3. Grid-based path planning

A* path planner

Youngsun Kwon

2020. 07. 09.

KAIST SGVR Lab.

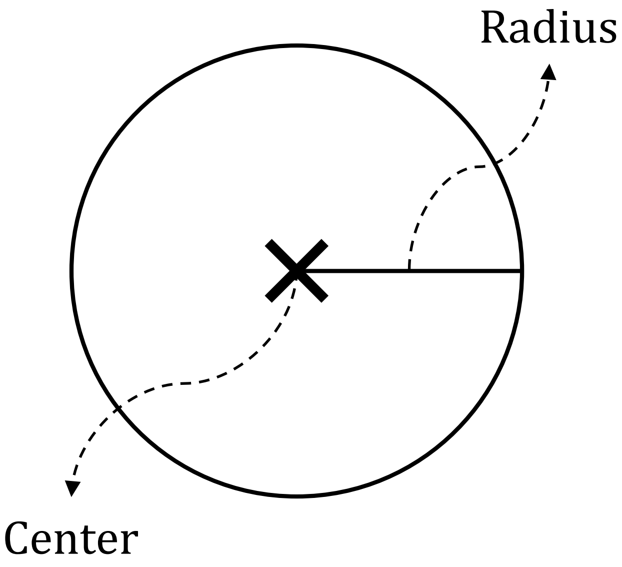
Preparation of tutorial

- **Checking your system for this tutorial**

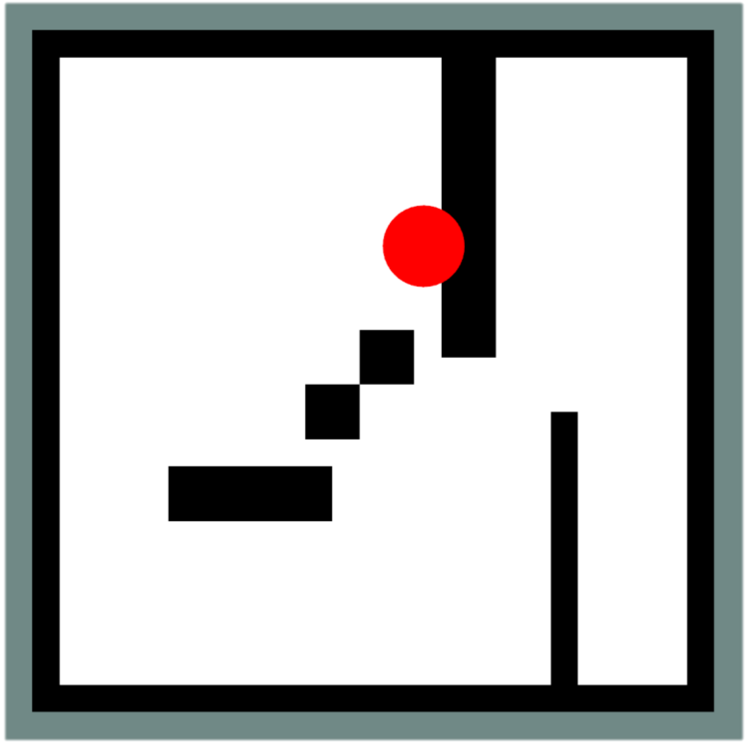
ROS Melodic	https://www.ros.org/	
SuperRay library	https://github.com/PinocchioYS/SuperRay	Mapping and collision detection in 2-D
Clion	https://www.jetbrains.com/ko-kr/clion/	IDE for C++
Tutorial sources	https://github.com/PinocchioYS/path_planning_tutorial	
CoppeliaSim(V-REP)	https://www.coppeliarobotics.com/	Simulation

Tutorial 0: collision detection

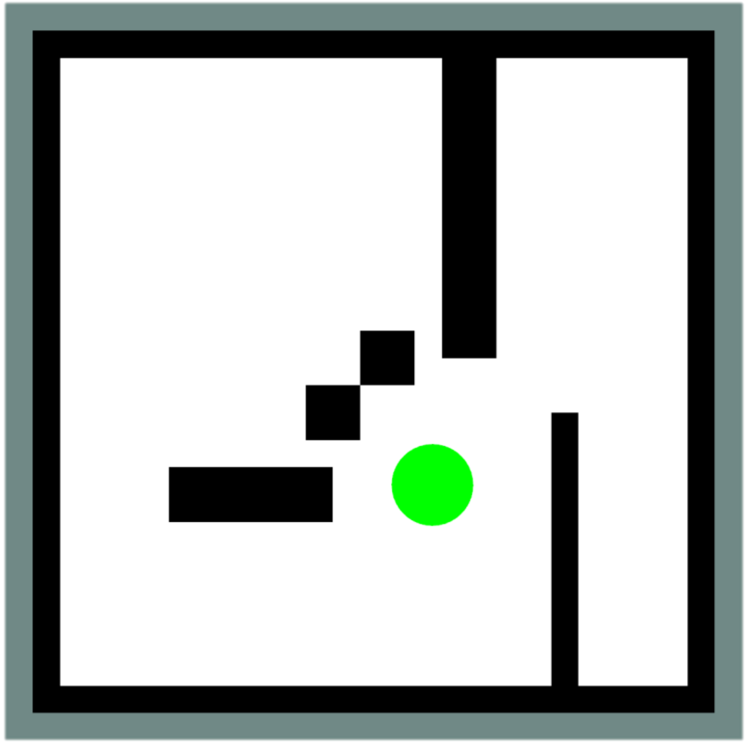
- **Bounding volume: circle**
  - Use the “2D Pose Estimate” tool of RViz to make a circle and test a collision.
  - Turn off the option, “#define USE_OBB_MODEL”, to use the circle shape.



Default value of radius: 0.3m



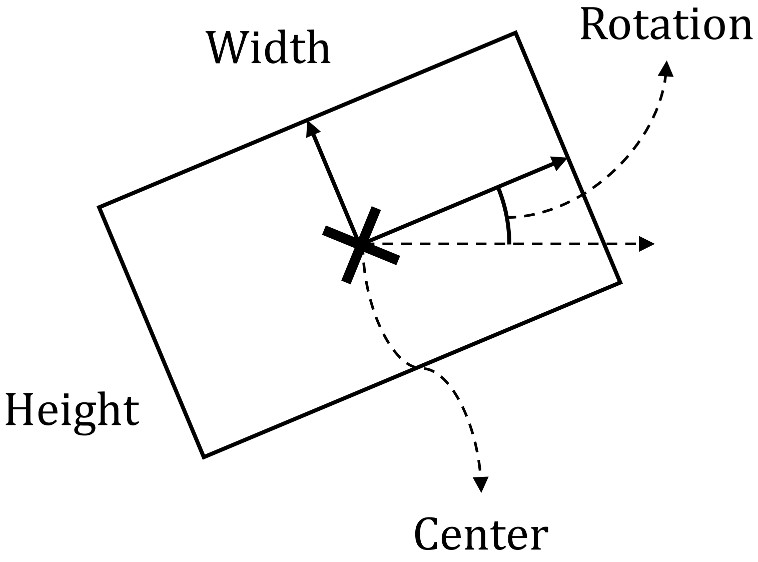
Collision (red color)



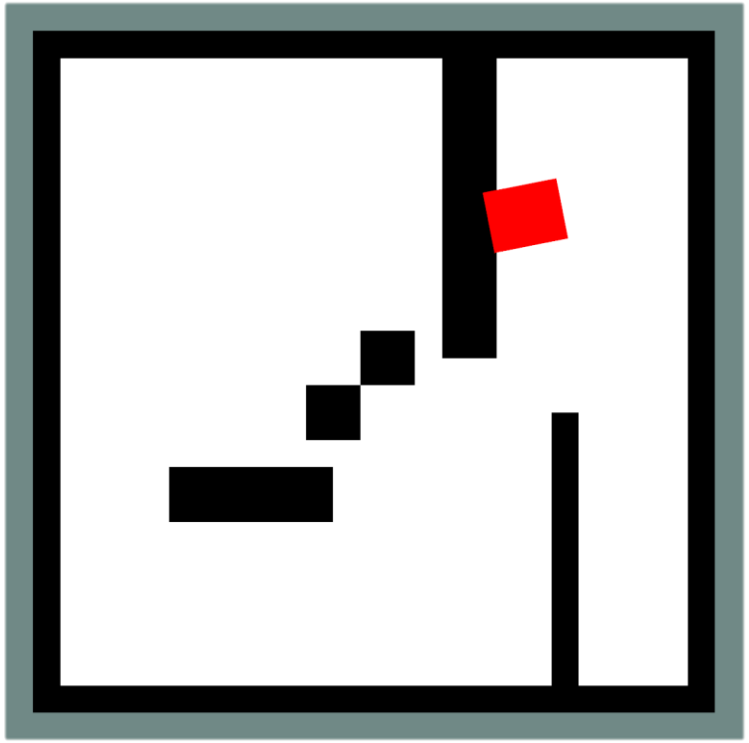
No collision (green color)

Tutorial 0: collision detection

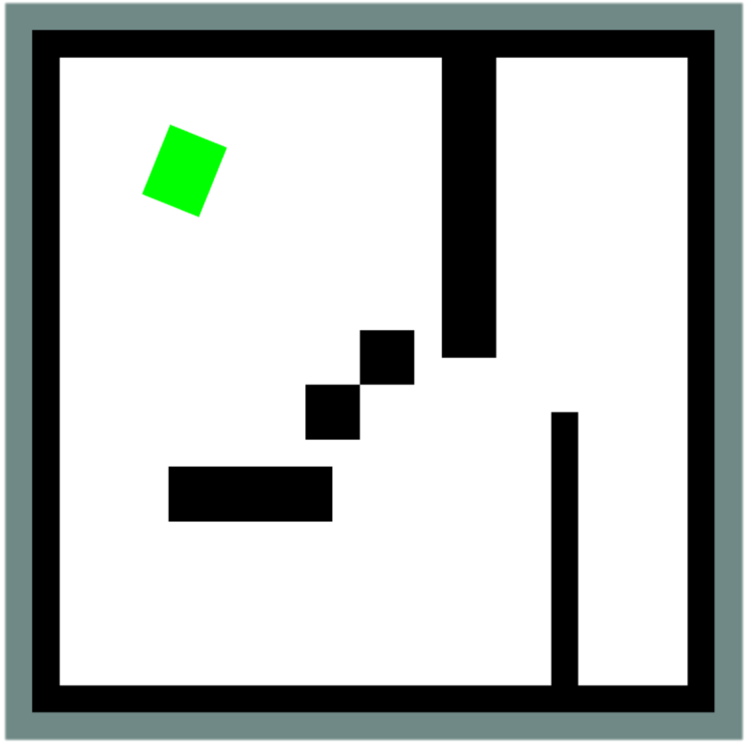
- **Bounding volume: oriented bounding box (OBB)**
  - Use the "2D Pose Estimate" tool of RViz to make an OBB and test a collision.
  - Turn on the option, "#define USE_OBB_MODEL", to use the OBB shape.



Default values of width and height: 0.55m and 0.45m



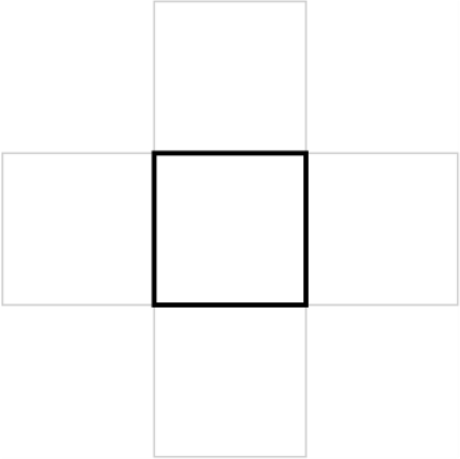
Collision (red color)



No collision (green color)

Tutorial 1: A* planner using a circle

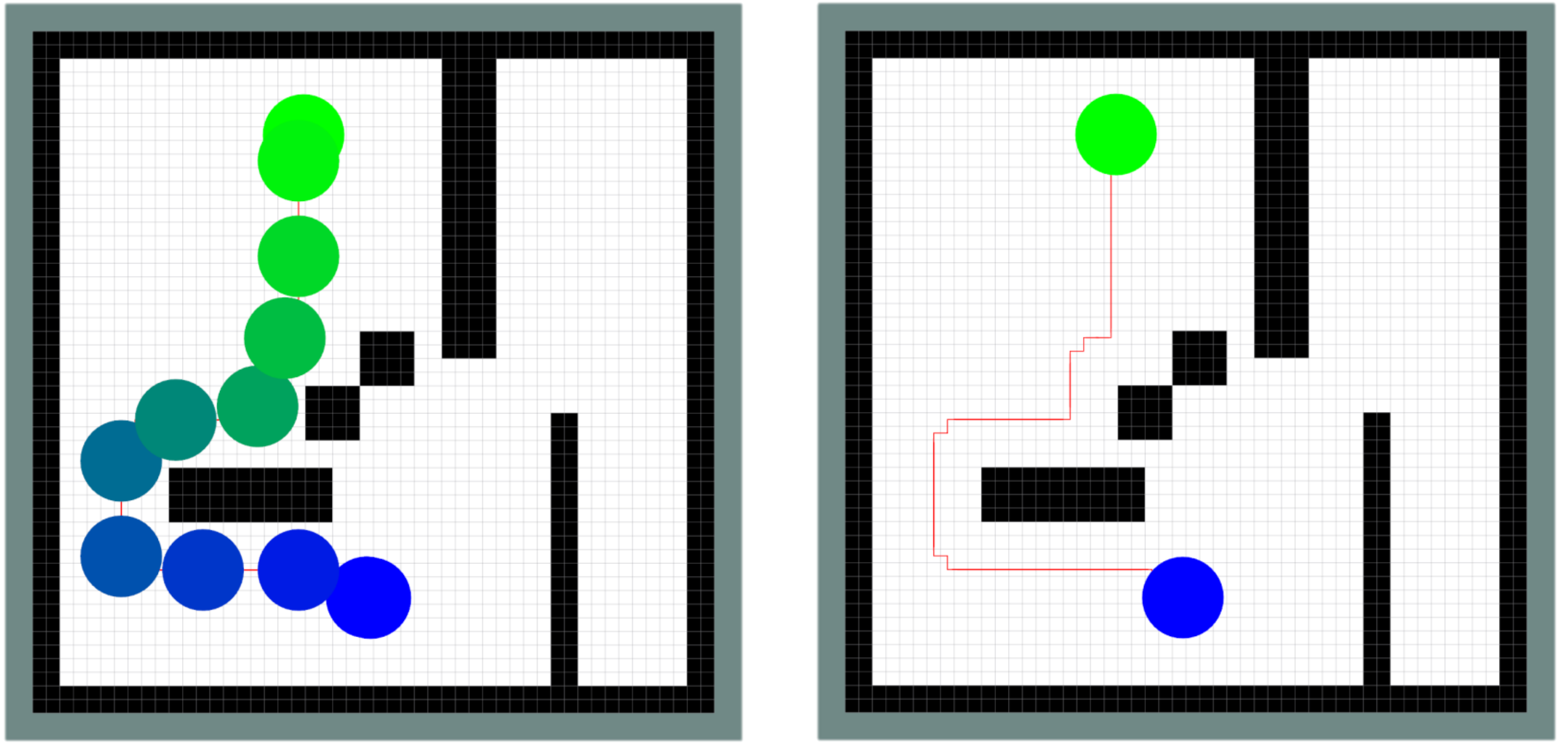
- **Planning in 2-D configuration space**
  - Use the “2D Pose Estimate” and “2D Nav Goal” tools of RViz to initialize the start and goal configurations respectively.



4-different propagations

turn off the option:

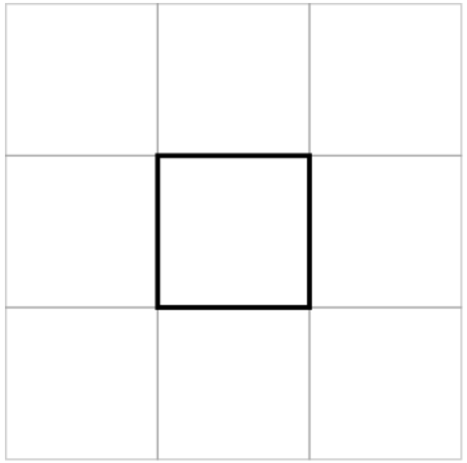
#define USE_8_MOVEMENT



Motions (from green to blue circles) and trajectory (red line)

Tutorial 1: A* planner using a circle

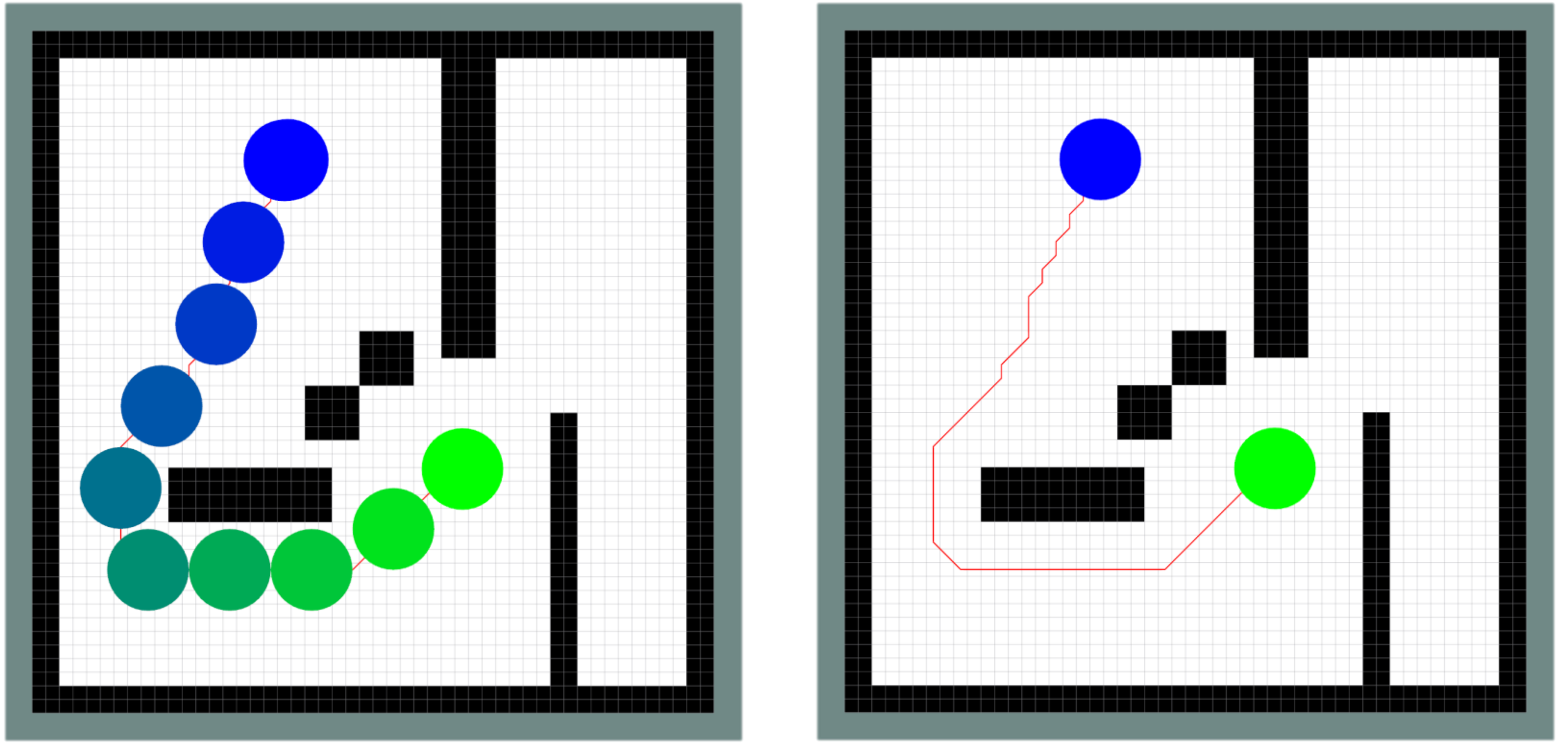
- **Planning in 2-D configuration space**
  - Use the “2D Pose Estimate” and “2D Nav Goal” tools of RViz to initialize the start and goal configurations respectively.



8-different propagations

turn on the option:

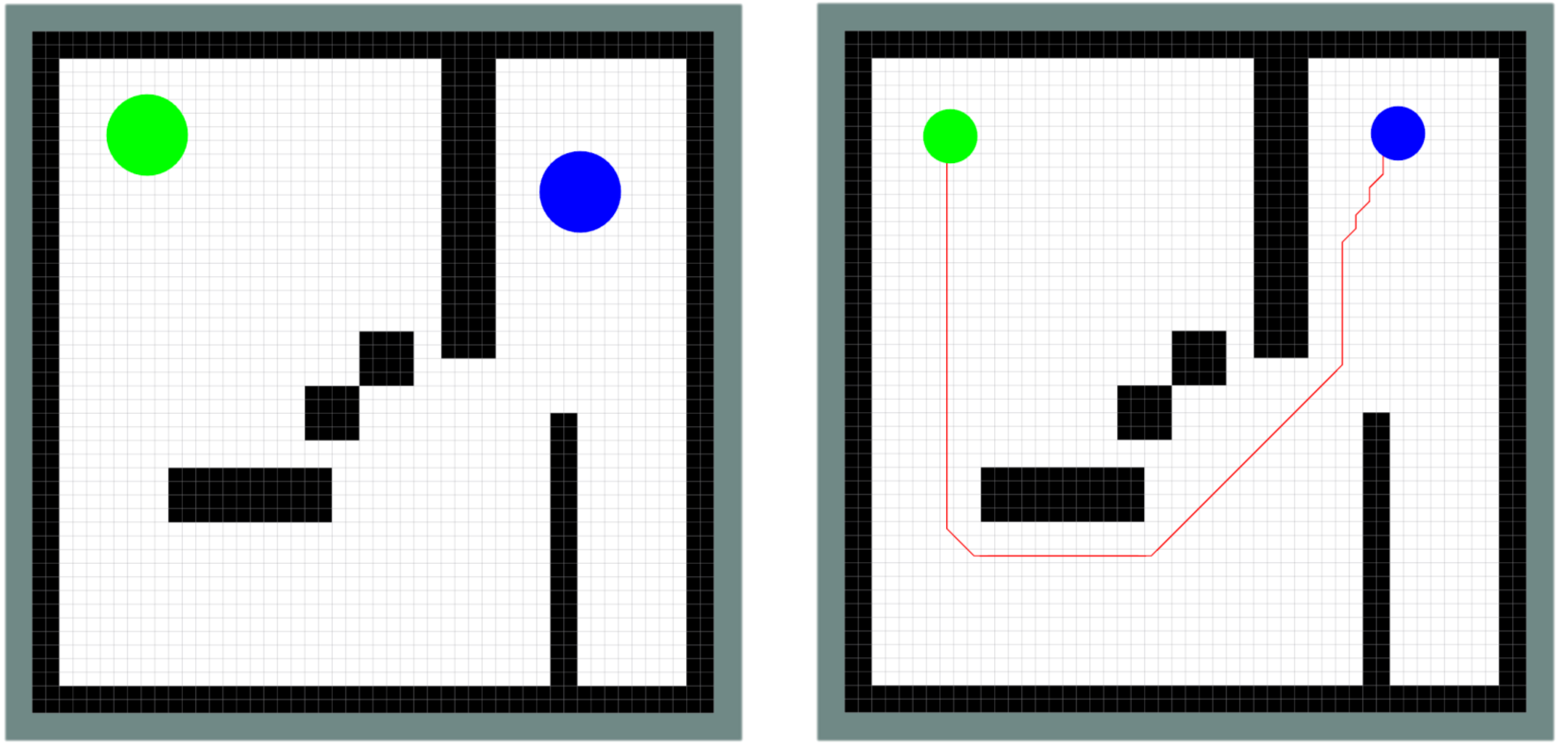
#define USE_8_MOVEMENT



Motions (from green to blue circles) and trajectory (red line)

Tutorial 1: A* planner using a circle

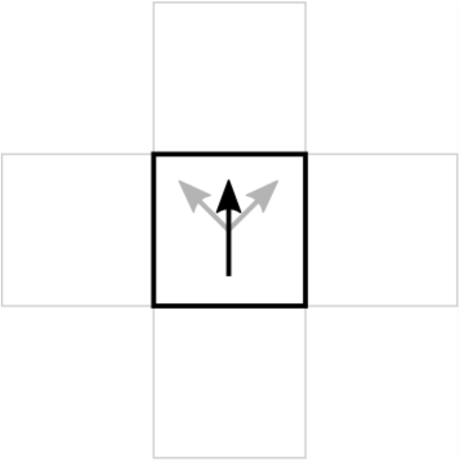
- **Planning in 2-D configuration space**
  - Use the “2D Pose Estimate” and “2D Nav Goal” tools of RViz to initialize the start and goal configurations respectively.
  - The big circle cannot pass a narrow space.
    → no path



Circles having two different radii: 0.3m(left) and 0.2m(right)

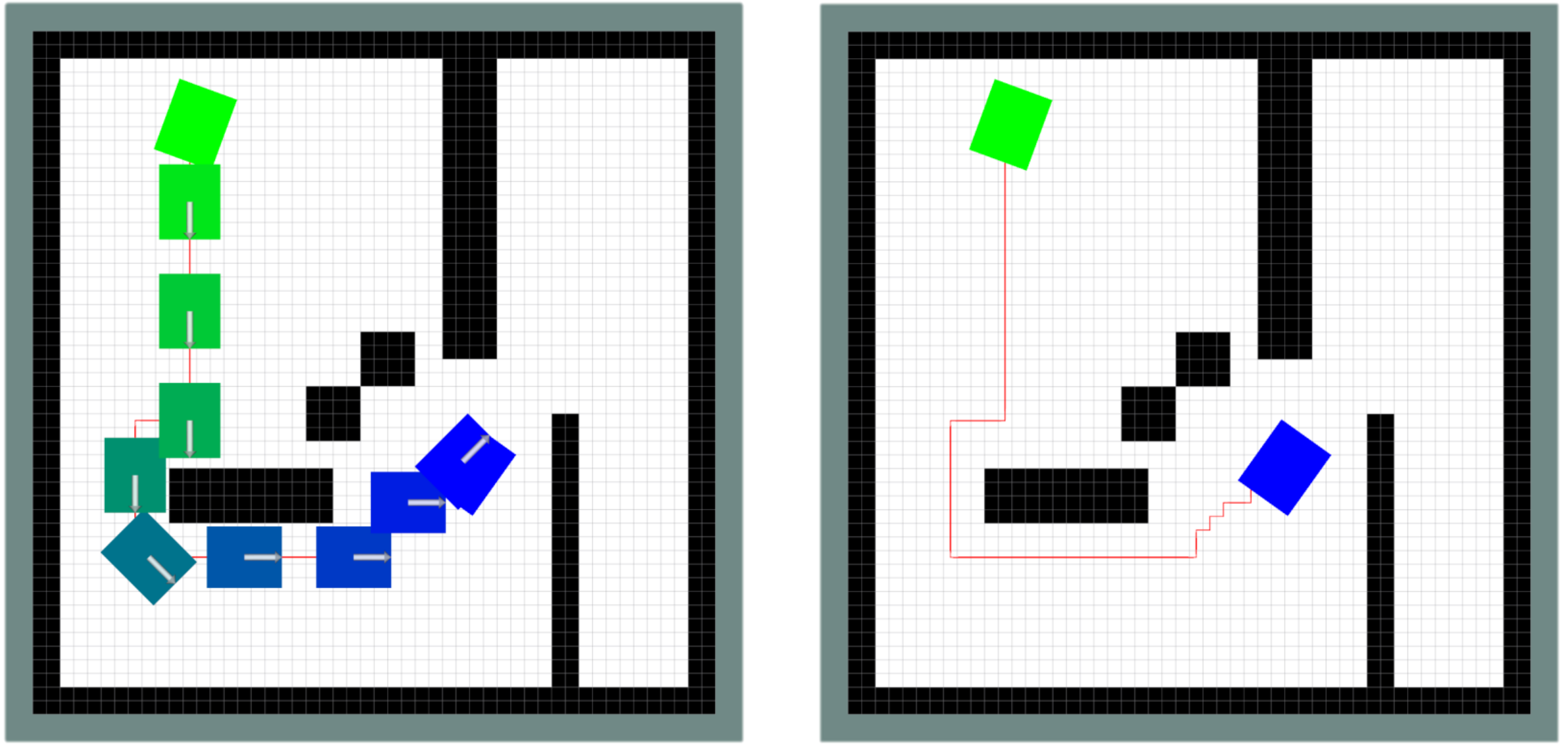
Tutorial 2: A* planner using an OBB

- **Planning in 3-D configuration space**
  - Use the “2D Pose Estimate” and “2D Nav Goal” tools of RViz to initialize the start and goal configurations respectively.



6-different propagations

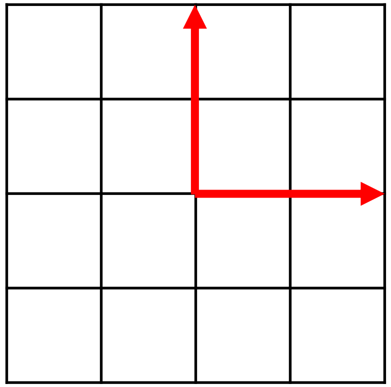
turn off the option:
#define USE_26_MOVEMENT



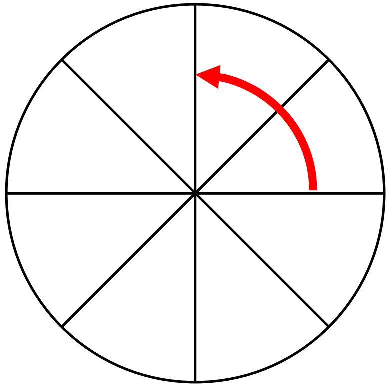
Motions (from green to blue circles) and trajectory (red line)

Tutorial 2: A* planner using an OBB

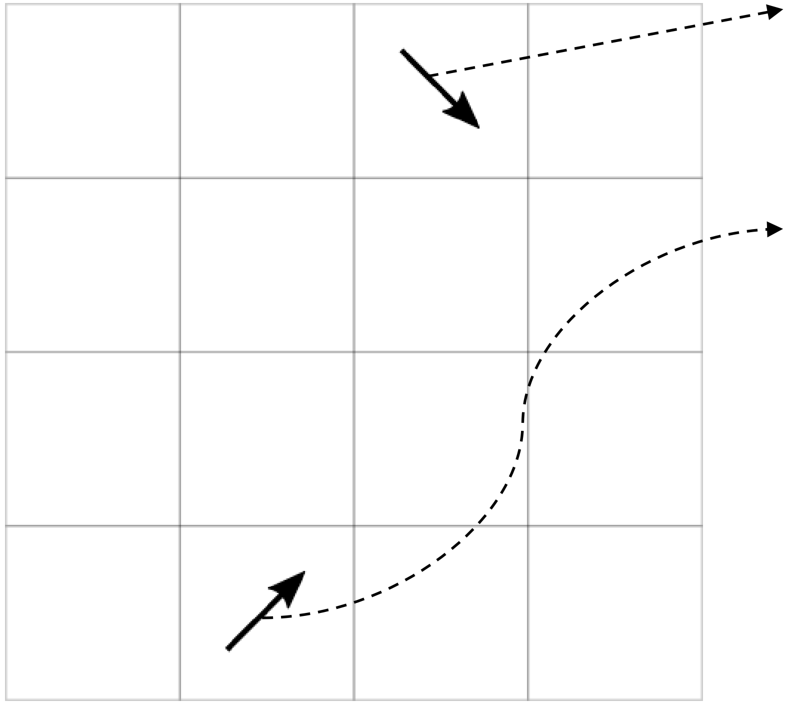
- **Planning in 3-D configuration space**
  - NOTE: implementation issue about designing a cost function



$X - Y$ configuration space partitioning



R configuration space partitioning



A configuration:
$(1, 2, \mathbf{7}) = (1, 2, -\mathbf{1})$

B configuration:
$(-1, -2, \mathbf{1})$
$= (-1, -2, -\mathbf{7})$

Cost from *B* to *A*

$$= \sqrt{\Delta X^2 + \Delta Y^2 + \Delta R^2}$$

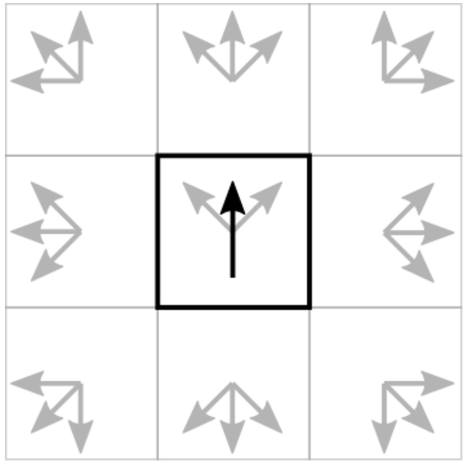
$\Delta X = 1 - (-1) = 2$

$\Delta Y = 2 - (-2) = 4$

$\Delta R = (-1) - 1 = -2$

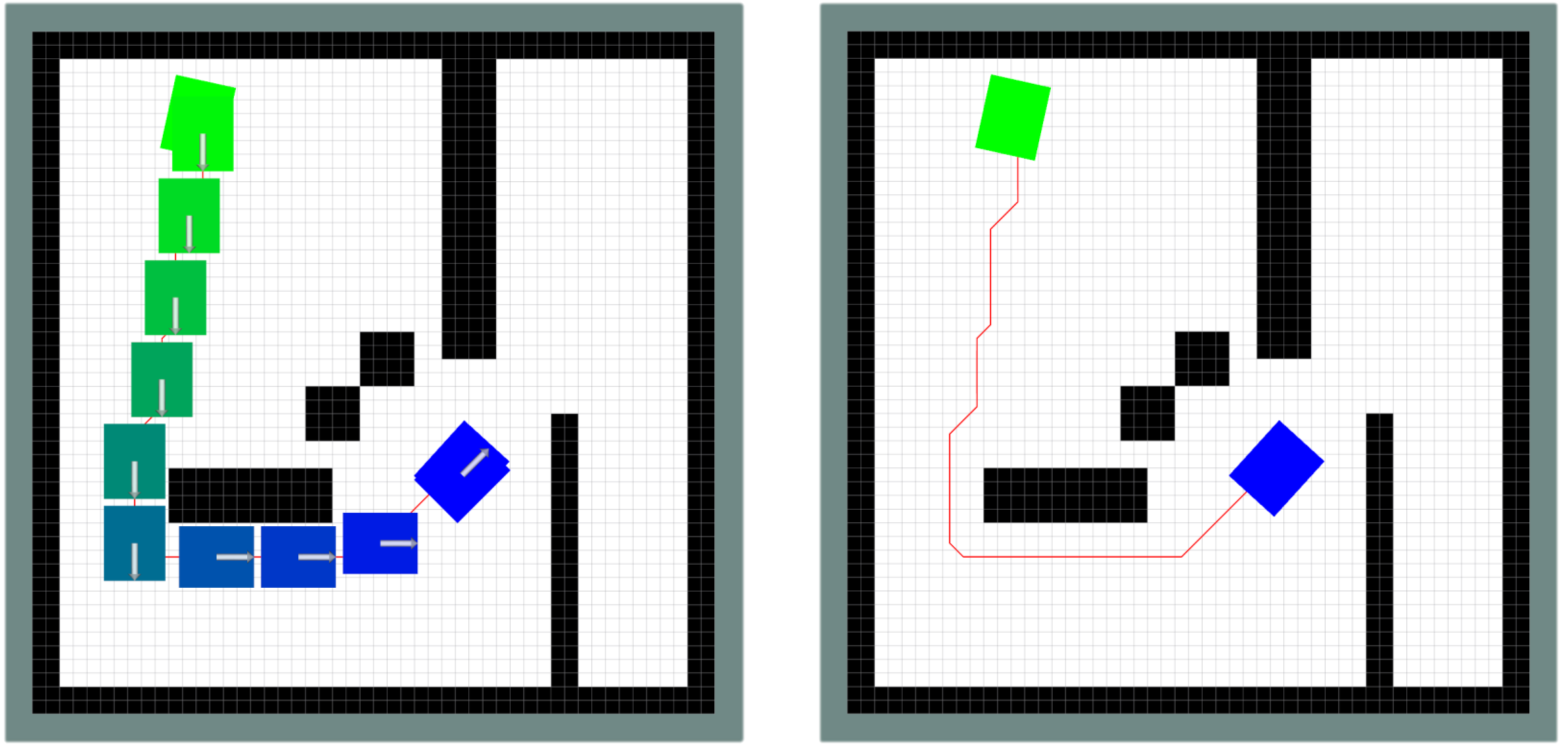
Tutorial 2: A* planner using an OBB

- **Planning in 3-D configuration space**
  - Use the “2D Pose Estimate” and “2D Nav Goal” tools of RViz to initialize the start and goal configurations respectively.



26-different propagations

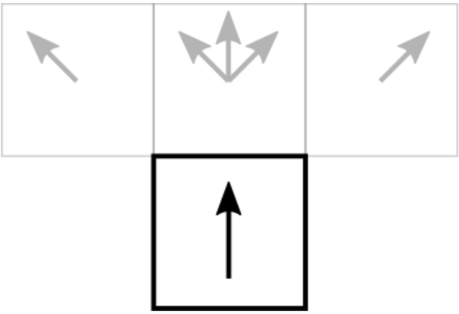
turn on the option:
#define USE_26_MOVEMENT



Motions (from green to blue circles) and trajectory (red line)

Tutorial 2: A* planner using an OBB

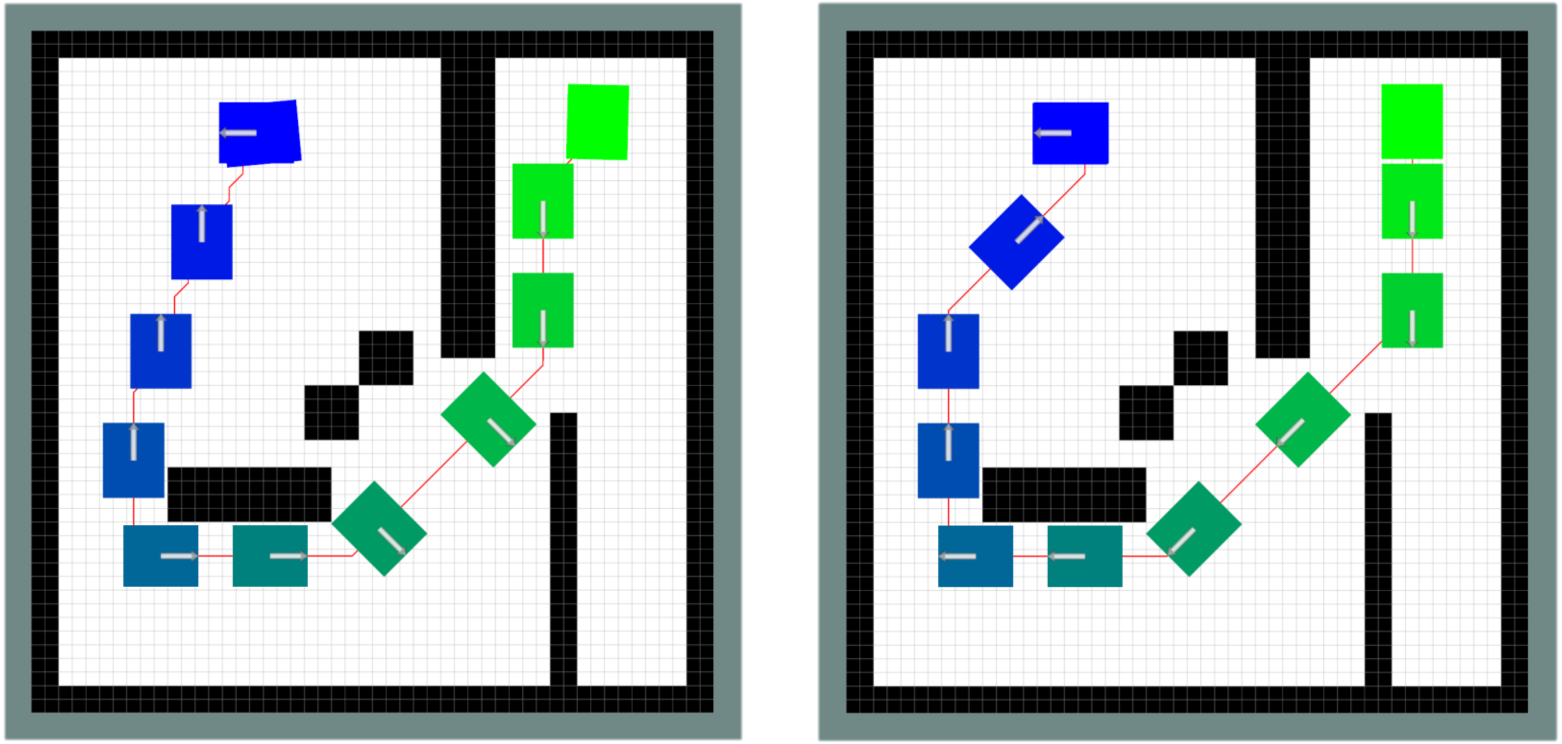
- **Planning in 3-D configuration space**
  - Use the “2D Pose Estimate” and “2D Nav Goal” tools of RViz to initialize the start and goal configurations respectively.



Non-holonomic constraint

turn on the option:

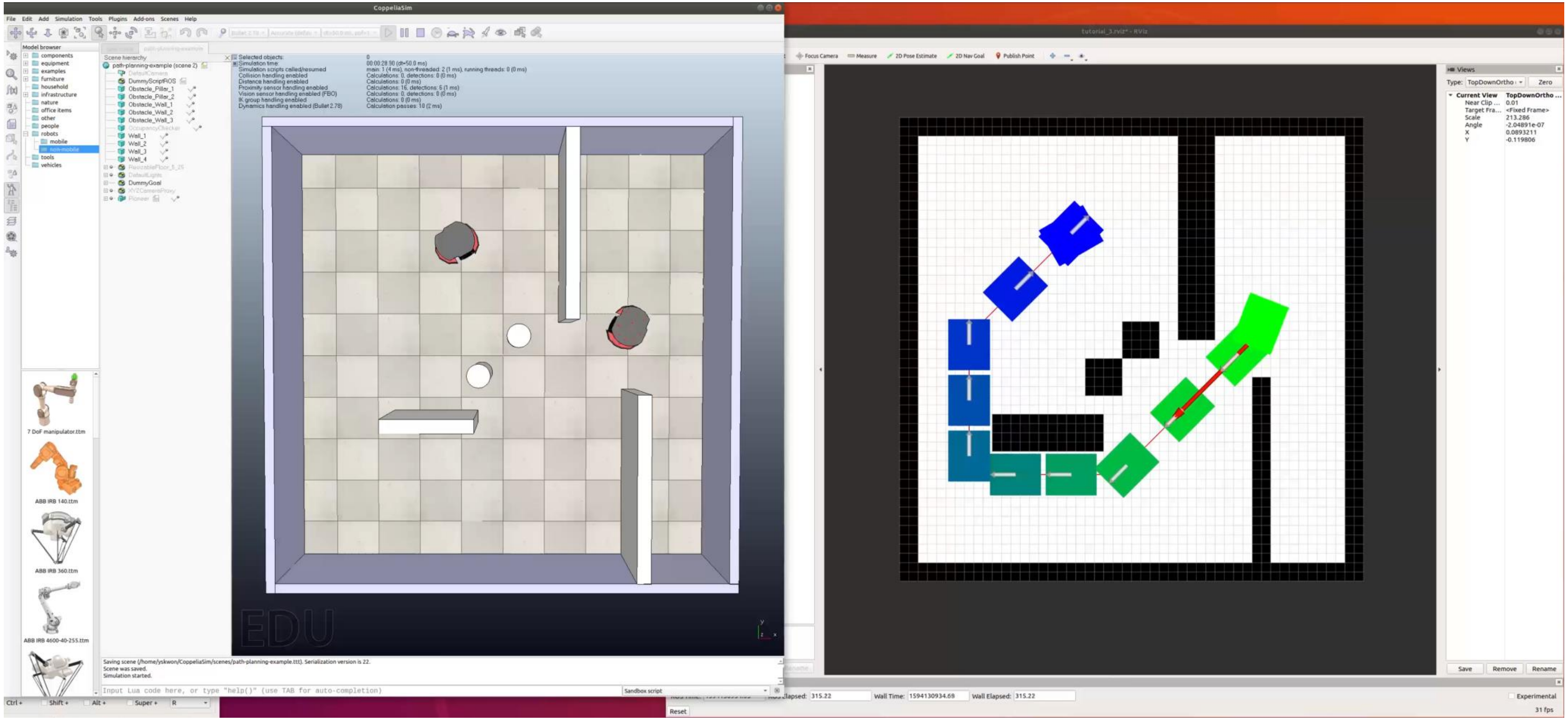
#define USE_NON_HOLONOMIC_CONSTRAINT



Holonomic(left) and Non-holonomic(right) A* planners

Tutorial 3: from planning to navigation

- **Application of planning to mobile robot navigation**
  - ROS + CoppeliaSim(V-REP)



Selected objects:	0
Simulation time:	00:00:28.90 (dt=50.0 ms)
Simulation scripts called/resumed	main: 1 (4 ms), non-threaded: 2 (1 ms), running threads: 0 (0 ms)
Collision handling enabled	Calculations: 0, detections: 0 (0 ms)
Distance handling enabled	Calculations: 0 (0 ms)
Proximity sensor handling enabled	Calculations: 16, detections: 6 (1 ms)
Vision sensor handling enabled (FBO)	Calculations: 0, detections: 0 (0 ms)
IK group handling enabled	Calculations: 0 (0 ms)
Dynamics handling enabled (Bullet 2.78)	Calculation passes: 10 (2 ms)

```
Saving scene (/home/yskwon/CoppeliaSim/scenes/path-planning-example.ttt). Serialization version is 22.
Scene was saved.
Simulation started.
Input Lua code here, or type "help()" (use TAB for auto-completion)
```

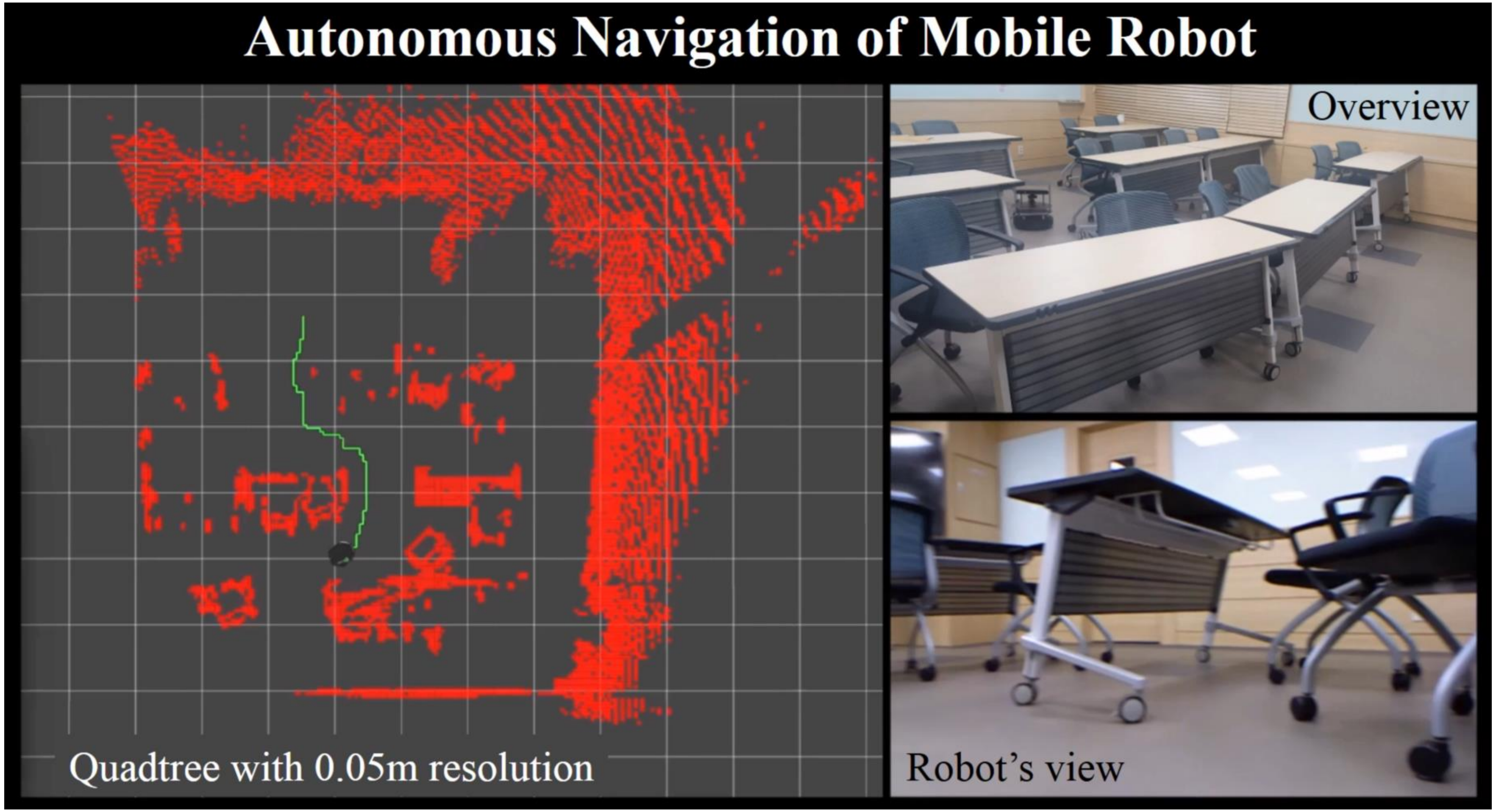
Type: TopDownOrtho

Current View	TopDownOrtho ...
Near Clip ...	0.01
Target Fra...	<Fixed Frame>
Scale	213.286
Angle	-2.04891e-07
X	0.0893211
Y	-0.119806

ROS Elapsed: 315.22 Wall Time: 1594130934.69 Wall Elapsed: 315.22

Tutorial 3: from planning to navigation

- **Application of planning to mobile robot navigation**
  - Real mobile robot, Kobuki, equipping with a RGB-D sensor


Autonomous Navigation of Mobile Robot
Overview
Quadtree with 0.05m resolution
Robot's view

Q&A

Thank you for listening