Tiji, a Generic Time-constrained Trajectory Generation Scheme

Vivien Delsart & Thierry Fraichard
vivien.delsart@inrialpes.fr *thierry.fraichard@inria.fr*
UNIVERSITÉ DE GRENOBLE

Problem

Goal
- Determining a trajectory reaching a goal state with a final time constraint
- Complex dynamic systems
- Real-time performance

Challenges
Fixed final time
⇒ Reachability problem:
- No guarantee that a solution exists
- How to find a trajectory that ends as close as possible to the goal?

Applications
Integrated into:
- Trajectory tracking
- Trajectory deformation

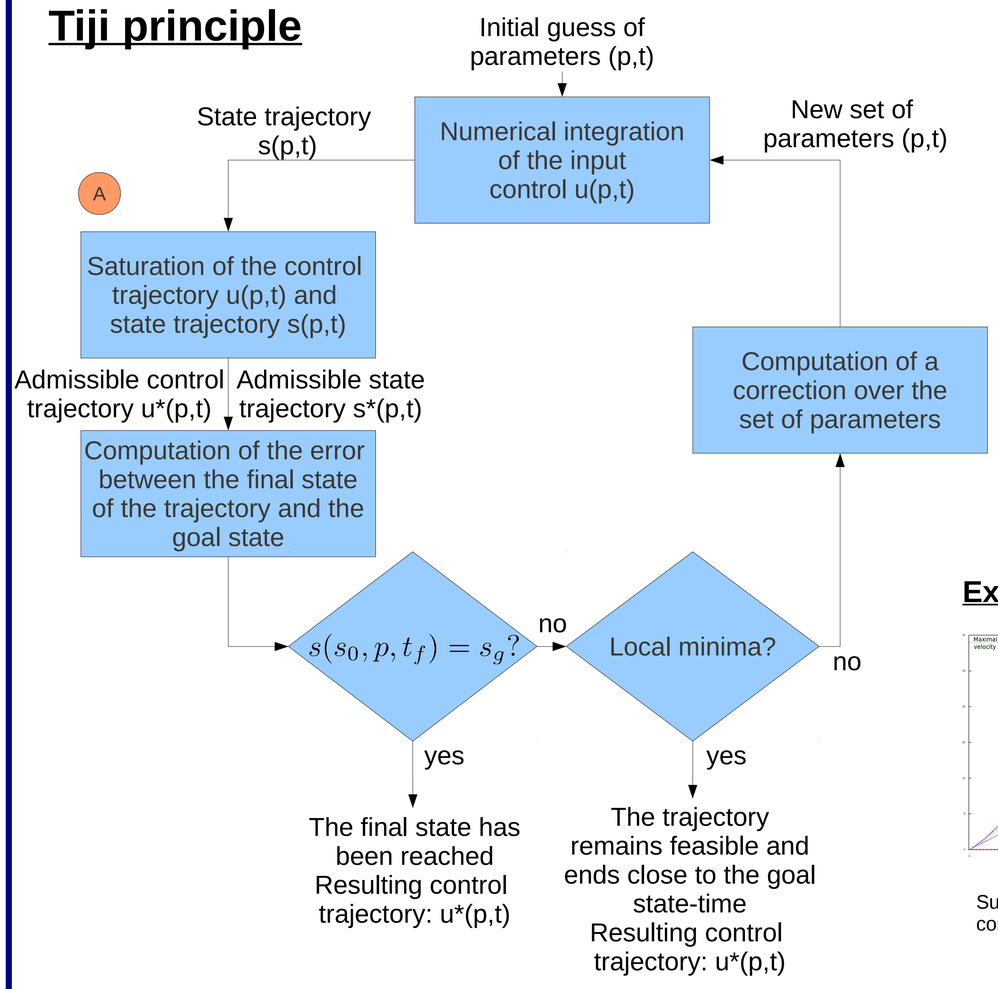
Trajectory Generator (Tiji)

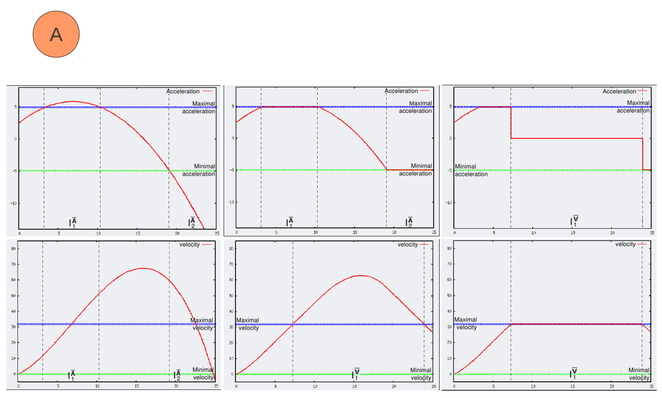
Use of a variational approach
- Parametrization of the input control $u(t) = u(p,t)$

- Switch from optimal control to constrained optimization problem:

$$\begin{array}{lll} \text{Find:} & u(p,t),\ \forall t \in [0,t_f] & \\ \text{Minimizing:} & J = \|s(s_0,p,t_f) - s_g\|^2 & \text{(weighted distance)} \\ \text{subject to:} & h_u(u) \leq 0 & \text{(control boundary constraints)} \\ & h_s(s) \leq 0 & \text{(state boundary constraints)} \\ & t_{min} \leq t_f \leq t_{max} & \text{(final time constraint)} \end{array}$$

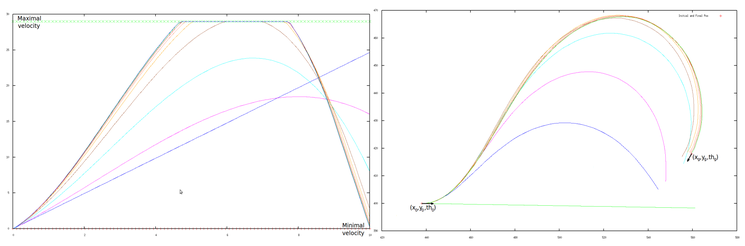
Tiji principle





Successive saturation of acceleration and velocity profiles of a robotic system to provide a trajectory that respects all the motion constraints of the system

Example



Successive trajectory profiles (path and velocity) of a robot after application of corrections over the set of parameters defining its control.

Applications on an Automated Wheelchair

State: $s = (x, y, \theta, v, \omega)$
Control: $u = (a, \mu)$

System dynamics:
$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ a \\ \mu \end{pmatrix}$$

System constraints:
$\omega \in [-\omega_{max}, \omega_{max}]$
$v \in [-v_{max}, v_{max}]$

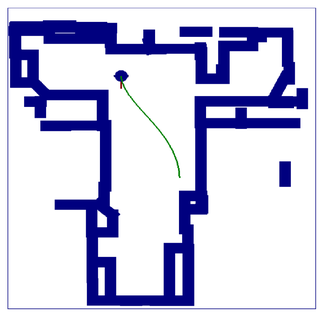
The automated wheelchair

Wheelchair features:
0.67m long
0.56m wide
$\omega_{max} = 1.5 rad/s$
$v_{max} = 1.39 m/s$
Sick laser scanner
→ FoV : 180 deg
→ resolution : 0.5 deg
→ range : 20m

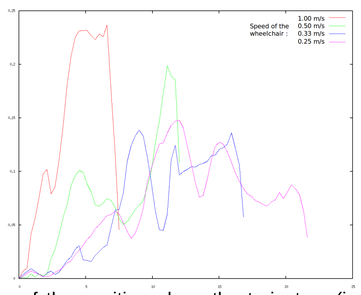
Trajectory tracking

Problem definition: Reaching and following a reference trajectory during the execution of the robot motion

Tiji's interest: Finding the best motion to move close to the trajectory while it is not reachable, and computing a feasible motion to stay on it when it has been reached



Reference trajectory

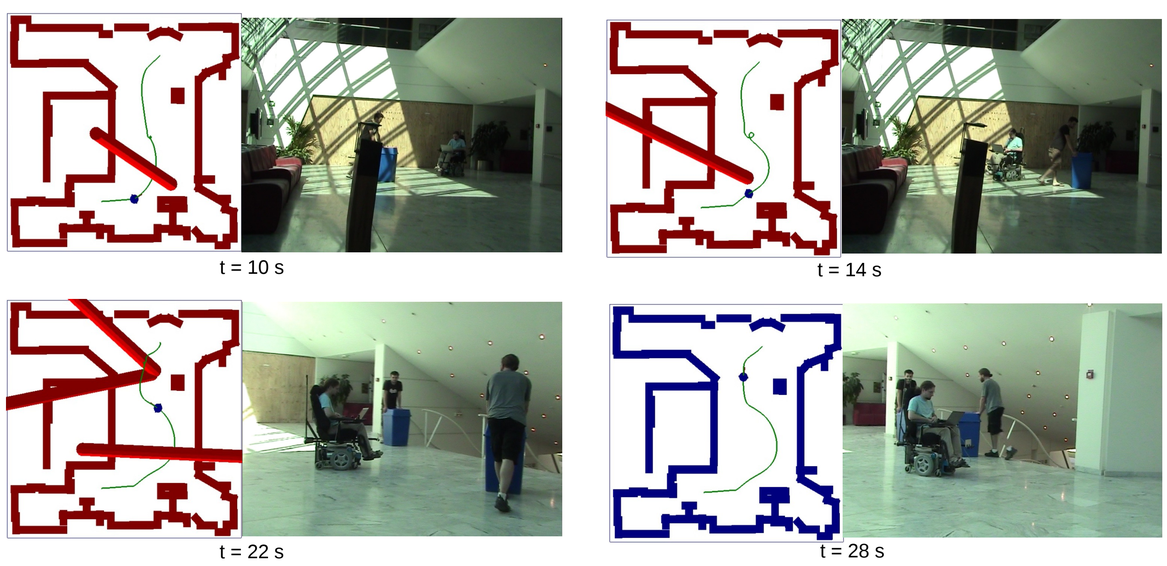


Derivation of the position along the trajectory (in meters) while following the reference trajectory at different speeds

Trajectory deformation

Problem definition: Avoiding obstacles during the motion execution of the robot while keeping a trajectory converging to the goal

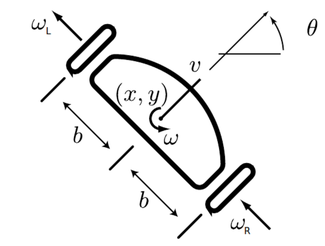
Tiji's interest: Keeping a trajectory respecting the motion constraints after the deformation process



Trajectory deformation process applied on an automated wheelchair navigating amongst several pedestrians. During the motion execution, the trajectory (in green) followed by the robot (in blue) is deformed in response to static obstacles (red polygons) and to a forecast model of the future behavior of dynamic obstacles (red cylinders).

Results

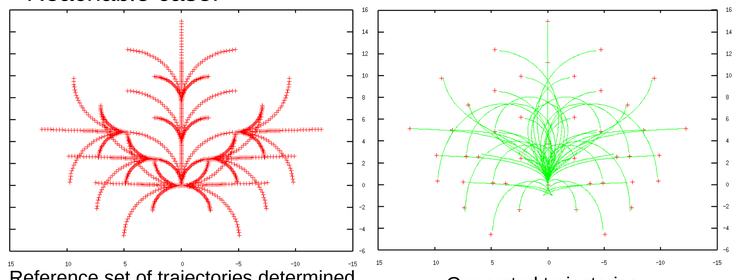
Differential drive



State: $s = (x, y, \theta, \omega_L, \omega_R)$
Control: $u = (\mu_L, \mu_R)$
System dynamics:
$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\omega_L} \\ \dot{\omega_R} \end{pmatrix} = \begin{pmatrix} \frac{\omega_R+\omega_L}{2}\cos\theta \\ \frac{\omega_R+\omega_L}{2}\sin\theta \\ \frac{\omega_R-\omega_L}{2b} \\ \mu_L \\ \mu_R \end{pmatrix}$$

System constraints:
$(\omega_L, \omega_R) \in [-\omega_{max}, \omega_{max}]$
$(\mu_L, \mu_R) \in [-\mu_{max}, \mu_{max}]$

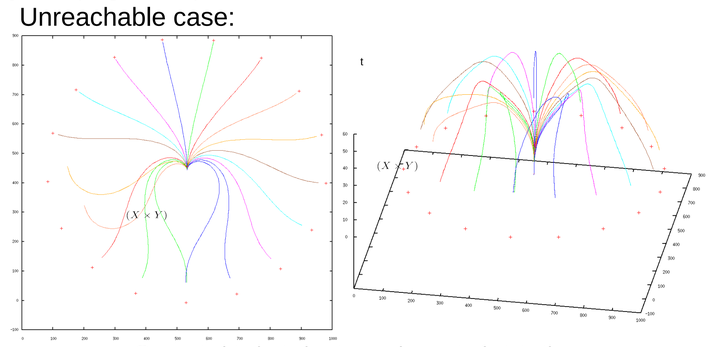
Reachable case:



Reference set of trajectories determined by a sampling of the input control

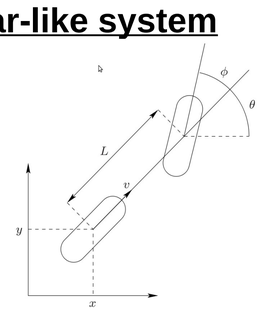
Generated trajectories

Unreachable case:



Generated trajectories to reach a set of state-times (crosses) surrounding the initial position of the robot

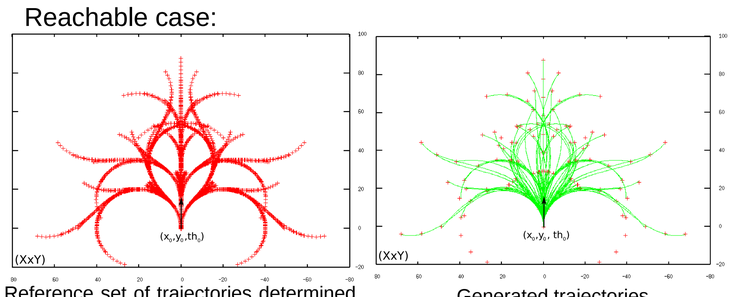
Car-like system



State: $s = (x, y, \theta, v, \omega)$
Control: $u = (a, \zeta)$
System dynamics:
$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} v\cos\theta \\ v\sin\theta \\ v\frac{\tan\phi}{L} \\ a \\ \zeta \end{pmatrix}$$

System constraints:
$v \in [0, v_{max}]$
$a \in [-a_{max}, a_{max}]$
$\phi \in [-\phi_{max}, \phi_{max}]$
$\zeta \in [-\zeta_{max}, \zeta_{max}]$

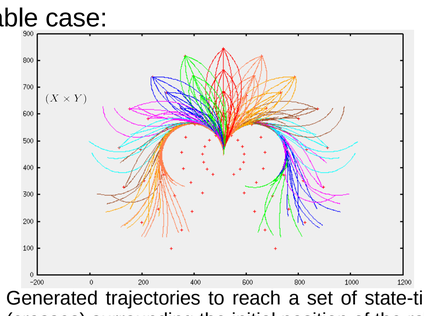
Reachable case:



Reference set of trajectories determined by a sampling of the input control

Generated trajectories

Unreachable case:



Generated trajectories to reach a set of state-times (crosses) surrounding the initial position of the robot

Performances

Implementation in C++ on a desktop computer (Corei7@2.66Ghz, Linux system)

System	Differential drive	Car-like system
Number of evaluated state-times	327680	393216
Success rate (%)	94.58	96.52
Average required steps	6.38	7.08
Average time by trajectory (ms)	4.78	5.12

Main Features

- Trajectory generation with a final time constraint
- Compute a trajectory that reaches the goal when possible and that ends as close as possible to the goal when it is not reachable

Other Possible Applications

- Local / global motion planning
- Multi-robot coordination
- Every obstacle avoidance process trying to anticipate the motion of moving obstacles