Vers une formalisation d'un système de suivi de trajectoires

Alexandre Chapoutot

ENSTA ParisTech

DGA MRIS meeting
June 28, 2016

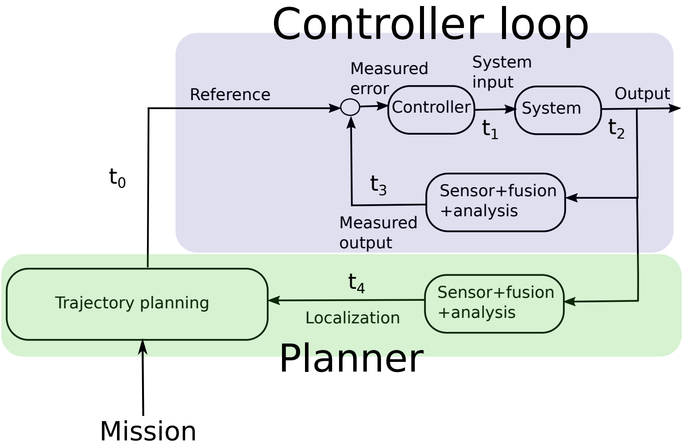
Disclaimer

This presentation

- borrows material from Magnus Egerstedt (Georgia Tech)
- comes from discussion with Julien Alexandre dit Sandretto, Emmanuel Battesti, David Filliat, François Pessaux, Olivier Mullier.
- is based on some part of material produced by Julien Alexandre dit Sandretto

In other terms no much idea really comes from me :-)

Autonomous vehicle



Goal try to understand main pieces of the system to validate their behavior and the behavior of the overall system.

Introduction - cont'

Heterogeneous components

System
: model of the vehicle, possibly with models of actuators
Various kinds of models more or less abstracted from the reality

Controller
: shall put the system into a given configuration (*e.g.*, position, orientation)
Many algorithms: PID, MPC, optimal controller, etc.

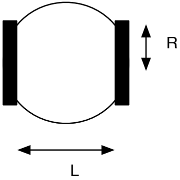
Sensor+fusion+analysis
: data centred algorithms to produce pertinent information about the system, *e.g.*, speed, position, etc.
Note: information may be incomplete/perturbed so need of observer methods or filters

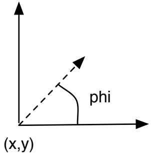
Trajectory planing
: from a given mission, try to compute a path (optimal or not)
Many possibilities: depending on the availability of a map or not, if there are some obstacles (static or dynamic) and so on

Cinematic of a Robot 2D

Various way of modelling the dynamic of a robot, mainly from Physics law, *e.g.*, Newton 2nd law

Example of a differential drive robot





$$\dot{x} = \frac{R}{2}(v_r + v_\ell)\cos\phi$$
$$\dot{y} = \frac{R}{2}(v_r + v_\ell)\sin\phi$$
$$\dot{\theta} = \frac{R}{L}(v_r - v_\ell)$$

Cinematic of a Robot 2D – abstraction

A common basis for a two wheel robots

$$\dot{x} = v\cos(\theta) \tag{1}$$
$$\dot{y} = v\sin(\theta) \tag{2}$$
$$\dot{\theta} = \omega \tag{3}$$

with possible constraints

- Unicycle $v \in [-1, 1]$ and $\omega \in [-\pi, \pi]$
- Dubins $v = 1$ and $\omega \in [-\pi, \pi]$

Note need of a relation between this abstraction and the more realistic model (*i.e.*, a link with actuators)

Example

$$v_r = \frac{2v + \omega L}{2R} \qquad \text{and} \qquad v_\ell = \frac{2v - \omega L}{2R}$$

More abstracted dynamics in 2D

Some simpler models can also be used, in particular, during the trajectory planning.

More precisely, the dynamics of particle is described by

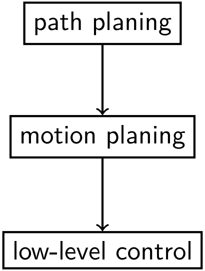
$$\dot{x} = u$$

with $x \in \mathbb{R}^2$.

We assume hence that we can control directly the position and the speed of a vehicle.

Note: u represents a trajectory that the particle has to follow

A hierarchical control



- **Path planing** generates a set of way points (does not take into account the dynamics of the vehicle) from a map (totally or partially) known, take into account obstacles (static)
- **Motion planing** generates a set of trajectories feasible for the dynamics considered and take into account obstacles (static and dynamic)
- **Low-level controller** tries to follow the (discretized) trajectory w.r.t. the dynamic of the vehicle

General scheme for algorithms

Controller or trajectory planner follow the main loop algorithm

```
while true do
   read sensors
   compute function with constraints/properties to respect
   write output
done
```

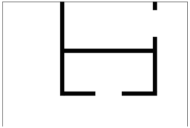
Notes:

- read sensors: shall consider uncertainties or noise
- apply function: shall respect properties (as stability, real-time, etc.) **but** properties differ between controller and trajectory planner

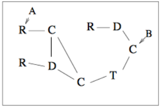
Path planning

From a (discrete) map, *i.e.*, a (weighted) graph,

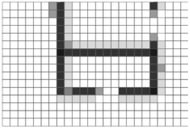
Real word



Topological map



Occupancy grid



Goal generates path according to the mission and the initial starting point.

Properties (?)

- Prove the existence or not of a path w.r.t. some constraints, **e.g.**, forbidden area, check points, etc.
- Optimize criteria, *e.g.*, time, fuel consumption, etc.

Algorithms A^*, RRT, Interval-based search, etc.

Motion planing

Goal from a list of way points, generate trajectory that the vehicle can follow while avoiding obstacles.

May use a simple model of dynamic such as **a particle** $\dot{x} = u$

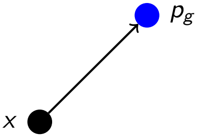
Main behaviors that compose a motion planner

- Go To Goal: from a given initial position and a final position F, generates a trajectory t for which the vehicle can reach F.
- Obstacle avoidance: the trajectory t shall avoid obstacles

Challenge: combine these behaviors to make the vehicle go to goal safely.

Motion planing – Go to goal

A particle $\dot{x} = u$ at position x shall reach position p_g



with

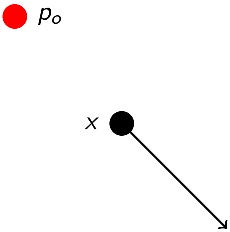
$$e = p_g - x$$

then we can define a control such that

$$u = -Ke \qquad \text{with} \quad K > 0$$

Motion planing – Obstacle avoidance

A particle $\dot{x} = u$ at position x shall avoid position p_o



with

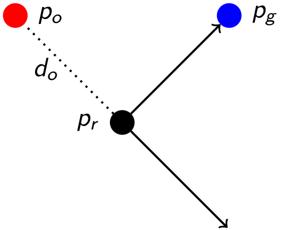
$$e = x - p_o$$

then we can define a control such that

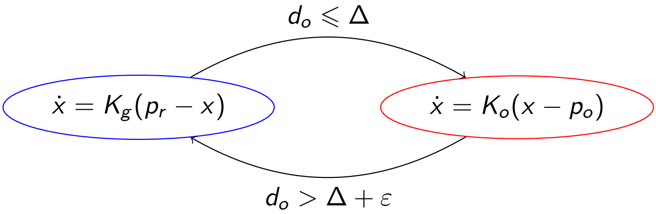
$$u = Ke \qquad \text{with} \quad K > 0$$

Motion planing – Combination of behaviors

A particle $\dot{x} = u$ at position x shall reach position p_g while avoid position p_o

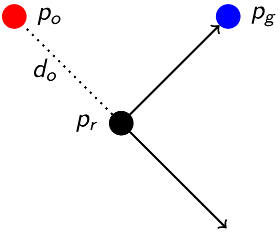


Note: different strategies can be used (hard vs blend behaviors)



Motion planing – Combination of behaviors

A particle $\dot{x} = u$ at position x shall reach position p_g while avoid position p_o



Note: different strategies can be used (hard vs blend behaviors)

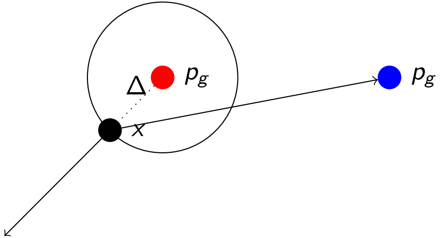
With σ a blending function in $[0, 1]$ we can define

$$\dot{x} = \sigma(d_o)K_g(p_r - x) + (1 - \sigma(d_o))K_o(x - p_o)$$

Note we can loose convergence

Motion planing – Combination of behaviors

An other solution of combine behaviors using sliding mode



Define a switching surface such that

$$g(x) = \frac{1}{2}\left(\| x - x_O \|^2 - \Delta^2\right) = 0$$

considering two functions:

$$f_1 = K_g(p_g - x)$$
$$f_2 = K_O(x - p_O)$$

Motion planing – Combination of behaviors

The induced mode is a convex combination such that

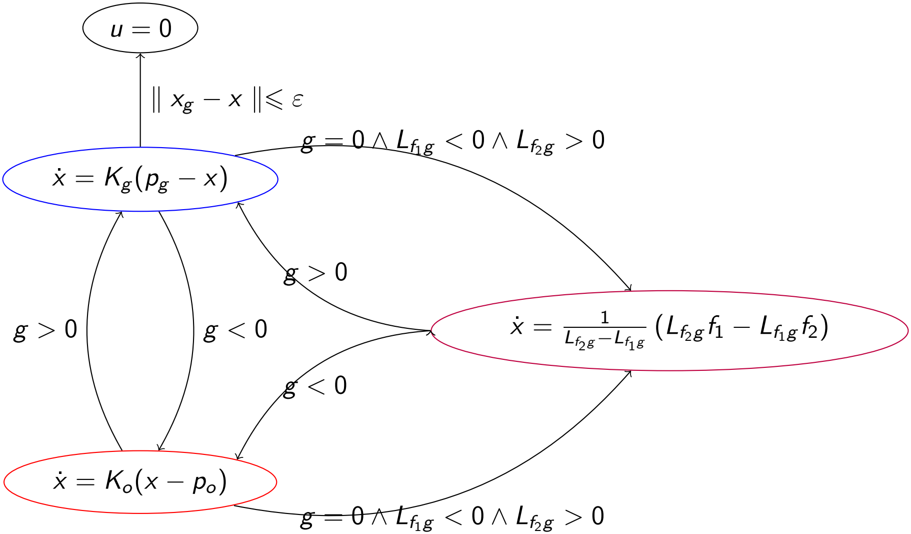
$$\dot{x} = \frac{1}{L_{f_2 g} - L_{f_1 g}} \left(L_{f_2 g} f_1 - L_{f_1 g} f_2 \right)$$

with L_{fg} the Lie derivative of g along f *i.e.*, $\frac{\partial g}{\partial x} f$

$$\frac{\partial g}{\partial x} = (x - x_O)^T, \qquad L_{f_1 g} = K_g (x - p_O)^T (p_g - x), \qquad L_{f_2 g} = K_o \parallel x - p_o \parallel^2$$

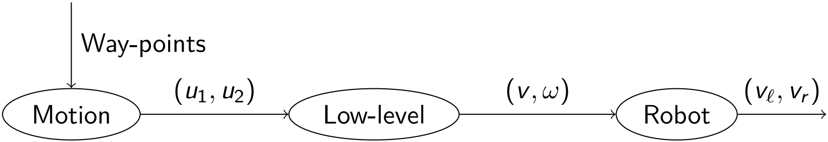
Note induced method can get rid of bump behavior

Motion planing – Combination of behaviors



Properties to prove (?): safety, no deadlock, reachability, etc.

Connecting motion planing and low-level controller



If the trajectory reference is given by $u = (u_1, u_2)$ we know that

$$\phi_d = \tan\left(\frac{u_1}{u_2}\right)$$

then

$$\begin{aligned} e' &= \arctan 2(\sin(e), \cos(e)) \qquad \text{with} \quad e = \phi_d - \phi \\ \omega &= PID(e') \\ v &= \sqrt{u_1^2 + u_2^2} \end{aligned}$$

Properties to prove (?):

Conclusion

- Presented a small example of autonomous vehicle
- Shew some algorithms in the control hierarchy

Next

- Instantiate on a more realistic vehicle
- Define properties we wan/can prove
- Model this system in an appropriate language

Under development

- DynIBEX and contractor on tubes and predicate on tubes (Julien Alexandre dit Sandretto)
- Extension to *n*-dimensional case of Dominique Monnet's implementation for viability computation (Olivier Mullier)
- Combining OpenSMT2 and DynIBEX $\Rightarrow$ SMT modulo ODE (Robin Morier)