Aalto University
School of Electrical
Engineering

Outdoor Motion Planning and Control

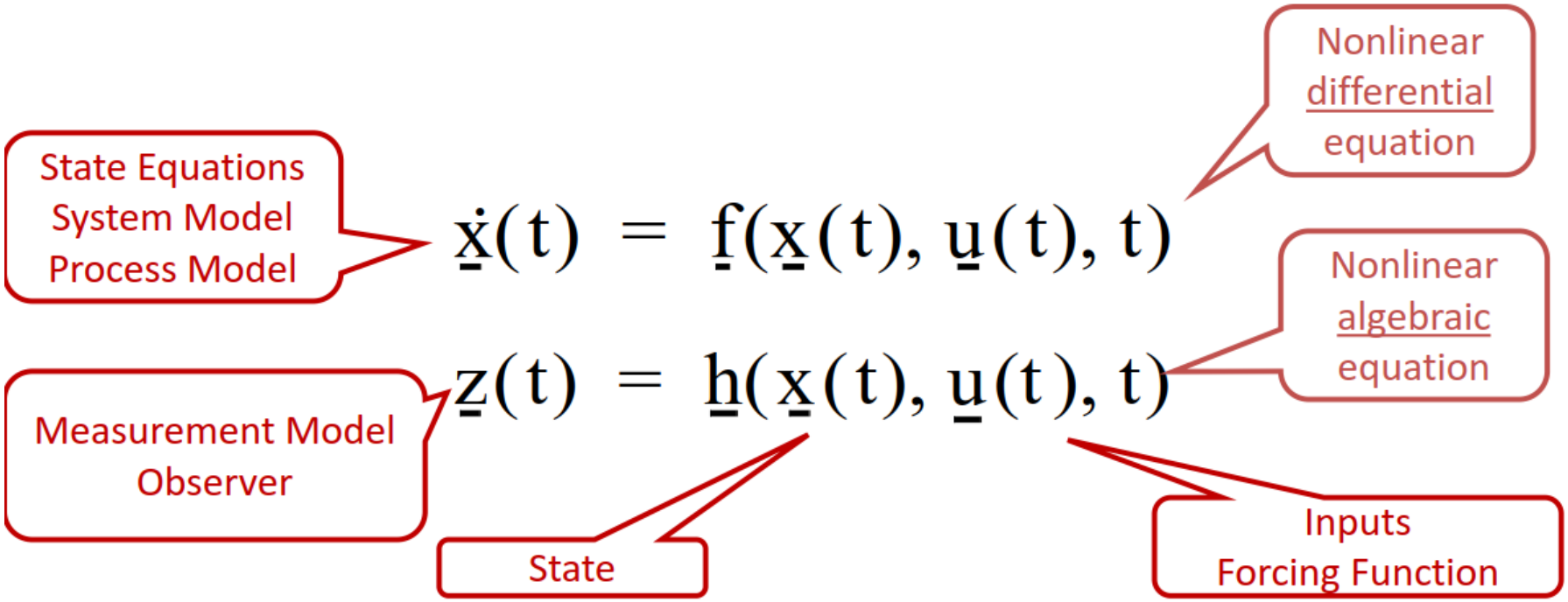
Mainly on the basis of book: Mobile Robotics - Prof Alonzo Kelly, CMU RI

April 4, 2019

Arto Visala

Predictive Modelling and System Identification

- Nonlinear **Dynamical** System, state-space model



Introduction – the “ives”, the role of Dynamics

Mobile robots must often be:

- Deliberative – decide among options
- Perceptive – aware of the surroundings
- Reactive – capable of fast action

They must be both

- smart and
- fast

… doing that involves tradeoffs.

In support of the above, need to be …

- Predictive – able to project consequences
- Active – able to execute a plan of action

You need dynamics models for both of these.

Predictive Modeling

Must model …

- Information processing and propagation.
- Physical vehicle / environment interaction.

Often need to map …

- what you can do (exert forces)
- what you care about (trajectory through space).

Latter requires integrating the dynamics.

Cases:

- Braking
- Turning
- Vehicle Rollover
- Wheel Slip and Yaw Stability

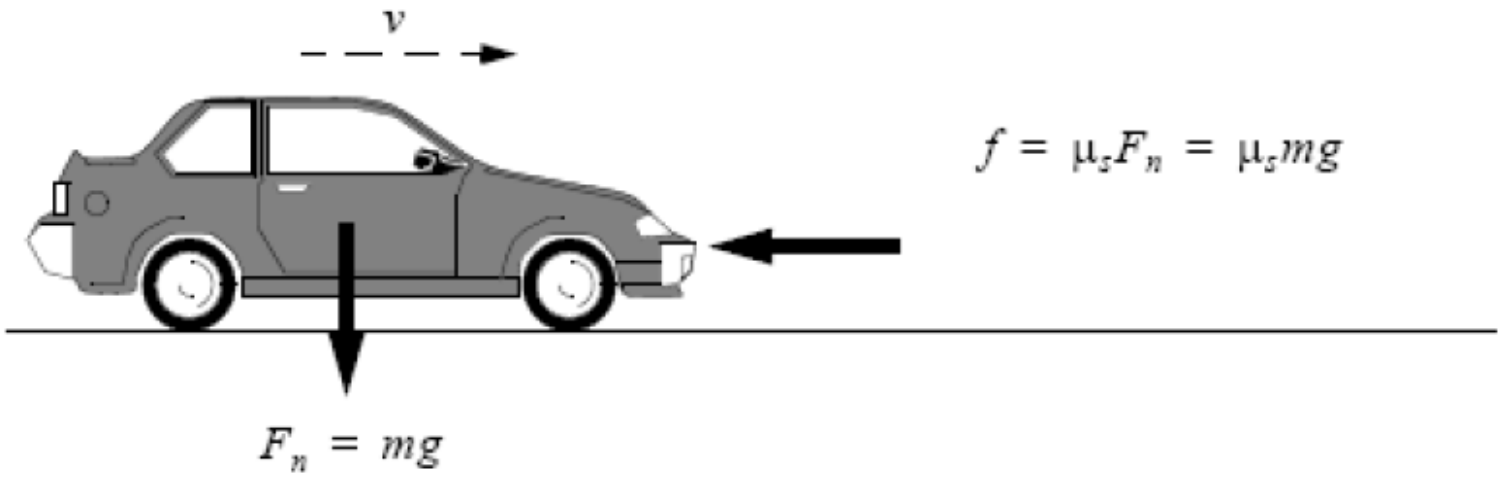
Reasons for Braking

A) Last resort response to problems.
- – Collision is imminent due to
  - • no solution or
  - • inadequate planning or control.

B) Deliberately slow down.
- – On slopes
- – The motion is finished.
- – In order to turn around.

Braking Model

- Assume brakes are applied instantly:
- Free body diagram:
  - Friction and Weight are coupled.
- Do heavier vehicles take longer to stop?



Braking Model

Equate work done by external forces to initial Kinetic energy (assume it is all used up).

$$\frac{1}{2}mv^2 = \mu_s mgs_{brake}$$

Solve for braking distance:

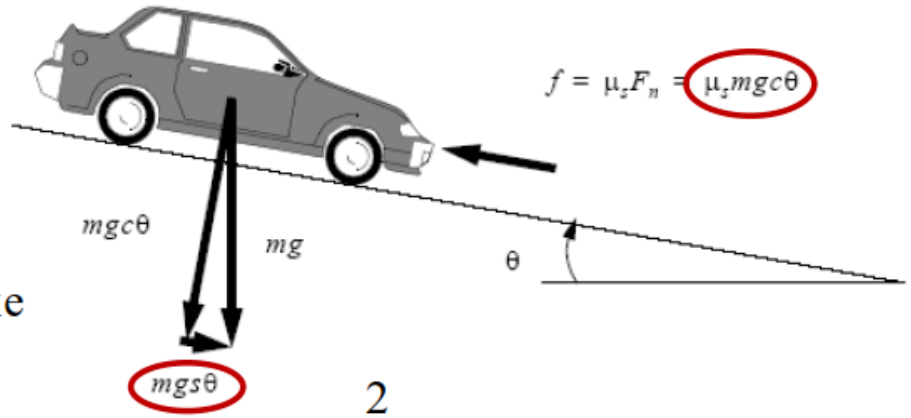
$$s_{brake} = \frac{v^2}{2\mu_s g}$$

Do heavier vehicles take more distance to stop?

Impact of Slopes

•Again equate work done to initial KE:

$$\frac{1}{2}mv^2 = (\mu_s mgc\theta - mgs\theta)s_{brake}$$



Solve for braking distance:

$$s_{brake} = \frac{v^2}{2g(\mu_s c\theta - s\theta)}$$

Effective coefficient of friction:

$$\mu_{eff} = (\mu_s c\theta - s\theta)$$

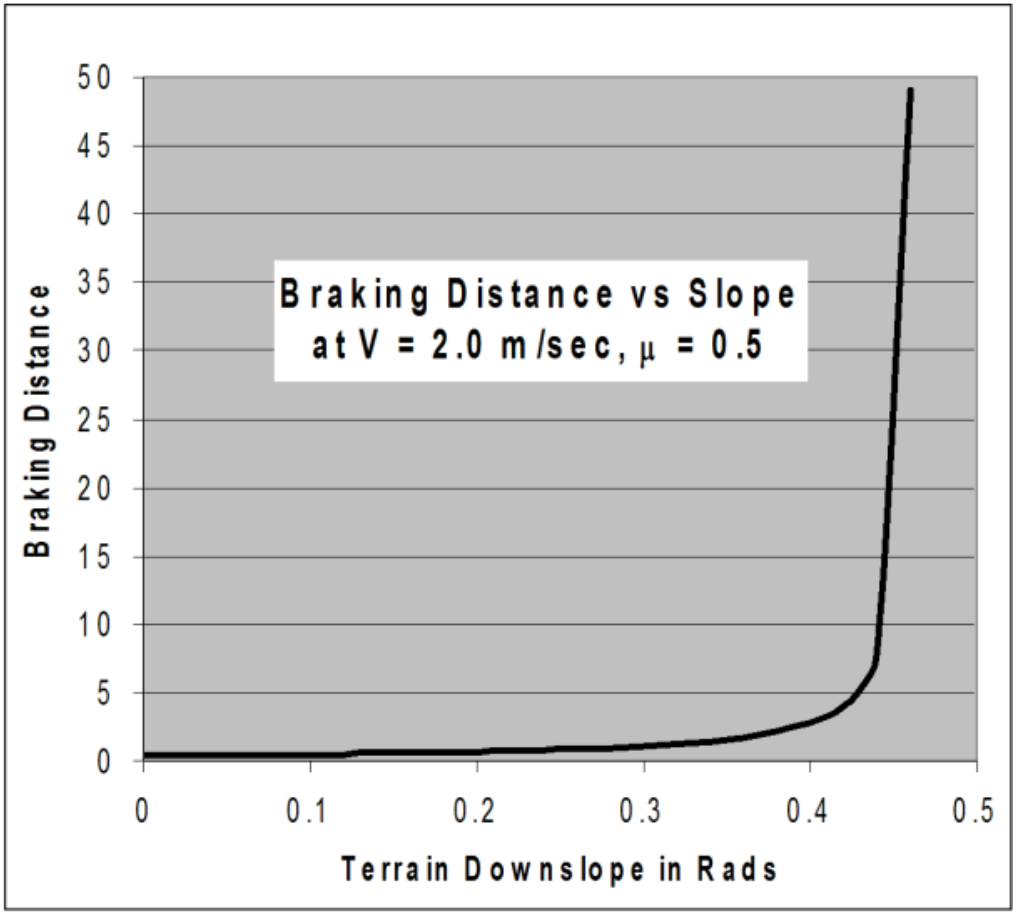
Then:

$$s_{brake} = \frac{v^2}{2\mu_{eff} g}$$

Simple Model on Slopes

Critical angle exists beyond which gravity overcomes friction….

Atan() is highly nonlinear.



General Case in Braking

More generally:

$$\int_0^s \vec{F} \bullet \vec{ds} = \frac{1}{2} m v^2$$

Robots can compute this.

- The terrain shape is known.
- Keep integrating until Kinetic Energy exhausted.
- Final value of s is stopping distance.

Rough Heuristic for Slopes

Make small angle assumptions, angles in radians:

$$c\theta = 1 \quad s\theta = \theta$$

Change in effective coefficient:

$$\mu_{eff}(\theta) = (\mu_s c\theta - s\theta) \approx \mu_s - \theta$$

10% slope reduces μ by 0.1

Ratio of sloped to level stopping distance:

$$\frac{s_\theta}{s_0} = \left[\frac{1}{1 - \frac{\theta}{\mu_s}}\right] \approx \left[1 + \frac{\theta}{\mu_s}\right]$$

Stopping distance increases or decreases by the factor

$$\theta / \mu_s$$

Turning

Goal is to cause terrain to exert a moment on the vehicle

- By 3rd law, vehicle must exert a moment on the terrain.

May actuate:

- Wheel steering (Ackerman)
- Wheel speeds (Differential, skid)

Simple Motion Prediction

For small steer angles: curvature $\kappa(t) = \alpha(t)$

Integrate the differential equations using "back substitution":

The mapping from steer angle and velocity onto the path the robot follows. Assumes flat terrain.

$$\theta(t) = \theta_0 + \int_0^t V(t)\alpha(t)dt$$

$$x(t) = x_0 + \int_0^t V(t)\cos(\theta(t))dt$$

$$y(t) = y_0 + \int_0^t V(t)\sin(\theta(t))dt$$

Note mapping from inputs to outputs are integrals.

Errors in steering are integrated twice to determine errors in predicted position.

Reverse Turn, multiple speeds; speed coupling

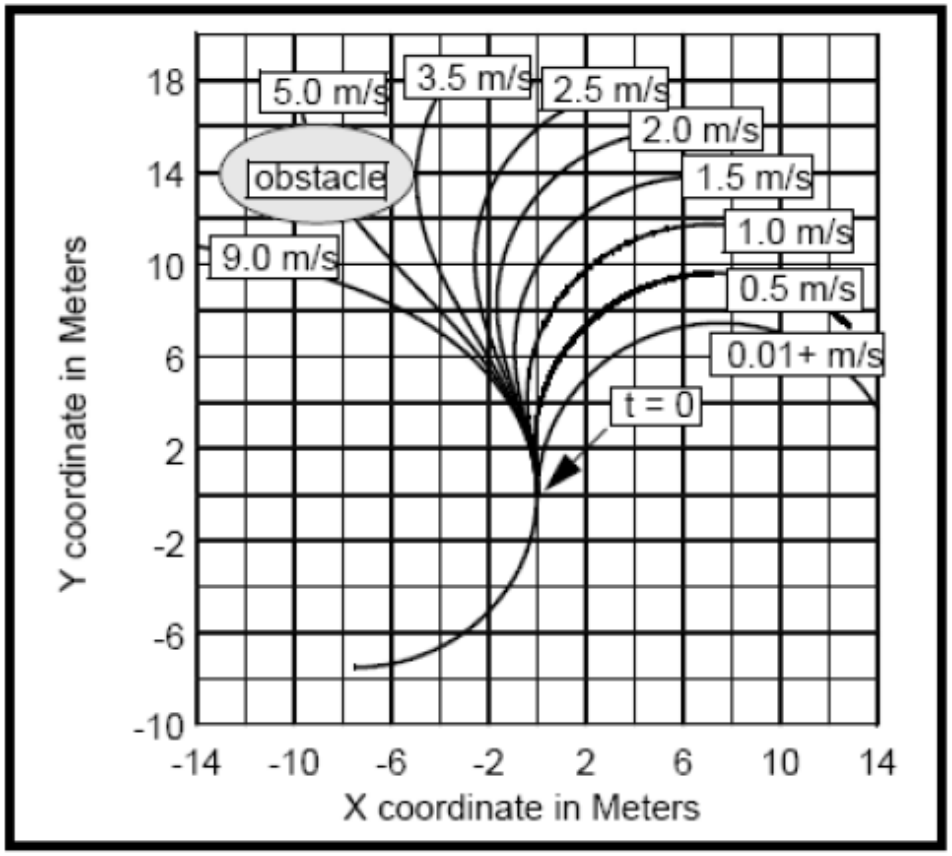
A curvature step is the most ambitious maneuver.

Not modelled steering response leads to collisions with obstacles above 3.5 m/sec speed

One Curvature
Various Speeds

The path followed is generally a function of speed.

Therefore, they must be estimated together.



"Reverse Turn"

Slipping and rollover

Two limits on curvature (slipping and rollover) can be computed

Assuming velocity is constant, and curvature rate is limited and constant

$$x = v\int \cos(\frac{v\dot{\kappa}_{\max}}{2}t^2)dt$$

$$y = v\int \sin(\frac{v\dot{\kappa}_{\max}}{2}t^2)dt$$

Slipping, centrifugal > friction

$$\kappa_{slip} = \frac{\mu g}{v^2}$$

Rollover, centrifugall > gravitation

$$\kappa_{roll} = \frac{T}{2hv^2}g$$

Vehicle Rollover

Contemporary mining, forestry, agriculture, and military vehicles, operate

- on slopes and/or
- at high speeds

Field robots do rollover!

- They at least need a reactive system if predictive elements fail

Vehicle Rollover

More likely in factory and field robots.

Happens due to combinations of:

– narrow wheel spacing,

– high centers of gravity

– high inertial forces (speeds and curvatures)

– steep slopes

Incidents may be:

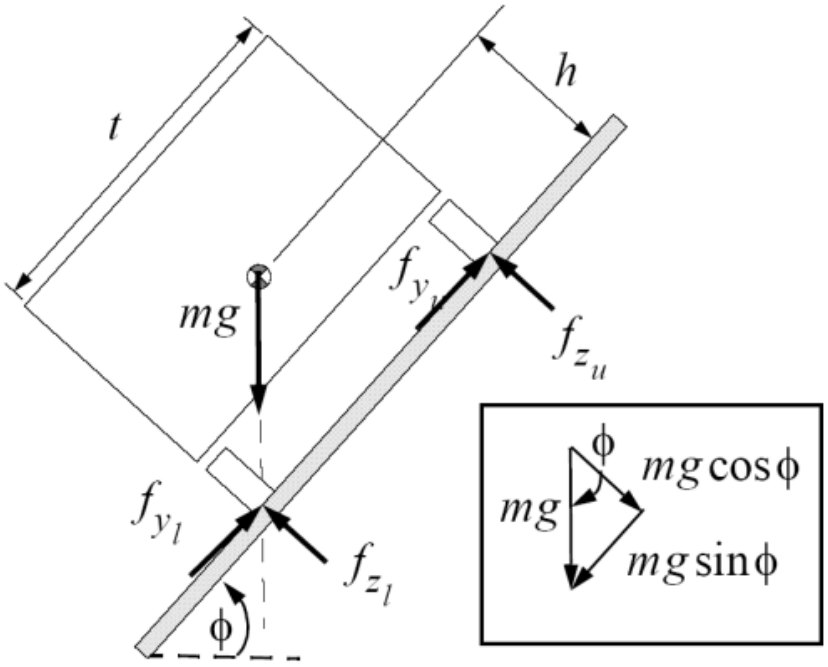
– Terrain induced (slide sideways into a curb)

– Maneuver induced (turn too sharp on a hill)

Vehicle Rollover, Static Liftoff

Since we are talking about a moment of a single force…
– Result can be understood in terms of the direction of gravity.

Liftoff criterion is first satisfied when gravity vector:
– emanating from the center of gravity (cg)
– points at the lower wheel contact point.



Vehicle Rollover, Dynamic Case

Use D'Alemberts principle:
– I.E. treat – ma like a real force.

Moment balance:

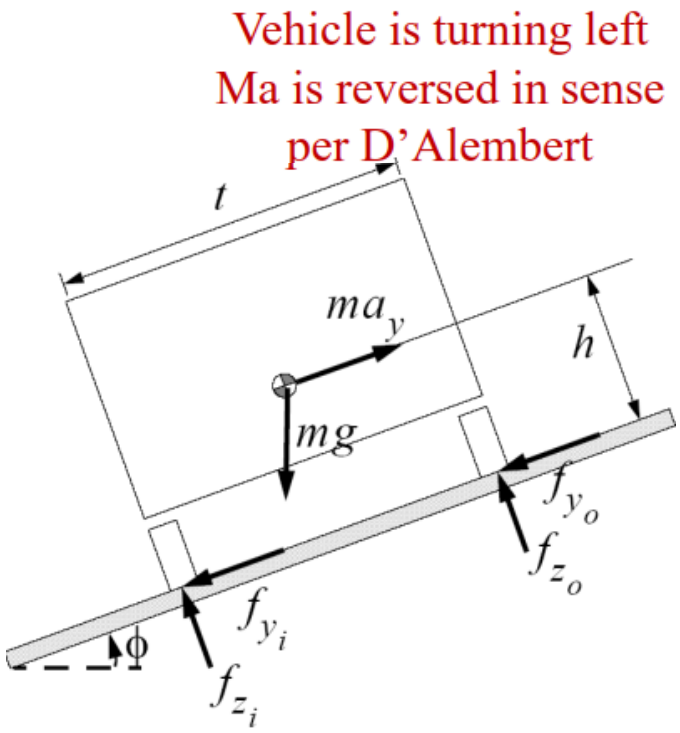
$$-f_{z_i}t - ma_y h + mgs\phi h + mgc\phi\frac{t}{2} = 0$$

Solve for lateral acceleration in g's:

$$\frac{a_y}{g} = \left[\frac{t}{2}c\phi + hs\phi - \frac{tf_{z_i}}{mg}\right]/h$$

The lateral acceleration threshold.

$$\boxed{\frac{a_y}{g} = \left[\frac{t}{2}c\phi + hs\phi\right]/h}$$



Vehicle Rollover, Dynamic Case

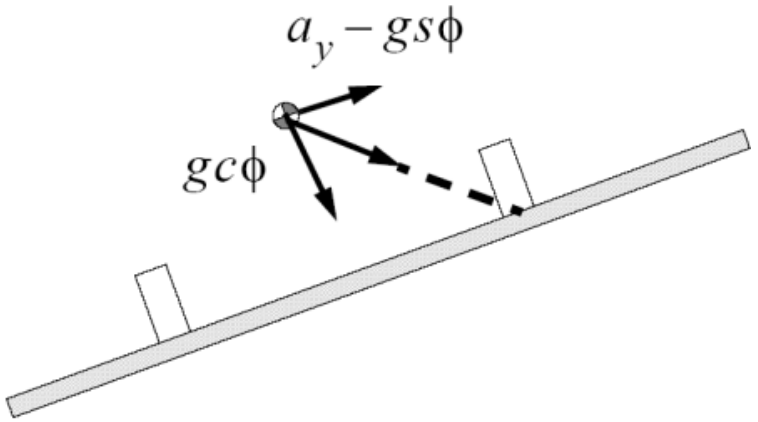
Rewrite last result:

$$\frac{a_y - gs\phi}{gc\phi} = \frac{t}{2h}$$

Liftoff when net noncontact specific force:

$$\vec{f} = \vec{g} - \vec{a}$$

Points at the outside wheel contact point.



A pendulum mounted at the cg aligns with this

Vehicle Rollover, Dynamic Case, Interpretations

Static case is just special case of dynamic (ay=0)

Stability increases with:

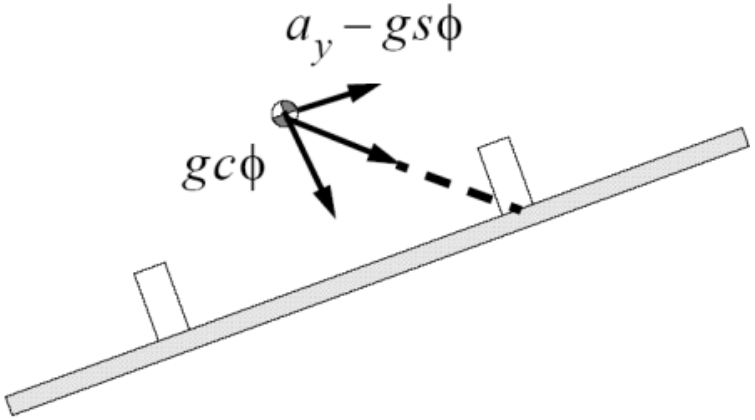
– Lower cg h

– Wider tread t

– Lowering slope

– Decreasing acceleration

- Slowing down
- Reducing curvature



$$\frac{a_y - gs\phi}{gc\phi} = \frac{t}{2h}$$

Stability Pyramid

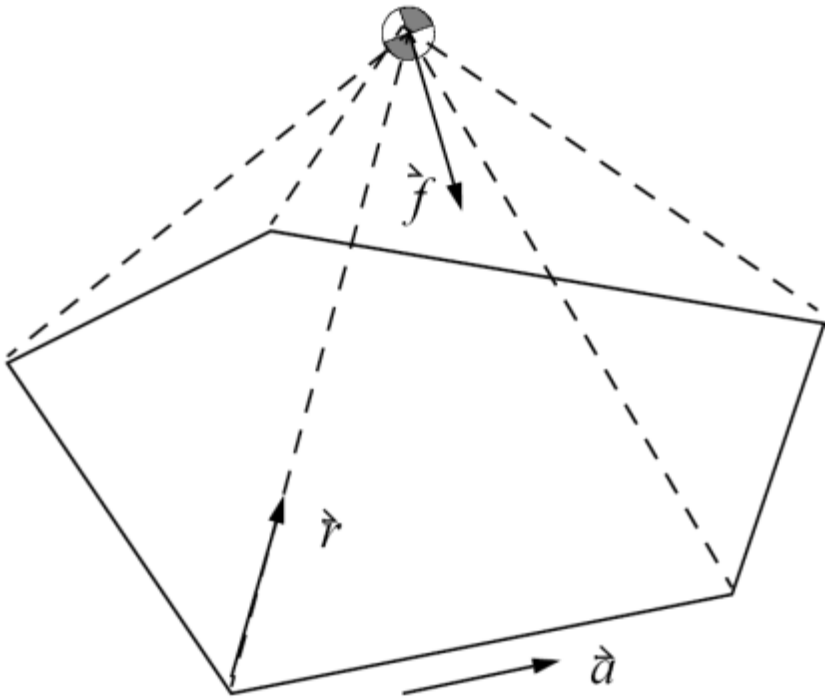
Theory generalizes to vehicles of any shape.

Stability pyramid = the pyramid formed with the wheel contact points with the cg at the apex. Each edge is a potential tipover axis.

– Unbalanced when: net noncontact specific force is outside one of the edges

$$\vec{M} = \vec{r} \times \vec{f}$$

$$\vec{M} \bullet \hat{a} > 0$$



Wheels need not be in the same plane.

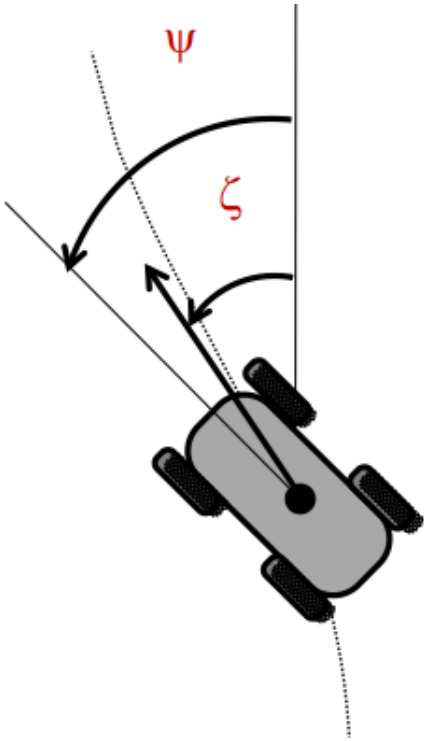
Wheel Slip and Yaw Stability

Slip Angle $\beta = \psi - \zeta$

Define the angle between the actual and intended Velocity

$$\beta = \operatorname{acos}\left[(\tilde{\underline{V}} \cdot \underline{V})/(|\tilde{\underline{V}}||\underline{V}|)\right]$$

actual reference



The velocity may be incorrect in all 3 degrees of freedom.

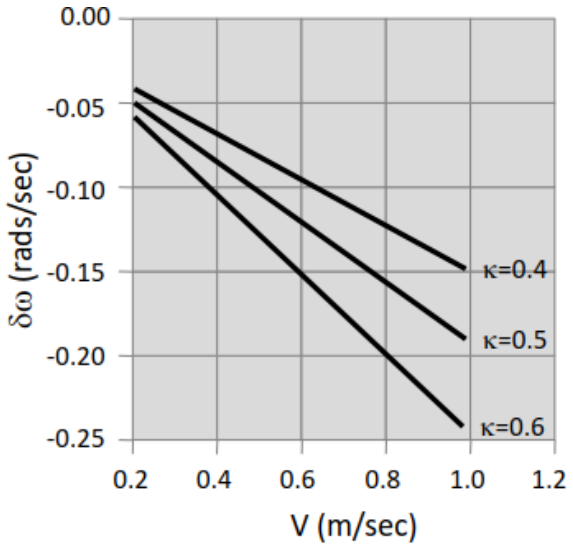
- Express errors in body coordinates:

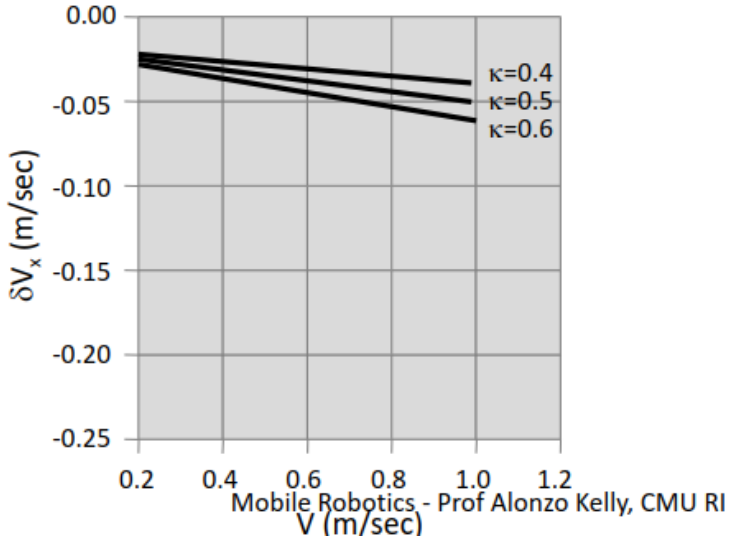
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} + \begin{bmatrix} \delta V_x \\ \delta V_y \\ \delta\omega \end{bmatrix} \right)$$

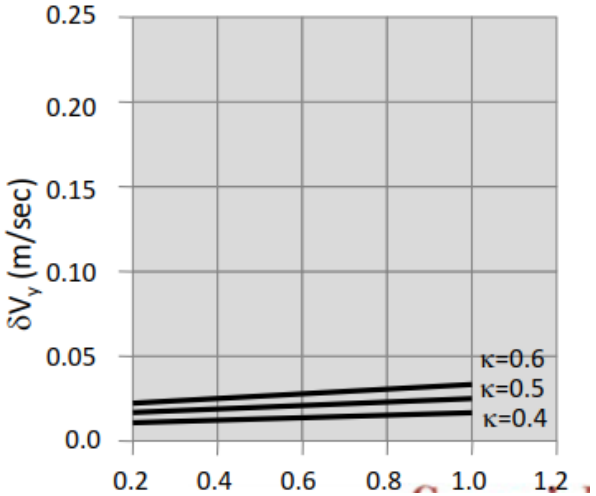
$$^{w}\tilde{\underline{V}} = R(\theta)(\underline{V} + \delta V)$$

actual reference

Wheel Slip Graphs



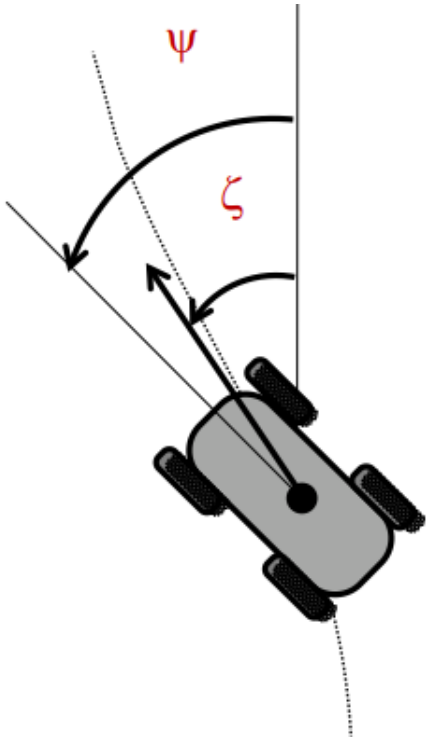




Removing Slip with Prediction

• Slip can be expressed as a function of actual or reference velocity (and other things):

• Compensate in body coordinates.



reference actual

$$\underline{V} = R^{-1}(\theta)\,{}^{w}\tilde{\underline{V}} - \delta V$$

Summary

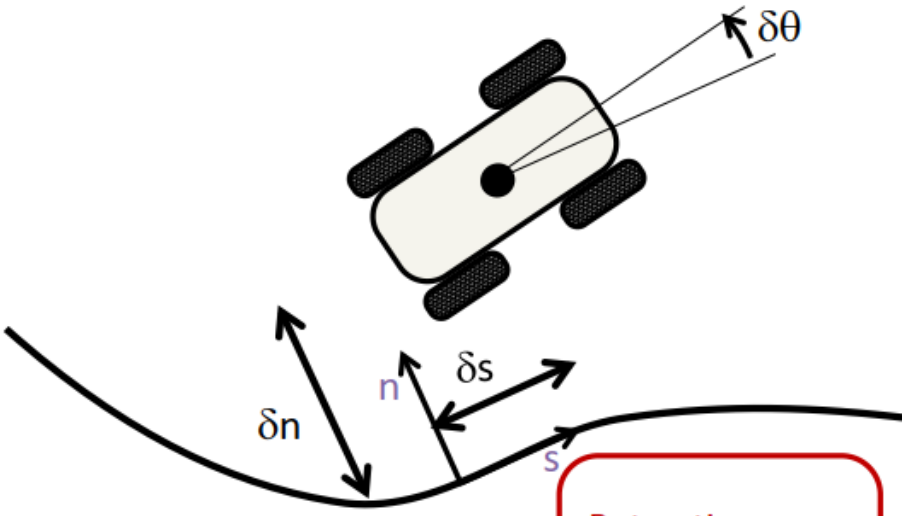
- Braking distance:
  - increases quadratically with initial speed
  - depends heavily on slope
- Turning and Swerving:
  - predicting steering maneuvers requires calibrated dynamic models.
- Rollover stability can be measured with a pendulum at the cg.

Control of mobile robot

1. Robot Trajectory Following
2. Perception Based Control
3. Steering Trajectory Generation
4. Optimal and Model Predictive Control
5. Intelligent Control

Representing Trajectories

- Assume velocity is fwd along body x only
- States: $\underline{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$
- Inputs: $\underline{u} = \begin{bmatrix} \kappa & V \end{bmatrix}^T$
- Nonlinear state space model in terms of time.



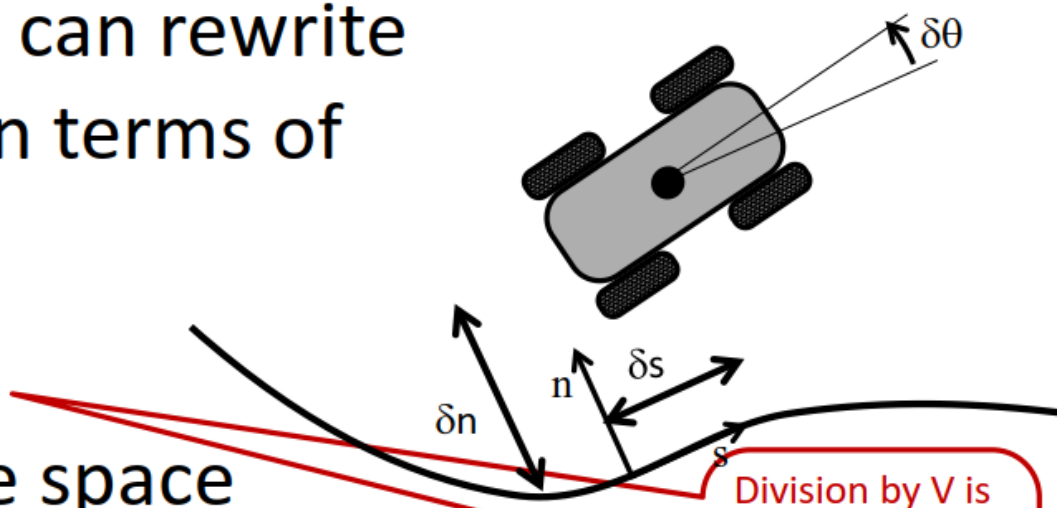
But notice speed V can be factored out.

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ \psi \end{bmatrix} = \begin{bmatrix} \cos\psi \\ \sin\psi \\ \kappa \end{bmatrix} v \qquad \begin{bmatrix} x(t) \\ y(t) \\ \psi(t) \end{bmatrix} = \begin{bmatrix} x(0) \\ y(0) \\ \psi(0) \end{bmatrix} + \int_0^t \begin{bmatrix} \cos\psi \\ \sin\psi \\ \kappa \end{bmatrix} V dt$$

Representing Trajectories

- This means we can rewrite the dynamics in terms of distance

$$\frac{\mathrm{d}\underline{x}}{\mathrm{d}t} / V = \frac{\mathrm{d}\underline{x}}{\mathrm{d}t} / \frac{ds}{dt} = \frac{\mathrm{d}\underline{x}}{\mathrm{d}s}$$



- Nonlinear state space model in terms of distance.

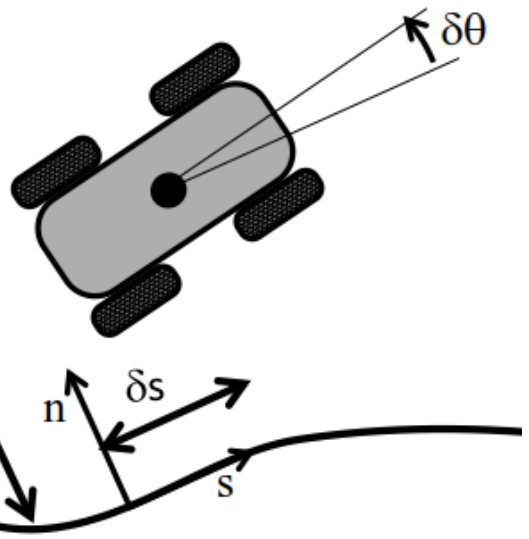
$$\frac{d}{ds}\begin{bmatrix} x \\ y \\ \psi \end{bmatrix} = \begin{bmatrix} cos\,\psi \\ sin\,\psi \\ \kappa \end{bmatrix} \qquad \begin{bmatrix} x(s) \\ y(s) \\ \psi(s) \end{bmatrix} = \begin{bmatrix} x(0) \\ y(0) \\ \psi(0) \end{bmatrix} + \int_0^t \begin{bmatrix} cos\,\psi \\ sin\,\psi \\ \kappa \end{bmatrix} ds$$

Division by V is only a problem if you insist on a curvature for every time.

Robot Trajectory Following

- States: $\underline{x} = \left[x \; y \; \theta\right]^T$
- Inputs: $\underline{u} = \left[\kappa \; V\right]^T$
- Nonlinear state space model:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \kappa \end{bmatrix} V$$



- Suppose a trajectory generator has produced a reference:

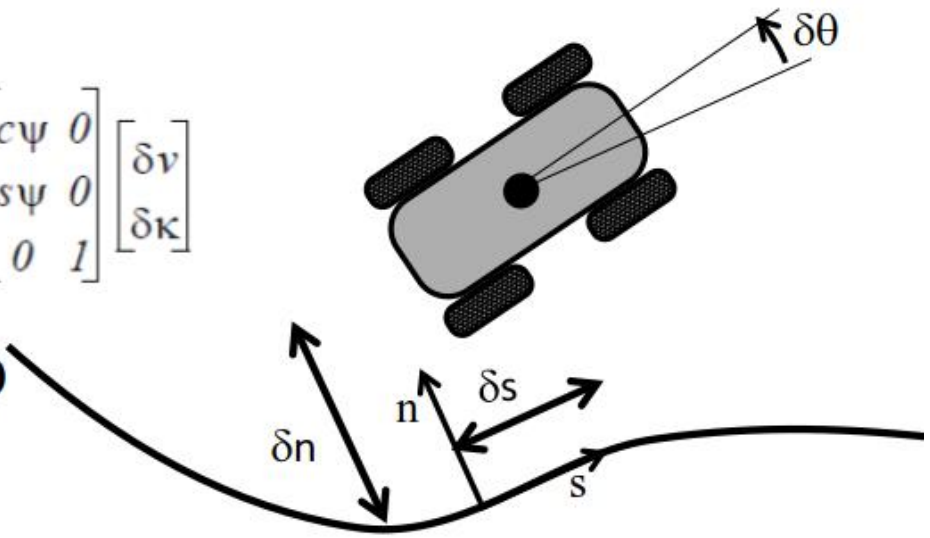
$$[\underline{u}_r(t), \underline{x}_r(t)]$$

- Assume full state feedback.

Linearization

- Linearized Dynamics:

$$\frac{d}{dt}\begin{bmatrix}\delta x\\ \delta y\\ \delta\psi\end{bmatrix} = \begin{bmatrix}0 & 0 & -vs\psi\\ 0 & 0 & vc\psi\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\delta x\\ \delta y\\ \delta\psi\end{bmatrix} + \begin{bmatrix}c\psi & 0\\ s\psi & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\delta v\\ \delta\kappa\end{bmatrix}$$



- Convert coordinates to path tangent frame.

$$\begin{bmatrix}\delta\dot{s}\\ \delta\dot{n}\\ \delta\dot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & V\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\delta s\\ \delta n\\ \delta\theta\end{bmatrix} + \begin{bmatrix}1 & 0\\ 0 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\delta V\\ \delta\kappa\end{bmatrix}$$

Very Simple System

Linear

Time Invariant for constant V

$$\underline{s} = \begin{bmatrix}s & n & \theta\end{bmatrix}^T$$

- Of the form:

$$\delta\dot{\underline{s}}(t) = F(t)\delta\underline{s}(t) + G(t)\delta\underline{u}(t)$$

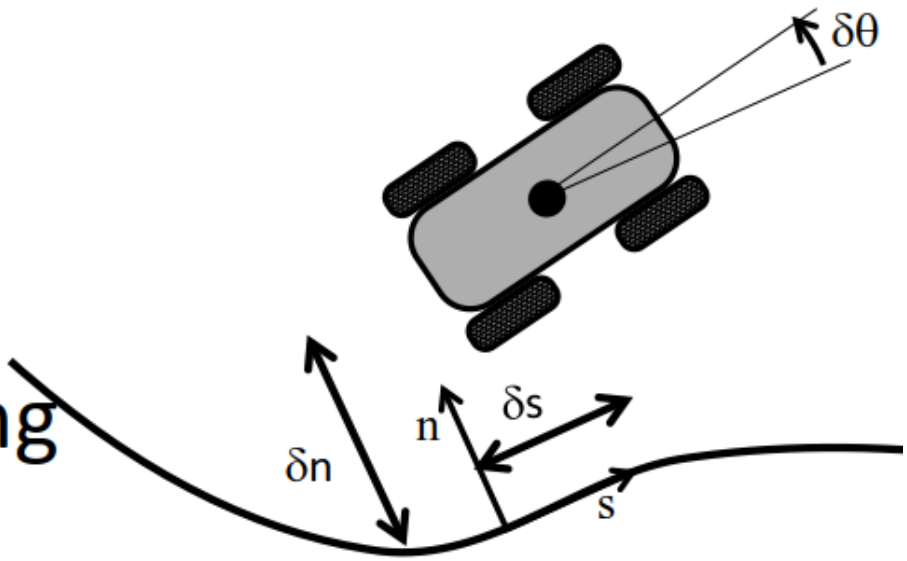
State Feedback Control Law

- State Feedback

$$\delta \underline{u}(t) = -K\delta \underline{s}(t)$$

- We know speed can control s and steering can control n and θ so:

$$K = \begin{bmatrix} k_s & 0 & 0 \\ 0 & k_n & k_\theta \end{bmatrix}$$



State Feedback Control Law (Total Control Law)

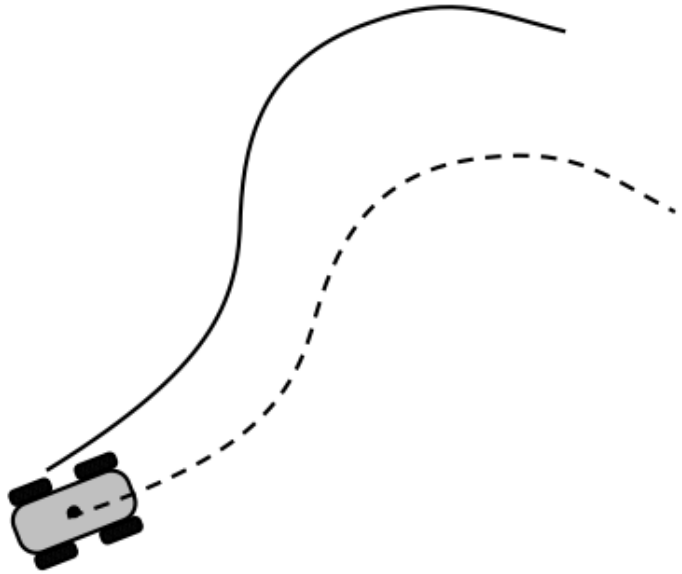
$$\underline{u}(t) = \underline{u}_r(t) + \delta\underline{u}(t) = \underline{u}_r(t) - \begin{bmatrix} k_s & 0 & 0 \\ 0 & k_n & k_\psi \end{bmatrix} \begin{bmatrix} \delta s \\ \delta n \\ \delta \psi \end{bmatrix}$$

Feedforward (path curvature and speed)

Feedback P and PD controllers

Behavior

- **Open Loop Servo Execution**



$$\kappa_d(s) = \kappa(s_{measured})$$

- While simple in principle, open loop execution does not reject disturbances.
- Even a single initial error can grow forever if not compensated.

Behavior

- **Distance Based Open Loop Control**
- Ignore speed by converting to distance as the independent variable.
- Recall, this implies a particular path through space because:

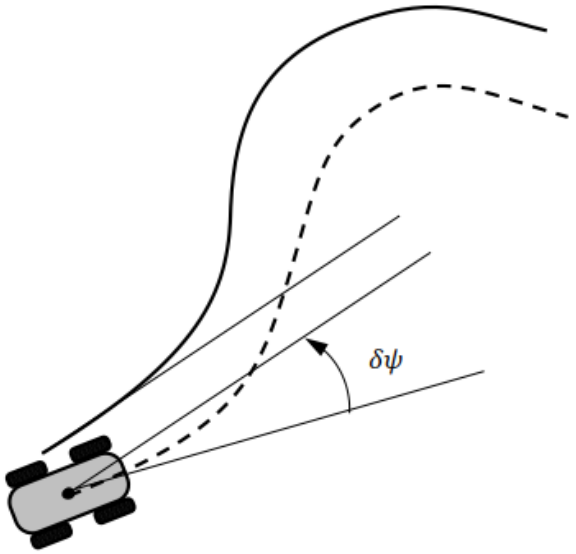
$$\psi(s) = \psi_0 + \int_0^s \kappa \, ds$$

$$x(s) = \int_0^s \cos[\theta(s)] ds$$

$$y(s) = \int_0^s \sin[\theta(s)] ds$$

Behavior

- **Heading Error Compensation**



$$\Delta\kappa = \Delta\psi / L$$

$$\kappa_d(s) = \kappa(s_{measured}) + \Delta\kappa$$

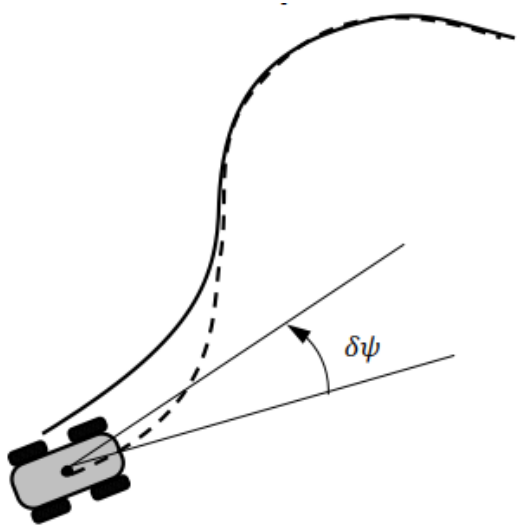
Pass original command to output

Plus a correction

- Passes original command directly to output.
- Bends the response path to be parallel to the desired.
- BUT: Does not move paths together.

Behavior

- **Full Pose Error Compensation, crosstrack error term**



$$\Delta\kappa = \Delta\psi/\mathrm{L}$$

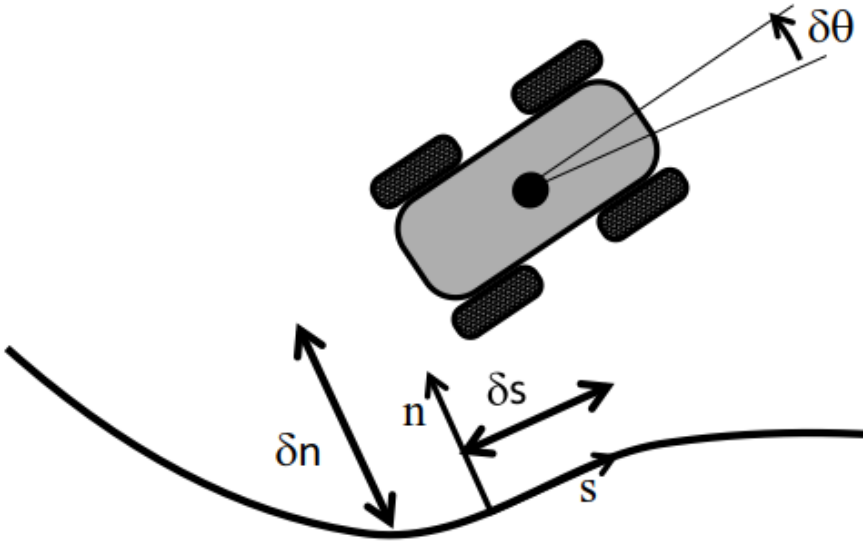
$$\Delta\kappa = 2\delta n/\mathrm{L}^2$$

- Passes original command directly to output.
- δn is coordinate of closest point in body coords.
- Also adds two corrective amounts intended to remove present error.

State feedback, Eigenvalue Placement

- New closed loop dynamics matrix:

$$F - GK = -\begin{bmatrix} k_s & 0 & 0 \\ 0 & 0 & -v \\ 0 & k_n & k_\psi \end{bmatrix}$$



- Characteristic poly:

$$det(\lambda I - F + GK) = \begin{vmatrix} \lambda + k_s & 0 & 0 \\ 0 & \lambda & -v \\ 0 & k_n & \lambda + k_\psi \end{vmatrix}$$

$$(\lambda^3 + \lambda^2(k_\psi + k_s) + \lambda(k_n v + k_s k_\psi) + k_s k_n v)$$

Any coefficients are possible so…..
Any roots are possible.

Gains explained

- Both curvature gains can be related to a characteristic length.

$$k_n = \frac{2}{L^2} \qquad k_\psi = \frac{1}{L}$$

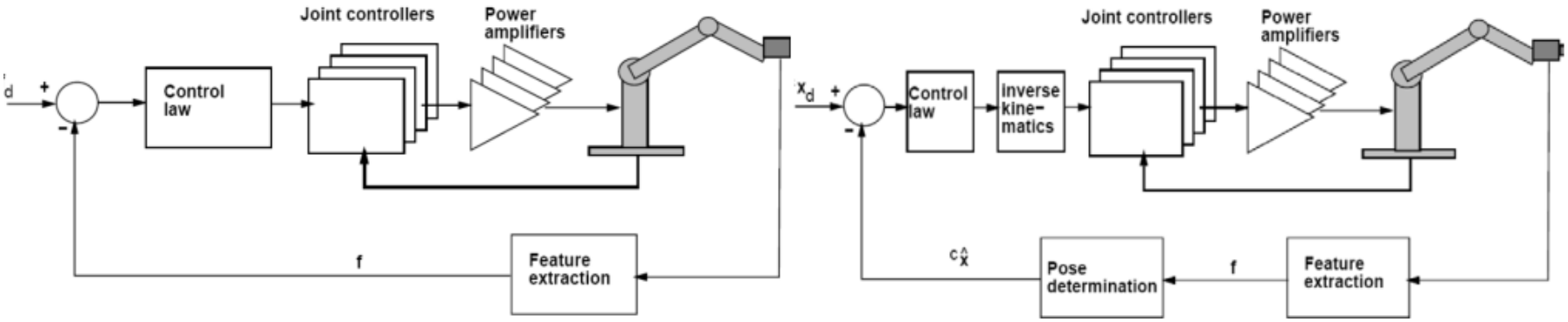
- Then $\delta\kappa_\psi = \delta\psi / L$ removes heading error after moving a distance L.
- And $\delta\kappa_n = 2\delta n / L^2$ removes crosstrack error after travelling a distance L.
- Also $\tau_s = 1/k_s$ is the time constant of speed error response.

Perception Based Control, Visual Servoing

- Observed feature residuals can be generated by:
  - Perceived errors in pose estimates in a region of overlap (registration) or ...
  - Real errors in pose itself in a positioning task.
- In the latter case, it is natural to close a servo loop and drive the system to move to reduce the error. This is visual servoing.
- Must maintain feature correspondences during motion:
  - Embedded feature tracking problem.

Perception Based Control, Visual Servoing Architecture : Errors

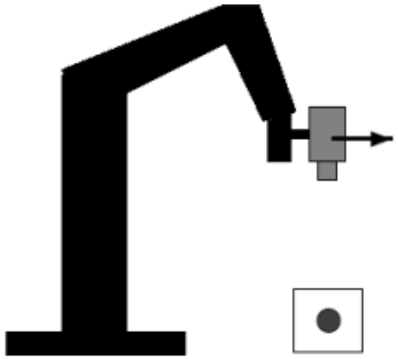
- Image-based control forms errors in image space.
  - Servoing done in image space
- Position-based control forms errors from object poses:
  - Poses derived from image features
  - Servoing done in pose space.



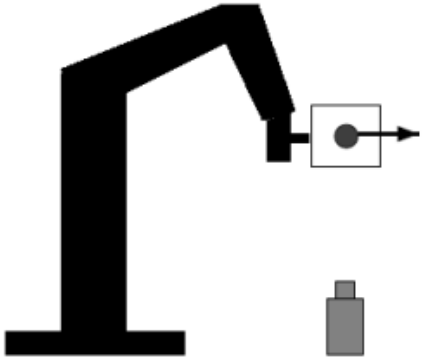
Perception Based Control, Visual Servoing Architecture : Camera Position

- Camera may be moving or stationary.
- Required motions are reversed with respect to each other.

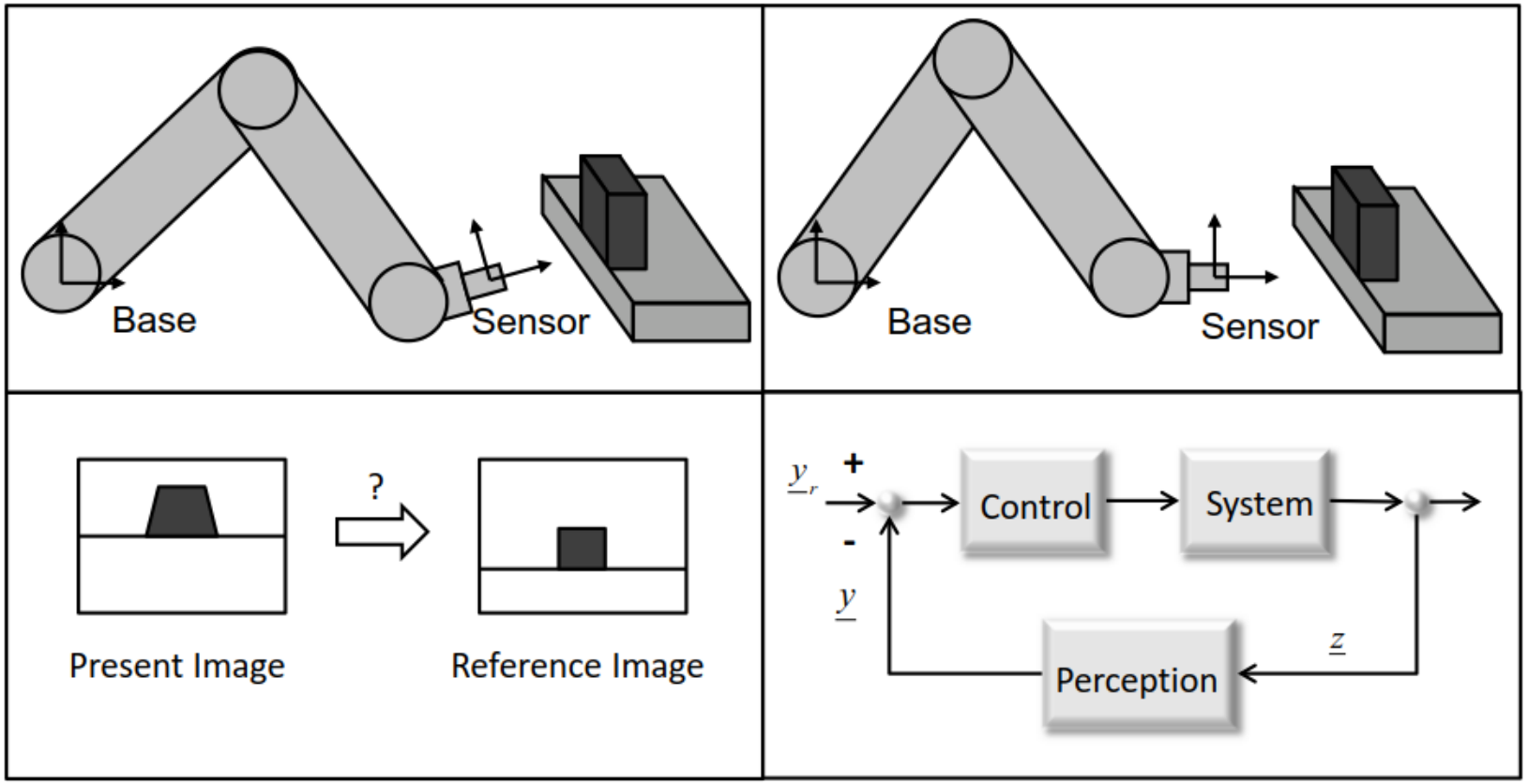
Eye-in-Hand system



Eye-to-Hand system

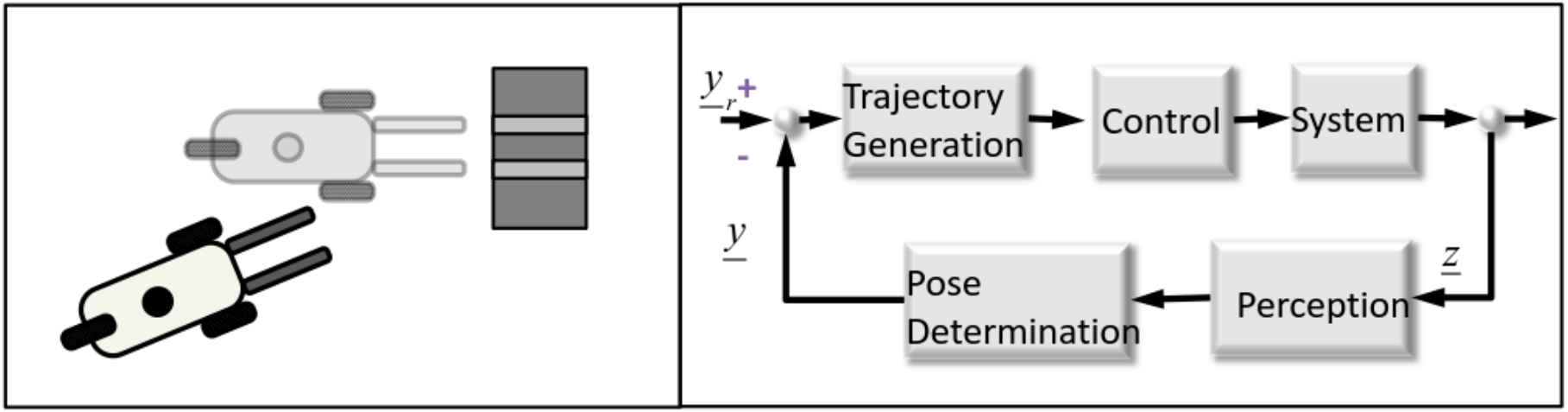


Error Coordinates : Image Based Visual Servoing



- Problem: drive the system (usually with a camera attached) to turn the present image into the desired image.
- An excellent way to drive up to something with a poor pose estimate.

Error Coordinates : Image Based Visual Servoing



- Explicitly calculate the pose of the object relative to the camera.
- Compute the error in the pose.

Steering Trajectory Generation

- Trajectory generation is necessary for any kind of precision control of mobile robots.
- The problem occurs in various forms:
  - "Steering" (curvature generation) problem.
  - "Smooth stopping" (velocity profile) problem.
  - Both at once
    - Sometimes in terms of linear and angular velocity.

Steering Trajectory Generation, Definitions

- Let a trajectory be a specification of an entire motion.
  - Could be explicitly in terms of state:

$$\{\underline{x}(t) | (t_0 < t < t_f)\}$$

  - Could be implicitly in terms of inputs:

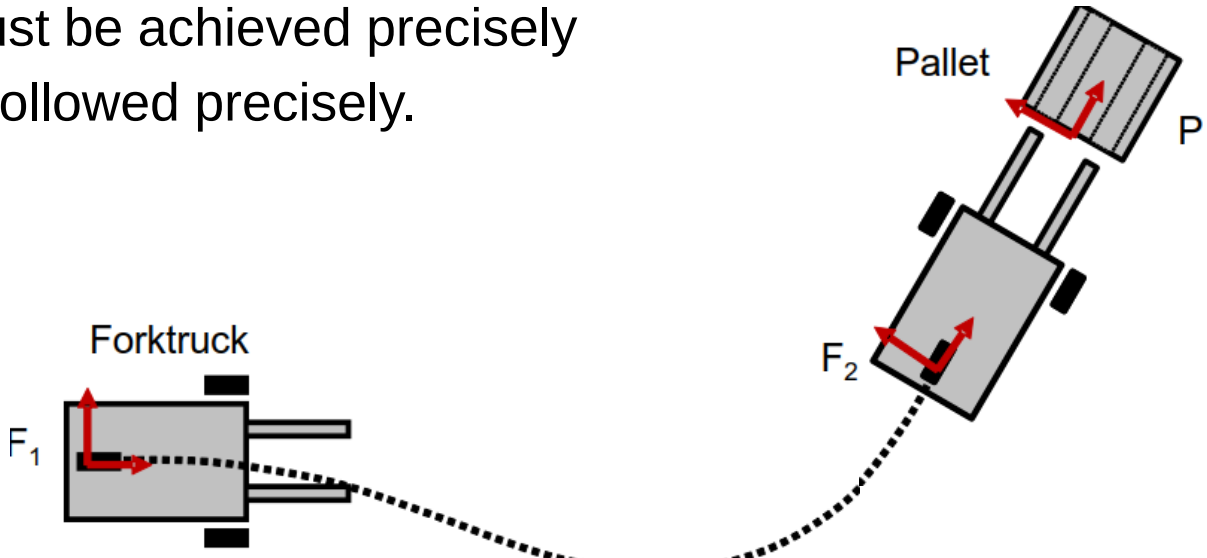
$$\{\underline{u}(t) | (t_0 < t < t_f)\}$$

  - Need both for 2 dof control
- Both can be visualized as the trajectory followed by the tip of a vector over time.

Steering Trajectory Generation, Motivation

- Load cannot be approached sideways.
- Visualize driving backward from goal.
- Maneuver must initially turn away from the pallet.
- Precision control is necessary:
  - when goals states must be achieved precisely
  - when paths must be followed precisely.

The robot must not only follow the intended curves but it must come to a stop neither too early (which would make achieving the next goal impossible) nor too late (which will cause a collision).



Steering Trajectory Generation, Problem Specification

- Dynamics: $\dot{\underline{x}} = f(\underline{x}, \underline{u})$
- Physical constraints: $|\underline{u}(t)| \leq \underline{u}_{max}(t) \quad |\dot{\underline{u}}(t)| \leq \dot{\underline{u}}_{max}(t)$
  - Turn radius bounded from below
  - Curvature is bounded by mechanisms and terrain friction.
- Boundary conditions:

	Start State	Goal State
$\underline{x}(t_0) = \underline{x}_0$	$(x,y,\theta,\kappa,V)_0$	$(x,y,\theta,\kappa,V)_f$
$\underline{x}(t_f) = \underline{x}_f$		

- Problem: determine an entire control function u(t) which generates some desired state trajectory x(t).
- It's the problem of inverting a differential equation.

Steering Trajectory Generation, Formulation as a Root finding Problem

- Every u(t) generates some x(t)...

$$x(t) = x(0) + \int_0^t f(\underline{x}, \underline{u}, t)\,dt$$

- However, many arbitrary x(t)'s represent infeasible motions.
  - Mathematical reasons – underactuation
  - Physics reasons - friction
  - Power related reasons - horsepower

Steering Trajectory Generation, Formulation as a Root finding Problem, Parameterization

- Function space of all $\underline{u}(t)$ is too large to search.
- Parameterize inputs: $\underline{u}(t) \rightarrow \underline{\tilde{u}}(\underline{p}, t)$
- Easy to see by Taylor series that $\underline{p}$ spans all possible $\underline{u}(t)$
  - Pick any $\underline{u}_k(t)$ you like.
  - Write its Taylor series
  - Coefficients $\underline{p}_k$ approximate $\underline{u}_k(t)$ arbitrarily well.
  - But $\underline{u}_k(t)$ was arbitrary too → so $\underline{p}_k$ spans everything!

Steering Trajectory Generation, Formulation as a Root finding Problem, Parameterization

- Now p determines u(t) which determines x(t), so dynamics become:

$$\dot{\underline{x}}(t) = \underline{f}[\underline{x}(\underline{p}, t), \underline{u}(\underline{p}, t), t] = \tilde{\underline{f}}(\underline{p}, t)$$

- The boundary conditions become:

Integrals are suppressed notationally – but they are still there.

$$\underline{g}(\underline{p}, t_0, t_f) = \underline{x}(t_0) + \int_{t_0}^{t_f} \tilde{\underline{f}}(\underline{p}, t)dt = \underline{x}_b$$

- This is conventionally written as:

$$\underline{c}(\underline{p}, t_0, t_f) = \underline{h}(\underline{p}, t_0, t_f) - \underline{x}_b = 0$$

Steering Trajectory Generation, Formulation as a Root finding Problem, Parameterization

$$\underline{c}(\underline{p}, t_0, t_f) = 0$$

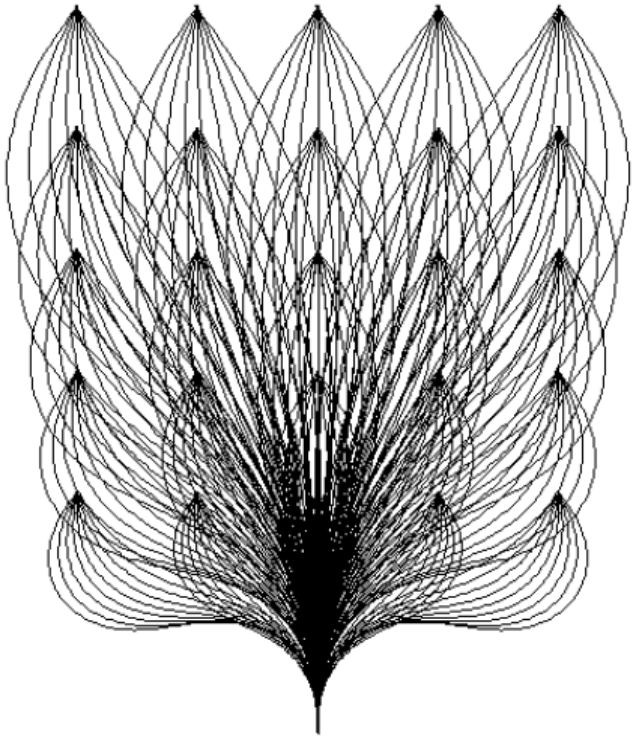
- That is a rootfinding problem!
- Conclusion:
  - the problem of inverting a nonlinear vector differential equation
  - can be converted to a rootfinding problem
  - using parameterization.

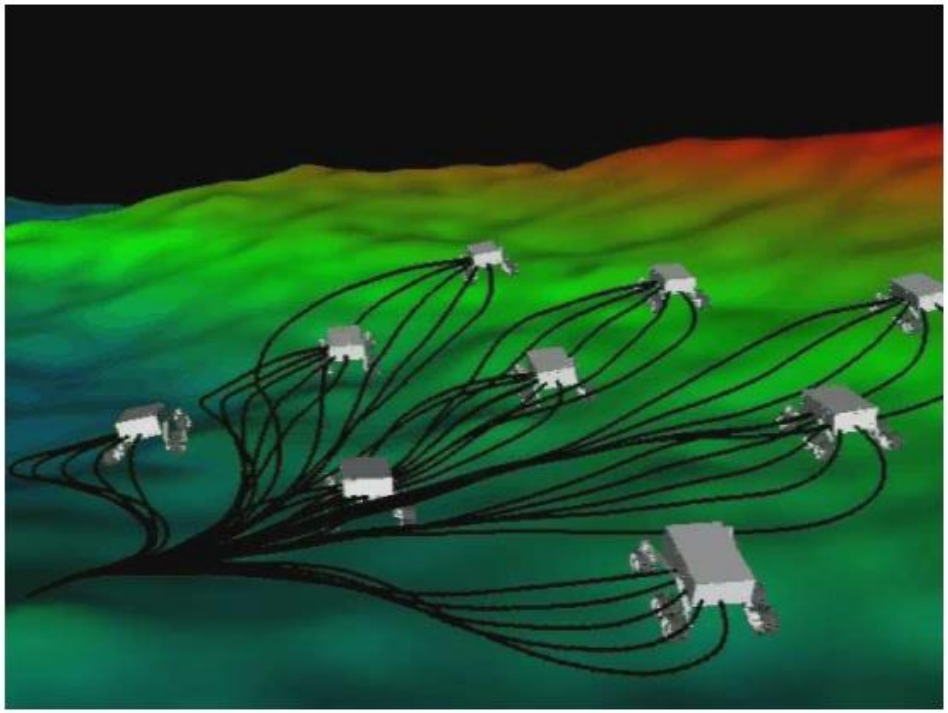
The detailed analysis is out of the scope of this course

Clothoids, linear curvature polynomial, and Polynomial Spirals can be used as primitives

Steering Trajectory Generation, Polynomial Spirals as example

- These curves can achieve any terminal posture.





Summary

- State Space control is powerful
- 2 dof control is a good way to follow trajectories.
- Parameterization is a good way to generate them for open loop control.
- Linearization is effective for nonlinear control, but….
- Visual servoing implements a closed loop using vision as the feedback sensor.
- A basic version tries to drive an image into coincidence with some reference image by:
  - forming errors in image space.
  - deriving corrective velocity commands from the errors.

Optimal and Model Predictive Control

- Prediction enables search
  - creates the capacity to elaborate alternatives.
- Optimality
  - creates the capacity to decide what to do.
- Mobile robots are intelligent (=perceptive and deliberative):
  - perceive the environment around them. — Perceptive
  - predict environmental interactions for candidate motions. — Deliberative
  - rank alternative actions. — Deliberative
  - execute a chosen action. — Deliberative
- The intelligent control of mobile robots is an optimal control problem

Receding Horizon MPC

- Perceptive horizon is intrinsically limited
- So, new information arrives all the time
- Have to keep changing the plan.
- Need models to do adequate prediction for planning.
- The intelligent control of mobile robots is a receding horizon MPC problem

In this course we jump over the details of the of optimal control and NMPC and their numerical implementation

Optimal Control

$$\text{minimize} \qquad J[\underline{x}, \underline{u}, t_f] = \phi(\underline{x}(t_f)) + \int_{t_0}^{t_f} L(\underline{x}, \underline{u})\, dt \qquad t_f \quad \text{free}$$

$$\text{subject to:} \quad \dot{\underline{x}} = f(\underline{x}, \underline{u}) \quad ; \quad \underline{u} \in U$$

$$\underline{x}(t_0) = \underline{x}_0 \quad ; \quad \underline{x}(t_f) = \underline{x}_f \ (\text{when } \phi(\underline{x}(t_f)) \text{ is absent})$$

Boltza Form

Problem has two main components:

– UTILITY: doing something useful (probably to get somewhere, maybe in some best fashion).

– CONSTRAINT: while respecting some constraints.

Optimal Control, Utility

In Bolza form, want to optimize some functional representing "cost" or "utility":

$$J = \phi[x(t_f)] + \int_{t_0}^{t_f} L(\underline{x}, \underline{u}, t)dt$$

Where:

- $\phi[x(t_f)]$ (endpoint cost function) may be used to represent the desire to reach some particular terminal state.
- the integral term can be used to, for example, express the cost of driving at high curvature.

Optimal Control, Utility

In Bolza form, want to optimize some functional representing “cost” or “utility”:

$$J = \phi[x(t_f)] + \int_{t_0}^{t_f} L(\underline{x}, \underline{u}, t)\,dt$$

Where:

- $\phi[x(t_f)]$ (endpoint cost function) may be used to represent the desire to reach some particular terminal state.
- the integral term can be used to, for example, express the cost of driving at high curvature.

Solution

- “The” Minimum Principle Or “the” Maximum Principle by Pontryagin
- Dynamic Programming, Bellman equations

Model Predictive Control , Receding Horizon Control

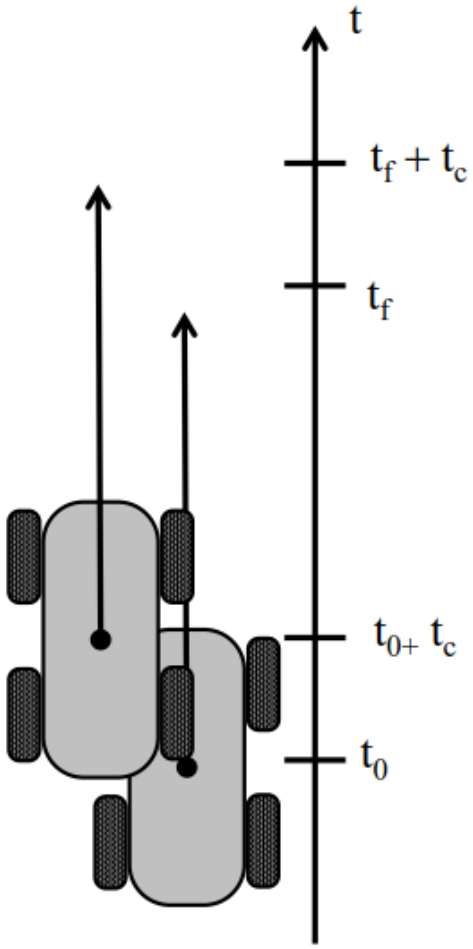
Solve the following problem for some finite prediction horizon

t_f :

$$J = \phi[x(t_f)] + \int_{t_0}^{t_f} L(\underline{x}, \underline{u}, t)\,dt$$

Execute the optimal control u*(t) for a control horizon t_c

Do it all over again for $t_0 + t_c$ and $t_f + t_c$



Model Predictive Control , Solution Methods (only) Direct Methods: Finite Differences

- Discretize the dynamic model:

$$\underline{x}(k+1) = \underline{x}(k) + f(\underline{x}(k), \underline{u}(k))\Delta t$$

- Discretize the objective:

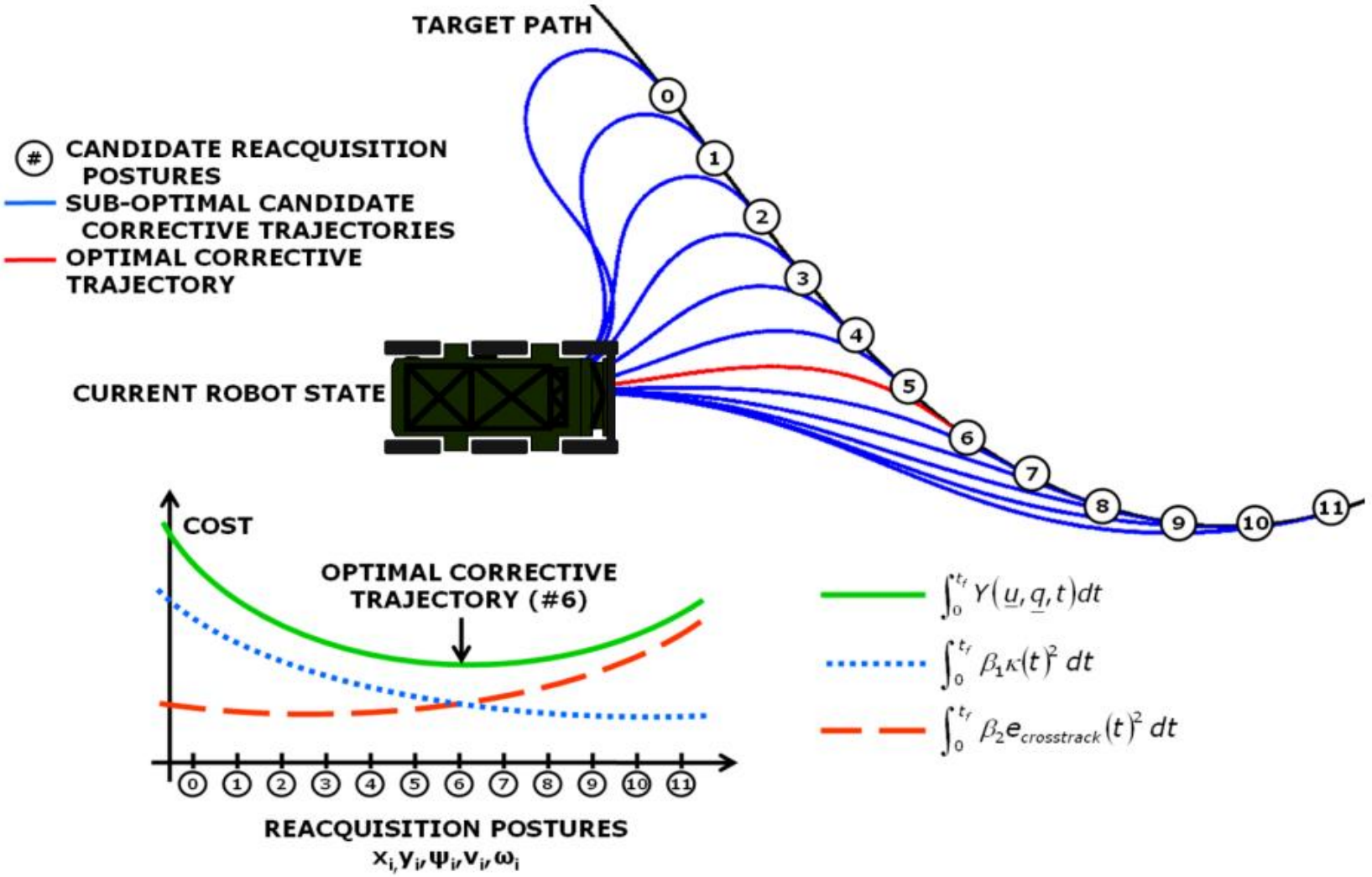
$$J = \phi(\underline{x}(n)) + \sum_{k=0}^{N-1} L(\underline{x}(k), \underline{u}(k), k)\Delta t$$

- This is a constrained optimization problem with linear constraints.
- There are Nm unknowns in $\underline{u}(\,)$ and Nn dof in $\underline{x}(\,)$ so there are N(n-m) dof left for optimization.

Direct Methods: Finite Differences

- Process:
  - Start with a guess of the inputs $\underline{u}(\ )$ for every k.
  - Integrate the system model to determine $\underline{x}(\ )$ for every k.
  - Compute $J(\underline{x},\underline{u})$.
  - Compute its gradient w.r.t. $\underline{u}$.
  - Line search the descent direction.
  - Repeat until convergence.

Model Predictive Control : Adaptive Horizon



Summary optimal and Receding Horizon MPC

- Optimal Control is a generalization of the Calculus of Variations which expresses the mobile robot control problem well.
- Trajectory generation fits very nicely into the standard form of an optimal control problem.
- Curvature polynomials of arbitrary order are a convenient representation of trajectories.

Aalto University
School of Electrical
Engineering

Nonlinear Model Predictive Control (NMPC) in Semiautonomous Systems at Aalto
Example: Autonomous driving of Tractor and Implement

The general structure of the NMPC problem

The goal is to minimize cost function *J(x(t),u(t),t)* that usually has quadratically weighted integral terms on states and controls compared to some reference trajectories over the length of the prediction horizon .

$$\min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} \quad J(\mathbf{x},\mathbf{u})$$

$$\text{s.t.} \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t))$$

$$\mathbf{x}(0) = \mathbf{x}_0$$

$$\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max}, t\epsilon[0,t_f]$$

$$\mathbf{x}_{min} \leq \mathbf{x}(t) \leq \mathbf{x}_{max}, t\epsilon[0,t_f]$$

The discretized NMPC problem

The discretized quadratic objective function (criteria) to be minimized

$$\overline{J}(\mathbf{x},\mathbf{u}) = \Sigma_{k=0}^{N-1}\left(\|\mathbf{x}(t_{k+1}) - \mathbf{x}_{ref}(t_{k+1})\|_{\mathbf{Q}}^2 + \|\mathbf{u}(t_k) - \mathbf{u}_{ref}(t_k)\|_{\mathbf{R}}^2\right) + \|\mathbf{x}(t_N) - \mathbf{x}_{ref}(t_N)\|_{\mathbf{P}}^2$$

$$\begin{aligned}
\min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} \quad & \overline{J}(\mathbf{x},\mathbf{u}) \\
\text{s.t.} \quad & \mathbf{x}(t_{k+1}) = \bar{\mathbf{f}}(\mathbf{x}(t_k),\mathbf{u}(t_k)) \\
& \mathbf{x}(0) = \mathbf{x}_0 \\
& \mathbf{u}_{min} \leq \mathbf{u}(t_k) \leq \mathbf{u}_{max} \\
& \mathbf{x}_{min} \leq \mathbf{x}(t_k) \leq \mathbf{x}_{max}
\end{aligned} \quad ,$$

The criteria is minimized in every control cycle.
In NMPC, only the first control is implemented.

The basic stages in the NMPC, e.g.

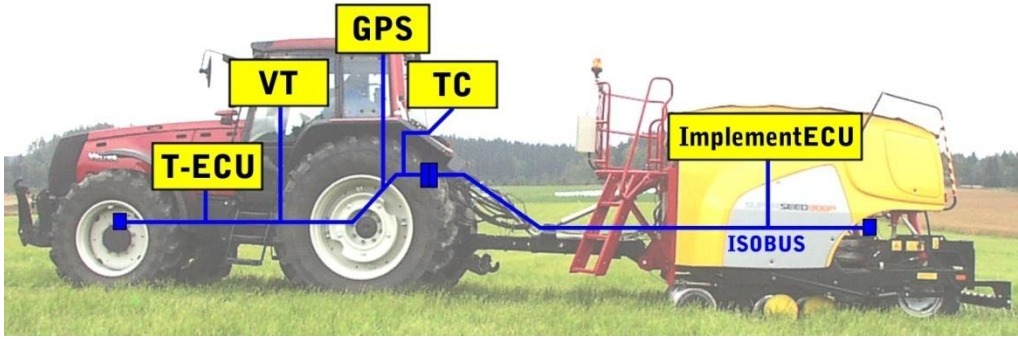
The basic stages in the developed toolkit VIATOC NMPC are:

- Integrating the state trajectory and calculating the sensitivities
- Calculating the direction of the steepest descent, projecting the steepest descent, and taking the appropriate step in that direction.
- These stages are repeated until a predefined number of iterations or a given time limit is reached.
- The goal is not to reach the exact optimum, but a trajectory that is sufficiently close to it.

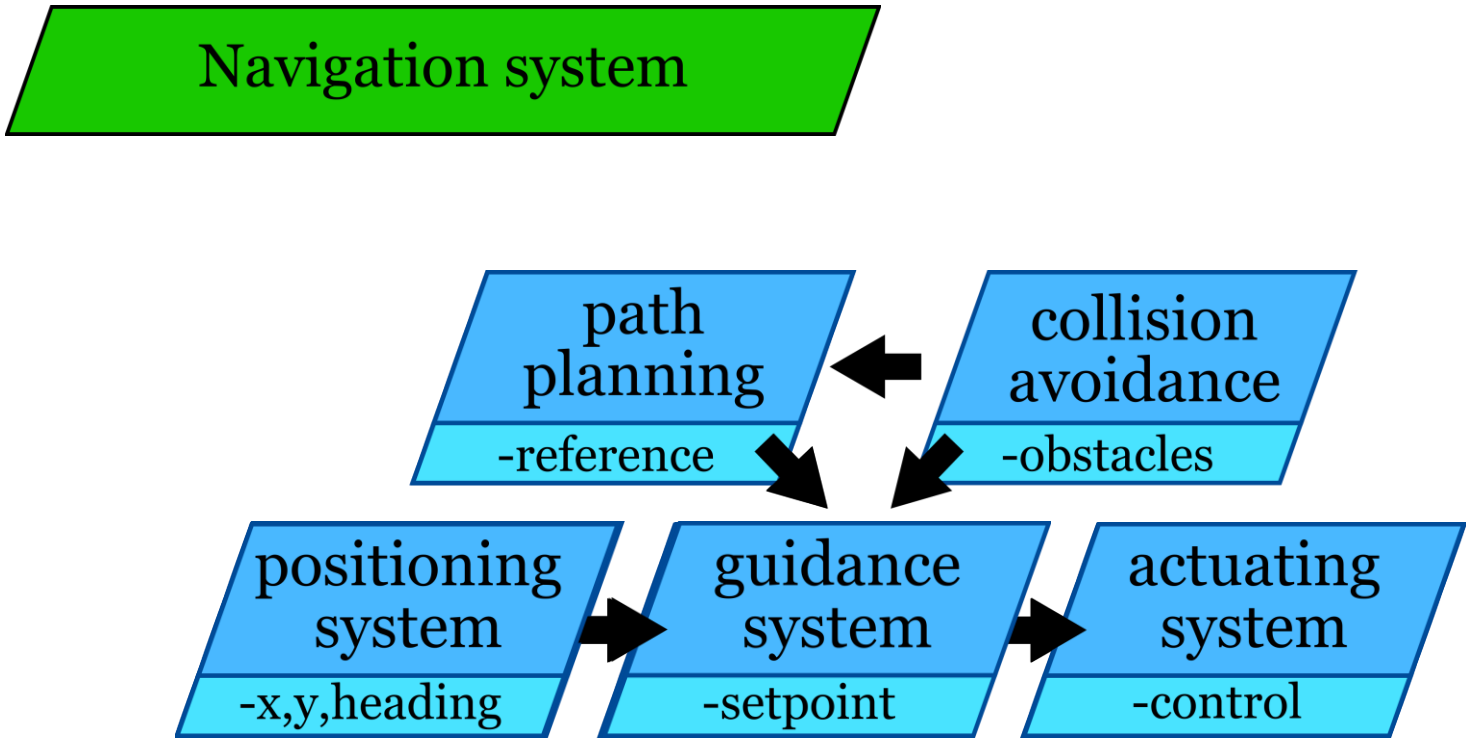
AGROMASSI
Assisting and adaptive agricultural machine
Partners: MTT, HY, Parker, Wapice, Valtra, Suonentieto, companies

- Distributed control system over ISO 11783
  Human friendly, natural HMI; Safety

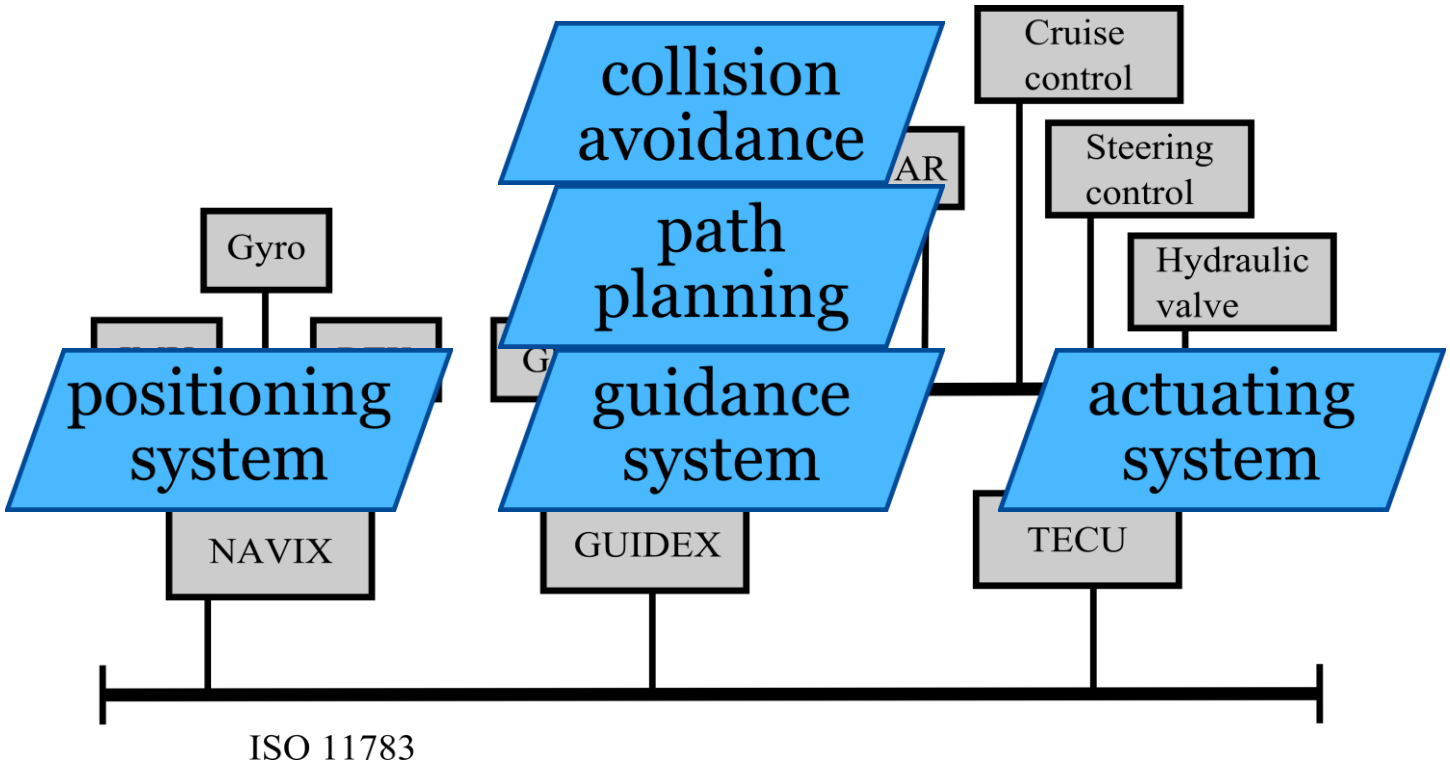
  Toolchain: Matlab/Simulink + C-code generation + PoolEdit + WinCE real-timecomputer
- Co-driver, autopilot
- Cartesian motion control
- Connection assistant
- Contract Team assistant



Agromassi Co-driver, Autopilot
Dissertation of Juha Backman

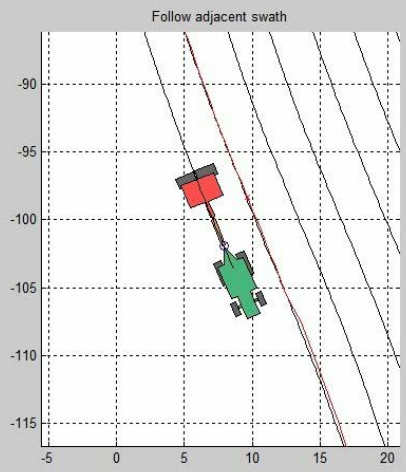


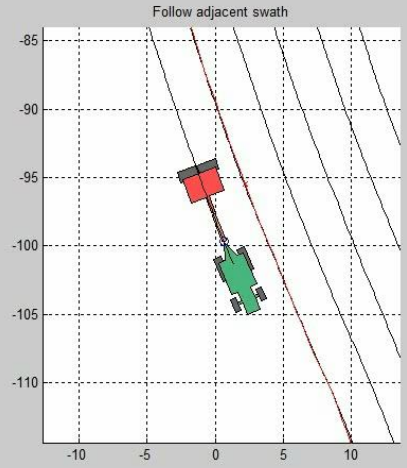
Co-driver, Autopilot



AGROMASSI

Integrated Navigation for Agricultural Machines
GOAL: “co-driver” , Doctor Dissertation of Juha Backman

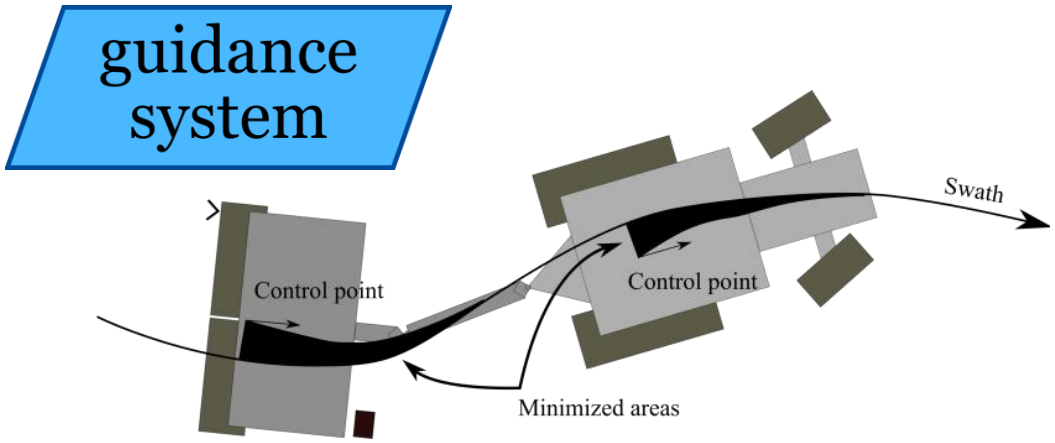


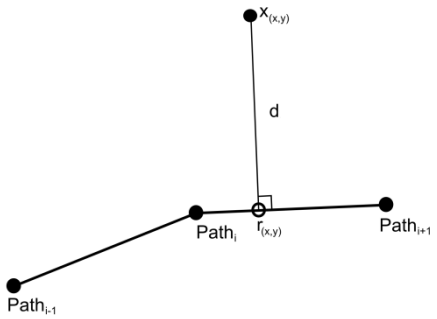


AGROMASSI Co-driver; pole collision avoidance, by Juha Backman

AGROMASSI Co-driver; pole collision avoidance, by Juha Backman

Control algorithms



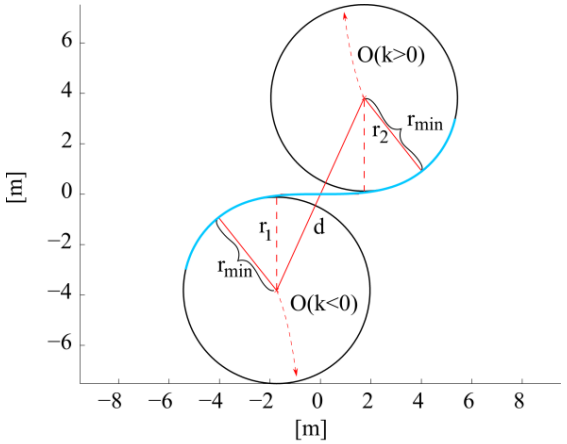


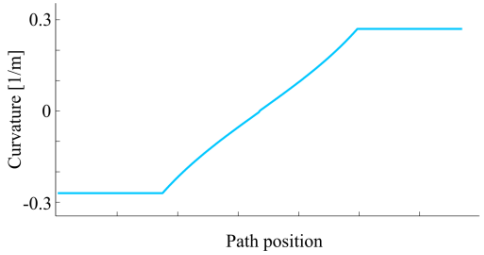
- Nonlinear Model Predictive Path Tracking for the combination of tractor and implement
- The kinematic model for the tractor and implement is used as dynamic model. Still quite complicate model!
- Because the tractor actuators are not infinitely fast, the dynamics of those are modelled, the desired speed and steering angle, with simple 1st order models.
- The dimension of state vector 11, the dimension of input vector 3.

Path planning for turnings in the headland

path planning

- Dubins' Curves consist of six different turning types. These turnings consist of arcs with a maximum curvature and a straight line segment between the two arcs.
- LRL and RLR are basic turnings types
- A simplified path planning algorithm based on Dubins curves.

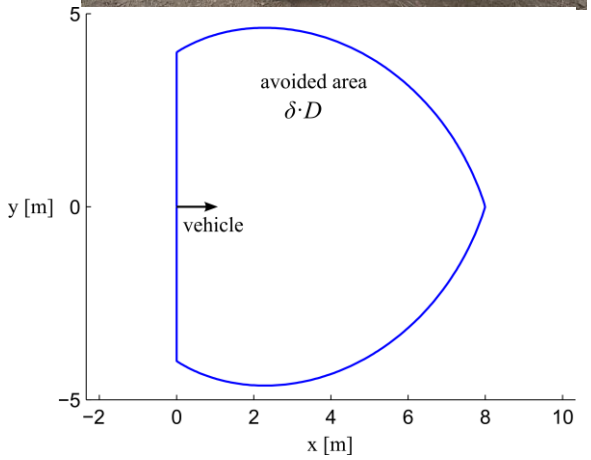




Avoiding of collisions via path planning

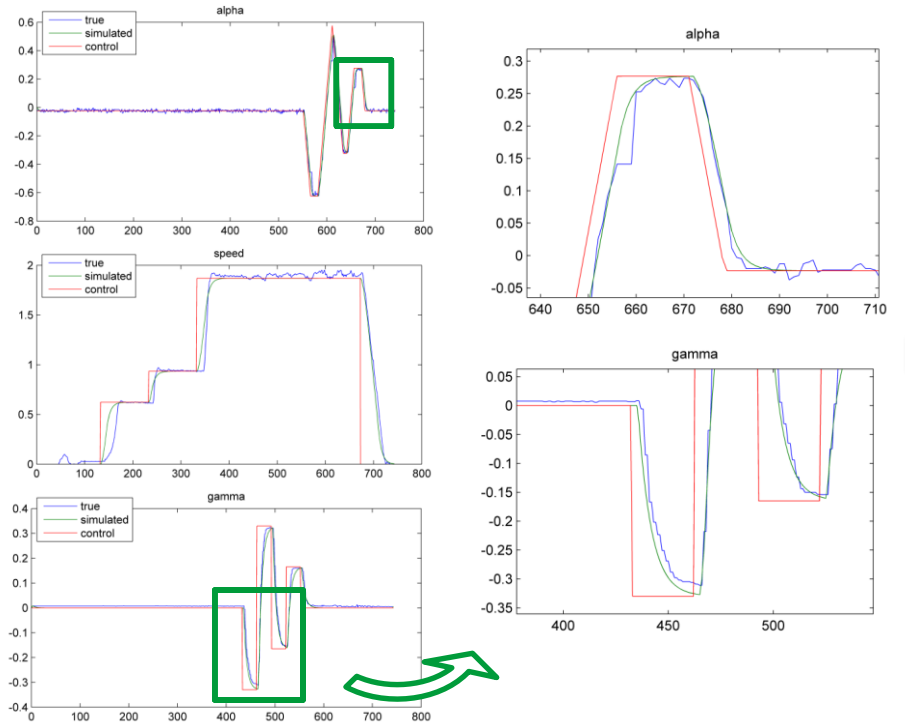
collision avoidance

- There are different ways to include the object avoidance problem within the NMPC
- The option of modifying the cost function was chosen
- A cost that ensures that the vehicle will drive past the obstacle is added
- The area where the original cost is changed into the avoiding cost is illustrated right.



The commissioning

- Experiment setup for estimating parameters



$\max |\dot{\alpha}| = 0.4\text{rad/s}$

$\max |\dot{v}| = 0.5\text{m/s}^2$

$\max |\dot{\gamma}| = 1.9\text{rad/s}$

$k_{\alpha} = 0.66$

$k_{v} = 0.87$

$k_{\gamma} = 0.85$

$\tau(\alpha_{meas}) = 100\text{ms}$

$\tau(v_{meas}) = 500\text{ms}$

$\tau(\lambda_{meas}) = 100\text{ms}$

Results: Complete operation on a real field

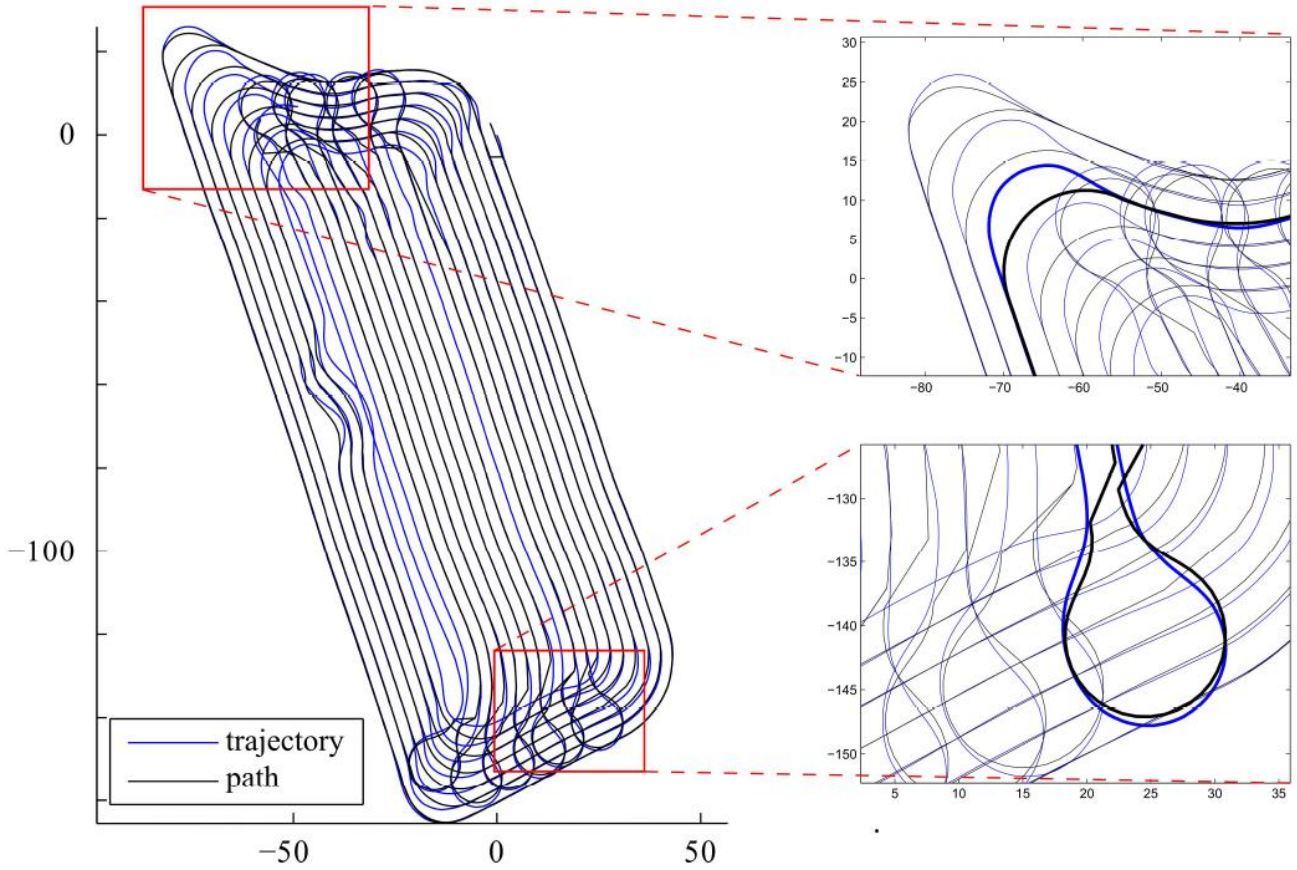
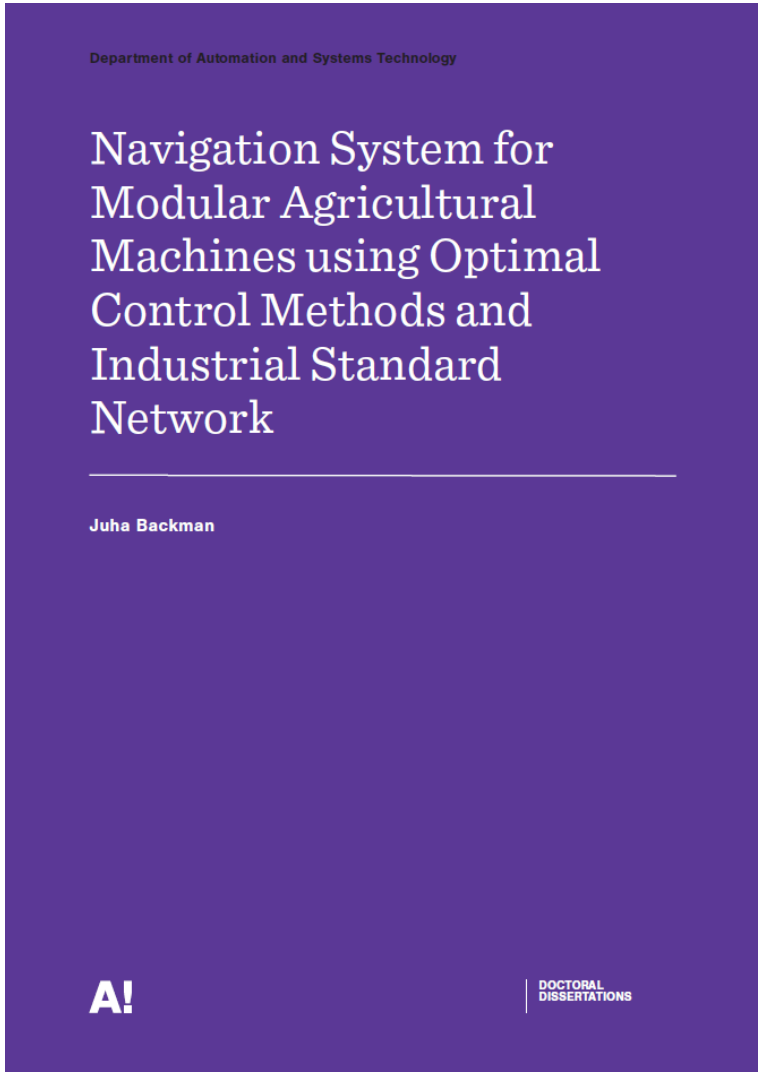


Figure 5.3. Driven trajectories (blue) and generated path (black) on a real field.



http://lib.tkk.fi/Diss/2013/isbn9789526053912/isbn9789526053912.pdf

Aalto-yliopisto

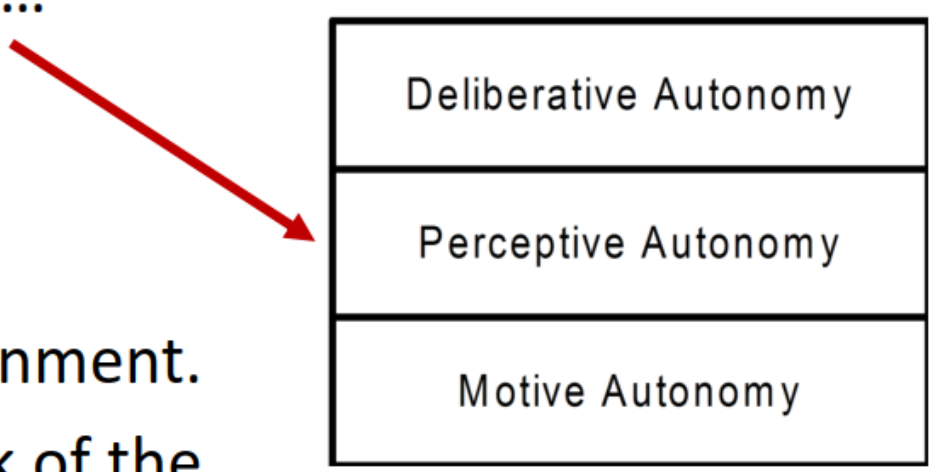
Summary NMPC Control

- Optimal Control is a generalization of the Calculus of Variations which expresses the mobile robot control problem well.
- Trajectory generation fits very nicely into the standard form of an optimal control problem.
- Curvature polynomials of arbitrary order are a convenient representation of trajectories.
- In NMPC, it is easy to take the mechanical constraints of the system into account. Virtual constraints, e.g the angular velocity of the individual joints can be constrained.
- NMPC is computationally expensive
- The redundancy problem of the controlled platform can be solved in elegant way:
  - Combined NMPC motion control of a mobile platform and a crane, reference path for boom tip: 7 DOF to control 3D position has been demonstarted by Autonomous systems group

Intelligent Control

Hierarchy

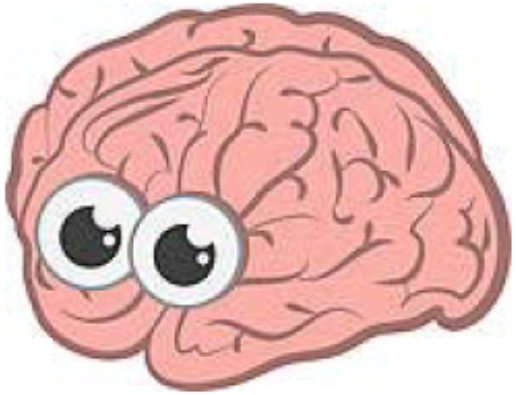
- We are here now …



- Responsible for responding to the immediate environment.
- Requires feedback of the state of the environment (e.g. perception).
- May only need relative position estimates.

Intelligent Predictive Control: Perceptive

- By assumption: Environment is partially unknown and must be measured.
- Don't know beforehand where the obstacles are - or you would have planned around them already.
- "Intelligent" means understanding your surroundings. Hence:
  - IC must be perceptive.
- Perception is discussed later.
  - Here, we will use an environmental model that was produced by perception.



Intelligent Predictive Control: Predictive

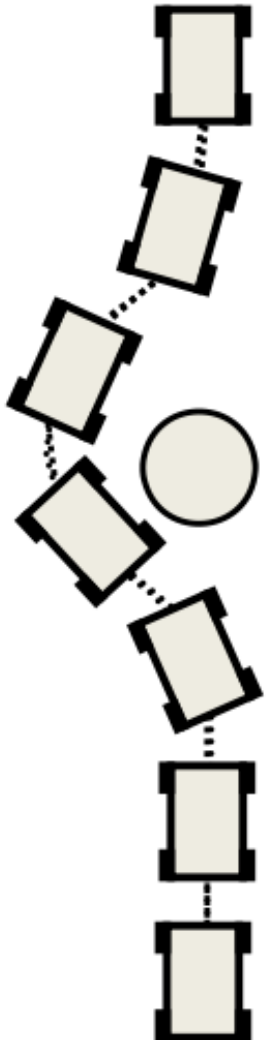
- Latencies and robot dynamics mean it takes time for actions to take effect.
- Robot may also be under-actuated.
- Elements in the scene may be dynamic.
- Hence IC must be predictive.

Intelligent Predictive Control: Reactive

- However, perception must be done continuously because effective sensor range is limited by:
  - Missing parts (occlusion, limited sensor range)
  - Uncertainty
- Also, prediction of dynamic obstacles is only valid for short periods of time.
- Must:
  - perceive continuously
  - react to what you can see.
  - do it all over again high frequency.

Generic Intelligent Control Loop

- 1: Consider “all” options for proceeding through space.
- Check each for problems.
- Eliminate those options which are definitely (or probably) problems.
- If any options remain, pick the best from the perspective of mission execution. Goto 1:
- If none remain, do something which reduces your losses
- If you survive that, ask for help, or execute other recovery mechanisms.



Generic Intelligent Control Loop: Elements of Effective IPC

- A model of your capacity to move
  - Motion prediction
- A model of the state of the environment
  - Representation
- A capacity to evaluate alternatives for
  - Trajectory evaluation
- A capacity to search through the space of possible motions
  - Optimal control

Formulation as Optimal Control: Objectives and Constraints

- Motions can be ranked based on cost/utility and satisfaction of hard constraints:
- Simple case:
  - Score each motion (utility)
  - Do not hit obstacles (constraint)
- However, obstacles can also be encoded as cost of traversal and there are motions which do not satisfy feasibility constraints.

Formulation as Optimal Control: Objectives and Constraints

- Objectives to Minimize
  - Risk level
  - Path following error
  - Path length to goal
  - Integral speed error.

- Constraints
  - Dynamics ("feasible") $\dot{\underline{x}} = f(\underline{x}, \underline{u}, t) \quad ; \quad \underline{u} \in U$
  - Don't hit obstacles ("admissible") $\underline{x}(t) \notin O$
  - Don't tip over ("stability")

Formulation as Optimal Control: Equations

- Over Time (Trajectory)

$$J[\underline{x}, t_f] = \phi(\underline{x}(t_f)) + \int_{t_0}^{t_f} L(\underline{x}, \dot{\underline{x}}, t)dt$$

$$\dot{\underline{x}} = f(\underline{x}, \underline{u}, t) \quad ; \quad \underline{u} \in U$$

$$\underline{x}(t_0) \in S \qquad \underline{x}(t_f) \in G$$

- Over Space (Path)

$$J[\underline{x}, s_f] = \phi(\underline{x}_f) + \int_{s_0}^{s_f} L(\underline{x}, \underline{u}, s)ds$$

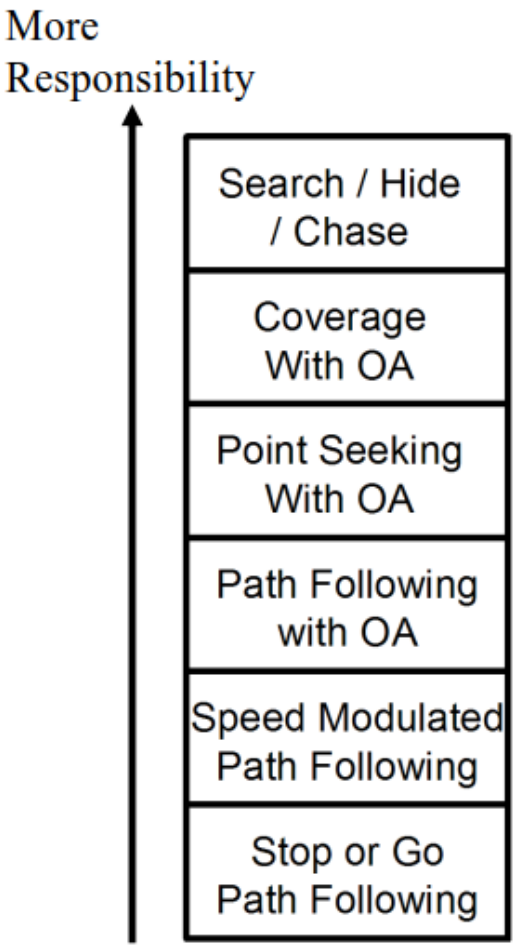
Line Integral

$$\underline{x}(s_0) \in S \qquad \underline{x}(s_f) \in G$$

$$\dot{\underline{x}} = f(\underline{x}, \underline{u}, s) \quad ; \quad \underline{u} \in U$$

Formulation as Optimal Control: Encoding the Mission in the Objective

- The objective may impart differing levels of responsibility to intelligent control.
- 1) Fixed, detailed path - keep going or stop. AGVs do this in factories.
- 2) Fixed path with speed modulation. Following behavior is a special case of this.
- 3) Follow default path with deviation to avoid obstacles permitted.
- 4) Sparse waypoints to hit with complete authority to plan the paths between them.
- 5) Cover an entire area (e.g. mow the grass).
- 6) Search for something, run from something, or pursue something.



Formulation as Optimal Control: Evaluation

- Methods to compute feasible trajectories were covered earlier in motion prediction (dynamics).
- This section is about how to evaluate trajectories.

Formulation as Optimal Control: Representation

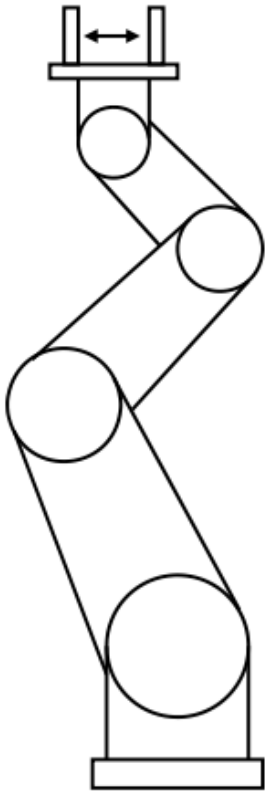
- Methods to compute feasible trajectories were covered earlier in motion prediction (dynamics).
- This section is about how to evaluate trajectories.
- Before we can cost a path, we must cost a point on a path. That means we must model:
  - The path
  - The vehicle
  - The environment

Formulation as Optimal Control: Representation

- Methods to compute feasible trajectories were covered earlier in motion prediction (dynamics).
- This section is about how to evaluate trajectories.
- Before we can cost a path, we must cost a point on a path. That means we must model:
  - The path
  - The vehicle
  - The environment

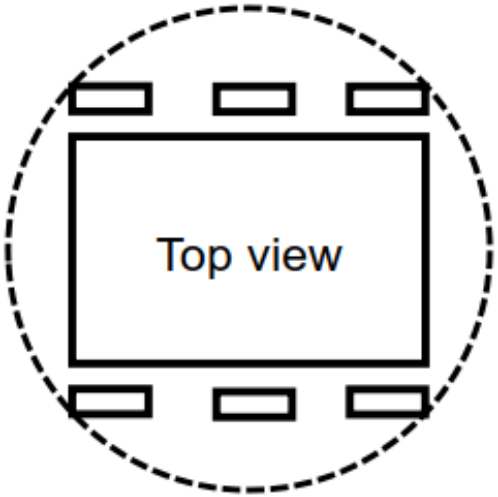
Formulation as Optimal Control: Representing Configurations, C Space Definitions

- A configuration of an object is a specification of the position of every point on the object (with respect to a fixed frame of reference).
- A Configuration Space is the space (i.e. set) of all configurations of the object.
- Informally, this is a set of generalized coordinates which completely determine the position of every point on the object.



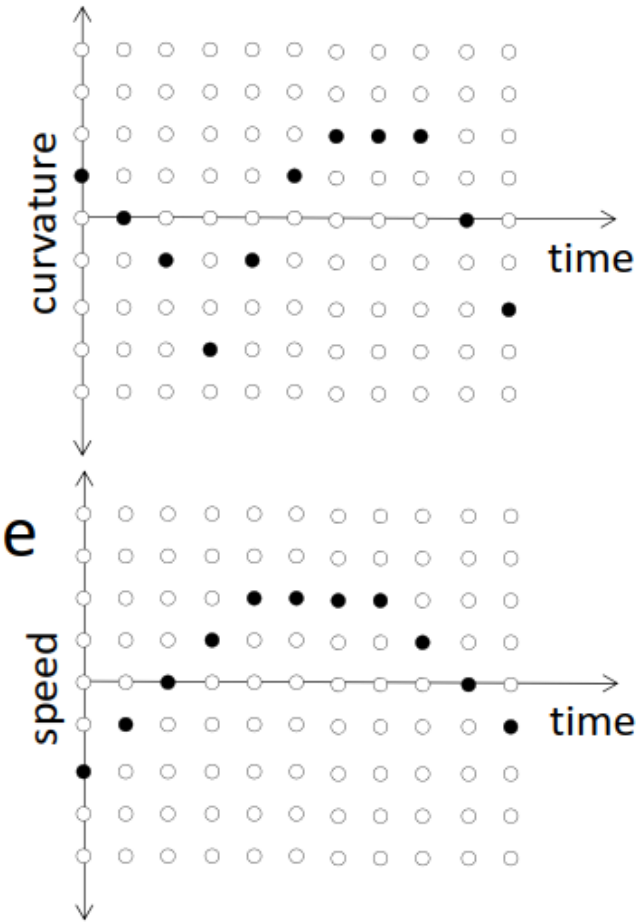
Formulation as Optimal Control: Computation

- Computational complexity of search is directly related to:
  - the dimension of C Space
  - the complexity of the obstacles
- At times, it is valuable to approximate a robot shape by a symmetric one (say, by a circle) in order to reduce the dimension of C space.



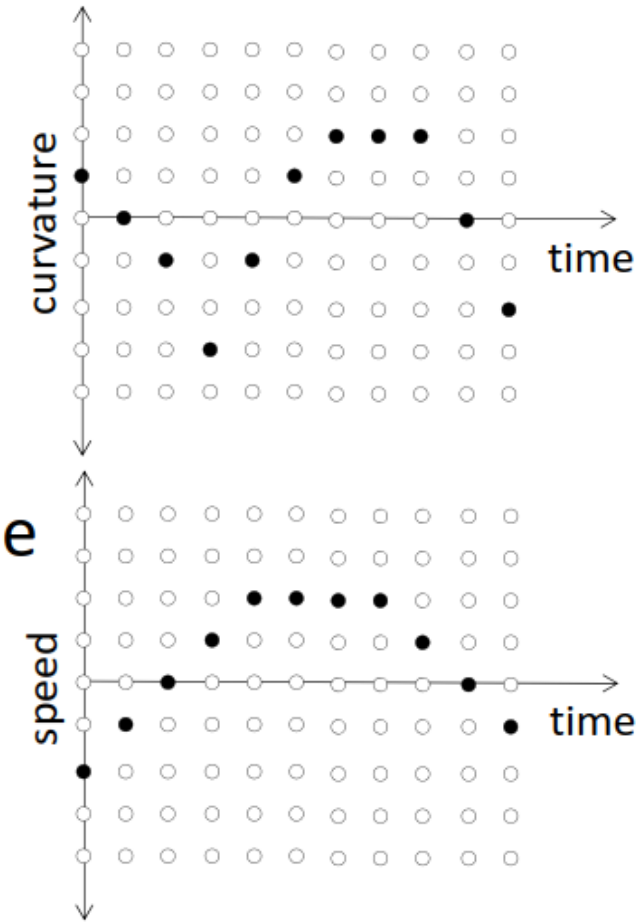
Formulation as Optimal Control: Search Sampling, Discretization, and Relaxation

- In full generality, there is a function space u(t) to search.
- Discretization and parameterization are two options.
- For 10 signal levels and 40 time samples, there are 10^{40} alternatives.
  - Not feasible to search at 10 Hz.



Formulation as Optimal Control: Search Sampling, Discretization, and Relaxation

- In full generality, there is a function space u(t) to search.
- Discretization and parameterization are two options.
- For 10 signal levels and 40 time samples, there are 10^{40} alternatives.
  - Not feasible to search at 10 Hz.



Formulation as Optimal Control: Search

- Methods for different kind of search problems are very essential intelligent control of mobile robotics
- Some standard approaches in path planning are represented in the next lecture tomorrow

Formulation as Optimal Control: Summary

Avoiding obstacles is a kind of planning problem.

- Motion prediction.
- Trajectory Evaluation
- Search

Its a real-time problem.

- If the choice is between smart and fast, semi-dumb robots rule here.

Dynamics matter in many ways.

Cleverness of several kinds are possible.

Alternative courses of action are evaluated based on models of environmental interaction.

A constrained optimization formulation applies.

- Obstacles and dynamics are constraints
- Feasible paths evaluated for utility.

A large number of options exist for the representations used in planning models.

- Each has its own issues and advantages from the perspective of computational complexity.