Navigation of autonomous vehicles

Kristijan Maček

Autonomous Systems Lab, EPF Lausanne

Department of Control and Computer Engineering in Automation, FER Zagreb

Contents

- Introduction
- Lane detection vision module
- Off- and on-line path planning
- Obstacle avoidance
- Path following
- Conclusions

Introduction - Navigation of autonomous vehicles

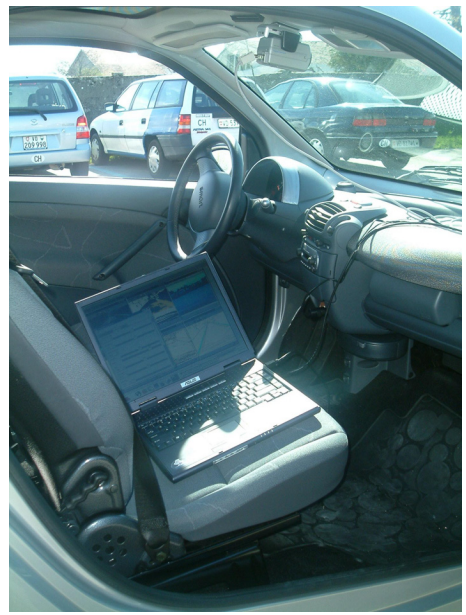
- Perception and sensing: extracting relevant information from the environment
- Map building and localization: environment information segmentation and determining of ego-position
- Motion planning: determining a feasible path from a start to a goal configuration
- Motion control: determining a sequence of motion commands for a given path

Lane detection vision module

- "SPARC" - "Secure Propulsion Using Advanced Redundant Control"
- Final objective: a driver support system that would assist (warning mode) and control (correction mode) the driver in a vehicle
- Primary target: determine position of the ego vehicle on the road
- Basic approaches:
  - Specifically developed roads with beacons
  - Vision system in the vehicle performing road boundary detection

Vehicle setup

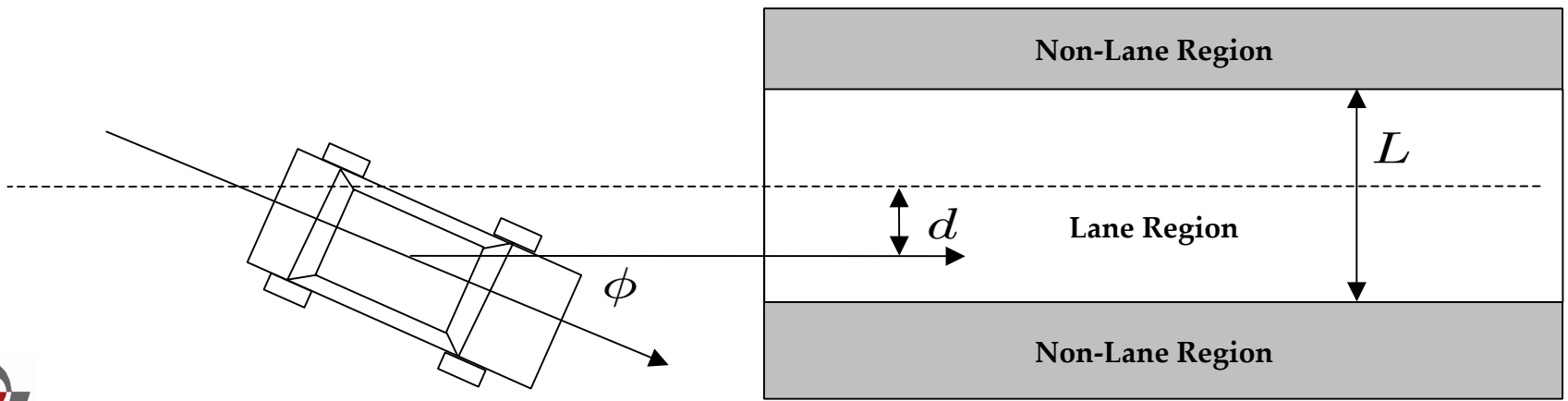
- Single digital camera positioned centrally
- Laser sensor for obstacle detection
- Firewire connected standard laptop with GenoM (*[Fleury et al., 1997]*)



System state description

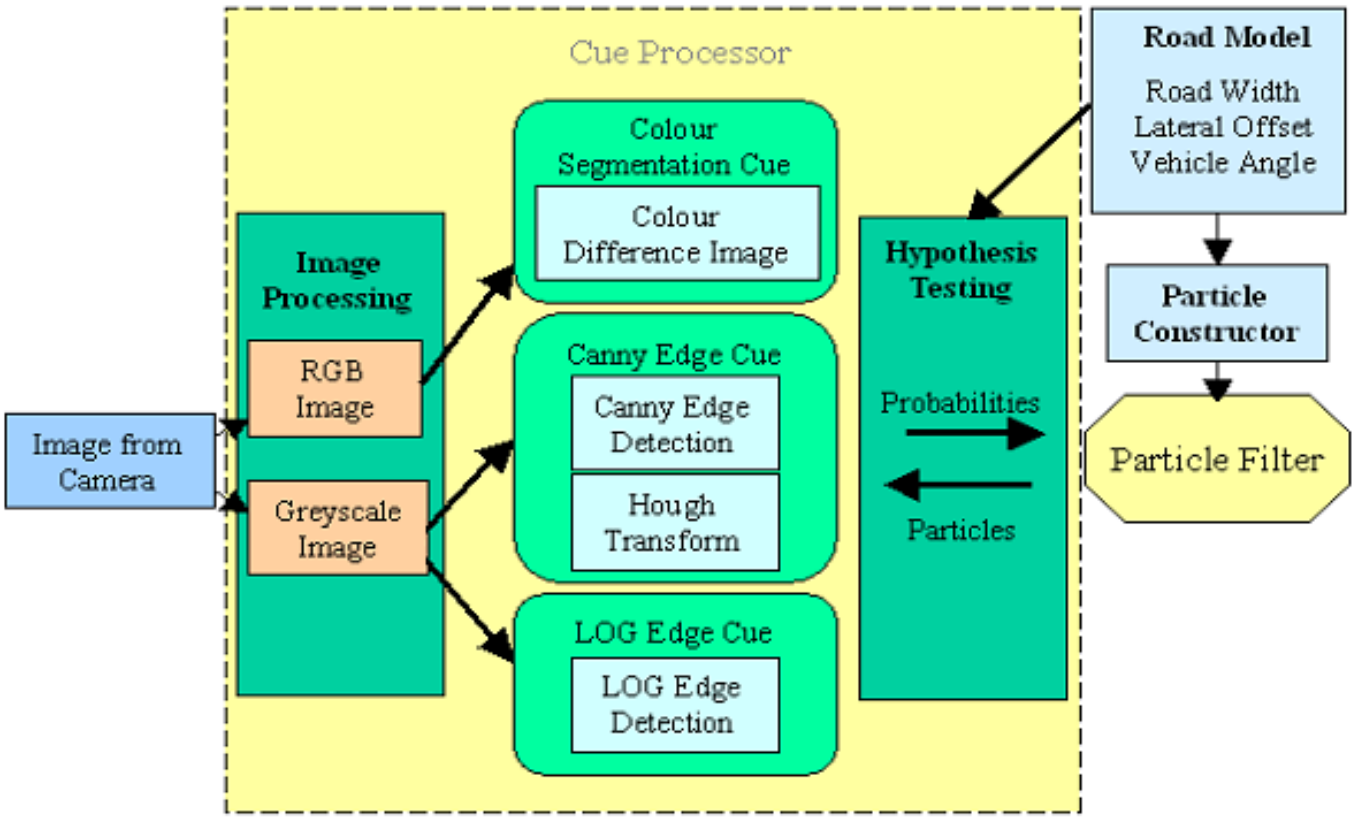
- Straight-line road model is assumed which is valid for small curvature roads (projective transformation preserves lines)
- Relative vehicle to road position consisting in three components: road width, yaw and offset:

$$X_d = \{x^{(i)} | x^{(i)} = (L^{(i)}, \phi^{(i)}, d^{(i)}) \ , i = 1, \ldots, N\}$$



Vision module scheme

- Three basic components: image processing, cue testing (*[Zelinsky et al., 2003]*), particle filtering.



Particle filtering

- Also known as Condensation or Monte Carlo algorithm (*[Isard et al., 1996]*)
- Based on Bayesian reasoning under Markov assumption
- Recursive Bayesian update equation:

$$Bel(x_t) = \eta P(o_t|x_t) \int P(x_t|x_{t-1}, a_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

$Bel(x_t)$ - current belief about system state

$P(o_t|x_t)$ - sensor model

$P(x_t|x_{t-1}, a_{t-1})$ - action model

Particle representation

- Idea: representing the continuous belief $Bel(x_t)$ by a set of N weighted samples – particles:

$$Bel(x_t) \approx \{x_t^{(i)}, \omega^{(i)}\}_{i=1,\ldots,N}$$

$x_t^{(i)}$ – sample state, a pose

$\omega^{(i)}$ – importance factor (probability measure)

- Recursive discrete belief update:

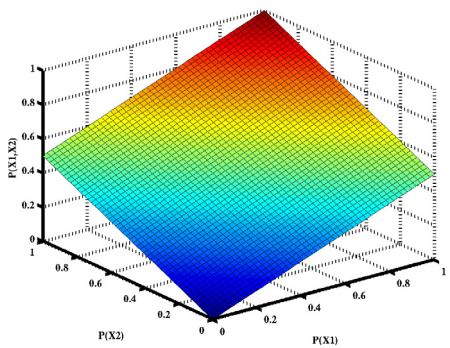
$$Bel(x_t) = \eta P(o_t|x_t) \sum_{i=1}^{N} P(x_t|x_{t-1}, a_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

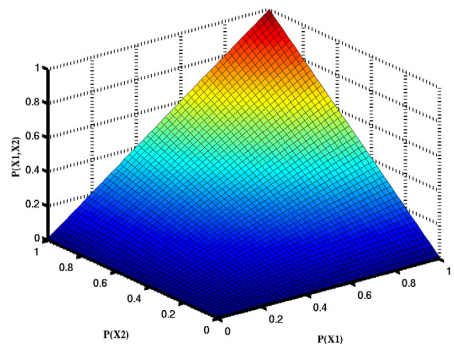
Cue processing - Inference

- A j-th cue (image filter) performs testing on the whole particle set $\{x_t^{(i)}, \omega^{(i)}\}$ to obtain j-th conditional probability distribution $P(o_t^{(j)}|x_t)$.
- Total a posteriori pdf for the sensor model:

$$P(o_t|x_t) = \prod_{j=1}^{M} P(o_t^{(j)}|x_t)$$

- Probabilistic inference - summation vs. product



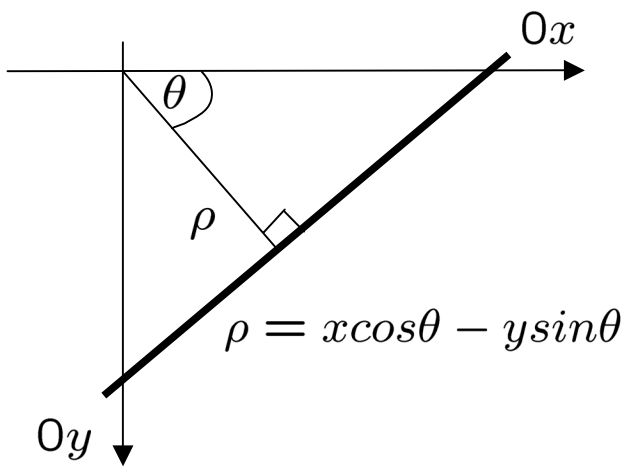


Cue testing

- Cues test different information on state of the vehicle relative to the road
- RGB input image for color cue processing
- Smoothed grayscale image for edge cue processing

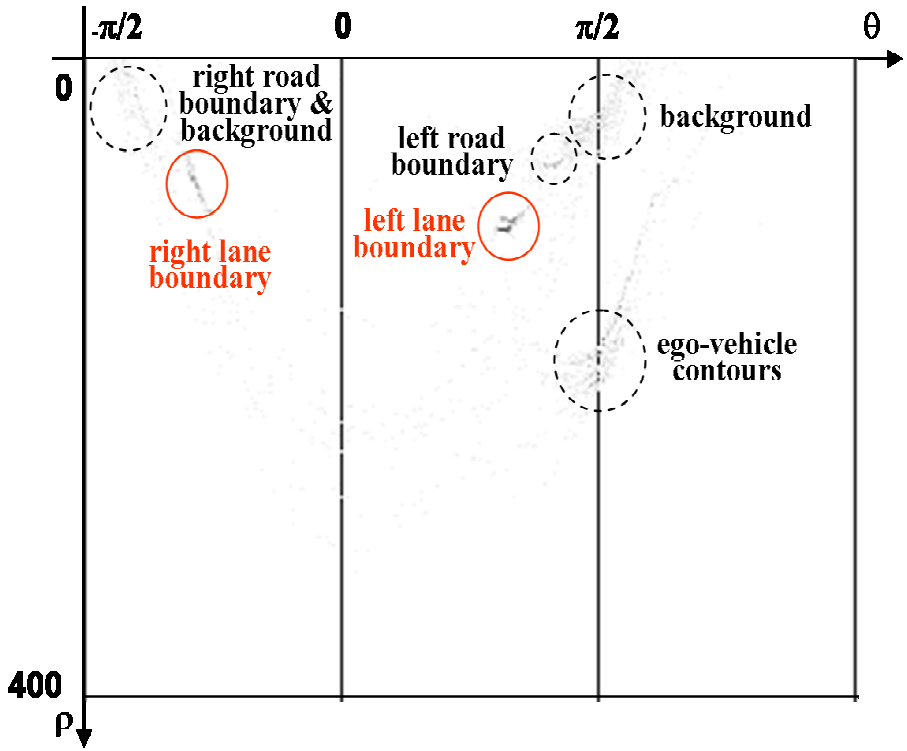
Canny - Hough filter cue (1)

- Canny edge filter:
  - First order adaptive threshold filter
  - Result: single pixel connected edges
- Hough transform:
  - A transform that accumulates pixels lying along a common line (or a generalized curve)
  - The line transform:



Canny - Hough filter cue (2)

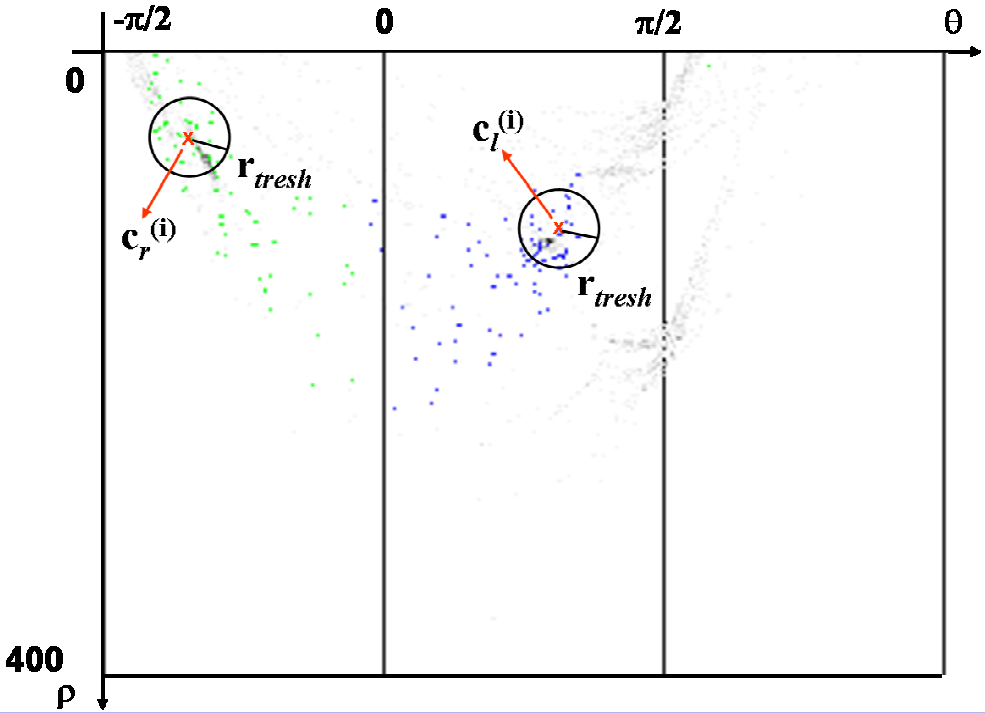
- A Canny edge pixel map and the resulting Hough transform array:



- Lane boundaries: $(\theta_l, \rho_l) = \{59^\circ, 105\}$, $(\theta_r, \rho_r) = \{-51^\circ, 78\}$.

Canny - Hough filter cue (3)

- Each particle's state $x^{(i)}$ defined by left and right lane edge Hough centers $c_r^{(i)} = \{\theta_{cr}^{(i)}, \rho_{cr}^{i}\}$, $c_l^{(i)} = \{\theta_{cl}^{(i)}, \rho_{cl}^{i}\}$.
- Testing of each lane edge against a circular cluster region of Hough edge map:



Canny - Hough filter cue (4)

- The choice of transformation origin determines sensitivity of Hough transform to gradient direction error of the Sobel derivative operator
- A single centrally placed transformation origin not satisfactory
- Solution: **two** Hough transforms for each particle – upper left L and right R image corner origin (3 quadrant transform) (*[Macek et al., 2004]*)
- Final *i*-th particle probability:

$$\omega_{Canny}^{(i)} = \omega_L^i \omega_R^i$$

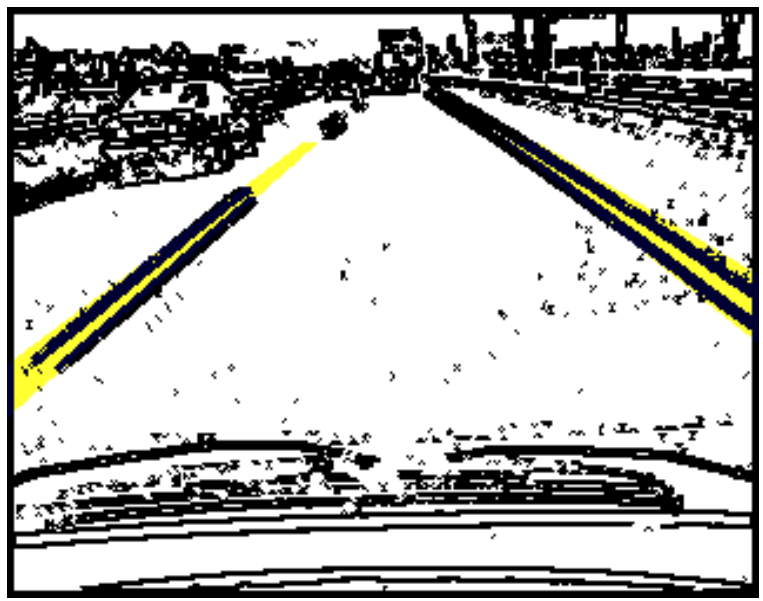
LoG filter cue

- Laplacian-Gaussian kernel edge filter with fixed tresholding
- Result: thick edge boundaries suitable for direct perspective mask testing in image space
- Total edge pixel hit within a boundary:

$$\delta_{l,r}^{(i)} = \sum_{k=1}^{N_{l,r}} I_{l,r}$$

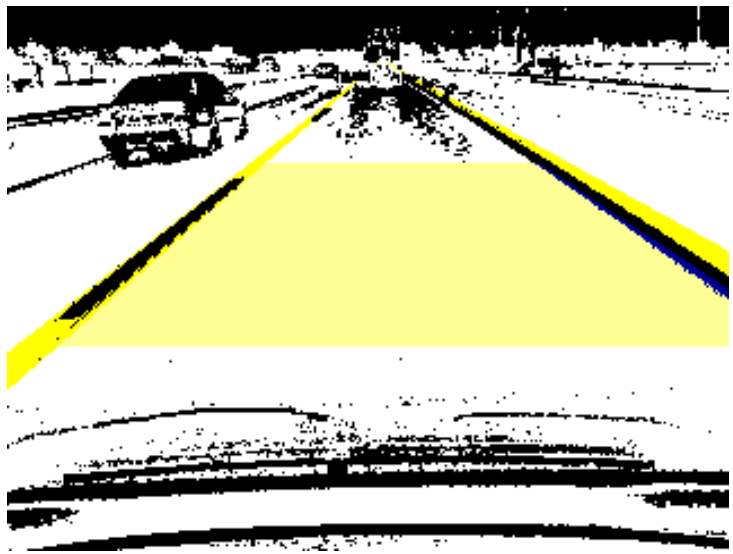
- Total i-th particle probability:

$$\omega_{LoG}^{(i)} = \left(\frac{\delta_l^{(i)} - \delta_{l_{min}}}{\delta_{l_{max}} - \delta_{l_{min}}} + p_0\right)\left(\frac{\delta_r^{(i)} - \delta_{r_{min}}}{\delta_{r_{max}} - \delta_{r_{min}}} + p_0\right)$$



Color segmentation filter cue

- RGB color difference map (delta map)



- Road and non-road pixel count:

$$\delta_{l,r,c}^{(i)} = \sum_{k=1}^{N_{l,r,c}} I_{l,r,c}$$

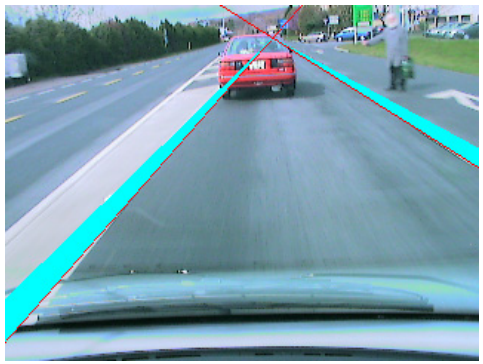
- Testing of left and right lane non-road colour boundary stripes and central road colour area:

$$\omega_{Color}^{(i)} = \left(\frac{\delta_l^{(i)} - \delta_{l_{min}}}{\delta_{l_{max}} - \delta_{l_{min}}} + p_0\right)\left(\frac{\delta_c^{(i)} - \delta_{c_{min}}}{\delta_{c_{max}} - \delta_{c_{min}}} + p_0\right)\left(\frac{\delta_r^{(i)} - \delta_{r_{min}}}{\delta_{r_{max}} - \delta_{r_{min}}} + p_0\right)$$

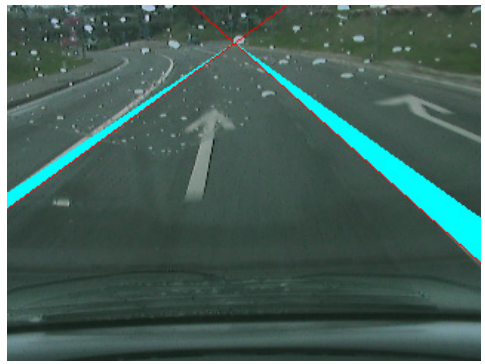
Results - Magistral road

- Various road conditions tested

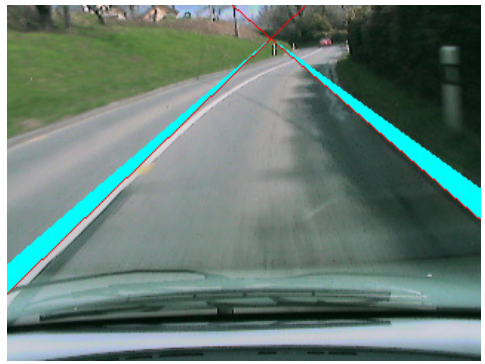
Inner city traffic



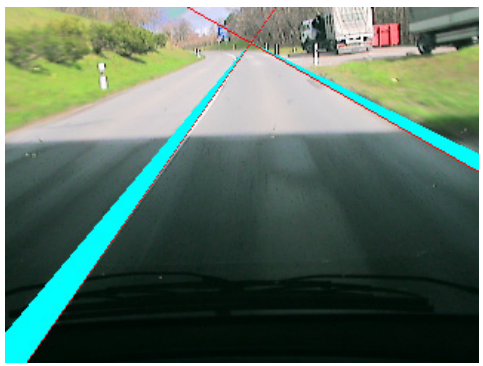
Ground signs



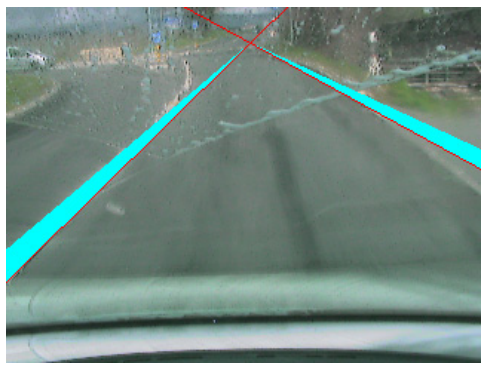
Country-side lane



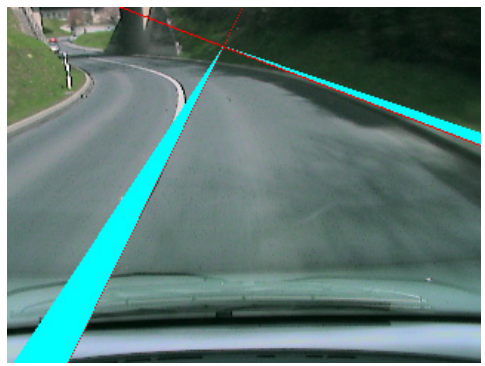
Tunnel exit



Obscured windscreen

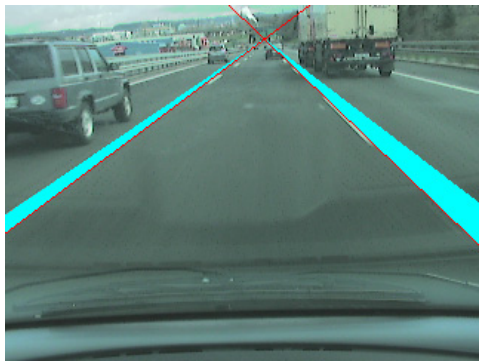


High curvature

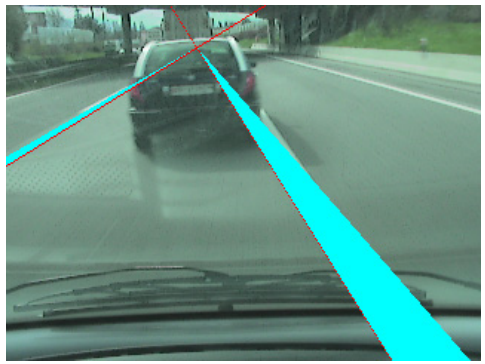


Results - Highway

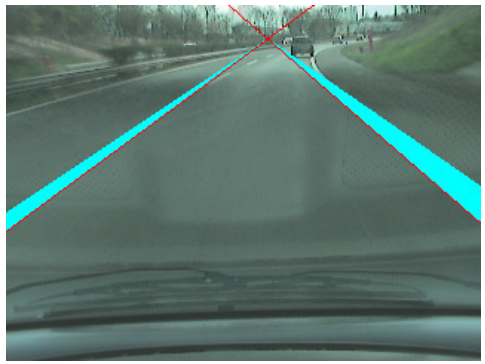
Multilane traffic



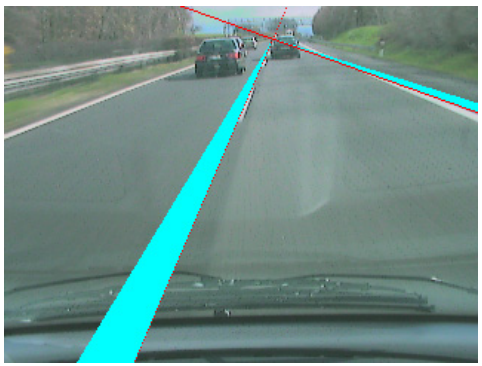
Car occluding



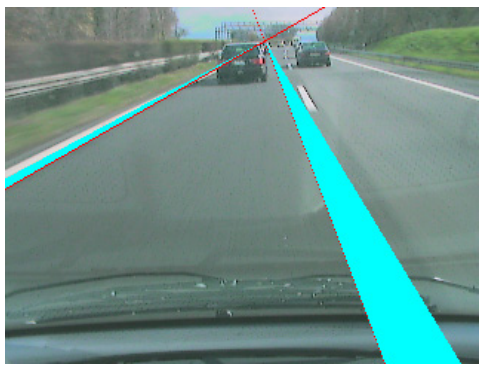
High curvature



Lane changing (1)

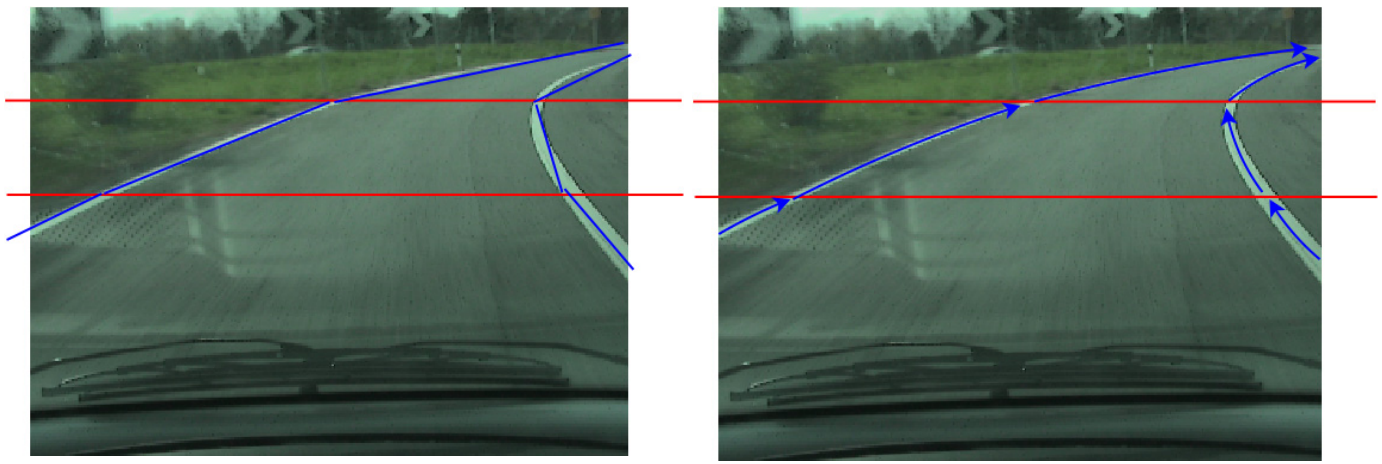


Lane changing (2)



Extensions

- Cue processing flexible for adding different information (also non-image based)
- Extension to a curved model based on a polyline fit



- Introduction
- Lane detection vision module
- Off- and on-line path planning
- Obstacle avoidance
- Path following
- Conclusions

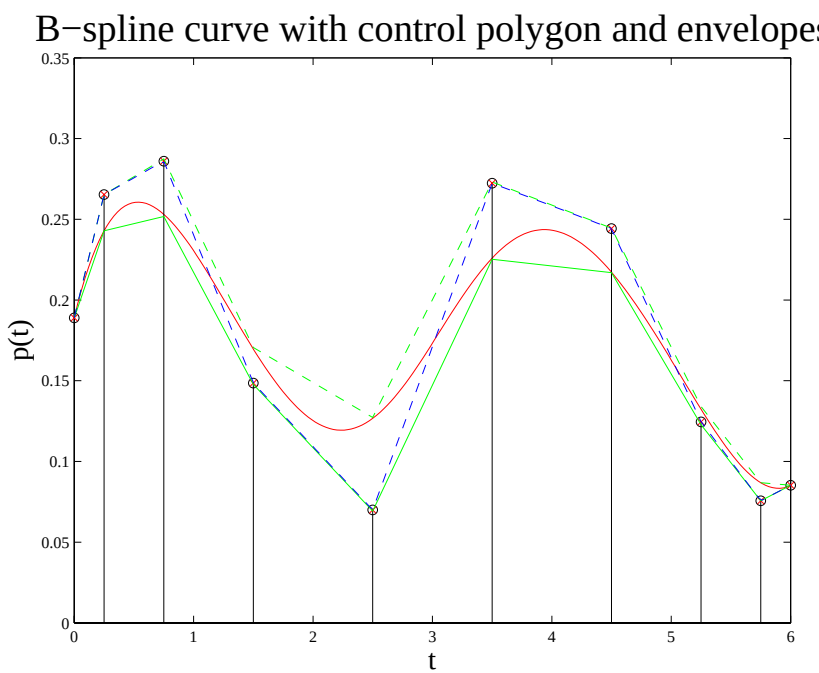
Off-line path planning

- Smooth paths with different optimality criteria possible
- B-spline curve representation:
  - Smoothness to any order can be acquired
  - Local controllability of the curve
  - Convex hull property of B-spline curve within its control polygon
- B-spline curve described by control point vector $\vec{b} = [b_0, \ldots, b_m]$ and basis functions N_j^d of degree d on the knot vector $t_0 \leq t_1 \ldots \leq t_{m+d+1}$:

$$p(t) = \sum_{j=0}^{m} b_j \, N_j^d(t)$$

Control polygon and envelopes

- Control polygon $l(t)$ - a piecewise linear interpolant at control points $\vec{b}$
- If the knot vector is fixed, tight linear envelope bounds $e_{low}(t), e_{high}(t)$ can be defined (*[Lutterkort, 2000]):*

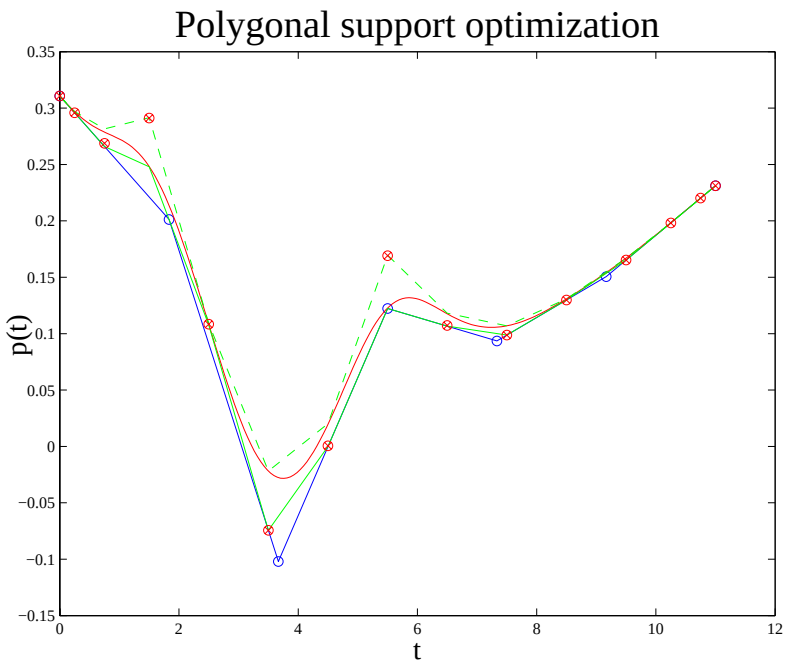


Support problem optimization

- Environment is a polygonal chain *c* - structured environment
- SUPPORT problem: given a poygonal chain *c* find a B-spline curve that fits as close as possible to *c*
- Linear optimization problem:

$$\min_{\vec{b}} \sum_{j=0}^{m} (b_j - cj)$$

subj. to: $\left\{ c \leq e_{low} \right\}$



Channel problem optimization

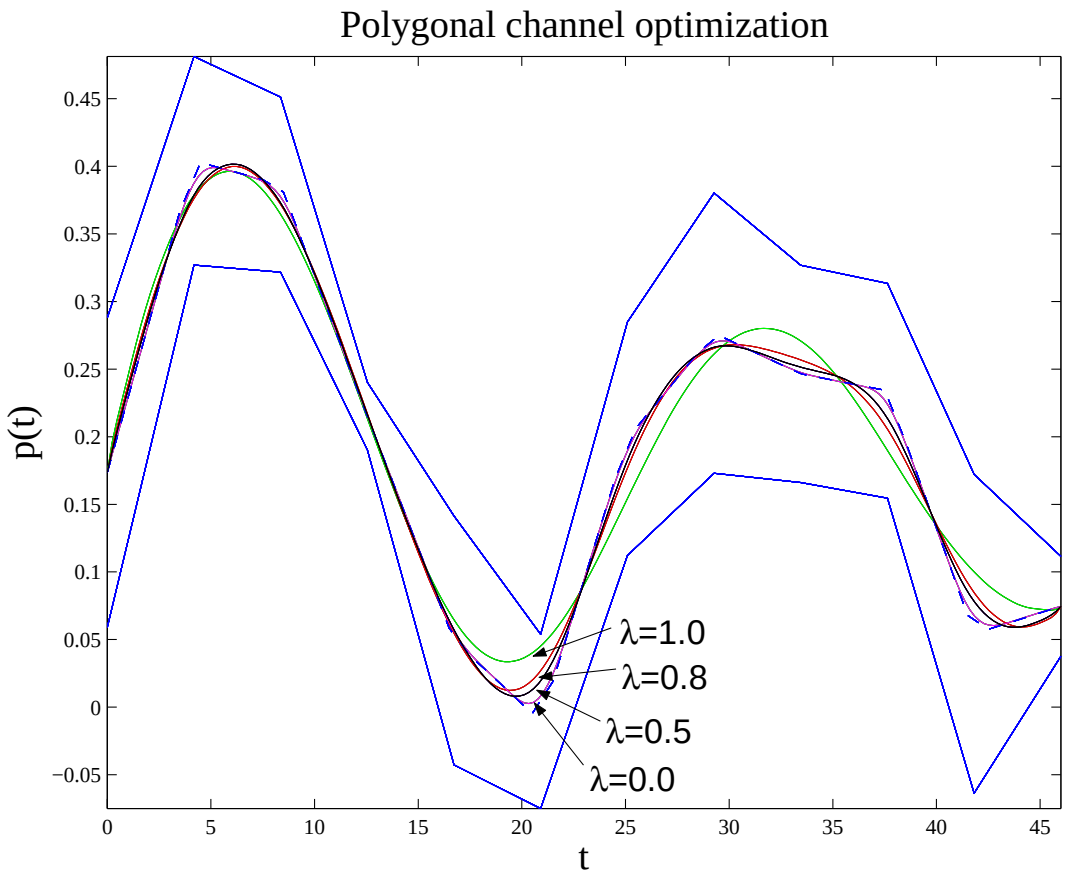
- CHANNEL optimization problem: given a poygonal chain c_{low} , c_{high} find a B-spline curve that minimizes the curvature change and maximizes the safety distance
- Nonlinear optimization problem:

$$\min_{\vec{b}} \left\{ \int_{t_0}^{t_{m+d+1}} \dot{K}(s)ds + \sum_{j=0}^{m} (b_j - c_j) \right\}$$

subj. to:

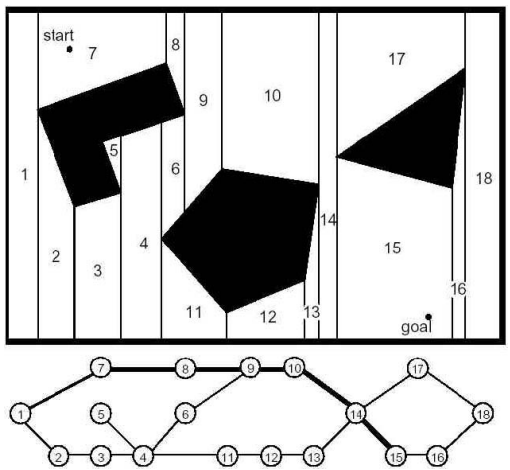
$$\left\{ \begin{array}{c} c_{low} \leq e_{low} \\ e_{high} \leq c_{high} \end{array} \right\}$$

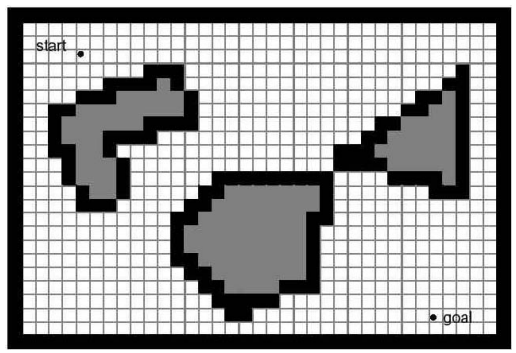
Smoothness and safety optimization

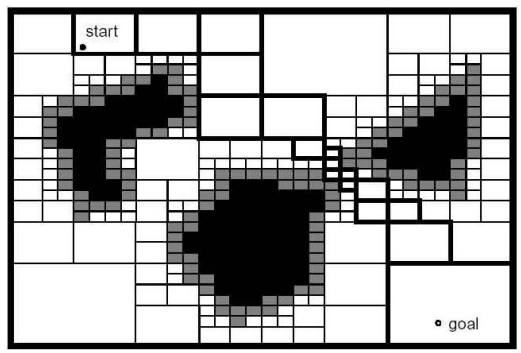


Path planning - Cell decompostion

- Cell decomposition path planning:
  - Exact cell decomposition
  - Approximate cell decomposition
  - Adaptive cell decomposition



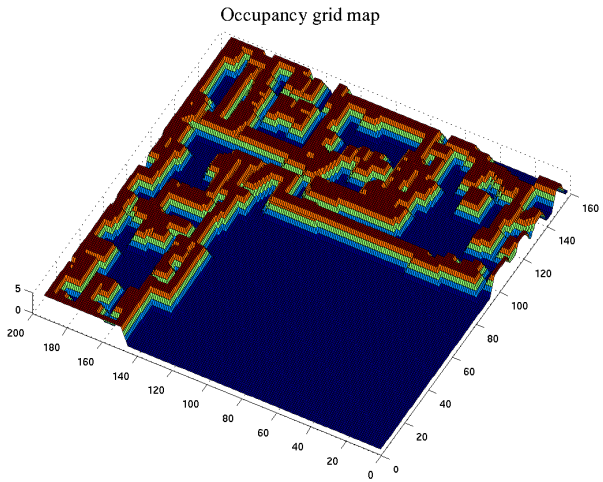




[*courtesy R.Siegwart et al.*]

Occupancy grid based graph search

- Occupancy grid map with cost mask expansion:



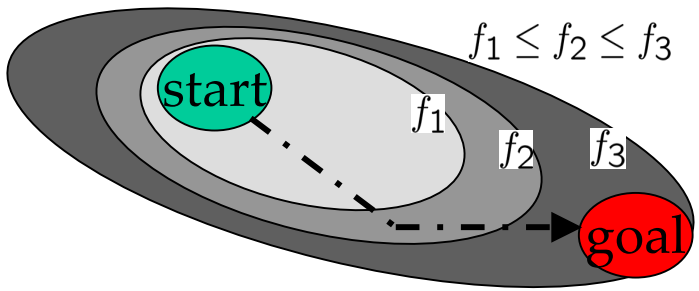
- Graph search algorithms implemented for finding the path between the start and a goal configuration
  - A* graph search (*[Dijkstra, 1959]*)
  - D* graph search (*[Stentz, 1994]*)

A* path search algorithm

- Uses node expansion from the start to goal position
- Each node (grid cell) is assigned a cost function:

$$f(n) = g(n) + h(n)$$

- Optimality and completeness guaranteed

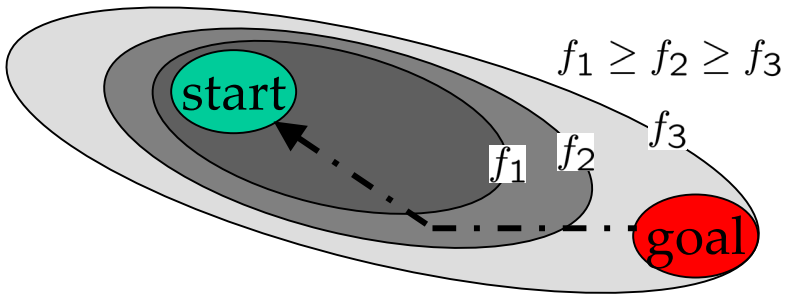


- Major drawback: computational inefficiency

in dynamic environments

D* path search algorithm (1)

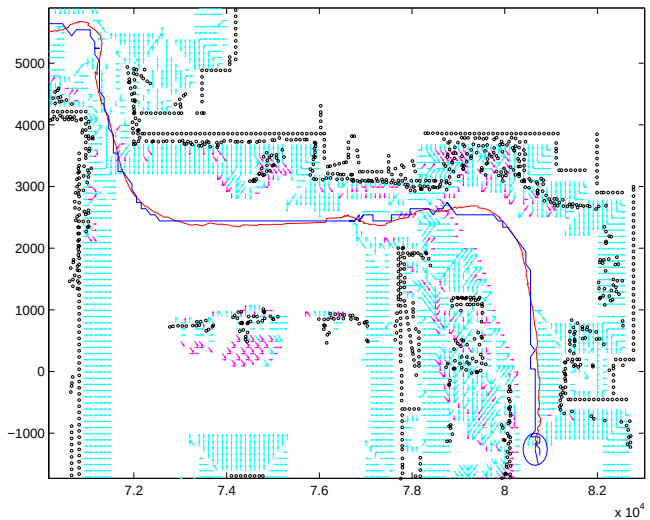
- Node expansion from goal to start position in an incremental manner by maintaining optimal wave front
- Reuses search information from previous steps

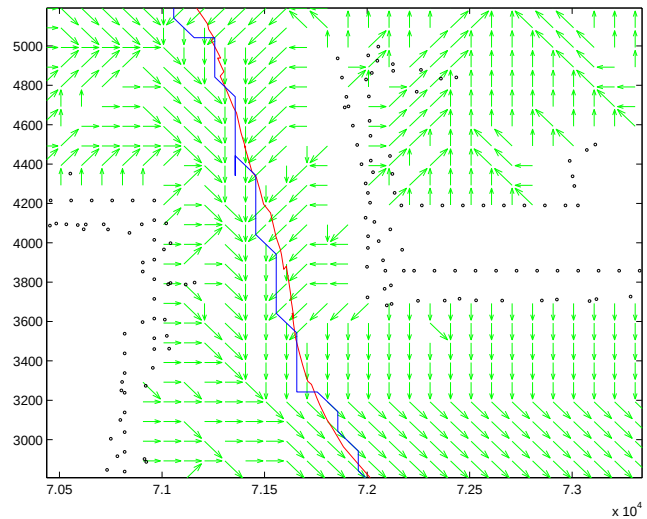


- Suitable for path search in dynamic environments by maintaing a list of RAISE and LOWER states

D* path search algorithm (2)

- Path represented as a list of pointers



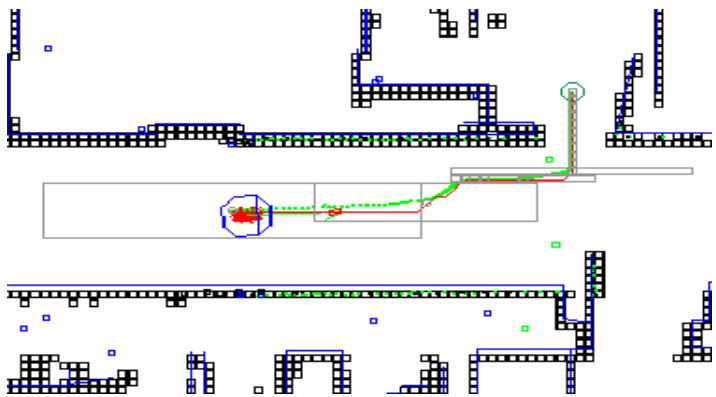


Free-space bubbles

- A coarse representation of the global geometric path composed of connected occupancy grid cells
- A rectangular bubble formed as free space around the current robot configuration $\vec{q}$ and the nearest obstacle (*[Quinlan et al., 1993]*):

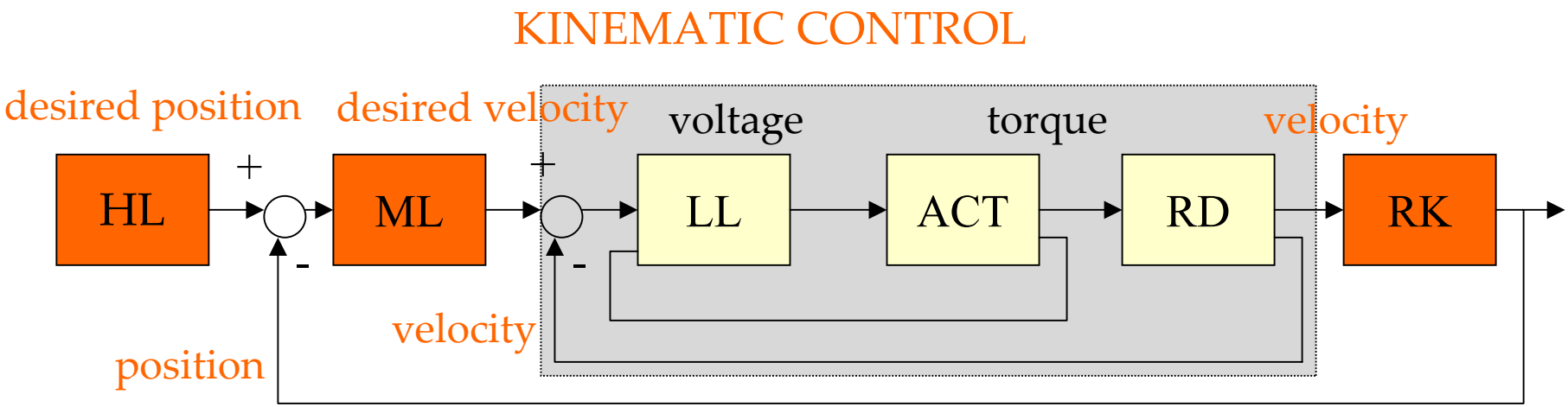
$$\beta(\vec{q}) = \{\vec{p} : |q_x - p_x| < d_x, |q_y - p_y| < d_y\}$$

- Spline smoothing determined by free-space bubble centers



- Introduction
- Lane detection vision module
- Off- and on-line path planning
- Obstacle avoidance
- Path following
- Conclusions

Motion control functional diagram



HL – high level control
ML- medium level control
LL – low level control

RK – robot kinematics
RD – robot dynamics

Obstacle avoidance -Velocity space based approaches

- Velocity space based approaches:
  - Curvature -Velocity method (*[Simmons, 1996]*)
  - Lane - Curvature method (*[Simmons, 1998]*)
- Dynamic Window approach (*[Fox et al., 1997]*)
- Key feature: obstacle avoidance problem formulated in velocity space - kinematic and dynamic robot constraints are taken directly into account
- Geometric local obstacle configuration compared to possible robot trajectories

General motion equations (1)

- General robot motion equations given initial state, x_{t_0} , y_{t_0} , θ_{t_0} and velocities $v(t)$ and $\omega(t)$ in interval $t \in [t_0, t_n]$:

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t) cos\theta(t) \mathrm{d}t$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t) sin\theta(t) \mathrm{d}t$$

$$\theta(t_n) = \theta(t_0) + \int_{t_0}^{t_n} \omega(t) \mathrm{d}t$$

General motion equations (2)

- Between two arbitrary points in time t_0 and t_n robot can be controlled only by limited number of acceleration commands $\dot{v}_i$ and $\dot{\omega}_i$ for $i = 1, 2, \ldots, n$ that are assumed piecewise constant in $\left[t_i, t_{i+1}\right]$.

- Piecewise independent trajectory generation:

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} \left(v(t_i) + \dot{v}_i \Delta_t^i\right) \cdot$$
$$\cdot cos\left(\theta_{t_i} + \omega(t_i)\Delta_t^i + \frac{1}{2}\dot{\omega}_i(\Delta_t^i)^2\right) \mathrm{dt}$$

Piecewise circular arc trajectories (1)

- Approximation by sequence of circular arcs:

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} \left(F_x^i(t_{i+1}) \right)$$

where:

$$F_x^i(t) = \begin{cases} \frac{v_i}{\omega_i} \left(sin\theta(t_i) - sin\left(\theta(t_i) + \omega_i \left(t - t_i\right)\right)\right) & : \quad \omega_i \neq 0 \\ v_i cos\left(\theta(t_i)\right) t & : \quad \omega_i = 0 \end{cases}$$

- Center of rotation:

$$M_x^i = -\frac{v_i}{\omega_i} sin\theta(t_i)$$

Piecewise circular arc trajectories (2)

- Robot trajectory in interval $\left[t_i, t_{i+1}\right]$:

$$\left(F_x^i - M_x^i\right)^2 + \left(F_y^i - M_y^i\right)^2 = \left(\frac{v_i}{\omega_i}\right)^2$$

where v_i and ω_i are constant.

- Simulation of robot trajectory for a fixed control sequence:

$$v_1 = v_2 = \ldots = v_n \quad \& \quad \omega_1 = \omega_2 = \ldots = \omega_n$$

reduces to a single circular arc.

Velocity search space for Dynamic Window approach

- Generating robot trajectory to a goal position in $[t_0, t_n]$ is a 2-dimensional search of velocity vectors (v_i, ω_i) in each control interval – one step optimization.
- Circular trajectory constraints
- Possible velocity space – max. velocity constraints :

$$V_s = \{v, \omega | v \leq v_{max} \wedge \omega \leq \omega_{max}\}$$

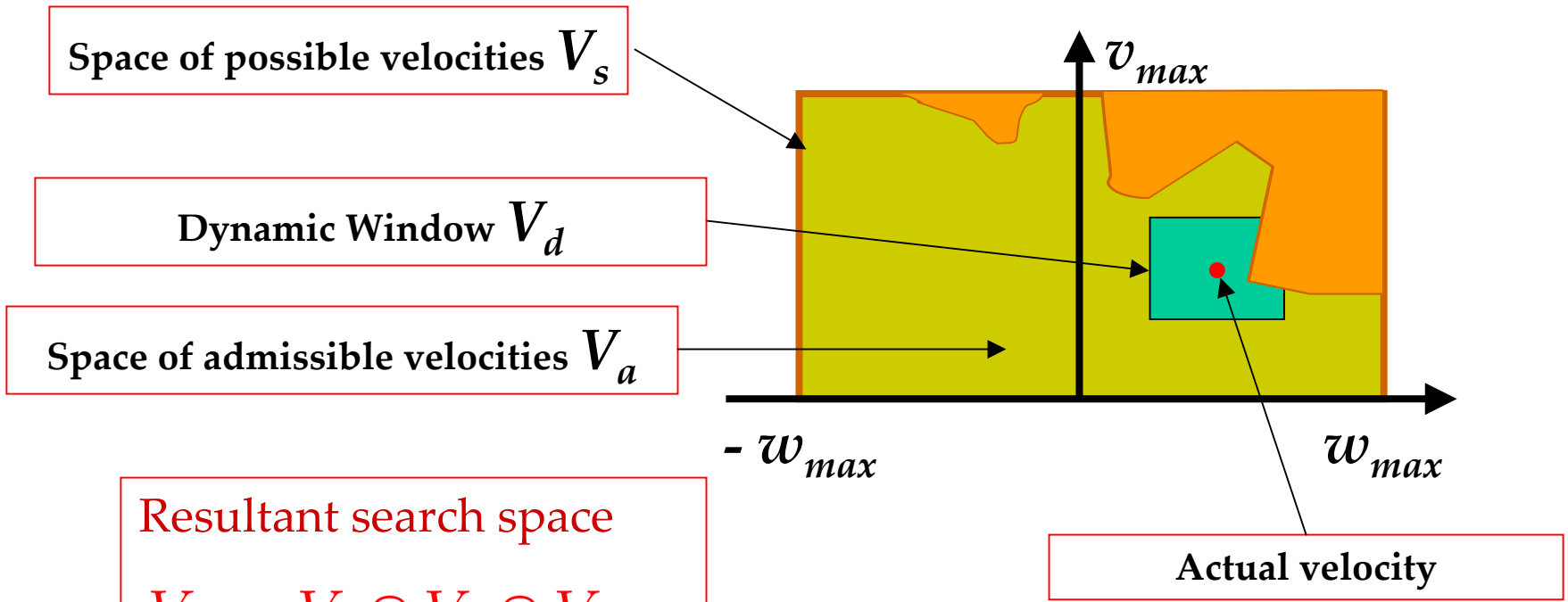
- Admissible velocity space – obstacle constraints:

$$V_a = \left\{v, \omega \leq \sqrt{2\rho_{min}(v,\omega)\dot{v}_b} \wedge \sqrt{2\rho_{min}(v,\omega)\dot{\omega}_b}\right\}$$

Resultant search space

- Dynamic window – acceleration constraints:

$$V_d = \{(v, \omega) | v \in [v_a - \dot{v}t, v_a + \dot{v}t] \wedge \omega \in [\omega_a - \dot{\omega}t, \omega_a + \dot{\omega}t]\}$$



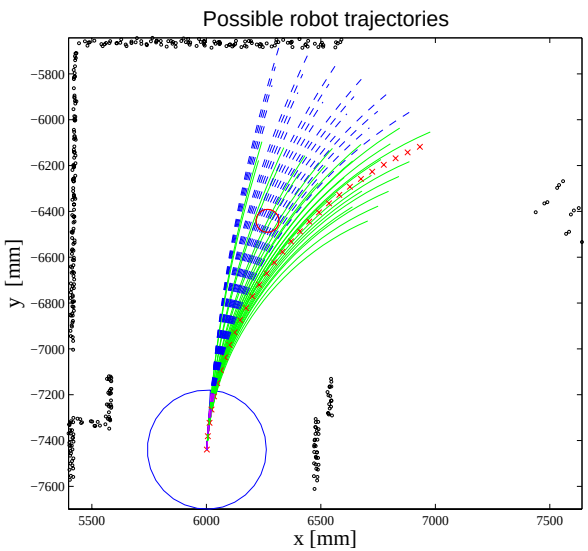
Resultant search space

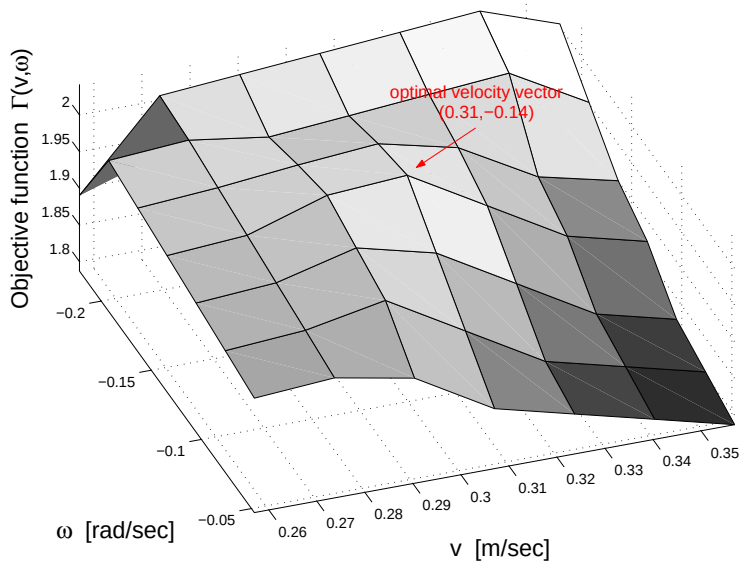
$$V_r = V_s \cap V_a \cap V_d$$

Global objective

- From a discrete set of velocity vectors (v_k^i, ω_k^i) within resultant search space V_r, the $(v_{optim}^i, \omega_{optim}^i)$ is chosen that maximizes objective function:

$$\Gamma(v,\omega) = \lambda_{clear}\vartheta clear + \lambda_{head}\vartheta head + \lambda_{vel}\vartheta vel$$

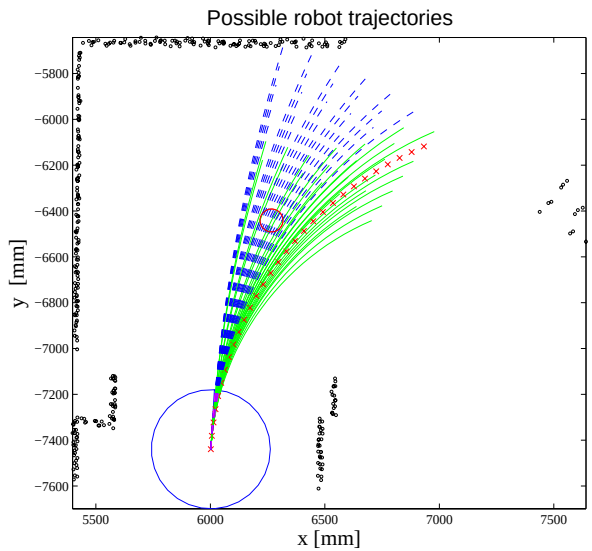


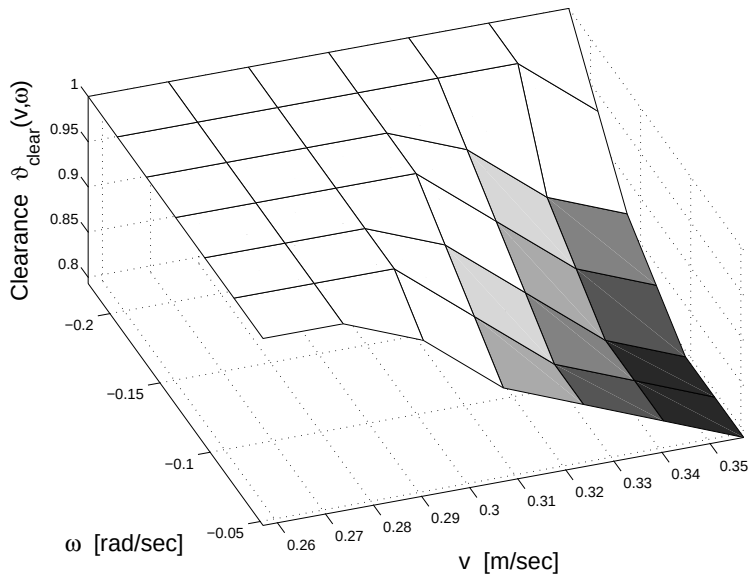


Clearance objective

- Time to collision with obstacles based measure:

$$\vartheta_{clear}(v,\omega) = \begin{cases} 0 & : \quad t_{col}(v,\omega) \leq T_b(v,\omega) \\ \frac{t_{col}(v,\omega) - T_b(v,\omega)}{T_{max} - T_b(v,\omega)} & : \quad T_b(v,\omega) < t_{col}(v,\omega) < T_{max} \\ 1 & : \quad t_{col}(v,\omega) > T_{max} \end{cases}$$

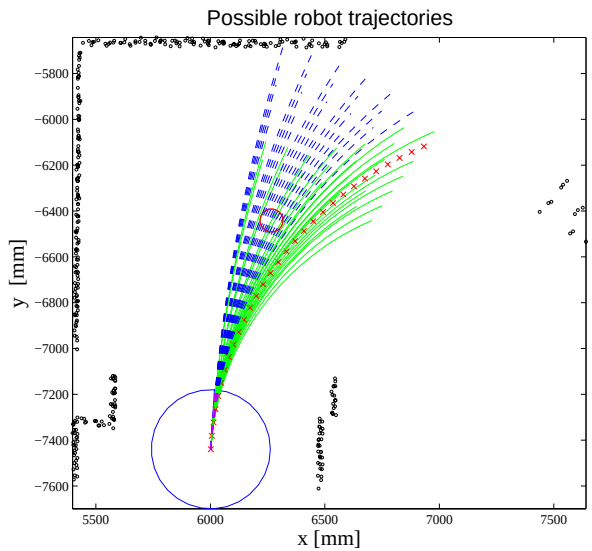


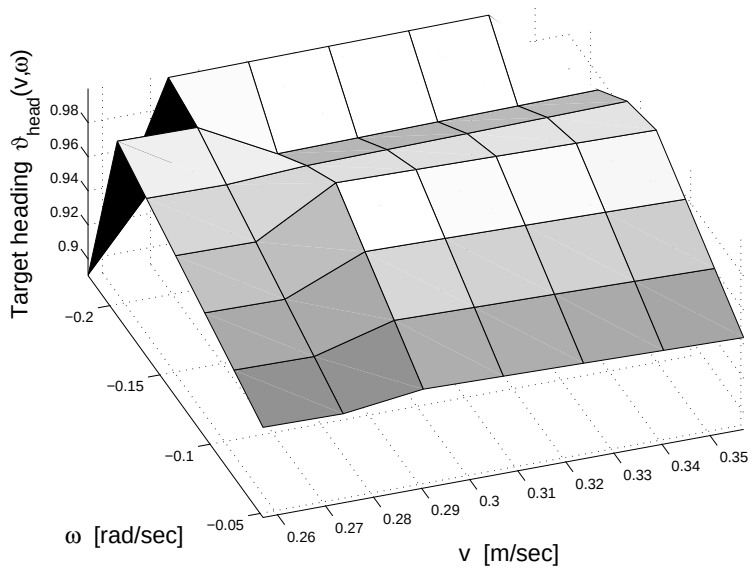


Target heading objective

- Based on heading difference between the current goal and predicted robot orientation in next step:

$$\vartheta_{head}(v,\omega) = 1 - \frac{|\theta_t(v,\omega) - \psi_p(v,\omega)|}{\pi}$$

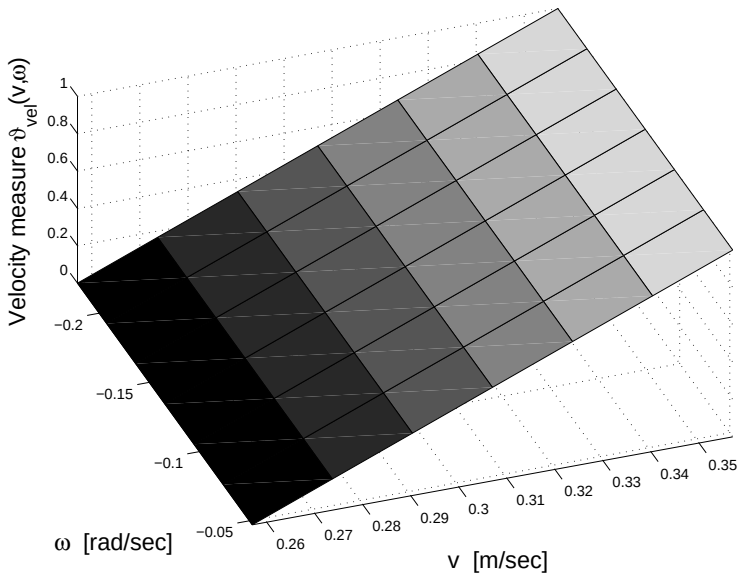




Translational velocity objective

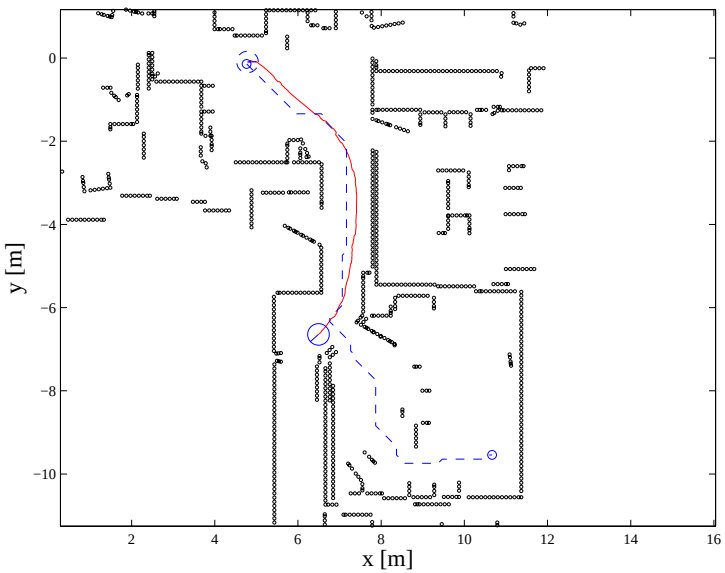
- Higher translational velocities within the current dynamic window preferred:

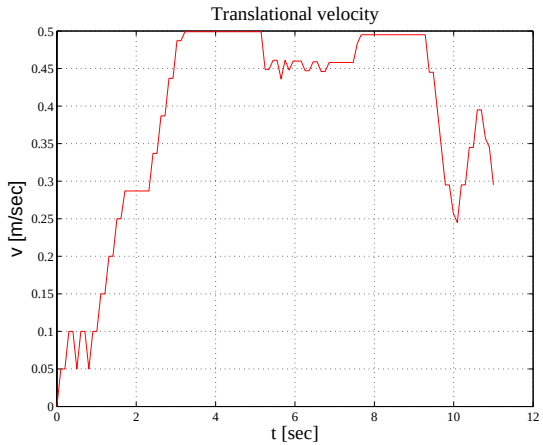
$$\vartheta_{vel}(v, omega) = \frac{v - v_{d_{min}}}{v_{d_{max}} - v_{d_{min}}}$$

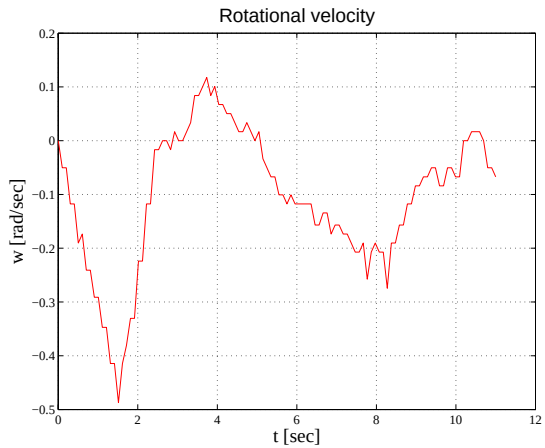


Sensitivity to different objectives (1)

- Overshoot situation – clearance vs. heading and velocity objectives.

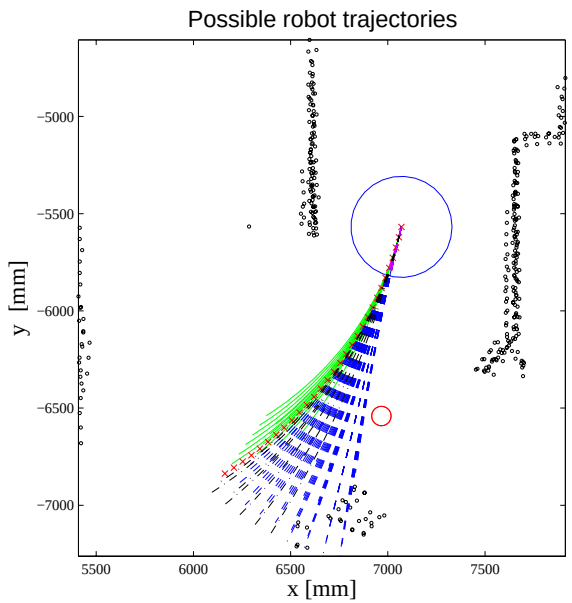


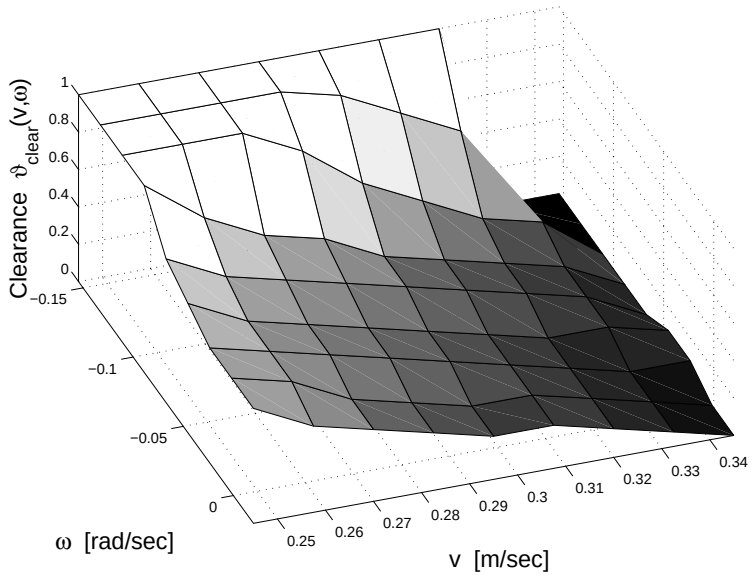




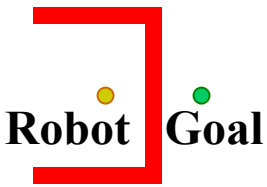
Sensitivity to different objectives (2)

- Snapshot of a doorway entering phase





- Local minima problem:



- Introduction
- Lane detection vision module
- Off- and on-line path planning
- Obstacle avoidance
- Path following
- Conclusions

Path following - Introduction

- Path following: kinematic model based motion control
- Path tracking: dynamic model based motion control
- Reference point based approaches: reference point determines the effective "line-of-sight" of the robot along the geometric path
- Two distinct approaches:
  - Global integration of local obstacle avoidance techniques - Global path integrated Dynamic Window
  - Inherent global path following techniques - Modified virtual vehicle approach

Global path integrated Dynamic Window approach - Introduction

- Aim: reduce sensitivity to opposing objectives in original Dynamic Window approach (path clearance vs. target heading and translational velocity measure)
- Solving local minima problem - global space connectivity achieved using a path generated by a graph search planner on occupancy grid map
- Initial global geometric path is line segmented with multiples of 45°- not suitable for path following

Related approaches

- Global Dynamic Window (*[Brock et al., 2000]*)

$$\Gamma(v,\omega) = \lambda_{clear}\vartheta clear + \lambda_{vel}\vartheta vel + \lambda_{nf1}nf1 + \lambda_{\Delta nf1}\Delta nf1$$

  - parameter tuning difficult
  - translational velocity objective not related to obstacle configuration

- Reduced Dynamic Window (*[Arras et al., 2002]*)

$$\Gamma(v,\omega) = \lambda_{clear}\vartheta clear + \lambda_{head}\vartheta head$$

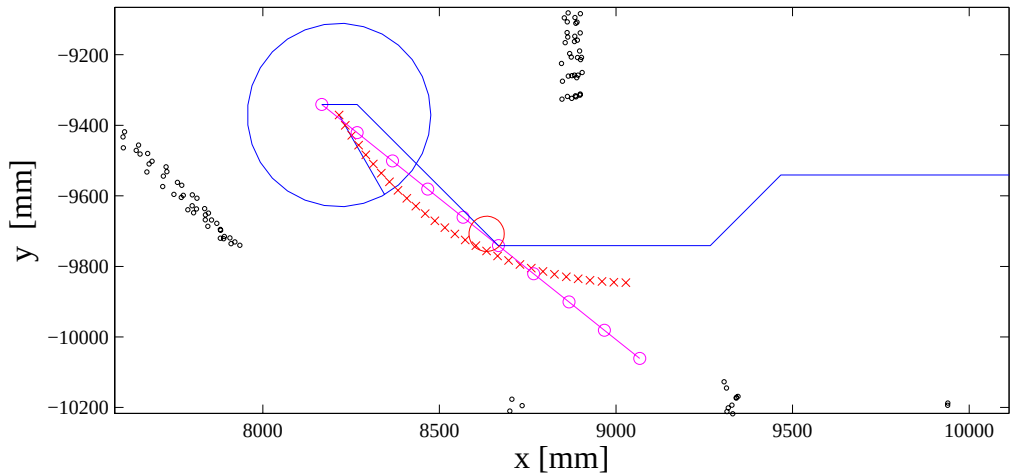
  - only a coarse set of predefined linear velocities allowed
  - polygonal shapes taken into account

Global connectivity - effective path

- Effective path introduced for alignment comparison between the DW generated trajectories and the global geometric path (*[Macek et al., 2003]*)
- Represents the global path in the local vicinity of the robot
- It is a straight line connecting the current robot position and the reference point on the global path.
- Its length determines the current translational velocity reference v_{ref} .
- Its orientation determines the current rotational velocity reference ω_{ref} .

Effective path vs. global path

- Its length and orientation change dynamically according to the current robot state and local path configuration (reference point)



- Path orientation change: related to inflection points on the global path
- Dynamic robot constraints: limit the effective path length

Path alignment measure

- Among all possible DW generated trajectories the optimal one is determined that is best aligned with the effective path
- Let k-th DW trajectory be represented by N_t points $\left\{(x_{t_k}(i), y_{t_k}(i)), i = 1, 2, \ldots, N_t\right\}$ and the effective path by $\{(x_p(j), y_p(j)), j = 1, 2, \ldots, N_p\}$ of N_p points and $d_{tk}(i, j)$ distance between two points on each curve
- Path alignment measure:

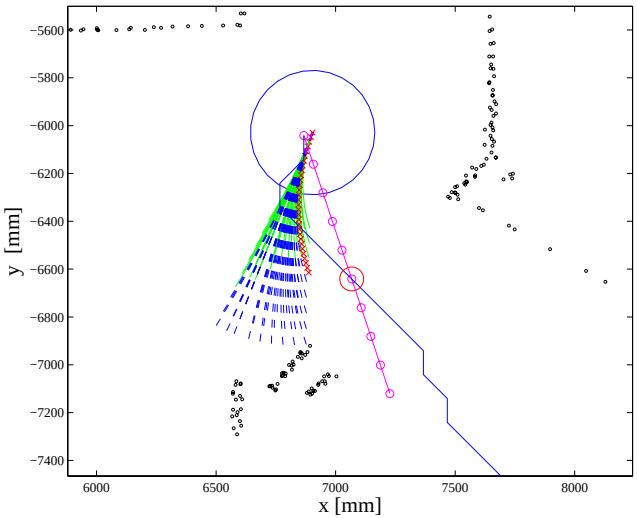
$$\vartheta_{path}(k) = \frac{\sum_{i=1}^{N_t} \sum_{j=1}^{N_p} j d_{t_k}(i, j) - D_{min}}{D_{max} - D_{min}}$$

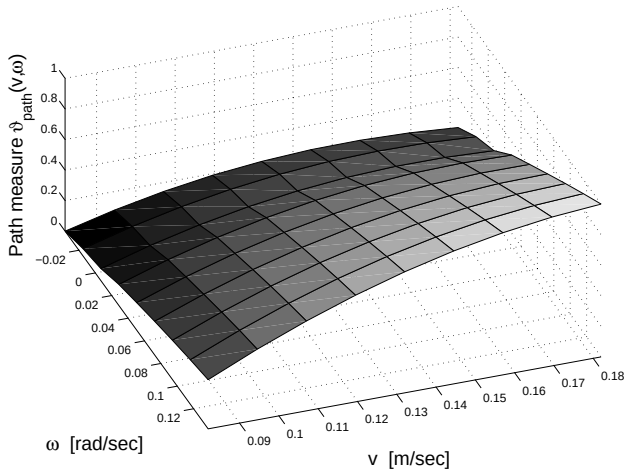
Global objective

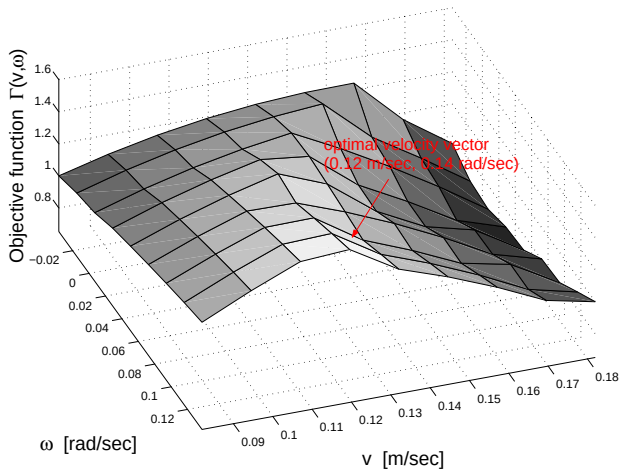
- Global path related objectives of heading $\vartheta head$ and translational velocity ϑvel integrated in a single measure $\vartheta path$:

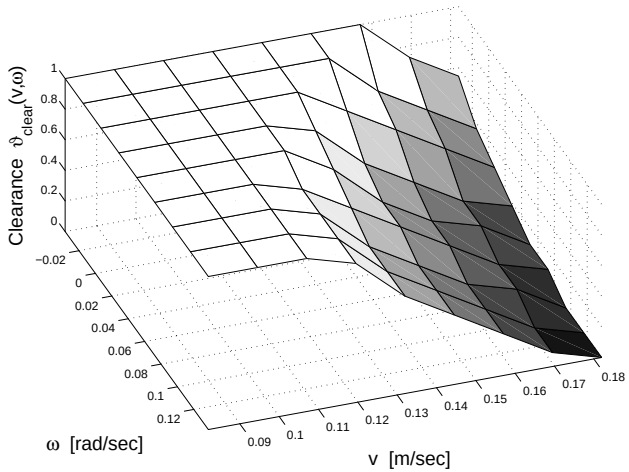
$$\Gamma(v, \omega) = \lambda_{clear} \vartheta clear + \lambda_{path} \vartheta path$$

A corridor environment run (1)



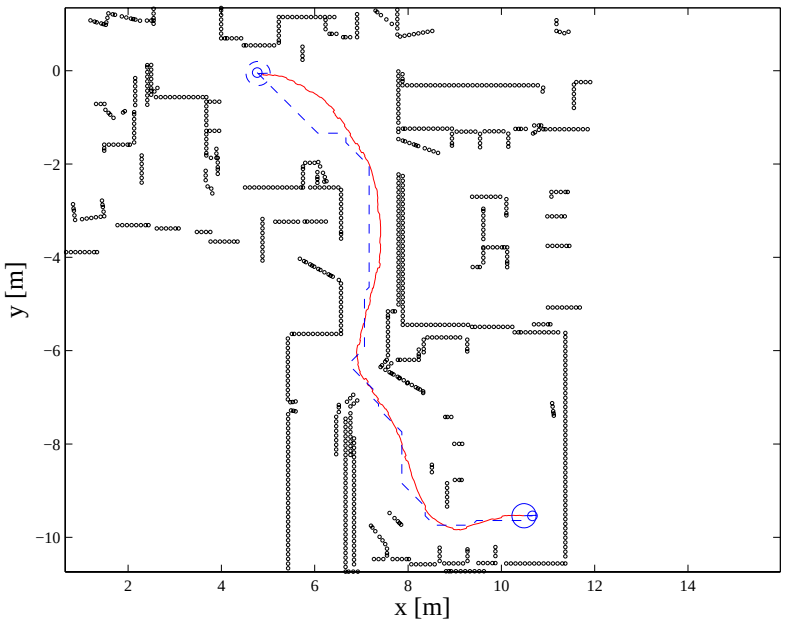


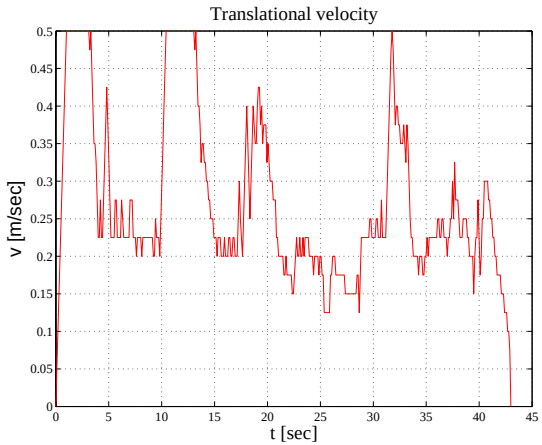


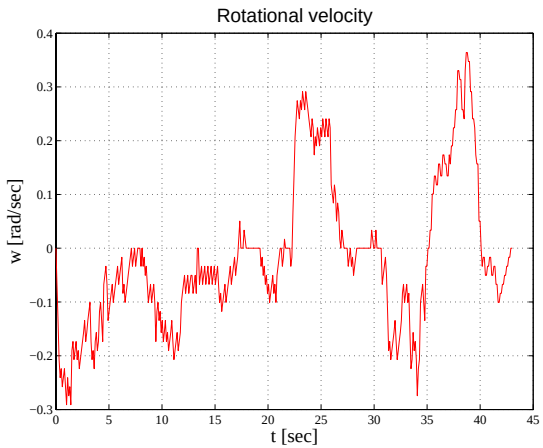


A corridor environment run (2)

- Successful doorway entry maneuver

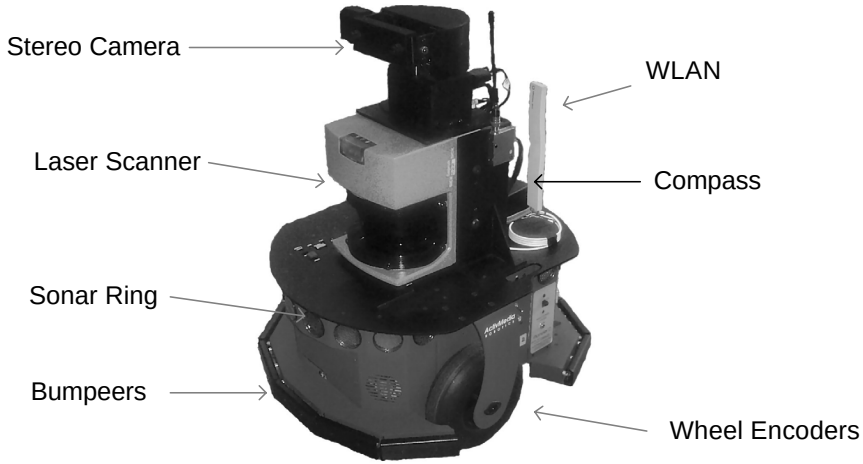






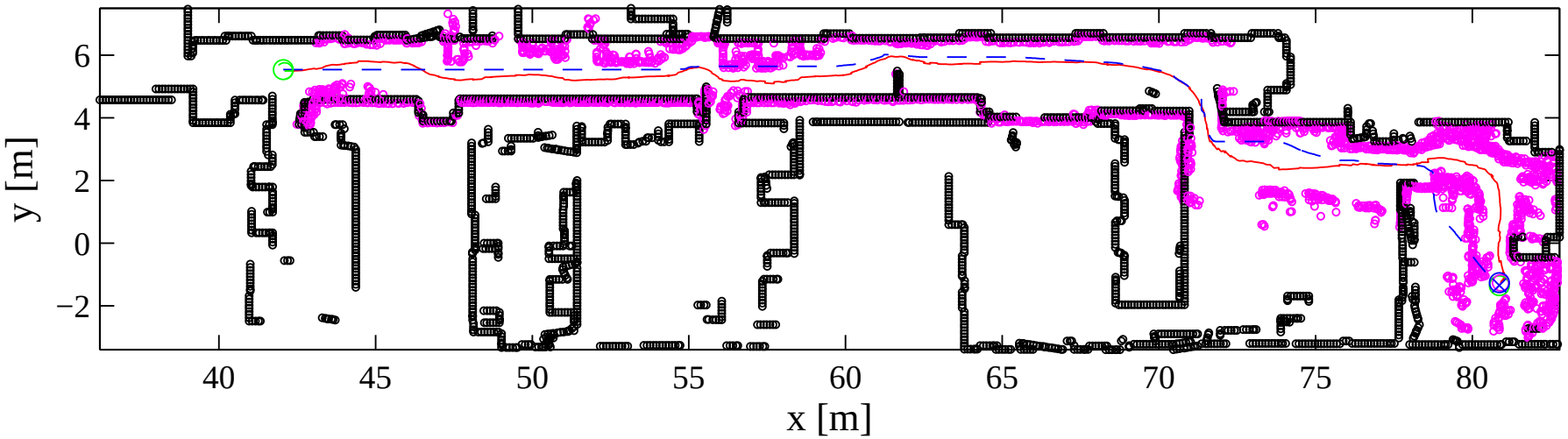
Experimental results – Robot setup

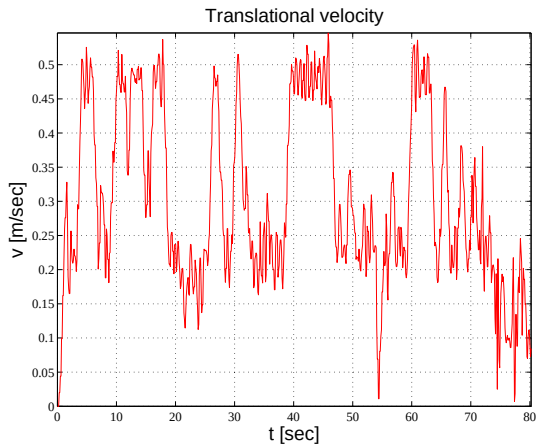
- A Pioneer 2 ActiveMedia Robotics robot platform

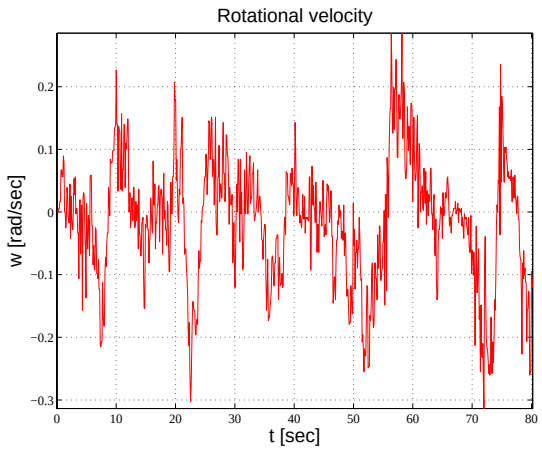


- A proprietary server-client software architecture
- An UDP socketing communication implemented for off-board computation

Experimental results – Global path integrated Dynamic Window (1)







Virtual vehicle approach

- Control algorithm statement – finding a longitudinal control $v(t)$ and lateral control $\omega(t)$ to follow a smooth path $c(s)$ described by curve parameter s
- Desired reference point coordinates on the curve $c(s)$:

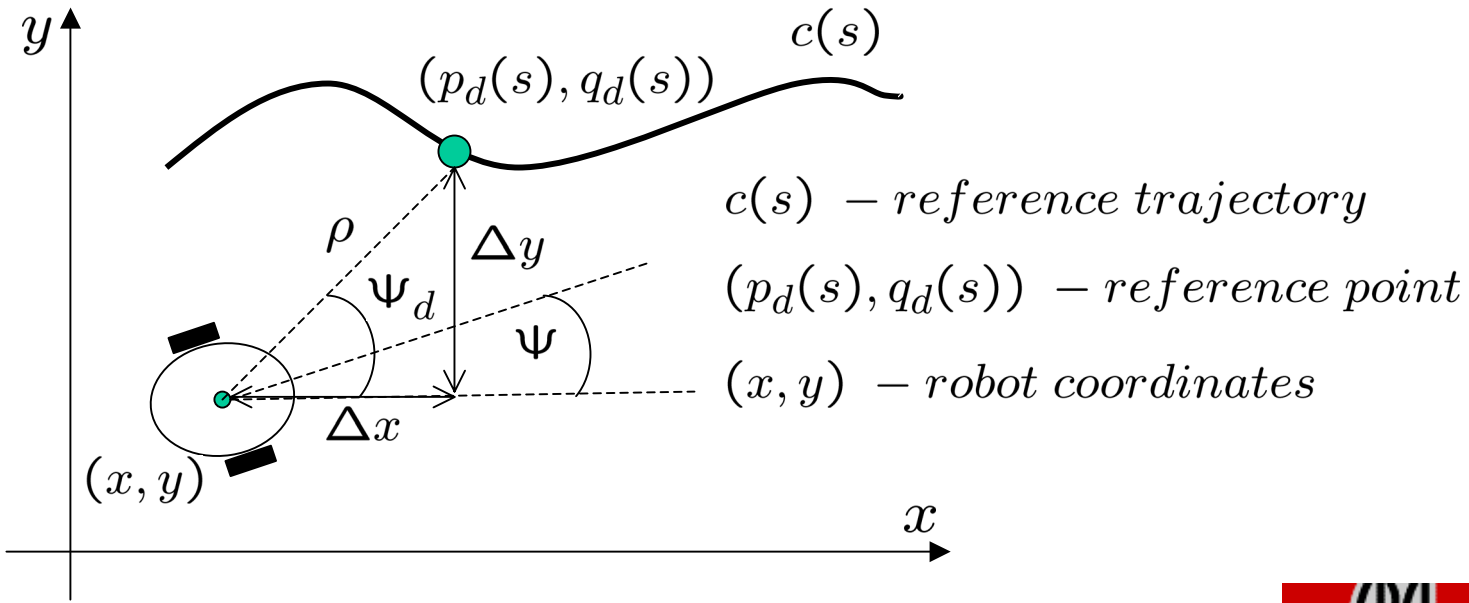
$$\begin{aligned} x_d &= p(s) \\ y_d &= q(s), (0 \leq s \leq s_f) \end{aligned}$$

where $\|c'(s)\| = \sqrt{p'^2 + q'^2} \neq 0, \forall s \in [0, s_f]$ (valid for unit speed curves).

Control objective

- Control objective: keep longitudinal and lateral difference within specific bounds:

$$\limsup_{t\to\infty} \rho_t \leq d_\rho$$
$$\limsup_{t\to\infty} \|\psi - \psi_d\| \leq d_\psi$$



Reference point dynamics

- Reference point acts as a virtual vehicle whose dynamics is governed by the tracking error feedback (*[Egerstedt et al., 2003]*):

$$\dot{s} = \frac{c_o e^{-\alpha\rho} v_n}{\sqrt{p'^2(s) + q'^2(s)}}$$

- Steady state conditions:

$$\rho(t) = d_\rho,\ v(t) = v_n,\ \Psi(t) = \Psi_d$$

- Ensures a globally stable path following with exponential tracking error decay (both positional and angular)

Control law

- Both translational and lateral control are proportional regulators:

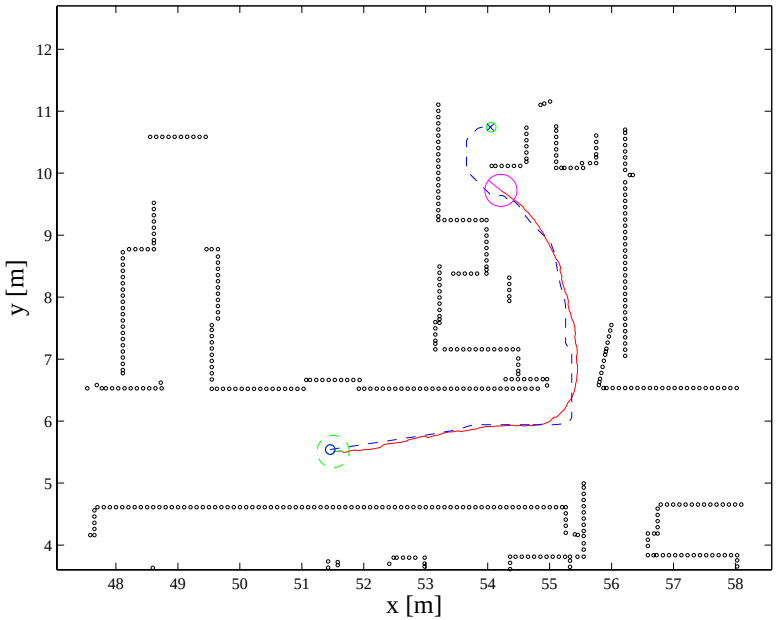
$$\begin{aligned} \omega_{ref} &= k\Delta\psi + \dot{\psi}_d \\ v_{ref} &= \gamma\rho cos(\Delta\psi) \end{aligned}$$

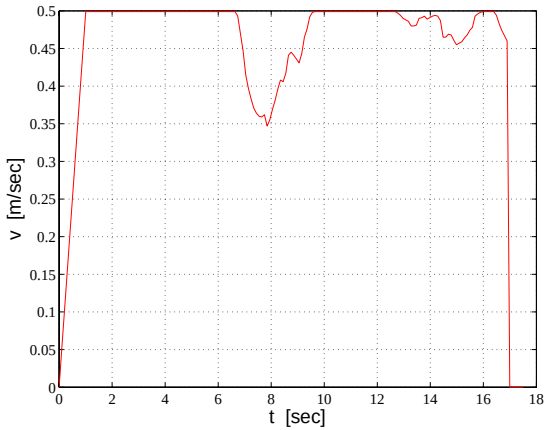
where $\Delta\Psi = \Psi_d - \Psi$.

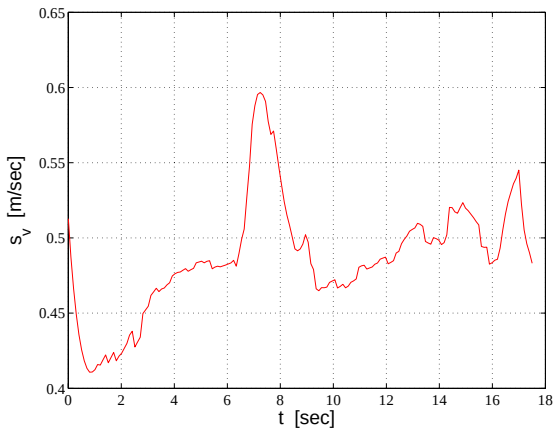
- It is assumed that the actuator level controllers are implemented – higher level velocity control that is model independent.

Off-line planned path run

- Nominal translational velocity maintained







Major drawbacks

- Essentially an off-line path following approach
- Dynamics of the reference point dependent only on tracking error feedback - local path configuration taken into account only at the instantaneous reference point position.

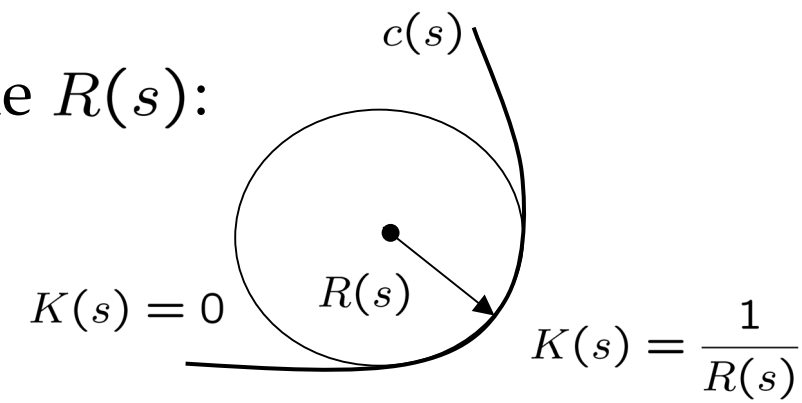
Modified virtual vehicle approach

- Keypoint: local curve configuration between the robot and reference point position included in the reference point dynamics
- Curvature $\kappa(s)$ describes how much a curve $c(s)$ bends locally:

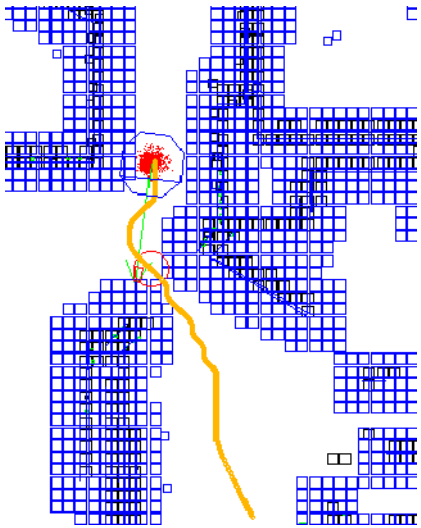
$$\kappa(s) = \frac{p'q'' - p''q'}{(p'^2 + q'^2)^{\frac{3}{2}}}$$

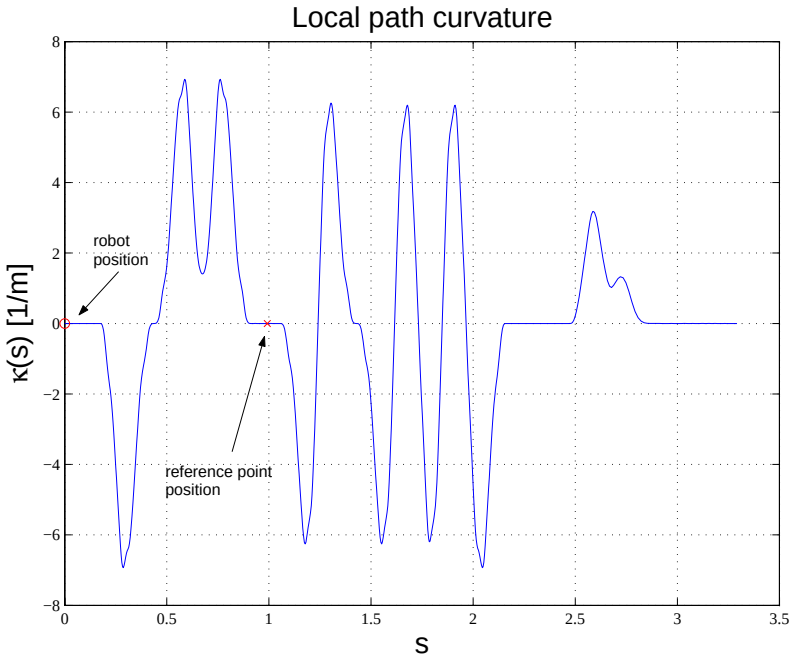
- Relation to the osculating circle $R(s)$:

$$\kappa(s) = \frac{1}{R(s)}$$



Curvature along the global path





- Inverse model tracking: current robot trajectory curvature $\kappa_r = \frac{\omega}{v}$ equals the path curvature κ_c.

Curvature effort

- For a smooth velocity transition the change in curvature must be considered - curve smoothness
- Curvature effort measure describes the local path curvature changes between the robot and reference point:

$$\chi(\rho) = \frac{\sum_{i=1}^{N_\rho} \|\Delta\kappa_i\| \Delta s_i}{\rho}, \ \forall \rho > 0$$

with $\{(x_c(s), y_c(s)) : s = (1, 2 \ldots, N_\rho)\}$ a collection of dense waypoints along the reference curve

- Differences: $\Delta\kappa_i = \kappa_i - \kappa_{i-1}$, $\Delta s_i = s_i - s_{i-1}$.

Modified reference point dynamics

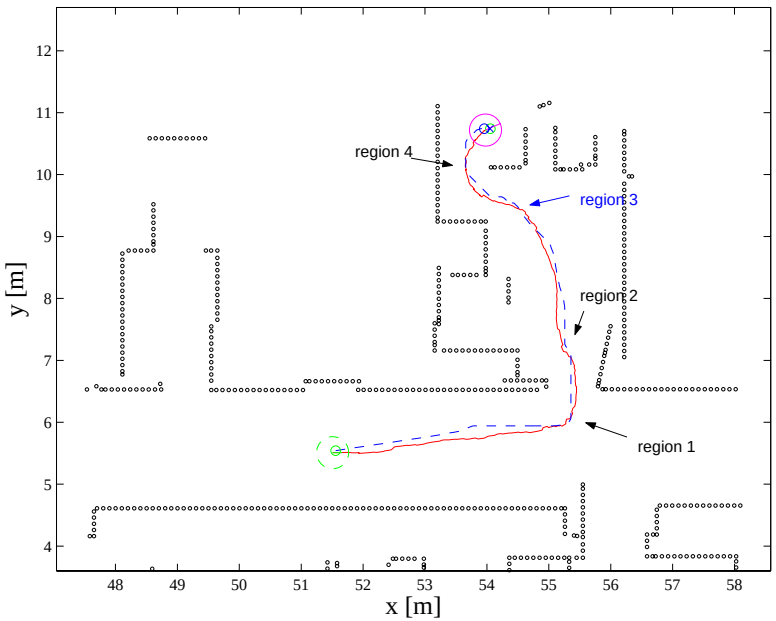
- When significant curvature changes are encountered along the local path, robot must slow down to follow the path at a safe margin
- Translational velocity of the robot is directly influenced by modified reference point dynamics:

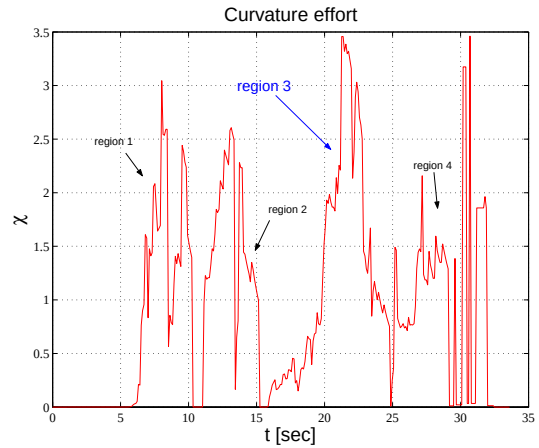
$$\dot{s} = \frac{c_o e^{-\alpha\rho} v_n}{\sqrt{p'^2(s) + q'^2(s)}} \left[1 - \frac{2}{\pi} atan(\zeta\chi(\rho))\right]$$

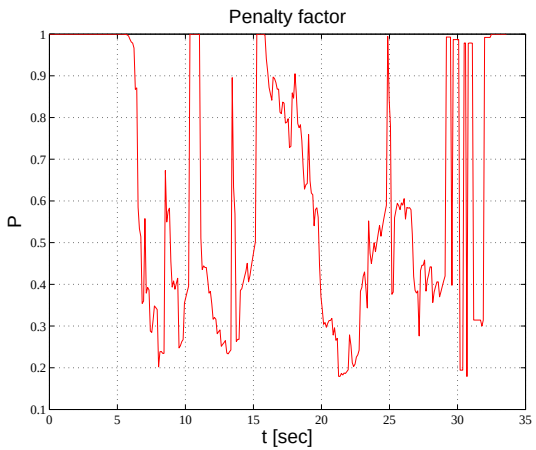
- Penalty factor $P(\rho) \in [0, 1)$:

$$P(\rho) = 1 - \frac{2}{\pi} atan(\zeta\chi(\rho))$$

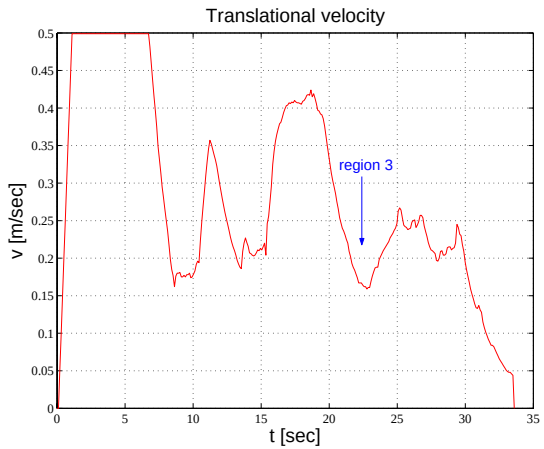
A modified virtual vehicle run (1)

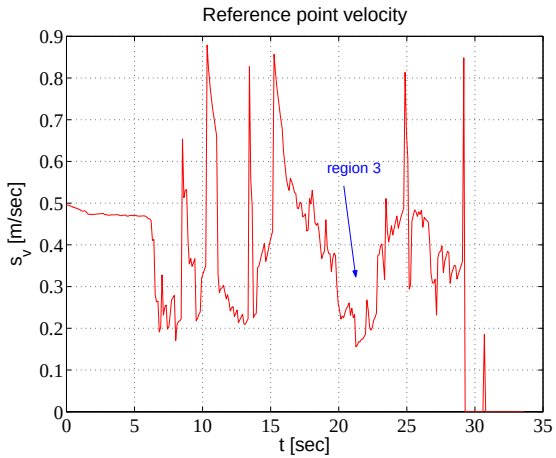


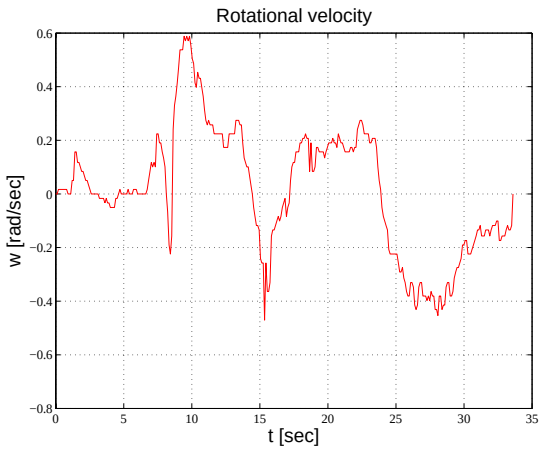


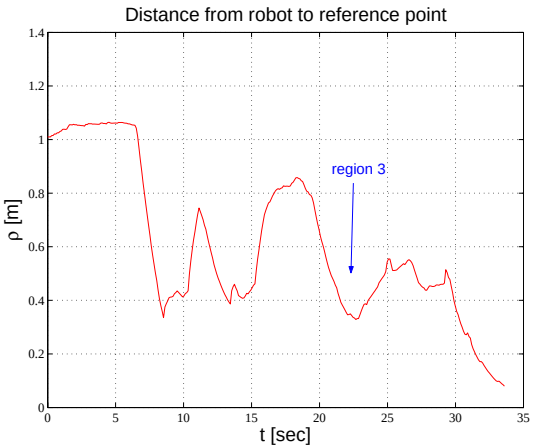


A modified virtual vehicle run (2)







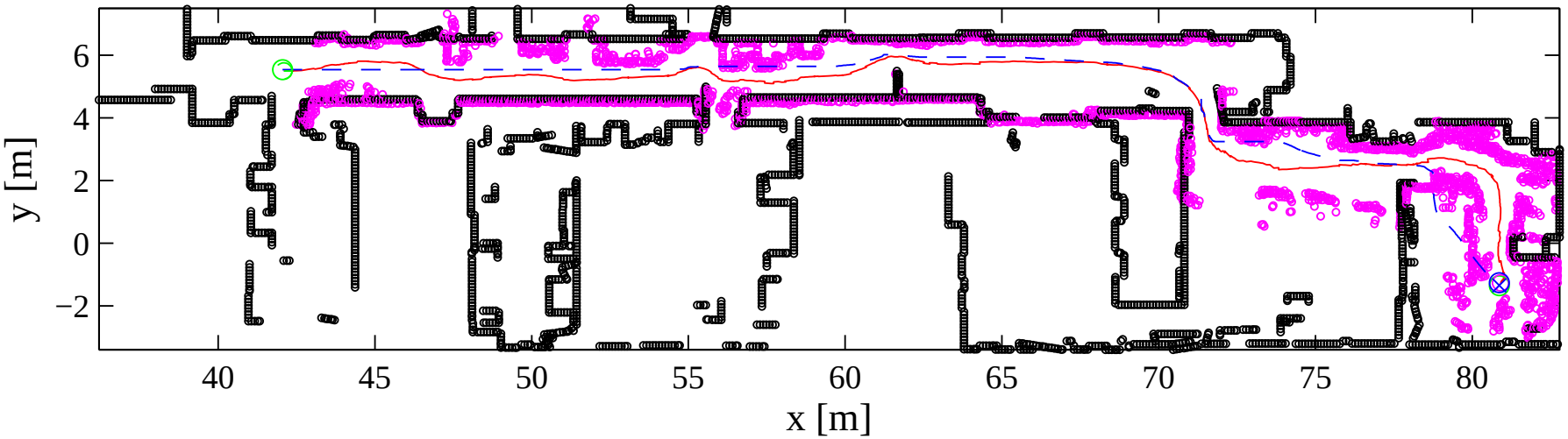


Incremental path build-up

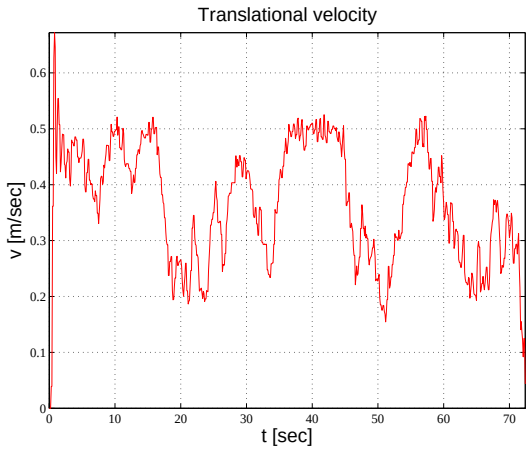
- On-line replanning based on local subgoal points
- Local curve characteristics and form changes (curve parameter values different) between two successive steps - artificial reference point oscillations could occur
- Solution - one step simulation of both robot and reference point motion
- Localization problem

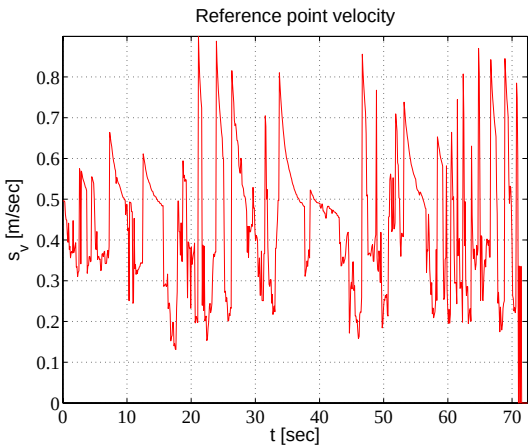
Experimental results – Modified Virtual Vehicle (1)

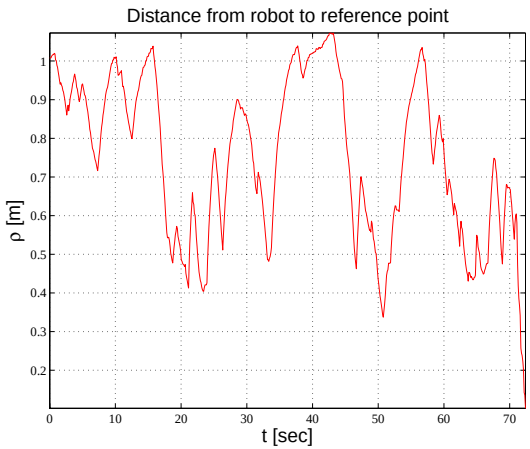
- Results comparable to commonly used Gradient navigation method (*[Konolige et al., 2003]):*

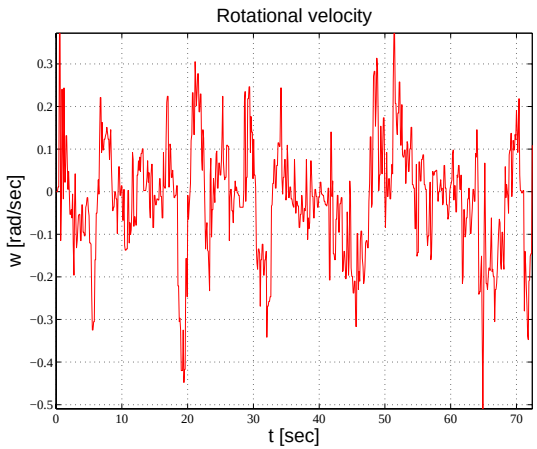


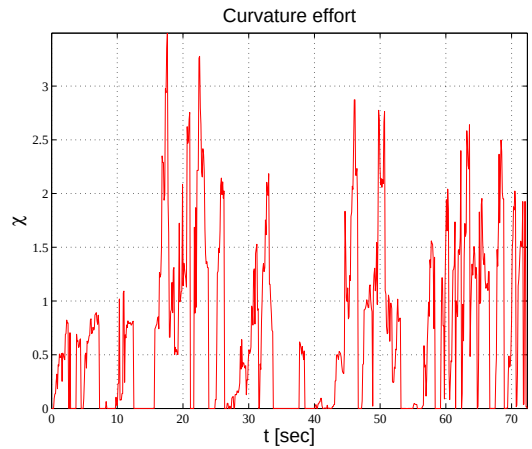
Experimental results – Modified Virtual Vehicle (2)

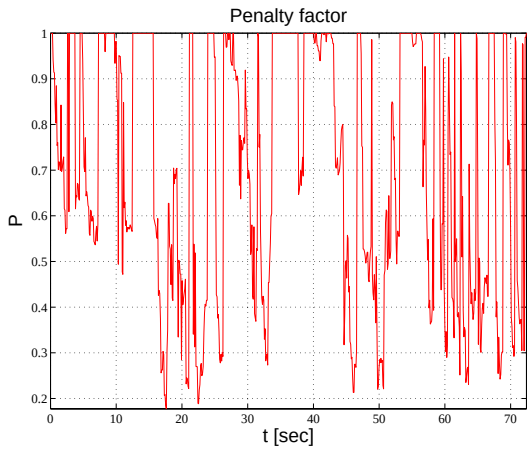












Conclusions

- Lane detection vision module for driver assistance based on a probabilistic framework
- Off-line path planning based on curvature change and safety distance optimality criteria
- On-line path planning based on graph search A* and D* algorithms on an occupancy grid map
- Global path integration method with Dynamic Window obstacle avoider for motion control
- Modified virtual vehicle approach to motion control with on-line global smooth path replanning scheme

THANK YOU!