Autonomous navigation in dynamic uncertain environment using probabilistic models of perception and collision risk prediction

Chiara Fulgenzi

INRIA Rhône-Alpes, Grenoble France

PhD thesis presentation
June 8, 2009

Thesis Advisor	Christian Laugier
Co-advisor	Anne Spalanzani

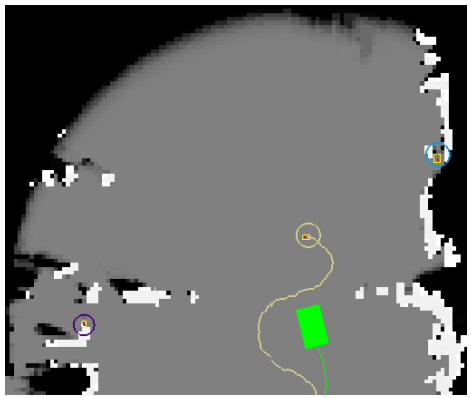
Autonomous navigation in unknown dynamic environment

Move autonomously in an unknown environment among moving vehicles or people

Autonomous navigation in unknown dynamic environment

Move autonomously in an unknown environment among moving vehicles or people

Autonomous navigation in unknown dynamic environment



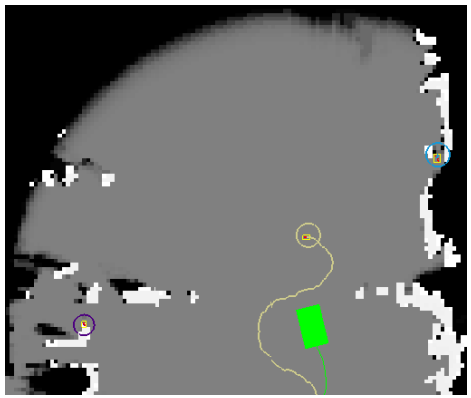
Static environment is explored:

- finite range
- sensor errors and accuracy
- hidden zones

Moving obstacles are detected and tracked:

- model uncertainty and errors
- model validity in time
- new obstacles entering the scene

Autonomous navigation in unknown dynamic environment



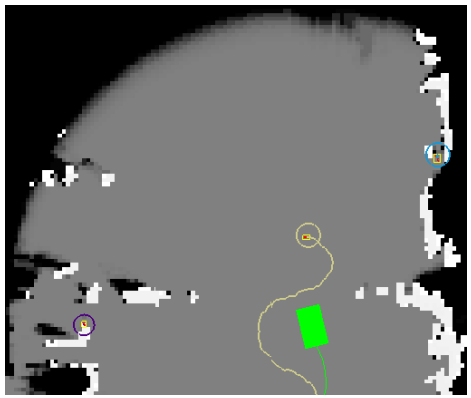
Static environment is explored:

- finite range
- sensor errors and accuracy
- hidden zones

Moving obstacles are detected and tracked:

- model uncertainty and errors
- model validity in time
- new obstacles entering the scene

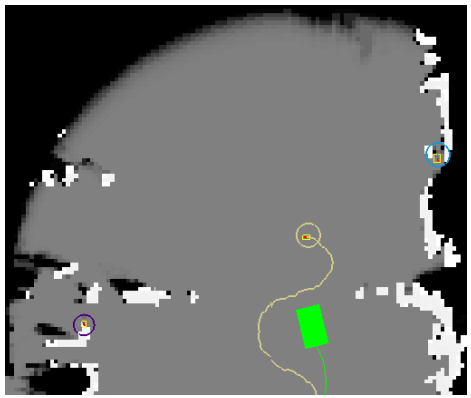
Autonomous navigation in unknown dynamic environment



Information about the environment is incomplete and uncertain in both time and space:

- Configuration Sensing
- Configuration Prediction
- Environment Sensing
- Environment Prediction

Autonomous navigation in unknown dynamic environment



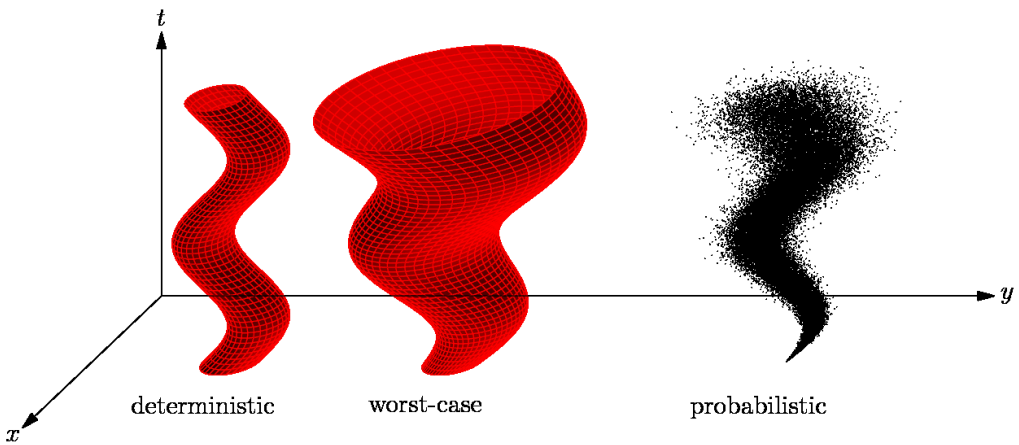
Information about the environment is incomplete and uncertain in both time and space:

- Configuration Sensing
- Configuration Prediction
- Environment Sensing
- Environment Prediction

Constraints

The navigation algorithm must take into account that:

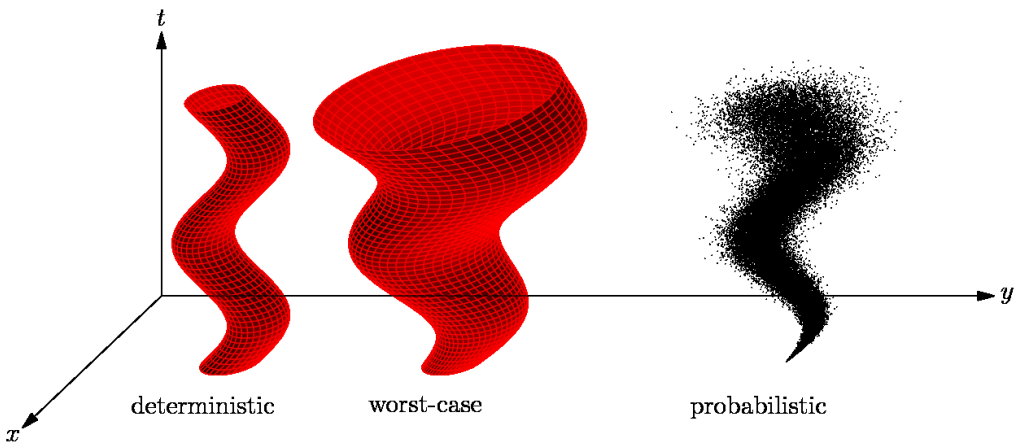
- The environment changes dynamically
  - limited time to take decisions
  - models and decisions must be continuously updated
- The current state of the environment is uncertain
  - represent the limits of information
  - represent the quality of information
- The future state is of the environment is uncertain
  - predict
  - represent the quality of information



Constraints

The navigation algorithm must take into account that:

- The environment changes dynamically
  - limited time to take decisions
  - models and decisions must be continuously updated
- The current state of the environment is uncertain
  - represent the limits of information
  - represent the quality of information
- The future state is of the environment is uncertain
  - predict
  - represent the quality of information



Thesis Contribution

Navigation algorithms based on the risk of collision

- Part I: Reactive method
  - dynamic occupancy grid (BOF, [Coué, 03])
  - probabilistic velocity obstacles (VO, [Shiller, 98])
- Part II: Motion Planning based on target-tracking
  - mapping and target tracking ([Vu, 06])
  - RRTs ([LaValle, 99])
  - Partial Motion Planning ([Fraichard, 05])
- Part III: Motion Planning based on typical patterns
  - Hidden Markov Models ([Vasquez, 07])
  - Gaussian Processes ([Tay, 07])

Outline

Reactive Navigation

Reactive Navigation Methods: only the next control is computed at each step.

Potential Fields [Khatib,85]	
	Vector Field Histogram [Borenstein, 91]
Curvature Velocity [Simmons, 96]	
	Lane Curvature Velocity [Simmons, 98]
Dynamic Window [Fox, 97]	
	Nearness Diagram [Montano,00]
Obstacle Restriction [Minguez,05]	
	Inevitable Collision States [Fraichard, 03]
Velocity Obstacles [Shiller, 98]	
	Dynamic Object Velocity [Montano, 05]

DYNAMIC ENVIRONMENT
PERCEPTION UNCERTAINTY

Reactive Navigation

Reactive Navigation Methods: only the next control is computed at each step.

Potential Fields [Khatib,85]	
	Vector Field Histogram [Borenstein, 91]
Curvature Velocity [Simmons, 96]	
	Lane Curvature Velocity [Simmons, 98]
Dynamic Window [Fox, 97]	
	Nearness Diagram [Montano,00]
Obstacle Restriction [Minguez,05]	
	Inevitable Collision States [Fraichard, 03]
Velocity Obstacles [Shiller, 98]	
	Dynamic Object Velocity [Montano, 05]

DYNAMIC ENVIRONMENT

PERCEPTION UNCERTAINTY

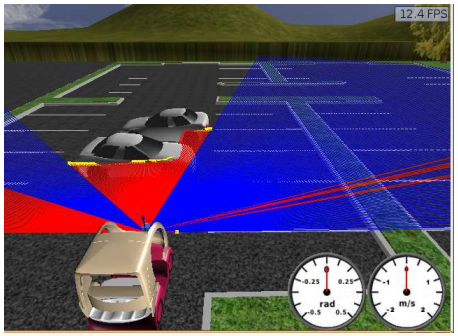
Reactive Navigation

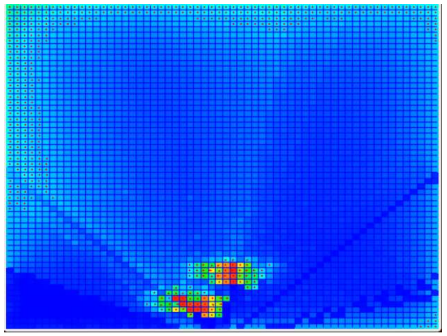
Reactive Navigation Methods: only the next control is computed at each step.

- Potential Fields [Khatib,85]
- Vector Field Histogram [Borenstein, 91]
- Curvature Velocity [Simmons, 96]
- Lane Curvature Velocity [Simmons, 98]
- Dynamic Window [Fox, 97]
- Nearness Diagram [Montano,00]
- Obstacle Restriction [Minguez,05]
- Inevitable Collision States [Fraichard, 03]
- Velocity Obstacles [Shiller, 98]
- Dynamic Object Velocity [Montano, 05]

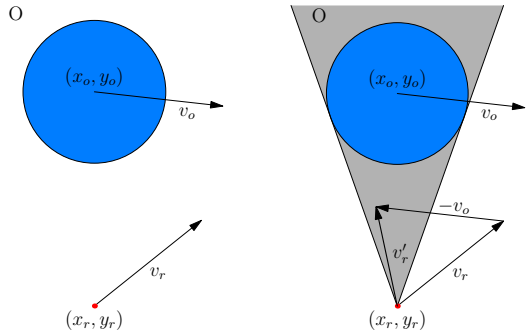
DYNAMIC ENVIRONMENT
PERCEPTION UNCERTAINTY

Related Work





BOF



VO

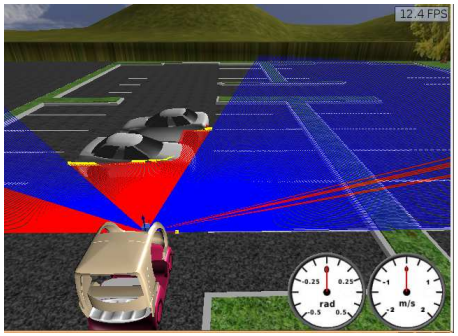
The Bayesian Occupancy Filter [Coué, 03]

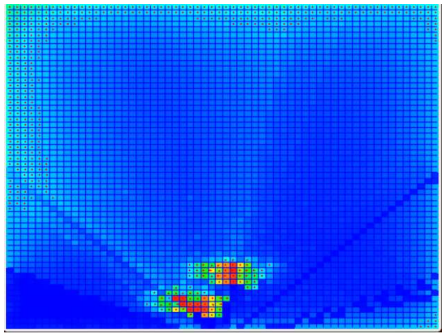
- The probability of occupancy in the space
- A distribution function over a discrete set of velocities

Velocity Obstacles [Shiller, 98]

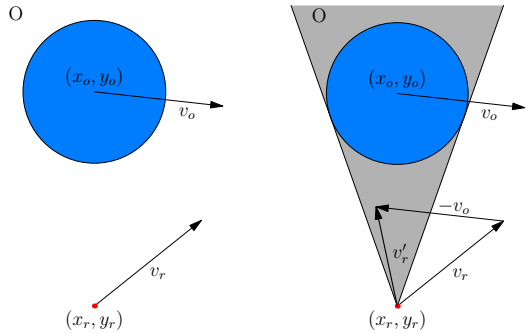
- Geometric method
- Tells if a linear velocity of the robot is in collision with moving obstacles

Related Work





BOF



VO

The Bayesian Occupancy Filter [Coué, 03]

- The probability of occupancy in the space
- A distribution function over a discrete set of velocities

Velocity Obstacles [Shiller, 98]

- Geometric method
- Tells if a linear velocity of the robot is in collision with moving obstacles

Probabilistic Velocity Obstacles

Compute the risk of collision for linear velocities of the robot

- with a cell-to-cell approach
- using a clustered grid

$$P(t_{coll} \in (t_0, t] | v_r, v_n) = \max_{o \in SO_t} P_o(Occ) \cdot P_o(v_n)$$

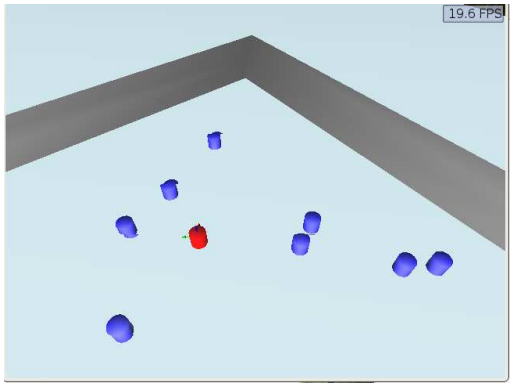
$$P(t_{coll} \in (t_0, t] | v_r) = 1 - \prod_{n=1}^{N} (1 - P_{coll}(v_r, v_n))$$

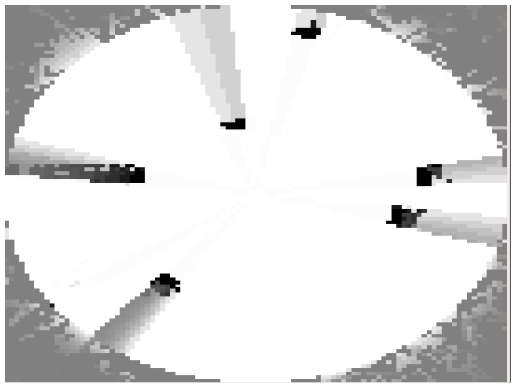
Navigation function dependent on the risk of collision

$$T_{safe}(v) = \epsilon + T_{brake}(v)$$

$$K^*(v) = K(v, goal) \cdot ((1.0 - P(t_{coll}(v) \in (t_0, t_0 + k\tau])) \cdot \frac{k\tau}{\max_v(T_{safe}(v))}$$

Results: simulation setup





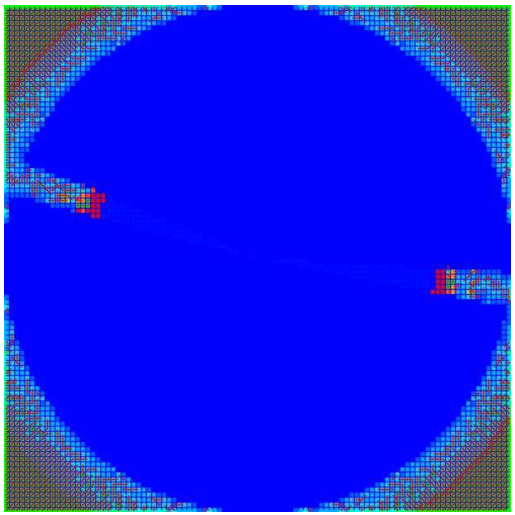

- Holonome Robot
- Distance sensor with limited range

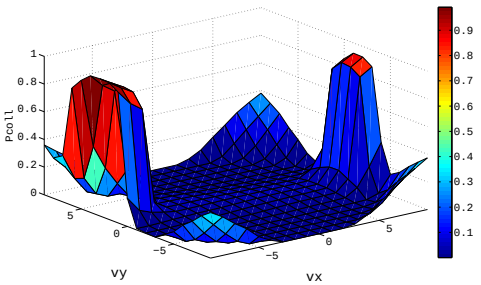
Results: cell-to-cell VS clustering

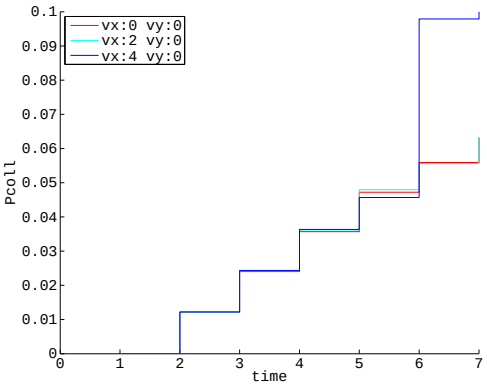
BOF

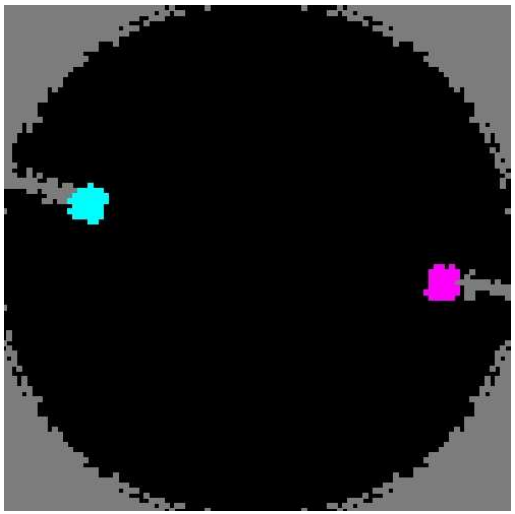
$P(t_{coll} \in (0, 3])$

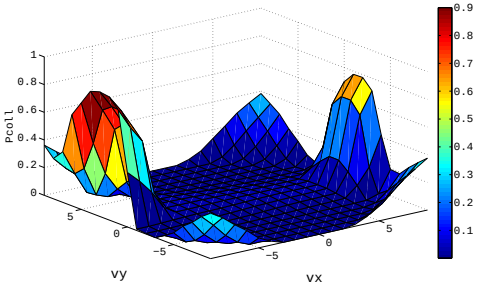
$P(t_{coll} \in (0, t])$

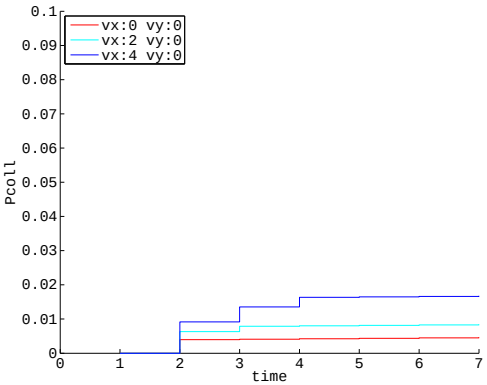






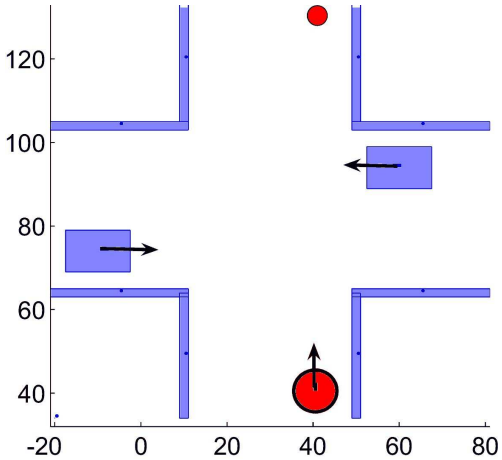




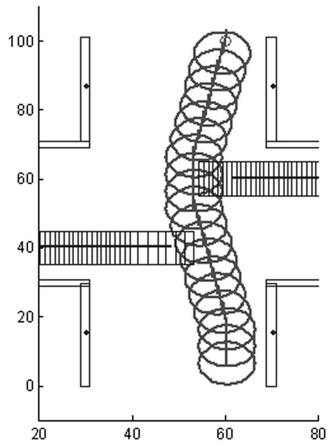


Results

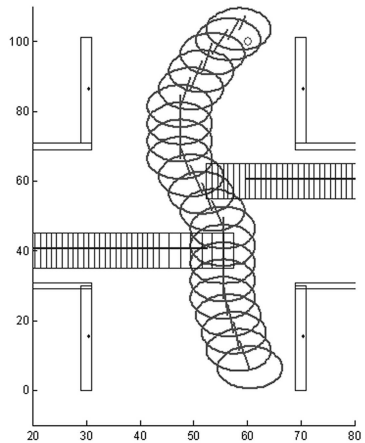
scenario



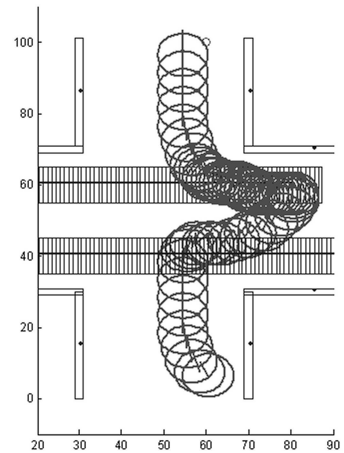
perfect



vel uncertainty



+ limited range



Conclusions

Complexity

- cell-to-cell: depends on the size of the grid, not on the number of obstacles
- computation is parallelizable for each (v_r, v_n)

Contributions

- computation of the risk of collision for linear velocities from a dynamic occupancy grid
- uncertainty rising from occlusion, limited range, velocity estimation uncertainty directly influences the choice of the next control

Conclusions

Limitations

One-step ahead reasoning

- efficiency issues
- safety issues

Outline

Motion planning in unknown dynamic environment

Path planning approaches: compute a complete path

Static or Dynamic, but deterministic environment

Combinatorial	Sampling Based
RoadMaps [Canny,88]	Probabilistic RM [Kavraki,95]
Cell decomposition [Schwartz,83]	Discretized C_{free} [Brooks,85]
Potential Fields [Khatib,80]	Randomized PF [Barraquand,90]
	Ariadne's clew [Bessière,93]
	RRTs [LaValle,99]

Unknown but Static environment

MDP	Anytime-RRTs [Ferguson,06]
POMDP [Foka, 07]	Particle-RRTs [Melchior,07]

Motion planning in unknown dynamic environment

Path planning approaches: compute a complete path

Static or Dynamic, but deterministic environment

Combinatorial	Sampling Based
RoadMaps [Canny,88]	Probabilistic RM [Kavraki,95]
Cell decomposition [Schwartz,83]	Discretized C_{free} [Brooks,85]
Potential Fields [Khatib,80]	Randomized PF [Barraquand,90]
	Ariadne's clew [Bessière,93]
	RRTs [LaValle,99]

Unknown but Static environment

MDP	Anytime-RRTs [Ferguson,06]
POMDP [Foka, 07]	Particle-RRTs [Melchior,07]

Motion planning in unknown dynamic environment

D* [Stentz, 95]

- computes the best path with the current knowledge
- execution time depends on the dimensions of the space

Hybrid methods: plan-react-replan

- static environment is known
- time for planning and replanning is not limited

Partial Motion Planning [Fraichard, 03]

- gives a partial safe path at anytime
- satisfies real-time constraints
- deterministic environment

Comparison

		Environment		Configuration		Time
		Sensing	Prediction	Sensing	Prediction	Constraints
Complete	D*	+	+	-	-	-
	MDP	-	+	-	+	-
	POMDP	+	+	+	+	-
Sampling Based	Anytime-RRTs	-	+-	-	-	-
	Particle-RRTs	+	-	-	+	-
	PMP	-	+-	-	-	+

Motivation

		Environment		Configuration		Time
		Sensing	Prediction	Sensing	Prediction	Constraints
Complete	D*	+	+	-	-	-
	MDP	-	+	-	+	-
	POMDP	+	+	+	+	-
Sampling Based	Anytime-RRTs	-	+-	-	-	-
	Particle-RRTs	+	-	-	+	-
	PMP	-	+-	-	-	+
Needed Properties		+	+	-	-	+

PMP → real-time constraints

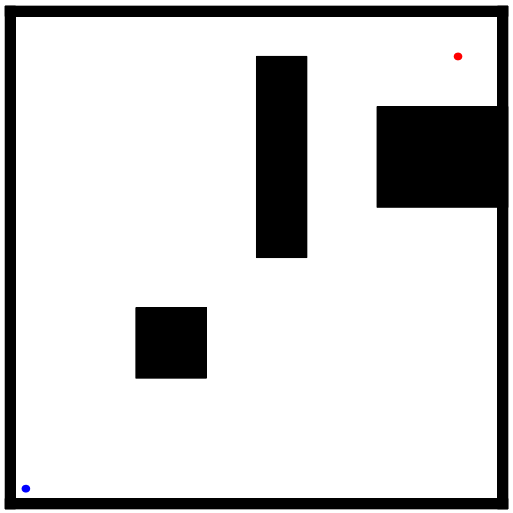
Complete methods are not suitable in dynamic environment → Sampling Based Method

RRT → incremental
→ non-holonomic constraints

Integrate and update uncertain information in the decision process

Rapidly-exploring Random Trees

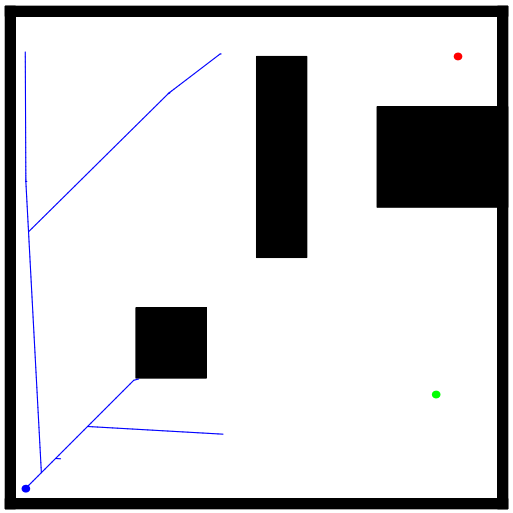
1. Each configuration q is deterministically known: $q \in \mathcal{C}_{free}$ or $q \in \mathcal{C}_{obs}$
2. A *random* point $p \in \mathcal{C}_{free}$ is chosen
3. The nearest node of the current tree is expanded toward p



4. The search ends when the goal configuration is in the tree or it continues till some other condition is satisfied
5. The path is retrieved from the goal to the root

Rapidly-exploring Random Trees

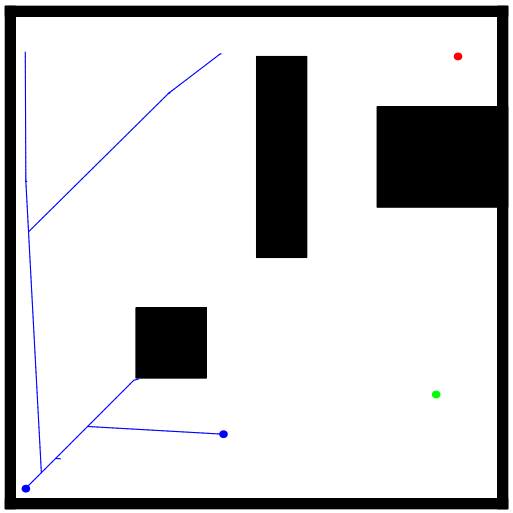
1. Each configuration q is deterministically known: $q \in \mathcal{C}_{free}$ or $q \in \mathcal{C}_{obs}$
2. A *random* point $p \in \mathcal{C}_{free}$ is chosen
3. The nearest node of the current tree is expanded toward p



4. The search ends when the goal configuration is in the tree or it continues till some other condition is satisfied
5. The path is retrieved from the goal to the root

Rapidly-exploring Random Trees

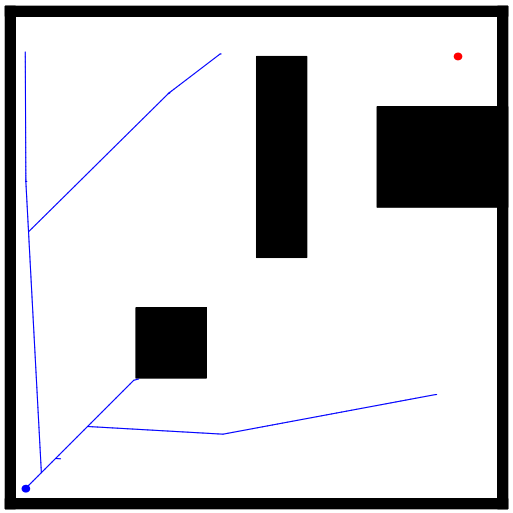
1. Each configuration q is deterministically known: $q \in \mathcal{C}_{free}$ or $q \in \mathcal{C}_{obs}$
2. A *random* point $p \in \mathcal{C}_{free}$ is chosen
3. The nearest node of the current tree is expanded toward p



4. The search ends when the goal configuration is in the tree or it continues till some other condition is satisfied
5. The path is retrieved from the goal to the root

Rapidly-exploring Random Trees

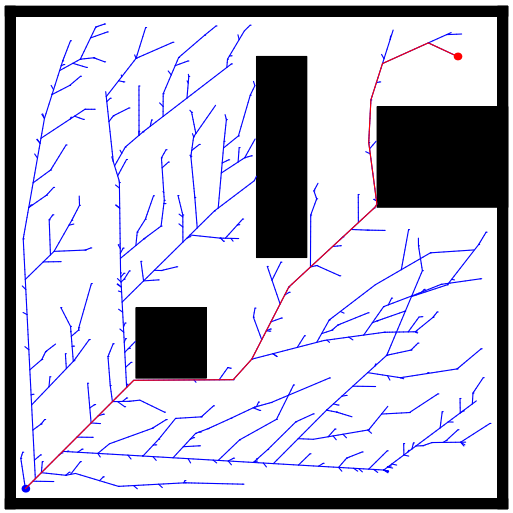
1. Each configuration q is deterministically known: $q \in \mathcal{C}_{free}$ or $q \in \mathcal{C}_{obs}$
2. A *random* point $p \in \mathcal{C}_{free}$ is chosen
3. The nearest node of the current tree is expanded toward p



4. The search ends when the goal configuration is in the tree or it continues till some other condition is satisfied
5. The path is retrieved from the goal to the root

Rapidly-exploring Random Trees

1. Each configuration q is deterministically known: $q \in \mathcal{C}_{free}$ or $q \in \mathcal{C}_{obs}$
2. A *random* point $p \in \mathcal{C}_{free}$ is chosen
3. The nearest node of the current tree is expanded toward p

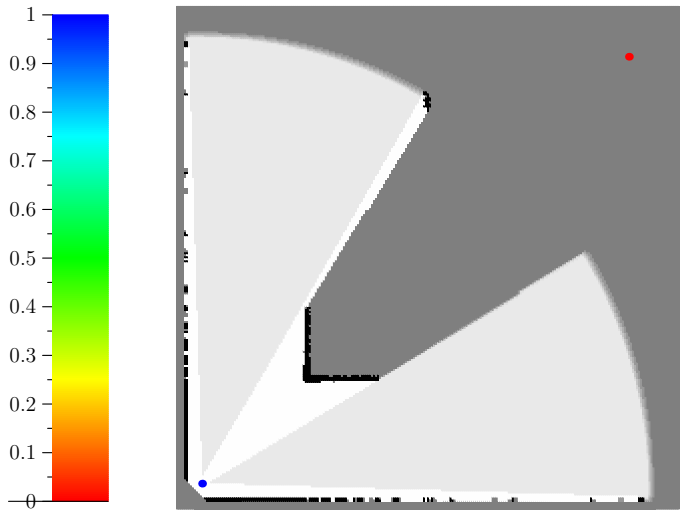


4. The search ends when the goal configuration is in the tree or it continues till some other condition is satisfied

5. The path is retrieved from the goal to the root

Probabilistic RRTs

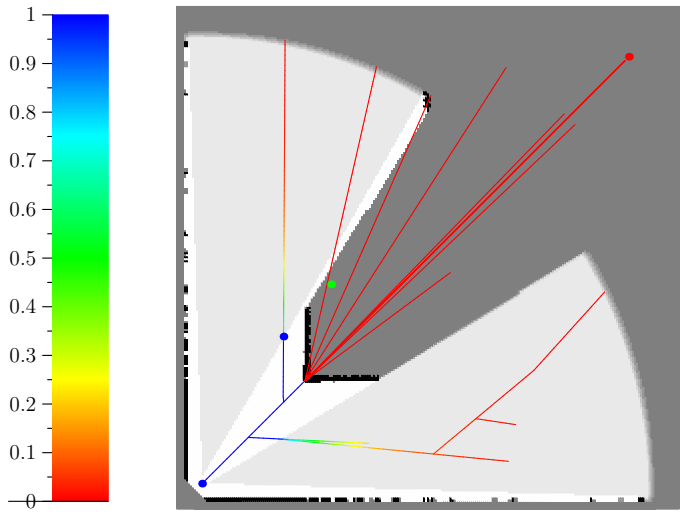
1. Each configuration q has a probability of collision $P_{coll}(q)$
2. A *random* point p is chosen
3. The node with the *most likely* path is expanded toward p



4. The search ends when the available time is out
5. The path is retrieved from the *most likely* node to the root

Probabilistic RRTs

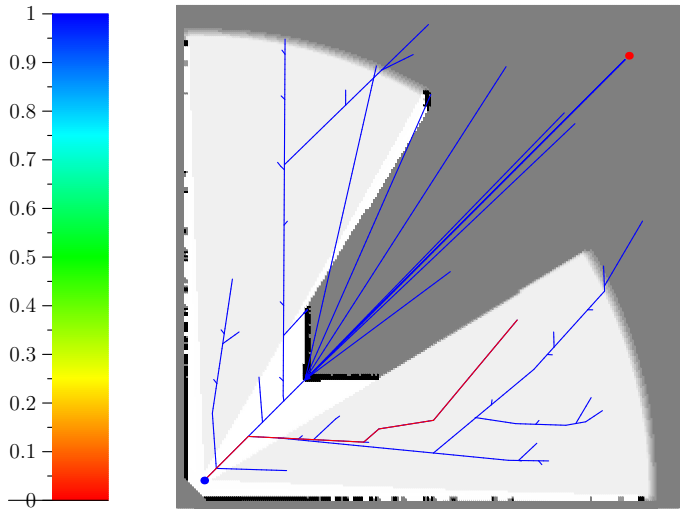
1. Each configuration q has a probability of collision $P_{coll}(q)$
2. A *random* point p is chosen
3. The node with the *most likely* path is expanded toward p



4. The search ends when the available time is out
5. The path is retrieved from the *most likely* node to the root

Probabilistic RRTs

1. Each configuration q has a probability of collision $P_{coll}(q)$
2. A *random* point p is chosen
3. The node with the *most likely* path is expanded toward p



4. The search ends when the available time is out
5. The path is retrieved from the *most likely* node to the root

Probability of collision

$P_{coll}(q)$: risk of collision of $q = (s, t)$, state s at time t

$$P_{coll}(q) = P_{cs}(s) + (1 - P_{cs}(s)) \cdot P_{cd}(s, t) =$$
$$P_{occ}(s) + (1 - P_{cs}(s)) \cdot \left(1 - \prod_{o=1}^{O} (1 - P_{cd}(s, t, o))\right)$$

$P_{coll}(\pi)$: risk of collision of path π from root q_0 to node q_N

$$P_{coll}(\pi) = 1 - \prod_{i=0}^{N} (1 - P_{coll}(q_i))$$

$L_\pi(q_N)$: *probability of success* of path π

$$L_\pi(q_N) = 1 - P_{coll}(\pi)$$

Probability of collision

$P_{coll}(q)$: risk of collision of $q = (s, t)$, state s at time t

$$P_{coll}(q) = P_{cs}(s) + (1 - P_{cs}(s)) \cdot P_{cd}(s, t) =$$
$$P_{occ}(s) + (1 - P_{cs}(s)) \cdot \left(1 - \prod_{o=1}^{O} (1 - P_{cd}(s, t, o))\right)$$

$P_{coll}(\pi)$: risk of collision of path π from root q_0 to node q_N

$$P_{coll}(\pi) = 1 - \prod_{i=0}^{N} (1 - P_{coll}(q_i))$$

$L_\pi(q_N)$: *probability of success* of path π

$$L_\pi(q_N) = 1 - P_{coll}(\pi)$$

Probability of collision

$P_{coll}(q)$: risk of collision of $q = (s, t)$, state s at time t

$$P_{coll}(q) = P_{cs}(s) + (1 - P_{cs}(s)) \quad \cdot \quad P_{cd}(s,t) =$$
$$P_{occ}(s) + (1 - P_{cs}(s)) \quad \cdot \quad \left(1 - \prod_{o=1}^{O} (1 - P_{cd}(s,t,o))\right)$$

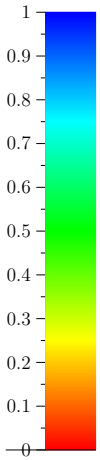
$P_{coll}(\pi)$: risk of collision of path π from root q_0 to node q_N

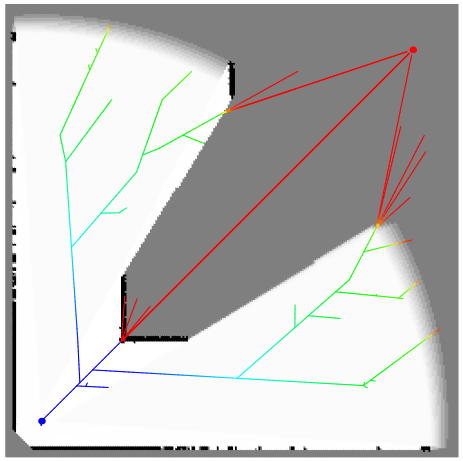
$$P_{coll}(\pi) = 1 - \prod_{i=0}^{N} (1 - P_{coll}(q_i))$$

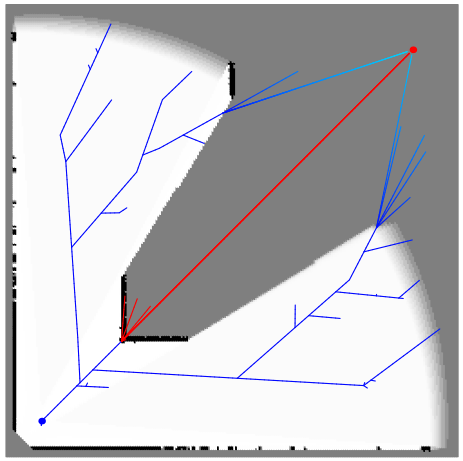
$L_\pi(q_N)$: *probability of success* of path π

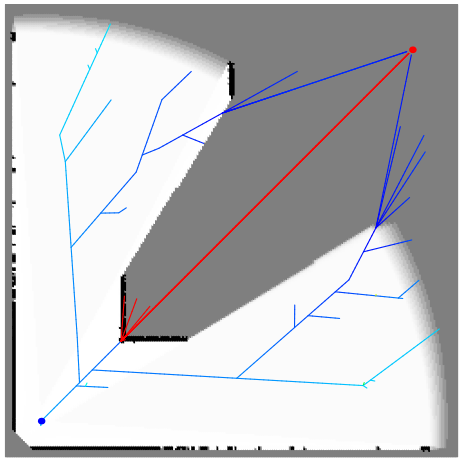
$$L_\pi(q_N) = 1 - P_{coll}(\pi)$$

Most likely node, most likely path







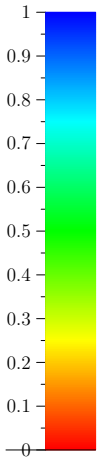


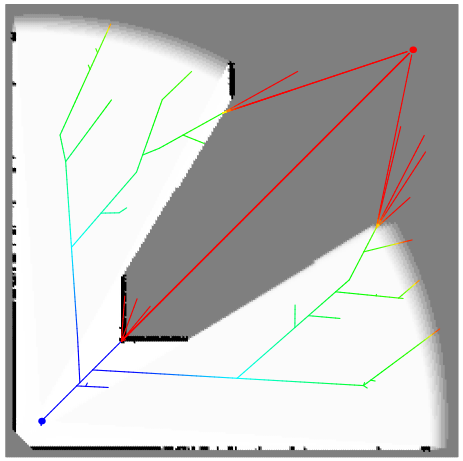
$L_\pi(q_N) = 1 - P_{coll}(\pi)$ $\tilde{L}_\pi(q_N) = \sqrt[N]{L_\pi(q_N)}$ $w_{q_N} = \frac{\tilde{L}_\pi(q_N)}{dist(q_0, q_N, Goal)}$

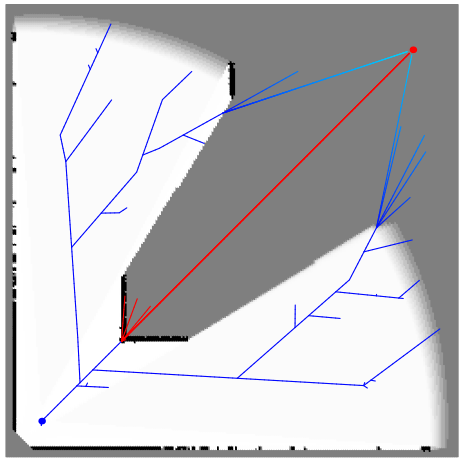
A weight is computed for each partial path:

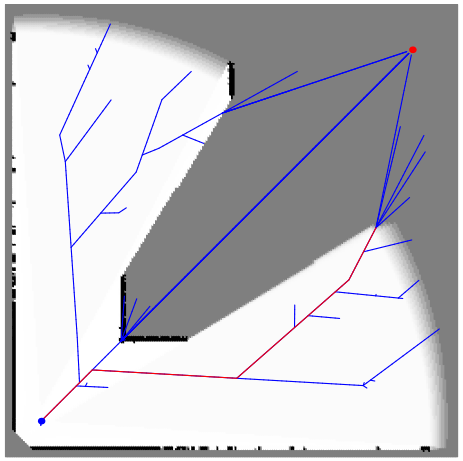
1. The likelihood is *normalized*
2. The estimated lenght of the path to *Goal* is considered
3. The path with the highest weight is chosen

Most likely node, most likely path









$L_\pi(q_N) = 1 - P_{coll}(\pi)$ $\tilde{L}_\pi(q_N) = \sqrt[N]{L_\pi(q_N)}$ $w_{q_N} = \frac{\tilde{L}_\pi(q_N)}{dist(q_0, q_N, Goal)}$

A weight is computed for each partial path:

1. The likelihood is *normalized*
2. The estimated lenght of the path to *Goal* is considered
3. The path with the highest weight is chosen

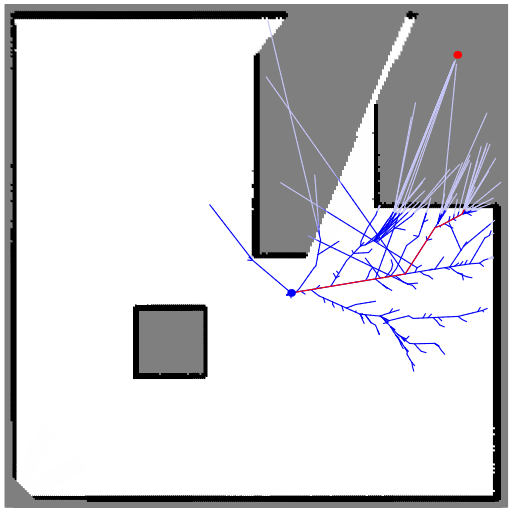
Updating the tree

environment is explored
environment is dynamic $\rightarrow$ update the tree and the path with new information

Partial Motion Planning [Fraichard, 05]

real-time constraints
safety issues (no ICS) $\rightarrow$ deterministic prediction up to infinite time

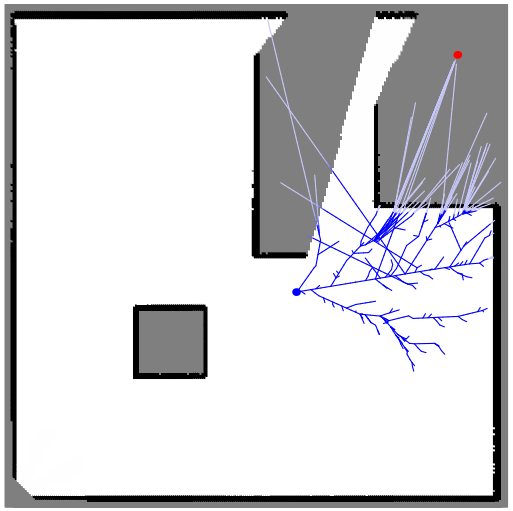
Updating the tree



Given the path chosen at the previous step, $\pi(q_N) = \{q_0 ... q_N\}$:

1. The robot moves to q_1
2. The search tree is pruned: q_1 is the new root
3. The tree is updated according to the new information
4. The tree is grown in the remaining time
5. A new path is chosen

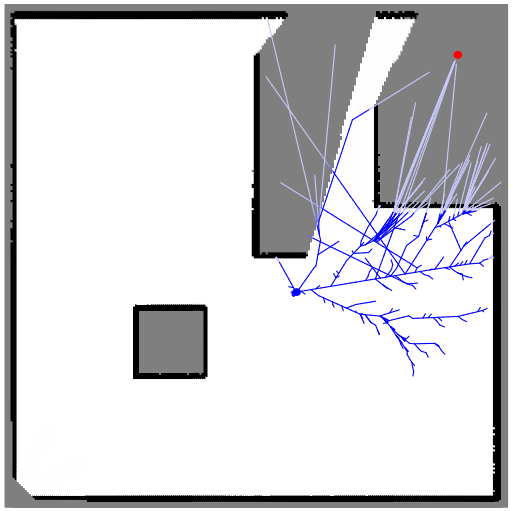
Updating the tree



Given the path chosen at the previous step, $\pi(q_N) = \{q_0 ... q_N\}$:

1. The robot moves to q_1
2. The search tree is pruned: q_1 is the new root
3. The tree is updated according to the new information
4. The tree is grown in the remaining time
5. A new path is chosen

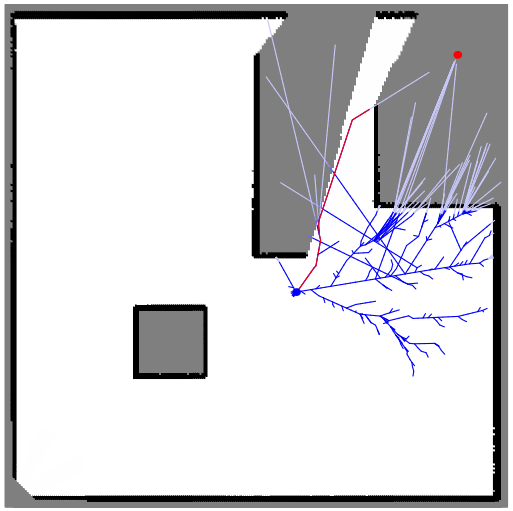
Updating the tree



Given the path chosen at the previous step, $\pi(q_N) = \{q_0 ... q_N\}$:

1. The robot moves to q_1
2. The search tree is pruned: q_1 is the new root
3. The tree is updated according to the new information
4. The tree is grown in the remaining time
5. A new path is chosen

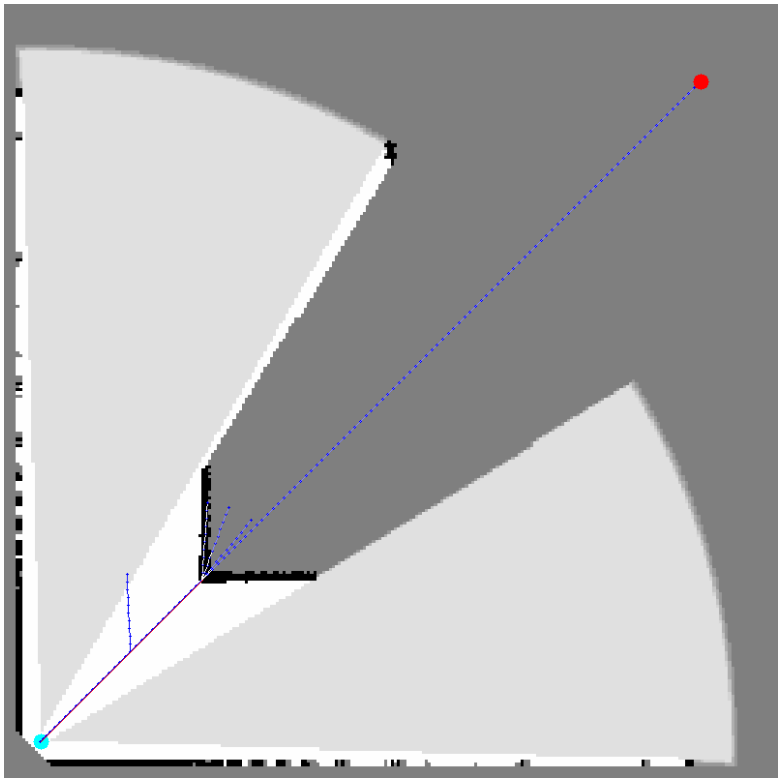
Updating the tree



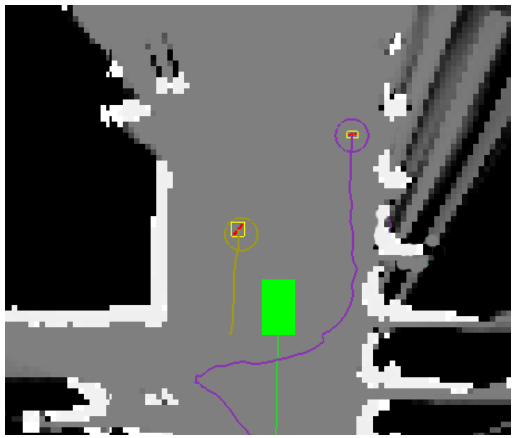
Given the path chosen at the previous step, $\pi(q_N) = \{q_0 ... q_N\}$:

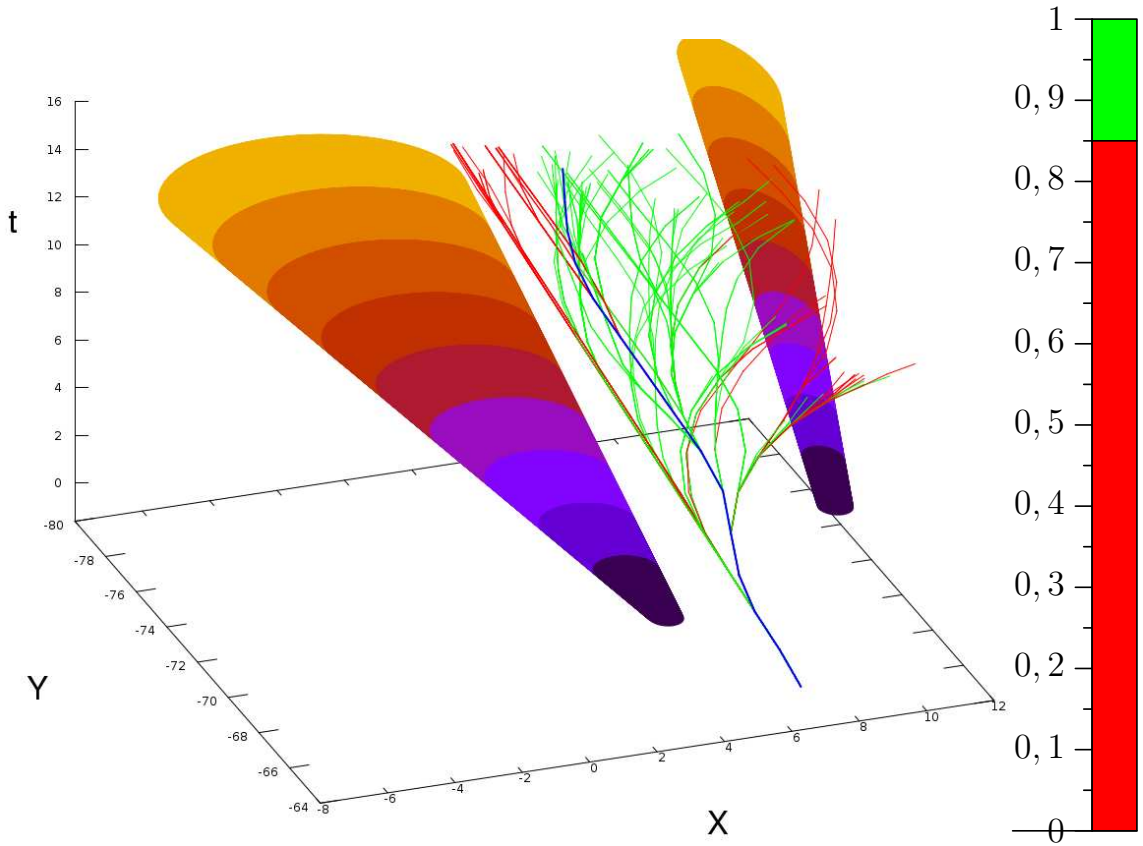
1. The robot moves to q_1
2. The search tree is pruned: q_1 is the new root
3. The tree is updated according to the new information
4. The tree is grown in the remaining time
5. A new path is chosen

Probabilistic RRTs: example



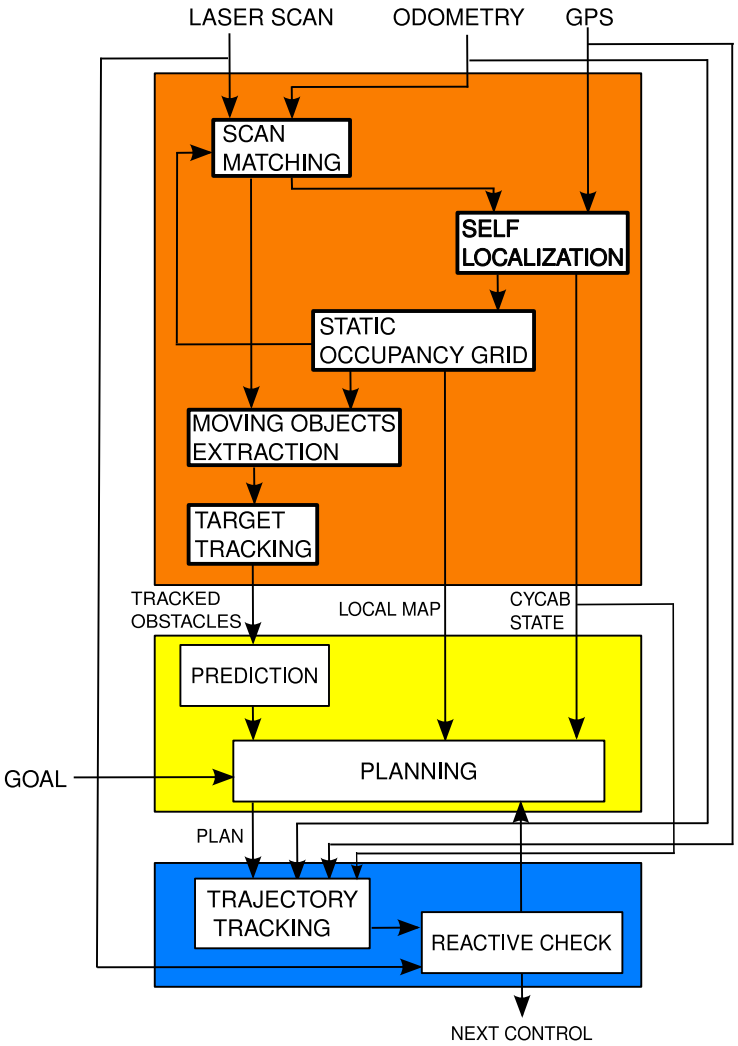
PRRTs with short term prediction

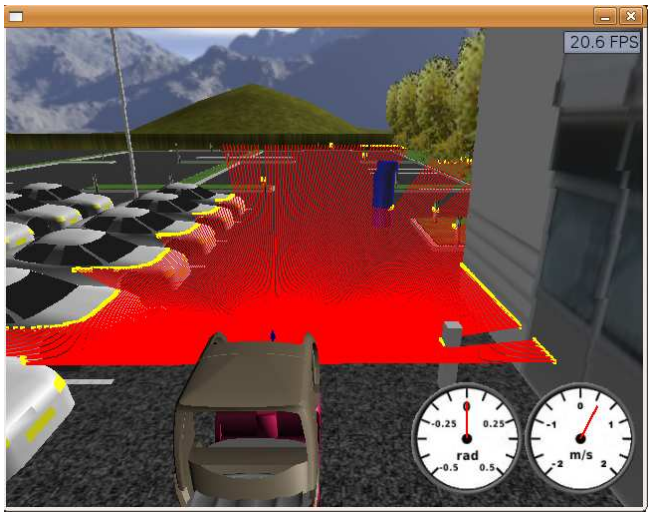




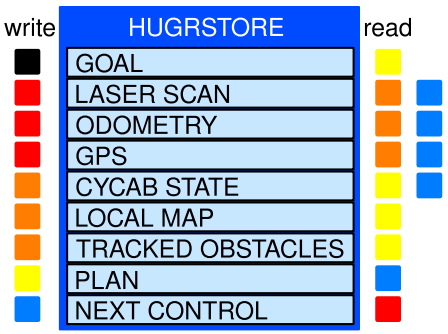
[Vu,07]

PP-RRTs with target tracking



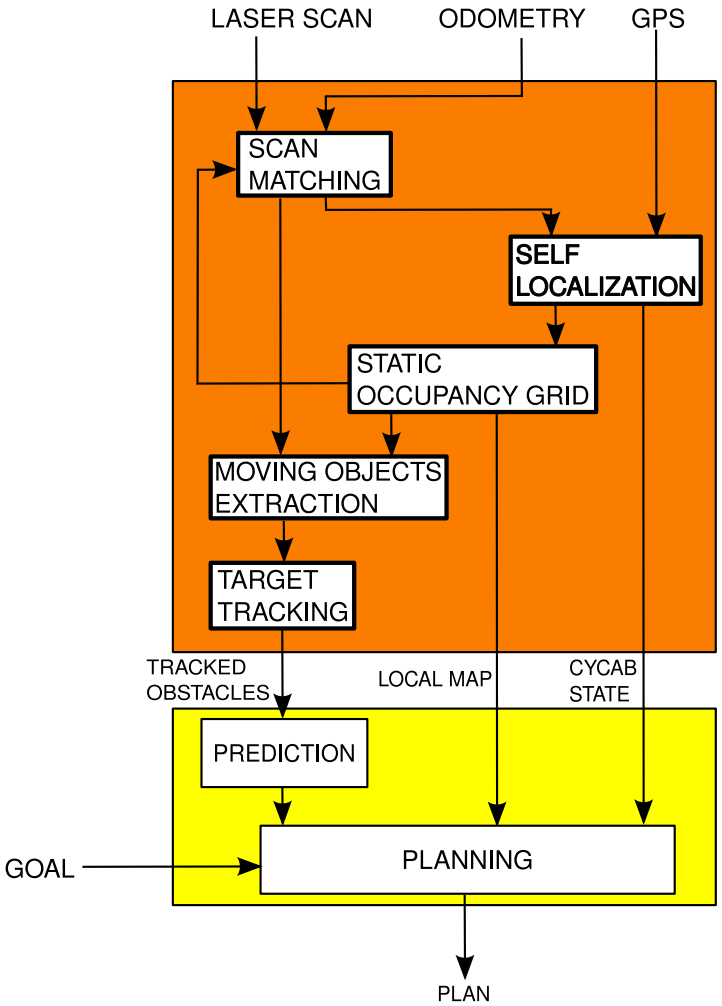


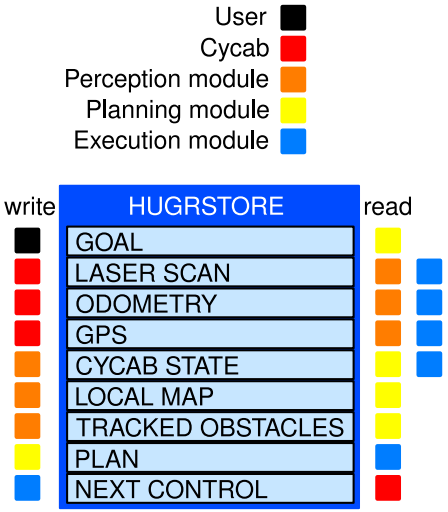
User
Cycab
Perception module
Planning module
Execution module



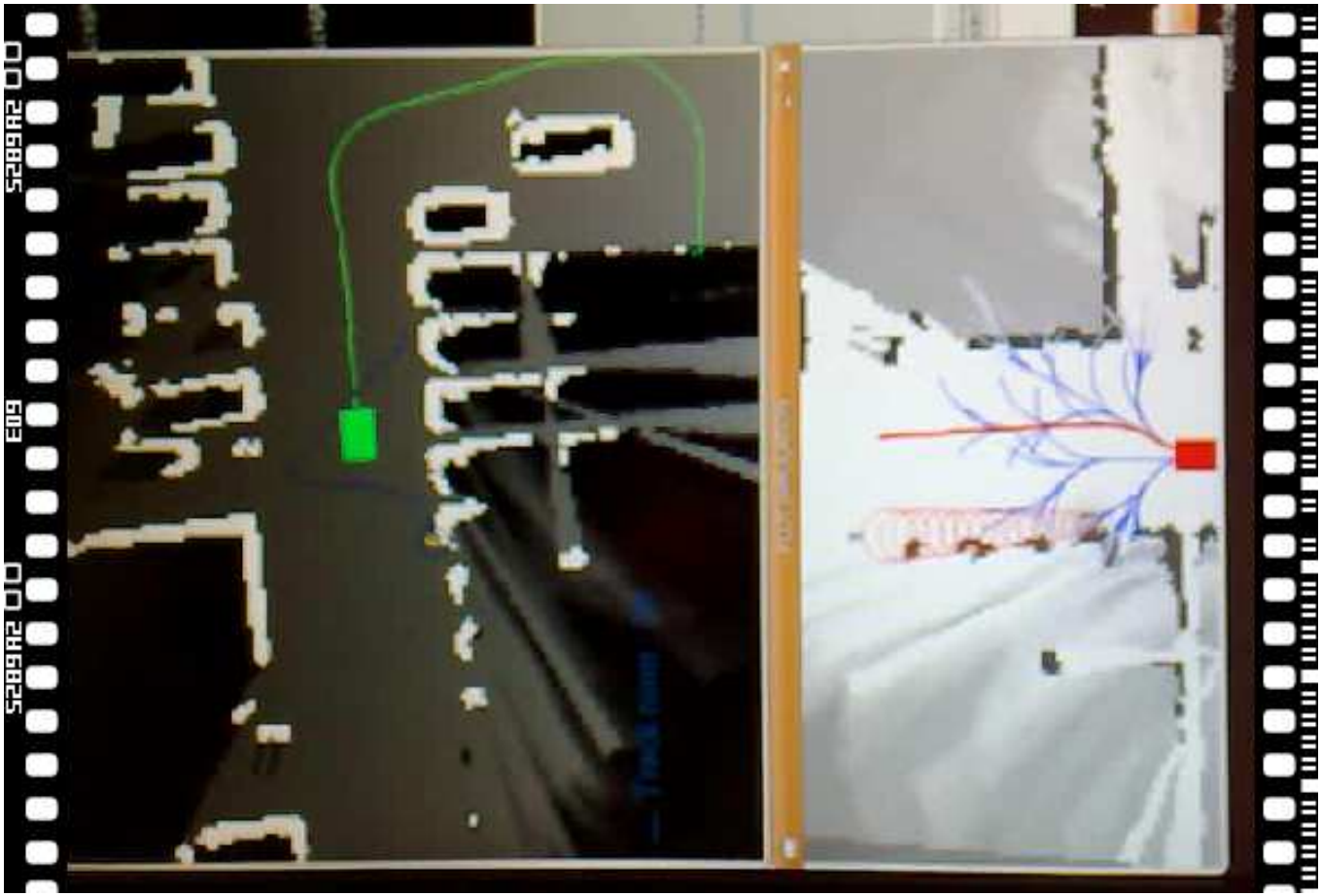
[click me]

Probabilistic RRTs with target tracking



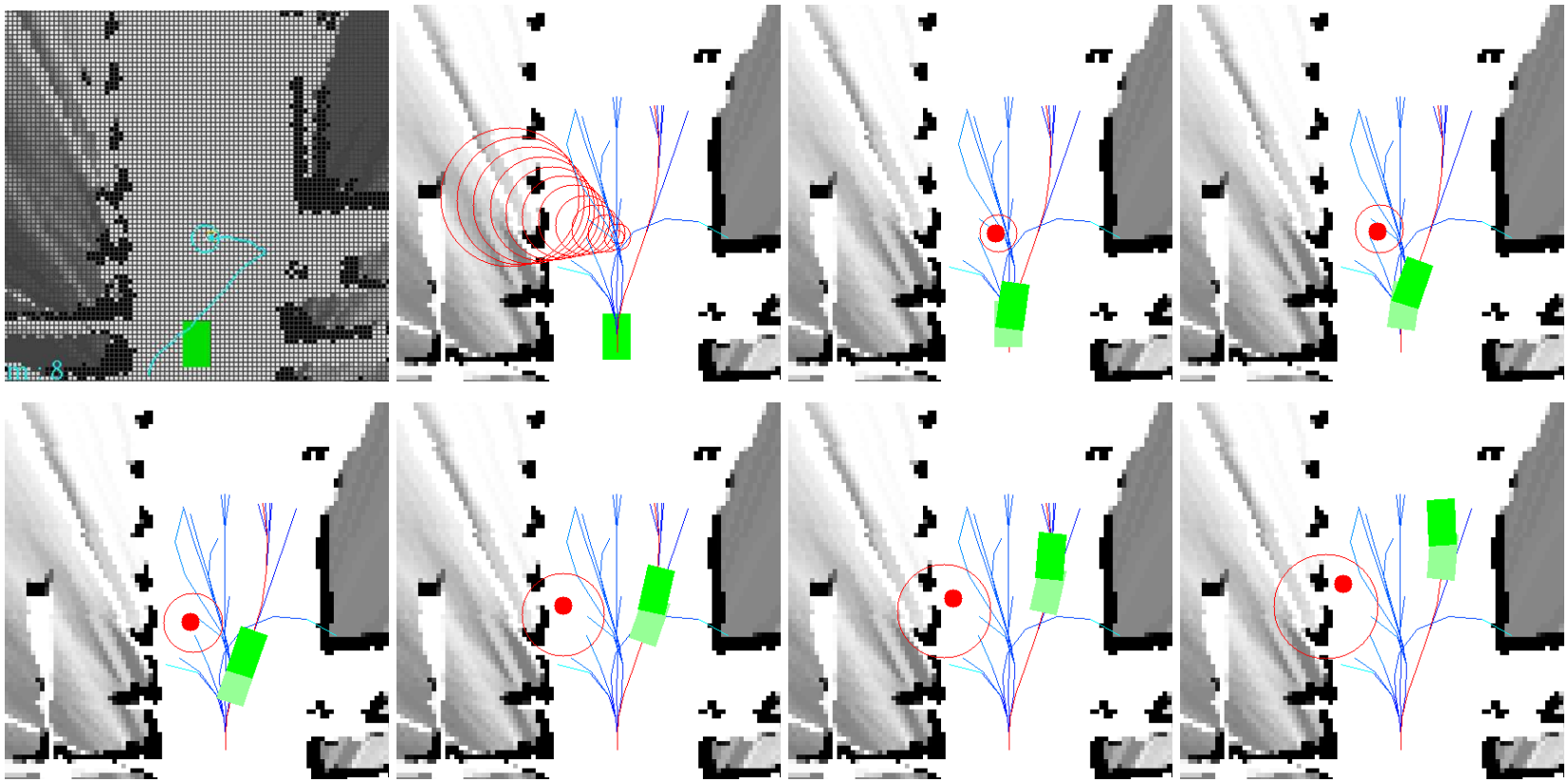


Probabilistic RRTs with target tracking: results

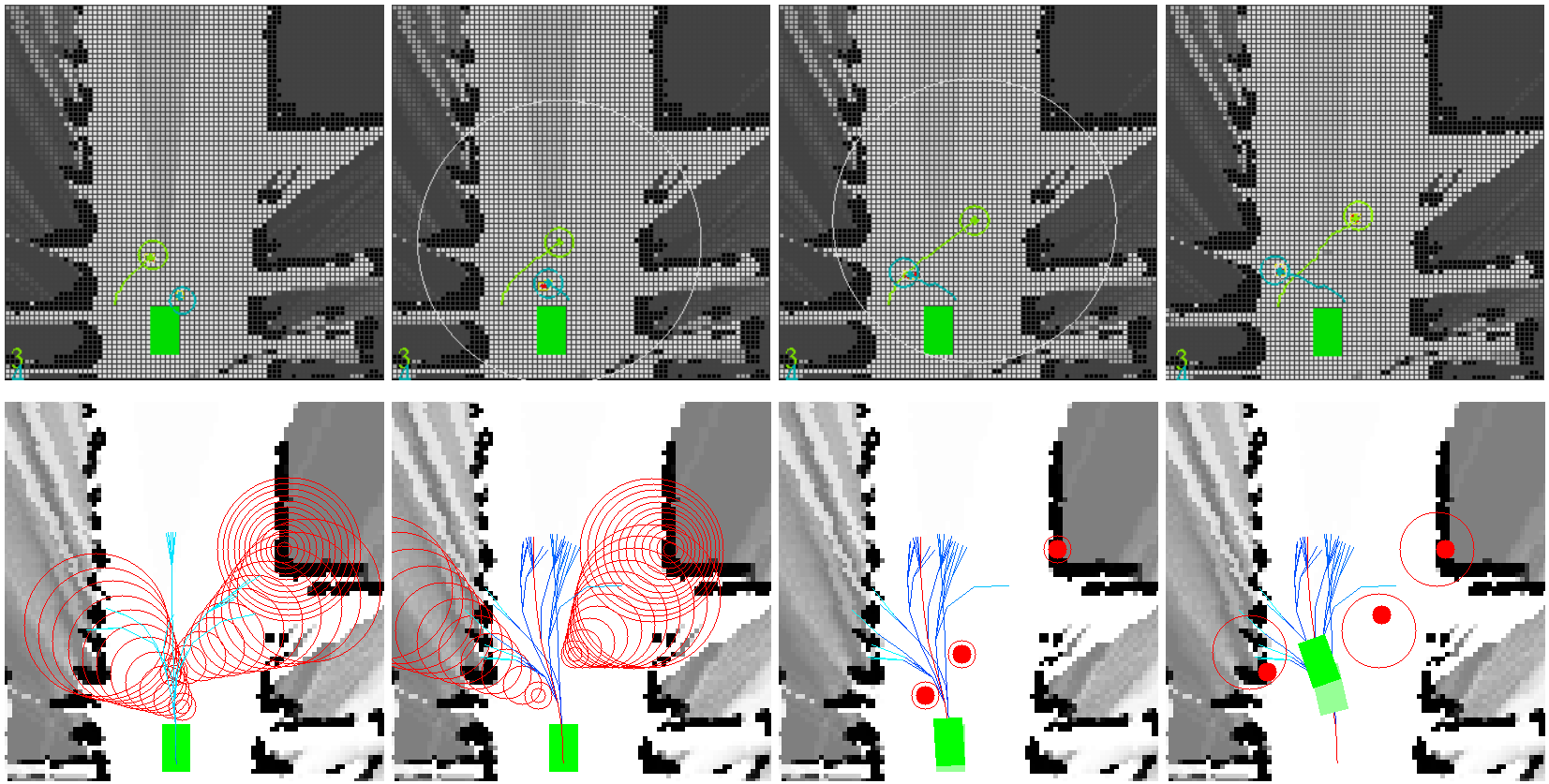


Target Tracking: $\sim 10Hz$ PRRT: $2Hz$

Probabilistic RRTs with target tracking



Probabilistic RRTs with target tracking



Conclusions

Contributions

- Probabilistic RRTs:
  - Static environment uncertainty $\rightarrow$ occupancy grid
  - Velocity estimation uncertainty $\rightarrow$ target-tracking based
- On-line information and decision updating

Target-tracking based prediction

PROs	CONs
low *a priori* information	short-term
reactive to behavior changes	linear

Conclusions

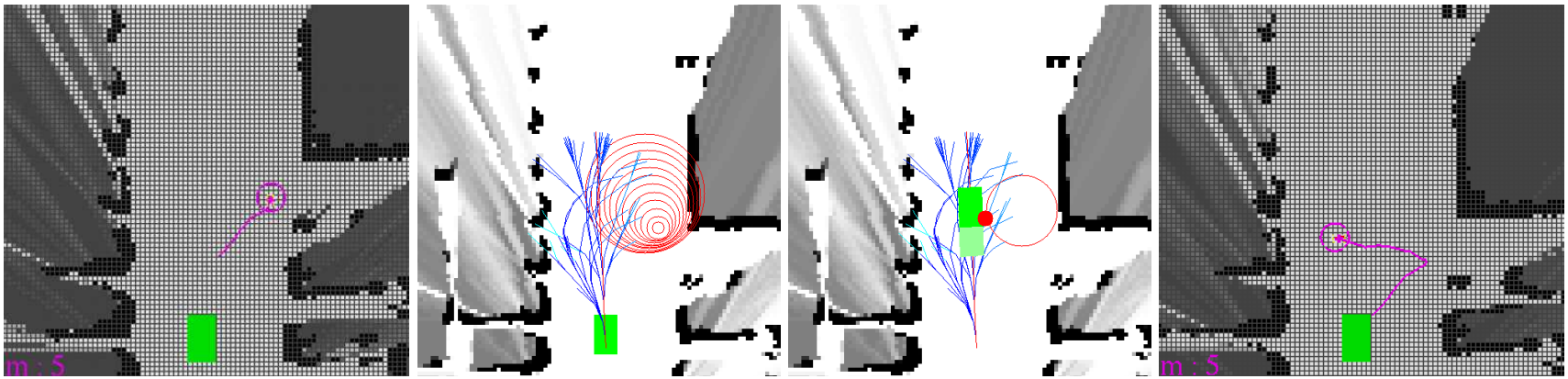
Contributions

- Probabilistic RRTs:
  - Static environment uncertainty $\longrightarrow$ occupancy grid
  - Velocity estimation uncertainty $\longrightarrow$ target-tracking based
- On-line information and decision updating

Target-tracking based prediction

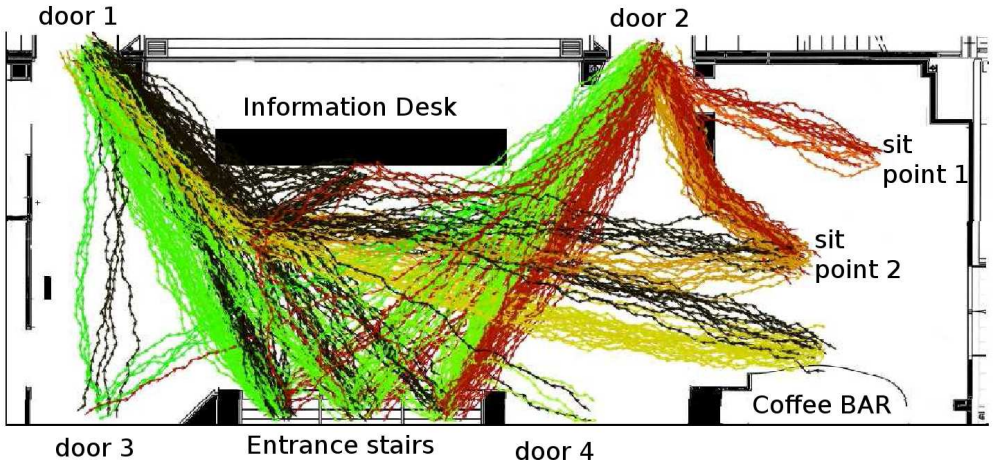
PROs	CONs
low *a priori* information	short-term
reactive to behavior changes	linear

Conclusions



Outline

Typical Patterns



Given an observed environment

- Moving objects follow typical patterns
- Patterns are learned off-line and modeled with HMMs or GPs
- Based on observations, probabilistic prediction is performed

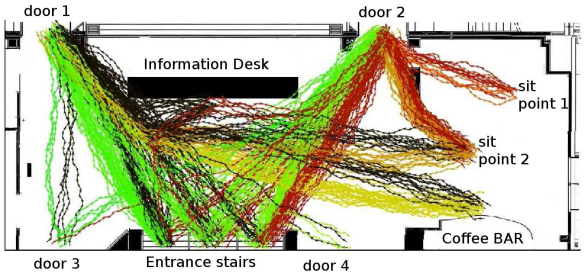
Prediction

- Takes into account the structure of the environment
- Uncertainty is limited around typical patterns

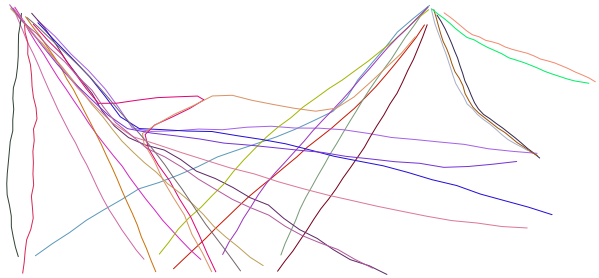
Gaussian Processes for Pattern Modelling [Tay, 2007]

A Typical Path $\rightarrow$ D dimensional Gaussian

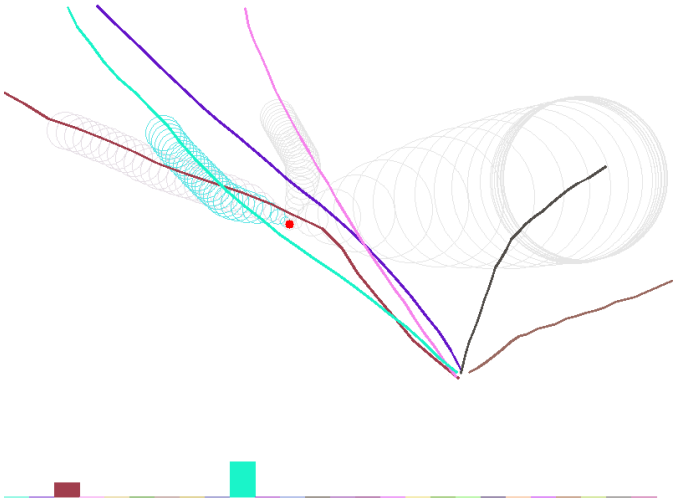
D $=$ # of points observed along a path



Dataset



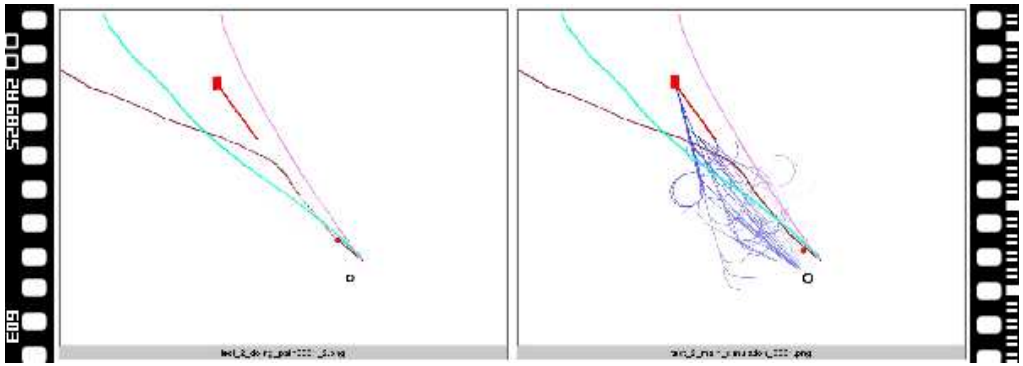
Learned GP means



$$P(O_m) = \sum_{k=1}^{K} l_{k,O_m} P(O_m|k)$$

Gaussian mixture

PPRRT with Gaussian Processes



$P_{cd}(q, O_m, k)$ collision with obstacle O_m in pattern k

$$P_{cd}(q, O_m) = \sum_{k=1}^{K} l_{k,O_m} \cdot P_{\pi}(q, O_m, k)$$

Tree update

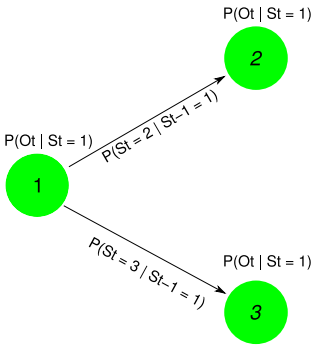
The weight of each Gaussian component l_{k,O_m} is updated.

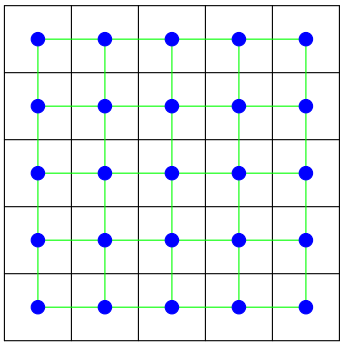
Hidden Markov Models for Pattern Modelling [Vasquez, 07]

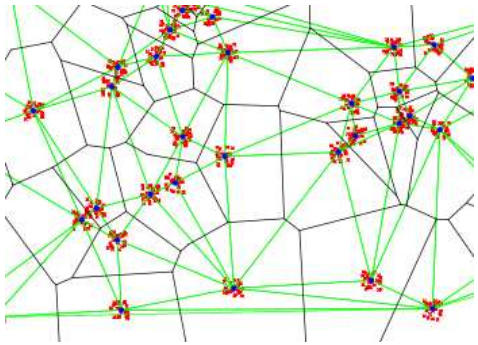
HMM: Bayesian filter for discrete state-space approach

A Typical Path $\rightarrow$ a directed graph

nodes $\rightarrow$ discrete states edges $\rightarrow$ transition probabilities







A HMM-graph represents all typical motions toward 1 goal

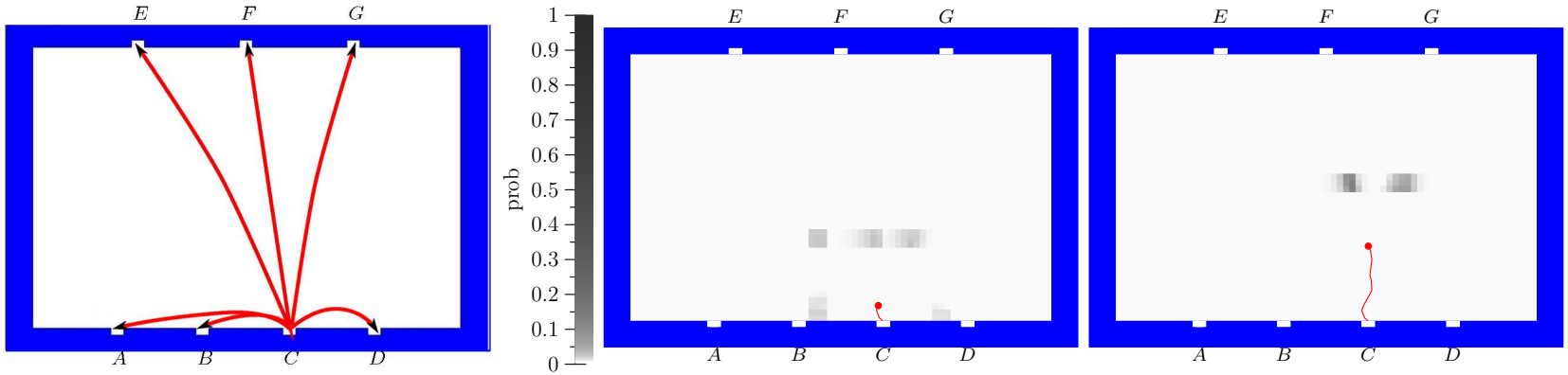
- States are given by position, velocity and intended goal
- Discretization is uniform or learned on the dataset
- Transition probabilities are learned

Hidden Markov Models based prediction

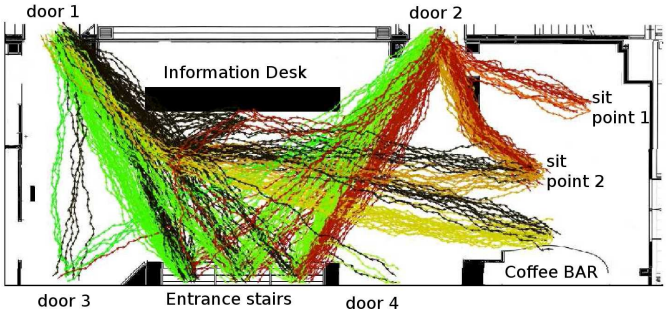
Discrete prediction

obtained letting the graph evolve under the transition probabilities

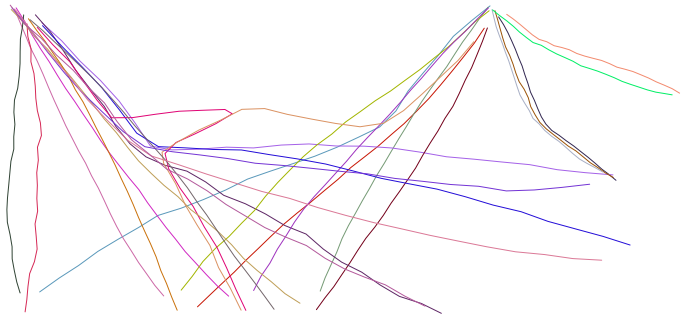
$$P(S_{t+k}|O_t) = \sum_{S_{t+k-1}} P(S_{t+k}|S_{t+k-1})P(S_{t+k-1}|O_t)$$



Simulation Results based on GPs



Dataset Simulated env



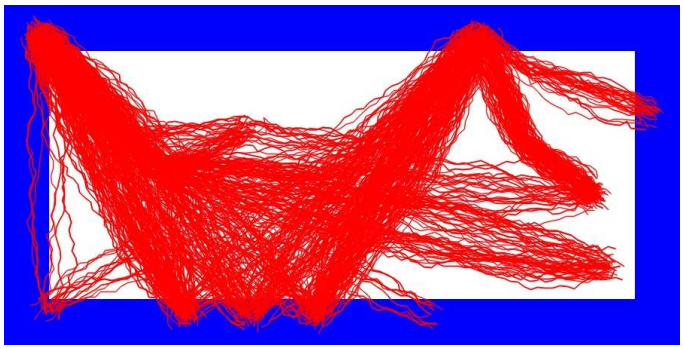
Learned GP means [Tay, 07]

Results for 100 goals reached; average obtained on 10 iterations.

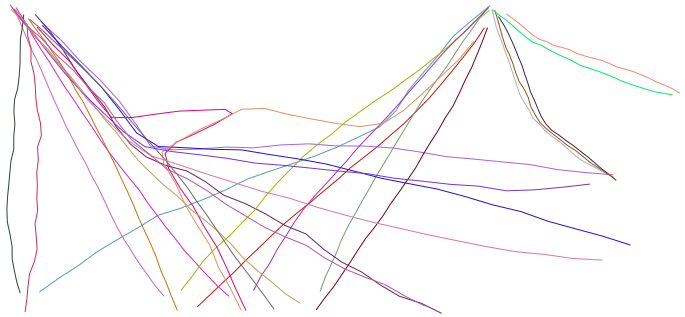
# obsts	# colls	with $v_r \neq 0$	% time
0	0	0	1
4	1.5	0	1.14
6	2.3	0	1.43
8	3.8	0	1.65
10	6.3	0	1.79
12	7.4	0	1.87

[click me]

Simulation Results based on GPs



Dataset Simulated env



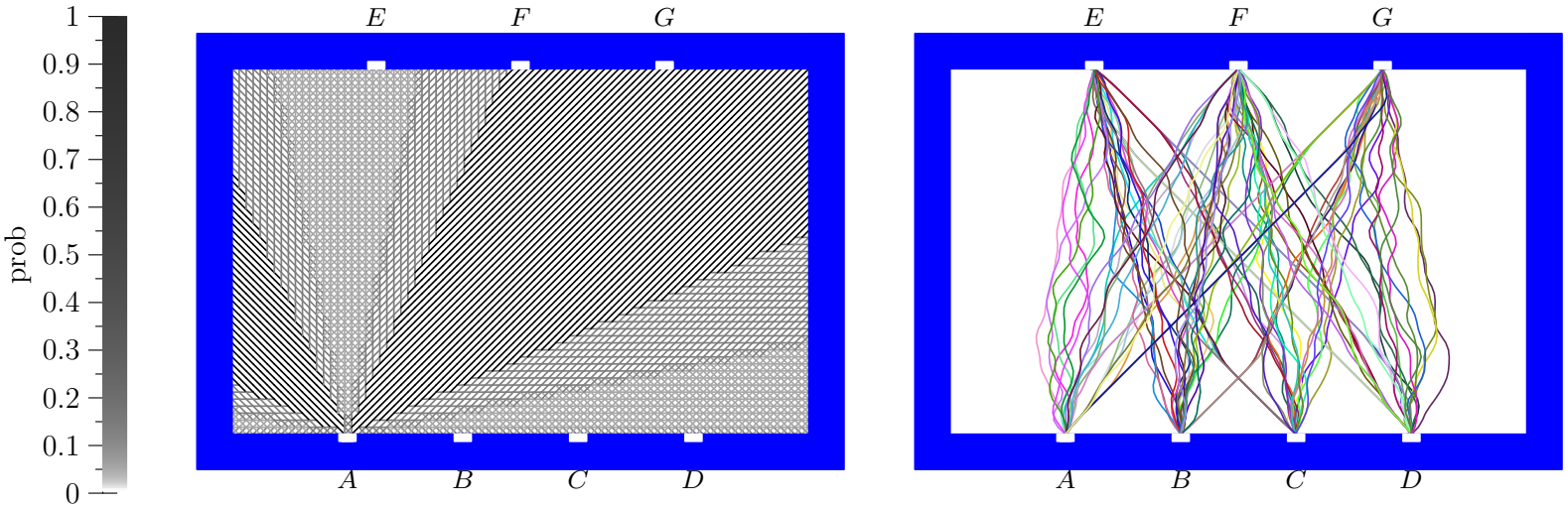
Learned GP means [Tay, 07]

Results for 100 goals reached; average obtained on 10 iterations.

# obsts	# colls	with $v_r \neq 0$	% time
0	0	0	1
4	1.5	0	1.14
6	2.3	0	1.43
8	3.8	0	1.65
10	6.3	0	1.79
12	7.4	0	1.87

[click me]

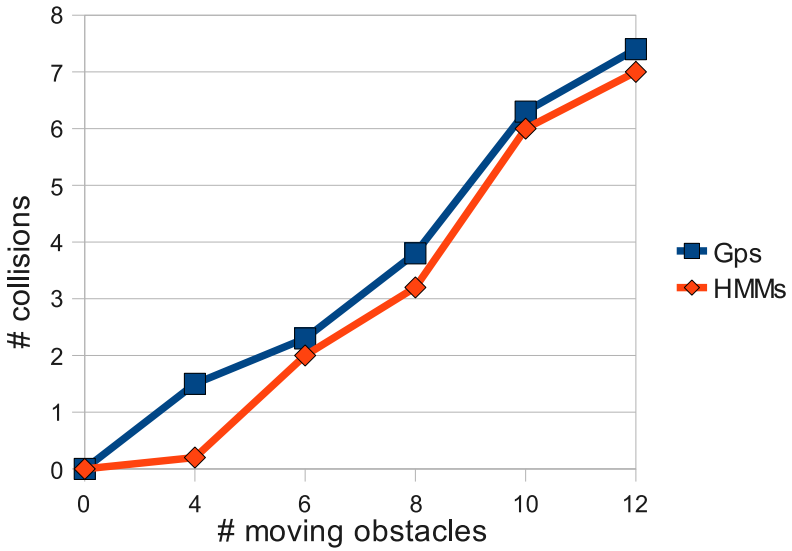
Simulation results based on HMMs

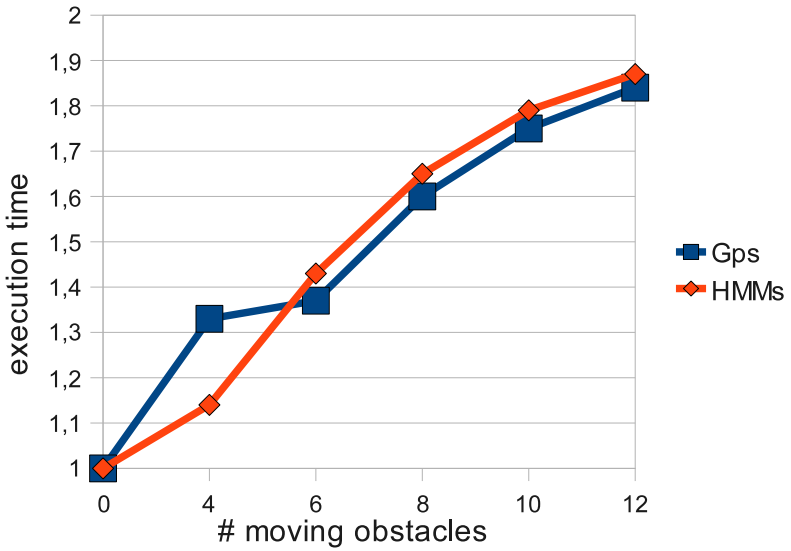


Results for 100 goals reached; average obtained on 10 iterations.

# obst	# colls	with $v_r \neq 0$	% time
0	0	0	1
4	0.2	0	1.33
6	2	0	1.37
8	3.2	0	1.60
10	6	0	1.75
12	7	0	1.84

Comparison





	HMM	GP
Performance	+	+
Velocity representation	discrete	*continuous*
Prediction	discrete	continuous
Prediction Update	complex	simple

Conclusions

Contributions:

- Reactive Method
- Partial Probabilistic RRTs:
  - Target tracking based algorithm
  - Typical patterns based prediction

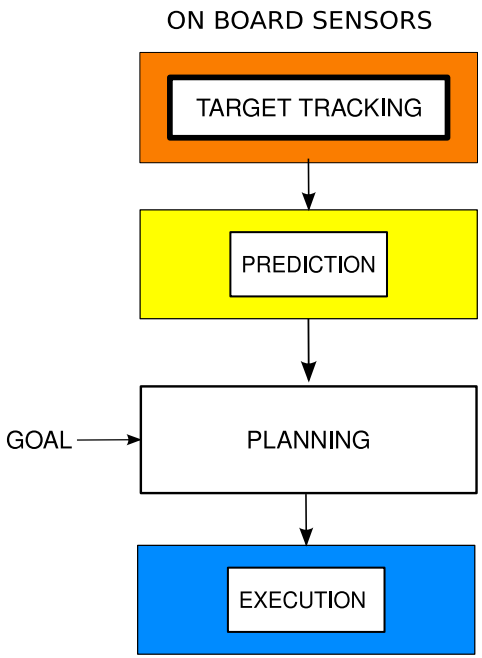
Properties:

- Probabilistic uncertainty of environment perception and prediction is meaningfully integrated into the navigation strategy
- Risk of collision is updated on-line with incoming estimation
- Known typical patterns
  - allow more reliable and non-linear predictions
  - more complex robot behaviors

Perspectives

Future work:

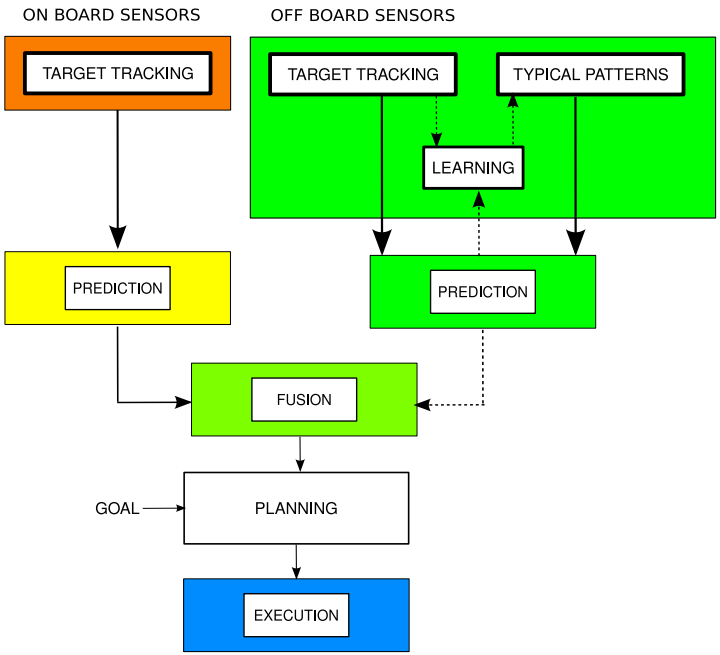
- From simulator to tests on the real robot
- Short and medium-term prediction used together in one framework (off-board platform)



Perspectives

Future work:

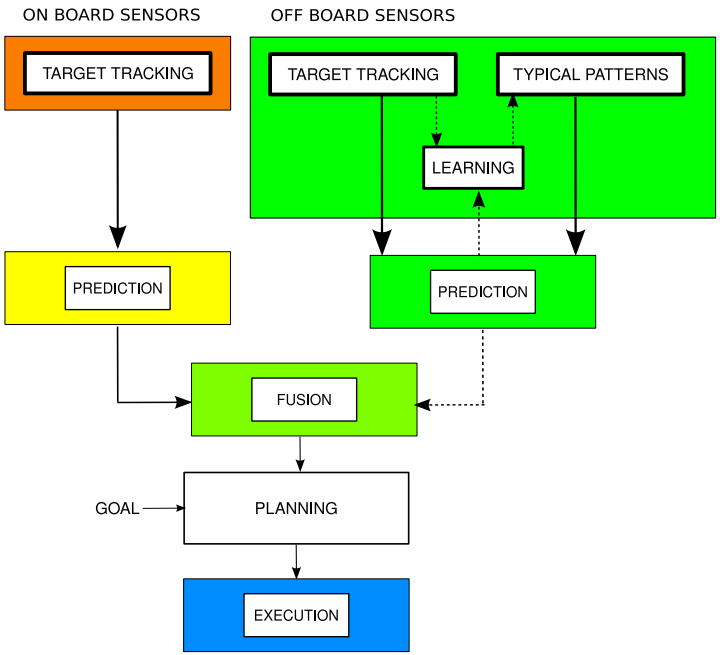
- From simulator to tests on the real robot
- Short and medium-term prediction used together in one framework (off-board platform)



Perspectives

Future work:

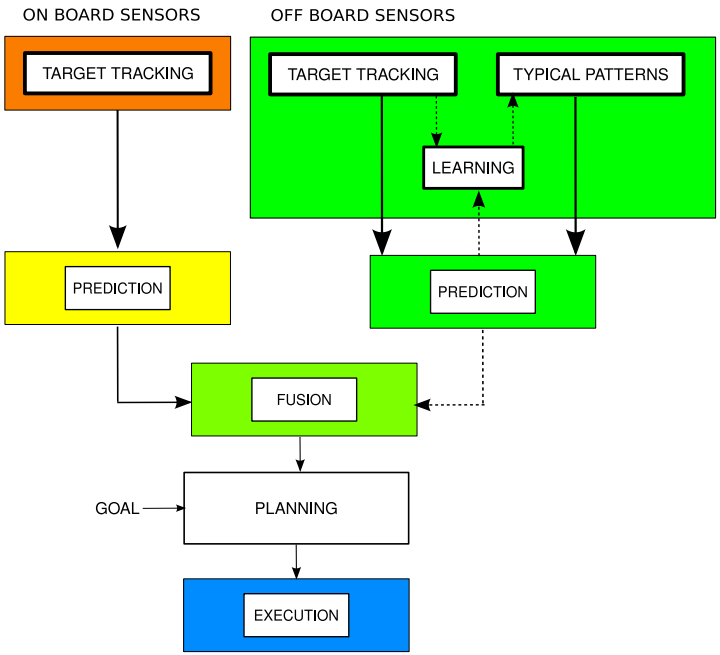
- From simulator to tests on the real robot
- Short and medium-term prediction used together in one framework (off-board platform)



Perspectives

Future work:

- From simulator to tests on the real robot
- Short and medium-term prediction used together in one framework (off-board platform)



Perspectives

Perspectives:

- Use the probabilistic framework to perform reflexive prediction
- Multiple robot coordination

Publications

- Fulgenzi, C., Spalanzani, A., Laugier, C. "*Dynamic Obstacle Avoidance in uncertain environment combining PVOs and Occupancy Grid.*", IEEE ICRA 2007
- Fulgenzi, C., Spalanzani, A., Laugier, C. "*Combining Probabilistic Velocity Obstacles and Occupancy Grid for safe Navigation in dynamic environments.*", Workshop on safe navigation in IEEE ICRA 2007
- Fulgenzi, C., Tay, C., Spalanzani, A., Laugier, C. "*Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian Processes.*", IEEE/RSJ IROS 2008
- Fulgenzi, C., Spalanzani, A., Laugier, C. "*Probabilistic Rapidly-exploring Random Trees for autonomous navigation among moving obstacles.*", Workshop on safe navigation in IEEE ICRA 2009