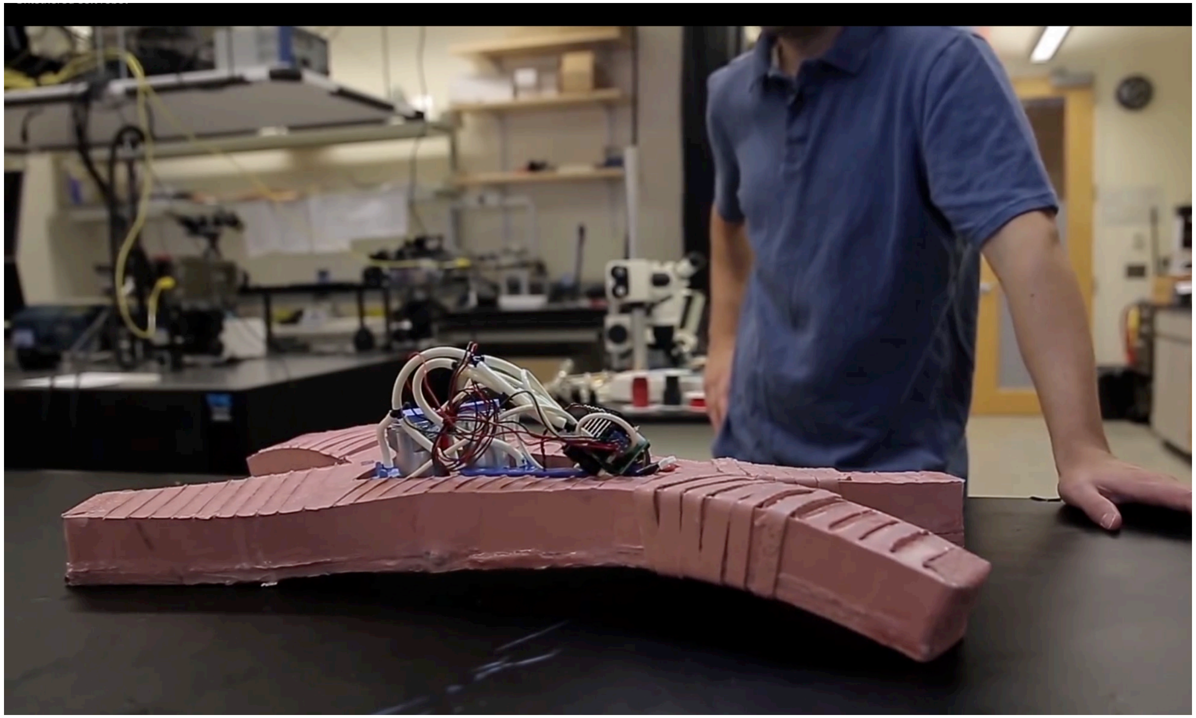
E160 – Lecture 4
Autonomous Robot Navigation

Instructor: Chris Clark

Semester: Spring 2016

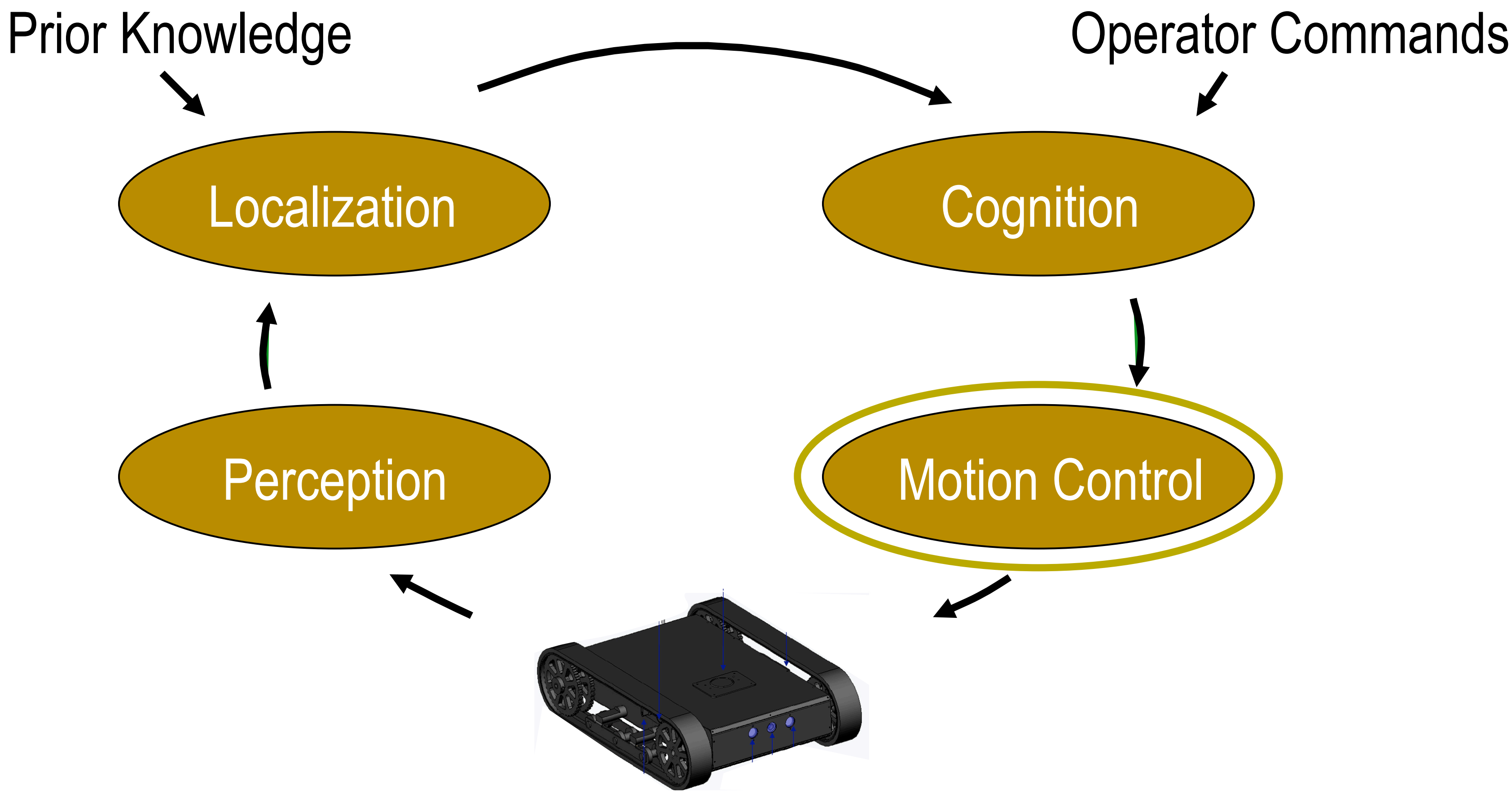
Figures courtesy of Siegwart & Nourbakhsh

Soft Robotics



https://www.youtube.com/watch?v=_OJrwCP24cI

Control Structures
Planning Based Control



Point Tracking

1. P Control
2. Linear Systems
3. Motion Control
4. Reachable Space

P Control

- Proportional Feedback Control – P Control
  - Uses the error between the desired and measured state to determine the control signal.

P Control

- If $x_{desired}$ is the desired state, and x is the actual state, we define the error as:

$$e = x_{desired} - x$$

P Control

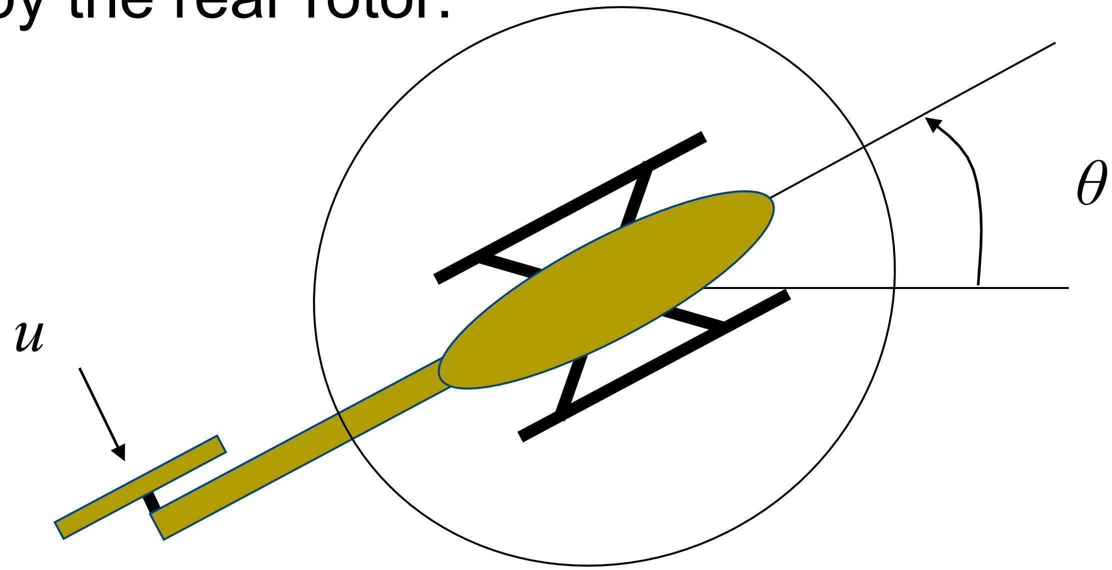
- The control signal u is calculated as

$$u = K_P e$$

where K_P is called the proportional gain.

P Control

- Example:
  - Consider the orientation control of an autonomous helicopter. Assume the orientation is completely controlled by the rear rotor.



P Control

- Example cont':
  - The control signal u is calculated as

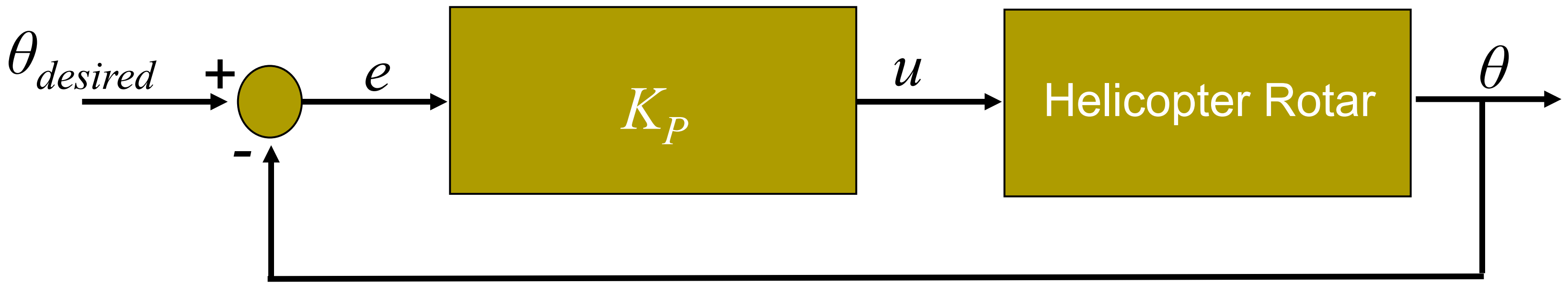
$$u = K_P(\theta_{desired} - \theta)$$

  - Notes:
    - If $\theta_{desired} = \theta$, the control signal is 0.
    - If $\theta_{desired} < \theta$, the control signal is negative, resulting in an decrease in θ.
    - If $\theta_{desired} > \theta$, the control signal is positive, resulting in an increase in θ.
    - The magnitude of the increase/decrease depends on K_p

P Control

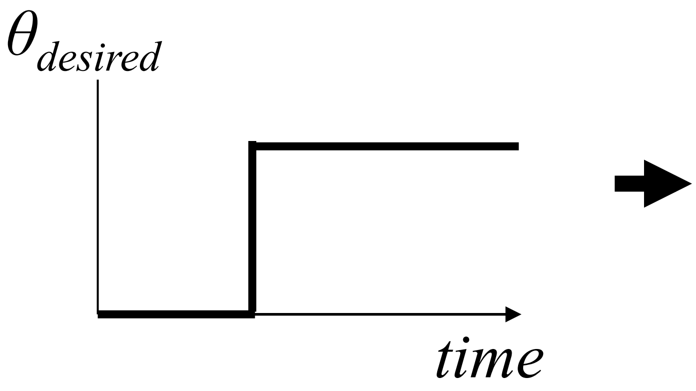
- Block Diagram:

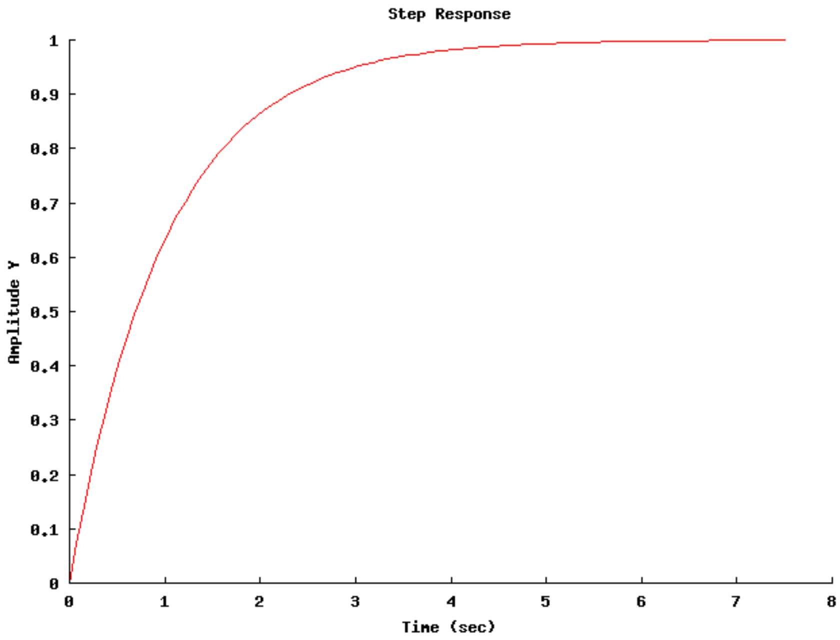
$$u = K_P(\theta_{desired} - \theta)$$



P Control

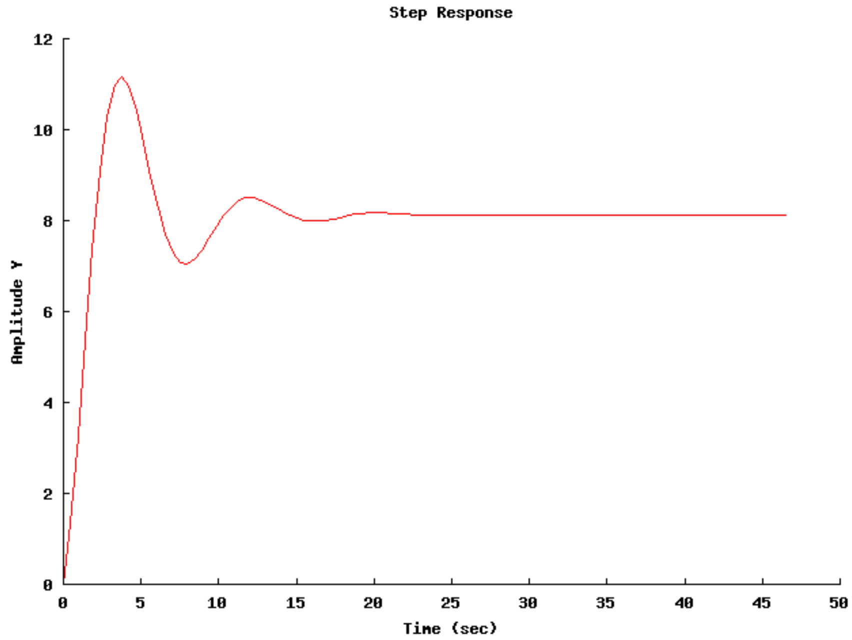
- Time Domain Response of step response
  - Step from $\theta_{desired} = 0$ to $\theta_{desired} = 1.$





P Control

- Time Domain Response:
  - Step from $\theta_{desired} = 0$ to $\theta_{desired} = 8$.
  - Different dynamics in this example… overshoot!



Point Tracking

1. P Control
2. Linear Systems
3. Motion Control
4. Reachable Space

Linear Systems

- Recall that the forward kinematics are a linear differential equation.
- We will use this equation to help develop a motion controller for point tracking
- We start by observing how the state x behaves if it obeys the following equation:

$$\dot{x} = dx/dt = ax$$

where a is a constant

Linear Systems

- It should be obvious that the solution to the equation

$$\dot{x} = ax$$

is

$$x(t) = x_0 \exp(at)$$

where

x_0 is the initial state

Linear Systems

- To confirm this solution, substitute into the original equation:

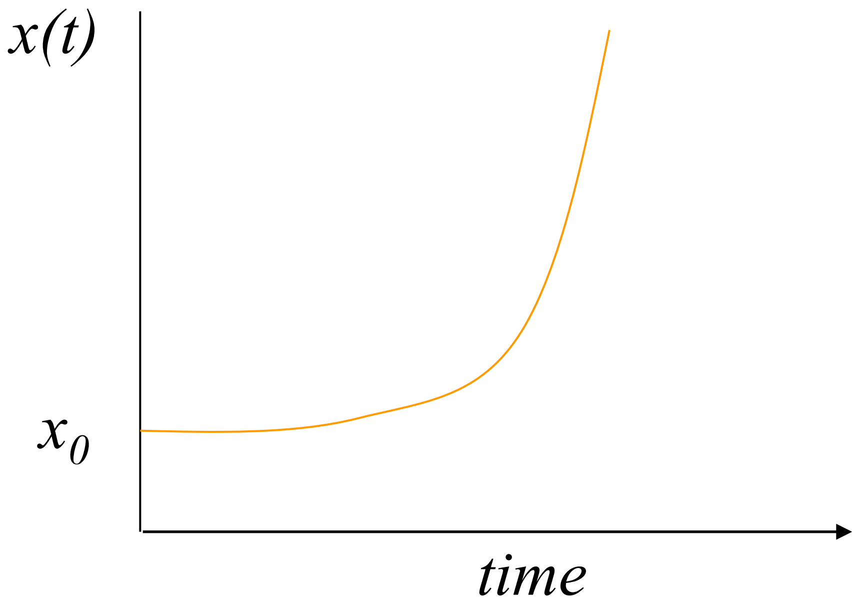
$$\dot{x} = ax$$

$$d[x_0 \exp(at)]/dt = a[x_0 \exp(at)]$$

$$ax_0 \exp(at) = ax_0 \exp(at)$$

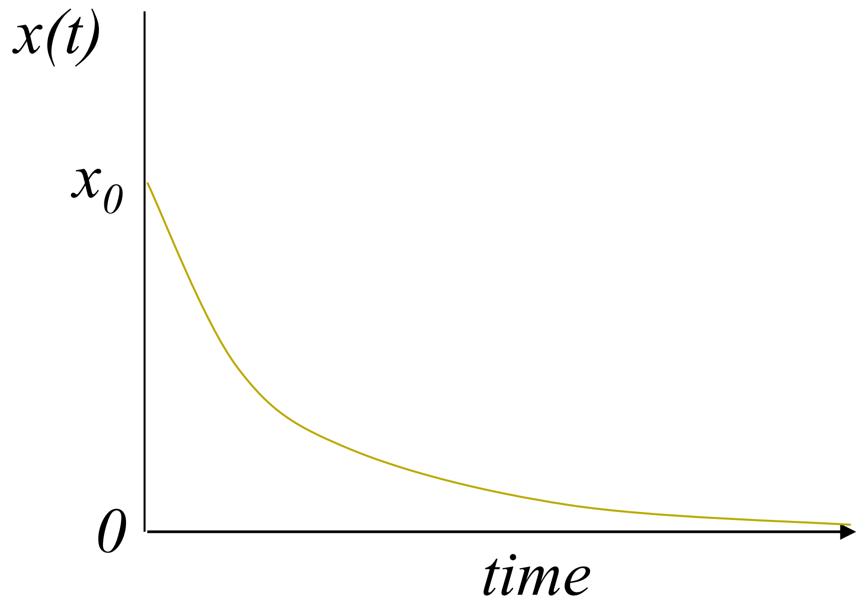
Linear Systems

- To view how the state x behaves over time, we can plot out $x = x_0\, exp(at)$, assuming a is positive:



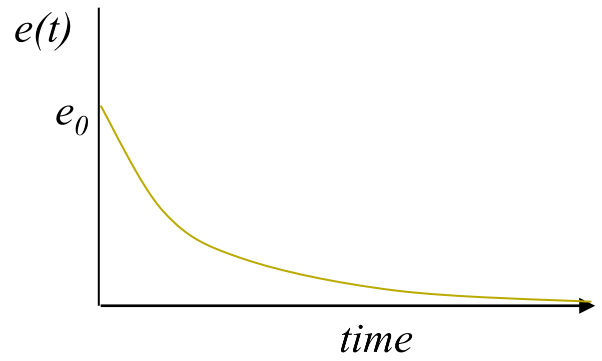
Linear Systems

- If a is negative and we can plot out $x = x_0 \exp(at)$, we get much different results:



Linear Systems

- This exponential decay informs us that the state x decays to zero over time.
  - We say this system is “STABLE”.
  - We use this property in control theory to drive states down to zero (e.g. if $e = x_{desired} - x$, drive e to 0).



Linear Systems

- The above example was a one dimensional linear system (i.e. single state x).
- Our system is a multi-dimensional system (i.e. 3 states x, y, θ).
- We need to describe the system with matrices:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x}$$

where A is a matrix such that $A \in R^{nxn}$

$\boldsymbol{x}$ is a vector such that $\boldsymbol{x} \in R^{1xn}$

Linear Systems

- The eigen values of A, represented by λ_i, are coefficients that satisfy the equation:

$$A\boldsymbol{x}_i = \lambda_i \boldsymbol{x_i}$$

for particular states called x_i, called the eigen vectors.

- A solution to the system can be written as the combination of eigen vectors:

$$\boldsymbol{x(t)} = \boldsymbol{x_1} e^{\lambda_1 t} + \boldsymbol{x_2} e^{\lambda_2 t} + \ldots + \boldsymbol{x_n} e^{\lambda_n t}$$

Linear Systems

- In this case, the system

$$\dot{x} = Ax$$

is said to be stable if the eigen-values of A are less than 0.

Linear Systems

- We solve for eigen values by noting:

$$(A - \lambda I)\mathbf{x} = 0$$

- For this to hold true,

$$\det (A - \lambda I) = 0$$

Linear Systems

- Example:

$$A = \begin{bmatrix} 3 & 6 \\ 1 & 4 \end{bmatrix}.$$

$$\begin{aligned} A - \lambda I &= \begin{bmatrix} 3-\lambda & 6 \\ 1 & 4-\lambda \end{bmatrix} \\ \det(A - \lambda I) &= (3-\lambda)(4-\lambda) - 6 \\ &= \lambda^2 - 7\lambda + 6 \\ &= (\lambda - 6)(\lambda - 1) \end{aligned}$$

Therefore $\lambda_1 = 6, \lambda_2 = 1$

The system is not stable!

Linear Systems

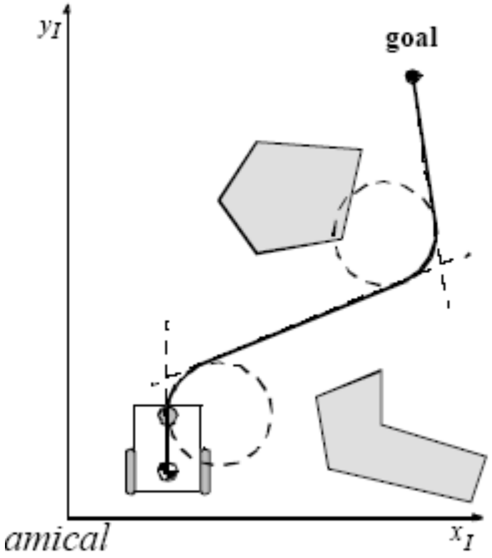
- Summary:
  - If our robot behaves like a system of the form $\dot{e} = Ae$, where the eigen values of A are negative and e represents the difference between desired and actual states, the system will move to our desired state!

Point Tracking

1. P Control
2. Linear Systems
3. Motion Control
4. Reachable Space

Motion Control

- Goal is to follow a trajectory from an initial state to some desired goal location.
- Several approaches
  - Could construct a global trajectory first, then track **points** on the trajectory locally



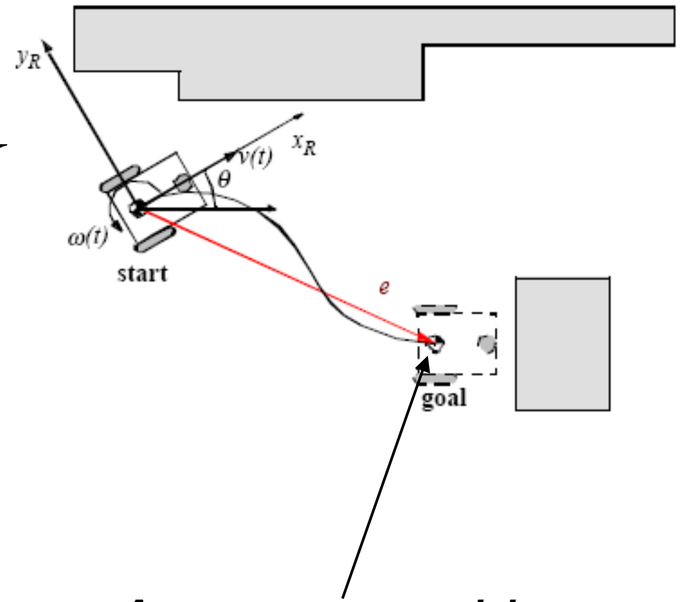
Motion Control

- If we define the error to be in the robot frame:

$$\mathbf{e}(t) = [\, x \; y \; \theta \,]^T$$

- Goal is to find gain matrix K such that control of $v(t)$ and $w(t)$ will drive the error $e(t)$ to zero.

$$\begin{bmatrix} v(t) \\ w(t) \end{bmatrix} = K\mathbf{e}(t)$$



Assume goal is at [0 0 0]

Motion Control

- Recall our forward kinematics

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix}$$

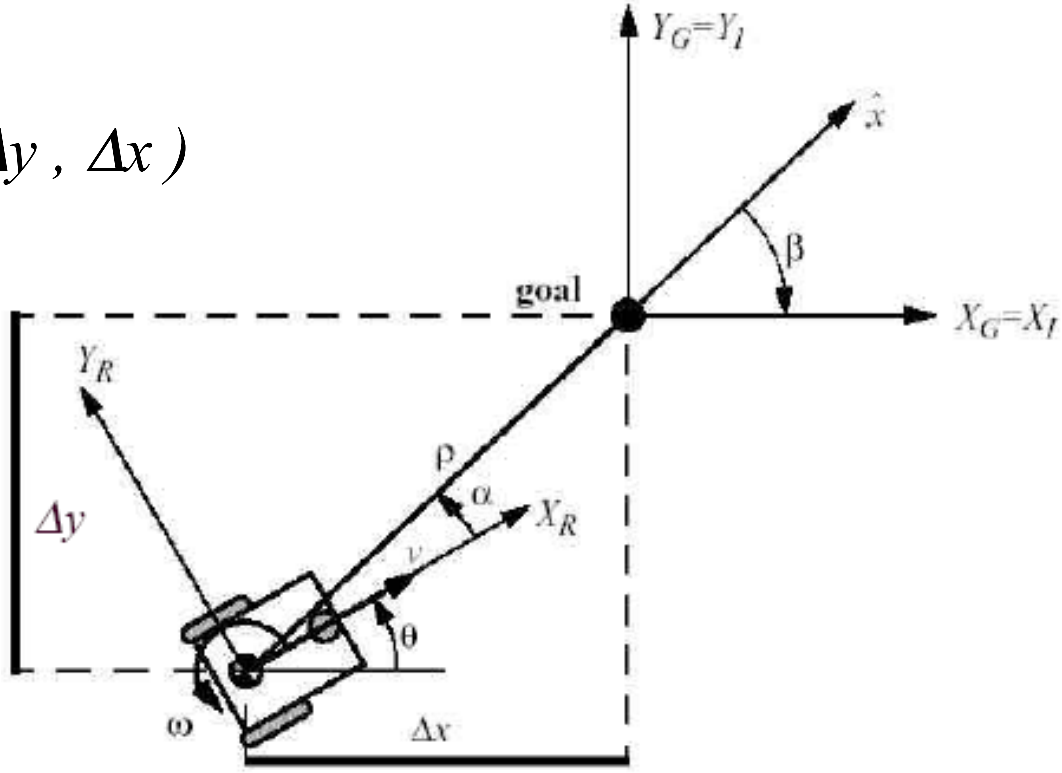
Motion Control

- We use the coordinate transformation

$$\rho = \sqrt{\Delta x^2 + \Delta y^2}$$

$$\alpha = -\theta + atan2(\Delta y, \Delta x)$$

$$\beta = -\theta - \alpha$$



Motion Control

- Now we define the problem as driving the robot to goal

$$\begin{pmatrix} \rho \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Motion Control

- We know this will happen if the dynamics of the system obey

$$\begin{pmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{pmatrix} = A \begin{pmatrix} \rho \\ \alpha \\ \beta \end{pmatrix}$$

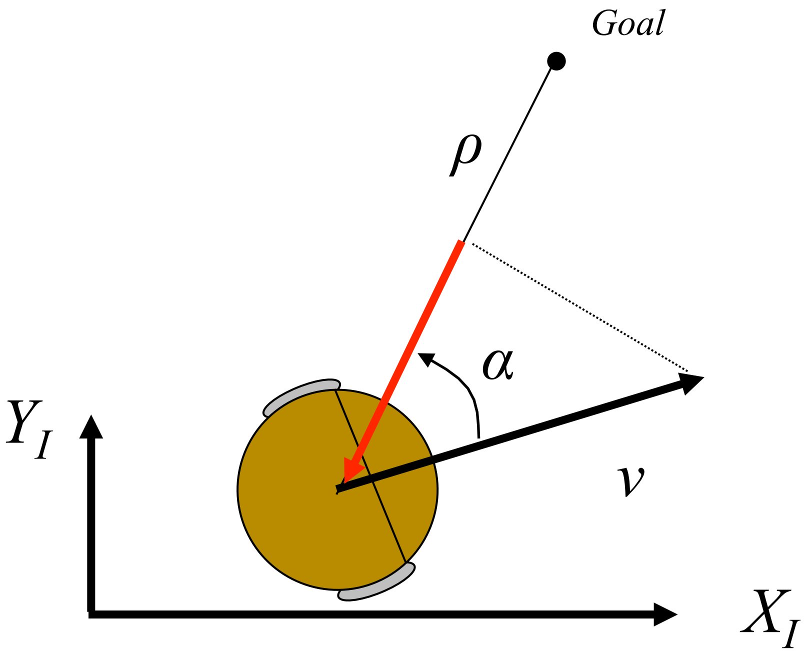
Where A is a $3x3$ matrix with eigen values less than 0.

Motion Control

- Using the coordinate transformation, calculate the new kinematics:

$\dot{\rho}$ = *projection of* v *on* ρ

= $- v \cos(\alpha)$



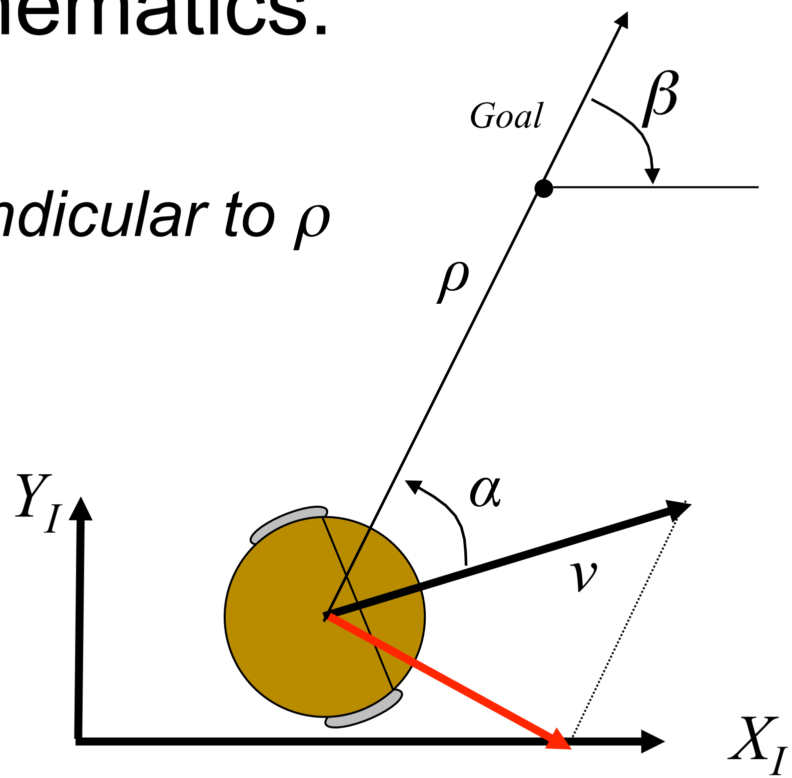
Motion Control

- Using the coordinate transformation, calculate the new kinematics:

$\rho\dot{\beta}$ = *projection of* v *perpendicular to* ρ

$$= -v\sin(\alpha)$$

$$\dot{\beta} = -v\sin(\alpha) / \rho$$



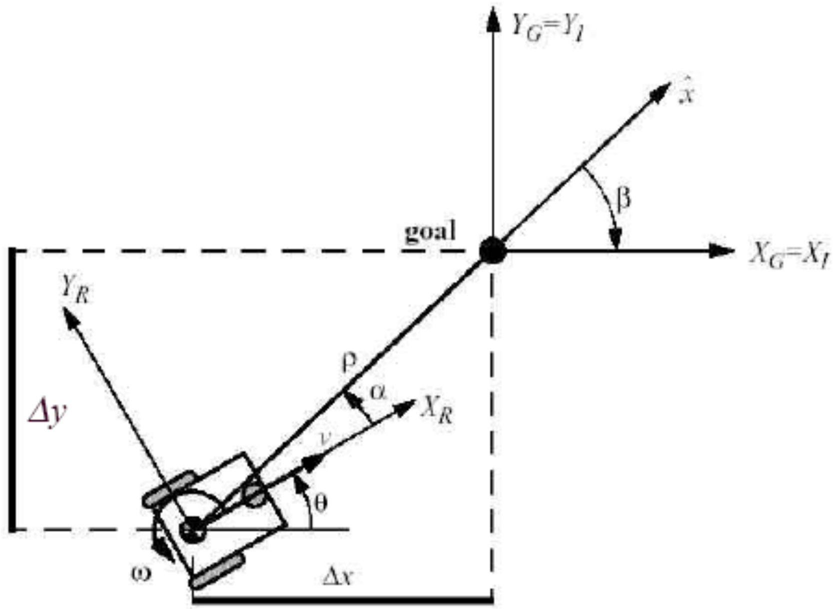
Motion Control

- Using the coordinate transformation, calculate the new kinematics:

$$\alpha = -\beta - \theta$$

$$\dot{\alpha} = -\dot{\beta} - \dot{\theta}$$

$$\dot{\alpha} = v \sin(\alpha)/\rho - w$$



Motion Control

- In matrix from:

$$\begin{pmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{pmatrix} = \begin{pmatrix} -\cos\alpha & 0 \\ \sin\alpha / \rho & -1 \\ -\sin\alpha / \rho & 0 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix}$$

for α within $(-\pi/2, \pi/2\,]$

Motion Control

- Let’s try the control law:

$$v = k_\rho \rho \qquad w = k_\alpha \alpha + k_\beta \beta$$

- Note that this is a form of P control, and if ρ, α, β all go to zero, then v and w will go to zero.

Motion Control

- To analyze controller, substitute control law into kinematics and linearize:
  - For small x, $\cos(x) \approx 1$ and $\sin(x) \approx x$
  - This is in the form…

$$\dot{\mathbf{e}} = A\,\mathbf{e}$$

Motion Control

- Check for stability:
  - Take the determinant of A and solving for eigen values leads to:

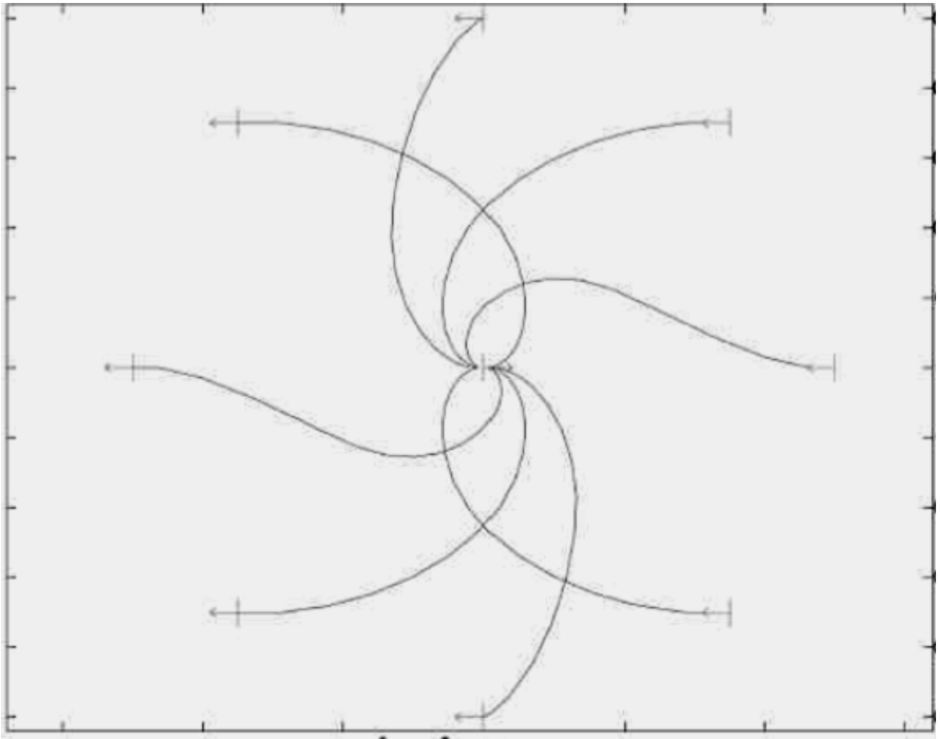
$$(\lambda + k_\rho)(\lambda^2 + \lambda(k_\alpha - k_\rho) - k_\rho k_\beta) = 0$$

  - Thus the system will be stable if:

$$k_\rho > 0 \qquad k_\beta < 0 \qquad k_\alpha - k_\rho > 0$$

Motion Control

- Testing this control law with many different start points:

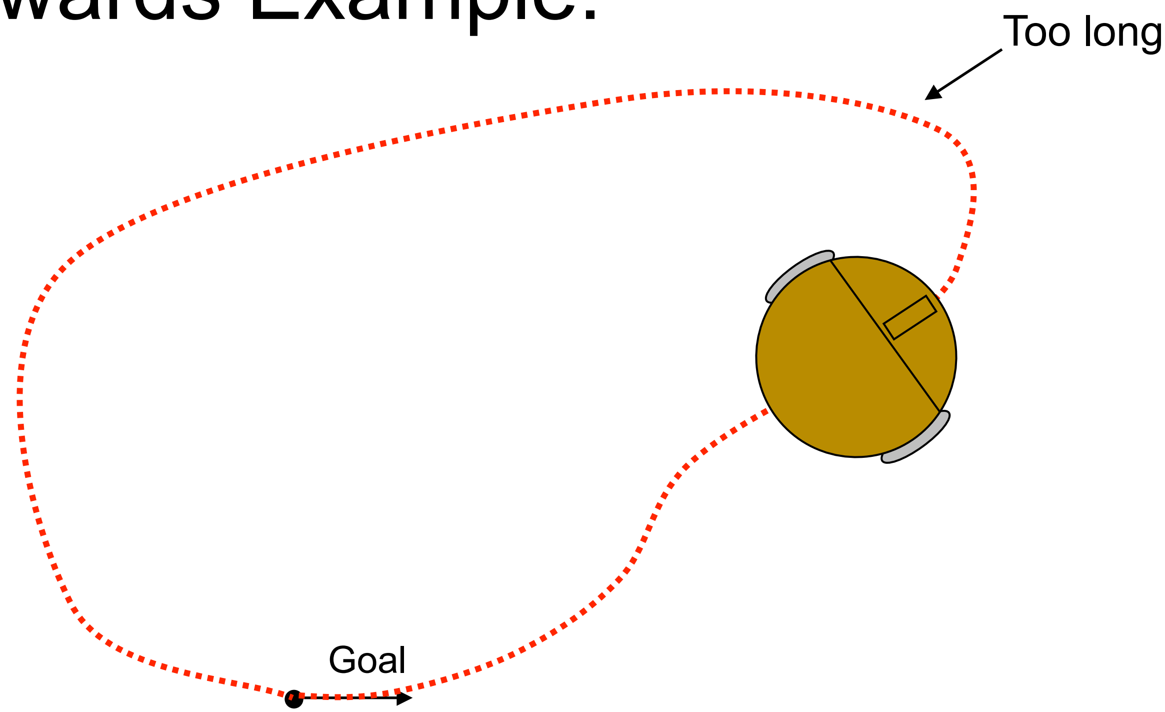


Motion Control

- The derived control law works well if $\alpha \in [-\pi/2, \pi/2]$
- For other cases where $abs(\alpha) > \pi/2$, we must modify the controller. So that the robot will move backwards to the desired position when required

Motion Control

- Backwards Example:



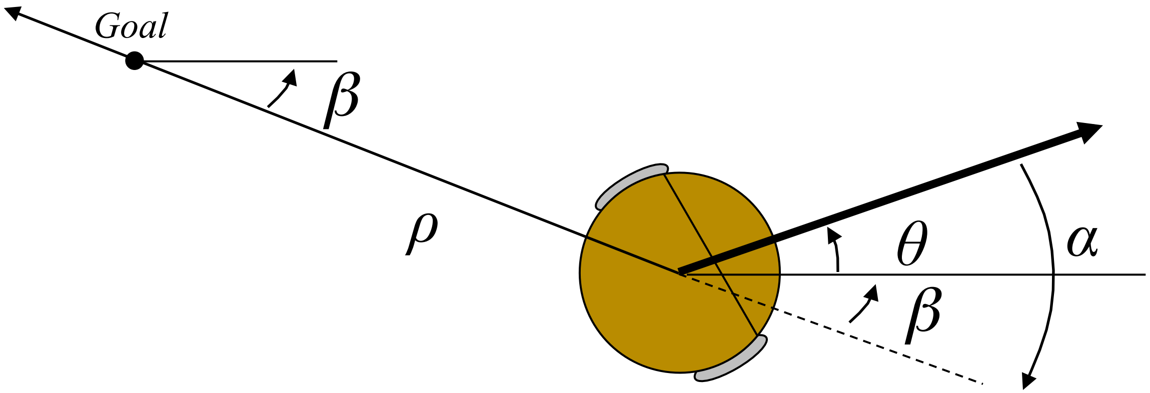
Motion Control

- Backwards Method:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2}$$

$$\alpha = -\theta + atan2(-\Delta y\, , -\Delta x\,)$$

$$\beta = -\theta - \alpha$$



Motion Control

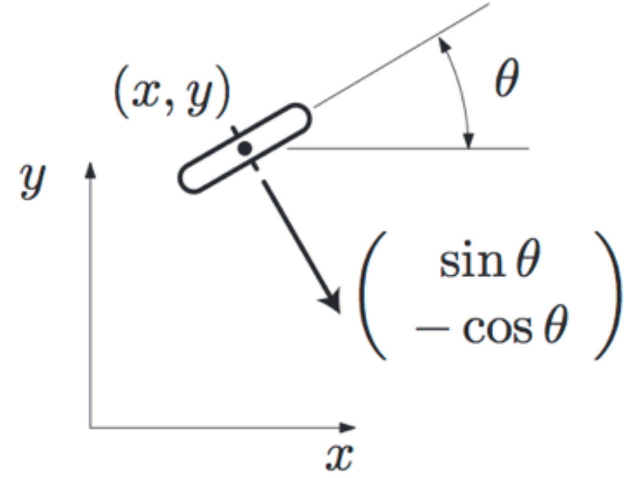
- Backwards Method Summary:
  - If $\alpha \in [-\pi/2, \pi/2]$
    - Use regular transform to polar coordinates
    - Use control law: $v = k_\rho \rho \quad w = k_\alpha \alpha + k_\beta \beta$
  - Else
    - Redefine α as shown in backwards method
    - Use control law: $v = -k_\rho \rho \quad w = k_\alpha \alpha + k_\beta \beta$

Point Tracking

1. P Control
2. Linear Systems
3. Motion Control
4. Reachable Space

Reachable Space

- Kinematic Constraints
  - One can calculate constraints on each individual wheel, then combine for constraints on entire robot.



http://www.cs.cmu.edu/afs/cs/academic/class/16741-s07/www/lecture5.pdf

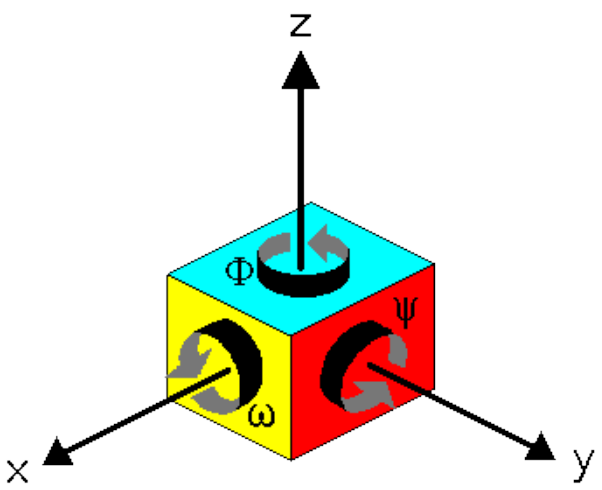
Reachable Space

- Two main constraints:
  1. Rolling Constraint: no slipping!
  2. Sliding Constraint: no lateral movement!

http://www.treadpros.com

Reachable Space

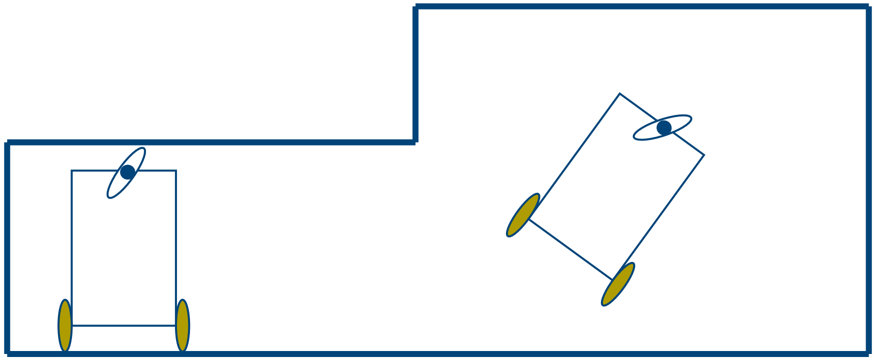
- Degrees of Freedom:
  - Def’n: The number of coordinates that it takes to uniquely specify the state of a system.
  - In 3D, there are 6 degrees of freedom associated to the movement of a rigid body: 3 for its position, and 3 for its orientation.



B J Stone, Univ. of Western Australia

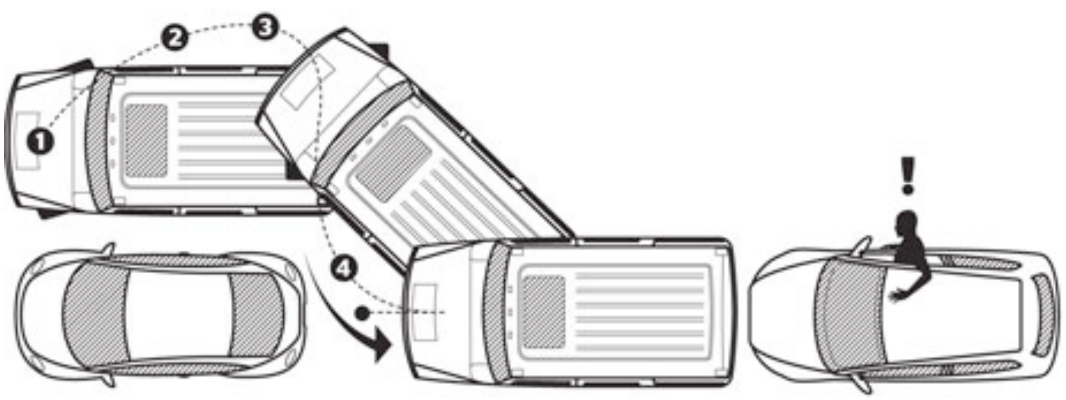
Reachable Space

- Configurations in the Workspace
  - A robot's workspace is defined by the Degrees Of Freedom of the robot state.
  - Not all robot configurations within the workspace are reachable



Reachable Space

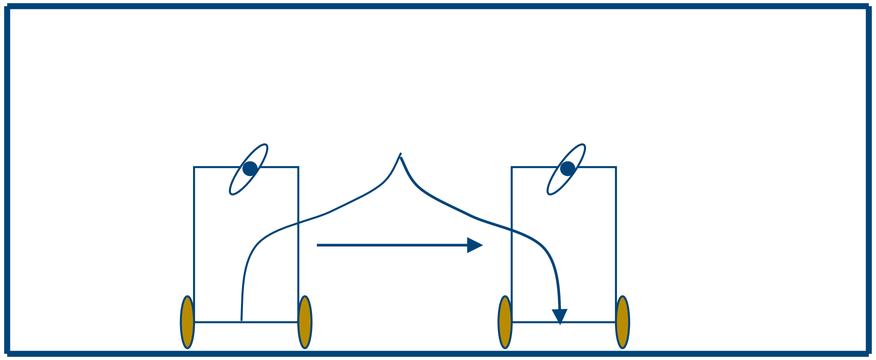
- NonHolonomic Robots
  - A nonholonomic constraint is one that is not integrable.



www.esquire.com

Reachable Space

- Paths in the Workspace
  - Path's in the workspace are limited, especially if the robot is nonholonomic



Reachable Space

- Trajectories in the Workspace
  - A trajectory is a path parameterized by time.
  - Admissible paths don't always lead to admissible trajectories.