Autonomous and Mobile Robotics

Final Project

Academic Year 2021/2022

Comparing classic and primitive-based versions of kinodynamic RRT*

SAPIENZA
UNIVERSITÀ DI ROMA

Professor: *Giuseppe Oriolo*
Supervisor: *Paolo Ferrari*

Students: *Lorenzo Mattia* 1793272
Michela Proietti 1739446
Sofia Santilli 1813509

Additional requirements

Differentially constrained systems

Optimal Kynodynamic Motion Planning Problem

Guaranteed optimality of resulting trajectories

Guaranteed feasibility of resulting trajectories

ision-free path veen an initial and a final nfiguration

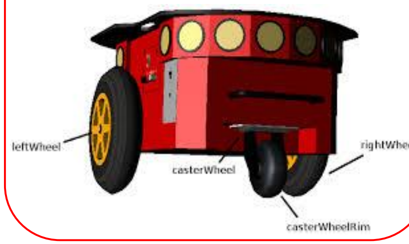
Problem statement

Unicycle

Configuration: $\boldsymbol{q} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$

Kinematic model: $\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta \\ sin\theta \\ \theta \end{bmatrix} \text{v} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w$

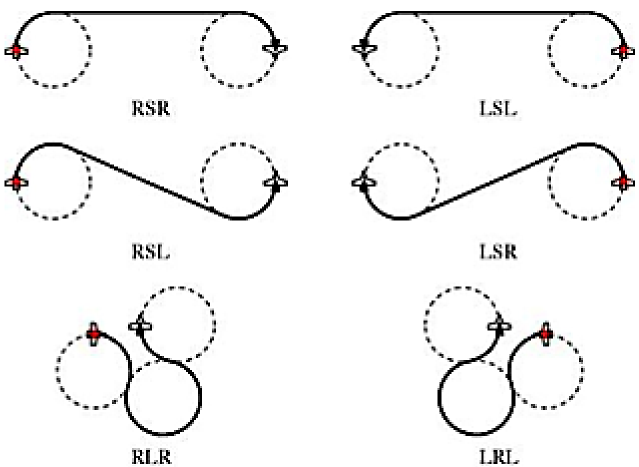
Pioneer p3dx



Dubins' vehicle

System's dynamics:

$$\dot{x} = v\ \cos\theta$$
$$\dot{y} = v\ \sin\theta$$
$$\dot{\theta} = u, |u| = \frac{v}{\rho}$$



Motion planning problem: find the optimal path, feasible for the system dynamics, connecting two configurations q_{ini} and q_{goal}

Randomly exploring Random Trees

Comparison between two implementations

RRT*

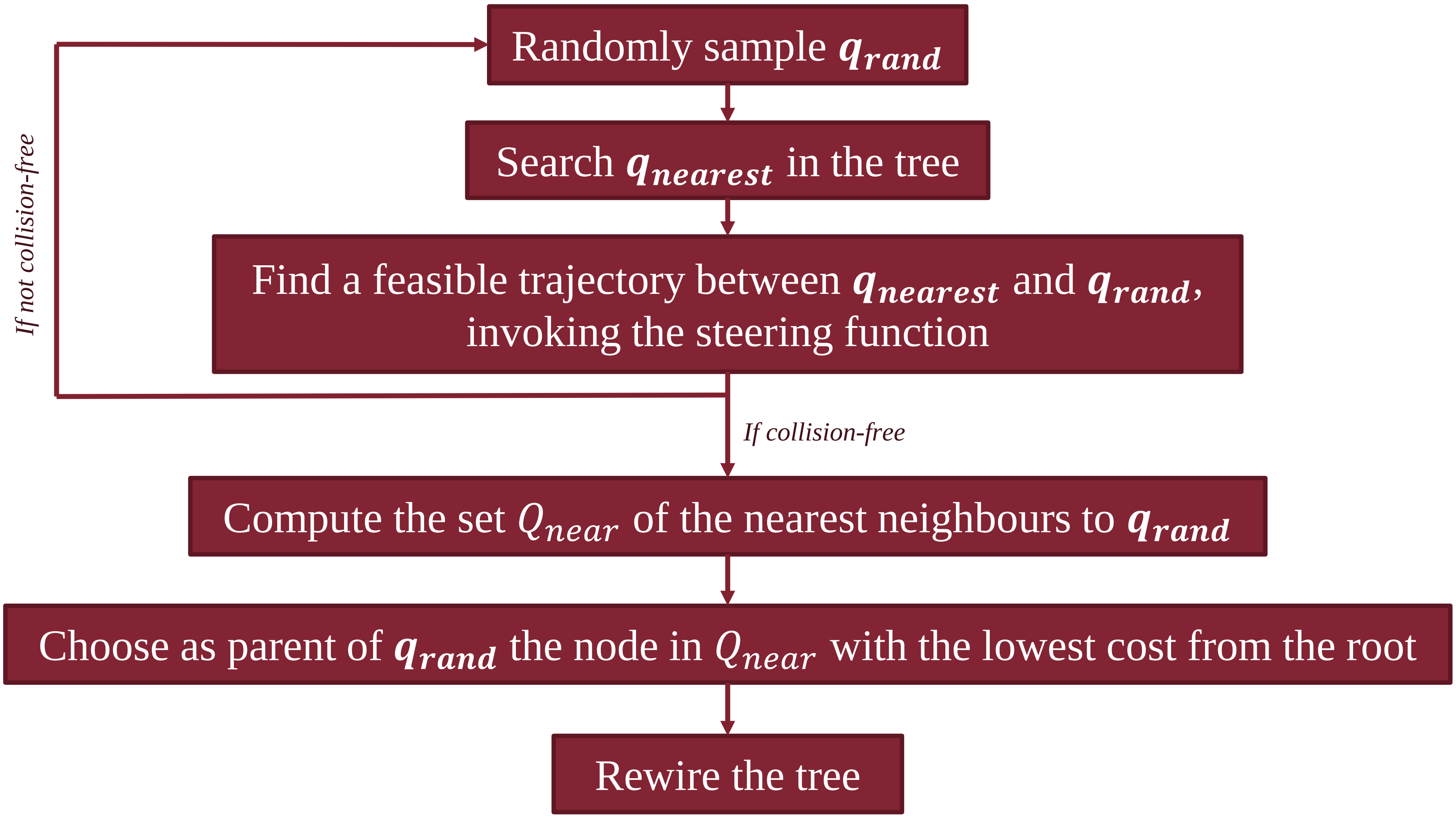
- Continuous configuration space
- Multiple invocations of the steering function

Primitive-based RRT*

- Discretized configuration space
- Precomputed catalogue of primitives

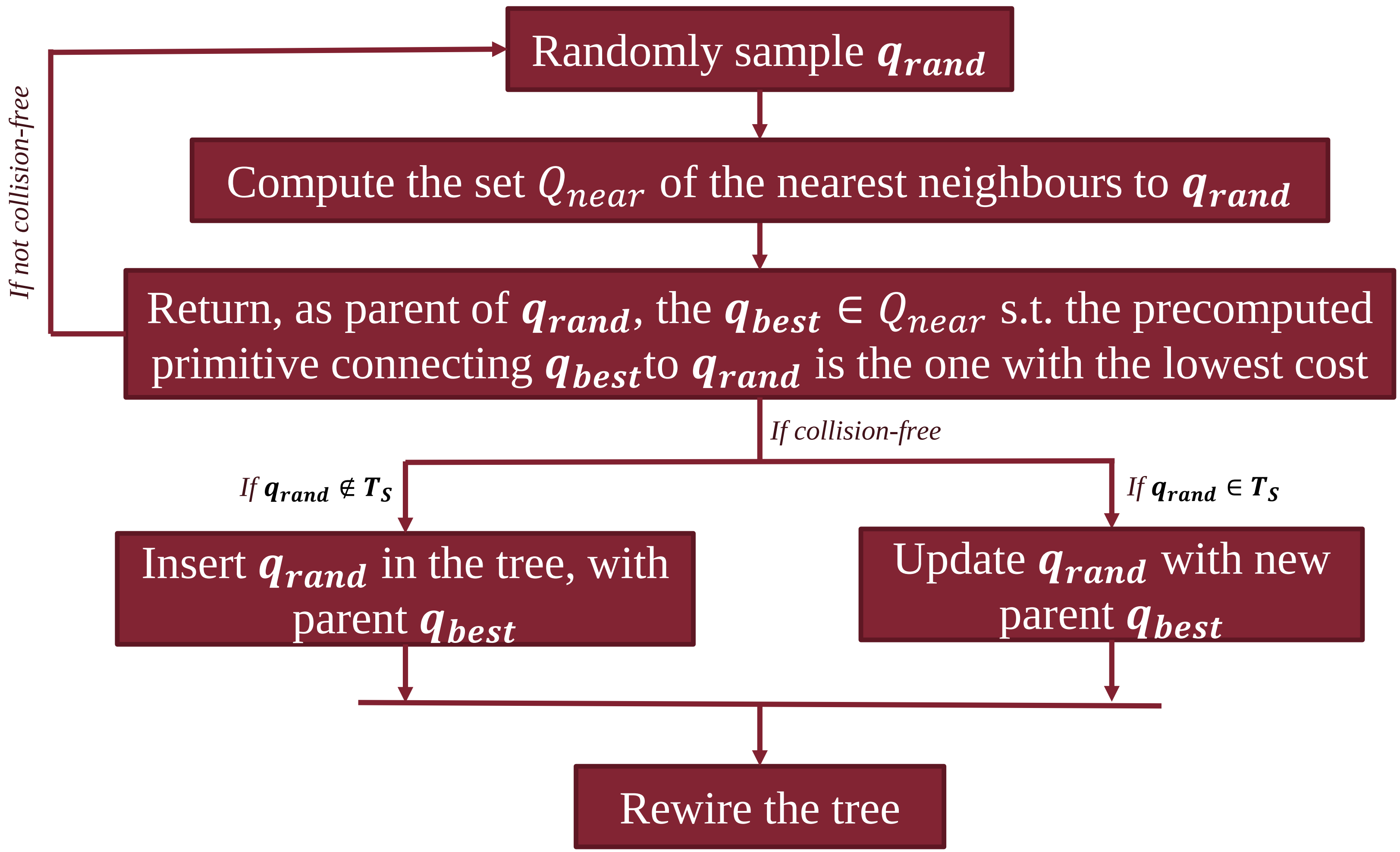
Classic RRT*

Basic iteration to build the tree T_s rooted at $\boldsymbol{q_s}$



Primitive-based RRT*

Basic iteration to build the tree T_s rooted at $\boldsymbol{q_s}$



Implementation: working environment

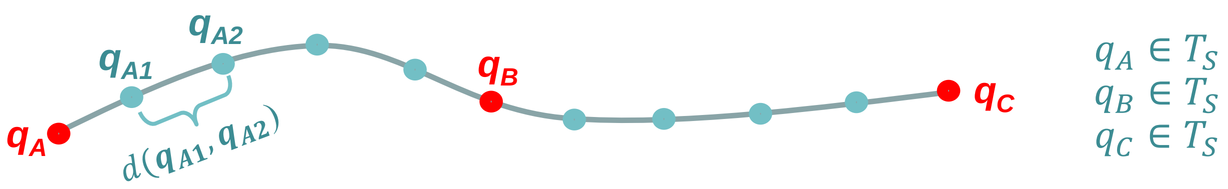
Ubuntu 18.04

CoppeliaSim 4.0 EDU

C++

Implementation: basic functions

- **Distance function**: $d(\mathbf{q_1}, \mathbf{q_2}) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + w|\theta_1 - \theta_2|$
- **Cost function :**

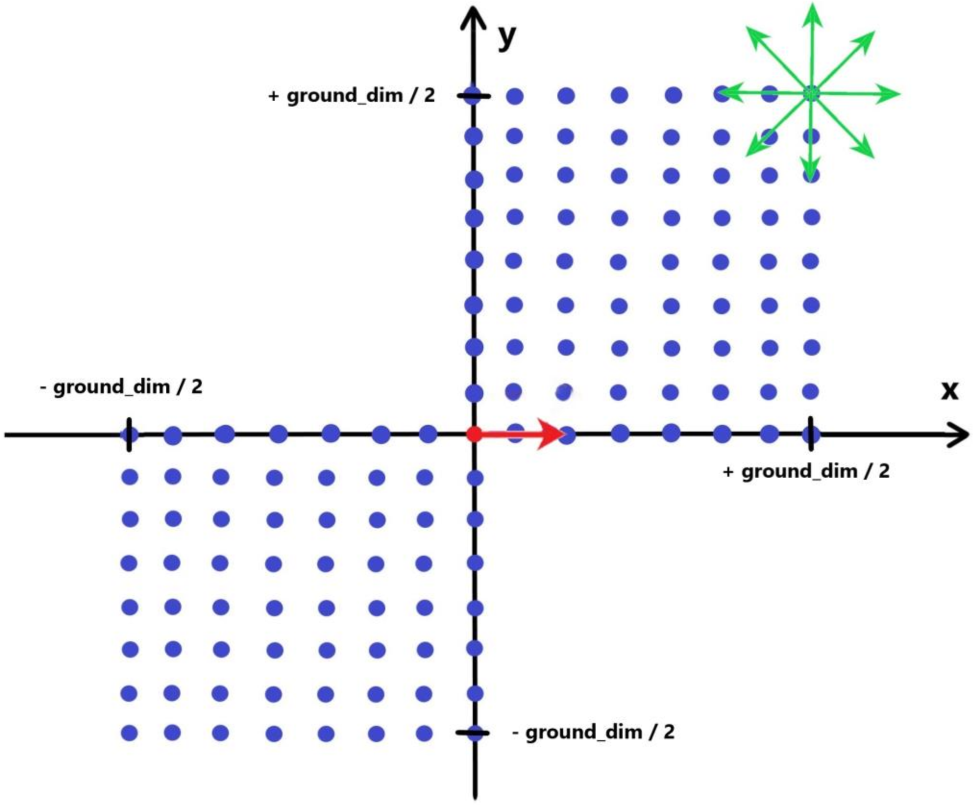


- **Q_{near} computation**: k nearest nodes to $\mathbf{q_{rand}}$
- **Exploration-exploitation strategy**: goal configuration sampled with 0.2 of probability
- **Goal updates**
- **Termination condition**: maximum number of iterations is reached

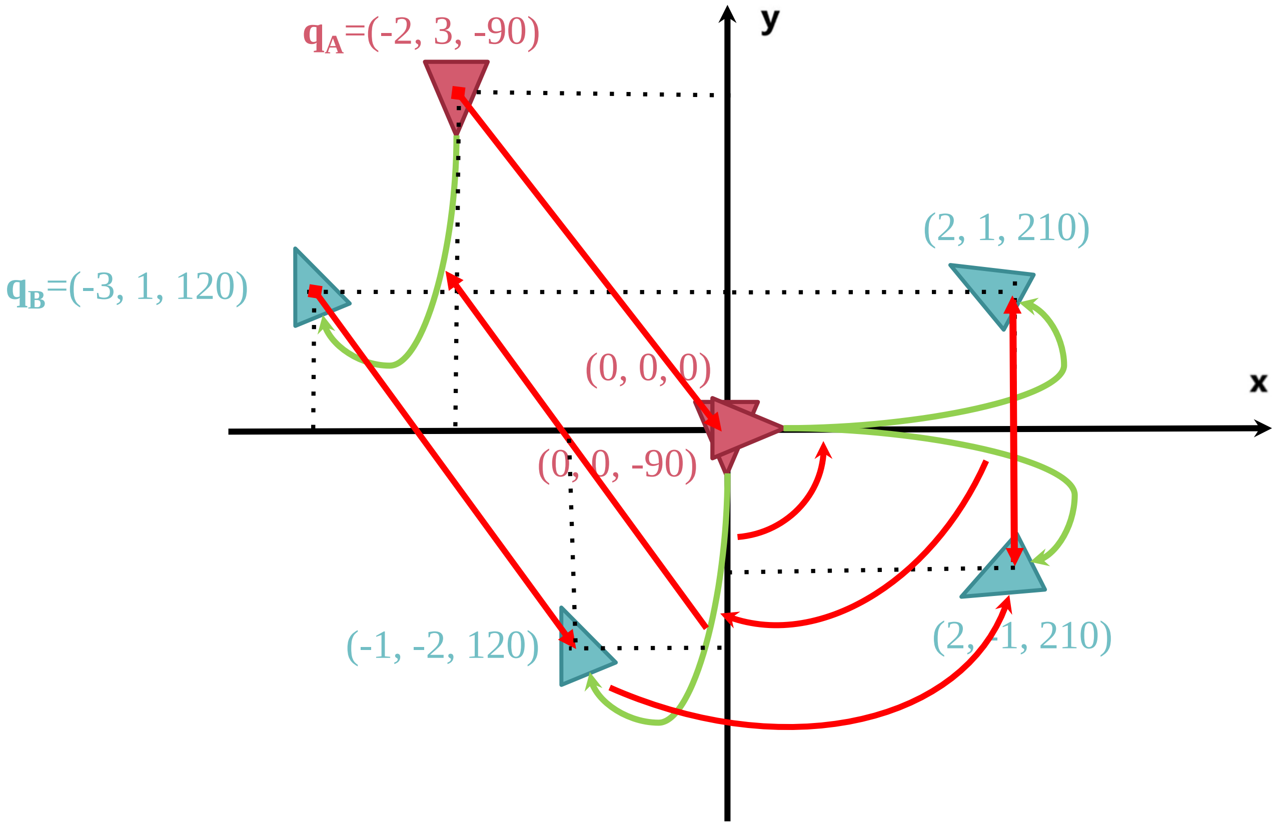
Implementation: catalogue of primitives

- ***Precomputation of the catalogue of primitives***:

  saved in a text file, with rows in the format (final node, cost, path)

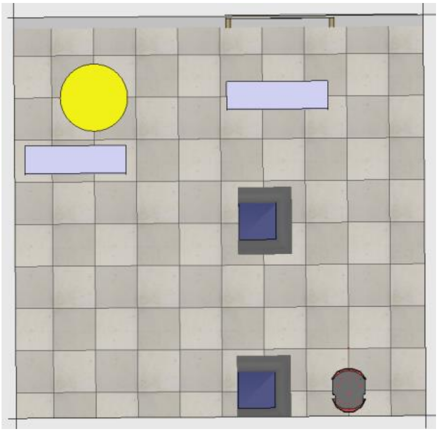


Implementation: primitive's selection



Planning experiments: scenarios

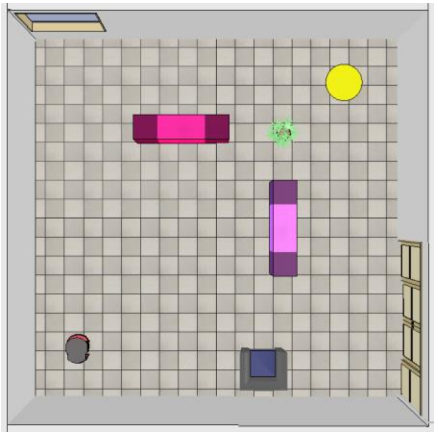
- Four simulation environments of increasing complexity
- Three different combinations of grid resolutions and possible orientations, for the primitive based version



Scene 1

Dimension: 5x5 [m]

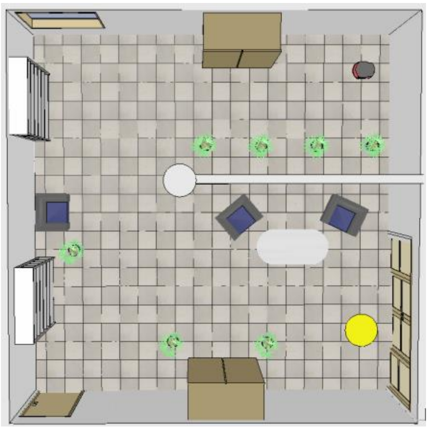
Grid res.	#orientations
1	4
1	8
0.5	8



Scene 2

Dimension: 10x10 [m]

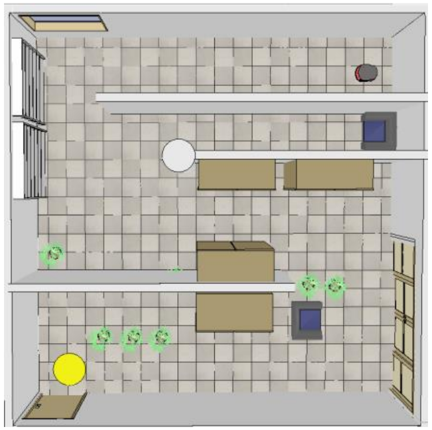
Grid res.	#orientations
4	4
1	8
0.5	8



Scene 3

Dimension: 10x10 [m]

Grid res.	#orientations
4	4
1	8
0.5	8



Scene 4

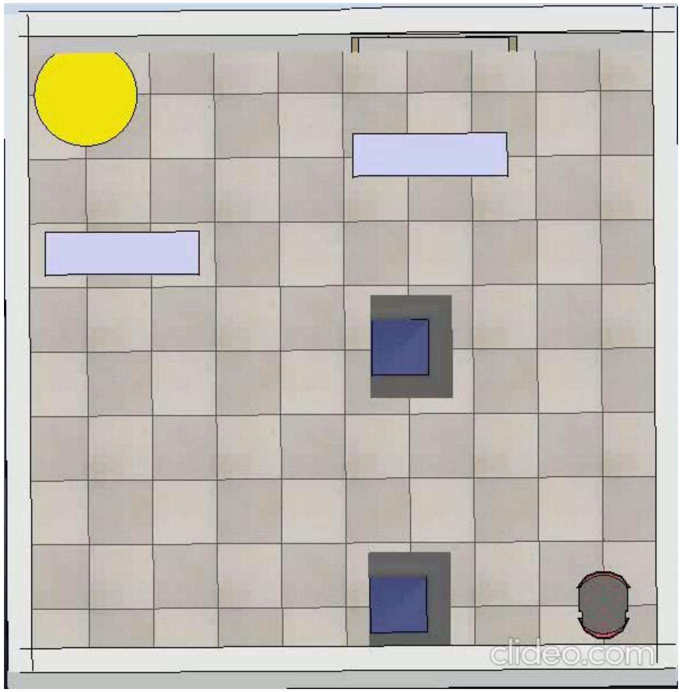
Dimension: 10x10 [m]

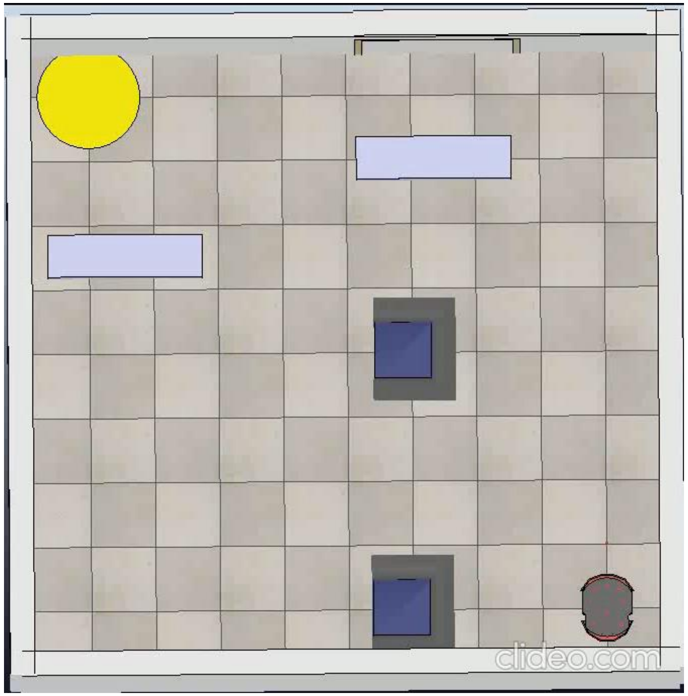
Grid res.	#orientations
4	4
1	8
0.5	8

Scene 1

SCENE 1	n.primitives	n.iters	tot.time(s)	steering time(s)	tree size	sol.cost	success rate(%)
Classic RRT*	—	100	8.5	7.5	53	11.1	100
		1.000	102.3	82.7	465.3	8.49	100
		10.000	860	587.9	5562.6	8.43	100
Primitive-based RRT*	63 (64)	100	3.9	< 1	22.5	8.61	80
		1.000	40.6	< 1	44.4	8.35	100
		10.000	179.4	< 1	46	8.16	100
	126 (128)	100	1.9	< 1	21.6	14.83	20
		1.000	24.5	< 1	73.1	12.59	80
		10.000	245.4	113.7	81.4	12.01	100
	370 (384)	100	3.1	< 1	31.1	11.37	80
		1.000	41.9	< 1	198.2	11.21	100
		10.000	472.2	238.2	301.9	11.12	100

Classic RRT*



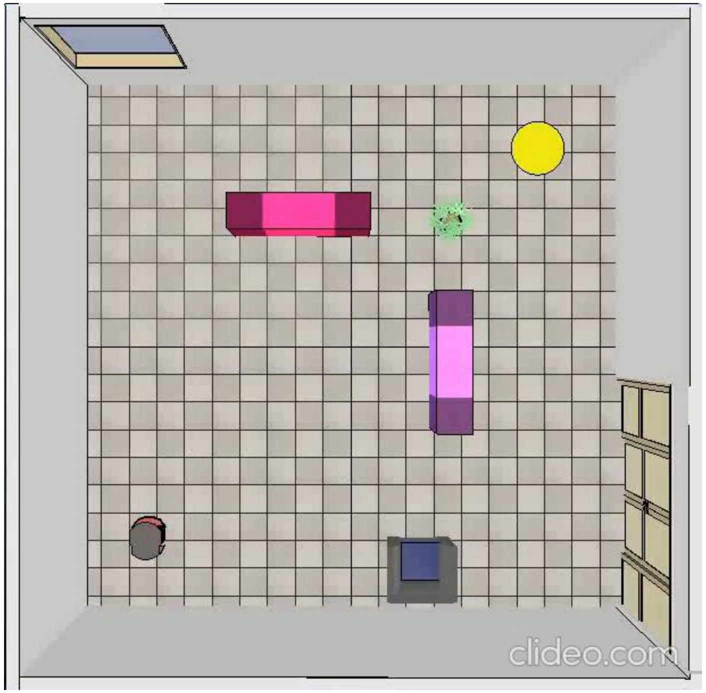


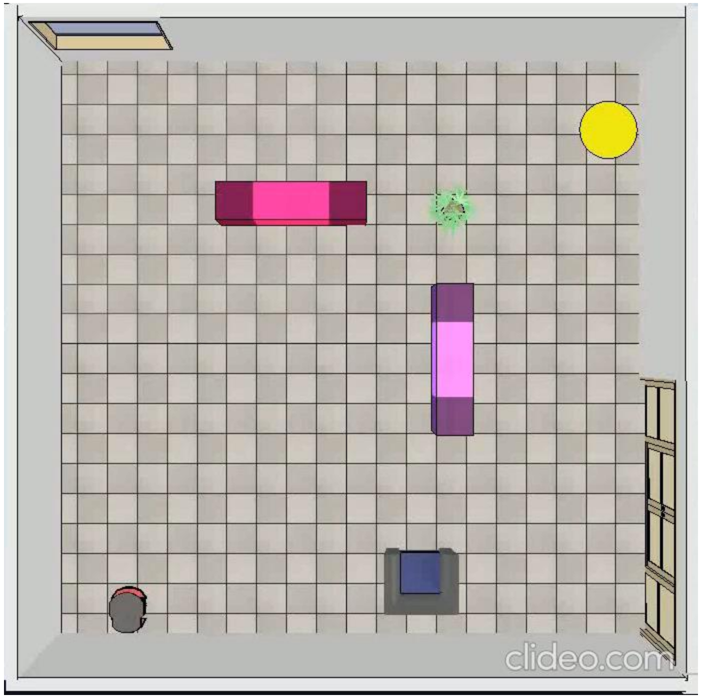
Primitive -based RRT*

Scene 2

SCENE 2	n.primitives	n.iters	tot.time(s)	steering time(s)	tree size	sol.cost	success rate(%)
Classic RRT*	—	100	31.9	25.9	60.2	14.31	90
		1.000	253.5	211.8	655.7	14.43	100
		10.000	2998.2	2187.6	5727.9	14.18	100
Primitive-based RRT*	14 (24)	100	< 1	< 1	1.5	//	0
		1.000	3.7	< 1	8.9	19.52	50
		10.000	172.8	< 1	42	16.56	100
	325 (560)	100	4.3	< 1	34	18.27	70
		1.000	48.4	35.5	275.7	18.25	100
		10.000	518.1	270.5	495.2	17.79	100
	1220 (1920)	100	8.9	7	40.1	21.72	80
		1.000	104.7	70	429.8	20.06	100
		10.000	1045.5	620	1703.1	18.11	100

Classic RRT*



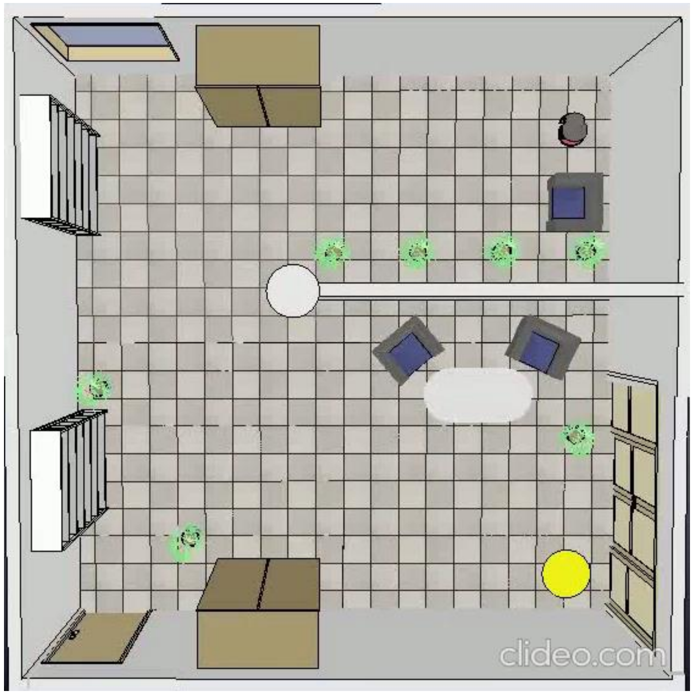


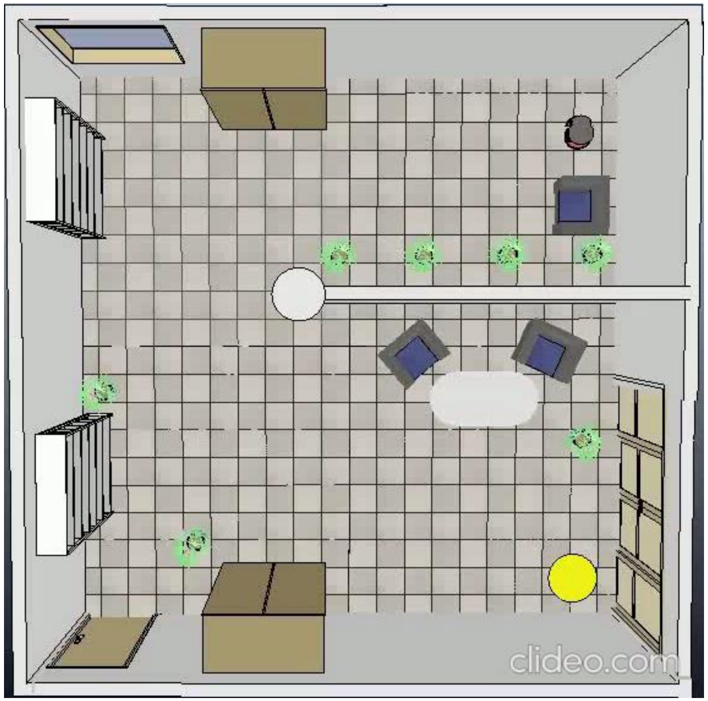
Primitive-based RRT*

Scene 3

SCENE 3	n.primitives	n.iters	tot.time(s)	steering time(s)	tree size	sol.cost	success rate(%)
Classic RRT*	—	100	14.6	8.4	51.8	24.26	100
		1.000	90.4	48.6	611.4	22.74	100
		10.000	874.2	419.4	6180.8	22.66	100
Primitive -based RRT*	14 (24)	100	< 1	< 1	1	//	0
		1.000	1	< 1	6.1	32.46	10
		10.000	99	< 1	43.1	30.3	100
	325 (560)	100	2.5	$<= 1$	21.1	26.78	50
		1.000	22.7	5.7	246.4	26.35	100
		10.000	162	50.5	433.1	25.28	100
	1220 (1920)	100	3.4	1.4	24.3	25.08	60
		1.000	37.8	18	383.5	24.67	100
		10.000	287.3	91.7	1485.9	23.95	100

Classic RRT*



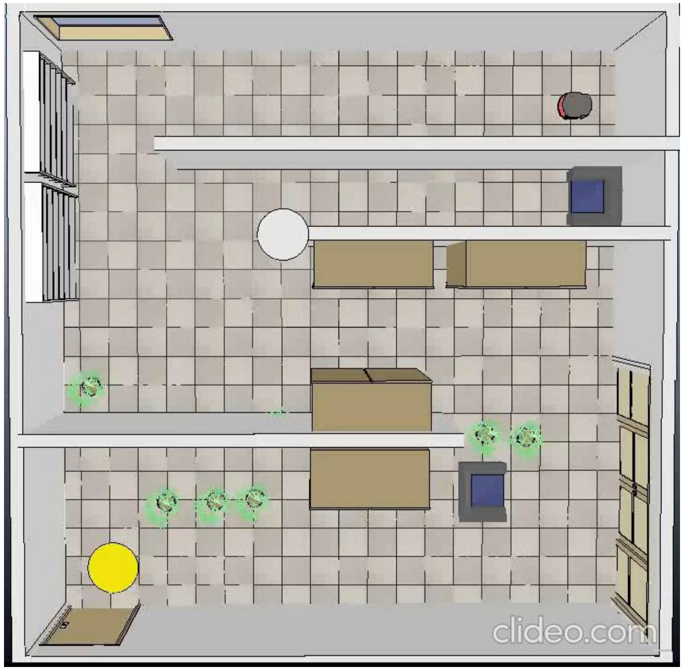


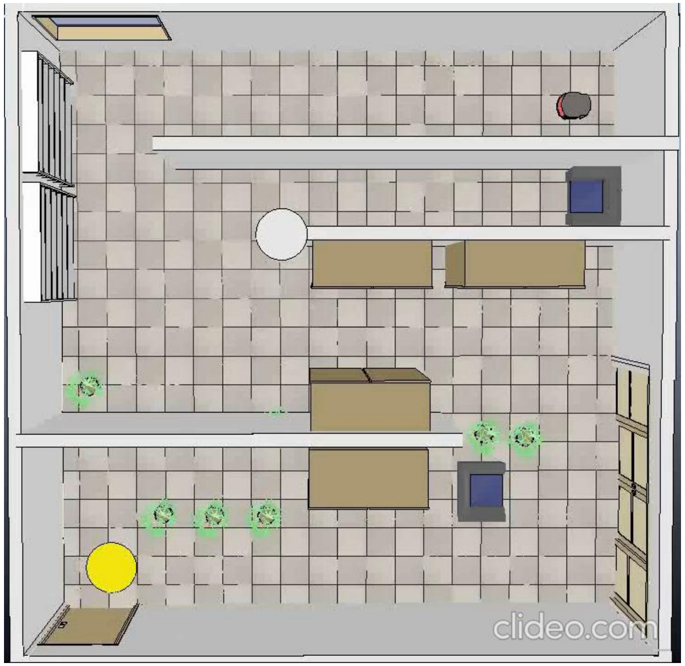
Primitive -based RRT*

Scene 4

SCENE 4	n.primitives	n.iters	tot.time(s)	steering time(s)	tree size	sol.cost	success rate(%)
Classic RRT*	—	100	38.4	29.9	18.6	//	0
		1.000	274.7	218.8	293.1	39.32	70
		10.000	3031.1	2232.2	3712.2	39.35	100
Primitive-based RRT*	14 (24)	100	< 1	< 1	1.4	//	0
		1.000	2.2	0.8	6.7	46.62	10
		10.000	151	7.1	49.5	39.66	100
	325 (560)	100	1	< 1	3.2	//	0
		1.000	44.3	13.25	164.7	48.7	90
		10.000	393.2	131.7	368.5	46.12	100
	1220 (1920)	100	2.7	1.15	7.2	//	0
		1.000	60.4	31.2	211.4	46.8	80
		10.000	978.8	485.1	1233.6	45.86	100

Classic RRT*



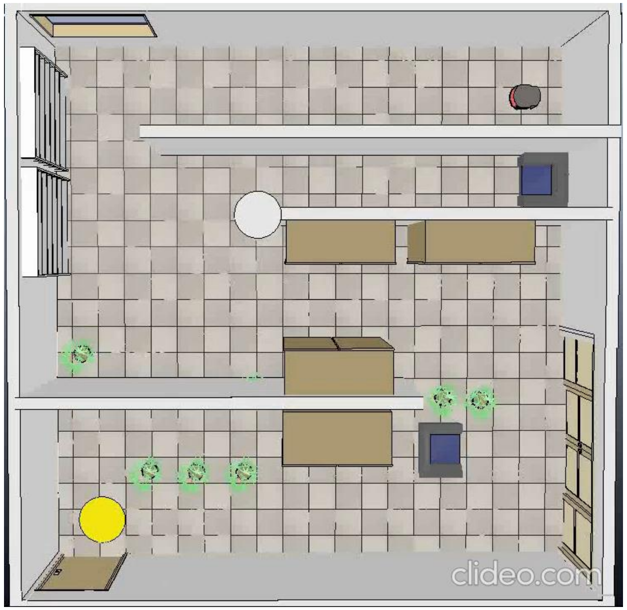


Primitive -based RRT*

Scene 4

Comparison varying the catalogue

Grid resolution: 4
4 orientations



Grid resolution: 1
8 orientations

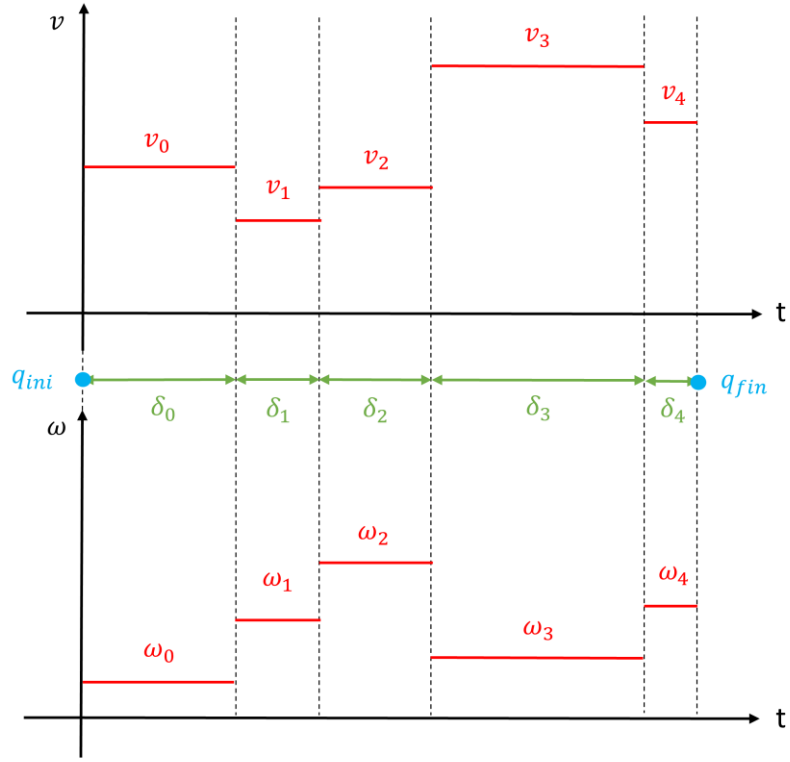
Grid resolution: 0.5
8 orientations

Extension to generic unicycle

The steering function requires to solve TPBVP: compute the best path between two nodes that satisfies a constrained optimality

$$\min_{X,Y.\Theta,V,\Delta,\Omega} 1 + 0.5\,V + 0.5\,\Omega + 0.5\,\Delta$$

$$\text{s.t.} \begin{cases} x_{k+1} = x_k + \delta_k v_k \cos(\theta_k) \\ y_{k+1} = y_k + \delta_k v_k \sin(\theta_k) \\ \theta_{k+1} = \theta_k + \delta_k w_k \\ x_0 = \overline{x_i}, \mathrm{y}_0 = \overline{y_i}, \theta_0 = \overline{\theta_i} \\ x_N = \overline{x_f}, \mathrm{y}_\mathrm{N} = \overline{y_f}, \theta_N = \overline{\theta_f} \\ v_{min} \le v_k \le v_{max} \\ w_{min} \le w_k \le w_{max} \end{cases}$$



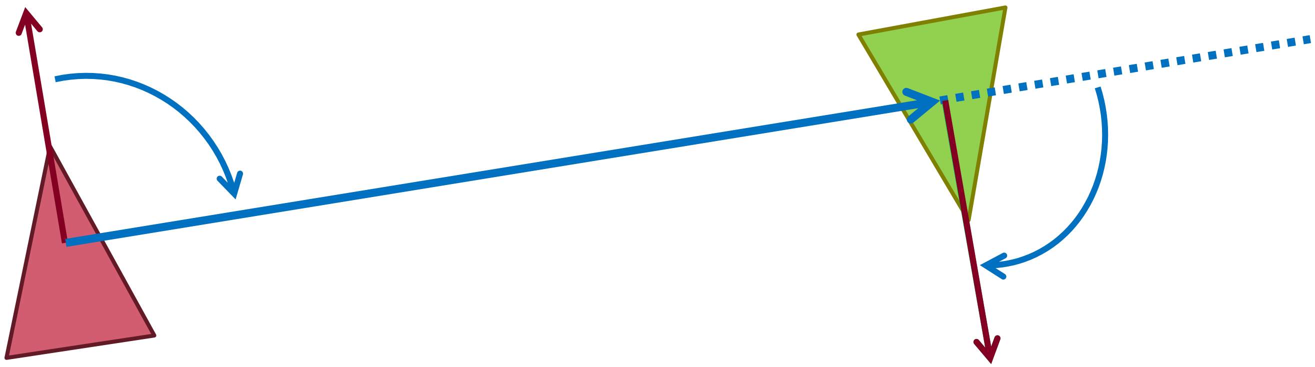
Executed in the steering function of the algorithm and solved through *Ifopt*, a light-weight, Eigen-based C++ interface to Nonlinear Programming solvers

TPBVP: trajectory reconstruction

- Exact integration

$$x_{k+1} = x_k + \frac{v_k}{w_k}(sin\theta_{k+1} - sin\theta_k)$$
$$y_{k+1} = y_k + \frac{v_k}{w_k}(cos\theta_{k+1} - cos\theta_k)$$
$$\theta_{k+1} = \theta_k + w_k T_s$$

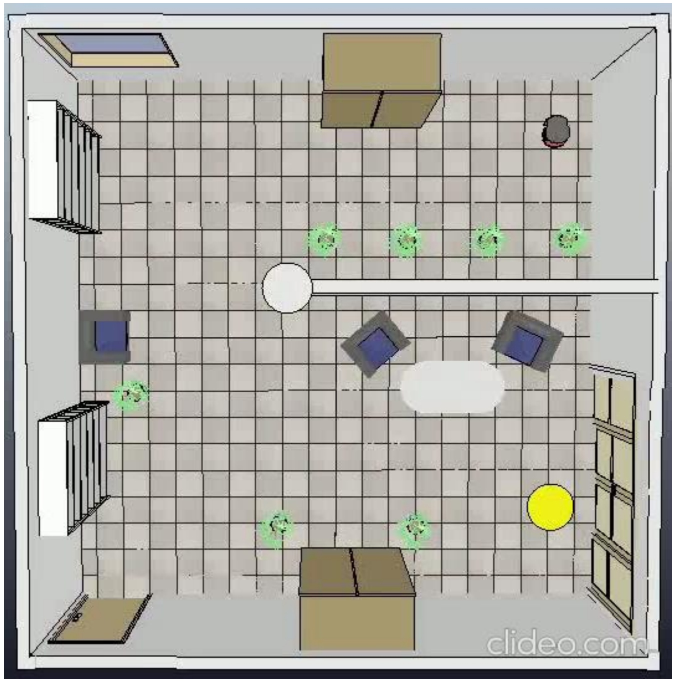
- Final maneuver

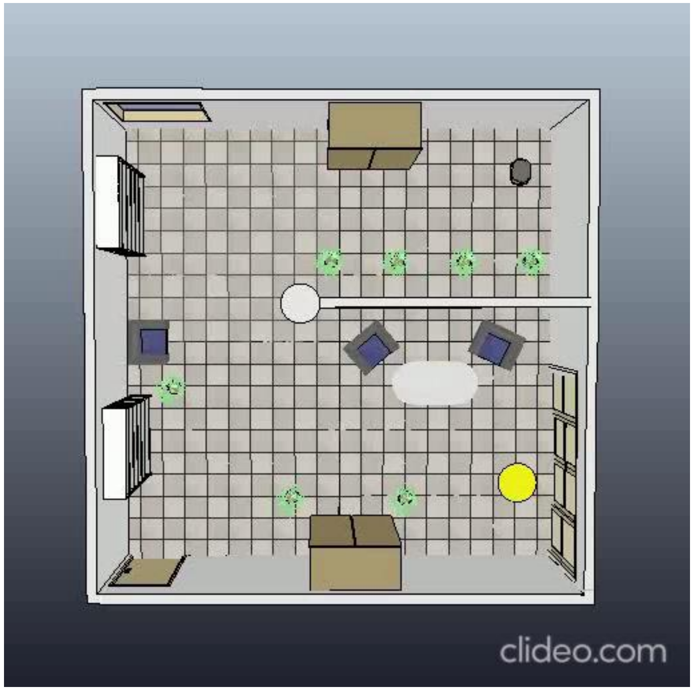


Planning experiments

SCENE 3	n.primitives	n.iters	tot.time(s)	steering time(s)	tree size	sol.cost	success rate(%)
Classic RRT*	—	100	79	36	37	34.2	50
		1.000	767	389	493	28	100
		10.000	8705	4238	6043	26.6	100
Primitive -based RRT*	325 (560)	100	36	< 1	41	34.5	60
		1.000	221	52.7	255	29.1	100
		10.000	1709	423.3	430	27.3	100

Classic RRT*





Primitive -based RRT*

Conclusions

Standard RRT*

- Continuous state space
- Slightly lower cost solutions
- Higher computational time
- Higher success rate

Primitive-based RRT*

- Discretized state space
- Slightly higher cost solutions
- Lower computational time
- Lower success rate
- Highly dependent on the number of primitives
- Might have problems with narrow passages

Choice depends on user’s needs and computational resources

Thanks for your attention