Robot Navigation and collision avoidance

Focus on controller

Local Planner DWA

Dynamic Windows Approach (DWA)

Theory and Ros implementation

Objectives

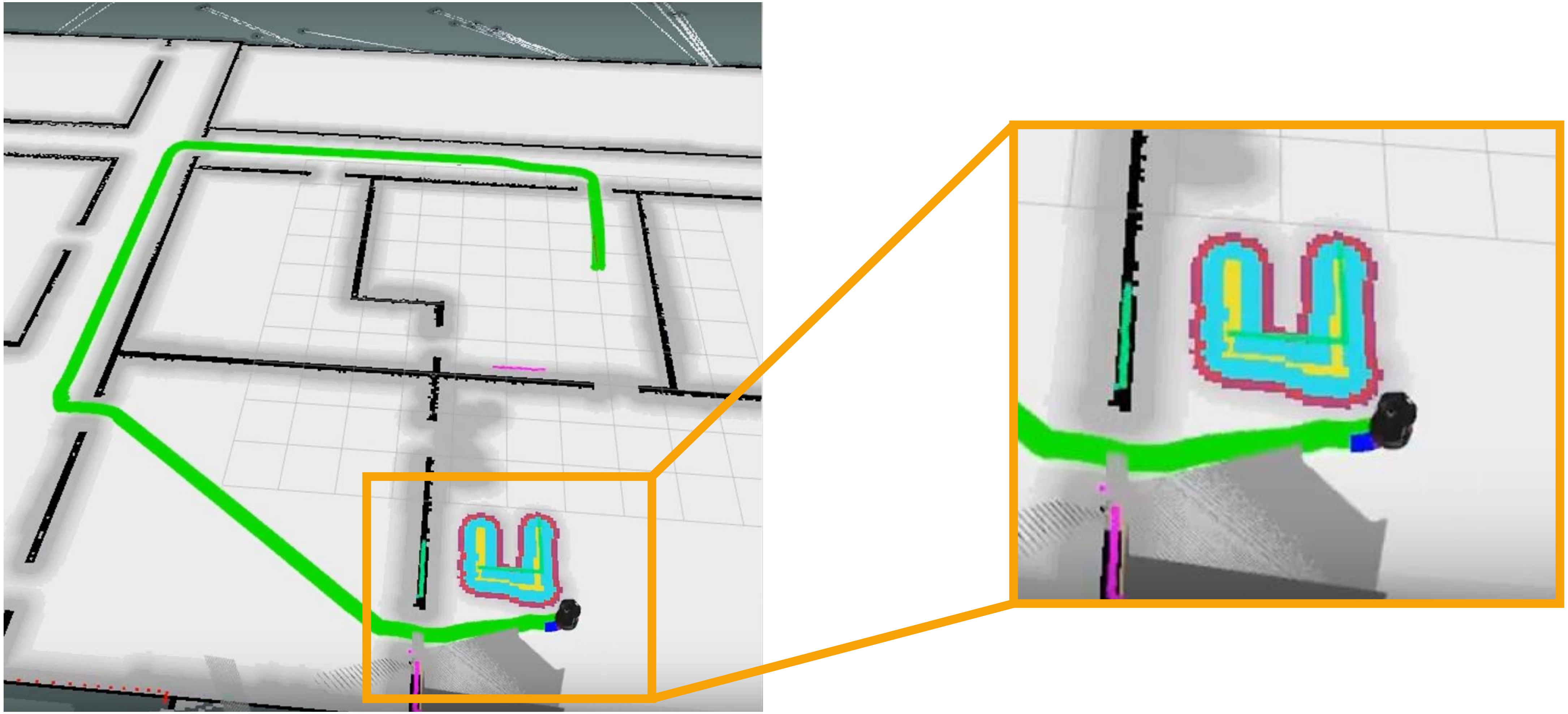
❑ Local Planner / Controller

- **Control robot** to reach goal on a **local environment**
- Send command to robot

❑ Gloal Planner / Planner

- **Build a path** to reach a goal in the **entire environment**
- Send a path (e.g succession of waypoints) to the local planner

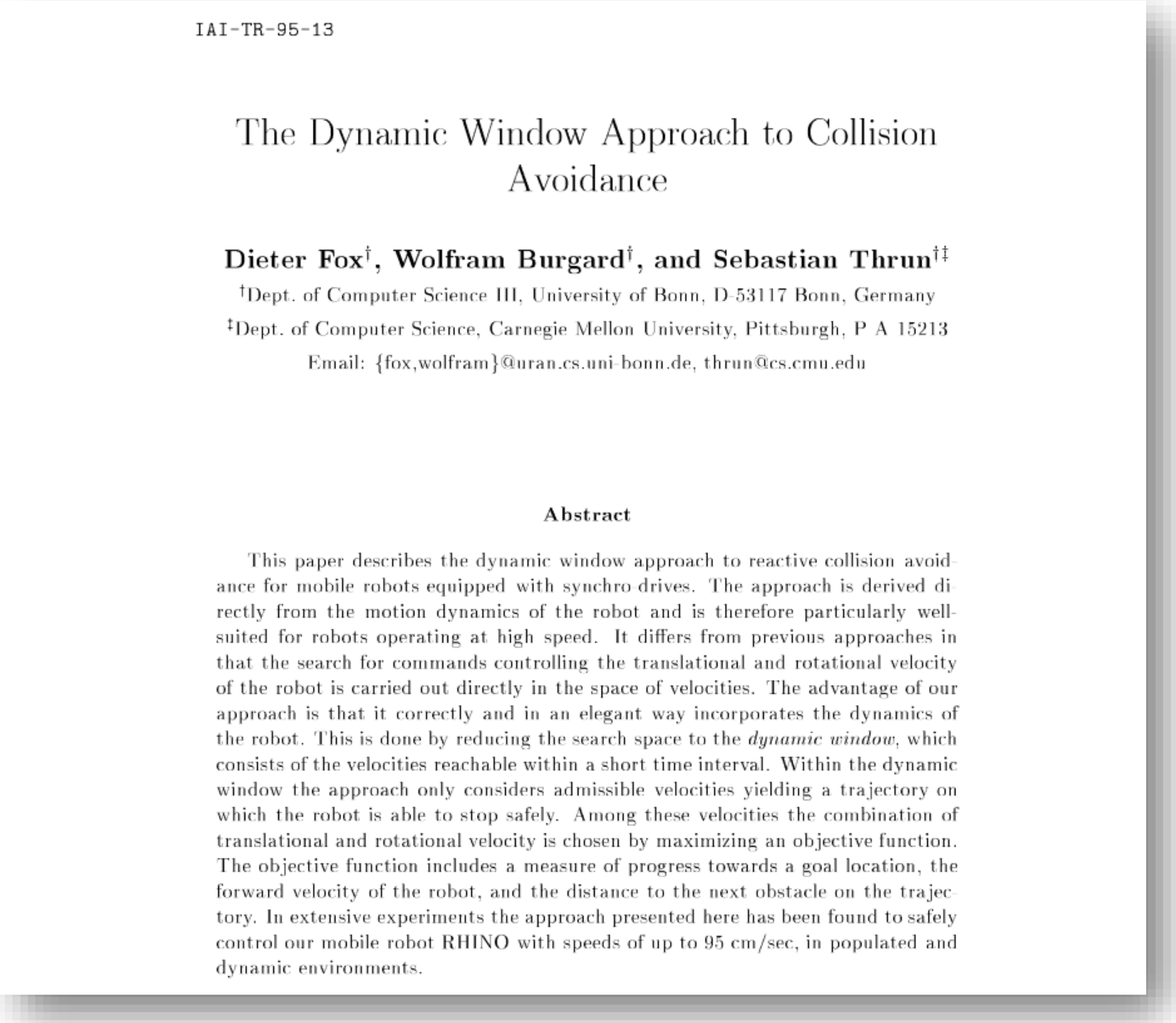
Objectives



Dynamic Window Approach (DWA)

- ❑ Dieter Fox and Co in 1997

- ❑ Mainline
  - ▪ Search best translation and rotation velocities in short interval time
  - ▪ Reduce the search space on admissibles velocities
  - ▪ Optimisation between
    - - Distance to goal
    - - Distance to next obstacle
    - - Foward Velocity



IAI-TR-95-13

The Dynamic Window Approach to Collision Avoidance

Dieter Fox[†], Wolfram Burgard[†], and Sebastian Thrun[†‡]

[†]Dept. of Computer Science III, University of Bonn, D-53117 Bonn, Germany

[‡]Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, P A 15213

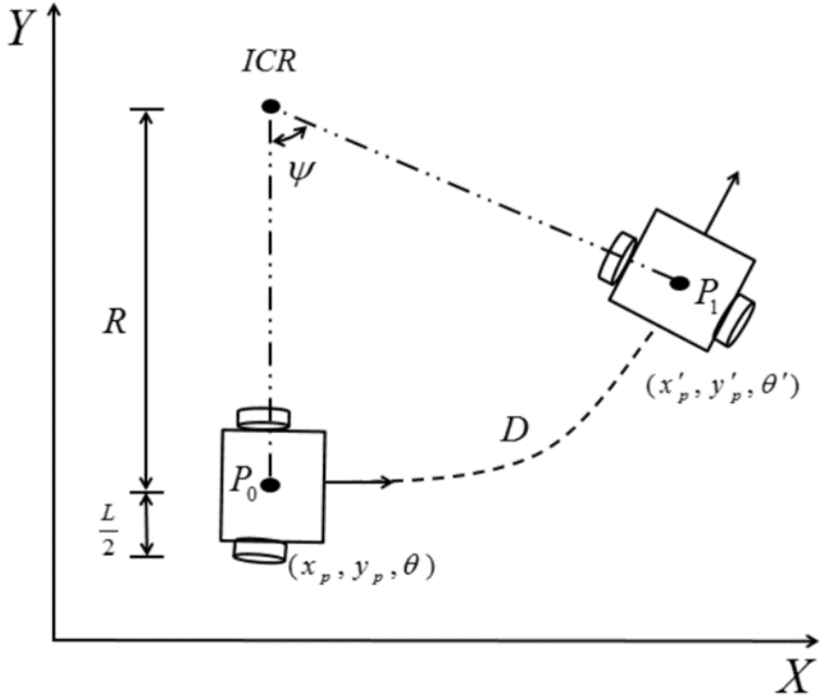
Email: {fox,wolfram}@uran.cs.uni-bonn.de, thrun@cs.cmu.edu

Abstract

This paper describes the dynamic window approach to reactive collision avoidance for mobile robots equipped with synchro drives. The approach is derived directly from the motion dynamics of the robot and is therefore particularly well-suited for robots operating at high speed. It differs from previous approaches in that the search for commands controlling the translational and rotational velocity of the robot is carried out directly in the space of velocities. The advantage of our approach is that it correctly and in an elegant way incorporates the dynamics of the robot. This is done by reducing the search space to the *dynamic window*, which consists of the velocities reachable within a short time interval. Within the dynamic window the approach only considers admissible velocities yielding a trajectory on which the robot is able to stop safely. Among these velocities the combination of translational and rotational velocity is chosen by maximizing an objective function. The objective function includes a measure of progress towards a goal location, the forward velocity of the robot, and the distance to the next obstacle on the trajectory. In extensive experiments the approach presented here has been found to safely control our mobile robot RHINO with speeds of up to 95 cm/sec, in populated and dynamic environments.

Motion Equations

$$
\begin{aligned}
x(t_n) &= x(t_0) + \int_{t_0}^{t_n} v(t) \cdot \cos\theta(t)\, dt \\
y(t_n) &= y(t_0) + \int_{t_0}^{t_n} v(t) \cdot \sin\theta(t)\, dt
\end{aligned}
$$



Tracking Control of Moving Sound Source Using Fuzzy-Gain Scheduling of PD Control, Electronics, Jong-Ho Han 2019

Motion Equations

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} \boxed{v(t)} \cdot \boxed{\cos\theta(t)}\, dt$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t) \cdot \sin\theta(t)\, dt$$

Constant acceleration

$$v(t) = v_0 + \dot{v}.t$$

$$\Delta x = \left(\frac{v+v_0}{2}\right) t$$

$$\Delta x = v_0.t + \frac{1}{2}\dot{v}.t^2$$

Non-constant acceleration

$$v(t) = v_0 + \int_0^t \dot{v}(t)dt$$

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} \boxed{\left(v(t_0) + \int_{t_0}^{t} \dot{v}(\hat{t})\, d\hat{t}\right)} \cdot \boxed{\cos\left(\theta(t_0) + \int_{t_0}^{t} \left(\omega(t_0) + \int_{t_0}^{\hat{t}} \dot{\omega}(\tilde{t})\, d\tilde{t}\right) d\hat{t}\right)} dt$$

$x(t_0)$ *initial pose* $\quad$ $v(t_0)$ *translational vitesse at* t_0 $\quad$ $\omega(t_0)$ *rotational vitesse at* t_0

$\theta(t_0)$ *initial angle* $\quad$ $\dot{v}(t)$ *translational acceleration at* t $\quad$ $\dot{\omega}(t)$ *rotational acceleration at* t

Motion Equations

Constant acceleration

$v(t) = v_0 + \dot{v}.t$

$\Delta x = \left(\frac{v+v_0}{2}\right)t$

$\Delta x = v_0.t + \frac{1}{2}\dot{v}.t^2$

$$
\begin{aligned}
x(t_n) &= x(t_0) + \int_{t_0}^{t_n} v(t)\cdot\cos\theta(t)\,dt \\
y(t_n) &= y(t_0) + \int_{t_0}^{t_n} v(t)\cdot\sin\theta(t)\,dt
\end{aligned}
$$

$$
x(t_n) = x(t_0) + \underline{\int_{t_0}^{t_n}} \left(v(t_0) + \underline{\int_{t_0}^{t} \dot{v}(\hat{t})\,d\hat{t}\Big)} \cdot \cos\left(\theta(t_0) + \underline{\int_{t_0}^{t} \Big(\omega(t_0) + \int_{t_0}^{\hat{t}} \dot{\omega}(\tilde{t})\,d\tilde{t} \Big)} \,d\hat{t} \right) dt
$$

Robot can only be controlled by a finite nb of acceleration cmd

Assuming that between small time interval acceleration is constant

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

Motion Equations

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t) \cdot \cos\theta(t)\,dt$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t) \cdot \sin\theta(t)\,dt$$

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} \left(v(t_0) + \int_{t_0}^{t} \dot{v}(\hat{t})\,d\hat{t} \right) \cdot \cos\left(\theta(t_0) + \int_{t_0}^{t} \left(\omega(t_0) + \int_{t_0}^{\hat{t}} \dot{\omega}(\tilde{t})\,d\tilde{t} \right) d\hat{t} \right) dt$$

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

If Δt is small translation term can be approximated by a velocitory $v_i \in [v(t_i), v(t_{i+1})\,]$

If Δt is small rotational term can be approximated by a velocitory $\omega_i \in [\omega(t_i), \omega(t_{i+1})\,]$

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} \underline{v_i} \cdot \cos\left(\theta(t_i) \underline{+ \omega_i \cdot (\hat{t} - t_i)}\right) d\hat{t}$$

Motion Equations

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t) \cdot \cos\theta(t)\, dt$$

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} v_i \cdot \cos\left(\theta(t_i) + \omega_i \cdot (\hat{t} - t_i)\right) d\hat{t}$$

Integral resolution

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

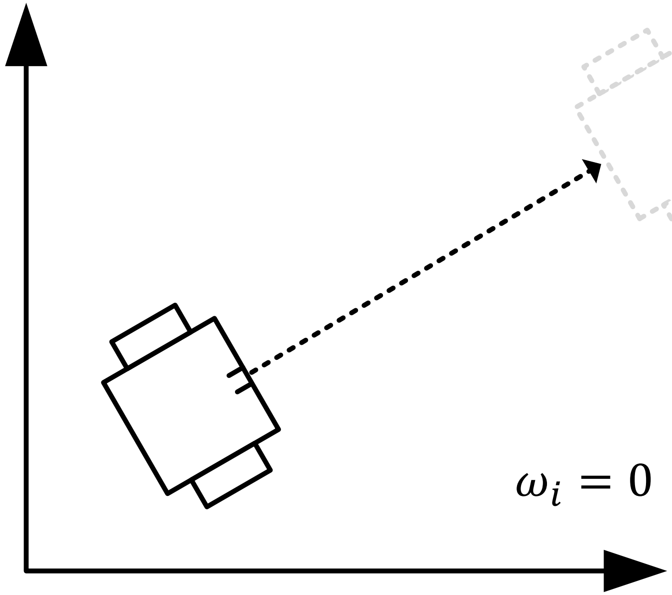
$$F_x^i(t) = \begin{cases} \frac{v_i}{\omega_i}(\sin\theta(t_i) - \sin(\theta(t_i) + \omega_i \cdot (t - t_i))), \; \omega_i \neq 0 \\ v_i \cos(\theta(t_i)) \cdot t, \; \omega_i = 0 \end{cases}$$

Motion Equations

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t)\cdot\cos\theta(t)\,dt$$

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

$$F_x^i(t) = \begin{cases} \frac{v_i}{\omega_i}(\sin\theta(t_i) - \sin(\theta(t_i) + \omega_i\cdot(t - t_i))),\ \omega_i \neq 0 \\ v_i\cos(\theta(t_i))\cdot t,\ \omega_i = 0 \end{cases}$$

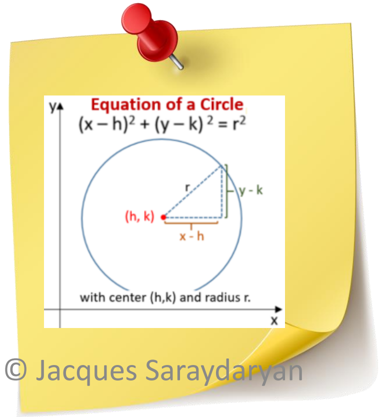


Resulted circle equation :

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

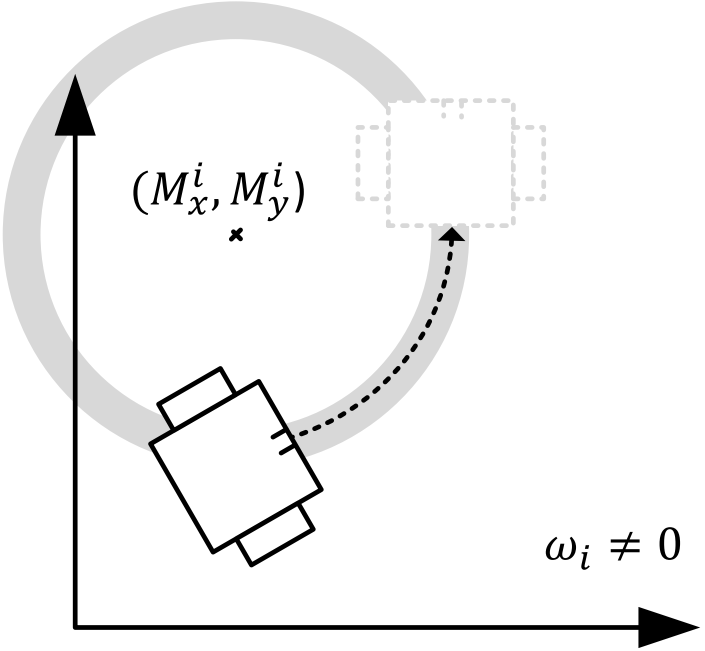
$$M_y^i = \frac{v_i}{\omega_i}\cdot\cos\theta(t_i)$$

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


Equation of a Circle

$(x - h)^2 + (y - k)^2 = r^2$

with center (h,k) and radius r.

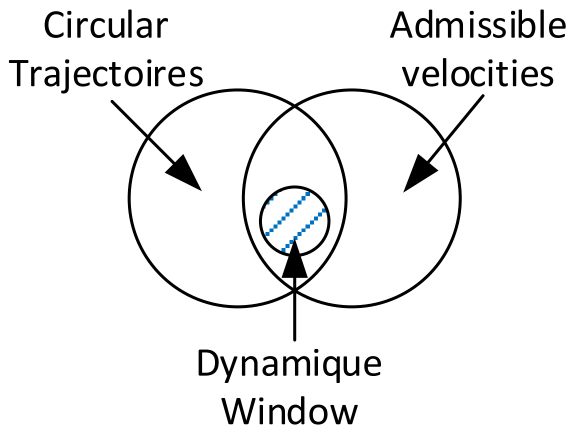


DWA Basic conciderations

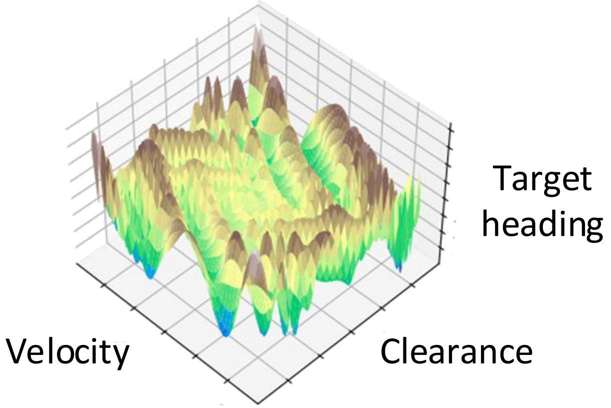
- ❑ Planner needs to determine robot velocites (v_i, ω_i) for the next time interval

- ❑2 Main steps:
  - Search Space
    - Evaluation and select set of possibles translational and rotational velocities
  - Velocities selection
    - Multi factors optimisation
      - Distance to objective
      - Distance to obstacle
      - Velocity

Space Search

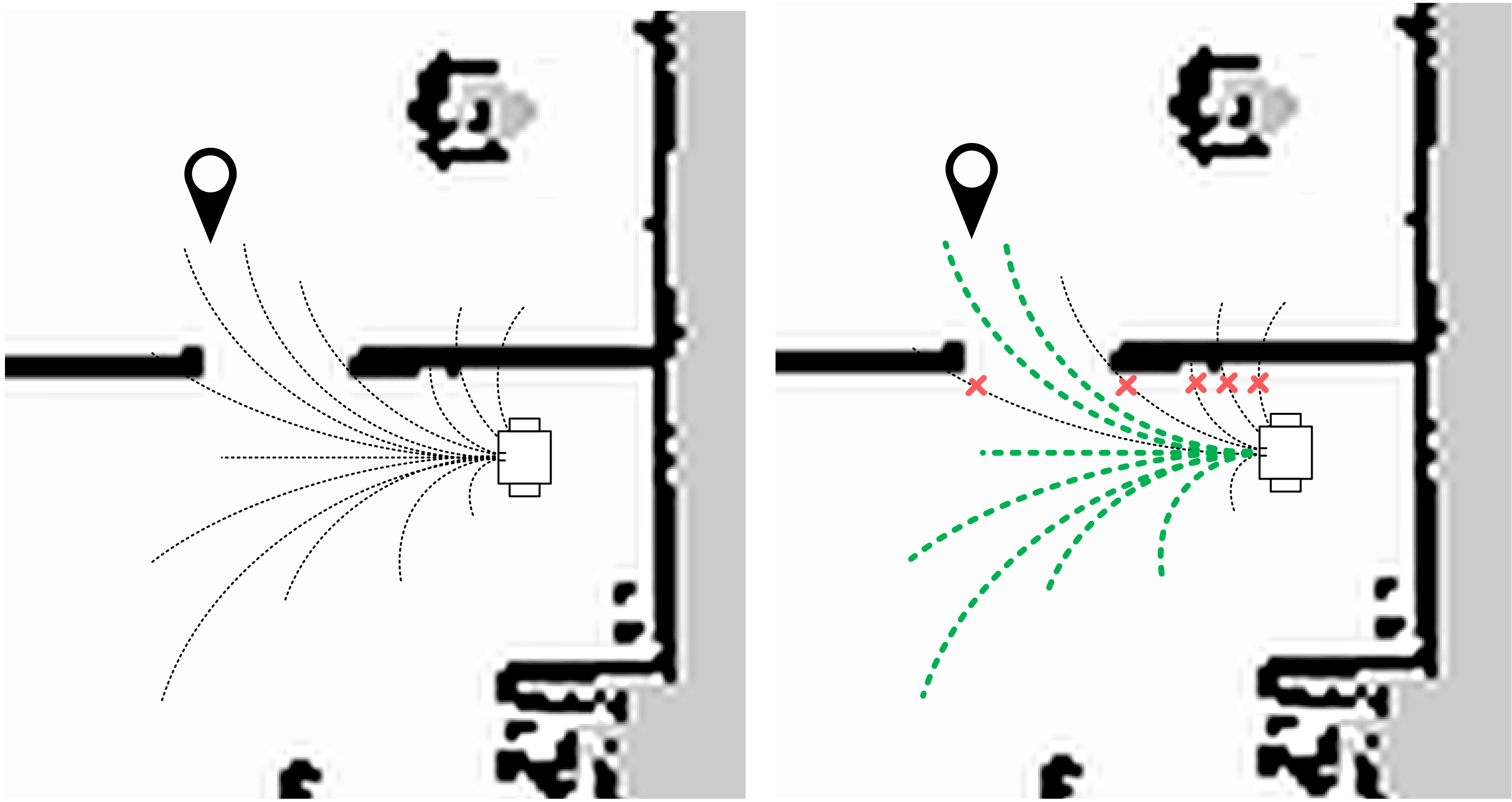


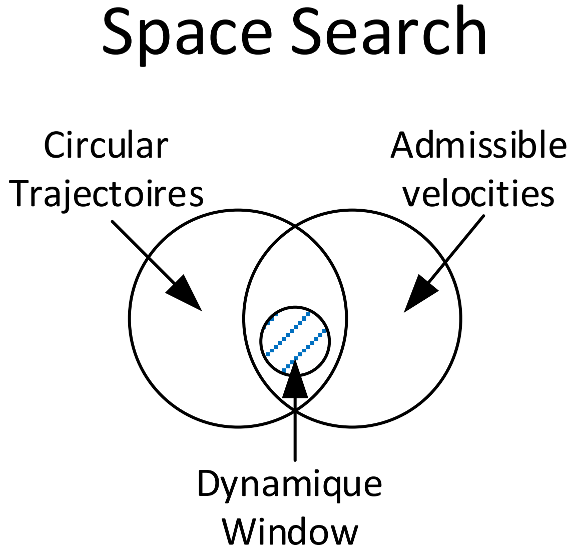
Optimization



DWA Space Search

- ❑ Circular Trajectories

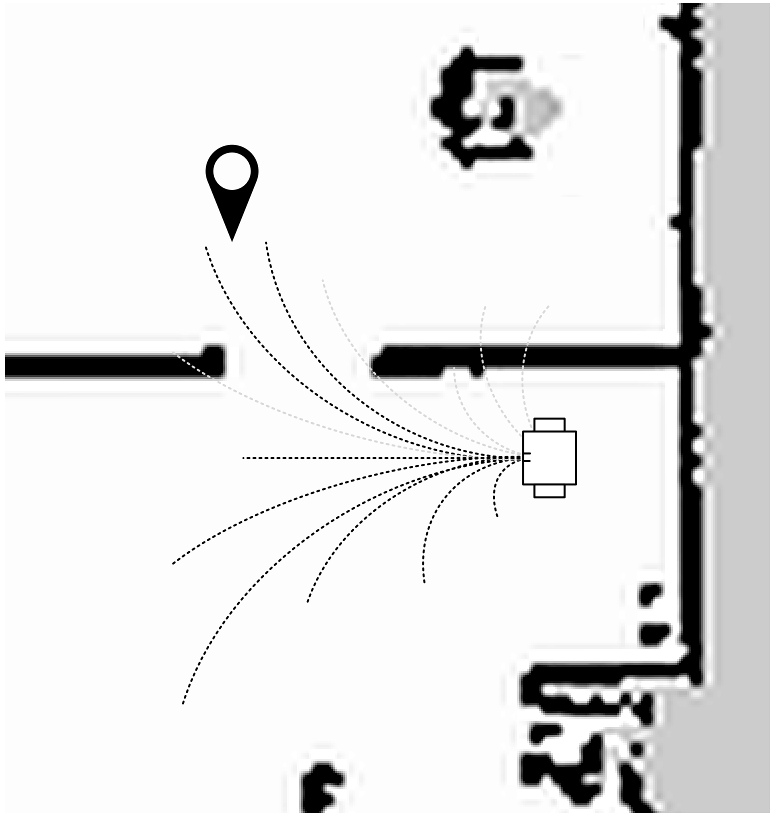


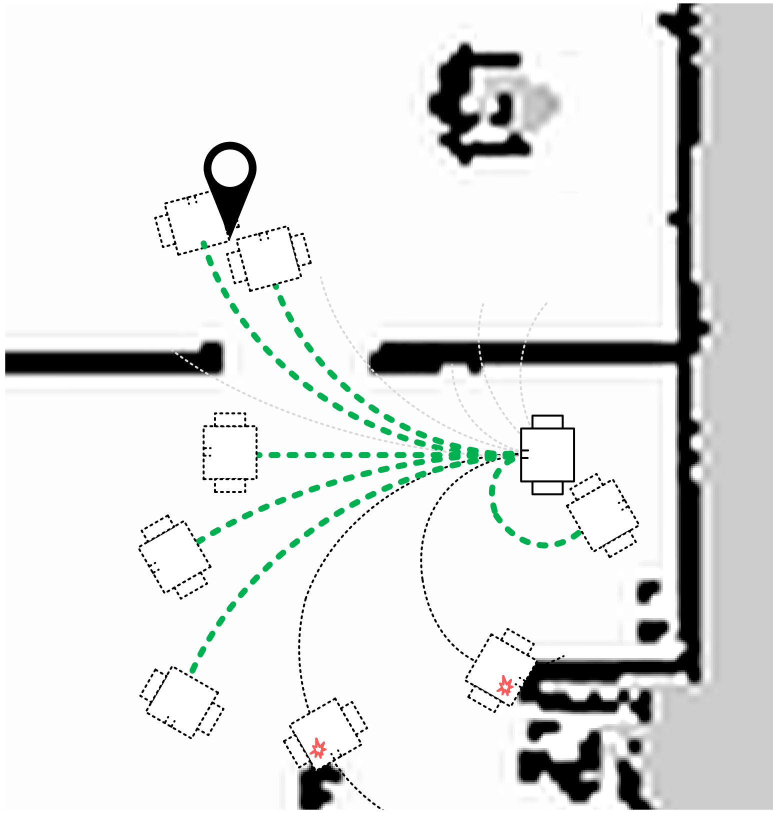


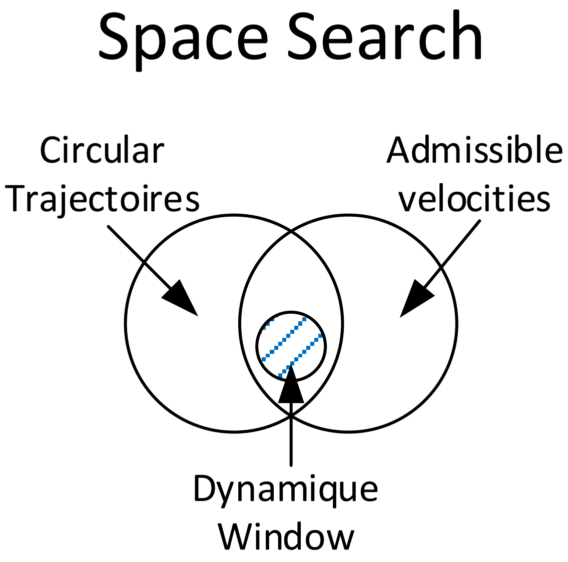
- ❑ Set of possible pair of (v_i, ω_i) velocities without obstacles intersections

DWA Space Search

- ❑ Admissible Velocities



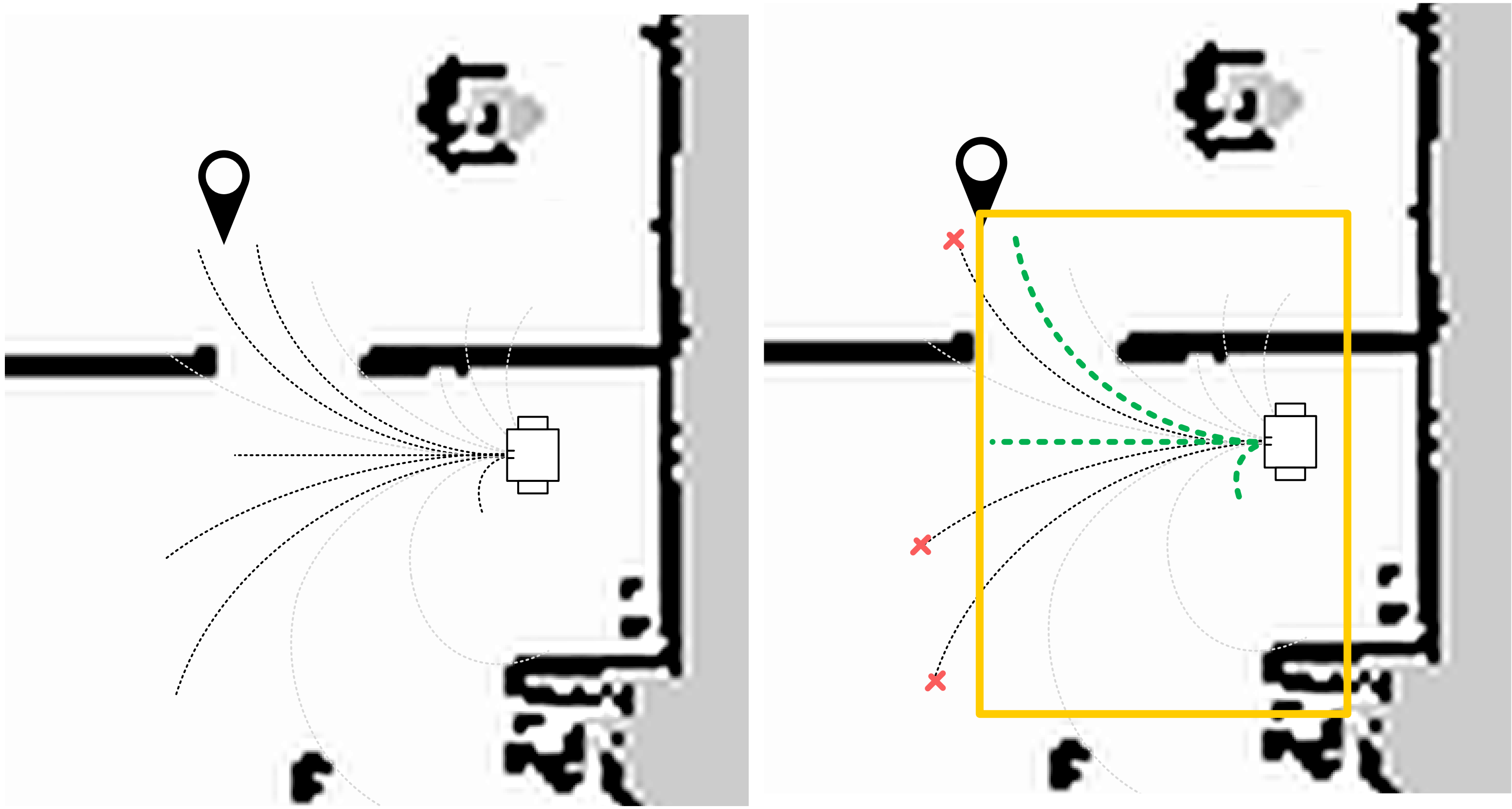




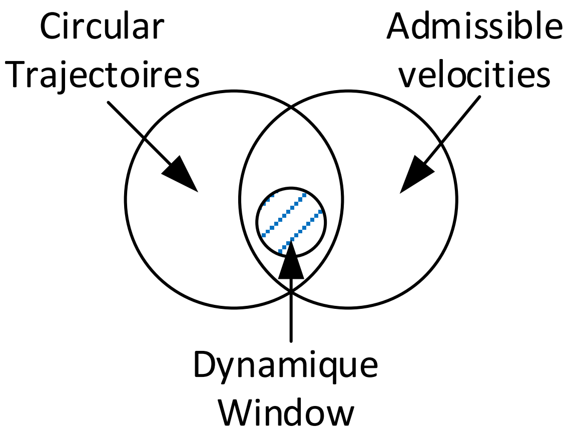
- ❑ Set of admissible velocities allowing robot to stop without colliding with obstacle

DWA Space Search

- ❑ Dynamic Window



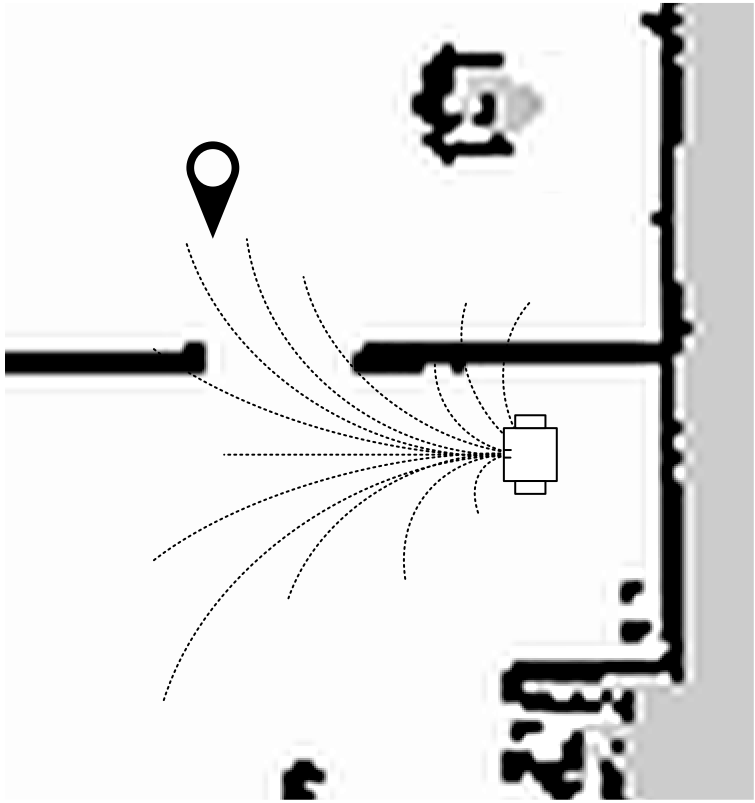
Space Search

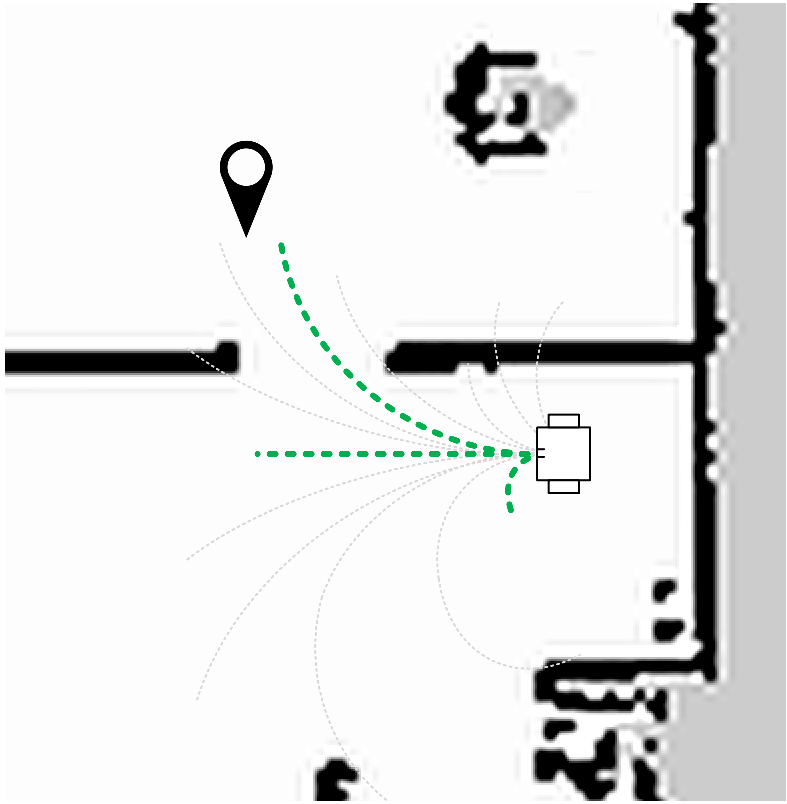


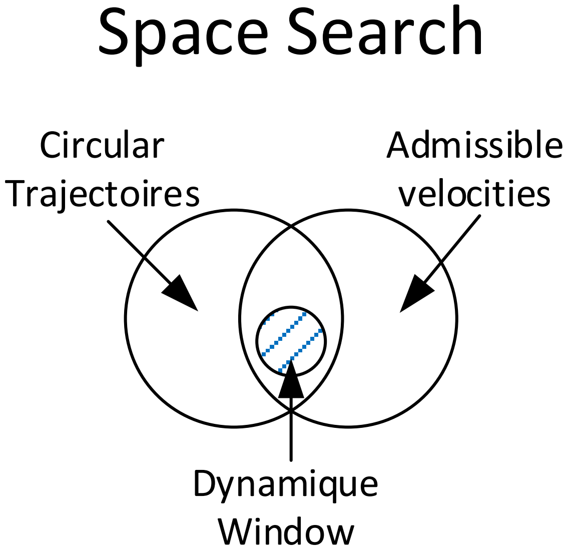
❑ Set of velocities that can be reached within a short time given the current velocity and the limit acceleration of the robot

DWA Space Search

- ❑ Space Search

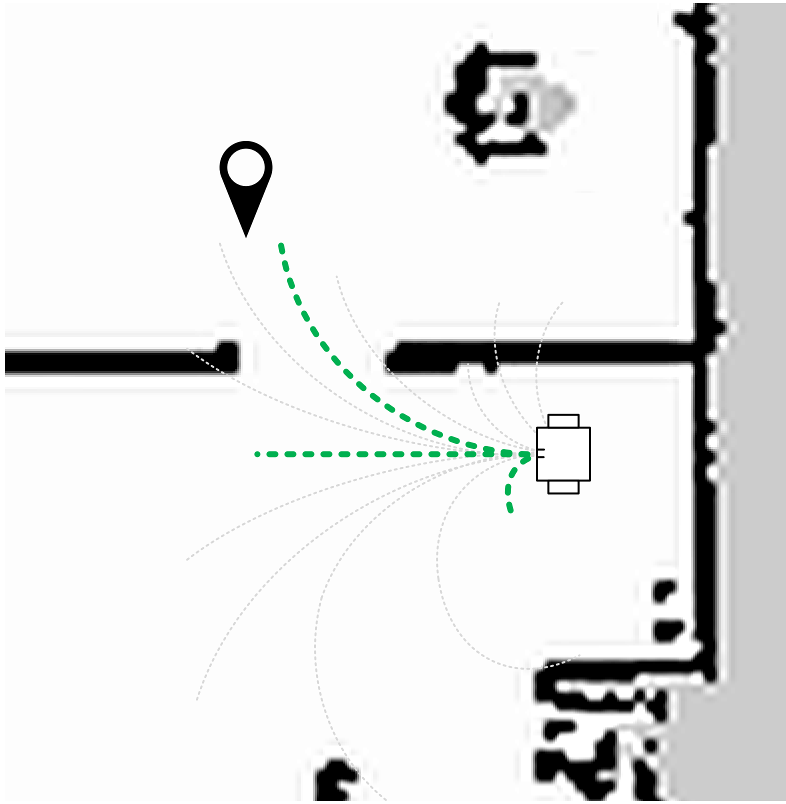


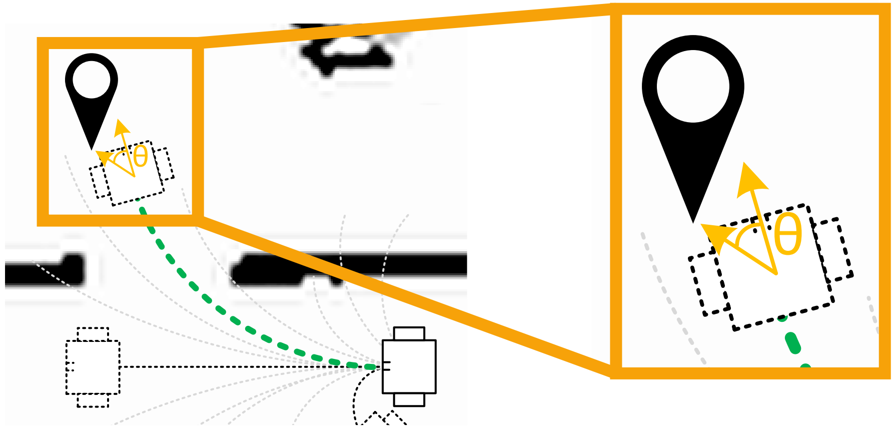


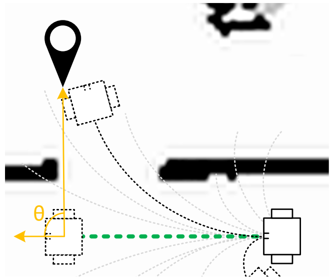


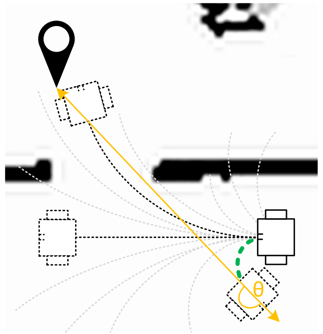
DWA Optimization

- ❑ Target Heading

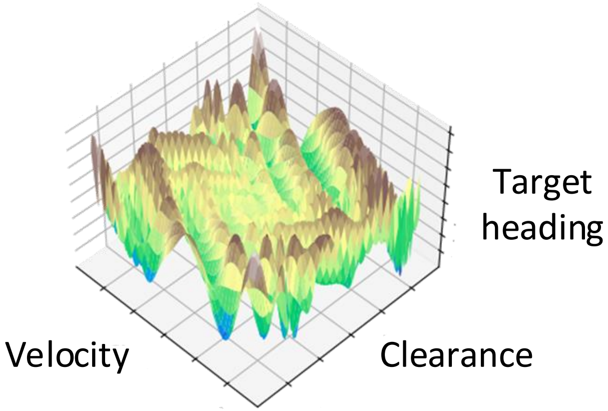








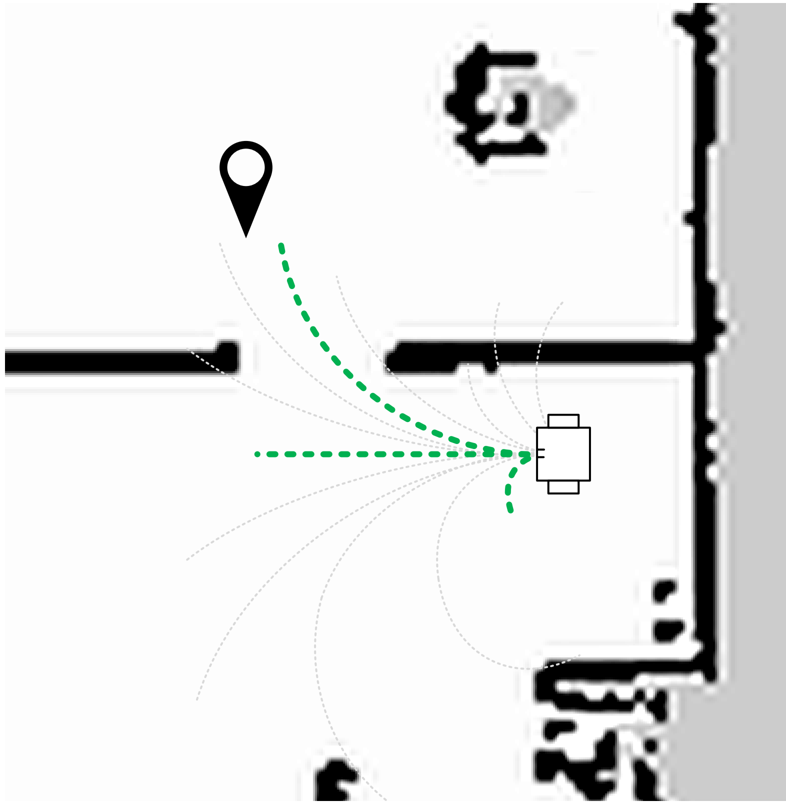
Optimization

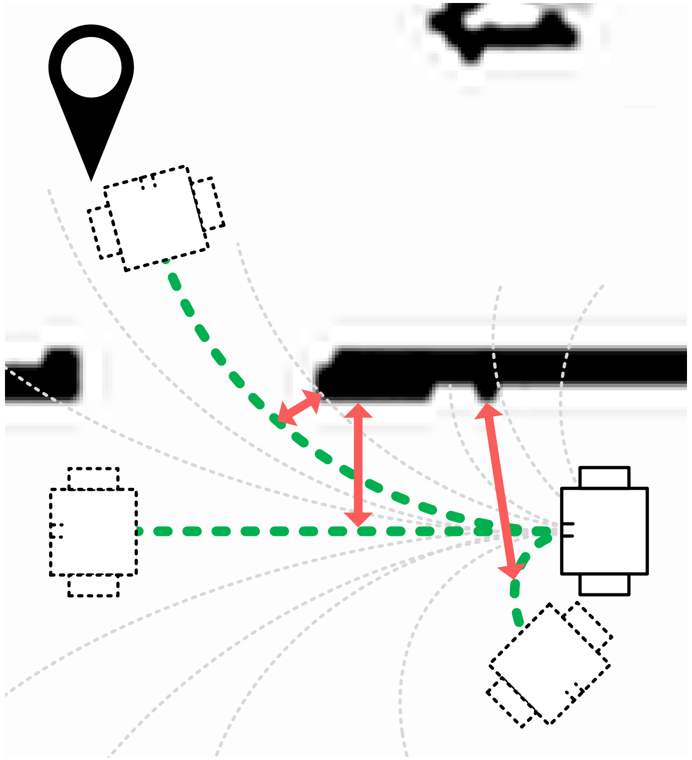


- ❑ Get the angle between the predicted robot orientation and the vector of the predicted robot pose and target $angle(v, \omega)$

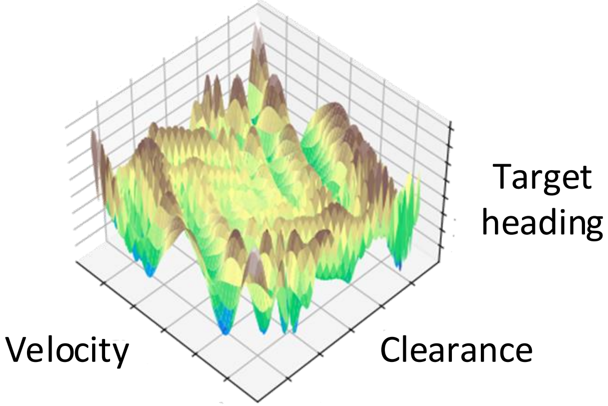
DWA Optimization

- ❑ Clearance





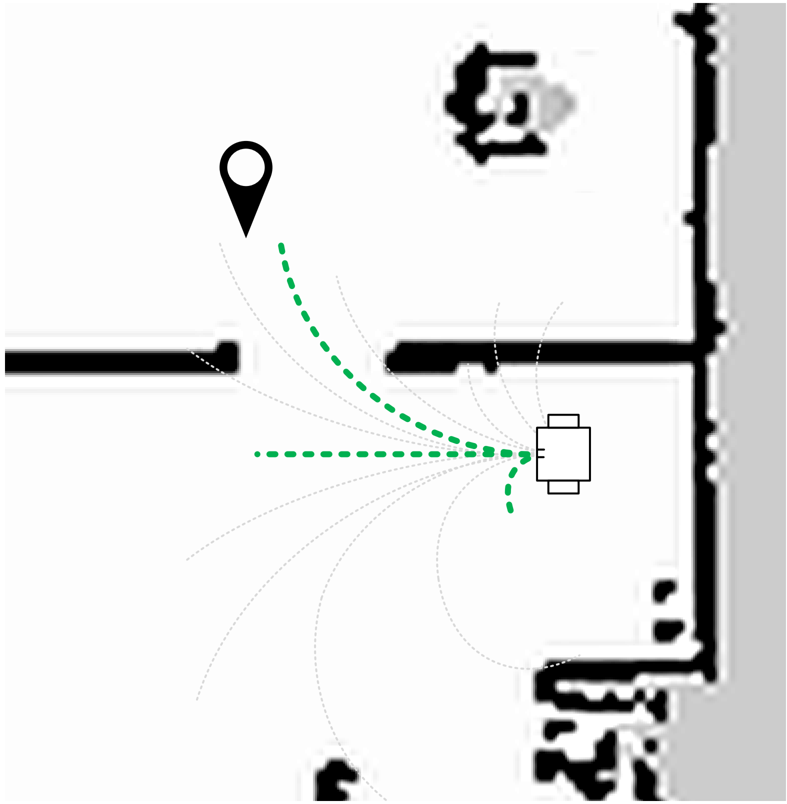
Optimization

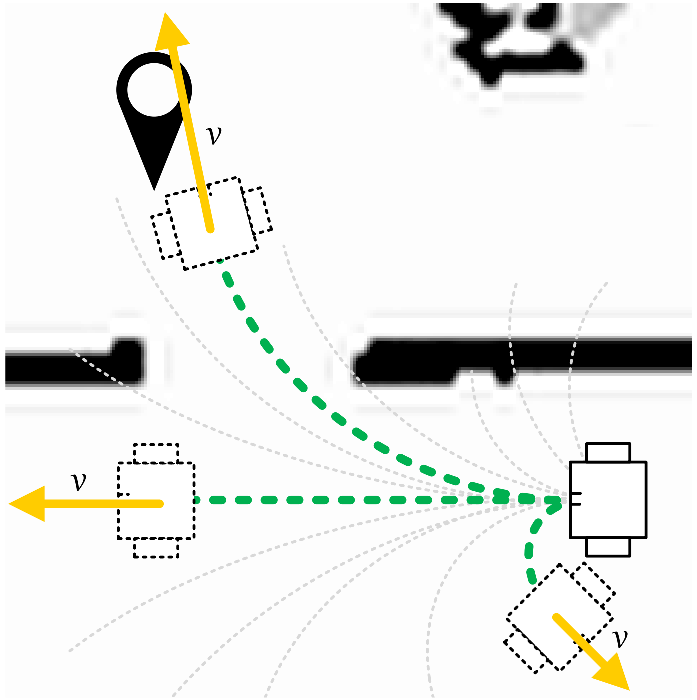


- ❑ Distance to the closest obstacle that intersects with the curvature $dist(v,\omega)$

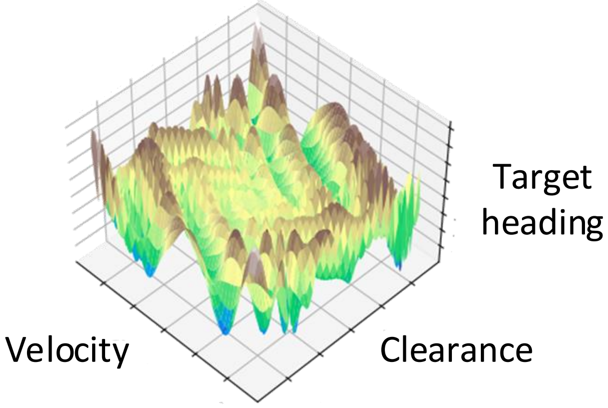
DWA Optimization

❑ Velocity





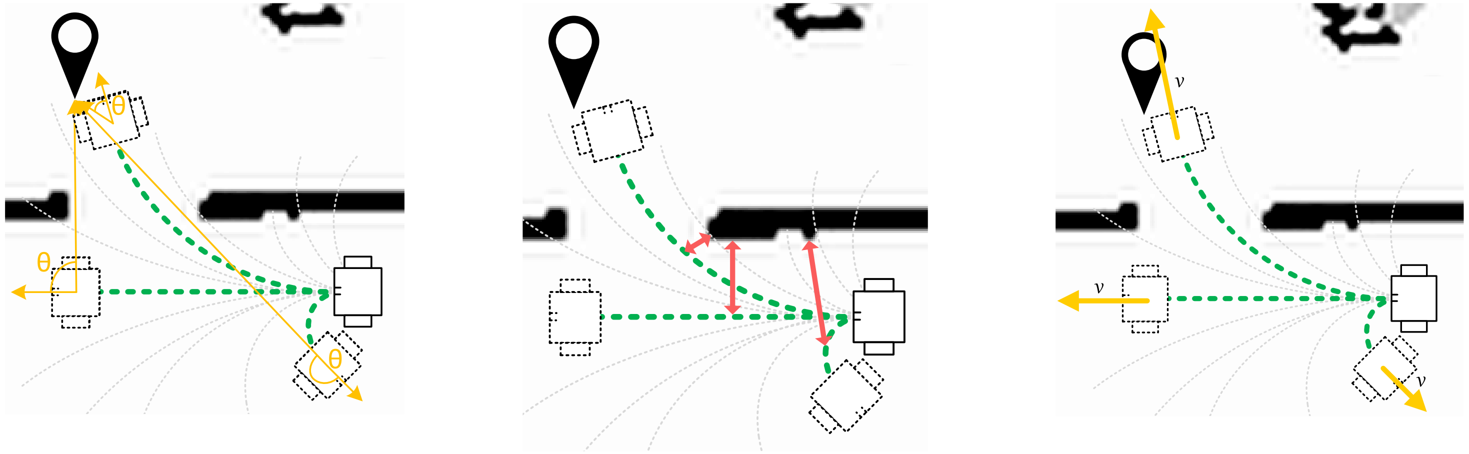
Optimization



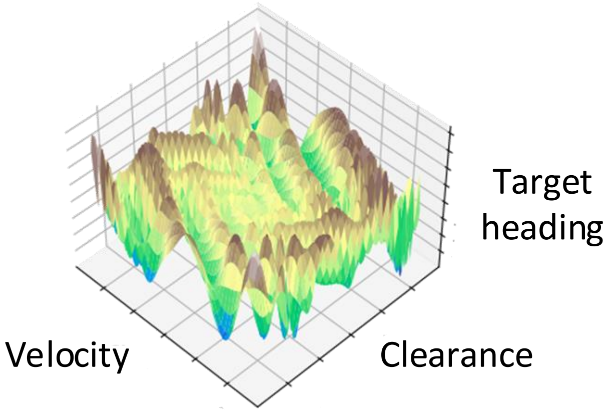
❑ Projection of the translational velocity $velocity(v, \omega)$

DWA Optimization

- ❑ Velocities selection



Optimization



$$G(v,\omega) \;=\; \sigma(\alpha\cdot\underline{\text{angle}(v,\omega)} + \beta\cdot\underline{\text{dist}(v,\omega)} + \gamma\cdot\underline{\text{velocity}(v,\omega)})$$

$$(v,\omega) = argmax\,(\,G(v_i,\omega_\mathrm{i}\,)\,)$$

Ros Updated implementation of DWA

Video recording and archiving is provided by the support of

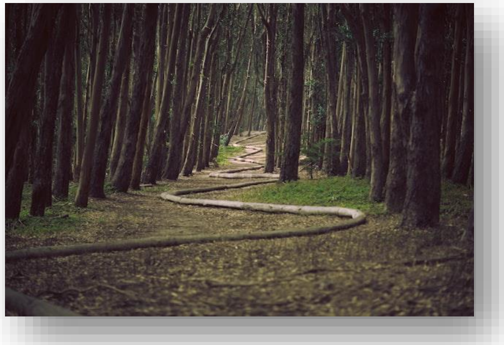
ROSCon 2017 https://roscon.ros.org/2017
#ROSCon #ROSCon2017 @rosorg @osrfoundation

https://vimeo.com/236487972

Ros Update implementation of DWA

- ❑ Implementation of the DWA

- ❑ Update The optimization function
  - **Updated parameters**
    - $dist(v, \omega)$ is computed by getting the cost of the costmap
    - $angle(v, \omega)$ is the same
    - $velocity(v, \omega)$ replaced by $dist_{goal}(v, \omega)$ (to be confirmed)
  - **New parameters**
    - $dist_{globalPath}(v, \omega)$ distance of the trajectory to the computed global path
    - $dist_{goal}(v, \omega)$ distance to the targeted goal
    - $angle_{globalPath}(v, \omega)$ angle between trajectory and the global path



Local Planner EBand

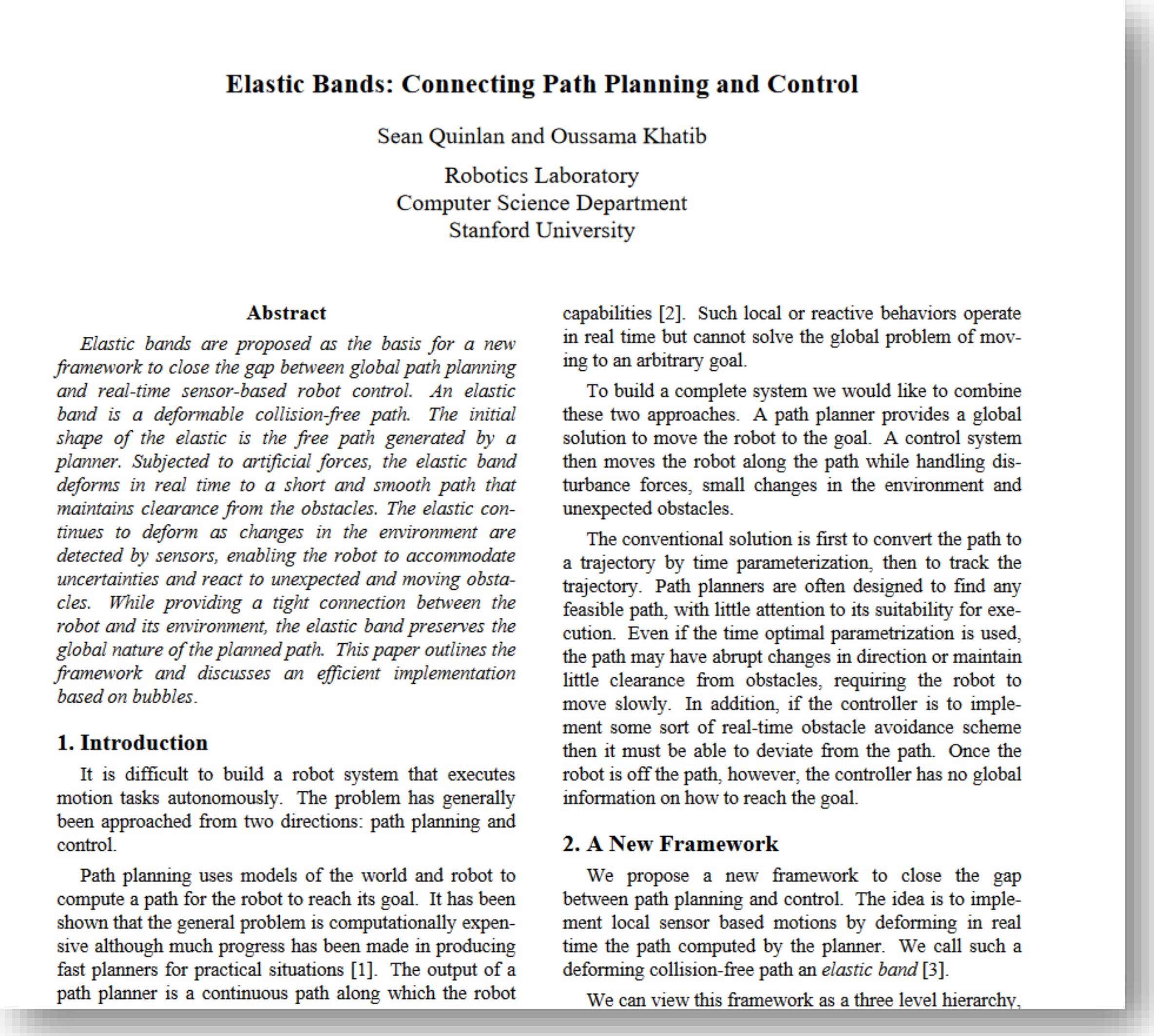
Elastic Band (EBand)

Theory and Ros implementation

Elastic Band

- ❑ Sean Quinlan and Oussama Khaltib 1993

- ❑ Mainline
  - Update quickly global path taking into account observed obstacles
  - Use local deformation of global path
  - Apply virtual forces
    - Internal contration (elastic effect)
    - External repulsion (obstacle)



Elastic Bands: Connecting Path Planning and Control

Sean Quinlan and Oussama Khatib

Robotics Laboratory
Computer Science Department
Stanford University

Abstract

Elastic bands are proposed as the basis for a new framework to close the gap between global path planning and real-time sensor-based robot control. An elastic band is a deformable collision-free path. The initial shape of the elastic is the free path generated by a planner. Subjected to artificial forces, the elastic band deforms in real time to a short and smooth path that maintains clearance from the obstacles. The elastic continues to deform as changes in the environment are detected by sensors, enabling the robot to accommodate uncertainties and react to unexpected and moving obstacles. While providing a tight connection between the robot and its environment, the elastic band preserves the global nature of the planned path. This paper outlines the framework and discusses an efficient implementation based on bubbles.

1. Introduction

It is difficult to build a robot system that executes motion tasks autonomously. The problem has generally been approached from two directions: path planning and control.

Path planning uses models of the world and robot to compute a path for the robot to reach its goal. It has been shown that the general problem is computationally expensive although much progress has been made in producing fast planners for practical situations [1]. The output of a path planner is a continuous path along which the robot capabilities [2]. Such local or reactive behaviors operate in real time but cannot solve the global problem of moving to an arbitrary goal.

To build a complete system we would like to combine these two approaches. A path planner provides a global solution to move the robot to the goal. A control system then moves the robot along the path while handling disturbance forces, small changes in the environment and unexpected obstacles.

The conventional solution is first to convert the path to a trajectory by time parameterization, then to track the trajectory. Path planners are often designed to find any feasible path, with little attention to its suitability for execution. Even if the time optimal parametrization is used, the path may have abrupt changes in direction or maintain little clearance from obstacles, requiring the robot to move slowly. In addition, if the controller is to implement some sort of real-time obstacle avoidance scheme then it must be able to deviate from the path. Once the robot is off the path, however, the controller has no global information on how to reach the goal.

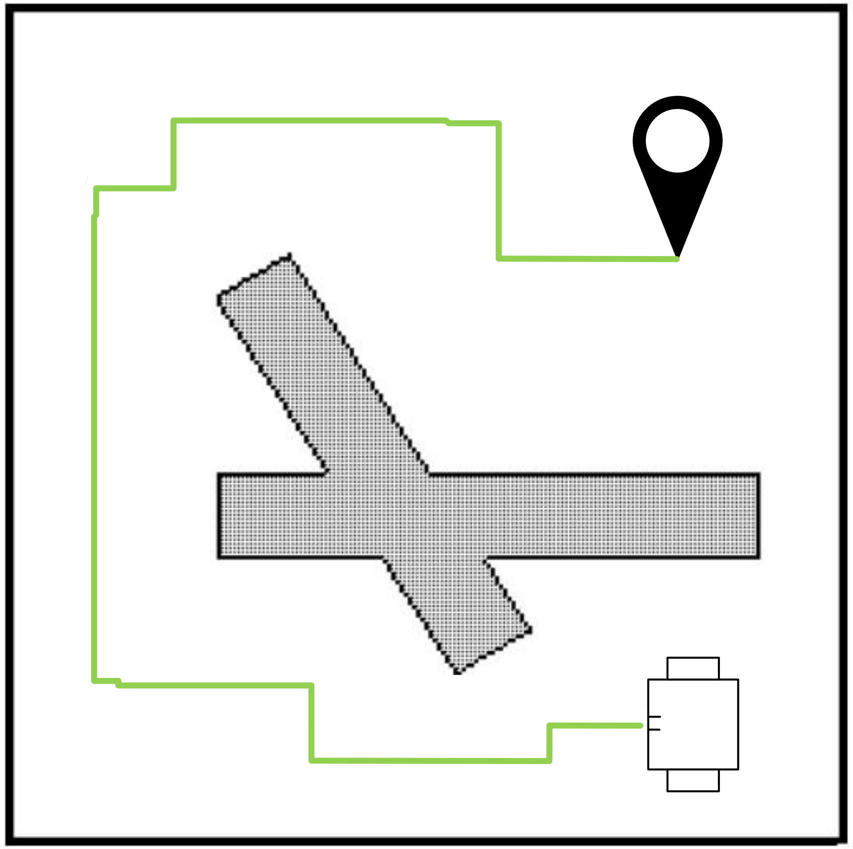
2. A New Framework

We propose a new framework to close the gap between path planning and control. The idea is to implement local sensor based motions by deforming in real time the path computed by the planner. We call such a deforming collision-free path an *elastic band* [3].

We can view this framework as a three level hierarchy,

Elastic Band main line

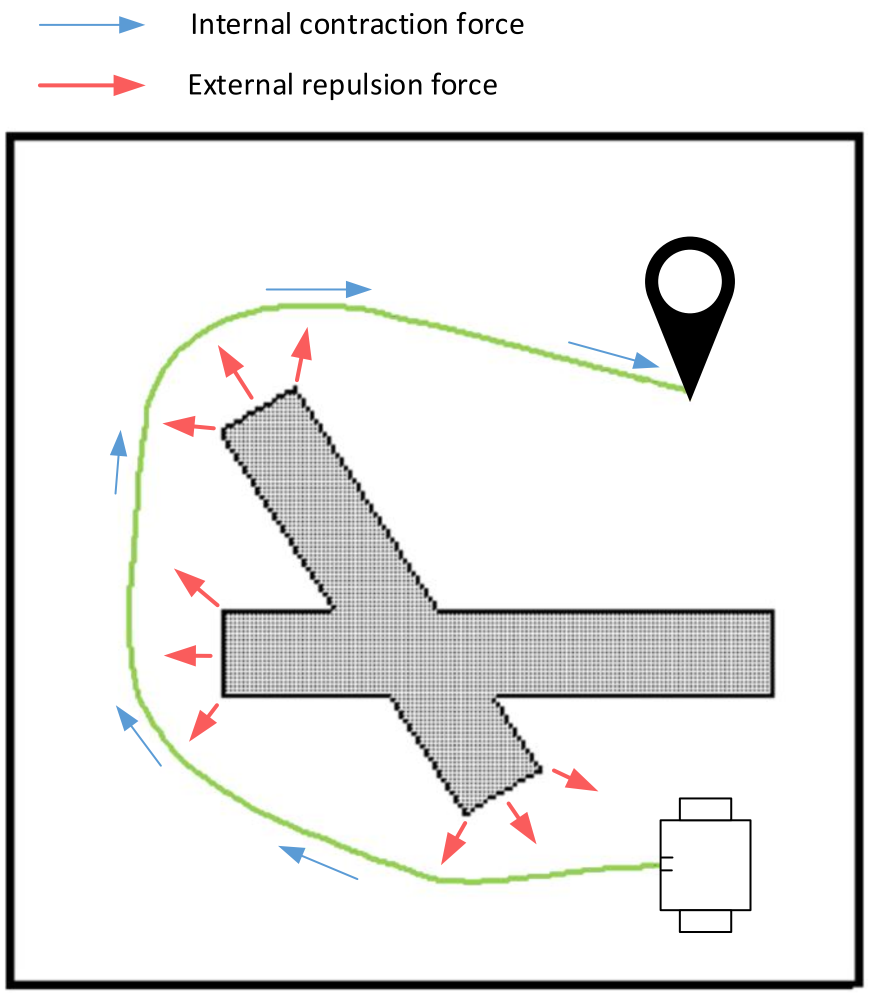
- ❑ Path generated by global planner



Elastic Band main line

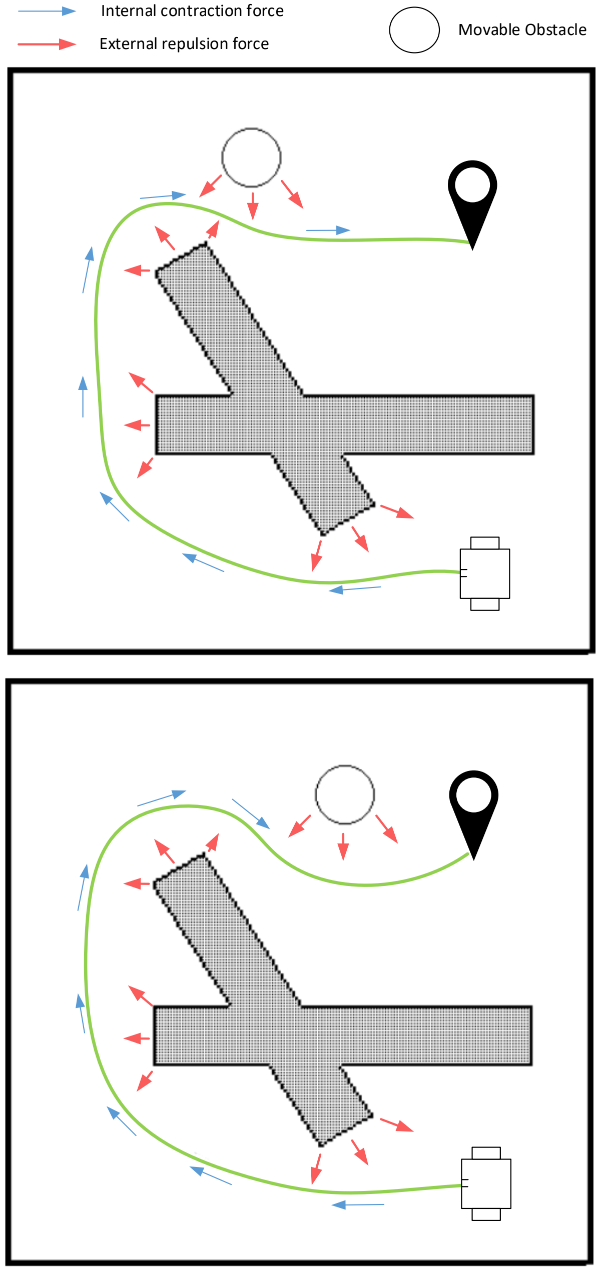
- ❑ Path generated by global planner

- ❑ Applying both internal contraction force and external repultion force



Elastic Band main line

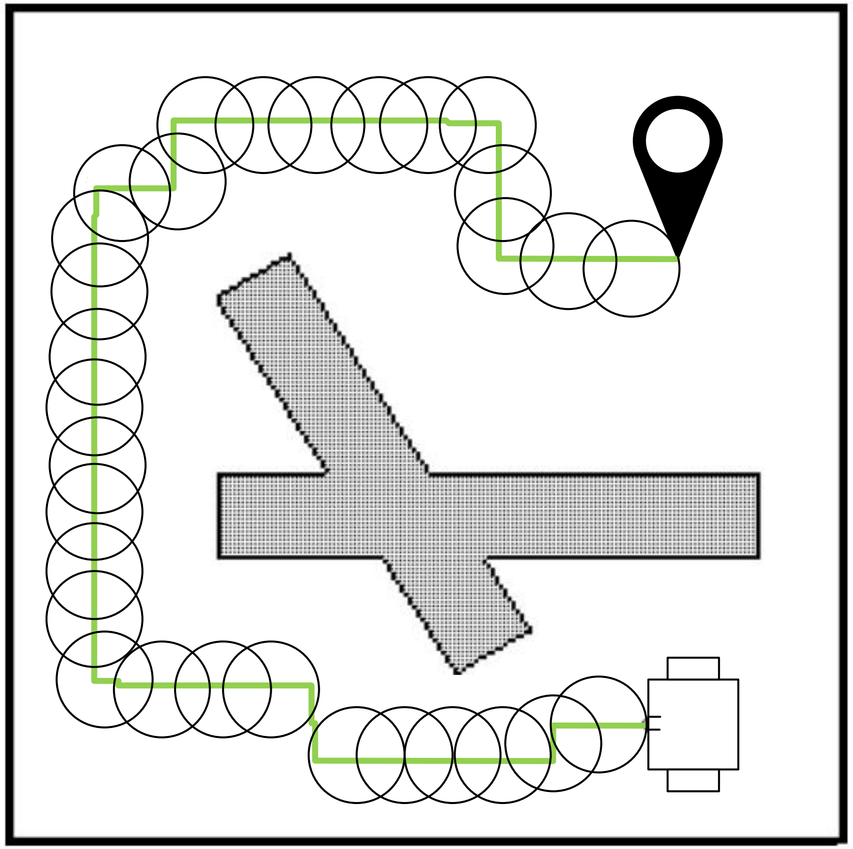
- □ Path generated by global planner
- □ Applying both internal contraction force and external repultion force
- □ Updating repulsion force according new obstation (e.g moving obs.)



Elastic Band How to ?

❑ Generate Bubbles along the path

1. Create bubble is distance to next > const$_{disc}$
2. Determine closed obstacle and calculate bubble radius (max radius needs to be set)
3. Calculate repulsion force to the closest obstacle (e.g (max – r)/r)
4. Calculate internal contraction force. (e.g normalized vector of previous and following bubbles center)
5. Update bubble new position according forces
6. Remove Bubble is necessairy (to closed each other) and check that bubble raidus > robot radius

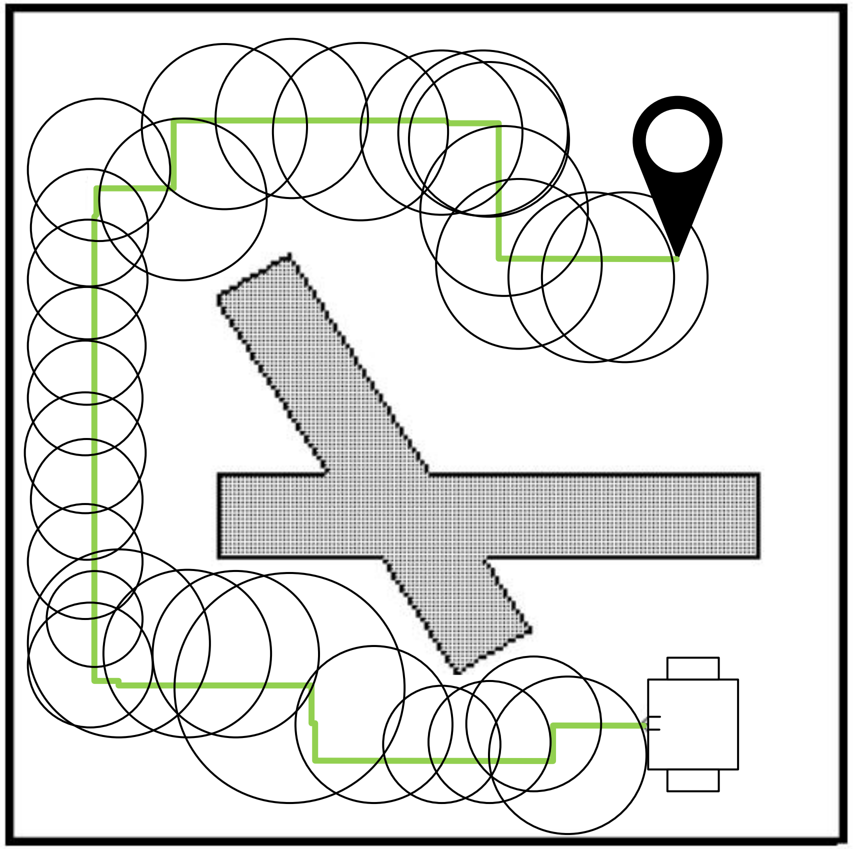


❑ Optimise bubbles set (remove if overlaps, ensure 2 bubbles neighbors connection)

Elastic Band How to ?

❑ Generate Bubbles along the path

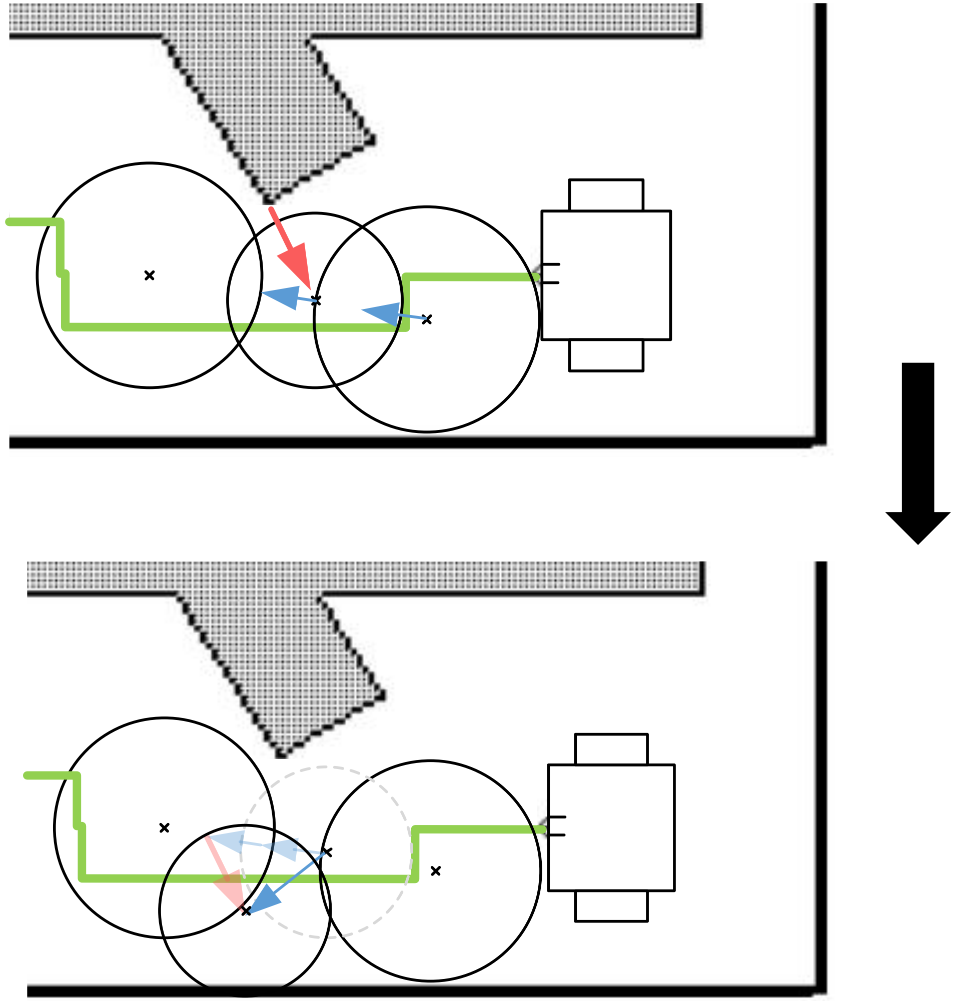
1. Create bubble is distance to next > const$_{disc}$
2. Determine closed obstacle and calculate bubble radius (max radius needs to be set)
3. Calculate repulsion force to the closest obstacle (e.g (max – r)/r)
4. Calculate internal contraction force. (e.g normalized vector of previous and following bubbles center)
5. Update bubble new position according forces
6. Remove Bubble is necessairy (to closed each other) and check that bubble raidus > robot radius



❑ Optimise bubbles set (remove if overlaps, ensure 2 bubbles neighbors connection)

Elastic Band How to ?

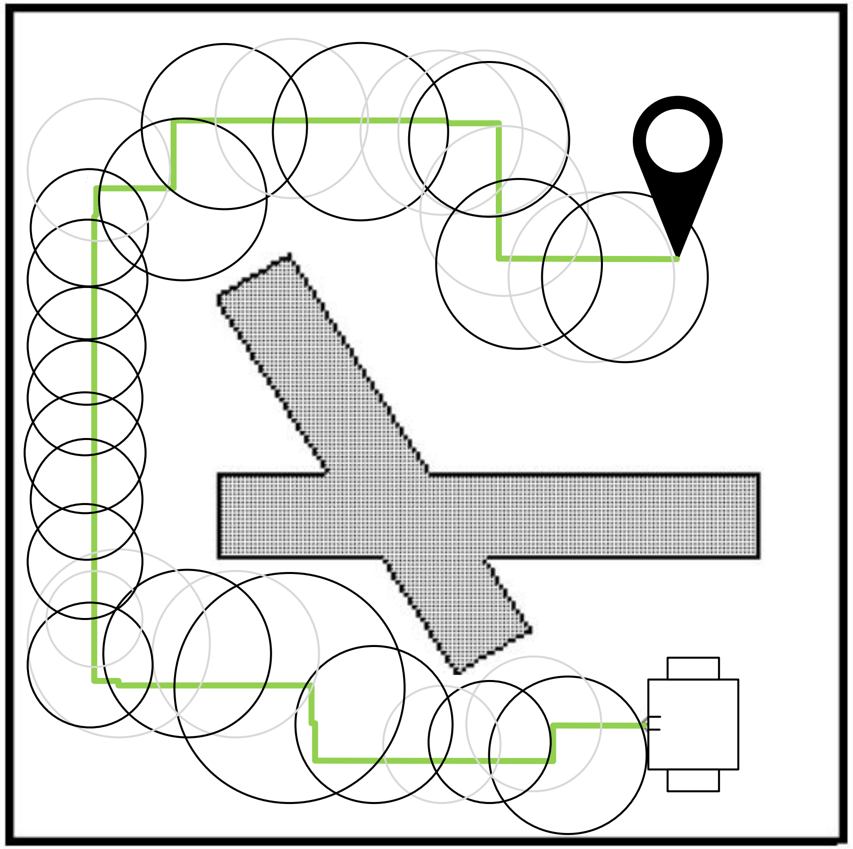
- ❑ Generate Bubbles along the path
  1. Create bubble is distance to next > const$_{disc}$
  2. Determine closed obstacle and calculate bubble radius (max radius needs to be set)
  3. Calculate repulsion force to the closest obstacle (e.g (max – r)/r)
  4. Calculate internal contraction force. (e.g normalized vector of previous and following bubbles center)
  5. Update bubble new position according forces
  6. Remove Bubble is necessairy (too closed each other) and check that bubble raidus > robot radius



Elastic Band How to ?

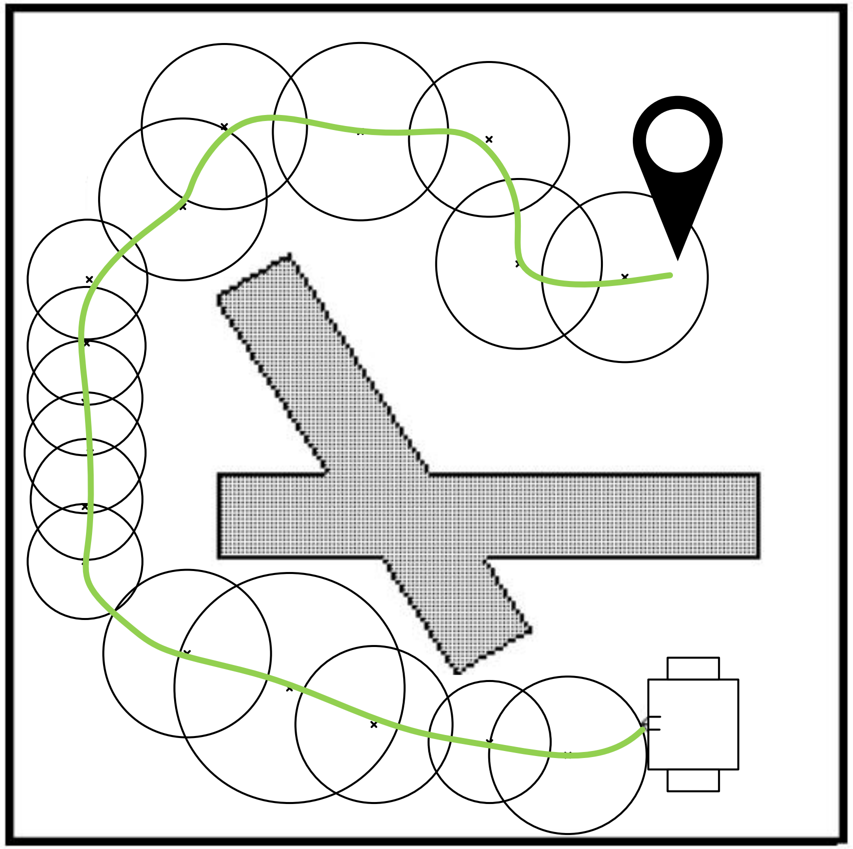
❑ Generate Bubbles along the path

1. Create bubble is distance to next > const$_{disc}$
2. Determine closed obstacle and calculate bubble radius (max radius needs to be set)
3. Calculate repulsion force to the closest obstacle (e.g (max – r)/r)
4. Calculate internal contraction force. (e.g normalized vector of previous and following bubbles center)
5. Update bubble new position according forces
6. Remove Bubble is necessairy (too closed each other) and check that bubble raidus > robot radius

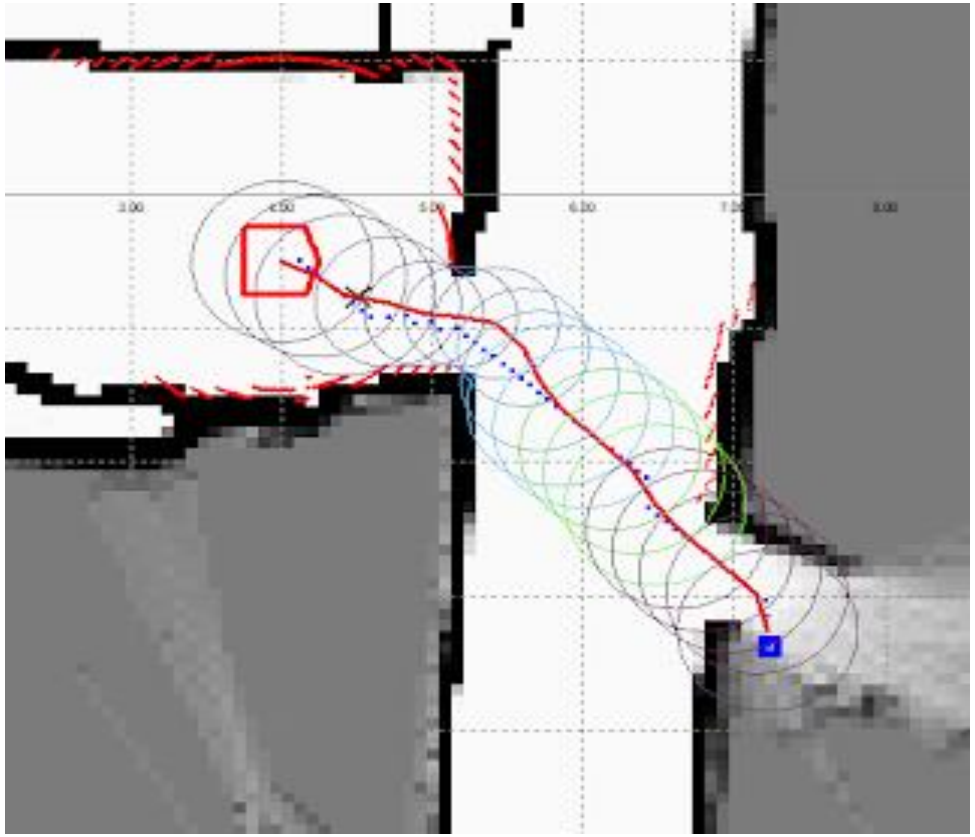


Elastic Band How to ?

- ❑ Generate Bubbles along the path
  1. Create bubble is distance to next > $\text{const}_{\text{disc}}$
  2. Determine closed obstacle and calculate bubble radius (max radius needs to be set)
  3. Calculate repulsion force to the closest obstacle (e.g (max – r)/r)
  4. Calculate internal contraction force. (e.g normalized vector of previous and following bubbles center)
  5. Update bubble new position according forces
  6. Remove Bubble is necessairy (too closed each other) and check that bubble raidus > robot radius



Elastic Band Usage Exemple



http://adrianboeing.blogspot.com/2012/03/elastic-band-realtime-pathfinding.html

https://youtu.be/KJgHAhJxUr0

http://wiki.ros.org/eband_local_planner

Local Planner TEB

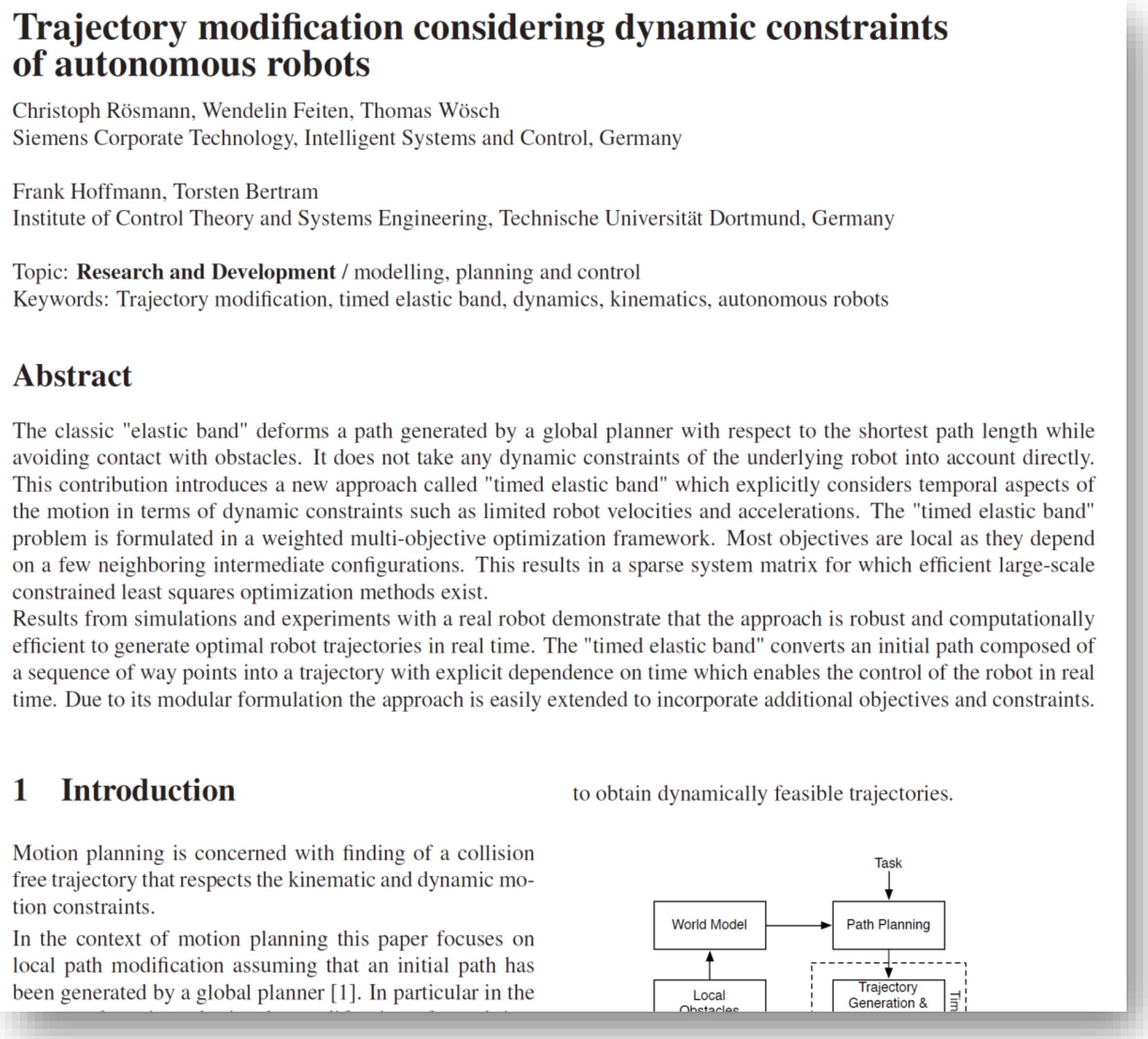
Timed Elastic Band (TEB)

Theory and Ros implementation

Timed Elastic Band

- ❑ C. Rösmann, W. Feiten, T. Wösch 2012 (extended in 2017)

- ❑ Mainline
  - Inspired of Elastic Band but taking into account robot dynamic contraints
  - Act as a weighted multi-objective optimisation framework
  - Use multiple objective function
    - Distance to path
    - Distance to obstacle
    - Velocity and acceleration
    - Fatest path



Trajectory modification considering dynamic constraints of autonomous robots

Christoph Rösmann, Wendelin Feiten, Thomas Wösch
Siemens Corporate Technology, Intelligent Systems and Control, Germany

Frank Hoffmann, Torsten Bertram
Institute of Control Theory and Systems Engineering, Technische Universität Dortmund, Germany

Topic: **Research and Development** / modelling, planning and control
Keywords: Trajectory modification, timed elastic band, dynamics, kinematics, autonomous robots

Abstract

The classic "elastic band" deforms a path generated by a global planner with respect to the shortest path length while avoiding contact with obstacles. It does not take any dynamic constraints of the underlying robot into account directly. This contribution introduces a new approach called "timed elastic band" which explicitly considers temporal aspects of the motion in terms of dynamic constraints such as limited robot velocities and accelerations. The "timed elastic band" problem is formulated in a weighted multi-objective optimization framework. Most objectives are local as they depend on a few neighboring intermediate configurations. This results in a sparse system matrix for which efficient large-scale constrained least squares optimization methods exist.
Results from simulations and experiments with a real robot demonstrate that the approach is robust and computationally efficient to generate optimal robot trajectories in real time. The "timed elastic band" converts an initial path composed of a sequence of way points into a trajectory with explicit dependence on time which enables the control of the robot in real time. Due to its modular formulation the approach is easily extended to incorporate additional objectives and constraints.

1 Introduction

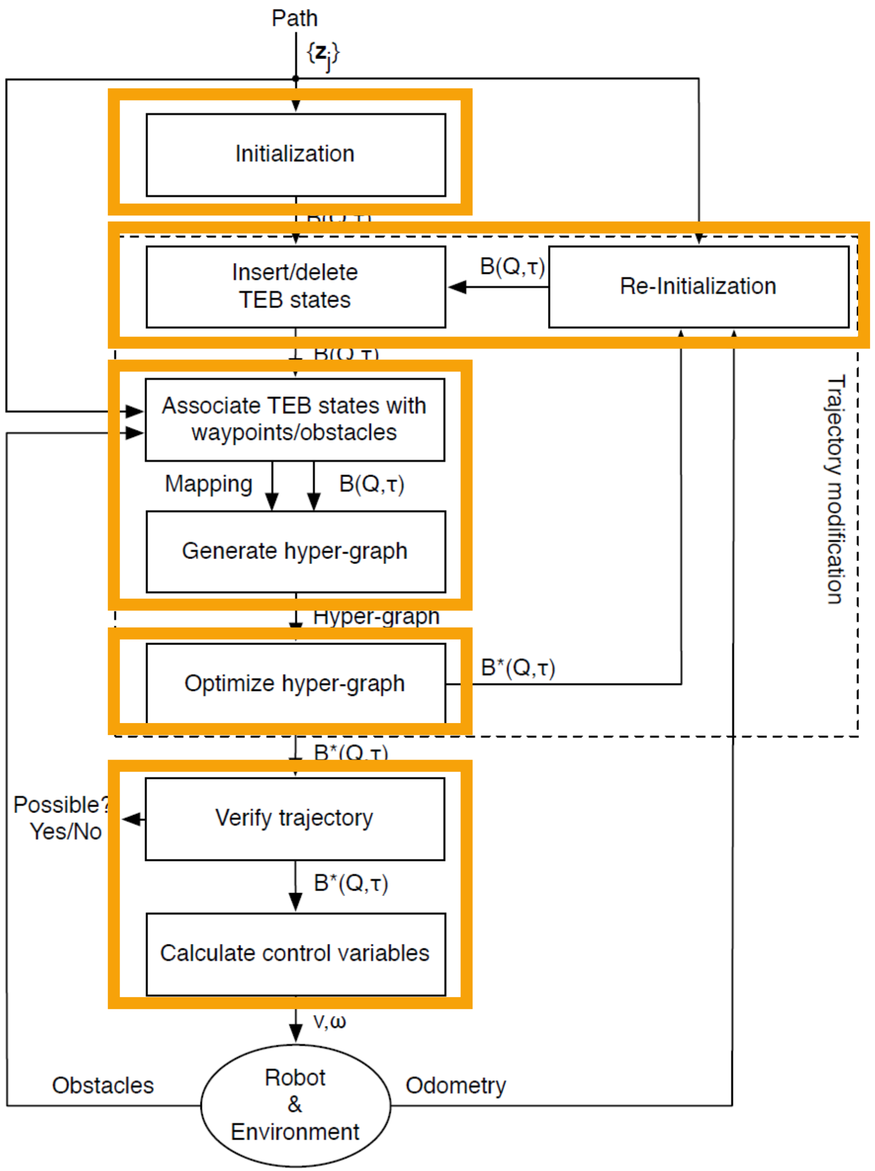
Motion planning is concerned with finding of a collision free trajectory that respects the kinematic and dynamic motion constraints.

In the context of motion planning this paper focuses on local path modification assuming that an initial path has been generated by a global planner [1]. In particular in the

to obtain dynamically feasible trajectories.

Timed Elastic Band main line

- ❑ Generate configuration candidates from start to goal
- ❑ Associate states and formulate Objective function
- ❑ Optimize functions (Hyper graph)
- ❑ Compute translational and rotational velocity
- ❑ [Iterate] Update way point and add/delete new configuration according spacial/temporal resolution to the remaining trajectory



Timed Elastic Band main line

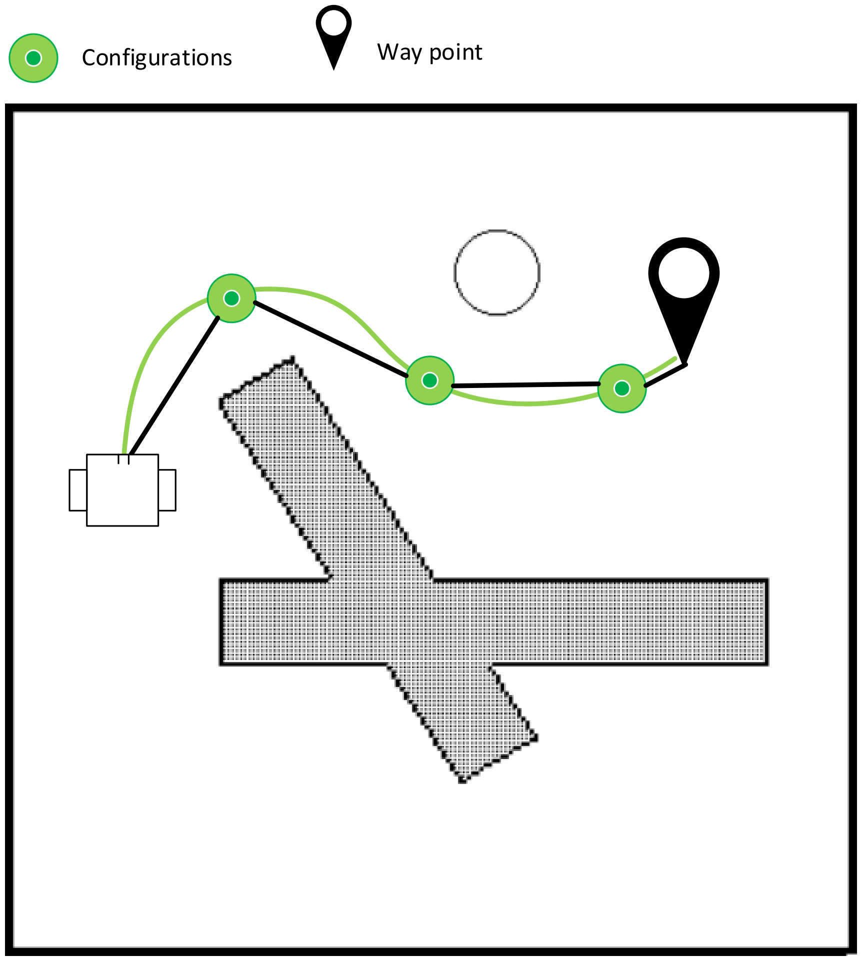
- □ Generate configuration candidates from start to goal

$$s_k = (x_k, y_k, \beta_k)$$

1 configuration

$$b = [s_1, \Delta T_1, s_2, \Delta T_2, \dots, s_n, \Delta T_{n-1}]$$

Use Tuple mixing configuration and associated duration



Timed Elastic Band main line

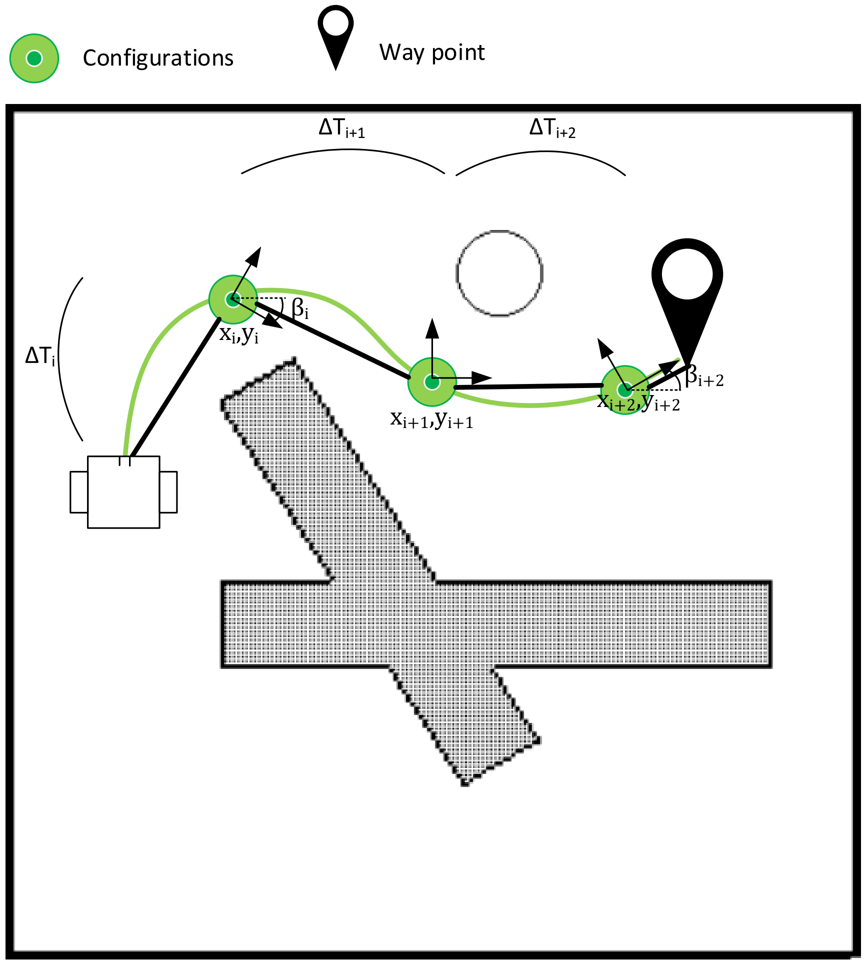
- Generate configuration candidates from start to goal

$$s_k = (x_k, y_k, \beta_k)$$

1 configuration

$$b = [s_1, \Delta T_1, s_2, \Delta T_2, \dots, s_n, \Delta T_{n-1}]$$

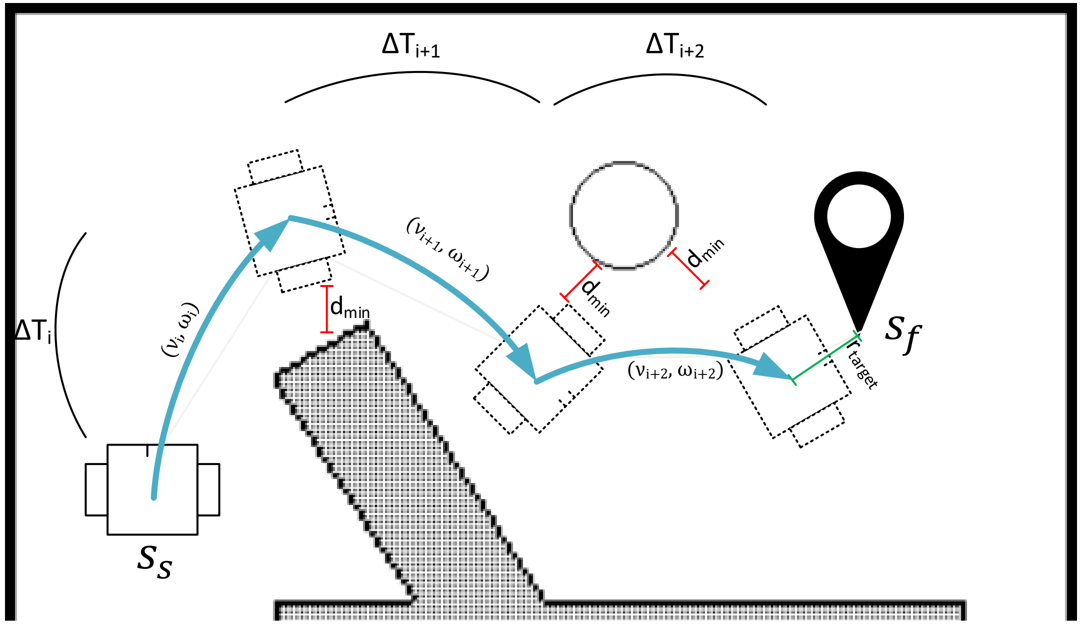
Use Tuple mixing configuration and associated duration



Timed Elastic Band main line

- ❑ Associate states and formulate Objective function

$$V^*(b) = \min_{b} \sum_{k=1}^{n-1} \Delta T_k^2$$



Optimization with following constraints :

$\boldsymbol{h_k(s_{k+1}, s_k)}$ Kinematic constraints between 2 consecutive poses

$\boldsymbol{r_k(s_{k+1}, s_k)}$ Minimun turning radius (in case of carlike robot)

- Robot velocity and acceleration constraints

$\boldsymbol{v_k(s_{k+1}, s_k, \Delta T_k)}$ $\boldsymbol{a_k(s_{k+2}, s_{k+1}, s_k, \Delta T_{k+1}, \Delta T_k)}$

$\boldsymbol{o_k(s_k)}$ Minimal respect of distance to obstacle

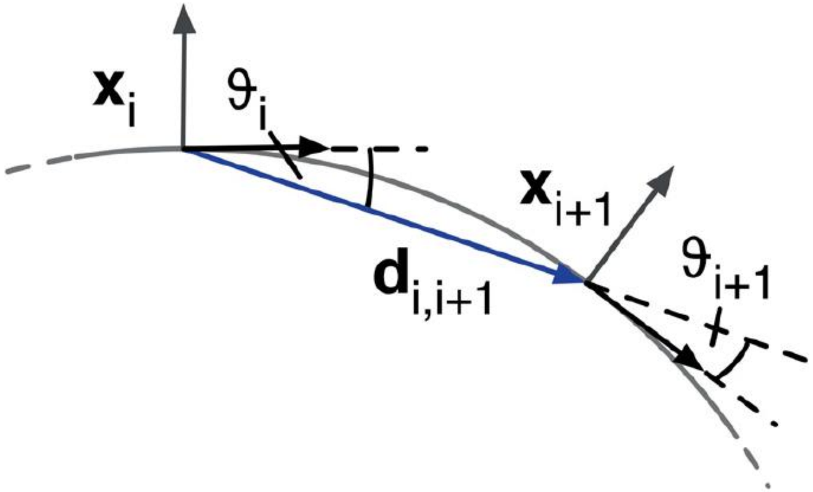
Timed Elastic Band main line

- ❑ Associate states and formulate Objective function

Optimization with following constraints :

- Kinematic constraints between 2 consecutive poses
- Minimun turning radius (in case of carlike robot)
- Robot velocity and acceleration constraints
- Minimal respect of distance to obstacle

Non-holonomic kenematics



$$\theta_k = \theta_{k+1}$$

$$\begin{bmatrix}\cos(\beta_k)\\ \sin(\beta_k)\\ 0\end{bmatrix} \times d_{k,k+1} = \begin{bmatrix}\cos(\beta_{k+1})\\ \sin(\beta_{k+1})\\ 0\end{bmatrix} \times d_{k,k+1}$$

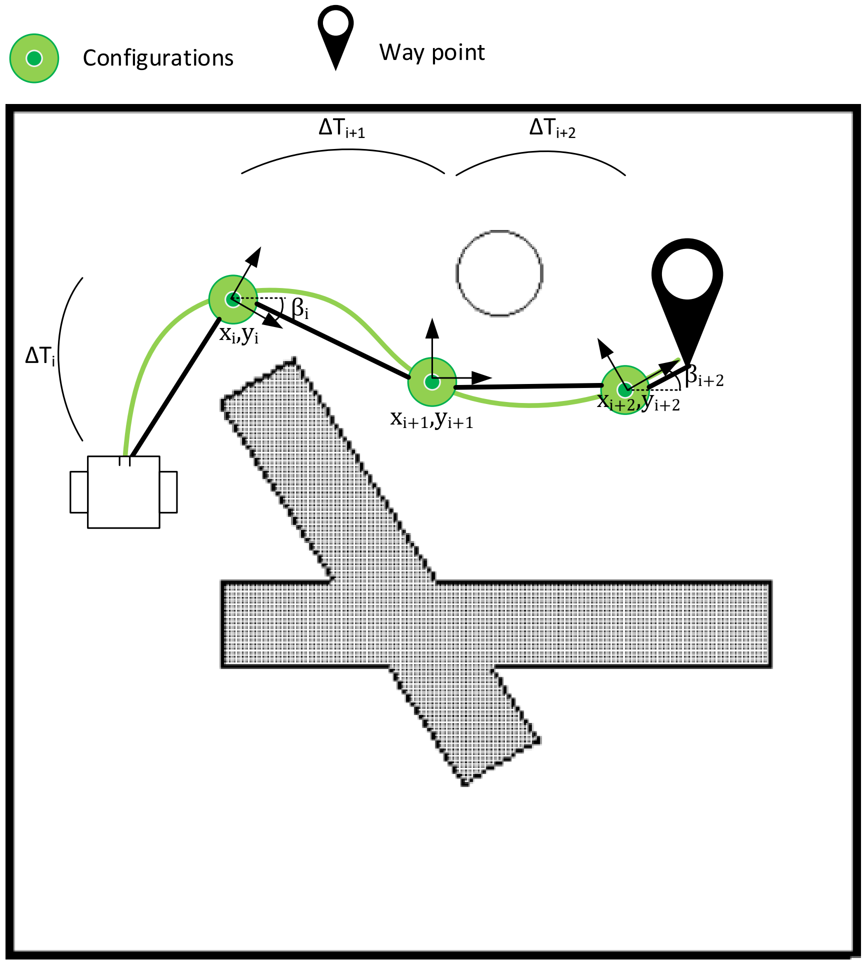
$$h_k(s_{k+1}, s_k) = \begin{bmatrix}\cos(\beta_k)\\ \sin(\beta_k)\\ 0\end{bmatrix} + \begin{bmatrix}\cos(\beta_{k+1})\\ \sin(\beta_{k+1})\\ 0\end{bmatrix} \times d_{k,k+1} \qquad \boldsymbol{h_k(s_{k+1}, s_k) = 0}$$

Timed Elastic Band main line

❑ Associate states and formulate Objective function

Optimization with following constraints :

- Kinematic constraints between 2 consecutive poses
- Minimun turning radius (in case of carlike robot)
- Robot velocity and acceleration constraints
- Minimal respect of distance to obstacle



$$v_k = \frac{\|[x_{k+1} - x_k, y_{k+1} - y_k]\| \gamma(s_k, s_{k+1})}{\Delta T_k} \qquad a_k = \frac{2(v_{k+1} - v_k)}{\Delta T_k + \Delta T_{k+1}}$$

$$\omega_k = \frac{(\beta_{k+1} - \beta_k)}{\Delta T_k}$$

$$v_k(s_{k+1}, s_k, \Delta T_k) = [v_{max} - |v_k|, \omega_{max} - |\omega_k|]$$

$$a_k(s_{k+2}, s_{k+1}, s_k, \Delta T_{k+1}, \Delta T_k) = [a_{max} - |a_k|, \dot{\omega}_{max} - |\dot{\omega}_k|]$$

$$\boldsymbol{v_k(s_{k+1}, s_k, \Delta T_k) \geq 0}$$

$$\boldsymbol{a_k(s_{k+2}, s_{k+1}, s_k, \Delta T_{k+1}, \Delta T_k) \geq 0}$$

Timed Elastic Band main line

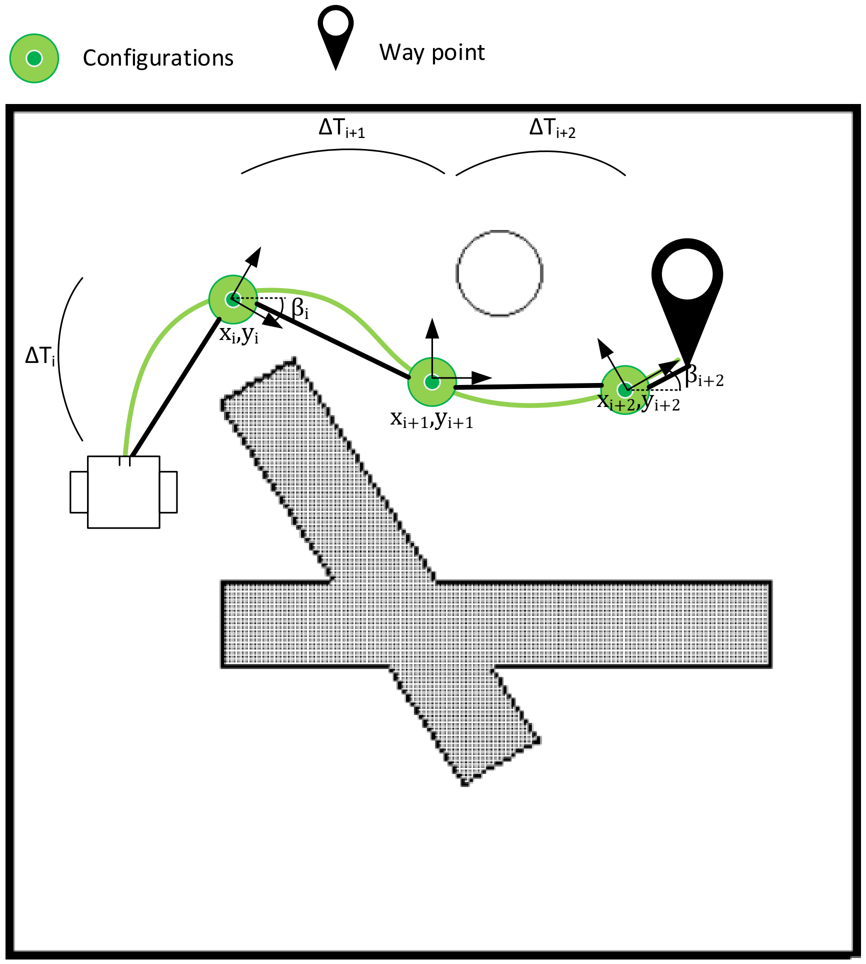
- ❑ Associate states and formulate Objective function

  Optimization with following constraints :

  - Kinematic constraints between 2 consecutive poses
  - Minimun turning radius (in case of carlike robot)
  - Robot velocity and acceleration constraints
  - Minimal respect of distance to obstacle

$\rho(s_k, O)$ Euclidian distance between current pose and an obstacle

$$o_k(s_k) = [\rho(s_k, O_1), \rho(s_k, O_2), \ldots, \rho(s_k, O_R)] - [\rho_{min}, \rho_{min}, \ldots, \rho_{min}]$$

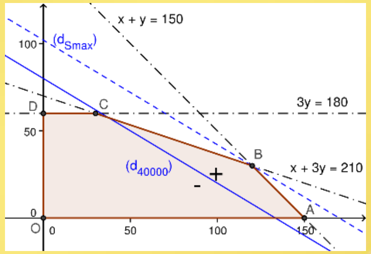


$$\boldsymbol{o_k(s_k) \geq 0}$$

Timed Elastic Band main line

Linear Optimization not possible

→ Nolinear Function



- ❑ Associate states and formulate Objective function

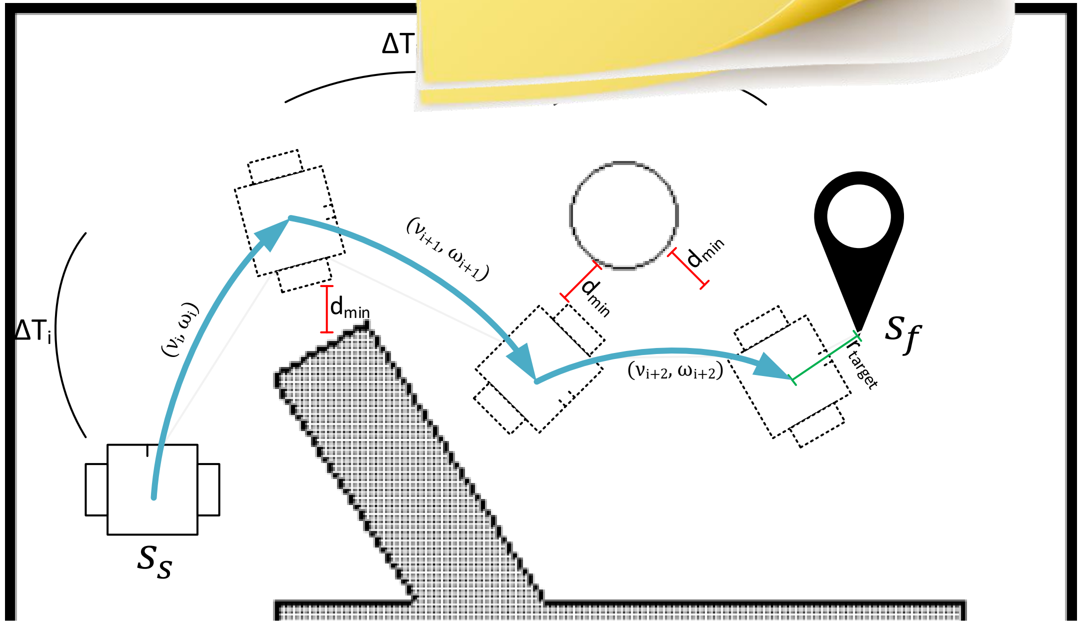
$$\boxed{V^*(b) = \min_{b} \sum_{k=1}^{n-1} \Delta T_k^2}$$

Optimization with following constraints :

- Kinematic constraints between 2 consecutive poses
- Minimun turning radius (in case of carlike robot)
- Robot velocity and acceleration constraints
- Minimal respect of distance to obstacle

$$h_k(s_{k+1}, s_k) = 0 \qquad v_k(s_{k+1}, s_k, \Delta T_k) \geq 0 \qquad o_k(s_k) \geq 0$$

$$a_k(s_{k+2}, s_{k+1}, s_k, \Delta T_{k+1}, \Delta T_k) \geq 0$$



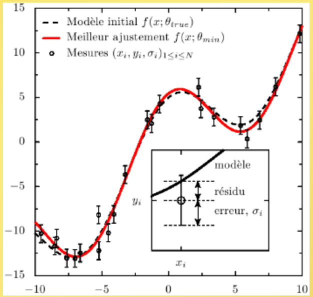
Timed Elastic Band main line

- ❑ How to resolve the constrainted optimization ?? → Approximative Least-sc

$$V^*(b) = \min_{b} \sum_{k=1}^{n-1} \Delta T_k^2$$

Approximative Least-square

$$S(\theta) = \sum_{i=1}^{N} \left(y_i - f(x_i;\theta)\right)^2 = \sum_{i=1}^{N} r_i^2(\theta)$$



Quadratic penality with scalar weight σ_h and identity I

$$\boldsymbol{h_k(s_{k+1}, s_k) = 0} \longrightarrow \varphi(h_k, \sigma_h) = \sigma_h h_k^T I h_k = \sigma_h \|h_k\|_2^2$$

$$\boldsymbol{v_k(s_{k+1}, s_k, \Delta T_k) \geq 0} \longrightarrow \chi(v_k, \sigma_v) = \sigma_v \|min\{0, v_k\}\|_2^2$$

$$\boldsymbol{a_k(s_{k+2}, s_{k+1}, s_k, \Delta T_{k+1}, \Delta T_k) \geq 0} \longrightarrow \chi(a_k, \sigma_a) = \sigma_a \|min\{0, a_k\}\|_2^2$$

$$\boldsymbol{o_k(s_k) \geq 0} \longrightarrow \chi(o_k, \sigma_o) = \sigma_o \|min\{0, o_k\}\|_2^2$$

Timed Elastic Band main line

- ❑ How to resolve the constrainted optimization ?? → Approximative Least-s

$$V^*(b) = \min_b \sum_{k=1}^{n-1} \Delta T_k^2$$

$$\boldsymbol{h_k(s_{k+1}, s_k) = 0} \longrightarrow \varphi(h_k, \sigma_h) = \sigma_h \|h_k\|_2^2$$

Quadratic penality with scalar weight σ_h

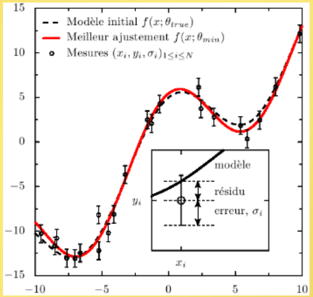
$$\boldsymbol{v_k(s_{k+1}, s_k, \Delta T_k) \geq 0}$$

$$\boldsymbol{a_k(s_{k+2}, s_{k+1}, s_k, \Delta T_{k+1}, \Delta T_k) \geq 0}$$

$$\boldsymbol{o_k(s_k) \geq 0}$$

Approximative Least-square

$$S(\theta) = \sum_{i=1}^{N} (y_i - f(x_i;\theta))^2 = \sum_{i=1}^{N} r_i^2(\theta)$$



Norme euclidienne

$$\|\vec{x}\|_2 = \sqrt{|x_1|^2 + \ldots + |x_n|^2}$$

Timed Elastic Band main line

- ❑ How to resolve the constrainted optimization ?? → Approximative Least-square optimization

$$V^*(b) = \min_{b} \sum_{k=1}^{n-1} \Delta T_k^2$$

Quadratic penality with scalar weight σ_h and identity I

$$\boldsymbol{h_k}(\boldsymbol{s_{k+1}}, \boldsymbol{s_k}) = \boldsymbol{0} \longrightarrow \varphi(h_k, \sigma_h) = \sigma_h \|h_k\|_2^2$$

$$\boldsymbol{v_k}(\boldsymbol{s_{k+1}}, \boldsymbol{s_k}, \Delta\boldsymbol{T_k}) \geq \boldsymbol{0} \longrightarrow \chi(v_k, \sigma_v) = \sigma_v \|min\{0, v_k\}\|_2^2$$

$$\boldsymbol{a_k}(\boldsymbol{s_{k+2}}, \boldsymbol{s_{k+1}}, \boldsymbol{s_k}, \Delta\boldsymbol{T_{k+1}}, \Delta\boldsymbol{T_k}) \geq \boldsymbol{0} \longrightarrow \chi(a_k, \sigma_a) = \sigma_a \|min\{0, a_k\}\|_2^2$$

$$\boldsymbol{o_k}(\boldsymbol{s_k}) \geq \boldsymbol{0} \longrightarrow \chi(o_k, \sigma_o) = \sigma_o \|min\{0, o_k\}\|_2^2$$

Timed Elastic Band main line

- ❑ How to resolve the constrainted optimization ?? → Approximative Least-square optimization

$$\mathrm{V}^*(\mathrm{b}) = \min_{b} \sum_{k=1}^{n-1} \Delta T_k^2$$

$$\tilde{V}(\mathrm{b}) = \sum_{k=1}^{n-1} \left[\Delta T_k^2 + \varphi(h_k, \sigma_h) + \chi(v_k, \sigma_v) + \chi(a_k, \sigma_a) + \chi(o_k, \sigma_o)\right] + \chi(a_n, \sigma_a)$$

$$b^* = \widetilde{ar}gmin\, \tilde{V}(\mathrm{b})$$

Quadratic penality with scalar weight σ_h and identity I

$\boldsymbol{h_k(s_{k+1}, s_k) = 0}$ → $\varphi(h_k, \sigma_h) = \sigma_h \|h_k\|_2^2$

$\boldsymbol{v_k(s_{k+1}, s_k, \Delta T_k) \geq 0}$ → $\chi(v_k, \sigma_v) = \sigma_v \|min\{0, v_k\}\|_2^2$

$\boldsymbol{a_k(s_{k+2}, s_{k+1}, s_k, \Delta T_{k+1}, \Delta T_k) \geq 0}$ → $\chi(a_k, \sigma_a) = \sigma_a \|min\{0, a_k\}\|_2^2$

$\boldsymbol{o_k(s_k) \geq 0}$ → $\chi(o_k, \sigma_o) = \sigma_o \|min\{0, o_k\}\|_2^2$

Timed Elastic Band main line

- ❑ Generate configuration candidates from start to goal

$$\mathbf{x}_i = (x_i, y_i, \beta_i)$$

1 configuration

$$Q = \{\mathbf{x}_i\}_{i=0\ldots n}$$

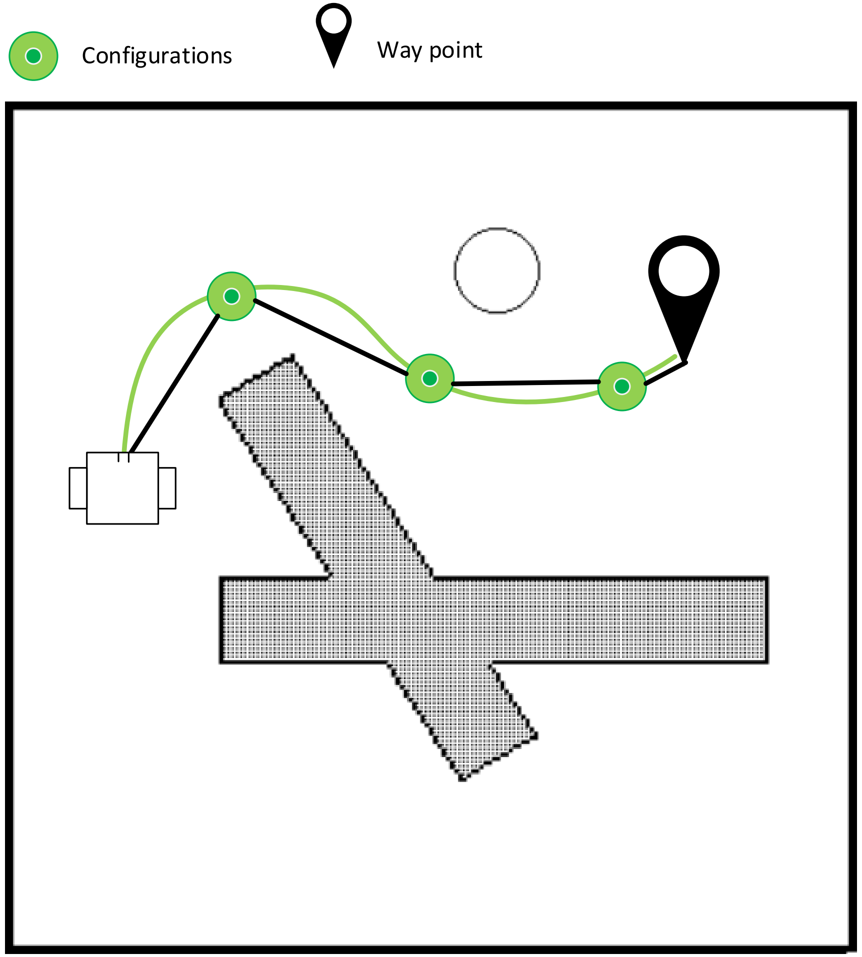
Set of configurations

$$\tau = \{\Delta T_i\}_{i=0\ldots n-1}$$

Set of duration to reach the next configuration

$$B := (Q, \tau)$$

Use Tuple mixing configuration and associated duration



Timed Elastic Band main line

- ❑ Generate configuration candidates from start to goal

$$\mathbf{x}_i = (x_i, y_i, \beta_i)$$

1 configuration

$$Q = \{\mathbf{x}_i\}_{i=0\ldots n}$$

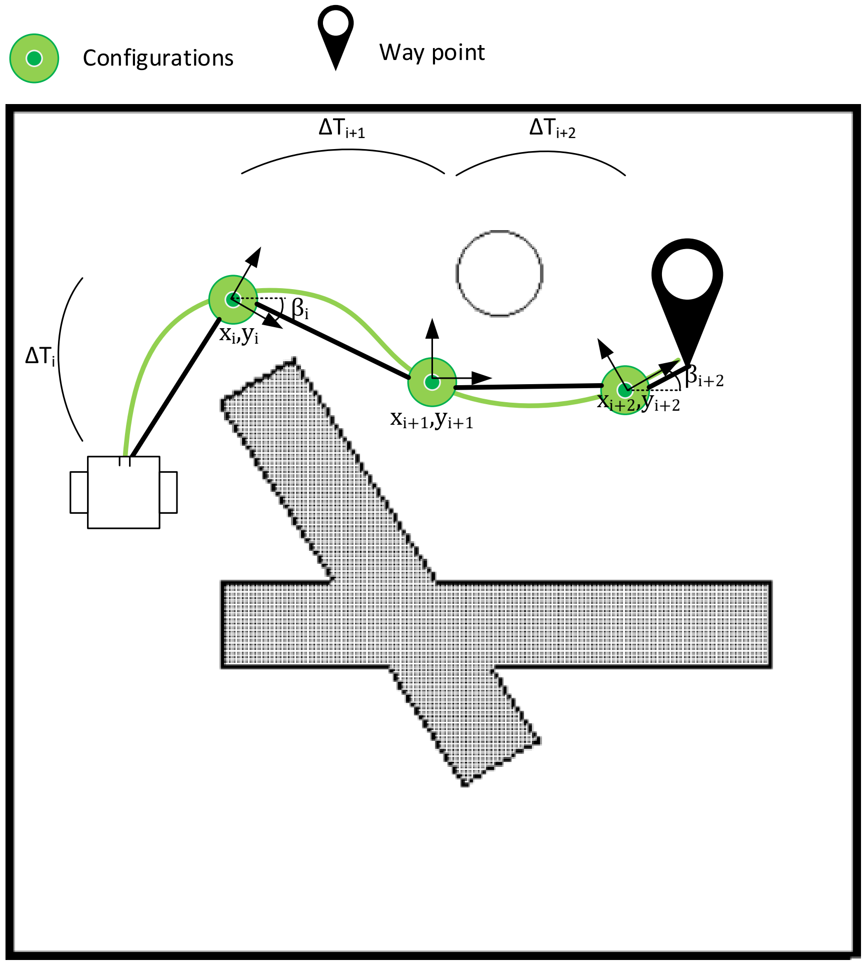
Set of configurations

$$\tau = \{\Delta T_i\}_{i=0\ldots n-1}$$

Set of duration to reach the next configuration

$$B := (Q, \tau)$$

Use Tuple mixing configuration and associated duration



Timed Elastic Band main line

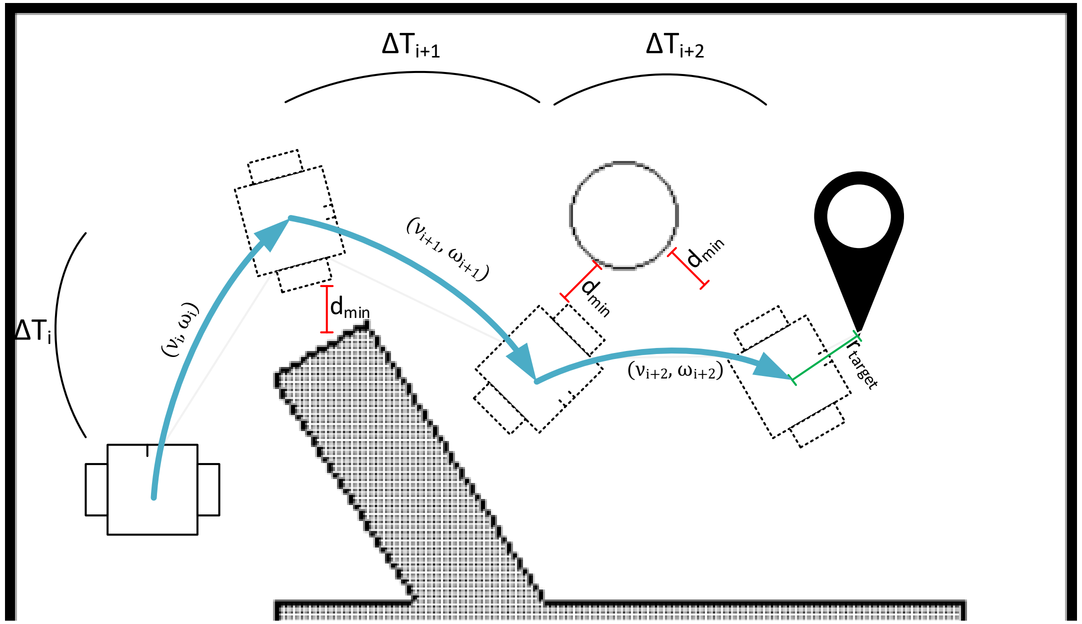
- ❑ Associate states and formulate Objective function

$$f(B) \quad = \quad \sum_k \gamma_k f_k(B)$$

Weighted multiobjective function

$$B^* \quad = \quad \underset{B}{\operatorname{argmin}} f(B)$$

Optimization



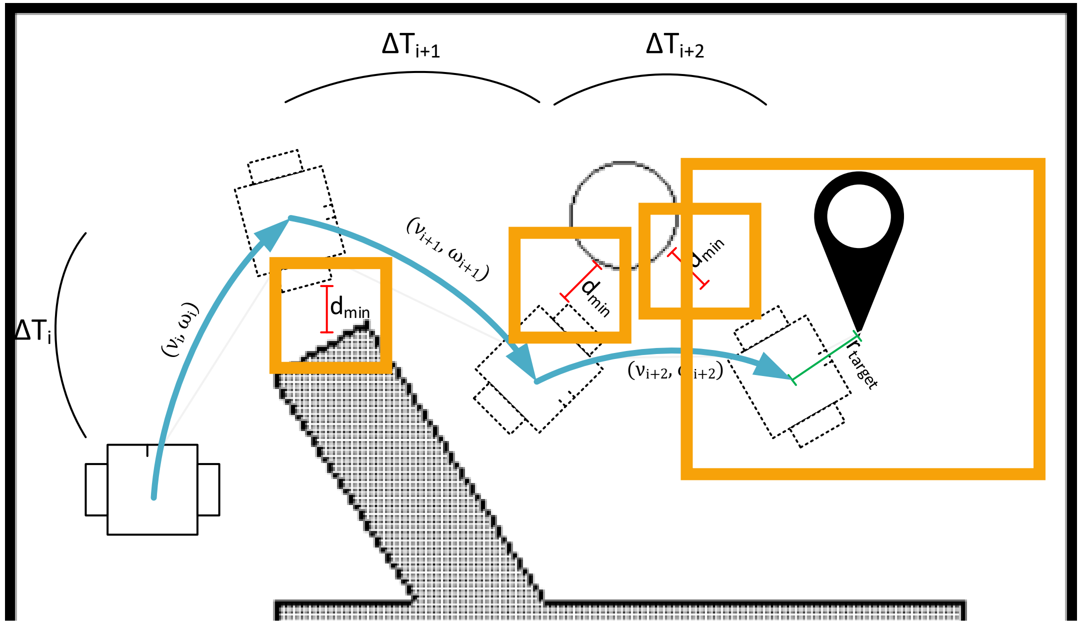
Timed Elastic Band main line

- ❑ Associate states and formulate Objective function

$$f(B) = \sum_k \gamma_k f_k(B)$$

r$_{target}$ $$f_{path} = e_\Gamma(d_{min,j}, r_{p_{max}}, \epsilon, S, n)$$

d$_{min}$ $$f_{ob} = e_\Gamma(-d_{min,j}, -r_{o_{min}}, \epsilon, S, n)$$



Timed Elastic Band main line

- ❑ Associate states and formulate Objective function

$$f(B) = \sum_k \gamma_k f_k(B)$$

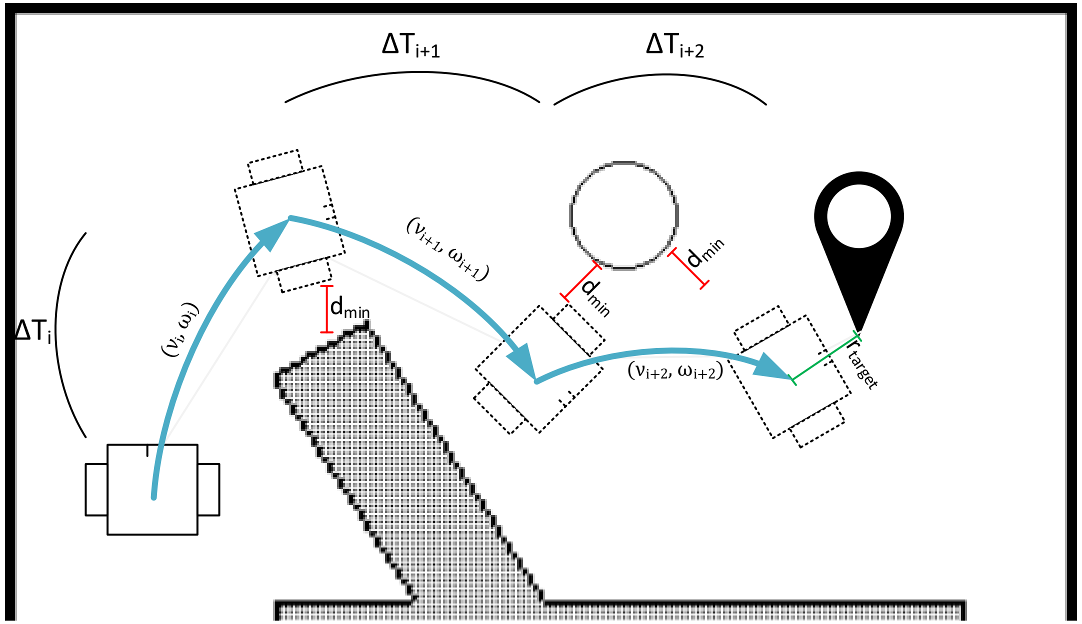
r_{target}

$$f_{path} = e_\Gamma(d_{min,j}, r_{p_{max}}, \epsilon, S, n)$$

d_{min}

$$f_{ob} = e_\Gamma(-d_{min,j}, -r_{o_{min}}, \epsilon, S, n)$$

$$e_\Gamma(x, x_r, \epsilon, S, n) \simeq \begin{cases} \left(\frac{x-(x_r-\epsilon)}{S}\right)^n & \text{if } x > x_r - \epsilon \\ 0 & \text{otherwise} \end{cases}$$



Timed Elastic Band main line

- ❑ Associate states and formulate Objective function

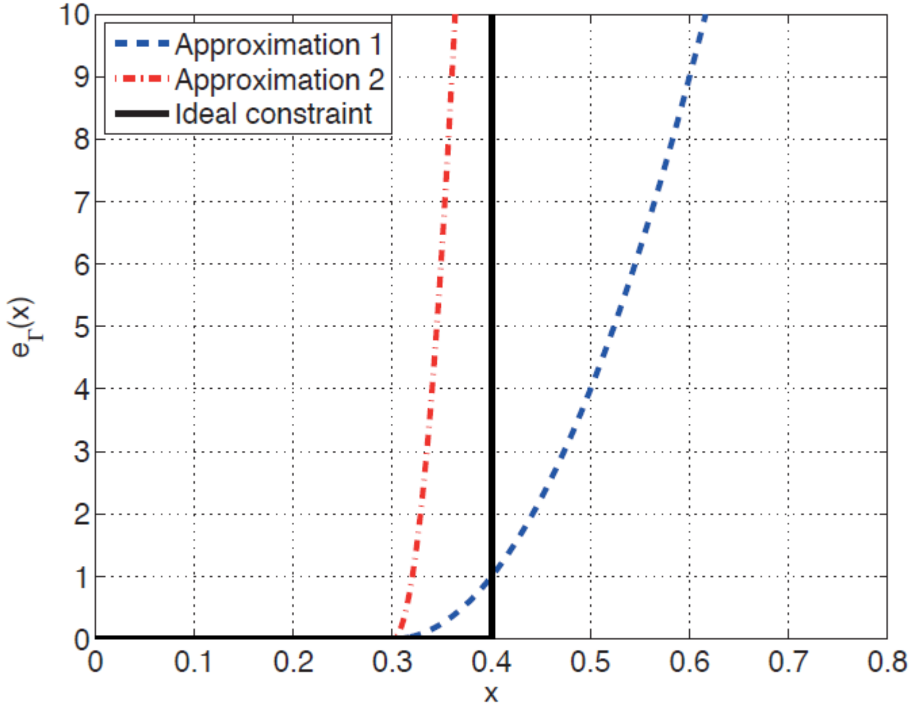
$$f(B) \quad = \quad \sum_k \gamma_k f_k(B)$$

r

r_{target} $$f_{path} \quad = \quad e_\Gamma(d_{min,j}, r_{p_{max}}, \epsilon, S, n)$$

d_{min} $$f_{ob} \quad = \quad e_\Gamma(-d_{min,j}, -r_{o_{min}}, \epsilon, S, n)$$

$$e_\Gamma(x, x_r, \epsilon, S, n) \simeq \begin{cases} \left(\frac{x-(x_r-\epsilon)}{S}\right)^n & \text{if } x > x_r - \epsilon \\ 0 & \text{otherwise} \end{cases}$$



Polynomial approximation of constraints

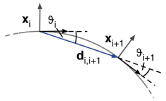
Timed Elastic Band main line

- ❑ Associate states and formulate Objective function

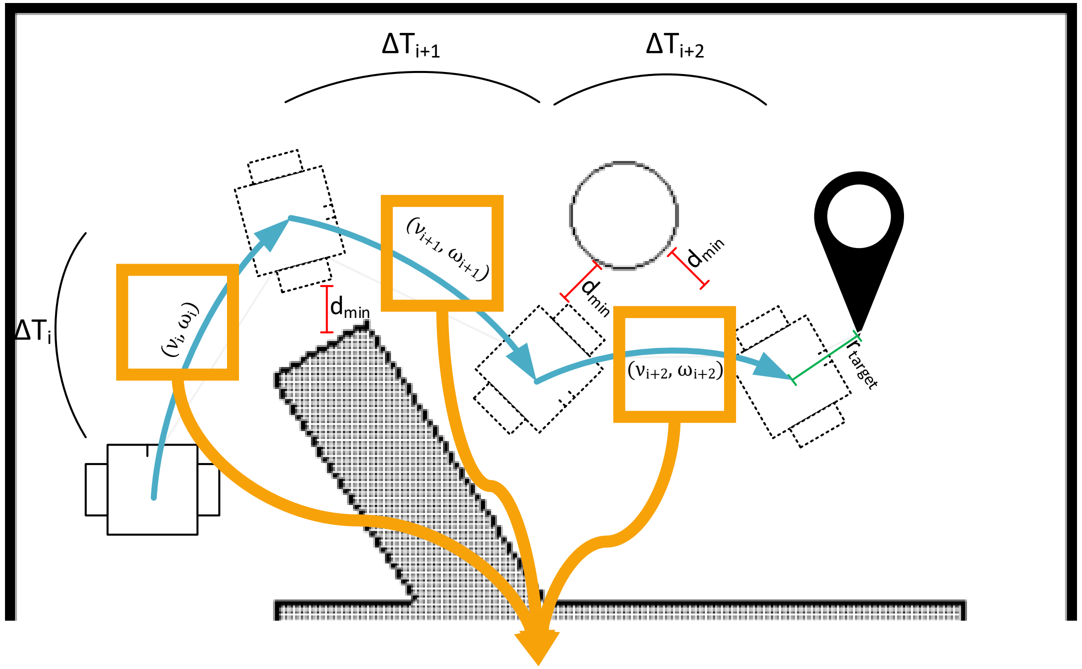
$$f(B) = \sum_k \gamma_k f_k(B)$$

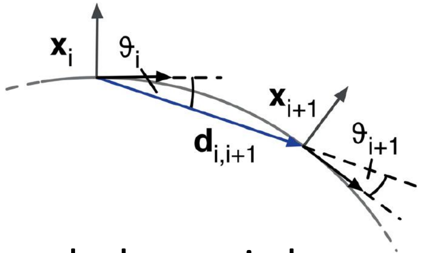
r_{target} $$f_{path} = e_\Gamma(d_{min,j}, r_{p_{max}}, \epsilon, S, n)$$

d_{min} $$f_{ob} = e_\Gamma(-d_{min,j}, -r_{o_{min}}, \epsilon, S, n)$$



$$f_k(\mathbf{x}_i, \mathbf{x}_{i+1}) = \left\| \left[\begin{pmatrix} \cos\beta_i \\ \sin\beta_i \\ 0 \end{pmatrix} + \begin{pmatrix} \cos\beta_{i+1} \\ \sin\beta_{i+1} \\ 0 \end{pmatrix} \right] \times \mathbf{d}_{i,i+1} \right\|^2$$





Non-holonomic kenematics

Timed Elastic Band main line

- ❑ Associate states and formulate Objective function

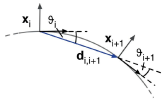
$$f(B) \;=\; \sum_k \gamma_k f_k(B)$$

r_{target}

$$f_{path} \;=\; e_\Gamma(d_{min,j}, r_{p_{max}}, \epsilon, S, n)$$

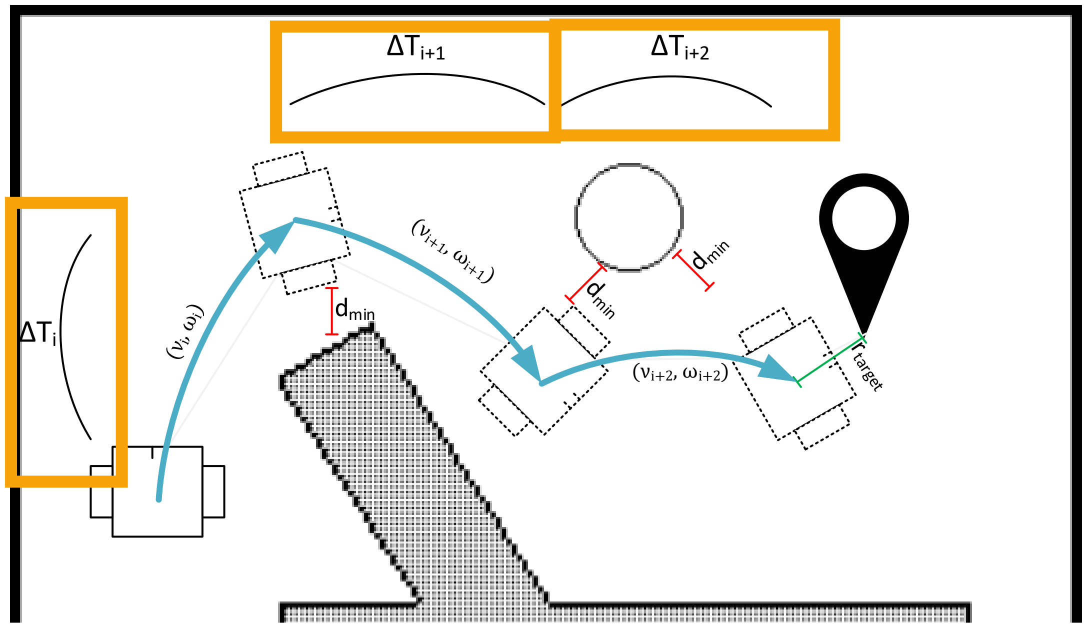
d_{min}

$$f_{ob} \;=\; e_\Gamma(-d_{min,j}, -r_{o_{min}}, \epsilon, S, n)$$



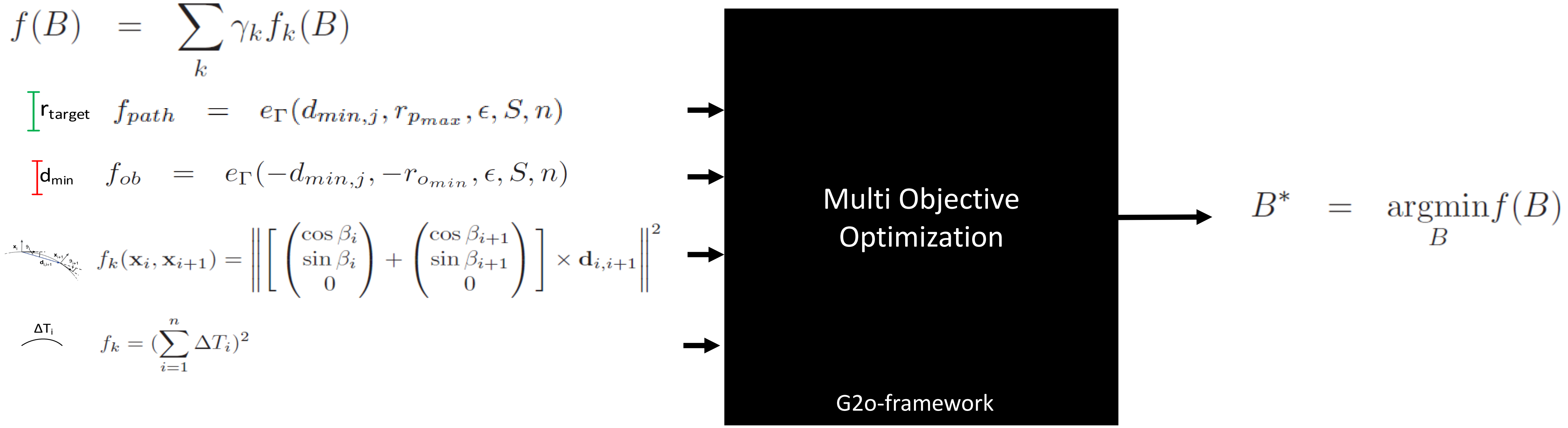
$$f_k(\mathbf{x}_i, \mathbf{x}_{i+1}) = \left\| \left[\begin{pmatrix} \cos\beta_i \\ \sin\beta_i \\ 0 \end{pmatrix} + \begin{pmatrix} \cos\beta_{i+1} \\ \sin\beta_{i+1} \\ 0 \end{pmatrix} \right] \times \mathbf{d}_{i,i+1} \right\|^2$$

$$f_k = (\sum_{i=1}^{n} \Delta T_i)^2$$



Timed Elastic Band main line

- ❑ Optimize functions (Hyper graph)



$$f(B) = \sum_k \gamma_k f_k(B)$$

r_{target} $$f_{path} = e_\Gamma(d_{min,j}, r_{p_{max}}, \epsilon, S, n)$$

d_{min} $$f_{ob} = e_\Gamma(-d_{min,j}, -r_{o_{min}}, \epsilon, S, n)$$

$$f_k(\mathbf{x}_i, \mathbf{x}_{i+1}) = \left\| \left[\begin{pmatrix} \cos\beta_i \\ \sin\beta_i \\ 0 \end{pmatrix} + \begin{pmatrix} \cos\beta_{i+1} \\ \sin\beta_{i+1} \\ 0 \end{pmatrix} \right] \times \mathbf{d}_{i,i+1} \right\|^2$$

ΔT_i $$f_k = (\sum_{i=1}^{n} \Delta T_i)^2$$

Multi Objective Optimization

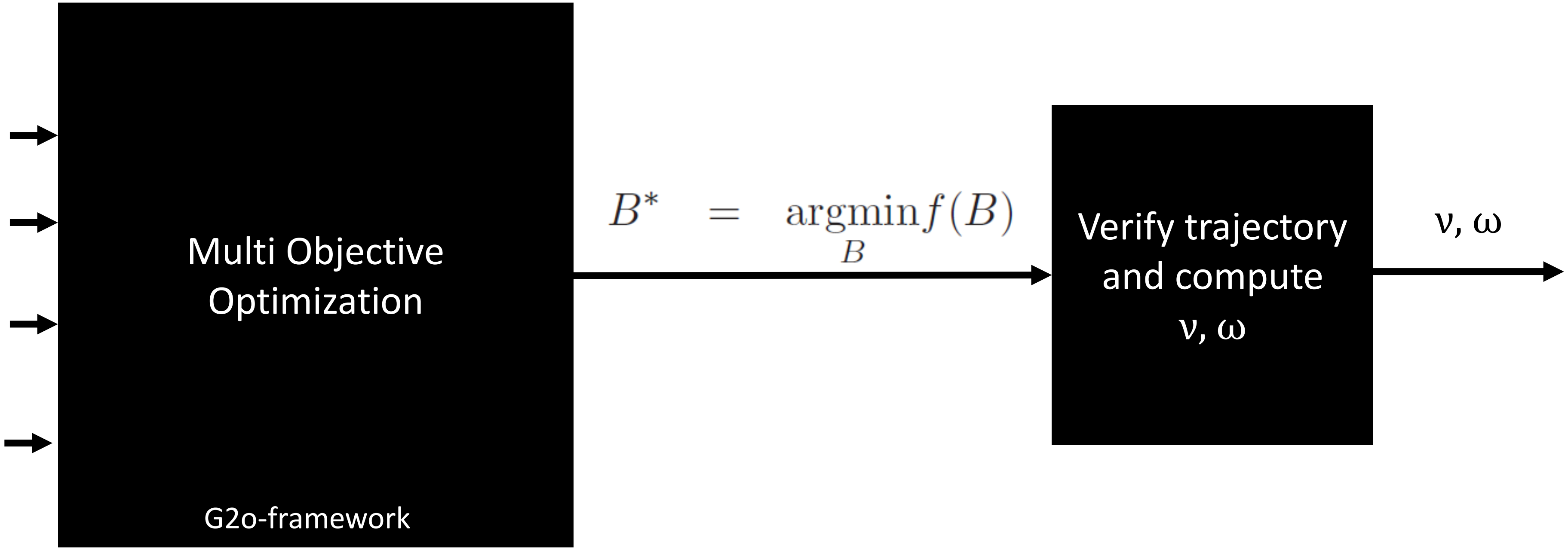
G2o-framework

$$B^* = \underset{B}{\operatorname{argmin}} f(B)$$

Timed Elastic Band main line

- ❑ Optimize functions (Hyper graph)

$$f(B) = \sum_{k} \gamma_k f_k(B)$$

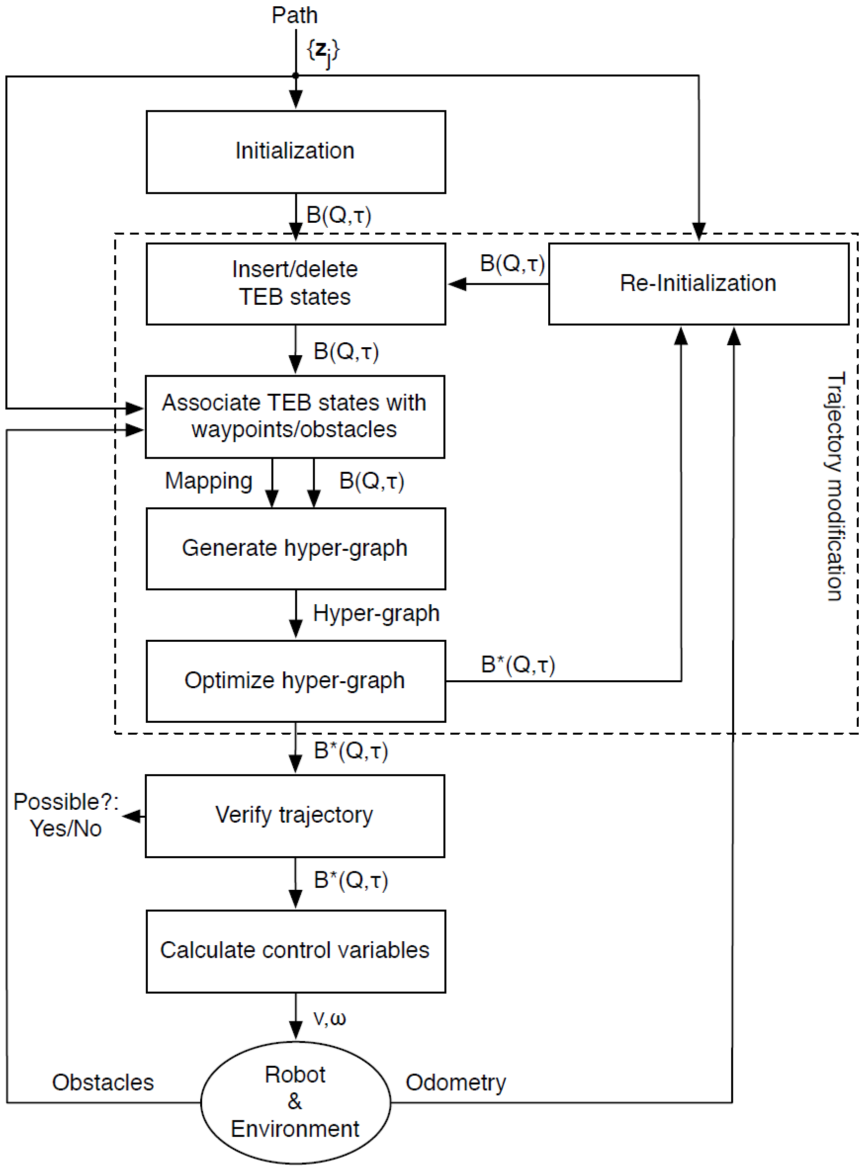


$$B^* = \underset{B}{\operatorname{argmin}} f(B)$$

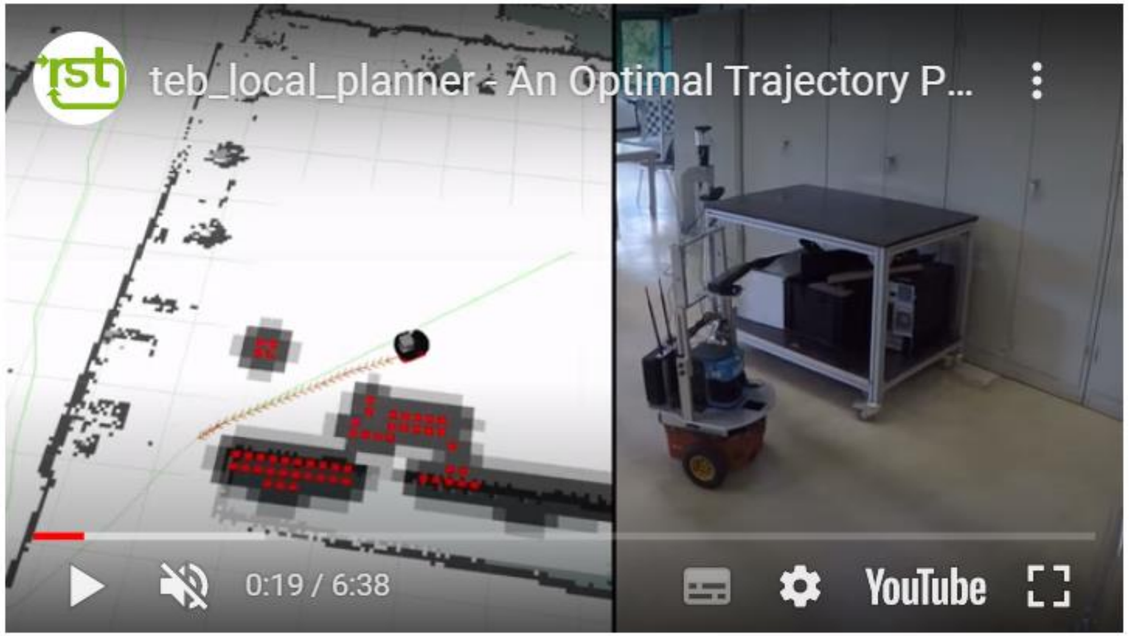
Multi Objective Optimization
G2o-framework
Verify trajectory and compute ν, ω
ν, ω

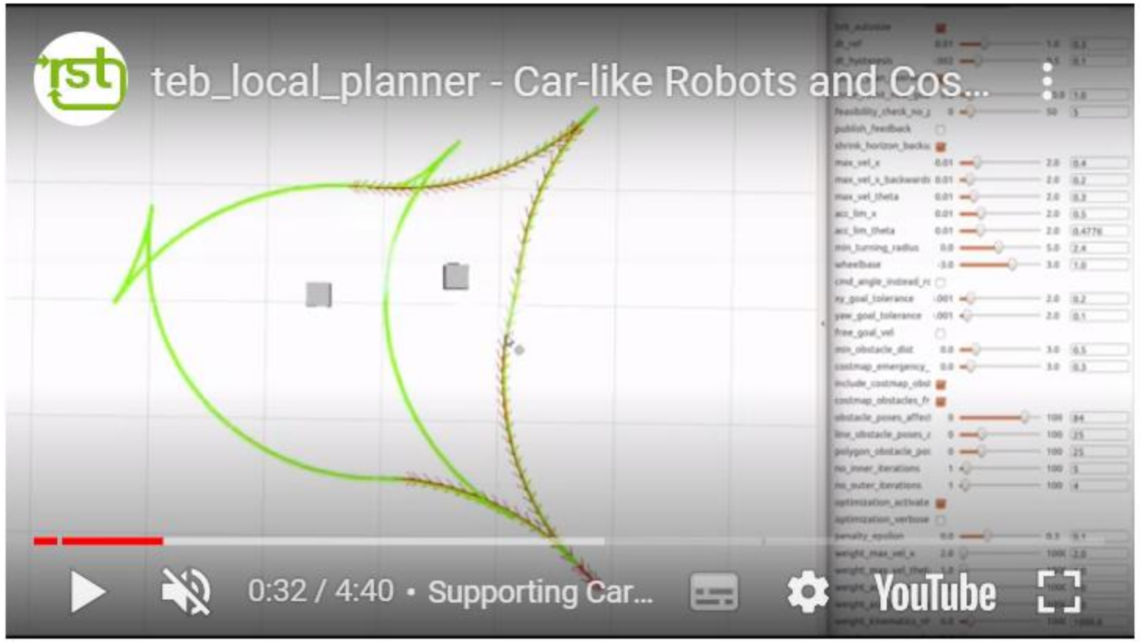
Timed Elastic Band main line

- □ Generate configuration candidates from start to goal
- □ Associate states and formulate Objective function
- □ Optimize functions (Hyper graph)
- □ Compute translational and rotational velocity
- □ [Iterate] Update way point and add/delete new configuration according spacial/temporal resolution to the remaining trajectory



Timed Elastic Band ros




tst
teb_local_planner - Car-like Robots and Cos...
0:32 / 4:40 • Supporting Car...
YouTube

http://wiki.ros.org/teb_local_planner

Pure Pursuit

Work in progress

Pure Pursuit

- ❑ S. Macenski , S. Singh, F. Martin and J. Ginés 2023
- ❑ Mainline
  - ▪ Inspired of Elastic Band but taking into account robot dynamic contraints

Regulated Pure Pursuit for Robot Path Tracking

Steve Macenski[1*], Shrijit Singh[2], Francisco Martín[3] and Jonatan Ginés[3]

[1*]R&D Innovations, Samsung Research America, Clyde Ave, Mountain View, 94043, CA, United States.
[2]Department of Computer Science, Manipal Institute of Technology, Udupi - Karkala Rd, Eshwar Nagar, Manipal, 576104, Karnataka, India.
[3]Intelligent Robotics Lab, Rey Juan Carlos University, Camino del Molino, Fuenlabrada, 28943, Madrid, Spain.

*Corresponding author(s). E-mail(s): s.macenski@samsung.com;
Contributing authors: shrijit.singh@learner.manipal.edu; francisco.rico@urjc.es; jonatan.gines@urjc.es;

Abstract

The accelerated deployment of service robots have spawned a number of algorithm variations to better handle real-world conditions. Many local trajectory planning techniques have been deployed on practical robot systems successfully. While most formulations of Dynamic Window Approach and Model Predictive Control can progress along paths and optimize for additional criteria, the use of pure path tracking algorithms is still commonplace. Decades later, Pure Pursuit and its variants continues to be one of the most commonly utilized classes of local trajectory planners. However, few Pure Pursuit variants have been proposed with schema for variable linear velocities - they either assume a constant velocity or fails to address the point at all. This paper presents a variant of Pure Pursuit designed with additional heuristics to regulate linear velocities, built atop the existing Adaptive variant. The *Regulated Pure Pursuit algorithm* makes incremental improvements on state of the art by adjusting linear velocities with particular focus on safety in constrained and partially observable spaces commonly negotiated by deployed robots. We present experiments with the Regulated Pure Pursuit algorithm on industrial-grade service robots. We also provide a high-quality reference implementation that is freely included ROS 2 Nav2 framework at https://github.com/ros-planning/navigation2 for fast evaluation.

Keywords: Service Robots, Mobile Robots, Motion Planning, Path Planning

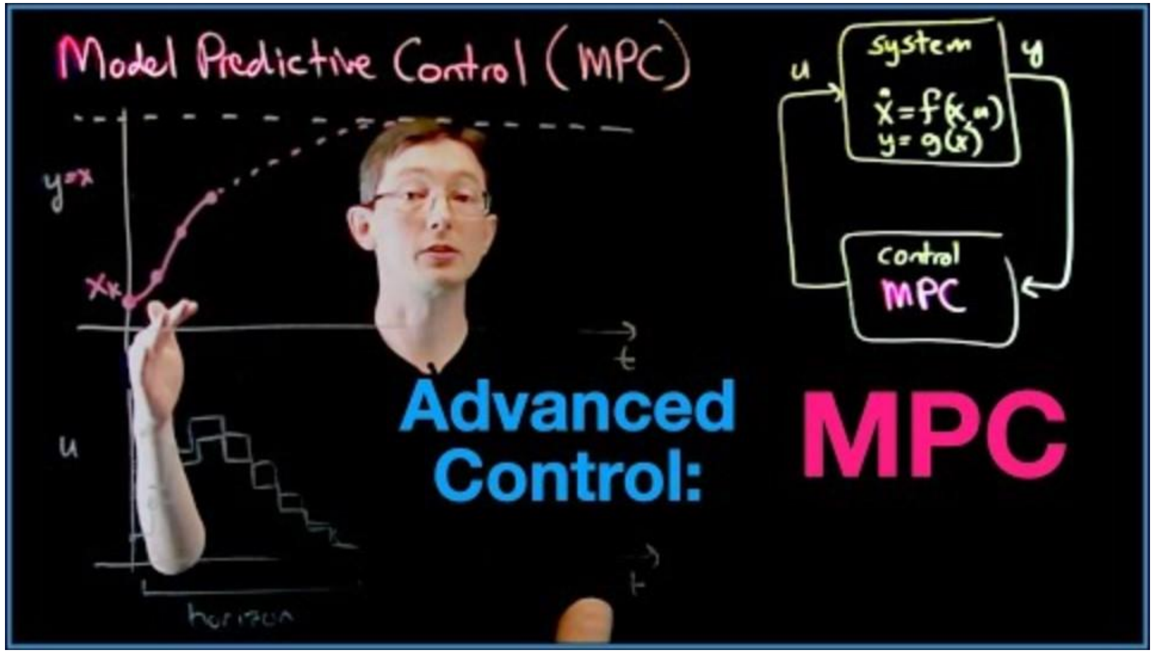
1 Introduction

Dynamic Window Approach (DWA) [1], Pure Pursuit (PP) [6], and Model Predictive Control (MPC) [2] are by far the most commonly deployed path trackers. They all have a strong heritage for reliability in a wide range of environmental conditions. DWA and MPC are often, but not always, formulated as multi-objective trajectory generation problems to maximize criteria such as avoiding dynamic obstacle collisions on top of path tracking. This has made them particularly well suited for many robotics applications where dynamic robot behaviors are rewarded. A great deal of work has been conducted on these allowing

Model Predictive Path Integral Controller (MPPI)

Model Predictive Control MPC

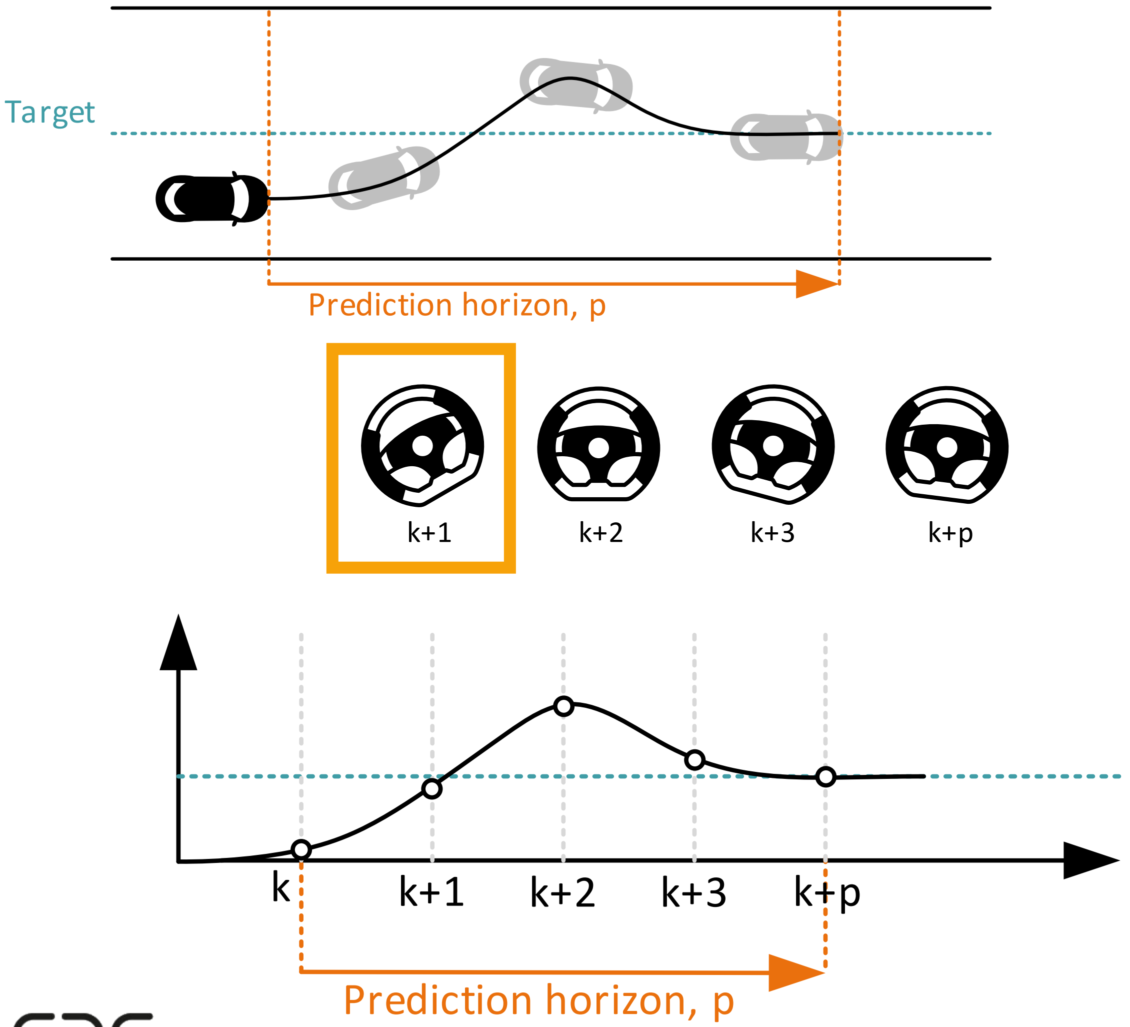
❑ Course references and pictures inspired of

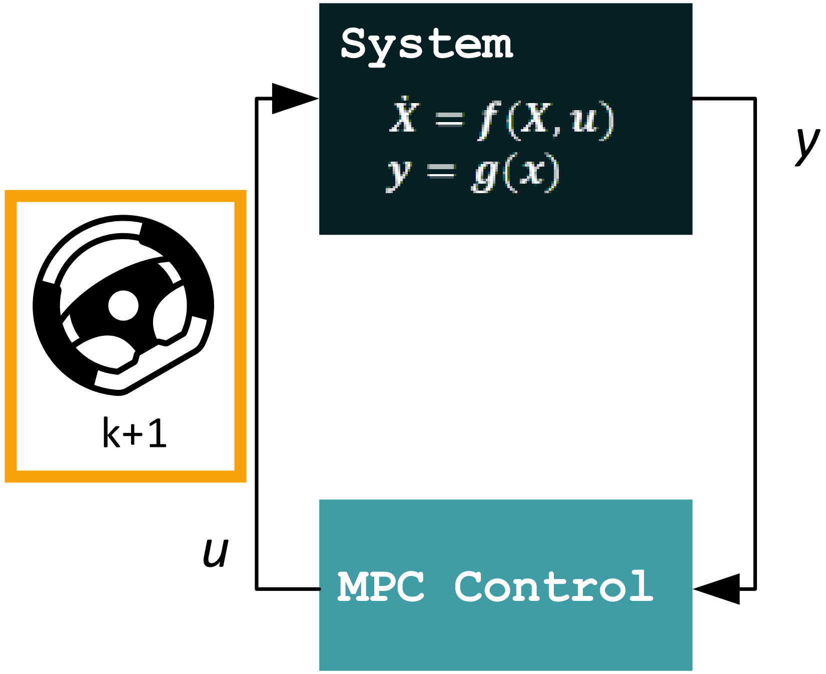


MPC: Steve L. Brunton, University of Washington
https://www.youtube.com/watch?v=YwodGM2eoy4

Understanding MPC: Melda Ulusoy, MathWorks
https://www.youtube.com/watch?v=cEWnixjNdzs

Model Predictive Control MPC





$$\boxed{K(X_k)} = u_{k+1}(X_k)$$

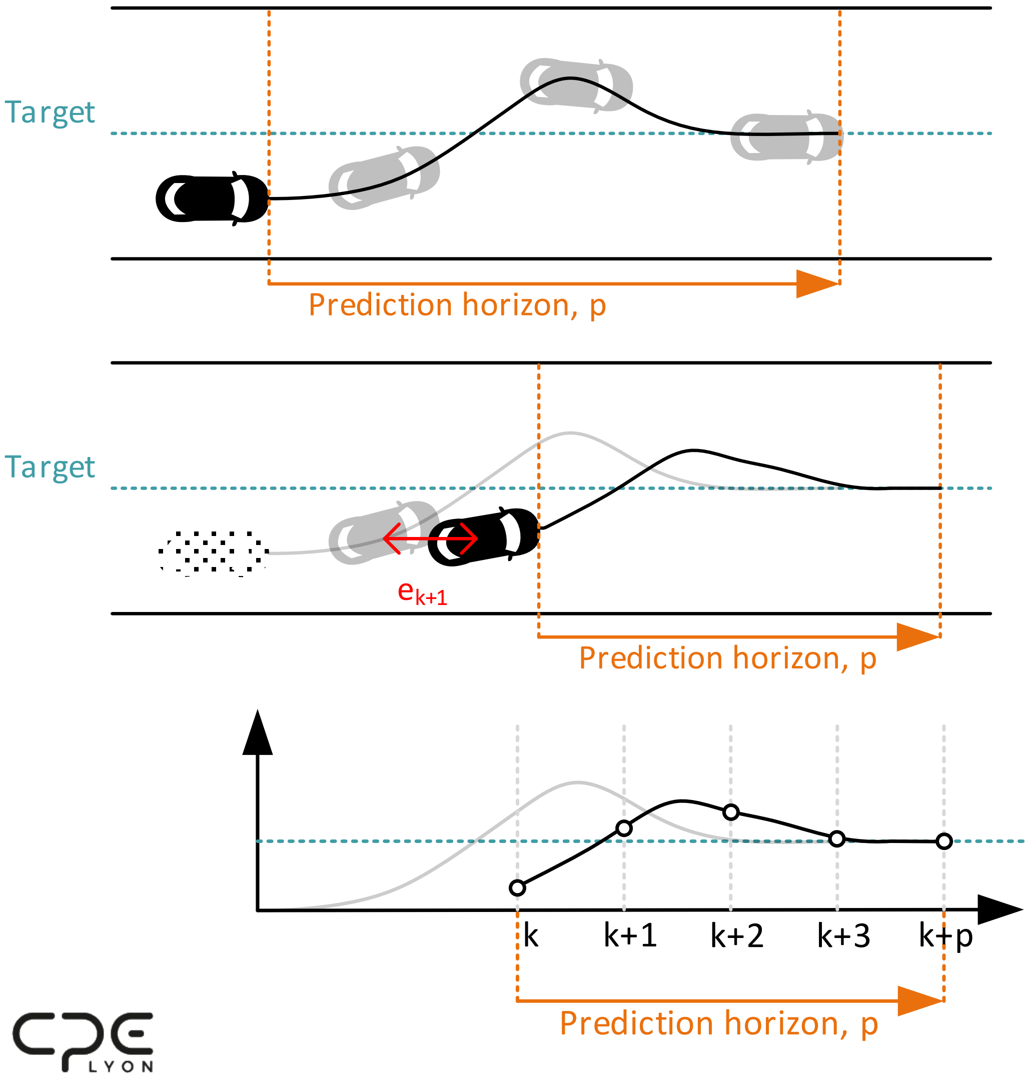
Control Signal

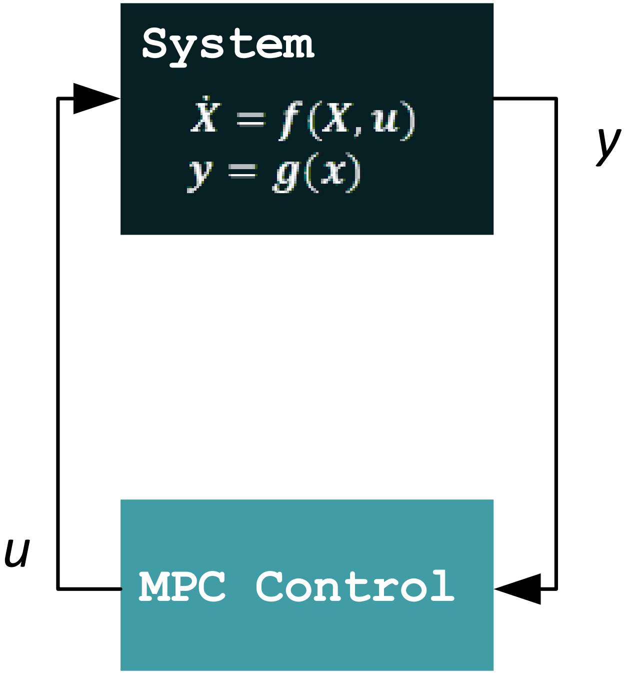
First command of the computed Control strategy

State vector at t=k

assuming y=x

Model Predictive Control MPC





$$K(X_k) = u_{k+1}(X_k)$$

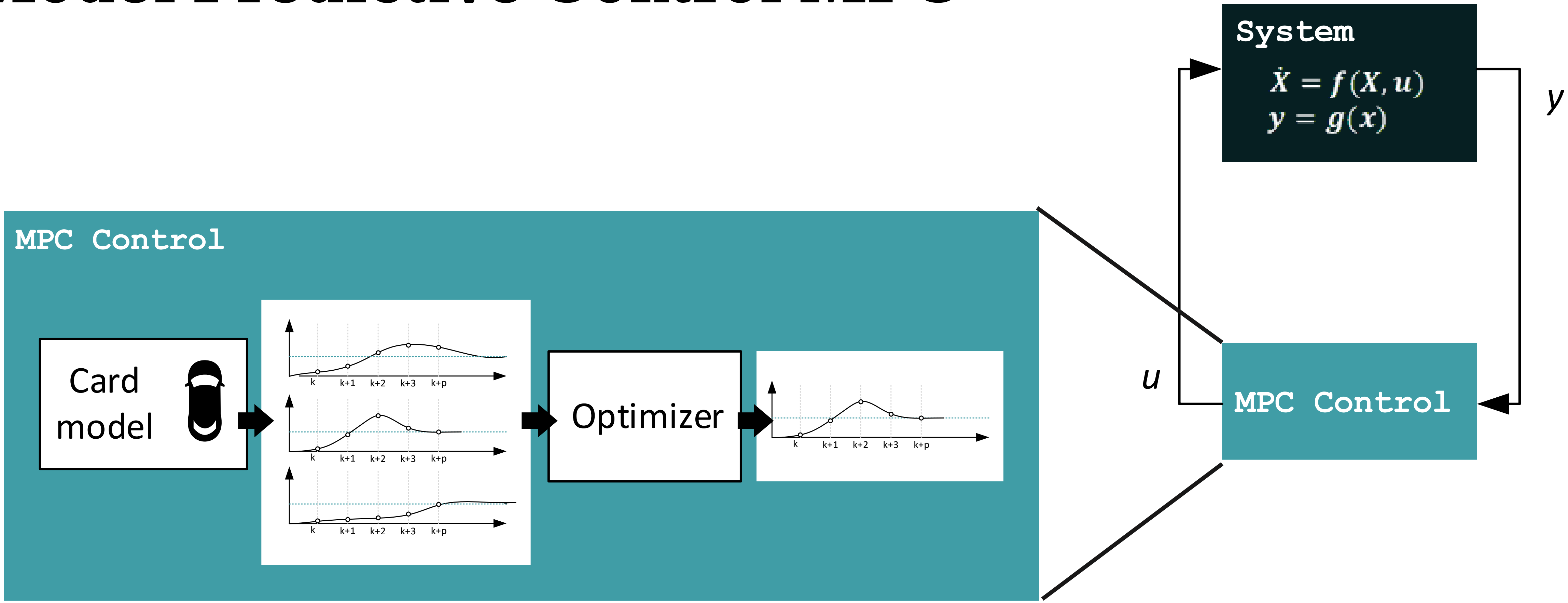
Control Signal

First command of the computed Control strategy

State vector at t=k

assuming y=x

Model Predictive Control MPC



System

$$\dot{X} = f(X, u)$$
$$y = g(x)$$

MPC Control

Card model

Optimizer

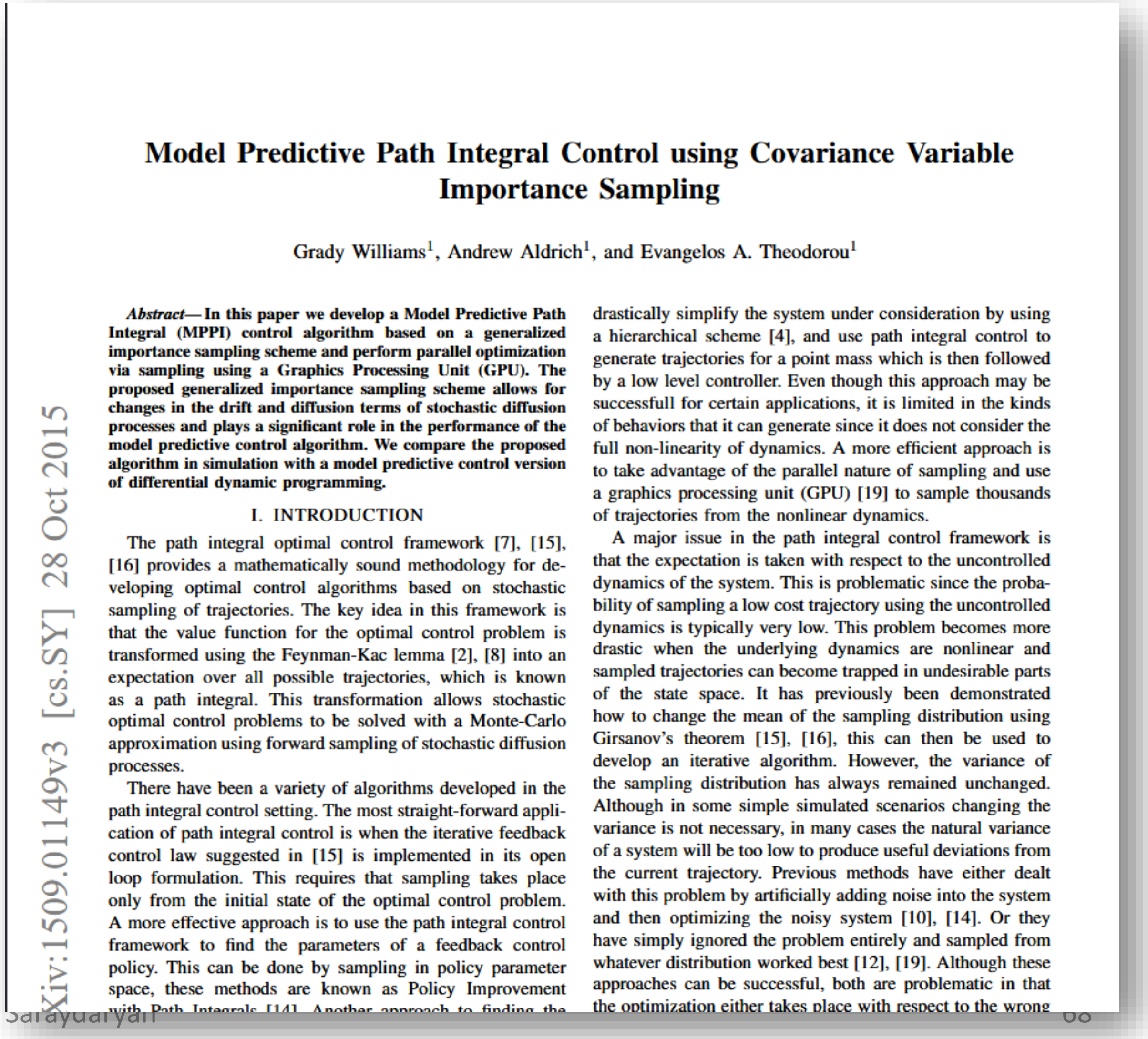
MPC Control

Model Predictive Control MPC

- ❑Advantages
  - ▪ Constraints can be applied
    - On command (output)
    - On state of the system (input)
  - ▪ Work on non-linear systems
- ❑ Hardwork
  - ▪ Optimization at each steps
  - ▪ Expensive

Model Predictive Path Integral Controller

- ❑G. Williams, A. Aldrich, and E. A. Theodorou1 2015
- ❑ Mainline
  - MPC Approach
  - New System representation
  - Stochastic Trajectory Optimization



Model Predictive Path Integral Control using Covariance Variable Importance Sampling

Grady Williams[1], Andrew Aldrich[1], and Evangelos A. Theodorou[1]

***Abstract*— In this paper we develop a Model Predictive Path Integral (MPPI) control algorithm based on a generalized importance sampling scheme and perform parallel optimization via sampling using a Graphics Processing Unit (GPU). The proposed generalized importance sampling scheme allows for changes in the drift and diffusion terms of stochastic diffusion processes and plays a significant role in the performance of the model predictive control algorithm. We compare the proposed algorithm in simulation with a model predictive control version of differential dynamic programming.**

I. INTRODUCTION

The path integral optimal control framework [7], [15], [16] provides a mathematically sound methodology for developing optimal control algorithms based on stochastic sampling of trajectories. The key idea in this framework is that the value function for the optimal control problem is transformed using the Feynman-Kac lemma [2], [8] into an expectation over all possible trajectories, which is known as a path integral. This transformation allows stochastic optimal control problems to be solved with a Monte-Carlo approximation using forward sampling of stochastic diffusion processes.

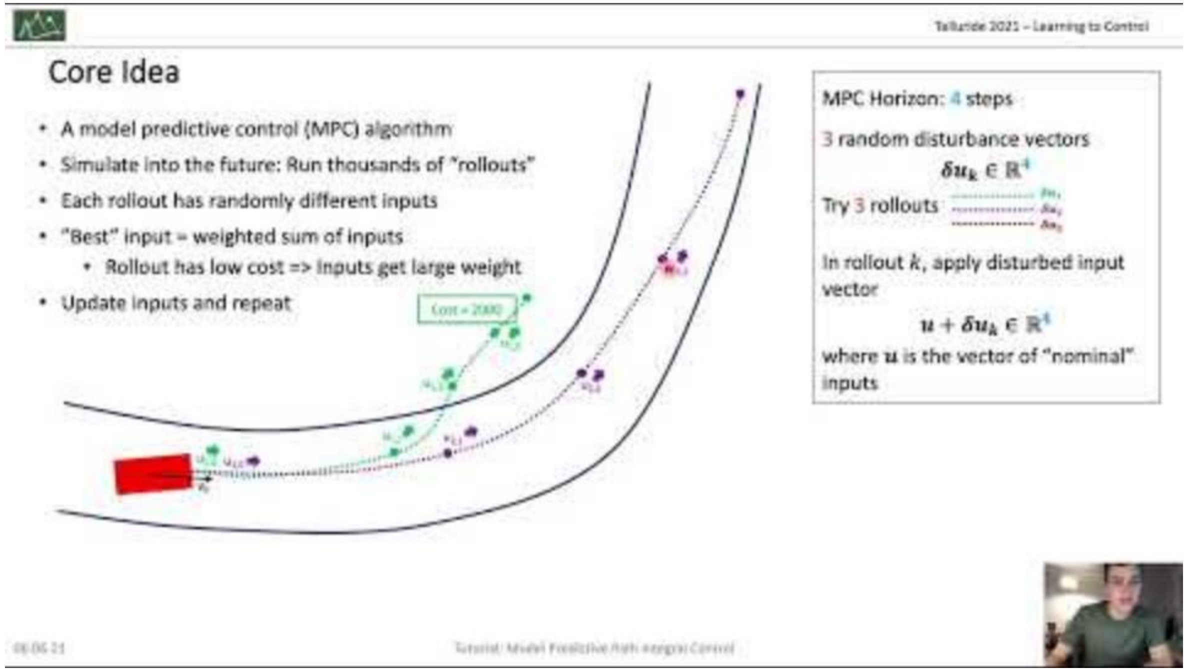
There have been a variety of algorithms developed in the path integral control setting. The most straight-forward application of path integral control is when the iterative feedback control law suggested in [15] is implemented in its open loop formulation. This requires that sampling takes place only from the initial state of the optimal control problem. A more effective approach is to use the path integral control framework to find the parameters of a feedback control policy. This can be done by sampling in policy parameter space, these methods are known as Policy Improvement with Path Integrals [14]. Another approach to finding the

drastically simplify the system under consideration by using a hierarchical scheme [4], and use path integral control to generate trajectories for a point mass which is then followed by a low level controller. Even though this approach may be successfull for certain applications, it is limited in the kinds of behaviors that it can generate since it does not consider the full non-linearity of dynamics. A more efficient approach is to take advantage of the parallel nature of sampling and use a graphics processing unit (GPU) [19] to sample thousands of trajectories from the nonlinear dynamics.

A major issue in the path integral control framework is that the expectation is taken with respect to the uncontrolled dynamics of the system. This is problematic since the probability of sampling a low cost trajectory using the uncontrolled dynamics is typically very low. This problem becomes more drastic when the underlying dynamics are nonlinear and sampled trajectories can become trapped in undesirable parts of the state space. It has previously been demonstrated how to change the mean of the sampling distribution using Girsanov's theorem [15], [16], this can then be used to develop an iterative algorithm. However, the variance of the sampling distribution has always remained unchanged. Although in some simple simulated scenarios changing the variance is not necessary, in many cases the natural variance of a system will be too low to produce useful deviations from the current trajectory. Previous methods have either dealt with this problem by artificially adding noise into the system and then optimizing the noisy system [10], [14]. Or they have simply ignored the problem entirely and sampled from whatever distribution worked best [12], [19]. Although these approaches can be successful, both are problematic in that the optimization either takes place with respect to the wrong

Model Predictive Path Integral Controller

- ❑Course references and pictures inspired of



Telluride 2021 – Learning to Control

Core Idea

- A model predictive control (MPC) algorithm
- Simulate into the future: Run thousands of "rollouts"
- Each rollout has randomly different inputs
- "Best" input = weighted sum of inputs
  - Rollout has low cost => inputs get large weight
- Update inputs and repeat

MPC Horizon: 4 steps

3 random disturbance vectors

$$\delta u_k \in \mathbb{R}^4$$

Try 3 rollouts

In rollout k, apply disturbed input vector

$$u + \delta u_k \in \mathbb{R}^4$$

where u is the vector of "nominal" inputs

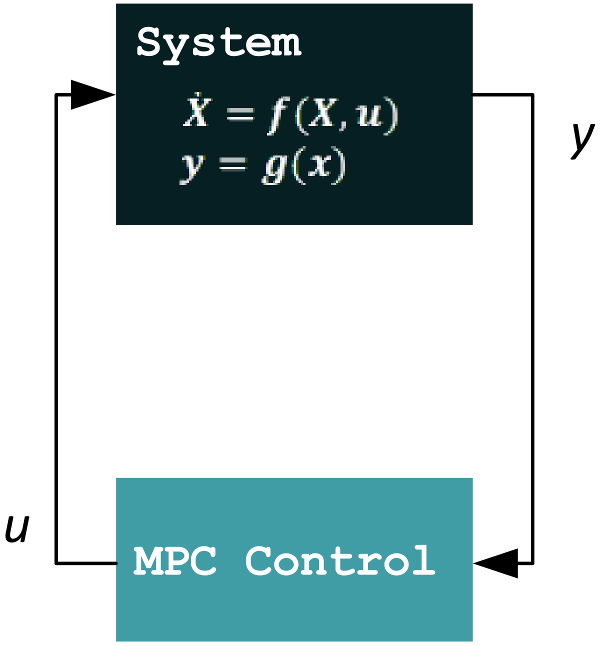
06.06.21

Tutorial: Model Predictive Path Integral Control

MPPI an introduction tutorial: F. Heetmeyer, M. Paluch, Telluride 2021
https://www.youtube.com/watch?v=19QLyMuQ_BE

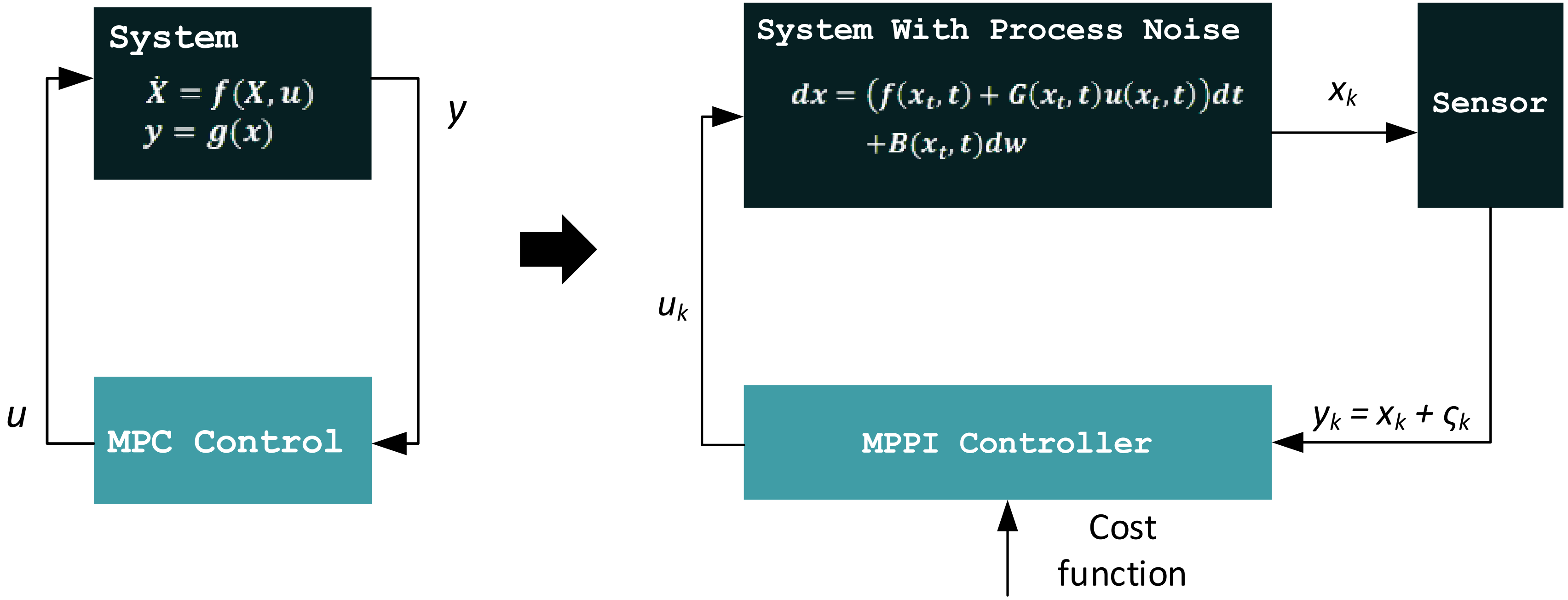
MPPI

- ❑ Core principle
  - MPC Algorithm
  - Simulate into the future thousand of trajectories
  - Each trajectory has a randomly different command
  - Selected command → Weighted sum of commands
  - Update command and repeat



MPPI

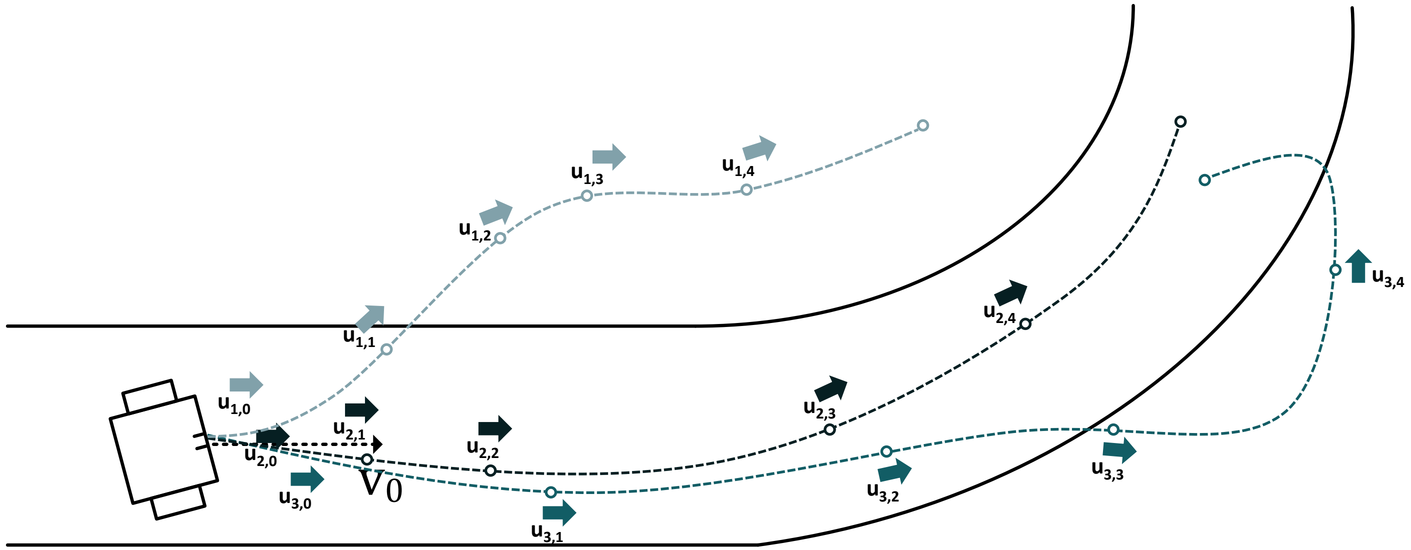
- ❑ Core principle

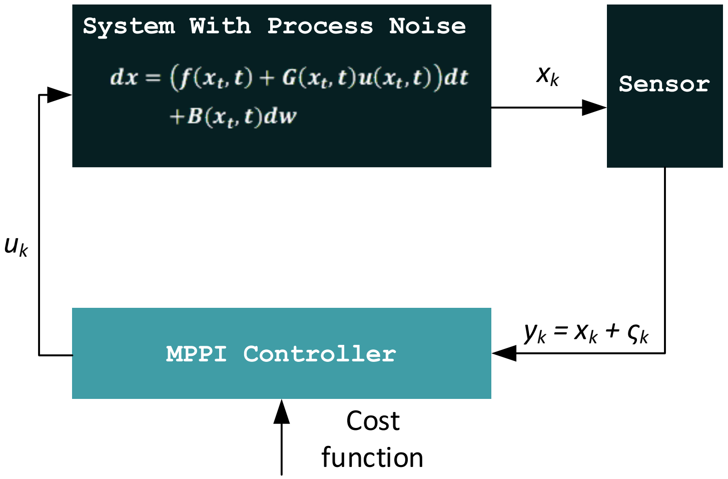


$$dx = \left(\overset{\text{non linear Term}}{f(x_t,t)} + \underbrace{G(x_t,t)u(x_t,t)}_{\text{Affine Control}}\right)dt + \overset{\text{Affine Disturbance}}{B(x_t,t)dw}$$

MPPI

❑ Core principle





System With Process Noise

$$dx = \big(f(x_t,t) + G(x_t,t)u(x_t,t)\big)dt + B(x_t,t)dw$$

x_k

Sensor

u_k

$y_k = x_k + \varsigma_k$

MPPI Controller

Cost function

- Compute candidate command series

- Each candidate command series is equal to :

$$u_{n,t} \sim N(u_t, \Sigma_n)$$

u_t Is the nominal vector

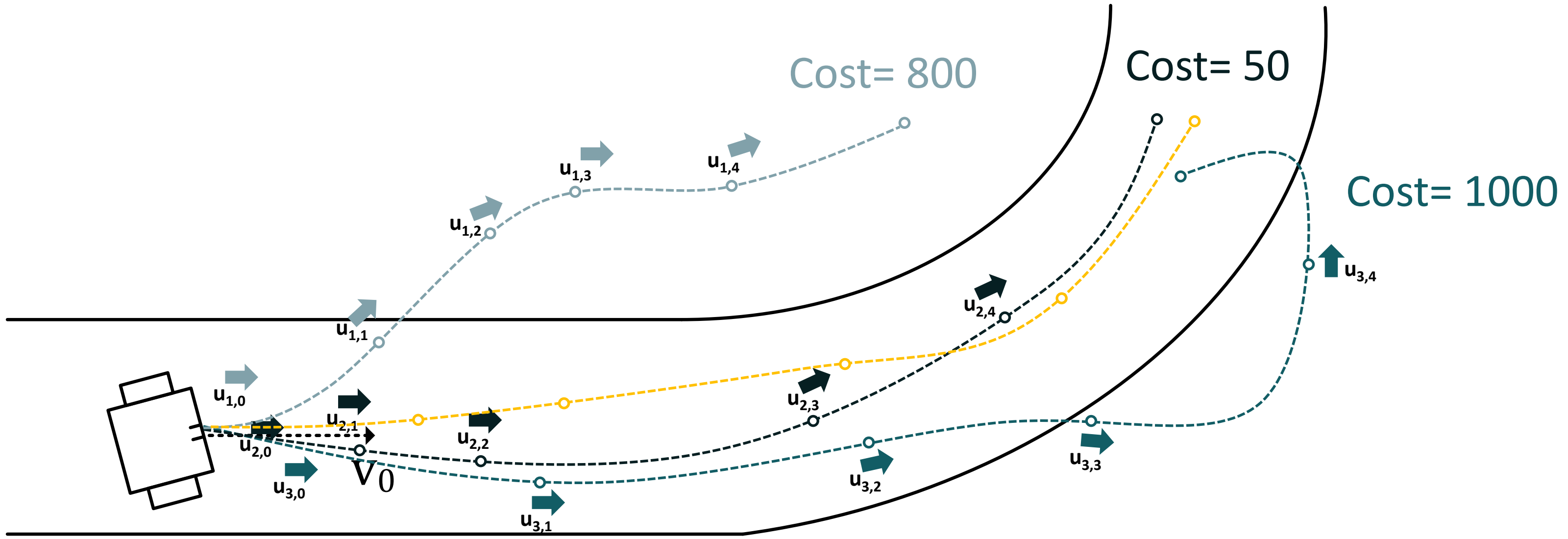
- - - - - - - - - - $u_{1,t} \sim N(u_t, \Sigma_1)$

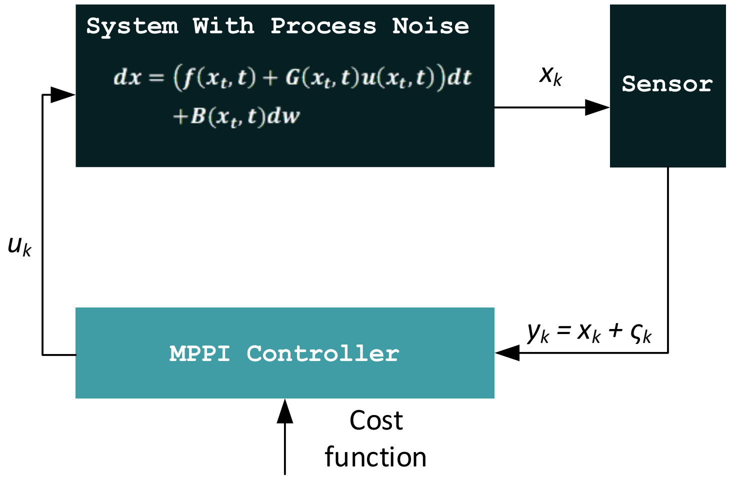
- - - - - - - - - - $u_{2,t} \sim N(u_t, \Sigma_2)$

- - - - - - - - - - $u_{3,t} \sim N(u_t, \Sigma_3)$

MPPI

❑ Core principle





System With Process Noise

$$dx = \left(f(x_t,t) + G(x_t,t)u(x_t,t)\right)dt + B(x_t,t)dw$$

Sensor

MPPI Controller

Cost function

- Compute candidate command series

- Score each command series

- Compute new command series

Final selected command is :

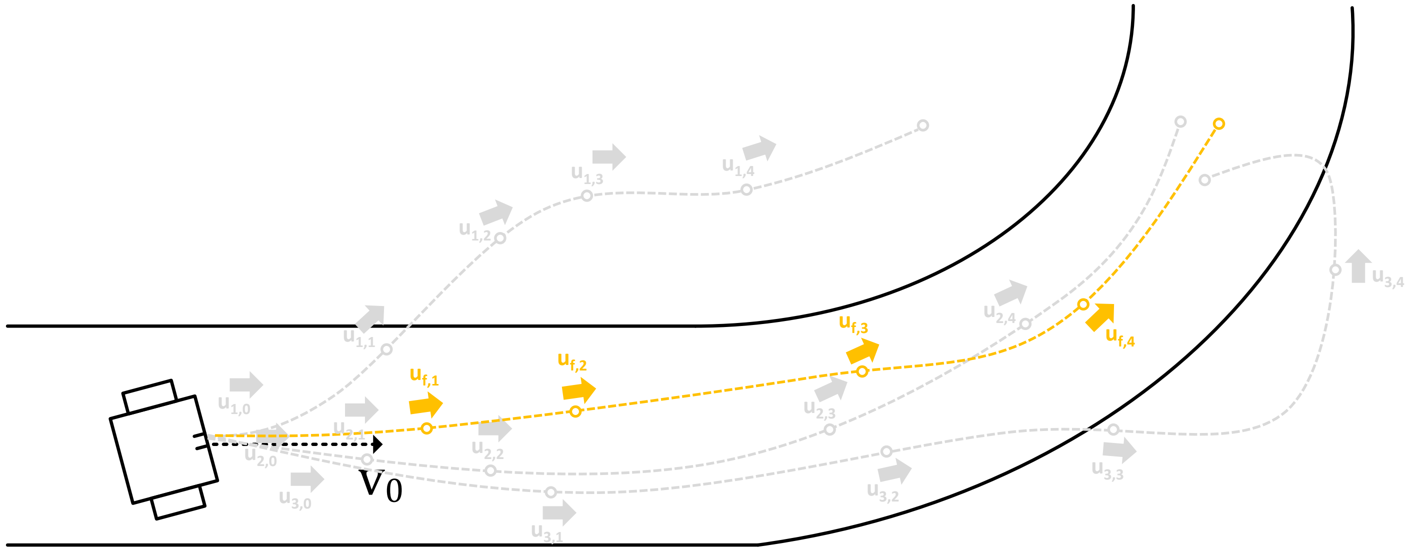
$$u_t = u_t + \frac{\Sigma_k w_k \delta u_k}{\Sigma_k w_k}$$

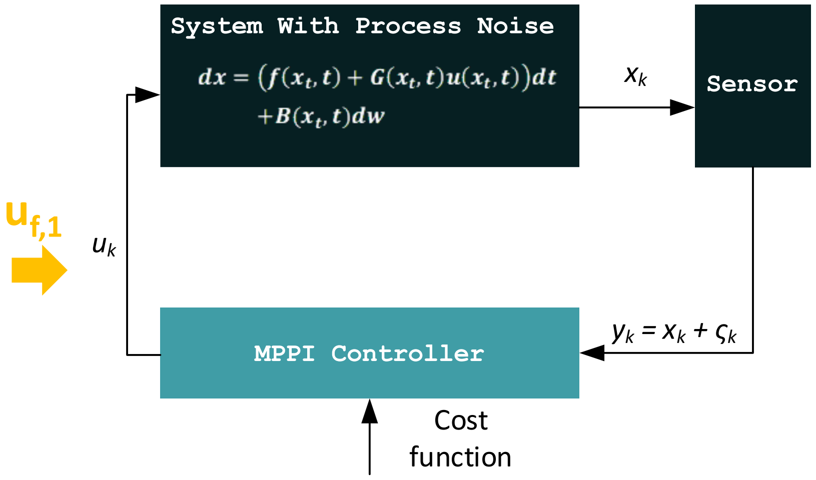
$$w_k = e^{-\frac{1}{\lambda}S_k}$$

S_k Is the score of the kth series

MPPI

❑ Core principle





System With Process Noise

$$dx = (f(x_t,t) + G(x_t,t)u(x_t,t))dt + B(x_t,t)dw$$

x_k

Sensor

$y_k = x_k + ς_k$

MPPI Controller

Cost function

u_k

$u_{f,1}$

- Compute candidate command series
- Score each command series
- Compute new command serie

Final selected command is :

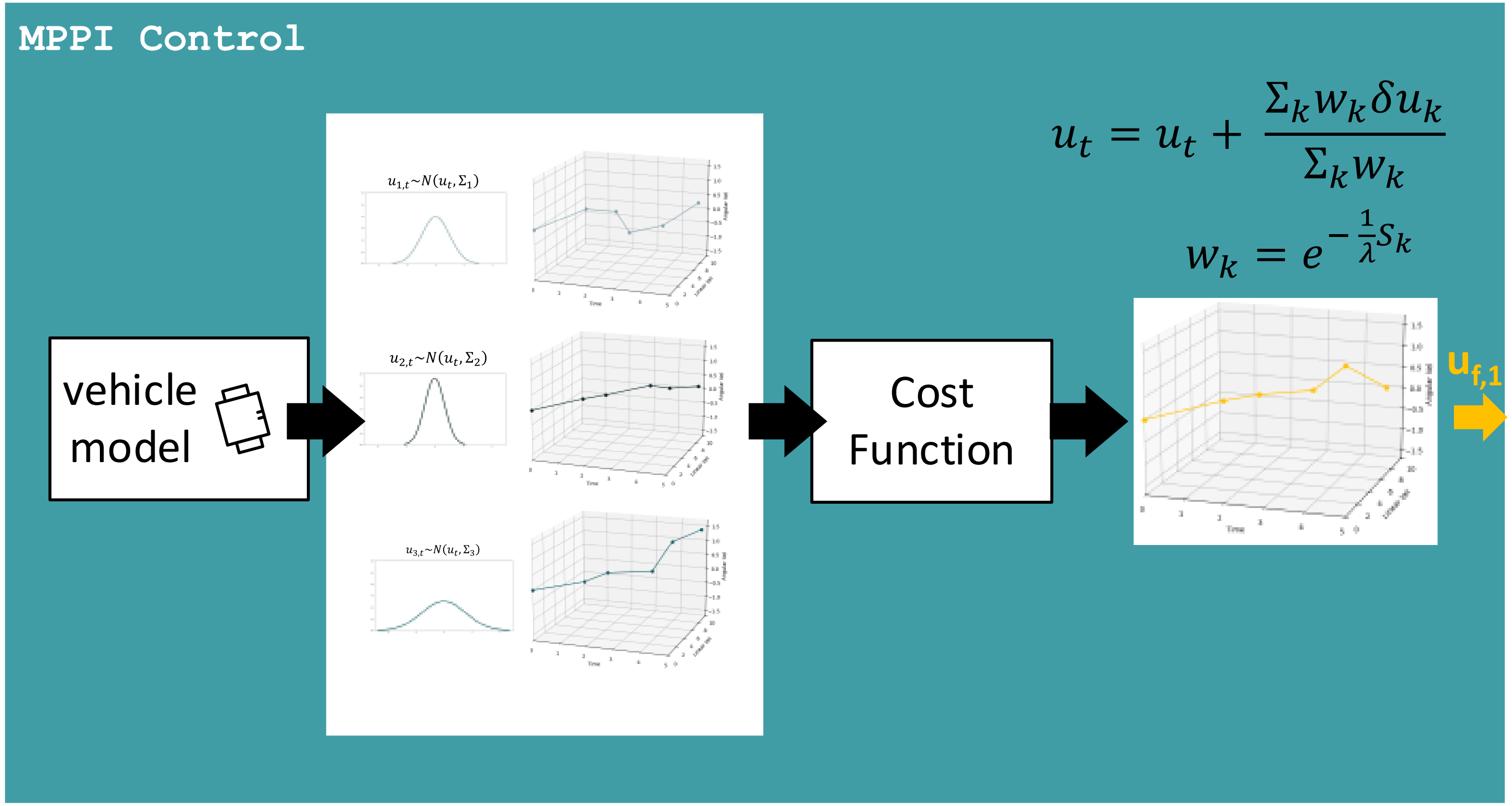
$$u_t = u_t + \frac{\Sigma_k w_k \delta u_k}{\Sigma_k w_k}$$

$$w_k = e^{-\frac{1}{\lambda}S_k}$$

S_k Is the score of the kth series

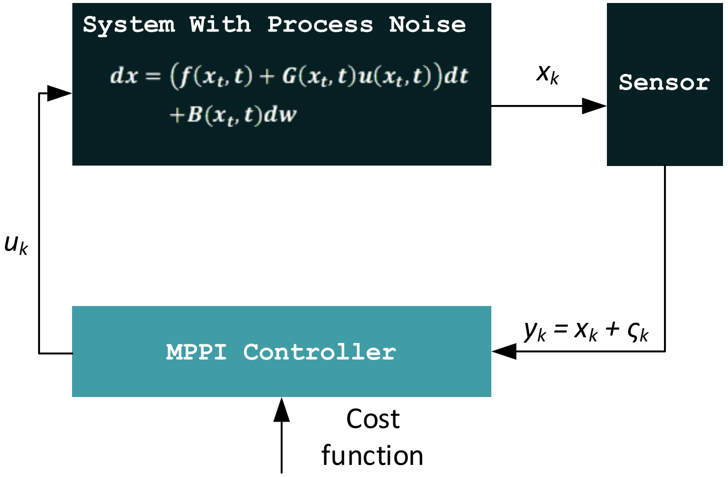
MPPI

❑ Core principle


MPPI Control
vehicle model
$u_{1,t} \sim N(u_t, \Sigma_1)$
$u_{2,t} \sim N(u_t, \Sigma_2)$
$u_{3,t} \sim N(u_t, \Sigma_3)$
Cost Function
$u_{f,1}$

$$u_t = u_t + \frac{\Sigma_k w_k \delta u_k}{\Sigma_k w_k}$$

$$w_k = e^{-\frac{1}{\lambda} S_k}$$


System With Process Noise
$dx = (f(x_t,t) + G(x_t,t)u(x_t,t))dt$
$+B(x_t,t)dw$
x_k
Sensor
u_k
MPPI Controller
$y_k = x_k + \varsigma_k$
Cost function

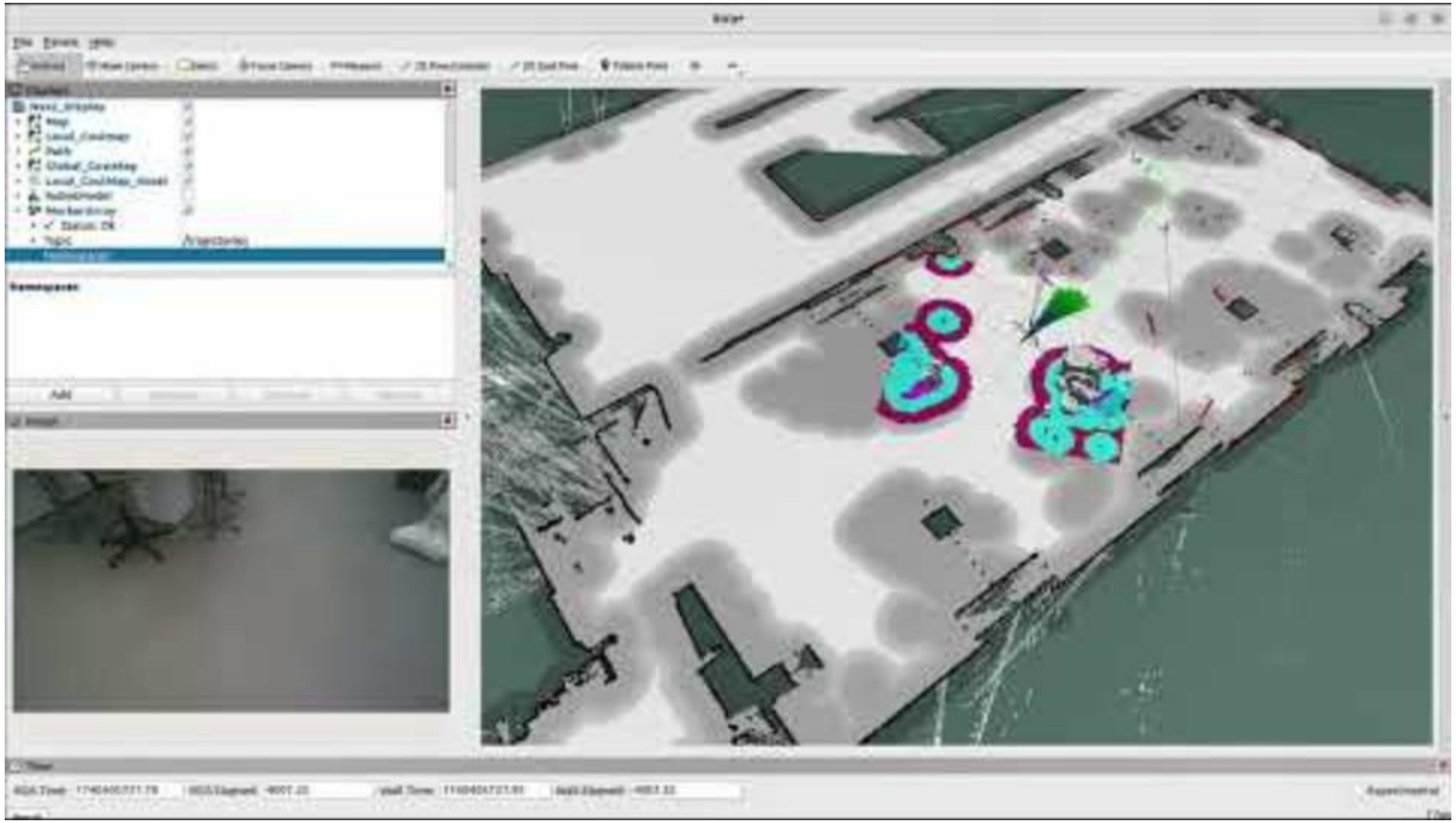
- Each candidate command series is equal to :

- Score each command series

- Compute new command serie

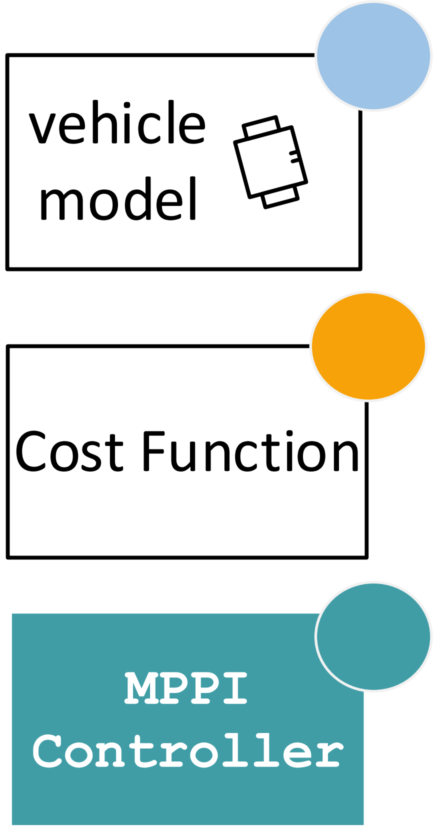
MPPI

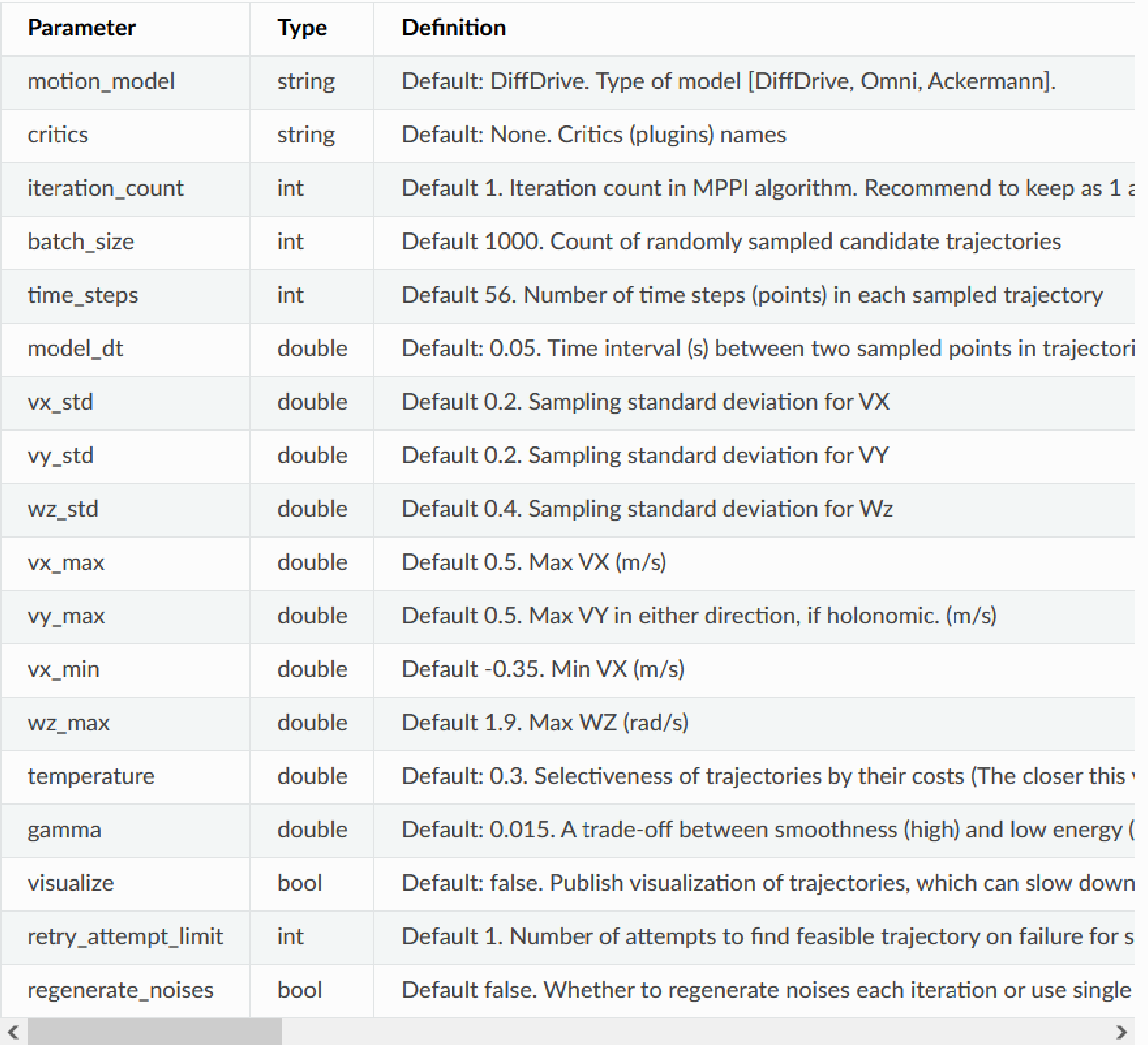
- ❑ Running examples



MPPI and ROS NAV2

- ❑ Configuration and Tuning
  - e.g Nav2 MPPI Controller





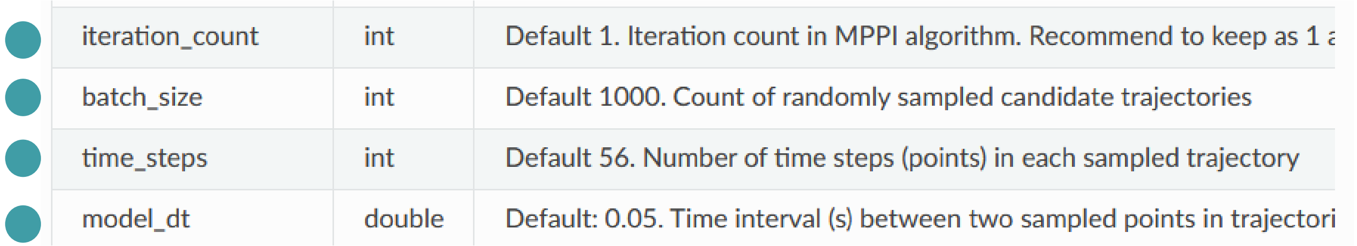
	Parameter	Type	Definition
🔵	motion_model	string	Default: DiffDrive. Type of model [DiffDrive, Omni, Ackermann].
🟠	critics	string	Default: None. Critics (plugins) names
🟢	iteration_count	int	Default 1. Iteration count in MPPI algorithm. Recommend to keep as 1 a
🟢	batch_size	int	Default 1000. Count of randomly sampled candidate trajectories
🟢	time_steps	int	Default 56. Number of time steps (points) in each sampled trajectory
🟢	model_dt	double	Default: 0.05. Time interval (s) between two sampled points in trajectori
🔵	vx_std	double	Default 0.2. Sampling standard deviation for VX
🔵	vy_std	double	Default 0.2. Sampling standard deviation for VY
🔵	wz_std	double	Default 0.4. Sampling standard deviation for Wz
🔵	vx_max	double	Default 0.5. Max VX (m/s)
🔵	vy_max	double	Default 0.5. Max VY in either direction, if holonomic. (m/s)
🔵	vx_min	double	Default -0.35. Min VX (m/s)
🔵	wz_max	double	Default 1.9. Max WZ (rad/s)
🟠	temperature	double	Default: 0.3. Selectiveness of trajectories by their costs (The closer this
	gamma	double	Default: 0.015. A trade-off between smoothness (high) and low energy (
	visualize	bool	Default: false. Publish visualization of trajectories, which can slow down
	retry_attempt_limit	int	Default 1. Number of attempts to find feasible trajectory on failure for s
🟢	regenerate_noises	bool	Default false. Whether to regenerate noises each iteration or use single

https://docs.nav2.org/configuration/packages/configuring-mppic.html

MPPI and ROS NAV2

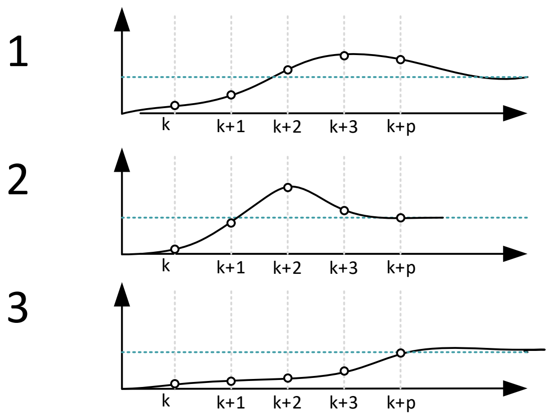
- ❑ Configuration and Tuning
  - e.g Nav2 MPPI Controller

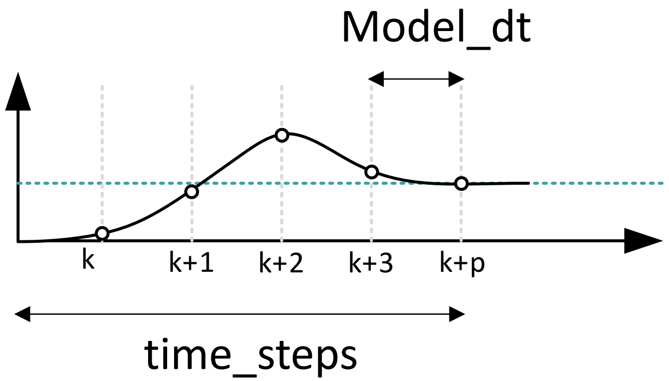
MPPI Controller



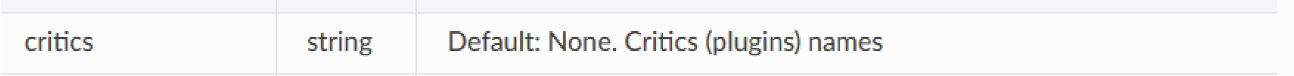
iteration_count	int	Default 1. Iteration count in MPPI algorithm. Recommend to keep as 1 a
batch_size	int	Default 1000. Count of randomly sampled candidate trajectories
time_steps	int	Default 56. Number of time steps (points) in each sampled trajectory
model_dt	double	Default: 0.05. Time interval (s) between two sampled points in trajectori

Batch_size





MPPI and ROS NAV2



critics	string	Default: None. Critics (plugins) names

- ❑ Configuration and Tuning
  - e.g Nav2 MPPI Controller

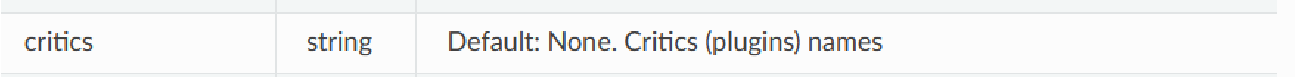
Cost Function

Constraint Critic	This critic penalizes trajectories that have components outside of the set dynamic or kinematic constraints
Goal Angle Critic	Angle Between Robot ang Goal
Goal Critic	Distance between robot and goal above which goal
Obstacles Critic	This critic incentivizes navigating away from obstacles and critical collisions using either a circular robot point-check or full SE2 footprint check using distances from obstacles. (inflation_layer_name, collision_cost, collision_margin_distance)
Cost Critic	This critic incentivizes navigating away from obstacles and critical collisions using either a circular robot point-check or full SE2 footprint check using the costmap values. (inflation_layer_name, collision_cost)
Path Align Critic	Aligns the robot to the path by looking at the integrated distance between the reference path and the trajectory and scoring negatively for the sum total discrepancy
Path Angle	This critic penalizes trajectories at a high relative angle to the path, looks at the angle wrt the robot and a point on the path some N points ahead and scores negatively if the angle is 'large'
Path Follow Critic	This critic incentivizes making progress along the path. This is what drives the robot forward along the path.
Prefer Forward Critic	This critic incentivizes moving in the forward direction, rather than reversing
Twirling Critic	This critic penalizes unnecessary ‘twisting’ with holonomic vehicles
Velocity Deadband Critic	This critic penalizes velocities that fall below the deadband threshold, helping to mitigate hardware limitations on certain platforms.

MPPI and ROS NAV2

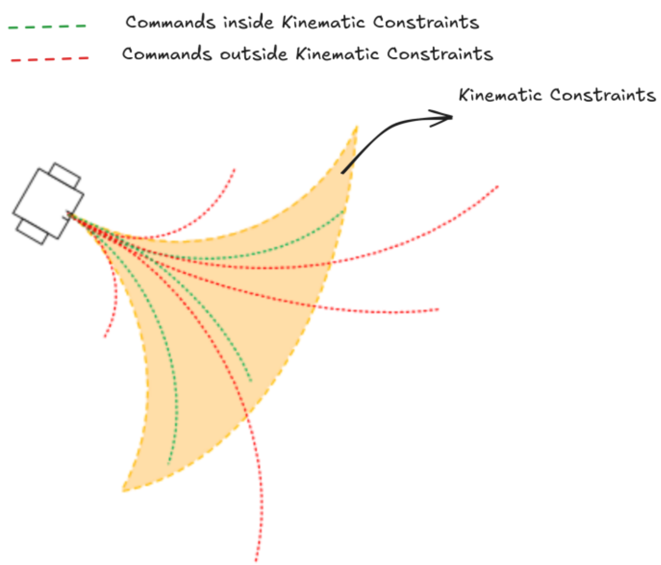
- ❑ Configuration and Tuning
  - e.g Nav2 MPPI Controller

Cost Function

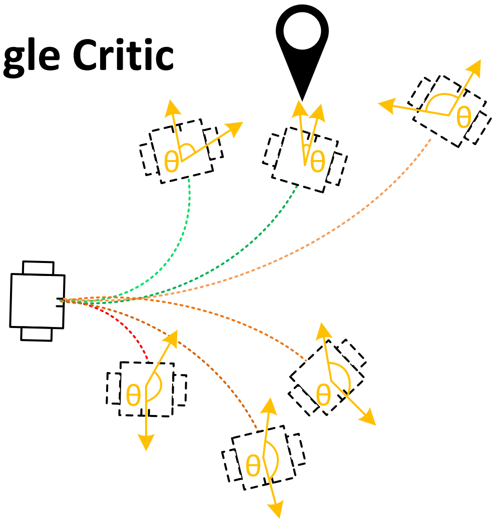


critics	string	Default: None. Critics (plugins) names

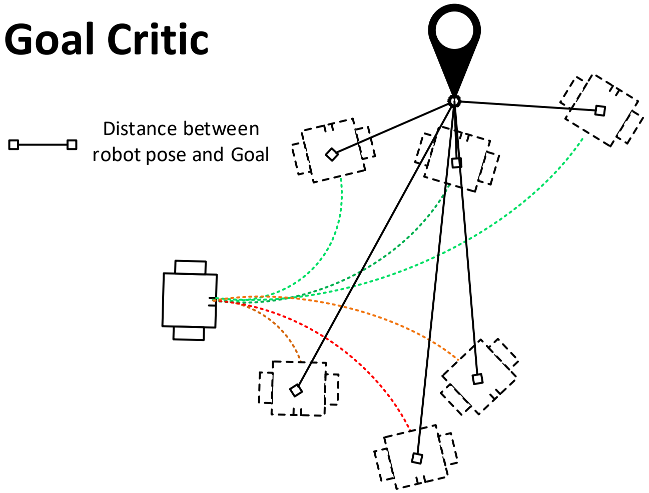
Constraint Critic


Commands inside Kinematic Constraints
Commands outside Kinematic Constraints
Kinematic Constraints

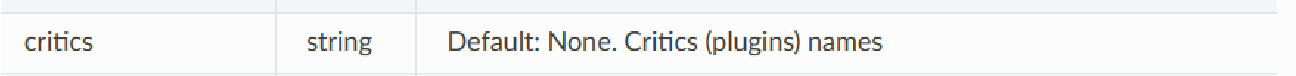
Goal Angle Critic



Goal Critic



MPPI and ROS NAV2

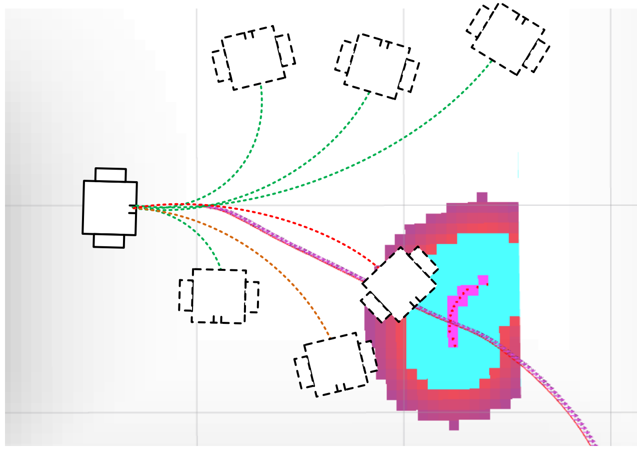


critics	string	Default: None. Critics (plugins) names

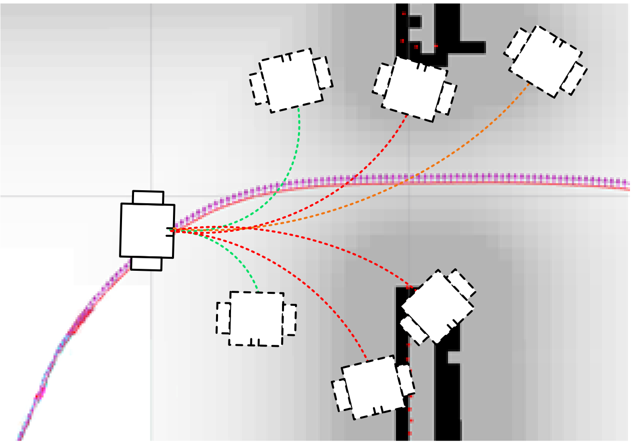
- ❑ Configuration and Tuning
  - e.g Nav2 MPPI Controller

Cost Function

Obstacle Critic **(e.g localcostmap)**



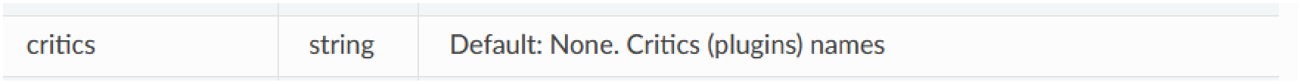
Cost Critic **(e.g globalcostmap)**



MPPI and ROS NAV2

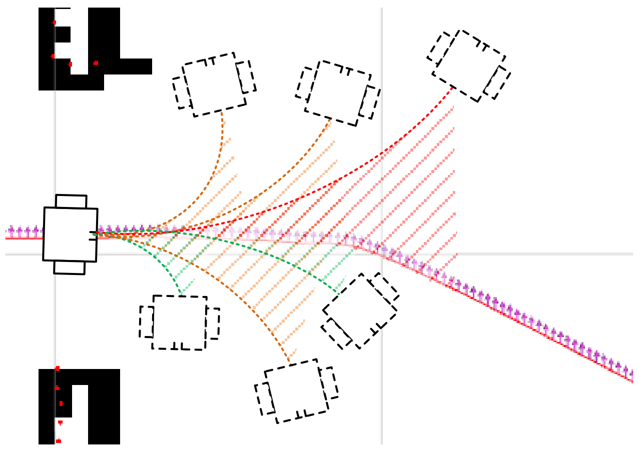
- ❑ Configuration and Tuning
  - e.g Nav2 MPPI Controller

Cost Function

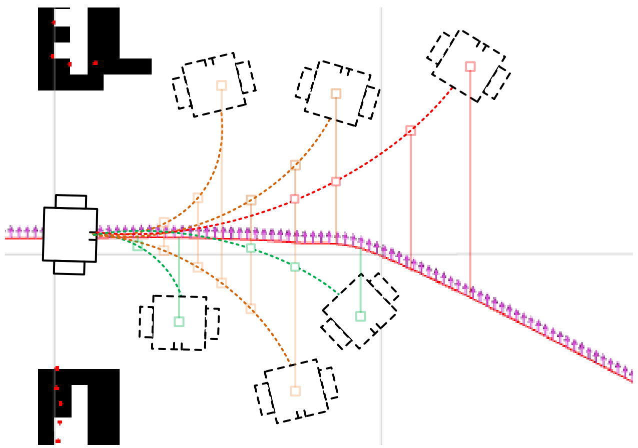


critics	string	Default: None. Critics (plugins) names

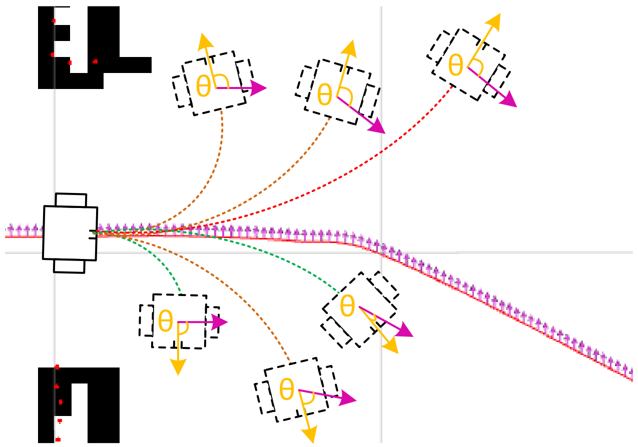
Path Align Critic



Path Follow Critic



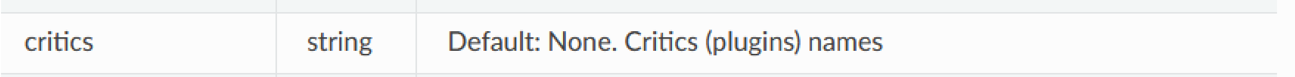
Path Angle Critic



MPPI and ROS NAV2

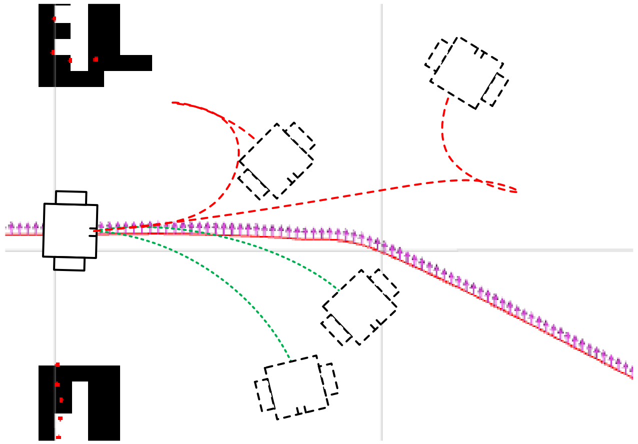
- ❑ Configuration and Tuning
  - e.g Nav2 MPPI Controller

Cost Function

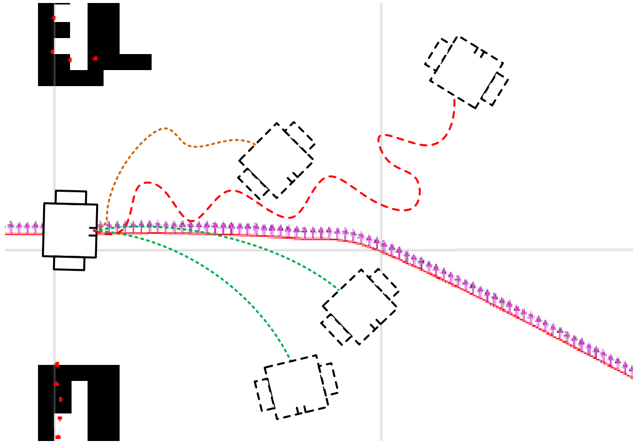


critics	string	Default: None. Critics (plugins) names

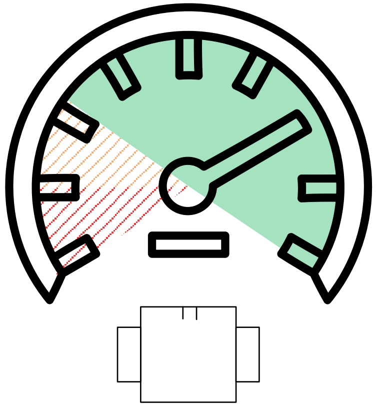
Prefer Forward Critic



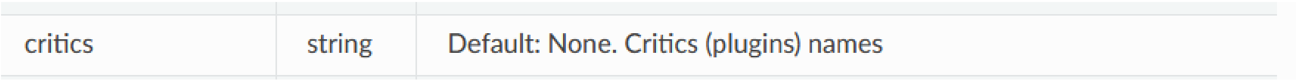
Twirling Critic



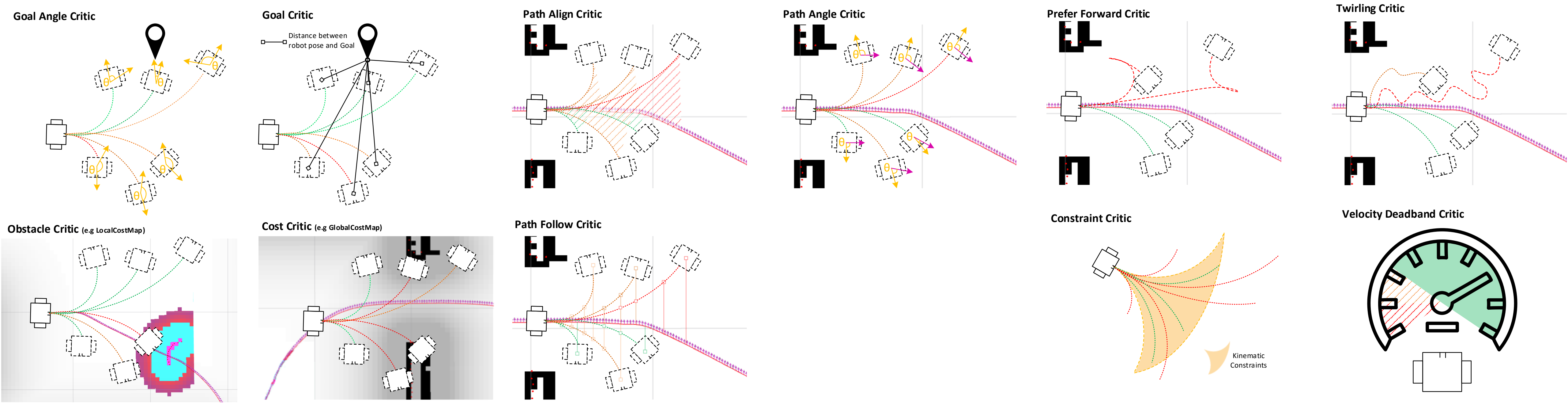
Velocity Deadband Critic



MPPI and ROS NAV2

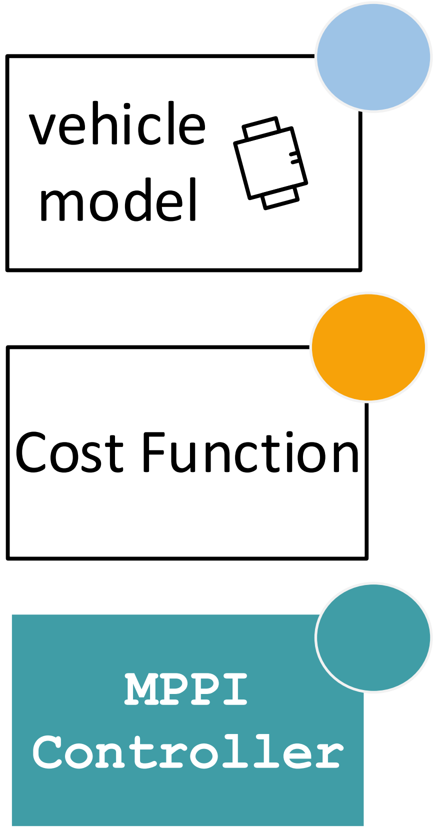


critics	string	Default: None. Critics (plugins) names



MPPI and ROS NAV2

- ❑ Nav2 Configuration

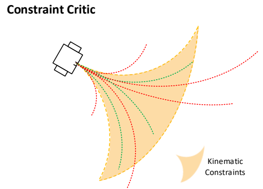


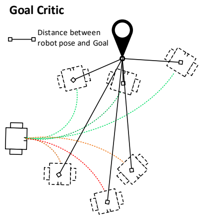
```yaml
controller_server:
 ros__parameters:
  controller_frequency: 30.0
  FollowPath:
   plugin: "nav2_mppi_controller::MPPIController"
   time_steps: 56
   model_dt: 0.05
   batch_size: 2000
   vx_std: 0.2
   vy_std: 0.2
   wz_std: 0.4
   vx_max: 1.0 #0.5
   vx_min: -0.35
   vy_max: 1.0 #0.5
   wz_max: 1.9
   ax_max: 0.5 #3.0
   ax_min: -0.5 #-3.5
   ay_max: 0.5 #3.0
   az_max: 0.5 #3.5
   motion_model: "DiffDrive"
   iteration_count: 1
   prune_distance: 1.7
   transform_tolerance: 0.1
   temperature: 0.3
   gamma: 0.015
   visualize: true #false
```

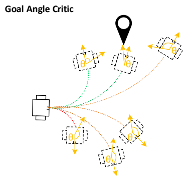
MPPI and ROS NAV2

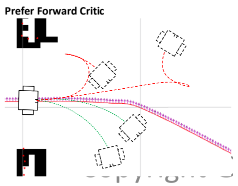
- ❑ Nav2 Configuration

Cost Function


Constraint Critic


Goal Critic


Goal Angle Critic


Prefer Forward Critic

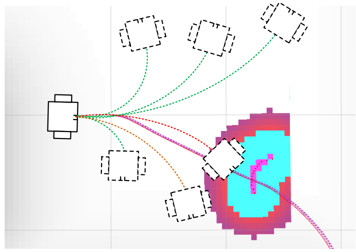
```
reset_period: 1.0 # (only in Humble)
   regenerate_noises: false
   TrajectoryVisualizer:
    trajectory_step: 5
    time_step: 3
   AckermannConstraints:
    min_turning_r: 0.2
   critics: ["ConstraintCritic", "CostCritic", "GoalCritic", "GoalAngleCritic",
"PathAlignCritic", "PathFollowCritic", "PathAngleCritic", "PreferForwardCritic"]
   ConstraintCritic:
    enabled: true
    cost_power: 1
    cost_weight: 4.0
   GoalCritic:
    enabled: true
    cost_power: 1
    cost_weight: 5.0
    threshold_to_consider: 1.4
   GoalAngleCritic:
    enabled: true
    cost_power: 1
    cost_weight: 3.0
    threshold_to_consider: 0.5
   PreferForwardCritic:
    enabled: true
    cost_power: 1
    cost_weight: 5.0
    threshold_to_consider: 0.5
```

MPPI and ROS NAV2

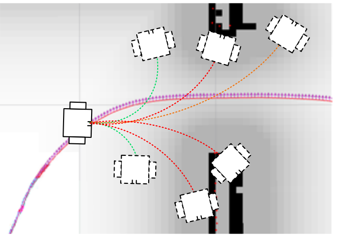
❑ Nav2 Configuration

Cost Function

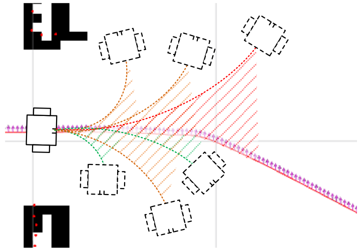
Obstacle Critic (e.g LocalCostMap)



Cost Critic (e.g GlobalCostMap)



Path Align Critic

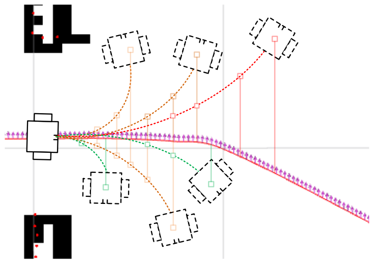


```
ObstaclesCritic:
   enabled: true
   cost_power: 1
   repulsion_weight: 1.5
   critical_weight: 20.0
   consider_footprint: false
   collision_cost: 10000.0
   collision_margin_distance: 0.1
   near_goal_distance: 0.5
   inflation_radius: 0.275 # (only in Humble)
   cost_scaling_factor: 10.0 # (only in Humble)
  CostCritic:
   enabled: true
   cost_power: 1
   cost_weight: 3.81
   critical_cost: 300.0
   consider_footprint: true
   collision_cost: 1000000.0
   near_goal_distance: 1.0
   trajectory_point_step: 2
  PathAlignCritic:
   enabled: true
   cost_power: 1
   cost_weight: 14.0
   max_path_occupancy_ratio: 0.05
   trajectory_point_step: 4
   threshold_to_consider: 0.5
   offset_from_furthest: 20
   use_path_orientations: false
```

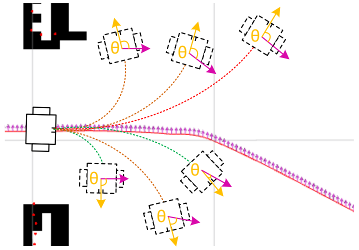
MPPI and ROS NAV2

- ❑ Nav2 Configuration

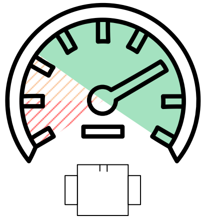
Path Follow Critic



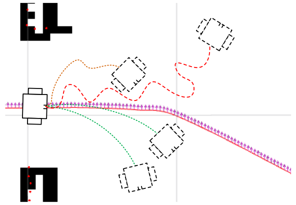
Path Angle Critic



Velocity Deadband Critic



Twirling Critic



```
PathFollowCritic:
    enabled: true
    cost_power: 1
    cost_weight: 5.0
    offset_from_furthest: 5
    threshold_to_consider: 1.4
  PathAngleCritic:
    enabled: true
    cost_power: 1
    cost_weight: 2.0
    offset_from_furthest: 4
    threshold_to_consider: 0.5
    max_angle_to_furthest: 1.0
    mode: 0
  # VelocityDeadbandCritic:
  #   enabled: true
  #   cost_power: 1
  #   cost_weight: 35.0
  #   deadband_velocities: [0.05, 0.05, 0.05]
  # TwirlingCritic:
  #   enabled: true
  #   twirling_cost_power: 1
  #   twirling_cost_weight: 10.0
```

References

- http://wiki.ros.org/dwa_local_planner
- Fox, D. and Burgard, W. and Thrun, S., DWA approach avoidance collision dynamic window, Robotics Automation Magazine, IEEE,1997
- Roscon, 2017, https://roscon.ros.org/2017/presentations/ROSCon%202017%20Fundamentals%20of%20Local%20Planning.pdf
- http://wiki.ros.org/teb_local_planner
- Quinlan S, Khatib O Elastic bands: connecting path planning and control. In: Proceedings of the IEEE international conference on robotics and automation, 1993
- Adrian_Boeing Blog : http://adrianboeing.blogspot.com/2012/03/elastic-band-realtime-pathfinding.html 2012
- Rösmann, C.; Feiten, W.; Wösch, T.; Hoffmann, F.; Bertram, T. Trajectory modification considering dynamic constraints of autonomous robots. In Proceedings of the ROBOTIK, 2012
- Filotheou, Alexandros et al. “Quantitative and Qualitative Evaluation of ROS-Enabled Local and Global Planners in 2D Static Environments.” Journal of Intelligent & Robotic Systems, 2020

✉ Jacques Saraydaryan

Jacques.saraydaryan@cpe.fr