FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE

ELECTIVE IN ROBOTICS

Quadrotor

Motion Planning Algorithms

Prof. Marilena Vendittelli
Prof. Jean-Paul Laumond

Jacopo Capolicchio
Riccardo Spica

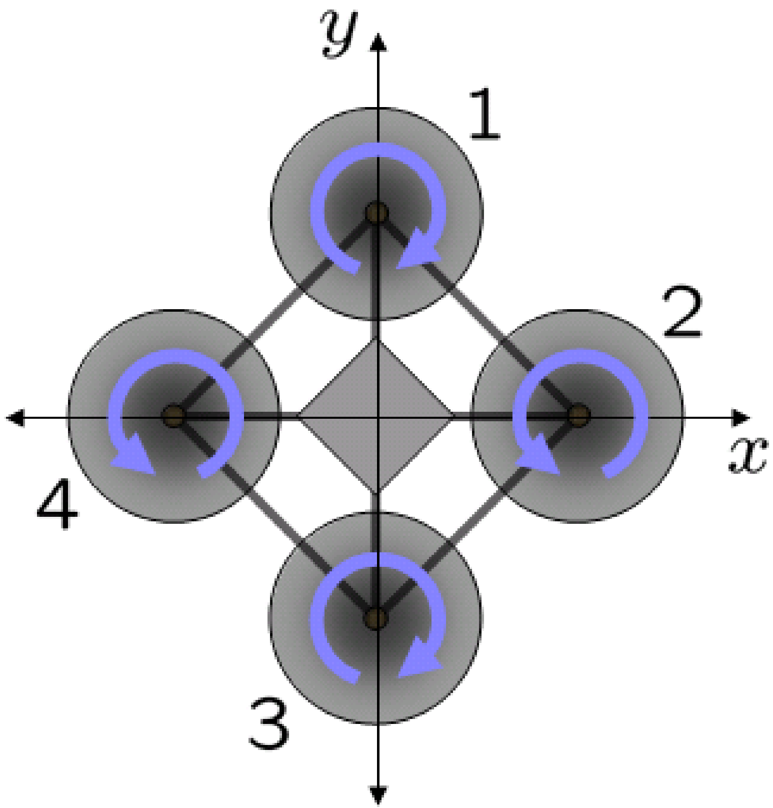
Academic Year 2010-2011

SAPIENZA
UNIVERSITÀ DI ROMA

Introduction

Small-scale quadrotors

- Four pitch-fixed rotors disposed at the vertices of a square, whose directions of rotation are equal in pairs
- Vertical and lateral displacements, as well as yaw rotations, by varying the relative turning speeds of rotors



Introduction

Pros and Cons

Pros	Cons
Simple Mechanics	Low Payloads
Low cost	Poor sensors equipment
Robustness	Low computational capabilities
High Maneuverability	
VTOL	
Hovering	

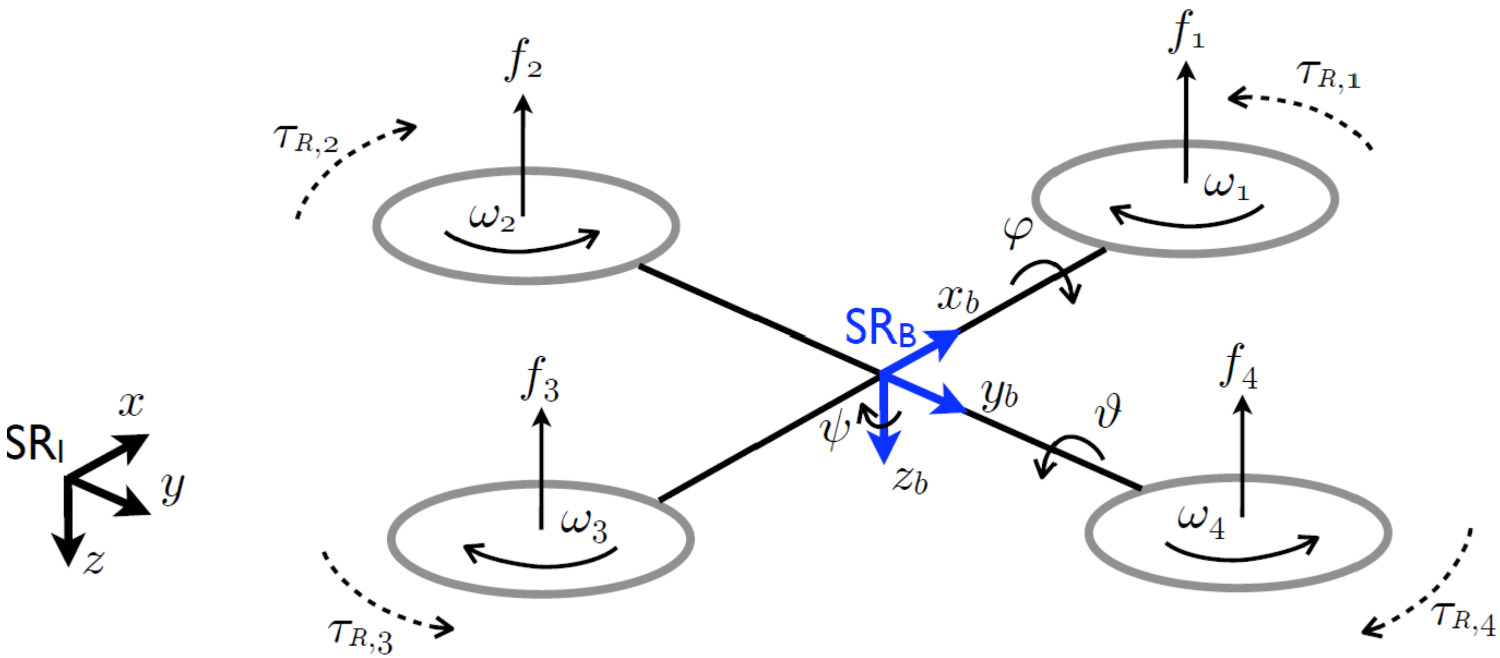
Dynamic Model

Preliminaries

- $\boldsymbol{SR_I}$: world inertial frame
- $\boldsymbol{SR_B}$: body-attached frame
- Configuration = Position + Orientation of $\boldsymbol{SR_B}$ w.r.t. $\boldsymbol{SR_I}$
- Assumptions:
  - Robot symmetry
  - No disturbances
  - Negligible gyroscopic and aerodynamic effects
  - Negligible motor dynamics
  - Low level controllers for blades rotational speed

Dynamic Model

Preliminaries (2)



$$f_i = b\omega_i^2$$
$$\tau_{R,i} = d\omega_i^2$$

b: *thrust factor*

d: *drag factor*.

Dynamic Model

Newton-Euler approach (1)

The orientation is described by using RPY angles $(\varphi, \vartheta, \psi)$:

$$
{}^{\mathrm{I}}\mathrm{R}_{\mathrm{B}} = \begin{pmatrix} c_{\psi}\ c_{\vartheta} & c_{\psi}\ s_{\vartheta}s_{\varphi} - s_{\psi}c_{\varphi} & c_{\psi}s_{\vartheta}c_{\varphi} + s_{\psi}s_{\varphi} \\ s_{\psi}c_{\vartheta} & s_{\psi}s_{\vartheta}s_{\varphi} + c_{\psi}c_{\varphi} & s_{\psi}s_{\vartheta}c_{\varphi} - c_{\psi}s_{\varphi} \\ -s_{\vartheta} & c_{\vartheta}s_{\varphi} & c_{\vartheta}c_{\varphi} \end{pmatrix}
$$

Control inputs transformation:

$$
\begin{cases} T = f_1 + f_2 + f_3 + f_4 \\ \tau_{\varphi} = l\,(f_2 - f_4) \\ \tau_{\vartheta} = l\,(f_1 - f_3) \\ \tau_{\psi} = -\tau_{R,1} + \tau_{R,2} - \tau_{R,3} + \tau_{R,4} \end{cases}
$$

Dynamic Model

Newton-Euler approach (2)

The translational dynamics is governed by the Newton equation:

$$m\begin{pmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{pmatrix} = \begin{pmatrix}0\\ 0\\ mg\end{pmatrix} + {}^{I}R_{B}\begin{pmatrix}0\\ 0\\ -T\end{pmatrix}$$

The angular acceleration is governed by the Euler equation:

$$I\begin{pmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{pmatrix} + \begin{pmatrix}p\\ q\\ r\end{pmatrix} \times I\begin{pmatrix}p\\ q\\ r\end{pmatrix} = \begin{pmatrix}\tau_{\varphi}\\ \tau_{\vartheta}\\ \tau_{\psi}\end{pmatrix}$$

For small roll and pitch angles:

$$\begin{pmatrix}p\\ q\\ r\end{pmatrix} = \begin{pmatrix}1 & 0 & -\sin\vartheta\\ 0 & \cos\varphi & \cos\vartheta\sin\varphi\\ 0 & -\sin\varphi & \cos\vartheta\cos\varphi\end{pmatrix}\begin{pmatrix}\dot{\varphi}\\ \dot{\vartheta}\\ \dot{\psi}\end{pmatrix} \sim \begin{pmatrix}\dot{\varphi}\\ \dot{\vartheta}\\ \dot{\psi}\end{pmatrix}$$

Dynamic Model

Newton-Euler approach (3)

System state:

$$\xi = [x \quad y \quad z \quad \varphi \quad \vartheta \quad \psi \quad \dot{x} \quad \dot{y} \quad \dot{z} \quad \dot{\varphi} \quad \dot{\vartheta} \quad \dot{\psi}]^T$$

Then:

$$\begin{cases} \ddot{x} = -[\cos(\psi)\sin(\vartheta)\cos(\varphi) + \sin(\psi)\sin(\varphi)]\dfrac{T}{m} \\ \ddot{y} = -[\sin(\psi)\sin(\vartheta)\cos(\varphi) - \cos(\psi)\sin(\varphi)]\dfrac{T}{m} \\ \ddot{z} = g - \cos(\vartheta)\cos(\varphi)\dfrac{T}{m} \\ \ddot{\varphi} = \dfrac{\tau_\varphi}{I_x} \\ \ddot{\vartheta} = \dfrac{\tau_\vartheta}{I_y} \\ \ddot{\psi} = \dfrac{\tau_\psi}{I_z} \end{cases}$$

Dynamic Model

Differential flatness (1)

Flat outputs: $(x, y, z,\ \psi)$

From the dynamic model one obtains:

$$\vartheta = \operatorname{atan}\left(\frac{\ddot{x}\cos(\psi) + \ddot{y}\sin(\psi)}{\ddot{z} - g}\right)$$

$$\varphi = \operatorname{atan}\left(\frac{\ddot{x}\sin(\psi) - \ddot{y}\cos(\psi)}{\ddot{z} - g}\cos(\vartheta)\right)$$

and by derivation

$$\dot{\vartheta} = \frac{1}{1 + \tan^2\vartheta}\,\frac{1}{\ddot{z} - g}\left[\left(x^{(3)} + \ddot{y}\dot{\psi}\right)\cos\psi + \left(y^{(3)} - \ddot{x}\dot{\psi}\right)\sin\psi - z^{(3)}\tan\vartheta\right]$$

$$\dot{\varphi} = \frac{1}{1 + \tan^2\varphi}\,\frac{1}{\ddot{z} - g}\left(\sin\psi\left(\ddot{y}\dot{\psi} + x^{(3)}\right)\cos\vartheta + \left(\ddot{y}\dot{\vartheta}\sin\vartheta\right)\right)$$
$$+ \cos\psi\left(\left(x\dot{\psi} \ddot{+} y^{(3)}\right)\cos\vartheta + \left(\ddot{y}\dot{\vartheta}\sin\vartheta\right) - \tan\varphi\right)$$

Dynamic Model

Differential flatness (2)

$$
\begin{aligned}
\ddot{\vartheta} = & -2\dot{\vartheta}^2 \tan\vartheta - \frac{z^{(3)}\dot{\vartheta}}{\ddot{z} - g} \\
& + \frac{1}{1+\tan^2\vartheta}\,\frac{1}{\ddot{z}-g}\Big[\big(x^{(4)} + 2y^{(3)}\dot{\psi} + \ddot{y}\ddot{\psi} - \ddot{x}\dot{\psi}^2\big)\cos\psi \\
& + \big(y^{(4)} - 2x^{(3)}\dot{\psi} - \ddot{x}\ddot{\psi} - \ddot{y}\dot{\psi}^2\big)\sin\psi - z^{(4)}\tan\vartheta \\
& - z^{(3)}\dot{\vartheta}(1+\tan^2\vartheta)\Big]
\end{aligned}
$$

$$
\begin{aligned}
\ddot{\varphi} = & -2\dot{\varphi}^2 \tan\varphi - \frac{z^{(3)}\dot{\varphi}}{\ddot{z} - g} \\
& + \frac{1}{1+\tan^2\varphi}\,\frac{1}{\ddot{z}-g}\Big[\cos\psi\,\big(\sin\vartheta\,\dot{\vartheta}\big(y^{(3)} - \ddot{x}\big) + \cos\vartheta\dot{\vartheta}^2\ddot{y} + \sin\vartheta\,\ddot{\vartheta}\ddot{y} \\
& + \cos\vartheta\,\big(-y^{(4)} + x^{(3)}\dot{\psi} + \ddot{x}\ddot{\psi} + x^{(3)}\dot{\psi} + \ddot{y}\dot{\psi}^2\big) + \sin\vartheta\,\dot{\vartheta}\big(y^{(3)} - \ddot{x}\dot{\psi}\big) \\
& + \sin\psi\,\big(\sin\vartheta\,\dot{\vartheta}\big(x^{(3)} - \ddot{y}\big) + \cos\vartheta\dot{\vartheta}^2\ddot{x} + \sin\vartheta\,\ddot{\vartheta}\ddot{x} \\
& + \cos\vartheta\,\big(x^{(4)} + y^{(3)}\dot{\psi} + \ddot{y}\ddot{\psi} + y^{(3)}\dot{\psi} - \ddot{x}\dot{\psi}^2\big) + \sin\vartheta\,\dot{\vartheta}\big(x^{(3)} - \ddot{y}\dot{\psi}\big) \\
& - z^{(4)}\tan\varphi - z^{(3)}\dot{\vartheta}(1+\tan^2\varphi)\dot{\varphi}\Big]
\end{aligned}
$$

Dynamic Model

Differential flatness (3)

Finally the torque inputs are:

$$\begin{cases} \tau_\varphi = \ddot{\varphi} I_x \\ \tau_\vartheta = \ddot{\vartheta} I_y \\ \tau_\psi = \ddot{\psi} I_z \end{cases}$$

while the thrust is:

$$T = m\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} - g)^2}$$

For $\mathcal{T} \to \infty$ all the derivatives of the flat outputs go to zero, then

$$\begin{bmatrix} \varphi \\ \vartheta \end{bmatrix} \to \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} T \\ \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \to \begin{bmatrix} mg \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dynamic Model

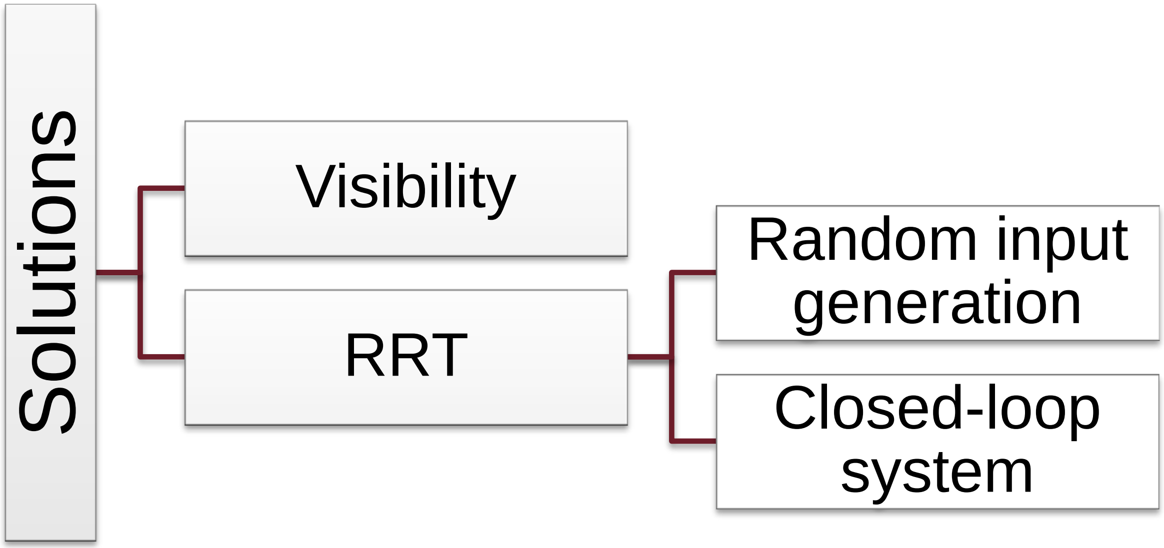
Differential flatness (4)

- The robot is able to track any given trajectory of the 3D position and yaw angle provided that it is “smooth enough”
- In particular to guarantee inputs continuity:
  - the position trajectory has to be continue up to the fourth order derivative
  - the yaw angle trajectory has to be continue up to the second order derivative
- To satisfy motors constraints $\mathcal{T}$ has to be “large enough”

Motion Planning

Introduction

Aim : *build a feasible trajectory going from a start to a goal, possibly considering yaw angle variations , in presence of obstacles and actuators constraints*



Visibility-Based Motion Planning

Introduction

The motion planning has been split in four steps:

1. Creation of a visibility-based probabilistic roadmap in the space of flat outputs
2. Searching of a piece-wise linear safe trajectory exploiting the roadmap
3. Approximation of the computed trajectory through a smooth spline
4. Time-scaling

Visibility-Based Motion Planning

Visibility roadmap construction (1)

Aim : *build a roadmap with few nodes but still sufficient for solving the global planning problem*

- For a given local method $\mathcal{L}$, the visibility domain of q is the set of configurations reachable by q through $\mathcal{L}$

$$\mathcal{V}_{\mathcal{L}}(q) = \{q' \in CS_{free}\ s.t.\ \mathcal{L}(q,q') \subset CS_{free}\}$$

- q is the *guard* of $\mathcal{V}_{\mathcal{L}}(\mathrm{q})$
- A set of guards constitutes a coverage of $\mathrm{CS}_{\mathrm{free}}$ if the union of their visibility domains covers $\mathrm{CS}_{\mathrm{free}}$
- Note: such a finite coverage may not always exist, depending on both the shape of $\mathrm{CS}_{\mathrm{free}}$ and the local method $\mathcal{L}$

Visibility-Based Motion Planning

Visibility roadmap construction (2)

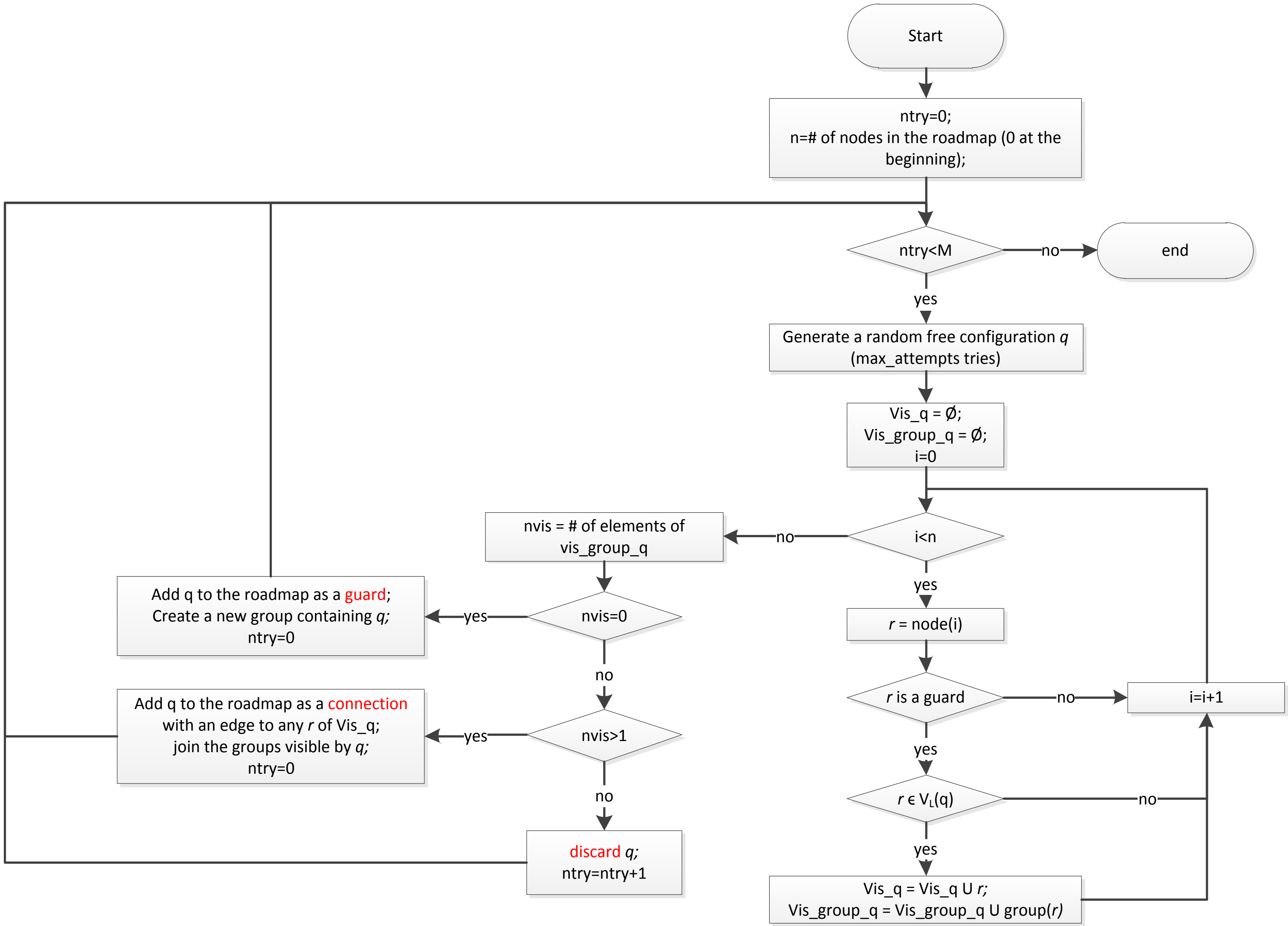
A free configuration q is added to the roadmap iff one of the following conditions is satisfied:

q is a guard

- q does not belong to the visibility domain of any other guard of the current roadmap → it enlarges the coverage of CS_{free}

q is a connection

- q lies in the intersection of the visibility domains of at least two guards belonging to different connected components of the current roadmap → it enhances the connectivity


Start
ntry=0;
n=# of nodes in the roadmap (0 at the beginning);
ntry<M
no
end
yes
Generate a random free configuration q (max_attempts tries)
Vis_q = Ø;
Vis_group_q = Ø;
i=0
i<n
no
nvis = # of elements of vis_group_q
yes
r = node(i)
nvis=0
yes
Add q to the roadmap as a guard;
Create a new group containing q;
ntry=0
no
r is a guard
no
i=i+1
nvis>1
yes
Add q to the roadmap as a connection with an edge to any r of Vis_q;
join the groups visible by q;
ntry=0
yes
r ϵ V(q)
no
no
discard q;
ntry=ntry+1
yes
Vis_q = Vis_q U r;
Vis_group_q = Vis_group_q U group(r)

Visibility-Based Motion Planning

Visibility roadmap construction (4)

- Operating space: flat outputs

$$\sigma = \begin{pmatrix} x \\ y \\ z \\ \psi \end{pmatrix}$$

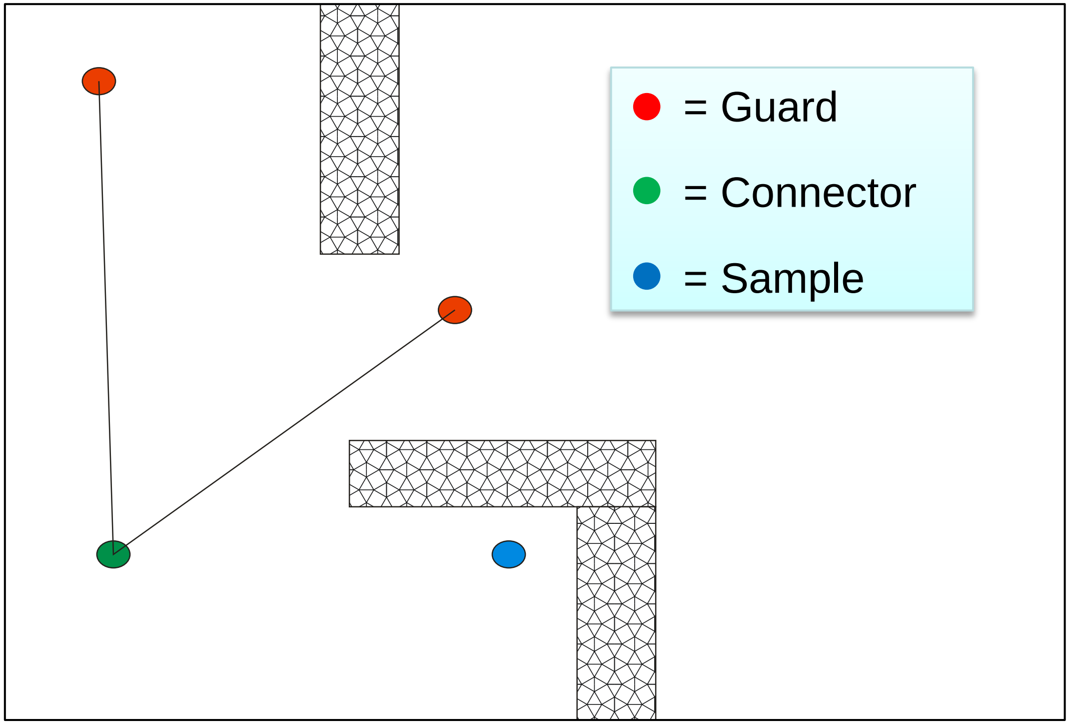
- Collision checking: obstacle expansion

$$arm\ length\ l = 0{,}25\ m$$

- Local method: straight lines

Visibility-Based Motion Planning

Visibility roadmap construction (5)



Why not to exploit connections to increase the coverage?

Visibility-Based Motion Planning

Visibility roadmap construction (6)

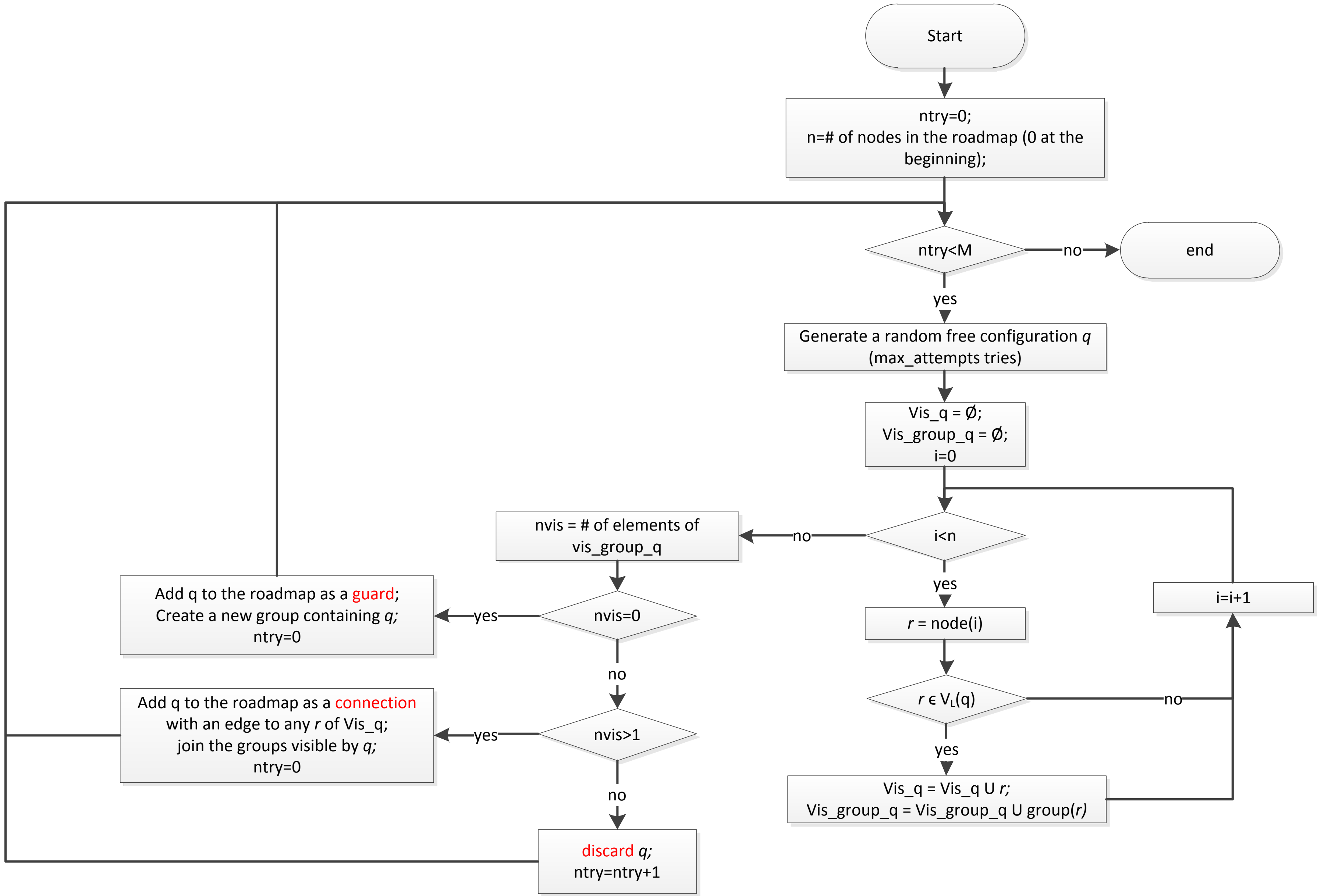
In the alternative version of algorithm a free configuration q is added to the roadmap iff one of the following conditions is satisfied:

q is a guard

- q does not belong to the visibility domain of any other node (either a guard or a connection) of the current roadmap

q is a connection

- q lies in the intersection of the visibility domains of at least two nodes (either guards or connections) belonging to two different connected components of the current roadmap

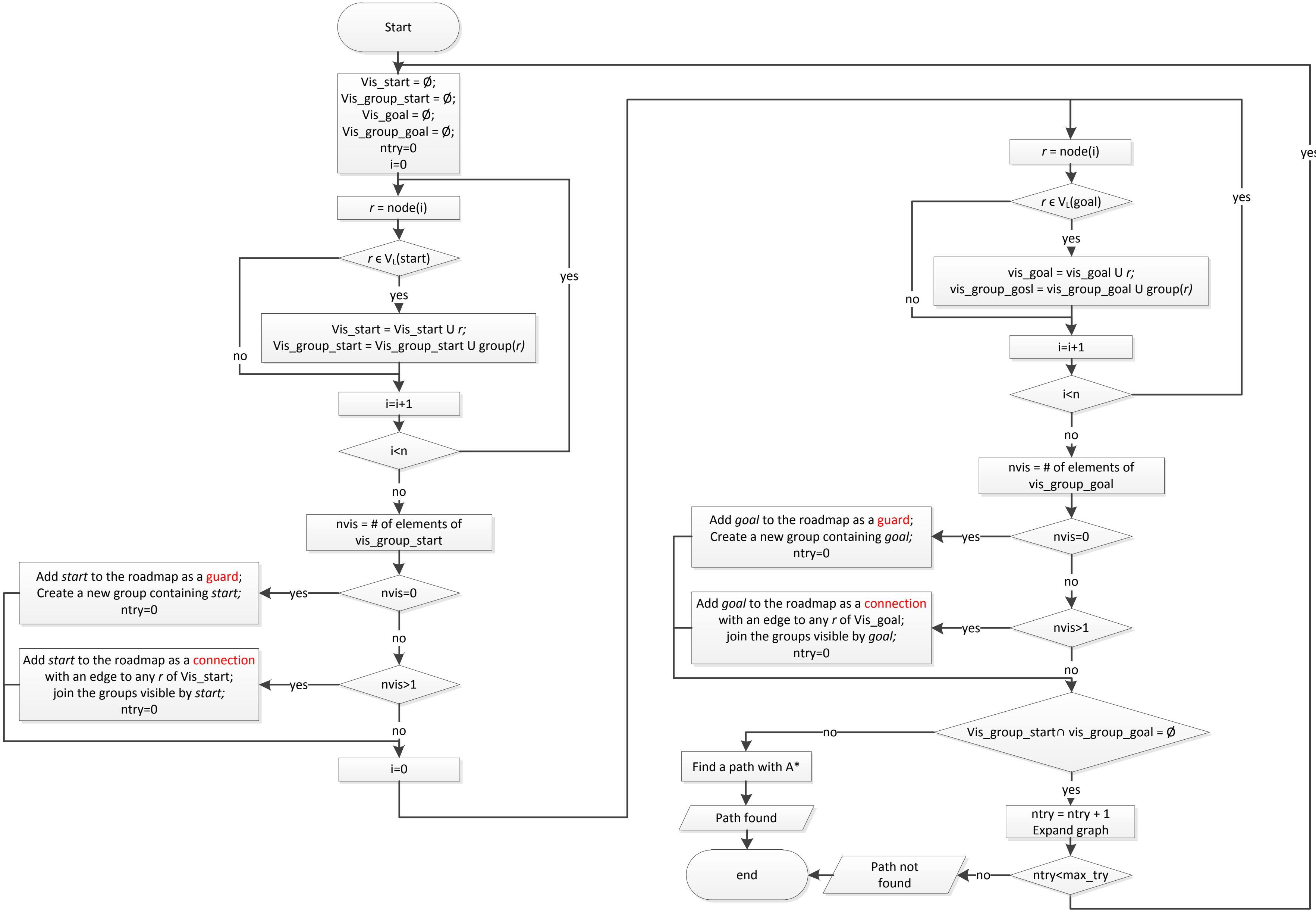

Start
ntry=0;
n=# of nodes in the roadmap (0 at the beginning);
ntry<M
no
end
yes
Generate a random free configuration q (max_attempts tries)
Vis_q = Ø;
Vis_group_q = Ø;
i=0
i<n
no
nvis = # of elements of vis_group_q
yes
r = node(i)
i=i+1
nvis=0
yes
Add q to the roadmap as a guard;
Create a new group containing q;
ntry=0
no
r ϵ VL(q)
no
yes
Vis_q = Vis_q U r;
Vis_group_q = Vis_group_q U group(r)
nvis>1
yes
Add q to the roadmap as a connection with an edge to any r of Vis_q;
join the groups visible by q;
ntry=0
no
discard q;
ntry=ntry+1

Visibility-Based Motion Planning

Searching a safe trajectory (1)

Aim : *compute (if possible) a safe piece-wise linear path that goes from the starting position to the goal*

- Such a trajectory requires the quadrotor to stop at each viapoint
- The searching of a safe solution is made iteratively:
  - try to connect (using straight lines) start and goal to two nodes q_s and q_g belonging to the same connected component of the roadmap
  - If success, a solution exists → A*
  - If not → expand the roadmap and try again


Start
Vis_start = Ø;
Vis_group_start = Ø;
Vis_goal = Ø;
Vis_group_goal = Ø;
ntry=0
i=0
r = node(i)
r ϵ V_L(start)
yes
no
Vis_start = Vis_start U r;
Vis_group_start = Vis_group_start U group(r)
i=i+1
i<n
yes
no
nvis = # of elements of vis_group_start
nvis=0
yes
Add start to the roadmap as a guard;
Create a new group containing start;
ntry=0
no
nvis>1
yes
Add start to the roadmap as a connection with an edge to any r of Vis_start;
join the groups visible by start;
ntry=0
no
i=0
r = node(i)
r ϵ V_L(goal)
yes
no
vis_goal = vis_goal U r;
vis_group_gosl = vis_group_goal U group(r)
i=i+1
i<n
yes
no
nvis = # of elements of vis_group_goal
nvis=0
yes
Add goal to the roadmap as a guard;
Create a new group containing goal;
ntry=0
no
nvis>1
yes
Add goal to the roadmap as a connection with an edge to any r of Vis_goal;
join the groups visible by goal;
ntry=0
no
Vis_group_start∩ vis_group_goal = Ø
no
Find a path with A*
Path found
end
yes
ntry = ntry + 1
Expand graph
ntry<max_try
no
Path not found
yes

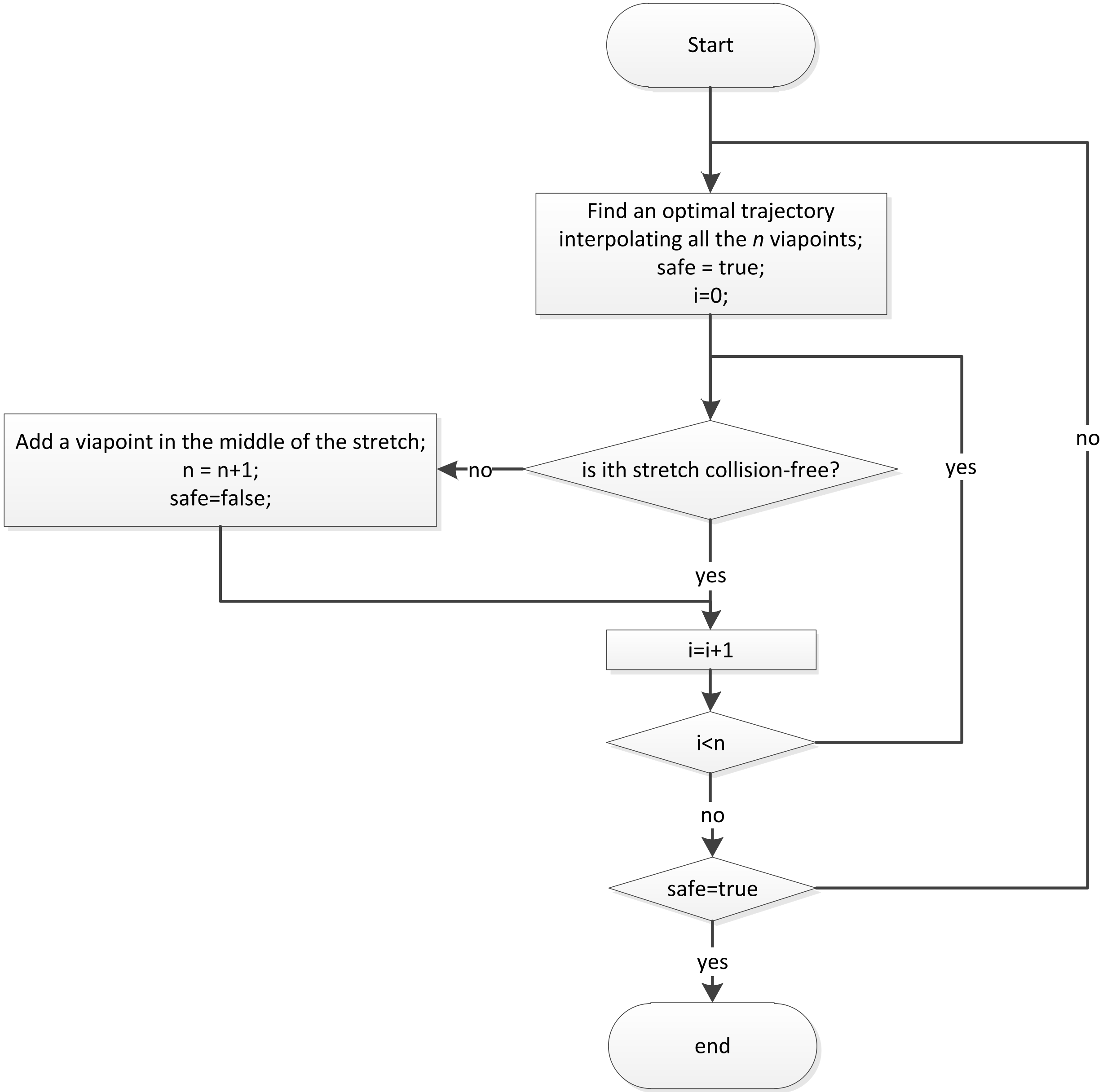
Visibility-Based Motion Planning

Spline interpolation (1)

Aim : *compute a smooth solution approximating the piece-wise linear path obtained in the previous step*

- Path computed by using a B-spline interpolating viapoints
- This kind of trajectory may significantly diverge from the safe piece-wise linear solution and consequently become unsafe

Introduction of a viapoint in the middle of any unsafe stretch


Start
Find an optimal trajectory interpolating all the *n* viapoints;
safe = true;
i=0;
is ith stretch collision-free?
no
Add a viapoint in the middle of the stretch;
n = n+1;
safe=false;
yes
i=i+1
i<n
yes
no
safe=true
no
yes
end

Visibility-Based Motion Planning

Spline interpolation (3)

Split the geometrical aspect from the temporal one

$$\left.\begin{array}{l} s_x(\tau) \in \mathcal{C}^4 : [0,1] \to \mathbb{R} \\ s_y(\tau) \in \mathcal{C}^4 : [0,1] \to \mathbb{R} \\ s_z(\tau) \in \mathcal{C}^4 : [0,1] \to \mathbb{R} \end{array}\right\} \longrightarrow 6^{th}\ order$$

$$\left. s_\psi(\tau) \in \mathcal{C}^2 : [0,1] \to \mathbb{R} \right\} \longrightarrow 4^{th}\ order$$

such that

$$\begin{cases} s_x(\tau_i) = x_i, & i = 1, \dots, n \\ s_y(\tau_i) = y_i, & i = 1, \dots, n \\ s_z(\tau_i) = z_i, & i = 1, \dots, n \\ s_\psi(\tau_i) = \psi_i, & i = 1, \dots, n \end{cases} \qquad \text{with } 0 = \tau_1 < \tau_2 < \cdots < \tau_n = 1$$

Visibility-Based Motion Planning

Spline interpolation (4)

$(\tau_1, \tau_2, \dots, \tau_n)$ are chosen by solving an optimization problem:

- Variables:

$$\delta_i = \tau_{i+1} - \tau_i \quad \text{for } i = 1, \dots, n-1$$

- Constraints:

$$0 < \delta_i < 1 \quad \text{for } i = 1, \dots, n-1$$

$$\sum_{i=1}^{n-1} \delta_i = 1$$

- Objective function:

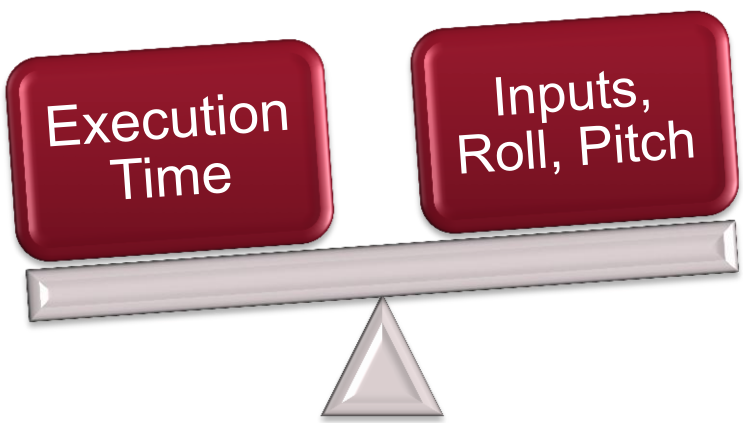
$$\int_0^1 \left[a \left(\left\| \frac{d^4 s_x(\tau)}{d\tau^4} \right\|^2 + \left\| \frac{d^4 s_y(\tau)}{d\tau^4} \right\|^2 + \left\| \frac{d^4 s_z(\tau)}{d\tau^4} \right\|^2 \right) + b \left\| \frac{d^2 s_\psi(\tau)}{d\tau^2} \right\|^2 \right] d\tau$$

Visibility-Based Motion Planning

Time scaling (1)

Aim : *optimize the time needed to complete the trajectory*

- Changing the time to navigate through the nodes by a factor of k (s.t. $t_i = k\tau_i$) the result is simply a time-scaled version of the non-dimensional solution



Visibility-Based Motion Planning

Time scaling (2)

k is chosen by solving an optimization problem:

- Variables:

$$k$$

- Constraints:

$$\begin{bmatrix} |\mathrm{T}| \\ |\tau_\varphi| \\ |\tau_\theta| \\ |\tau_\psi| \end{bmatrix} \leq \begin{bmatrix} \mathrm{T_M} \\ \tau_{\varphi,M} \\ \tau_{\theta,\mathrm{M}} \\ \tau_{\psi,M} \end{bmatrix}$$

- Objective function:

$$k = \mathcal{T}$$

Visibility-Based Motion Planning

Alternative choice for ψ

ψ is actuated independently $\Rightarrow$ we can neglect it during the roadmap construction

Putting:

$$\psi = \text{atan2}(\dot{y}, \dot{x})$$

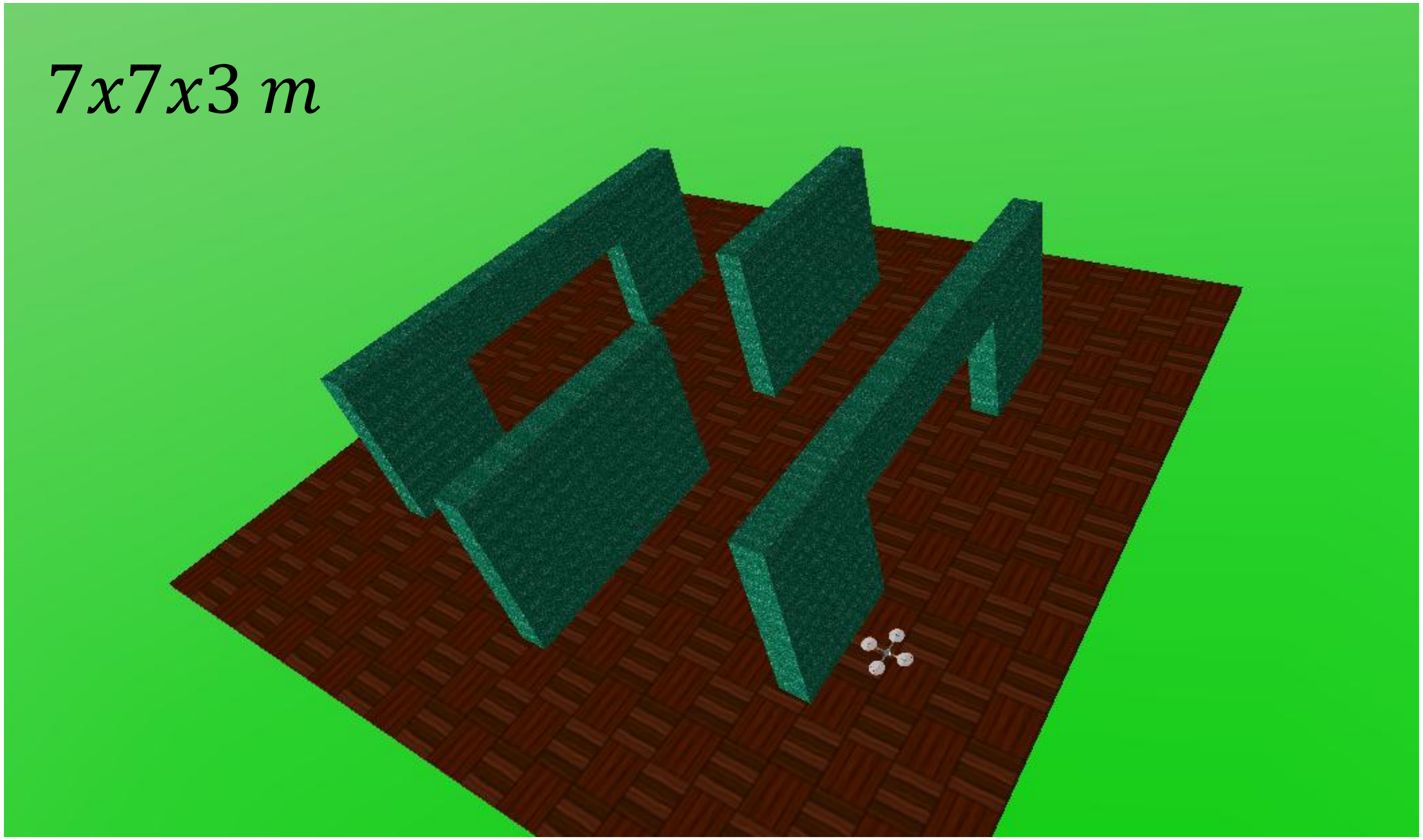
The robot points in the direction of motion and

$$\dot{\psi} = \frac{1}{1 + \left(\dot{y}/\dot{x}\right)^2} \left(\frac{\ddot{y}}{\dot{x}} - \frac{\dot{y}\ddot{x}}{\dot{x}^2} \right)$$

$$\ddot{\psi} = \frac{1}{1 + \left(\dot{y}/\dot{x}\right)^2} \left\{ \frac{1}{\dot{x}} \left[y^{(3)} + \frac{1}{\dot{x}} \left(2\frac{\dot{y}\ddot{x}^2}{\dot{x}} - 2\ddot{y}\ddot{x} - \dot{y}x^{(3)} \right) \right] \right\} - 2\frac{\dot{y}\dot{\psi}^2}{\dot{x}}$$

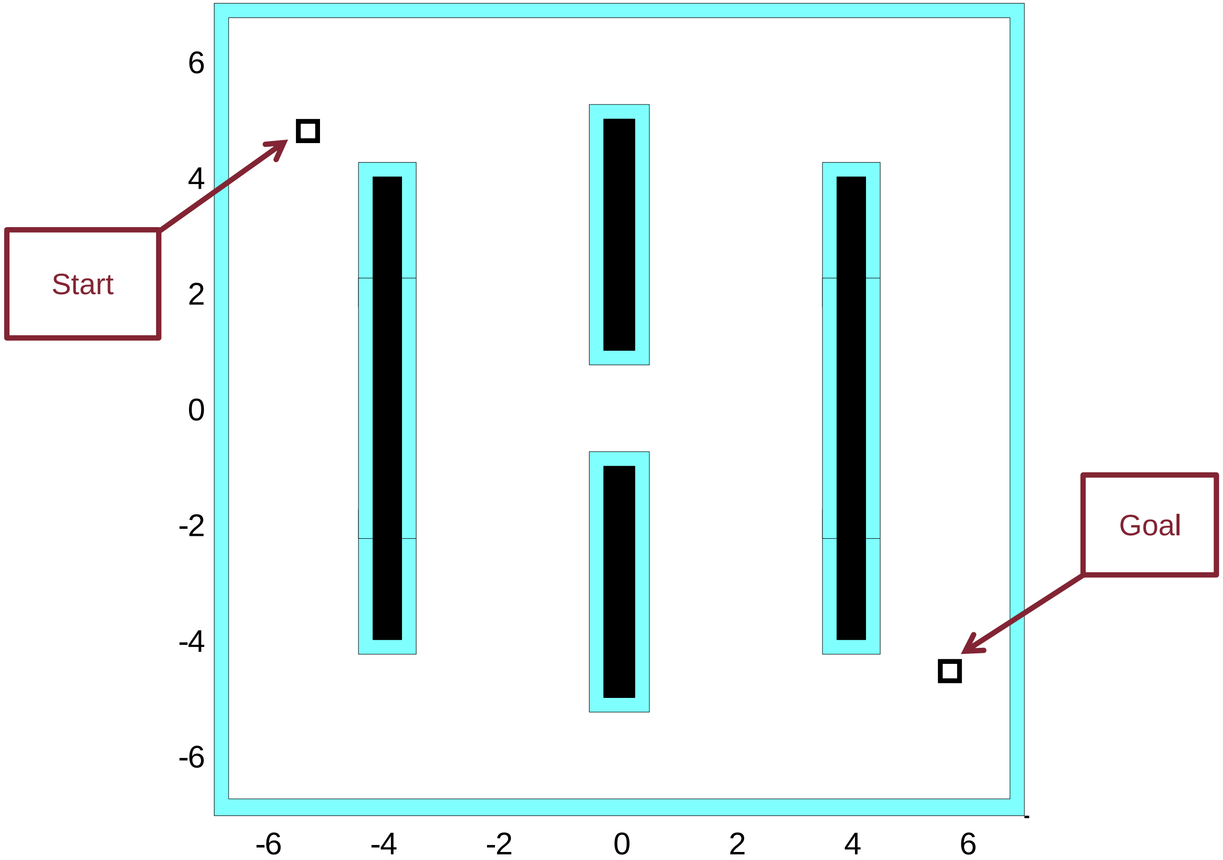
Visibility-Based Motion Planning

Simulation scenario



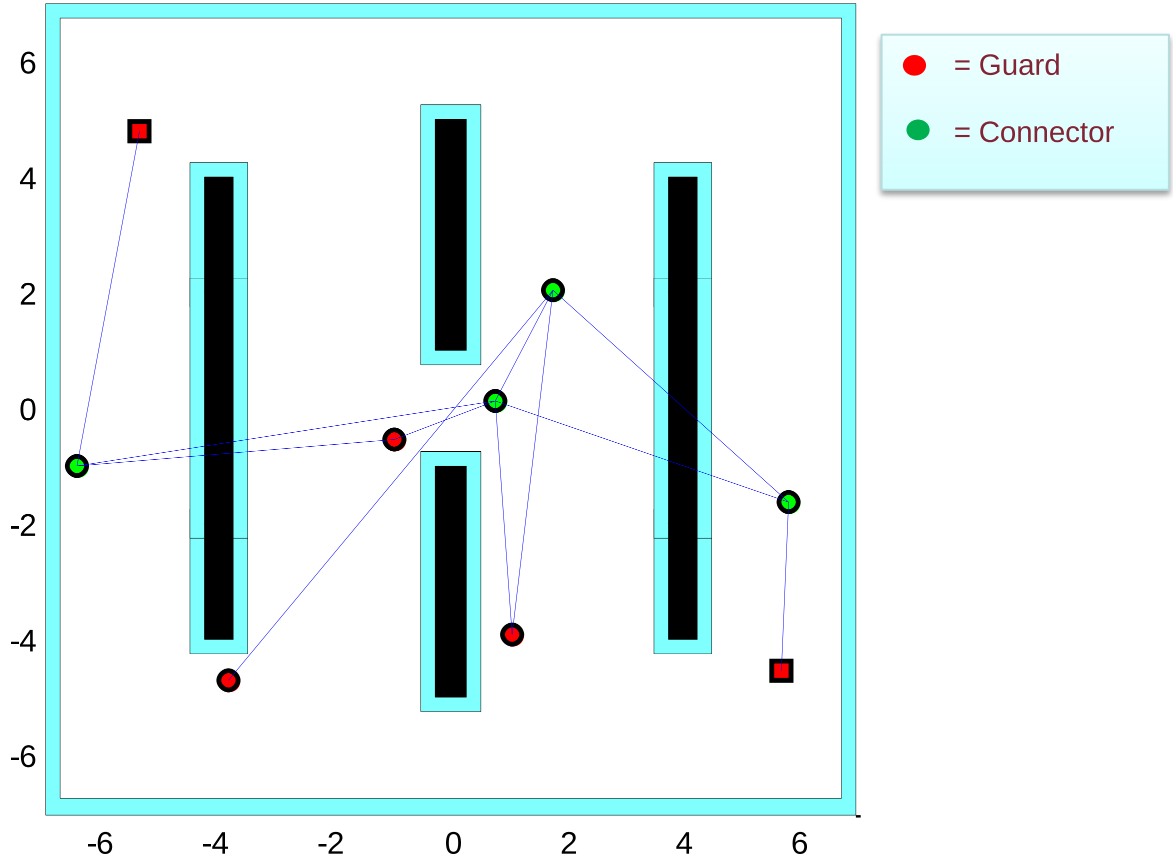
Visibility-Based Motion Planning

Start and Goal selection



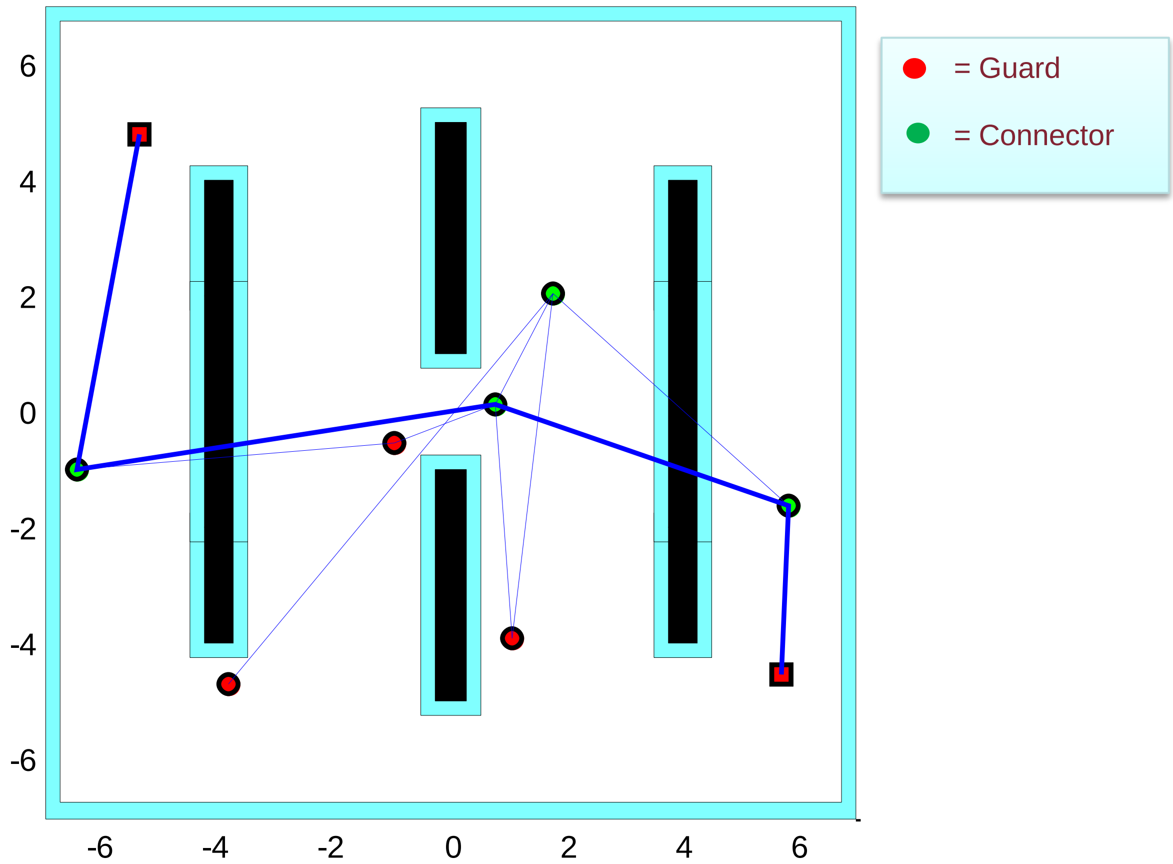
Visibility-Based Motion Planning

Roadmap construction



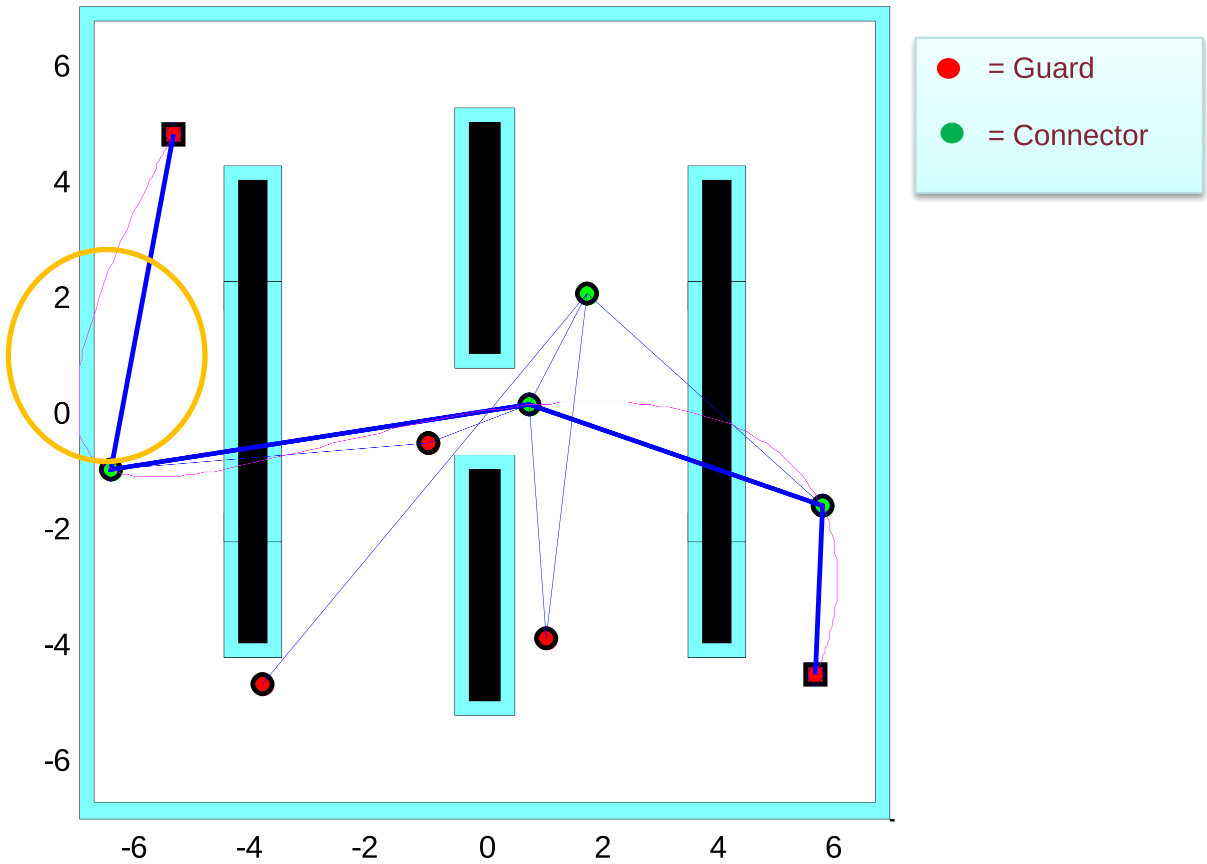
Visibility-Based Motion Planning

Path computed by A*



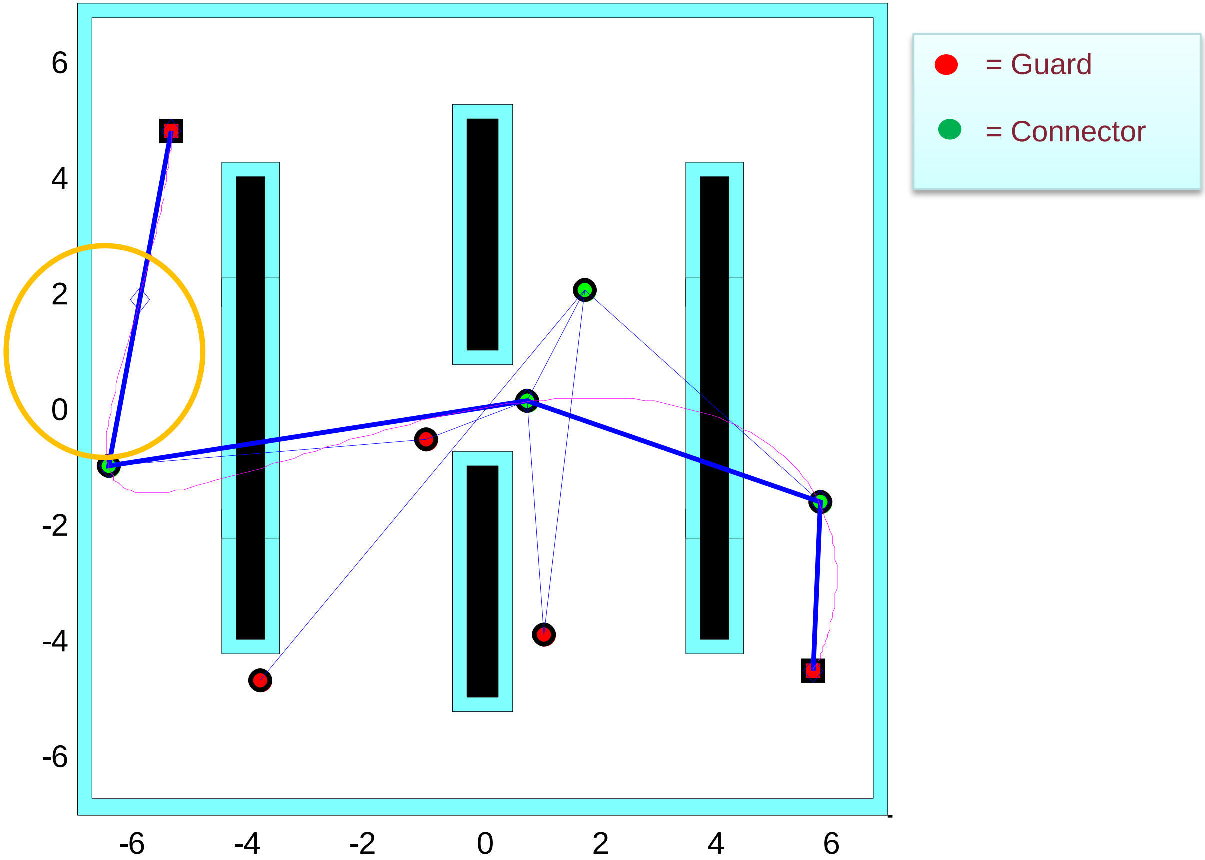
Visibility-Based Motion Planning

Path Computed after the Optimization



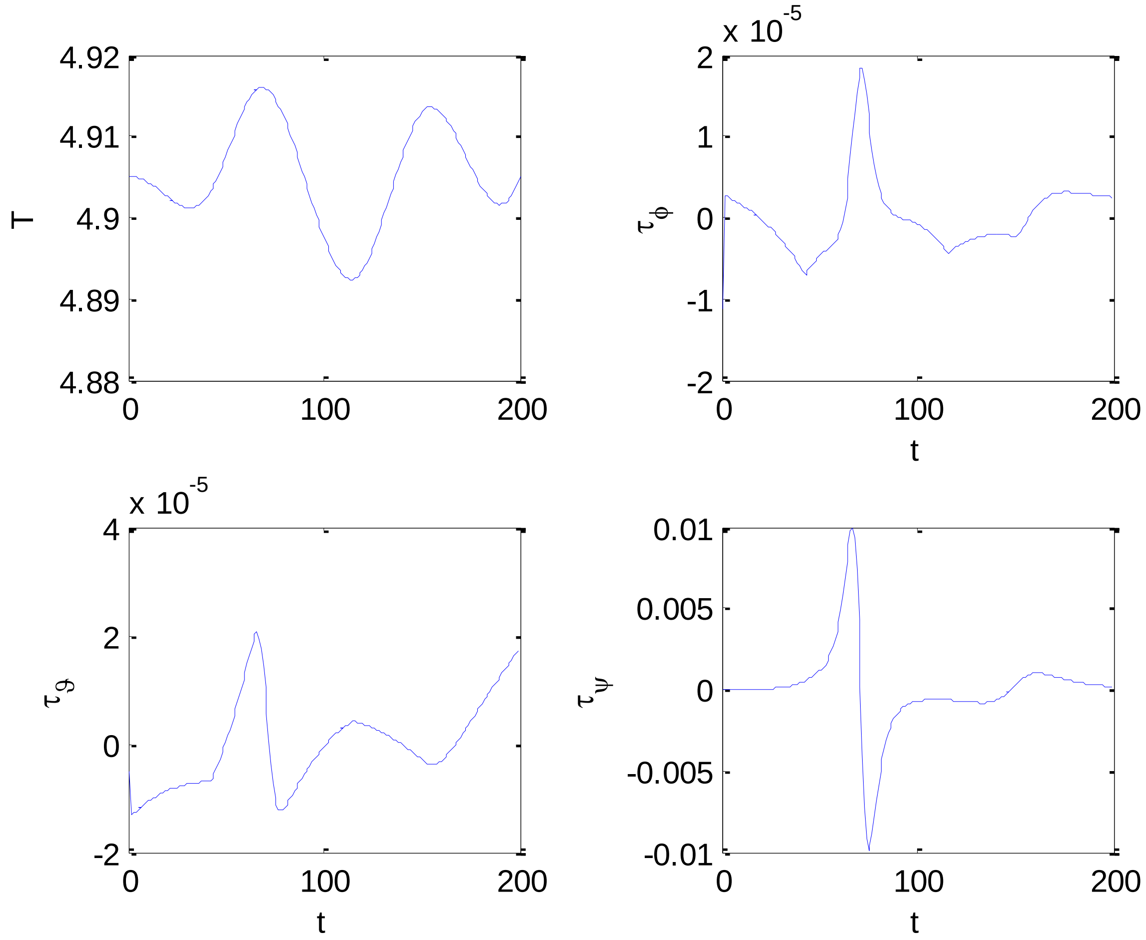
Visibility-Based Motion Planning

Safe Path Computed after the Optimization



Visibility-Based Motion Planning

Time Scaling



RRT-Based Motion Planning

Introduction

Key-idea: *incrementally grow a search tree by applying control inputs over short time intervals to reach new nodes*

Features:

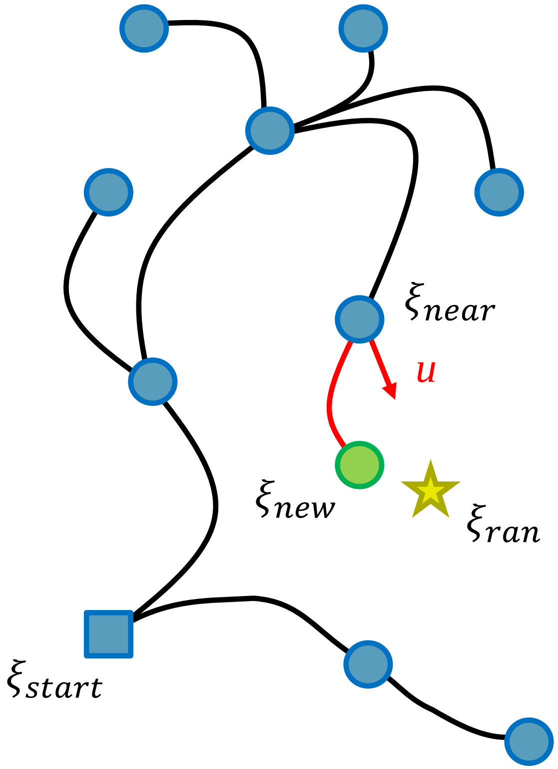
- Exploration biased toward unexplored portions of the space (Rapidly Exploring)
- Suitable for solving problems in high-dimensional spaces
- Takes into account both kinematic and dynamic constraints
- Generates a trajectory directly in the state space
- Provides control inputs needed to execute the trajectory

RRT-Based Motion Planning

Basic iteration

At each iteration:

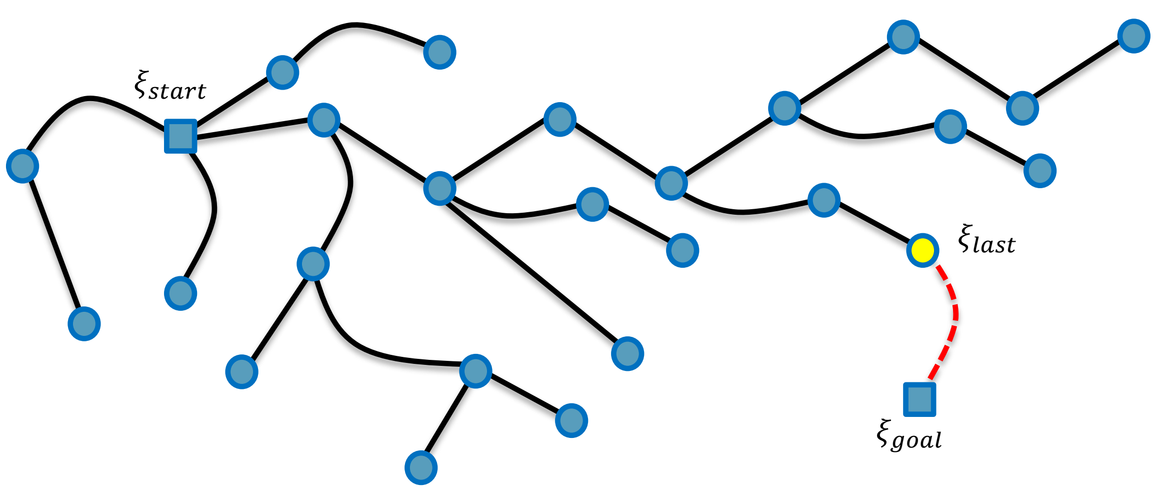
- Extract a random state ξ_{ran}
- Select ξ_{near} (according to the metric)
- Compute u to steer the robot towards ξ_{ran}
- Apply u for a time δt, reaching ξ_{new}
- If the path from ξ_{near} to ξ_{new} is safe, add ξ_{new} to the tree and save u
- Otherwise, discarded it



RRT-Based Motion Planning

Trajectory reconstruction

- Once $\|\xi_{last} - \xi_{goal}\| < \epsilon$, the algorithm stops
- The solution trajectory can then be found by traversing the tree backwards from ξ_{last} to ξ_{start}
- The last stretch might be computed (if necessary) using any local planner



RRT-Based Motion Planning

Exploration vs. Exploitation

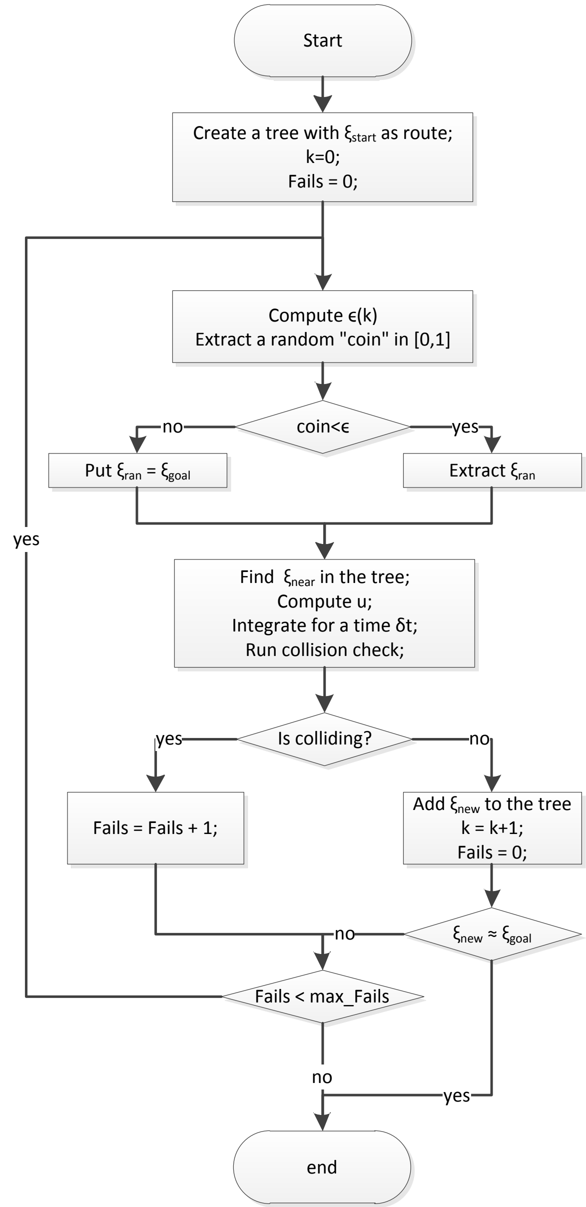
The tree will eventually cover the connected component of CS_{free} containing the start, coming arbitrarily close to any goal belonging to the same component

Convergence is often slow $\Longrightarrow$ switch between two phases:

- Exploration: the tree is expanded toward a random state
- Exploitation: the tree is expanded toward the goal ($\xi_{ran} = \xi_{goal}$)

ε-greedy strategy:

$$\epsilon = \frac{\epsilon_0}{1 + \alpha k} \Rightarrow \begin{cases} coin < \epsilon \rightarrow exploration \\ coin > \epsilon \rightarrow exploitation \end{cases}$$


Start
Create a tree with ξstart as route;
k=0;
Fails = 0;
Compute ϵ(k)
Extract a random "coin" in [0,1]
coin<ϵ
no
yes
Put ξran = ξgoal
Extract ξran
yes
Find ξnear in the tree;
Compute u;
Integrate for a time δt;
Run collision check;
Is colliding?
yes
no
Fails = Fails + 1;
Add ξnew to the tree
k = k+1;
Fails = 0;
no
ξnew ≈ ξgoal
Fails < max_Fails
no
yes
end

RRT-Based Motion Planning

Random inputs choice

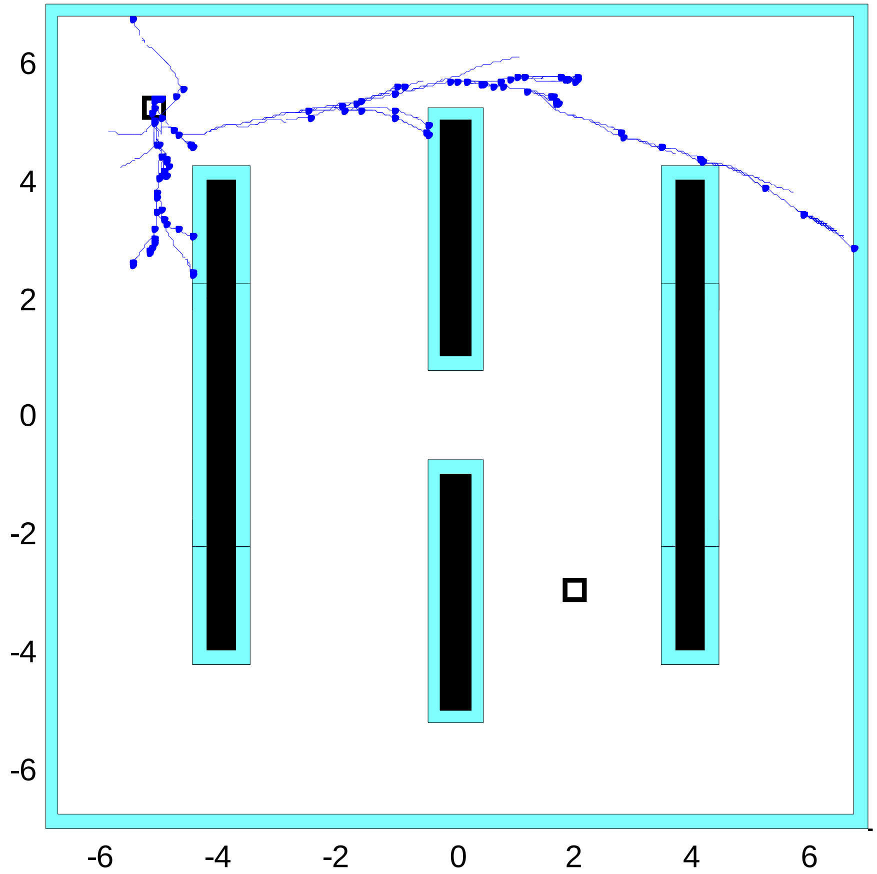
At each iteration:

- Extract 10 random values of the motors velocities between 0 and 1000 rad/s
- Compute the corresponding thrust and torques using input transformation
- Integrate the equations of motion (*ode45*) for a constant time $\delta t = 0.1$ s
- Choose the input vector that brings (safely) the robot closer to ξ_{ran}.

RRT-Based Motion Planning

Random inputs choice (Simulation Results)

6 hours later…



RRT-Based Motion Planning

Closed-loop system

Key-idea: *use a combination of the RRT planning along with a controller to ensure that the quadrotor moves toward* ξ_{ran}

Variable LQR controller: linearization around the current ξ_{near}

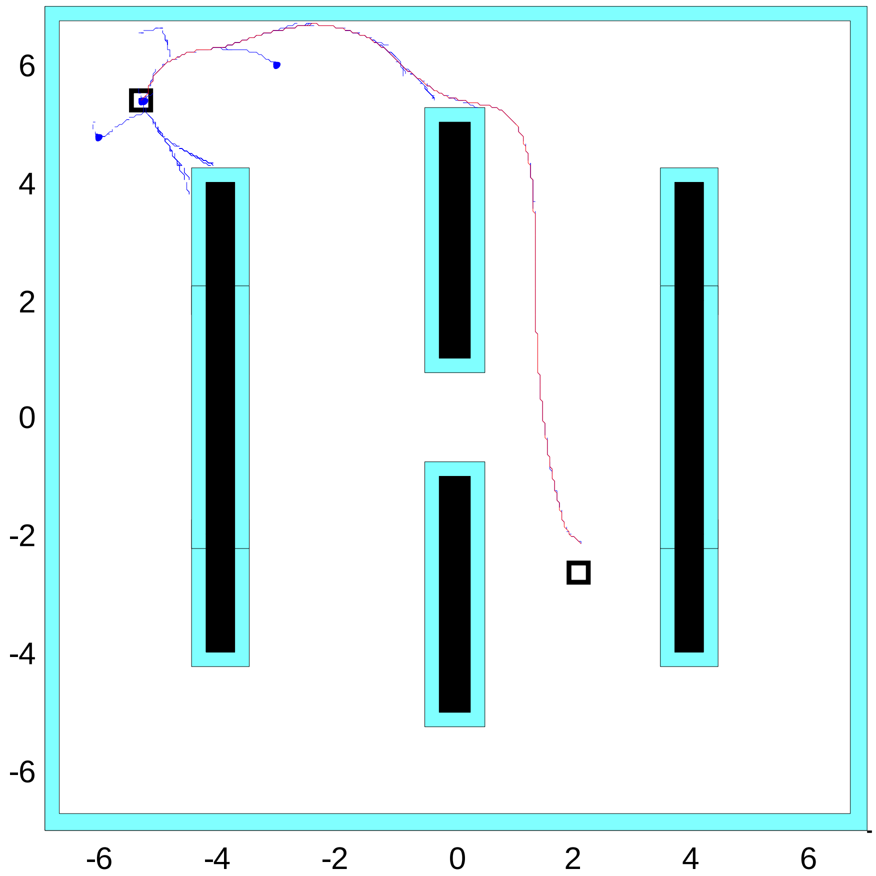
$$A = \left.\frac{\partial \dot{\xi}}{\partial \xi}\right|_{\substack{u=(mg \quad 0 \quad 0 \quad 0)^T \\ \xi=\xi_{near}}} \qquad B = \left.\frac{\partial \dot{\xi}}{\partial u}\right|_{\substack{u=(mg \quad 0 \quad 0 \quad 0)^T \\ \xi=\xi_{near}}}$$

K computed by Matlab® *lqr* function

$$u = (mg \quad 0 \quad 0 \quad 0)^T - K(\xi - \xi_{ran})$$

RRT-Based Motion Planning

Closed-loop system (Simulation Results)

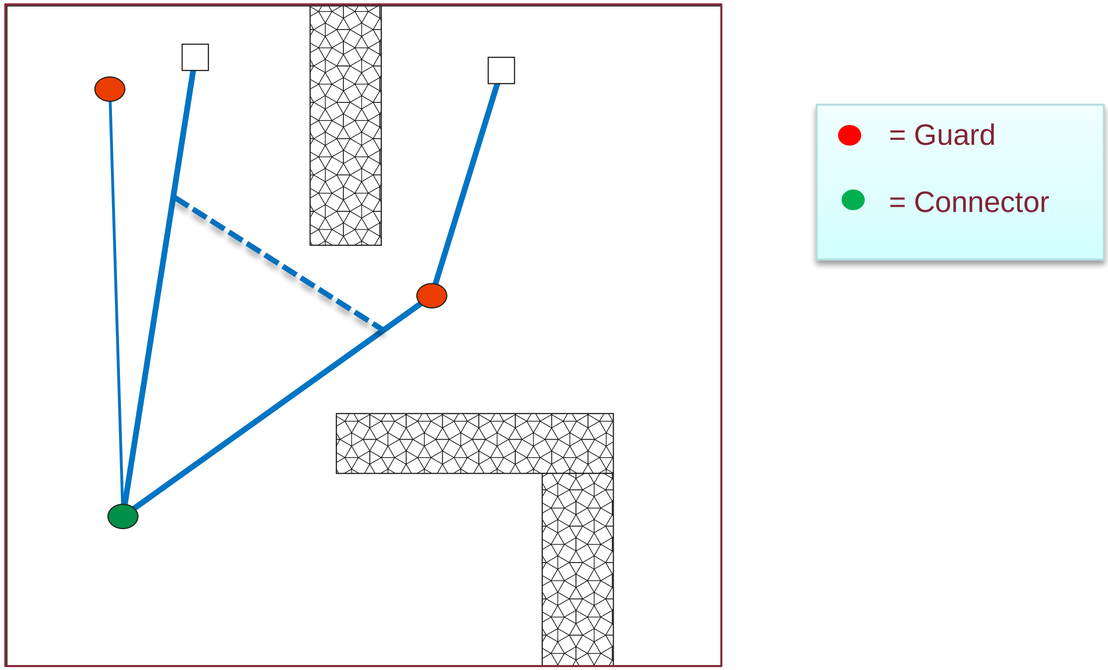


Conclusions

(1)

- Visibility algorithm is suitable for finding a roadmap with few nodes ⇨ easy to handle but, possibly not containing the shortest path

Smart-cutting



Conclusions
(2)

- The trajectory would be even smoother if the spline passed near to the nodes of the roadmap instead of interpolating them (more freedom to the optimizer)

- RRT planners seem to perform worse
- Enhancements may be achieved by:
  - Tuning δt, ϵ and the number of random inputs
  - Putting the system under the action of a controller
  - Tuning the gains Q and R of the LQR filter or by using any other controller (even non-linear)

THANKS FOR LISTENING