \pi/2$).
  - What is new pose ($\boldsymbol{x'}$, $\boldsymbol{y'}$, $\boldsymbol{\phi'}$) after turning θ radians?
    - New heading is now definitely:

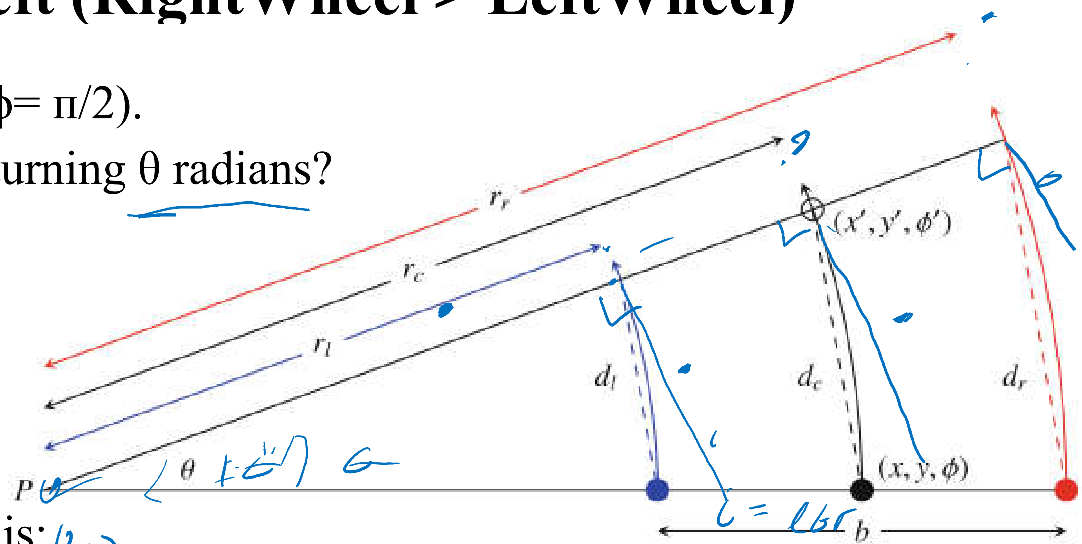
      $$\phi' = \phi + \theta.$$

    - Now, we need to solve for:

      $$x', y'.$$

  - Length of an arc of angle $\boldsymbol{\theta}$ radians is:

    $$2\pi r(\theta/2\pi) \rightarrow \theta r.$$



Geometry of 2-wheeled robot making left-turn

  - For small angles, d_l, d_r, d_c *approx equal to length of its arcs, so:*

    $$\theta = d_l/r_l = d_r/r_r = d_c/r_c. \qquad \textbf{(5.2)}$$

    where $\boldsymbol{r_l}$, $\boldsymbol{r_r}$, $\boldsymbol{r_c}$ *are the distances from origin of the turn (**P**).*

Odometry with Turns

- **Ex.: Robot turn slightly-left (RightWheel > LeftWheel)**

- Distances d_l & d_r , obtained from wheel rotation (**5.1**).

- Angle θ can be computed from (**5.2**) by:

  - Recall:

$$\theta = d_l / r_l = d_r / r_r = d_c / r_c .$$

  - Therefore:

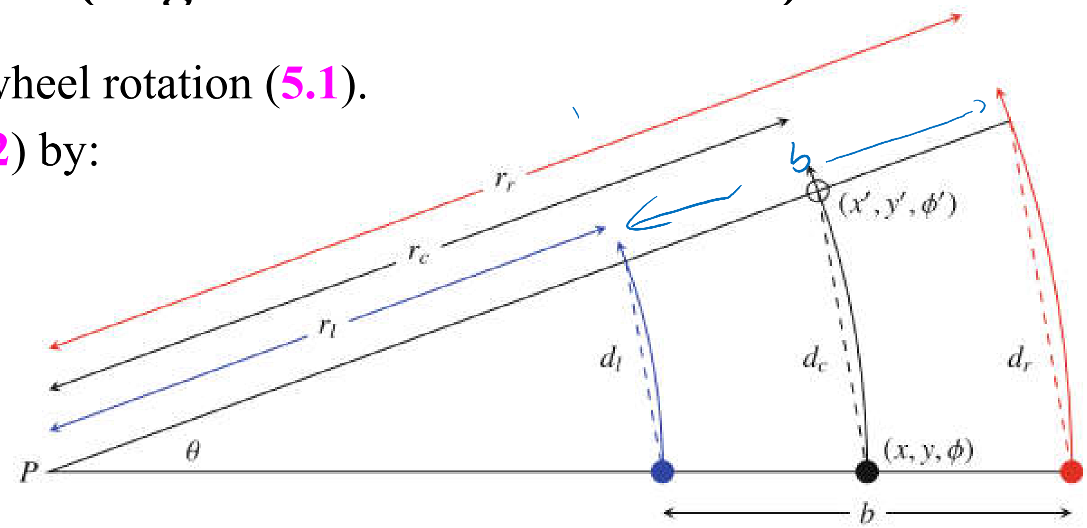
$$\theta r_r = d_r ,$$

$$\theta r_l = d_l ,$$

$$\theta r_r - \theta r_l = d_r - d_l$$

$$\theta = (d_r - d_l)/(r_r - r_l)$$

$$\theta = (d_r - d_l)/b .$$



Geometry of 2-wheeled robot making left-turn

where baseline b: fixed physical measurement of robot.

Odometry with Turns

- **Ex.: Robot turn slightly-left (RightWheel > LeftWheel)**

- Center is halfway bet wheels $r_c = (r_l + r_r)/2$, again by (5.2).

- Recall:

$$\theta = d_l / r_l = d_r / r_r = d_c / r_c .$$

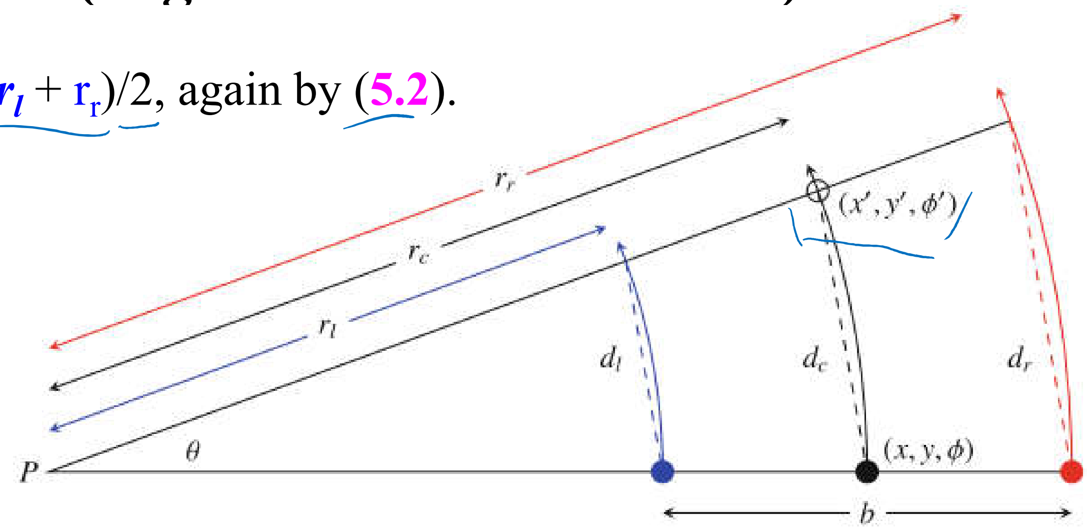
- Therefore:

$$d_c = \theta r_c$$

$$= \theta\left(\frac{r_l + r_r}{2}\right)$$

$$= \frac{\theta}{2}\left(\frac{d_l}{\theta} + \frac{d_r}{\theta}\right) = \frac{\cancel{\theta}}{2\cancel{\theta}}(d_l + d_2)$$

$$d_c = \frac{d_l + d_r}{2}.$$



Geometry of 2-wheeled robot making left-turn

Odometry with Turns

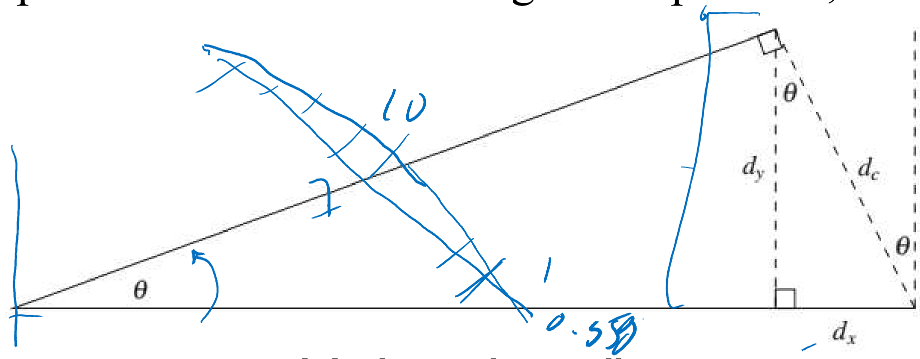
- **Ex.: Robot turn slightly-left (RightWheel > LeftWheel)**

  - If distance travelled is small, $\boldsymbol{d_c}$ approx perpendicular to radius through final position,
    - $\boldsymbol{\theta}$ is the change in heading of robot.
  - Using trigonometry, via similar triangles

$$d_x = -d_c \sin\theta, \;\; d_y = d_c \cos\theta.$$

  - Therefore, robot pose after turning is:

$$(x', y', \theta') = (-d_c \sin\theta, d_c \cos\theta, \phi + \theta).$$

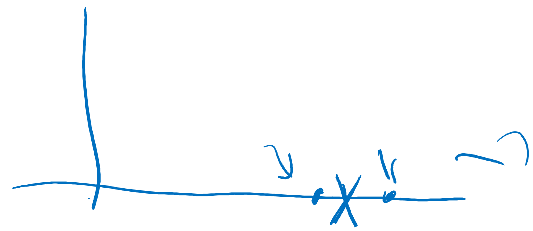


(a) Change in Heading

- We just solved for ***dx, dy, dθ*** when robot moves short distance
- **Frequent** computation if using odometry over long distance
- Why interval between computations must be short?
  1) *Constant speed assumption holds only for short distances*
  2) *Trigo computation simplified w/ assumption of short distance*

Errors in Odometry

- Recall: Odometry not accurate
  - *Inconsistent measurements*
  - *Irregularities in surface cause errors*



- We will see that small changes in robot movement direction
  - *Cause errors greater than changes in its linear motion*
- **Ex: Robot move along x-axis, check for specific object around**
  - *Effect of error **up to p%**?*
  - *If error is in measurement of **x**, then distance moved (Δx),error in **x**:*

$$\Delta x \leq \pm(10)\left(\frac{p}{100}\right)\mathrm{m} = \pm\left(\frac{p}{10}\right)\mathrm{m}$$

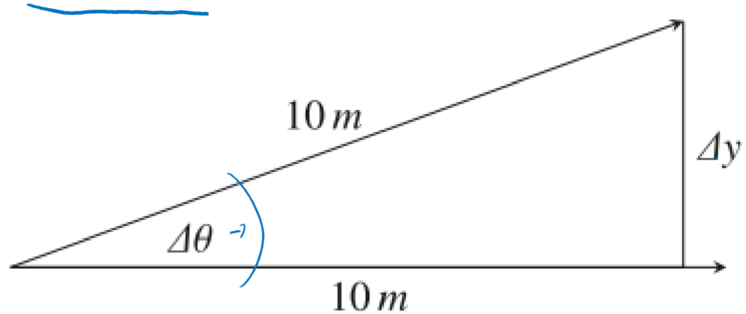
Linear

*where (+/-) since robot can move up to **p%** before or after target distance*

Errors in Odometry

- Now, ***p%*** error in ***heading*** & assume no error in distance moved

δx = x err



- *Robot to move* ***10 m*** *along x-axis, move slightly to left at angle* ***Δθ****.*
- *Compute left-right deviation hand* ***Δy***
- *By trigonometry, Δ y = 10 sin Δθ, then* ***p%*** *error in heading is:*

$$\Delta\theta = (360)\left(\frac{p}{100}\right)(3.6p)^{\circ}$$

- *Then left-right derivation*

$$\Delta y \leq \pm 10\sin(3.6p)$$

Errors in Odometry

- Differences bet linear error of ***p%*** & error in heading of ***p%***

$p\%$	Δx (m)
1	0.1
2	0.2
5	0.5
10	1.00

Linear error in p%

$p\%$	$\Delta\theta$ (°)	$\sin \Delta\theta$	Δy (m)
1	3.6	0.063	0.63
2	7.2	0.125	1.25
5	18.0	0.309	3.09
10	36.0	0.588	5.88

Error in heading of p%

- Small error of 2%:
  - *Linear: 0.2m (object still around vicinity)*
  - *Heading: 1.25m (little bit farther but probably still ok)*
- More significant error of 5% or 10%:
  - *Linear: 0.5m/1.0 m (probably still manageable)*
  - *Heading: 3.09m/5.88 m (too far, might miss the object)*

Errors in Odometry

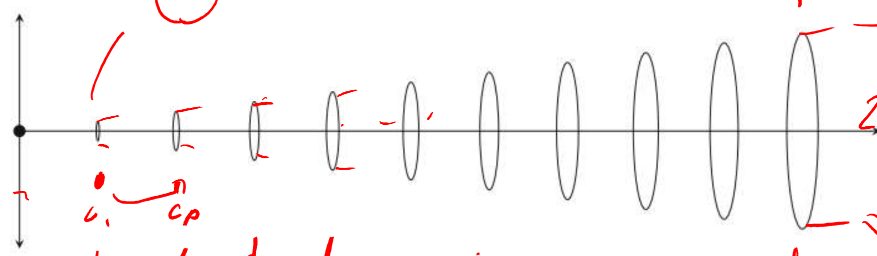
- Assuming
  - *Max ±4% (linear/heading)*
  - *Possible positions at* ***d***
    - 1, 2, …, 10 displayed as ellipses
  - *Linear errors (minor radii)*

    $$0.04s = 0.04, 0.08, ..., 0.4\text{m}$$

  - *Angular errors (major radii)*

    $$d\sin\left(0.04 * 360°\right) = d\sin 14.4° \approx 0.25m, 0.50m, ..., 2.5m$$

  - *Angular more significant than linear*



(a) Odometry errors over distance

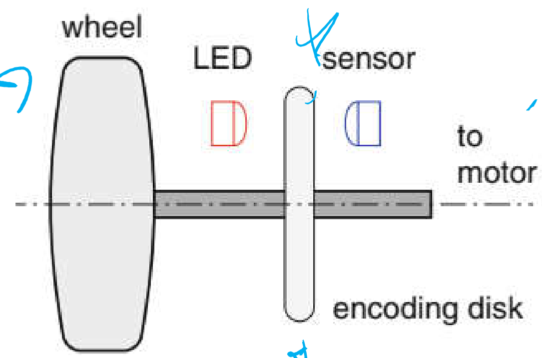
- Since error unavoidable
  - *Computed pose via odometry periodically compared w/ absolute position*
    - Determining absolute position discussed in later chapters
  - *Computed pose becomes new initial position*

Wheel Encoders

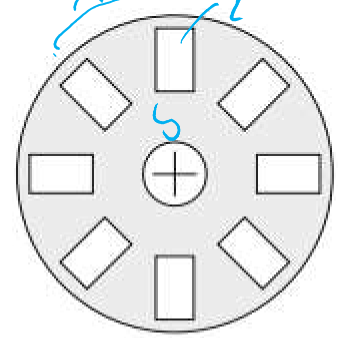
- Improve odometry in wheeled vehicle by:
  - *Measuring rotation of wheels instead of mapping motor power to velocity*
- Procedure
  - *Wheel circumference:* $2\pi r$ = d
  - *Radius of wheel in cm:* r
  - *Num of rotations counted:* n
  - *Therefore, robot movement :* $2\pi rn$
  - *If signal is generated 8 times per revolution, then:* $2\pi rn/8$
    - where ***n*** is now number of signals counted by the computer

Wheel Encoders

- Some implementations of wheel encoder



(a) Optical wheel encoder



(b) Encoding disk

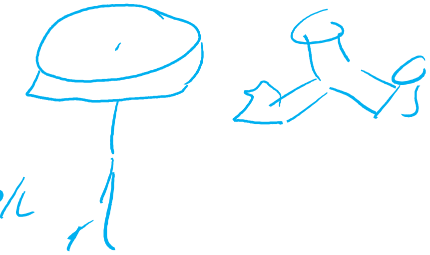
(a) Optical wheel encoder

– *Light source (ex. LED), light sensor, encoding disk attached to wheel axis*

(b) Disk perforated with holes

– *Sensor generates signal whenever light passes through hole*

Inertial Navigation Systems



- Directly measure linear acc & angular vel
  - *To calculate pose of a vehicle* (pos, θ)
  - *IMU (Inertial measurement Unit) also used, But INS refers to whole system*
- Integrating acc gives curr vel: $\int_0^{\tau} a(t)\,dt.$
- Integrating angular vel gives change in heading: $\int_0^{\tau} \omega(t)\,dt.$
- No continuous functions to integrate
  - *Acc & angular vel are sampled, summation replaces integration*

$$v_n = \sum_{i=0}^{n} a_n \Delta t, \quad \theta_n = \sum_{i=0}^{n} \omega_n \Delta t.$$

- Errors
  - *Measurement inaccuracies, environmental variations (temp), wear & tear*
  - *Often combined w/ GPS to update position w/ absolute location*

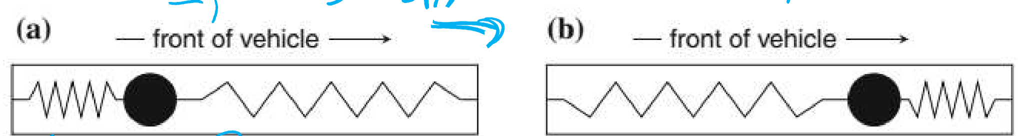
Pure → INS + GPS

{GPS + INS + DC = φ}

INS: Accelerometers

- We can feel it
  - *Plane: The force pushing us to our seat when plane is taking off*
    Force pushing away from seat when landing
  - *Car: the same when accelerate rapidly or make sudden stop*
- Acceleration related to force via: $F = ma$
  - *where F = force, and m = mass*
  - *Measuring force on an object, we can measure acceleration*

dir



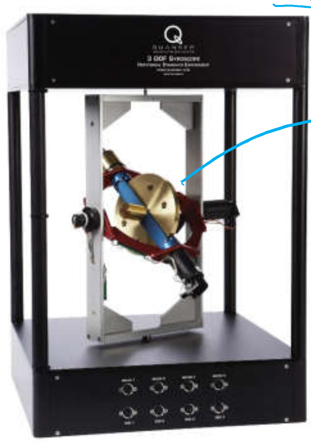
spring

Simple accelerometer

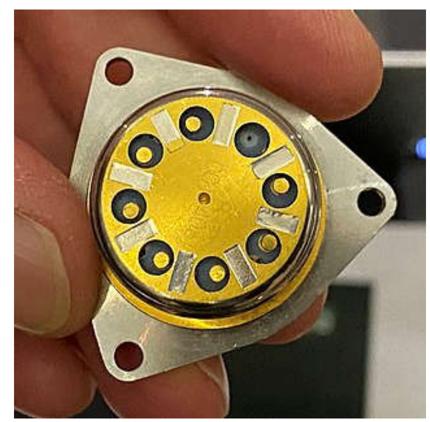
- Simple Accelerometer
  - *>> acc → >> force exerted on mass → spring compresses to front*
  - *<< acc → << force exerted on mass ← spring compresses to back*

INS: Gyroscopes

- Principle of Coriolis force to measure angular vel
  - *We'll not discuss it, check out Physics textbooks if you're curious ☺*
- Types of gyros
  - *Classical: spinning mechanical disks mounted on gimbals, extremely accurate, very heavy, consume lots of power (**aircrafts, rockets**)*
  - *Ring laser (RLG): almost no moving parts, two laser beams, better than mechanical*
  - *Coriolis vibratory gyros (CVG): MEMS (microelectromechanical sys), cheap & robust, accuracy not as good as previous ones (**smartphones, robots**)*

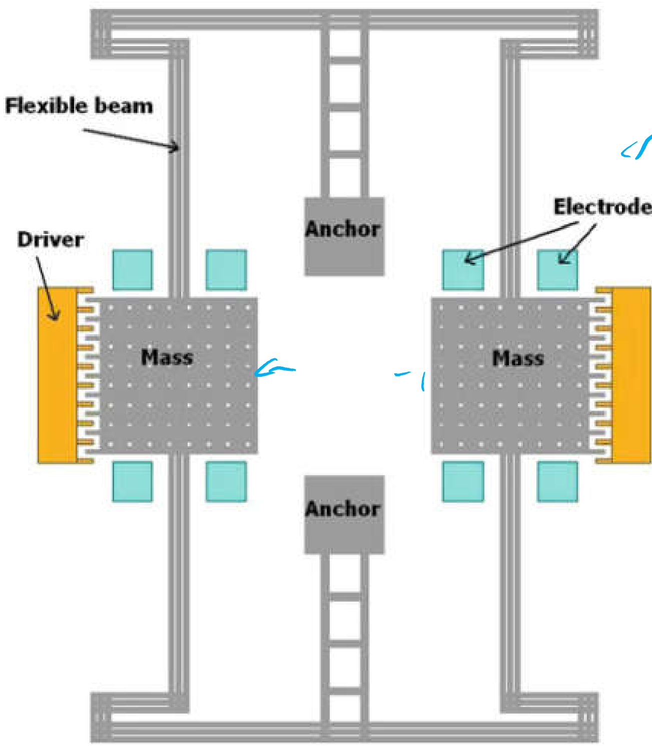


disks

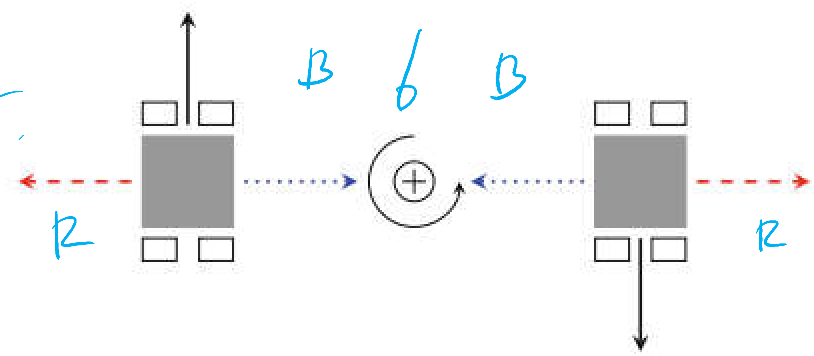


INS: Gyroscopes

- Example of a CVG



Tuning fork gyroscope

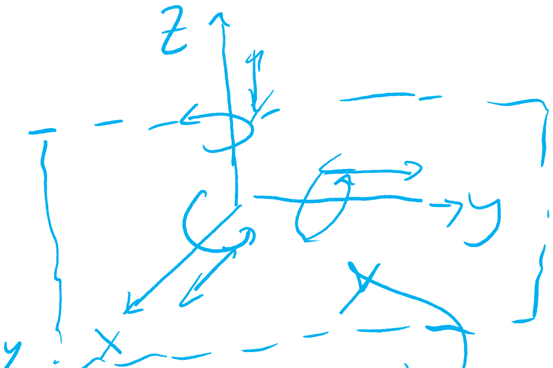


Physics of a tuning fork gyroscope

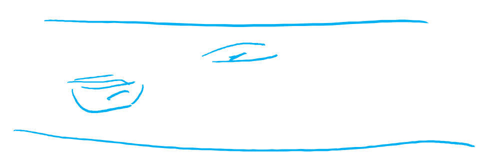
* ***Red-/Blue-dash***: direction of vibration

* ***Solid black*** : direction of Coriolis Force

INS: Applications



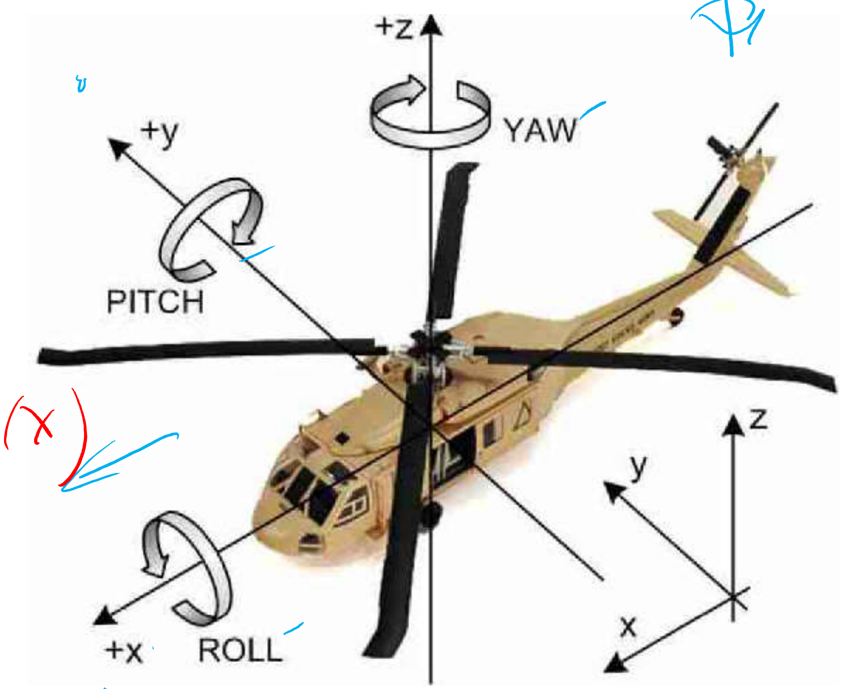
- An INS   acc/axis   ang velo/s
  - *Three accelerometers and three gyroscopes*
  - *So that pose can be computed in 3-D* acc_x, acc_y ; w_x, w_y
  - *Necessary for robotic aircraft and other robotic vehicles*   (ships) UAV
- Ex:
  - ***Accelerometer:*** ** detects rapid deceleration in front-back dir during a crash*
    ** up-down direction can detect if car has fallen into a pothole*
  - ***Gyroscope:*** ** rotation around vertical axis can detect skidding*
    ** rotation around front-rear axis can detect if car is rolling over*



Degrees of Freedom & Numbers of Actuators

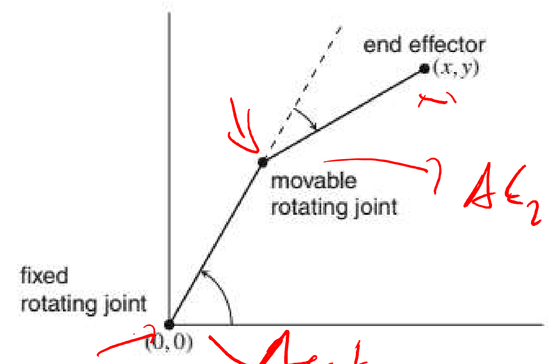
- Degrees of Freedom (DOF)
  - *Dimensionality of the coordinates to describe pose of mobile robot*
  - *Or pose of the end-effector of robotic manipulator*
- **Ex: Helicopter has 6 DOF**
  - Can move in three spatial dimensions, rotate around three axes

Helicopter's 3-axis
(a) **pitch**: nose moves up & down (y)
(b) **roll**: body rotates around its lengthwise axis (x)
(c) **yaw**: body rotates left & right around axis of its rotor (Z)

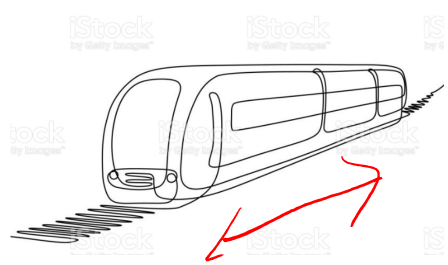


Degrees of Freedom & Numbers of Actuators

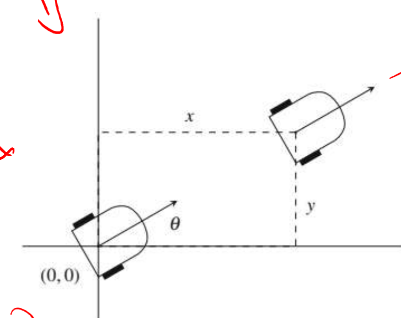
- The robotic arm in (a) only has 2-DOF
  - *End-effector moves in a plane & don't rotate*
  - *Described by a 2-D coordinate (x, y)*
- Mobile robot on flat-surface (b) has 3-DOF
  - *Pose defined by 3-D coordinate (x, y, θ)*
- Train on a track (c) has 1-DOF
  - *Can only move forward, sometimes backward*
  - *Only one coordinate, (x)*



(a) Two-link arm



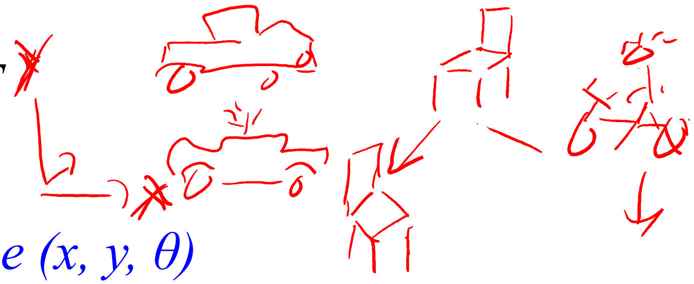
(c) Train on a Track



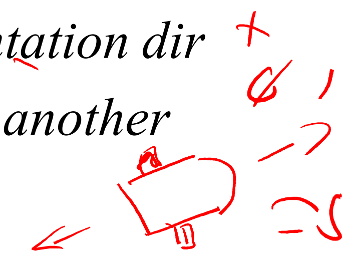
(b) Mobile robot

Degrees of Freedom & Numbers of Actuators

- Robotic motion
  - *Needs more information than just DOF*



- **Ex: A car, bicycle, chair**
  - *Three coordinates to describe their pose (x, y, θ)*
  - ***Chair**: Can be directly moved to any point in plane & orientation dir*
  - ***Car** & **Bicycle**: Cannot be directly moved from one pose to another*
    ** $(2, 0, 0^o)$ pointing along (+) x-axis → $(2, 2, 0^o)$*
    ** not possible, more complex maneuver is needed.*



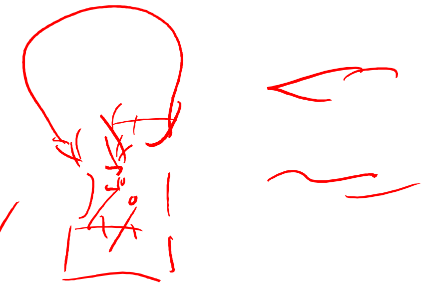
- Need to know num of actuators (motors) & its configuration
  - ***Diff drive robot** has 2 actuators, but robot has 3-DOF* (x, y, ϕ)
    - Motors move along one axis fwd & bck, can also change heading (remember??)
  - ***Two-link arm** has 2 actuators & 2-DOF*
  - ***Train** has 1 actuator & 1-DOF*

Relative Number of Actuators & DOF

- Let's analyze systems where
  - *# of actuators = # of DOF*
  - *# of actuators < # of DOF*
  - *# of actuators > # of DOF*
- # of actuators = # of DOF
  - *Advantage: * system relatively easy to control,*
    ** each actuator individually commanded to control its respective DOF*
  - ***Train** (1-Actuator, 1-DOF), two-line robotic arm (2-Actuators, 2-DOF)*
- # of actuators < # of DOF
  - *Advantage: Fewer actuators → less expensive*
    Disadvantage: Planning & motion control more difficult
  - ***Diff drive robot** & **car** only have 2 actuators, but can access all 3-D poses*

Relative Number of Actuators & DOF

- # of actuators < # of DOF
  - *Advantage: Fewer actuators → less expensive*
    Disadvantage: Planning & motion control more difficult
  - *Mobile robots usually belong to this group*
  - ***Extreme ex* is *hot-air balloon, only 1-actuator (up← heater→down)***



* *wind make balloon move in any of three spatial dirs, even partially rotate*

* *operator can never precisely control it*

** *different from elevator, both have 1-actuator, but it only has 1-DOF*

  - ***Another Extreme Ex*: *Helicopter controllers, 3-actuators, 6-DOF***

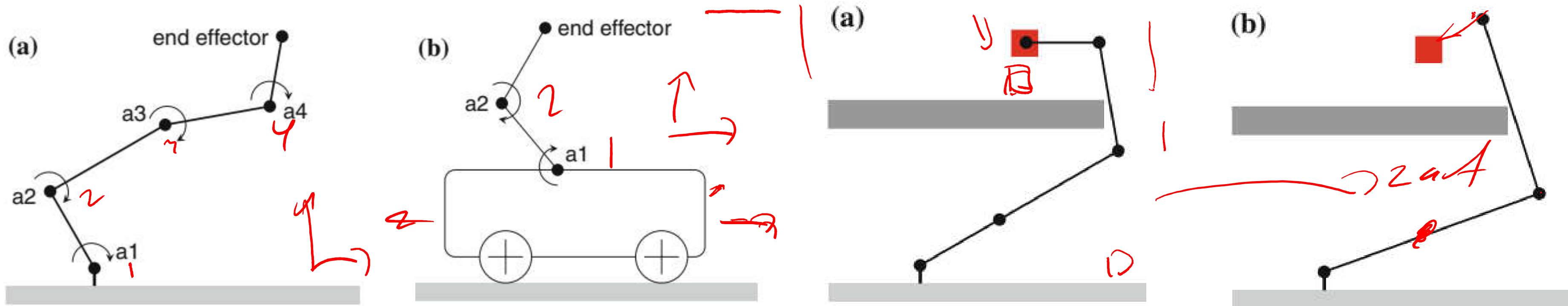
* ***cyclic* (*pitch*)** *: main rotor shaft (fwd, back, sideways)*

* ***collective* (*pitch*)** *: main rotor blades (up or down)*

* ***pedals* (*speed*)***: tail motor (where nose points)*

Relative Number of Actuators & DOF

- # of actuators > # of DOF
  - *Do not sound so good but is it useful in some configurations*



(a) Robot arm: 4-Actuators, 2-DOF

(b) Mobile robot & arm: 3-Actuators, 2-DOF

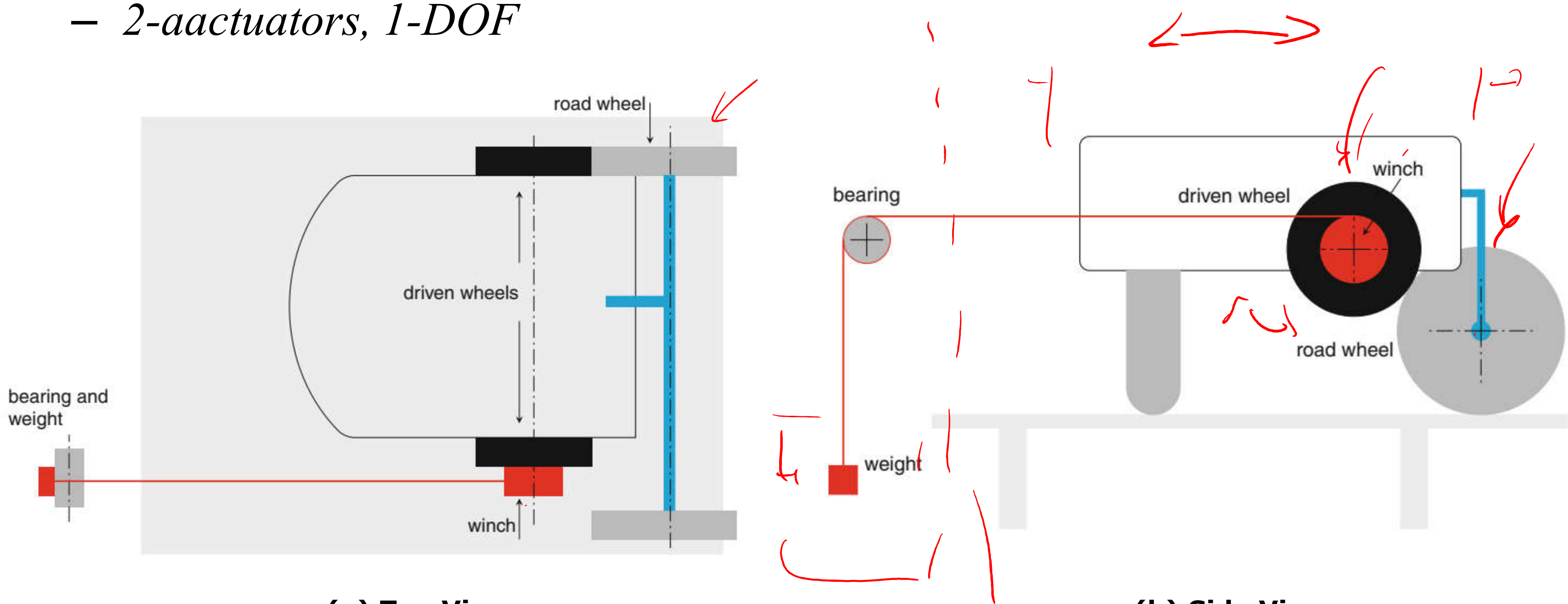
(c) Arm w/ 4-Actuators can reach hidden position

(d) Arm w/ 2-Actuators blocked by obstacle

  - *Engineers avoid more actuators than DOF*
    - It increases complexity and costs
  - *Redundant systems required since can't be performed w/ fewer actuators*
  - *Systems with different characteristics (b) very useful*

Relative Number of Actuators & DOF

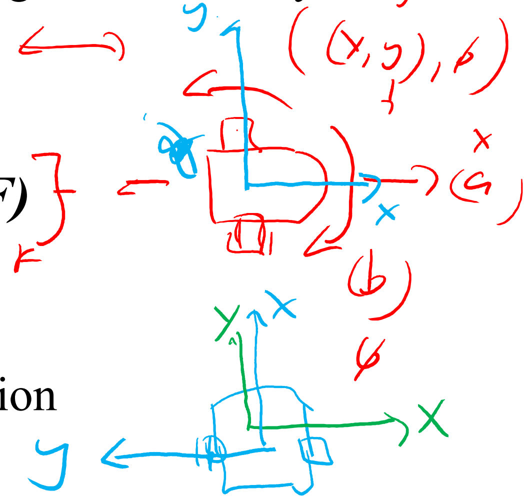
- # of actuators > # of DOF
  - *Robotic crane from mobile robot and a winch*
  - *2-aactuators, 1-DOF*



(a) Top View **(b) Side View**

Holonomic & Non-holonomic Motion

- Degree of Mobility (DOM, δ_m) DOF, # Act
  - *Links DOF and actuators in the case of a robot*
  - *Number of DOF that can be directly accessed by actuators*
  - ***Ex: Mobile robot on a plane***
    - At most 3-DOF (position, heading), It's maximal degree of mobility = 3
  - ***Ex.: Train (1-Actuator, 1-DOF), δ_m = 1***
    - Engine directly affects single DOF
  - ***Ex.: Differential drive robot (2-Actuators, 3-DOF)*** 1 DOF
    - (a) both wheels run same speed (fwd, back)
      (b) wheels opposite direction (rotate in place)
    - Can't directly access DOF of lateral axis of translation
    - Therefore, $\delta_m = 2 <$ DOF = 3

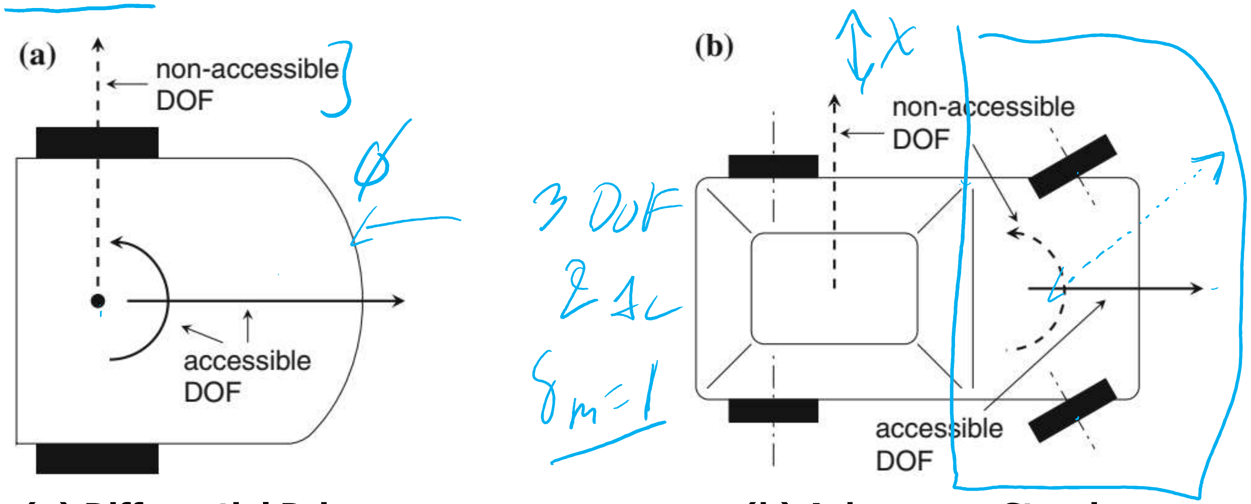


Holonomic & Non-holonomic Motion

- Degree of Mobility (DOM, δ_m)
  - *Ex.: Car, similar to Differential Drive (2-Actuators, 3-DOF)*
    - (a) motor gives direct access to DOF along longitudinal axis of car (fwd, back)
    - (b) steering wheel don't give direct access to any add'l DOF, only orients the first DOF
    - Car can't rotate vertical axis & can't move laterally, Therefore, $\boldsymbol{\delta_m} = 1$ only

3 DOF
2 Act
$\delta_m = 1$

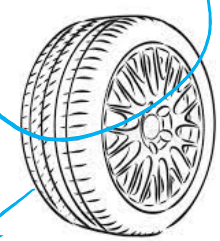
3 DOF
2 Ac
$\delta_m = 1$



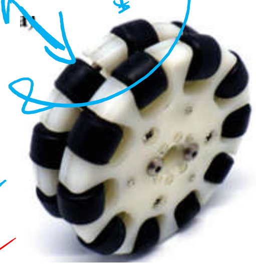
(a) Differential Drive **(b) Ackermann Steering**

Holonomic & Non-holonomic Motion

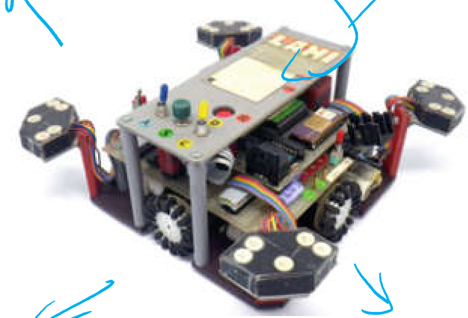
- Degree of Mobility (DOM, δ_m)
  - **Ex.:** ***(a) The Tire***
  - *Standard wheel has* $\boldsymbol{\delta_m} = 2$
  - *roll fwd & bck, rotate vertical axis*
  - *In a car,* $\boldsymbol{\delta_m} = 1$*, since there are two pairs of ties*
    - Gives stability to the car
  - **Ex.:** ***(b) Swedish wheels (3-Actuators, 3-DOF)***
    - Standard wheel with small free wheels along its rim
  - **Ex.:** ***(c) Omnidirectional Vehicle (3-Actuators, 3-DOF)***
    - Mobile robots that can directly access all DOF's



(a) The Tire



(b) Swedish Wheel

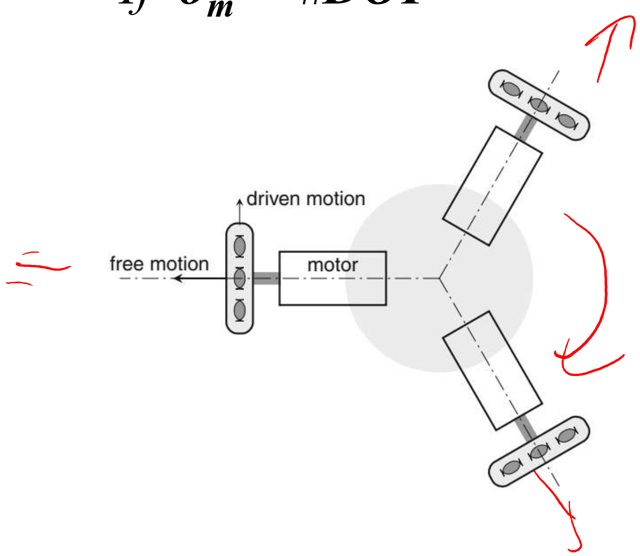


(c) Omnidirectional Vehicle

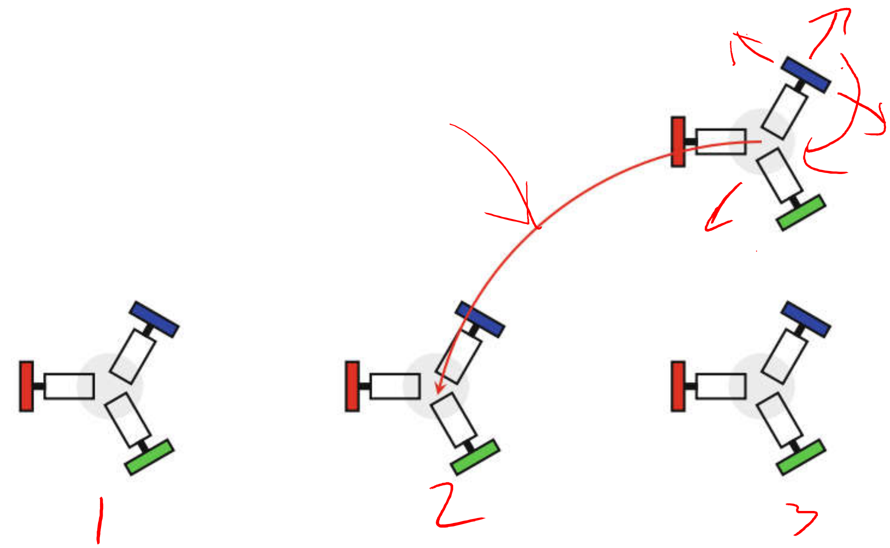
Actuators → DOF → DOM

Holonomic & Non-holonomic Motion

- Holonomic Motion
  - *If* $\delta_m = \#DOF$
- Non-holonomic Motion
  - *If* $\delta_m < \#DOF$



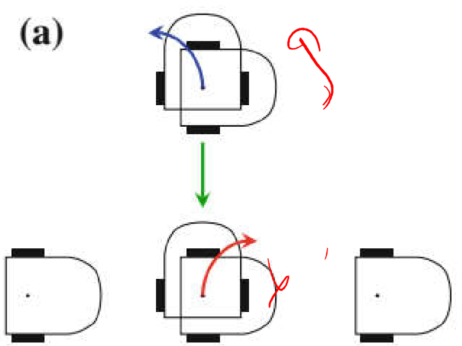
(a) Omnidirectional robot w/ 3 Swedish wheels



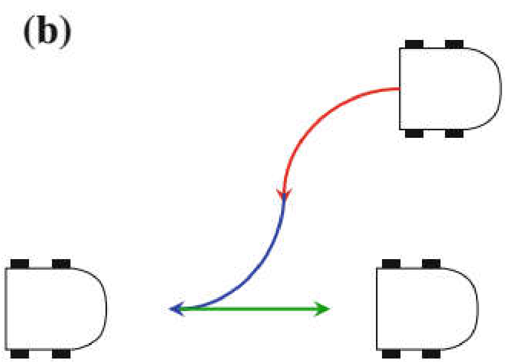
(b) Parallel parking by an omnidirectional vehicle

Holonomic & Non-holonomic Motion

- Non-holonomic vehicles
  - *Need complex maneuverers to parallel park*
- Differential drive robot
  - *Three separate simple movements (rotate left → move back → rotate right)*
- Car
  - *Three separate extremely difficult movements*



(a) Parallel parking for non-holonomic diff drive robot



(b) Parallel parking for a non-holonomic car

Summary

- **Odometry**
  - *Estimate speed & rotational vel from motor power to get position*
  - *Trigonometric calculations needed when turning*
  - *Still errors and very large if error is in heading*
  - *Wheel encoders improve odometry*
- **Inertial navigation**
  - *Use accelerometers & gyroscopes to improve accuracy*
  - *Microelectromechanical systems made IN cheap enough for robotics use*
- **DOF is number of dimensions w/c it can move**
  - *Num & config of actuators define its degree of mobility*
  - *Holonomic: DOM = DOF, Non-holonomic: DOM < DOF*

% = 10 %

LG HG

± 1m ± 5m

Thank you.