AIT
ASIAN INSTITUTE OF TECHNOLOGY

Mobile Robot Path Planning and Navigation Control

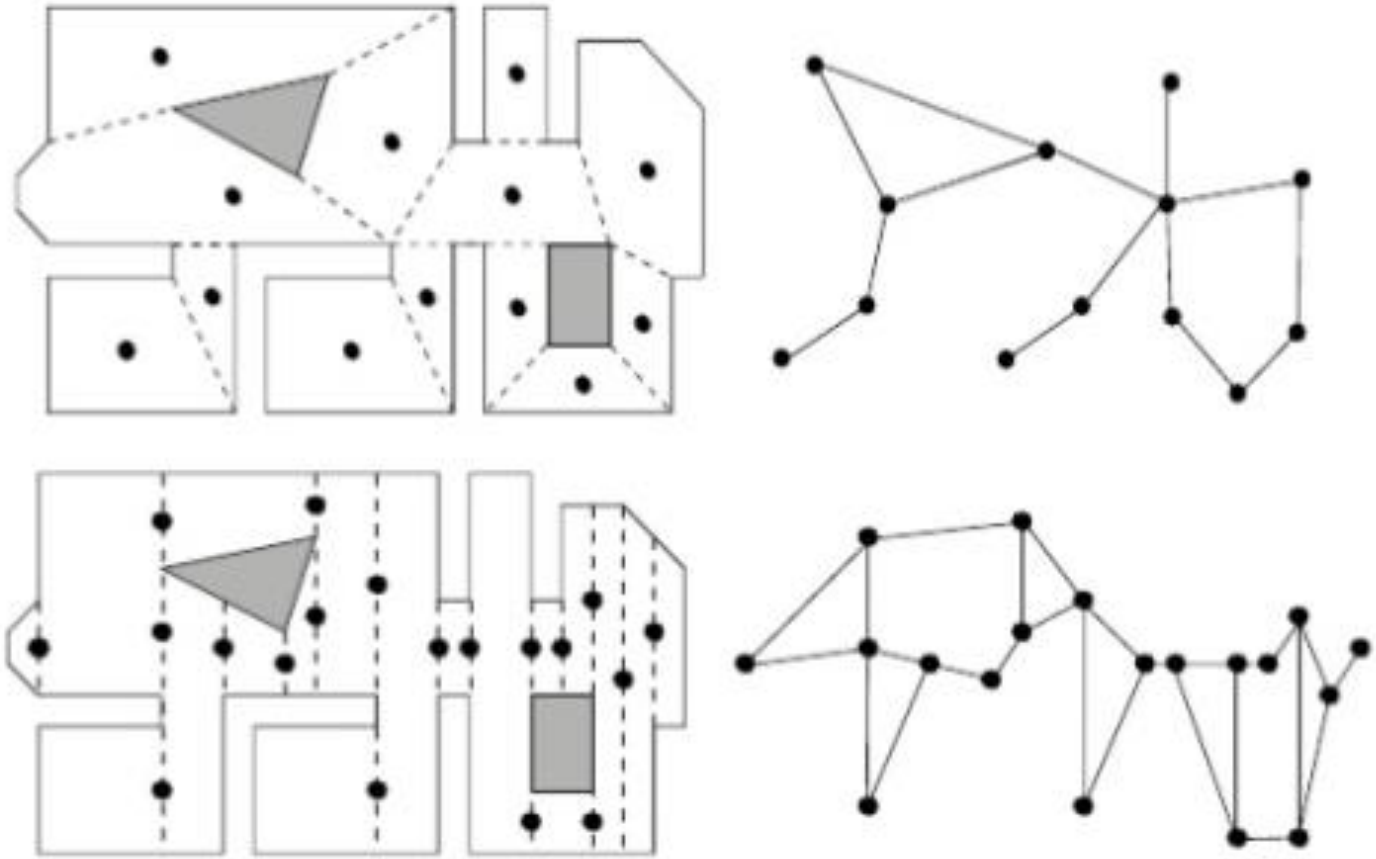
Manukid Parnichkun
School of Engineering and Technology
Asian Institute of Technology
manukid@ait.asia

Mobile Robot Path Planning

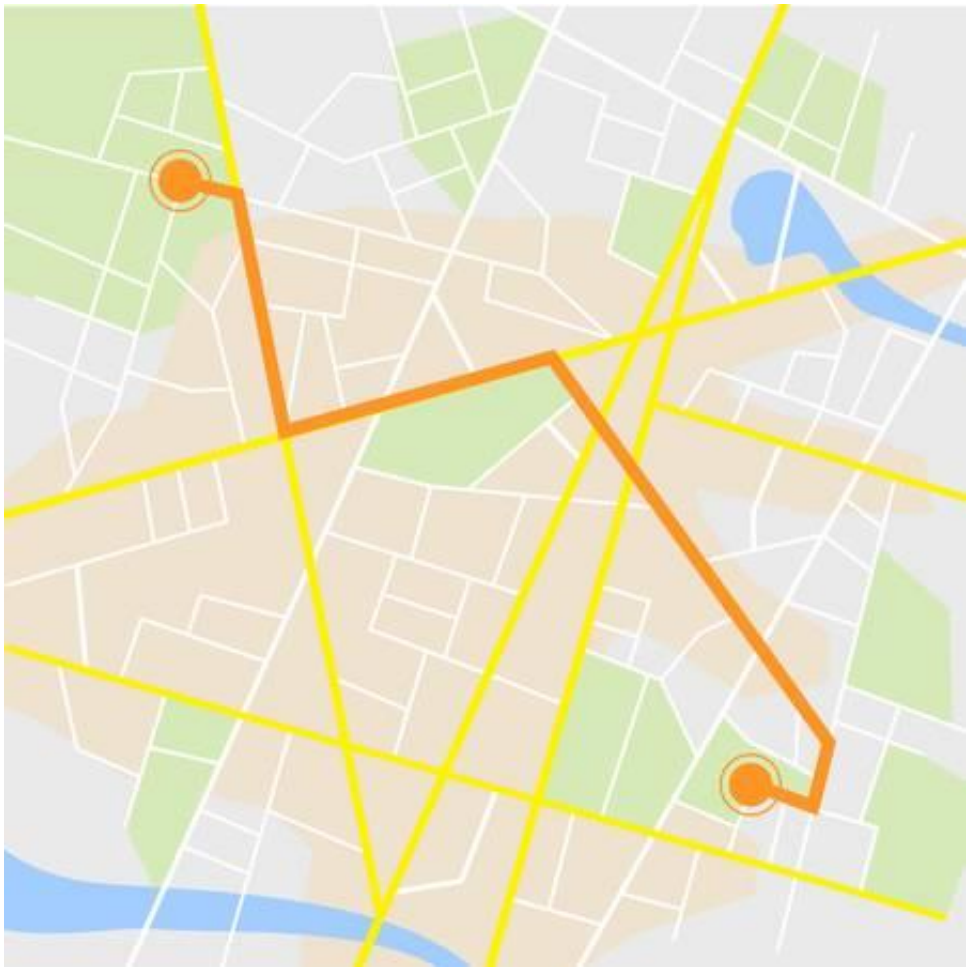
- Global Path Planning (Off Line)
  - Cell Decomposition
  - Road Map
  - Visibility Graph
  - Optimized by A-Star, PSO, GA, etc.
- Local Path Planning (On Line)
  - Potential Field
  - Virtual Forces
  - Rule Based
  - Behavior Based
  - Model Based (PID, State-Space, etc.)

Global Path Planning

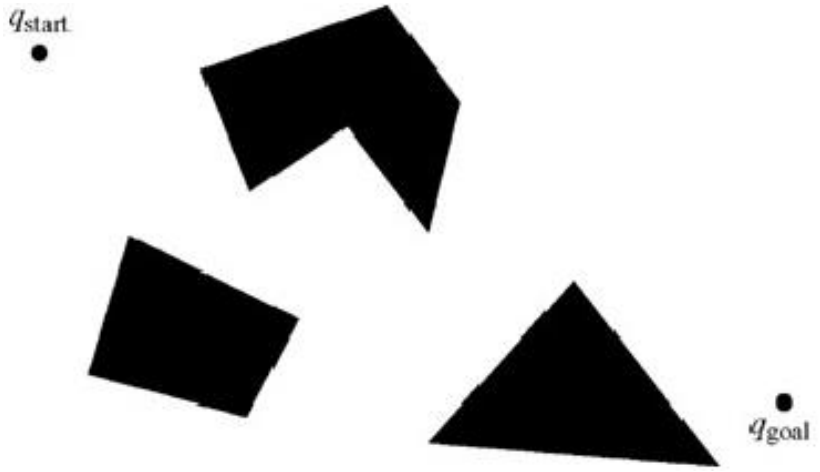
Cell Decomposition

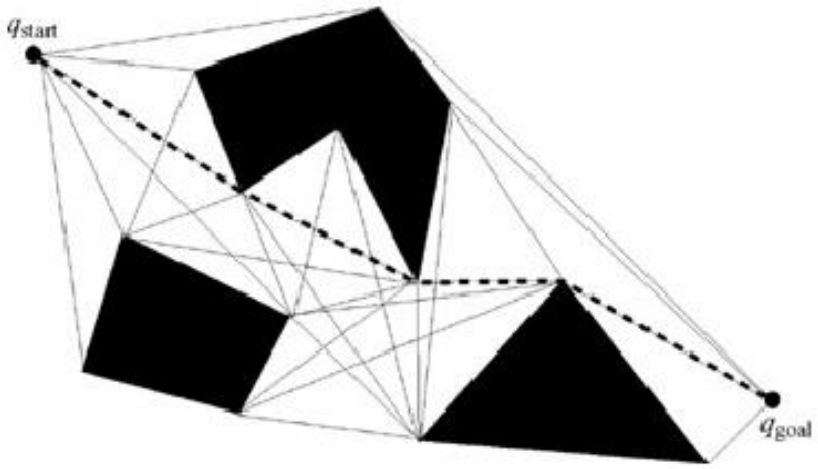


Road Map

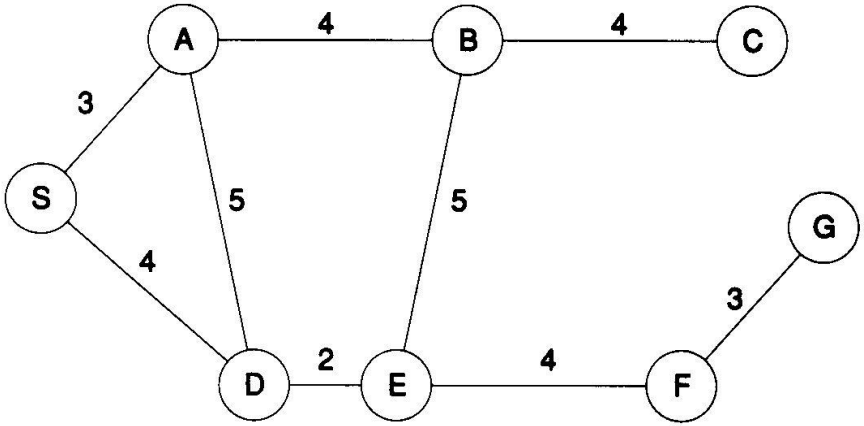


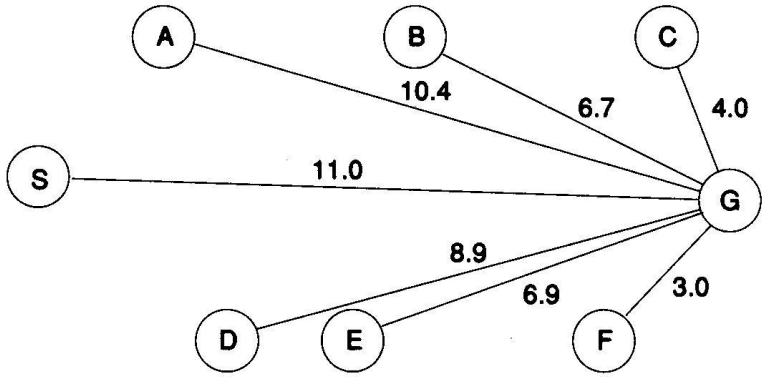
Visibility Graph



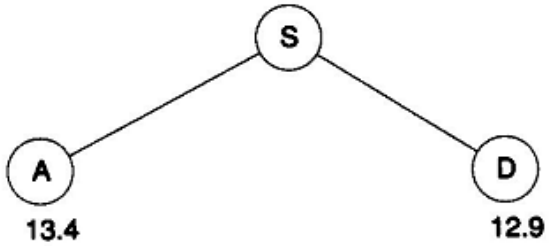


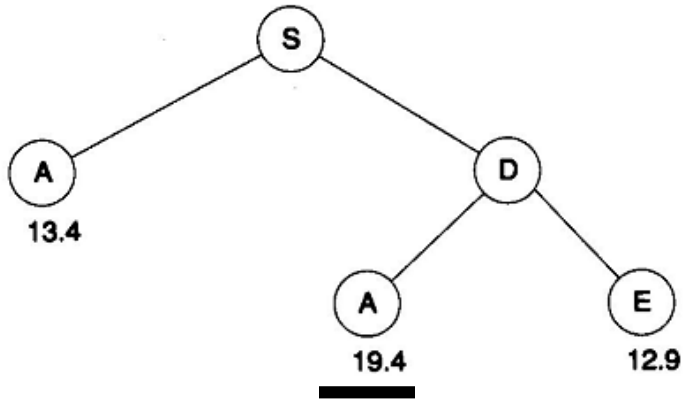
Optimal Search

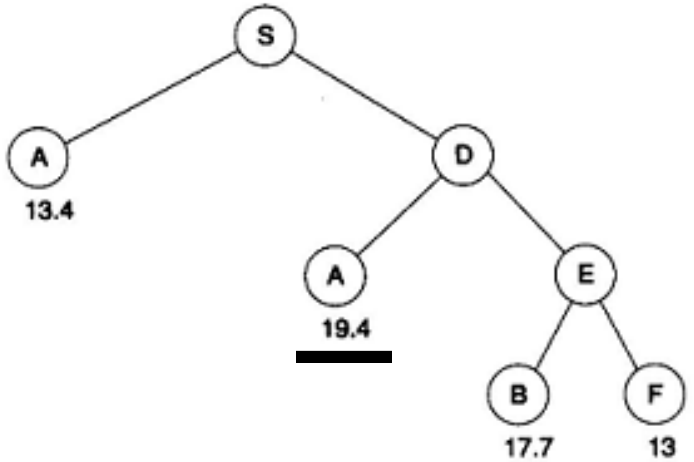


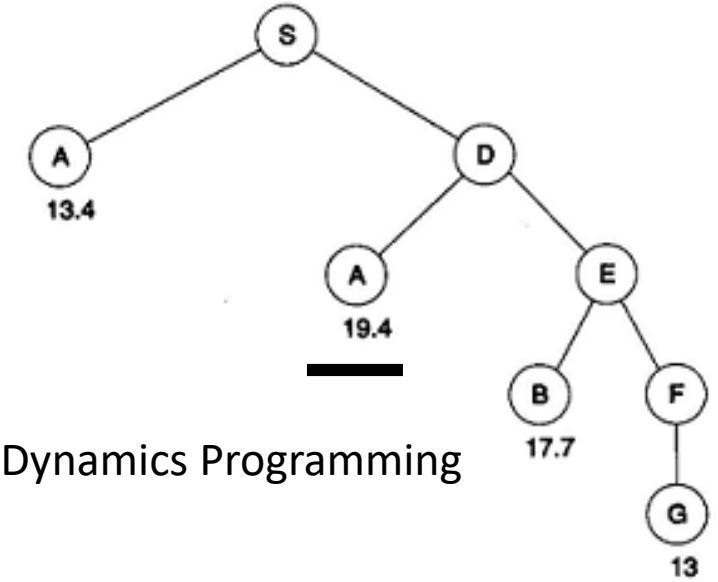


A-Star



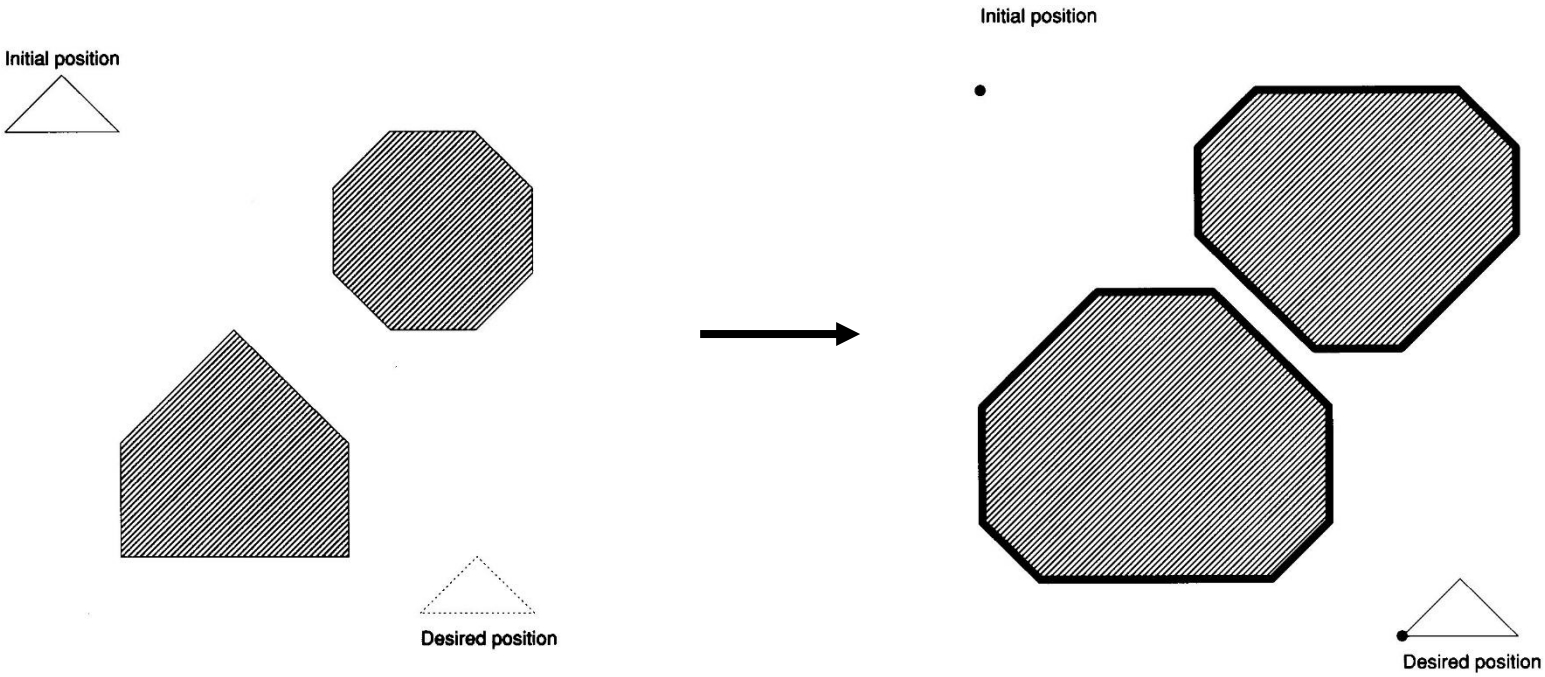




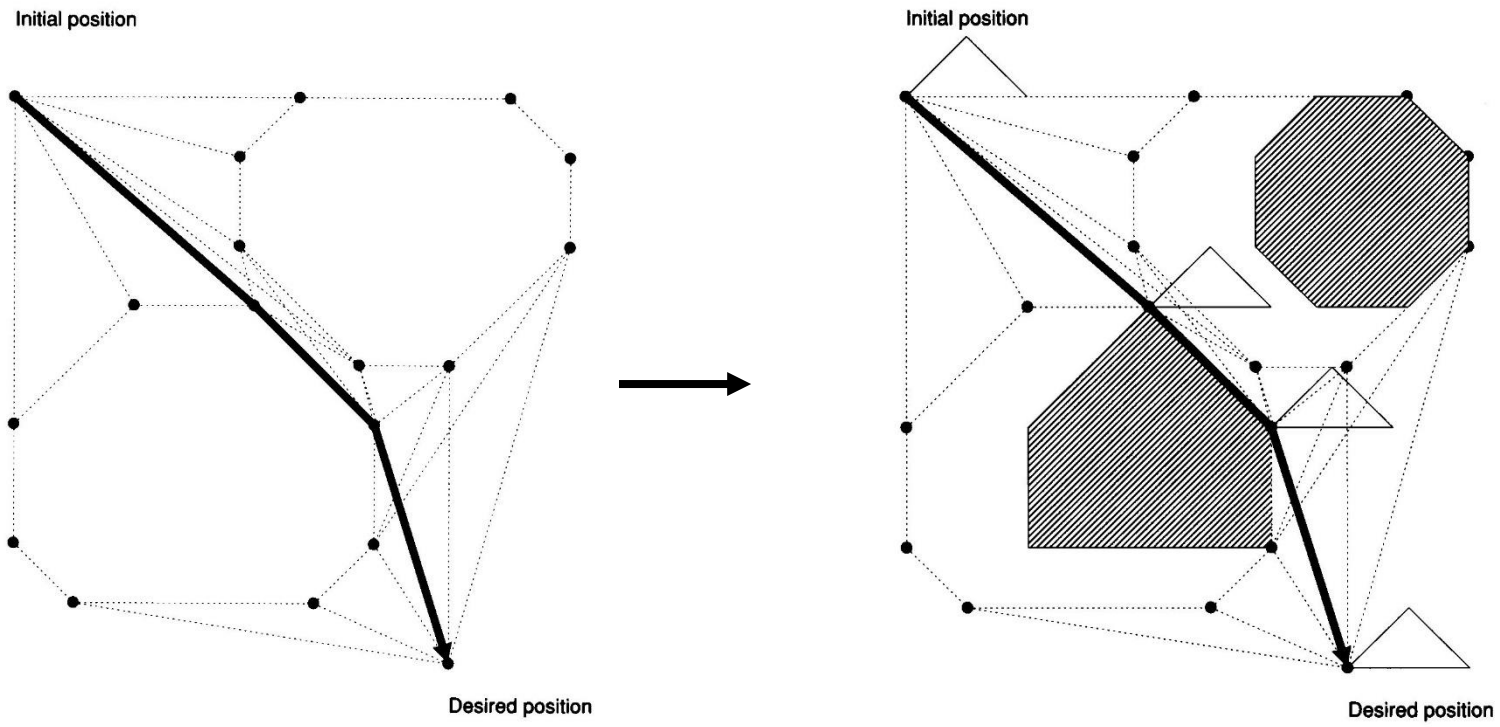


A-Star = Branch and Bound + Under Estimation + Dynamics Programming

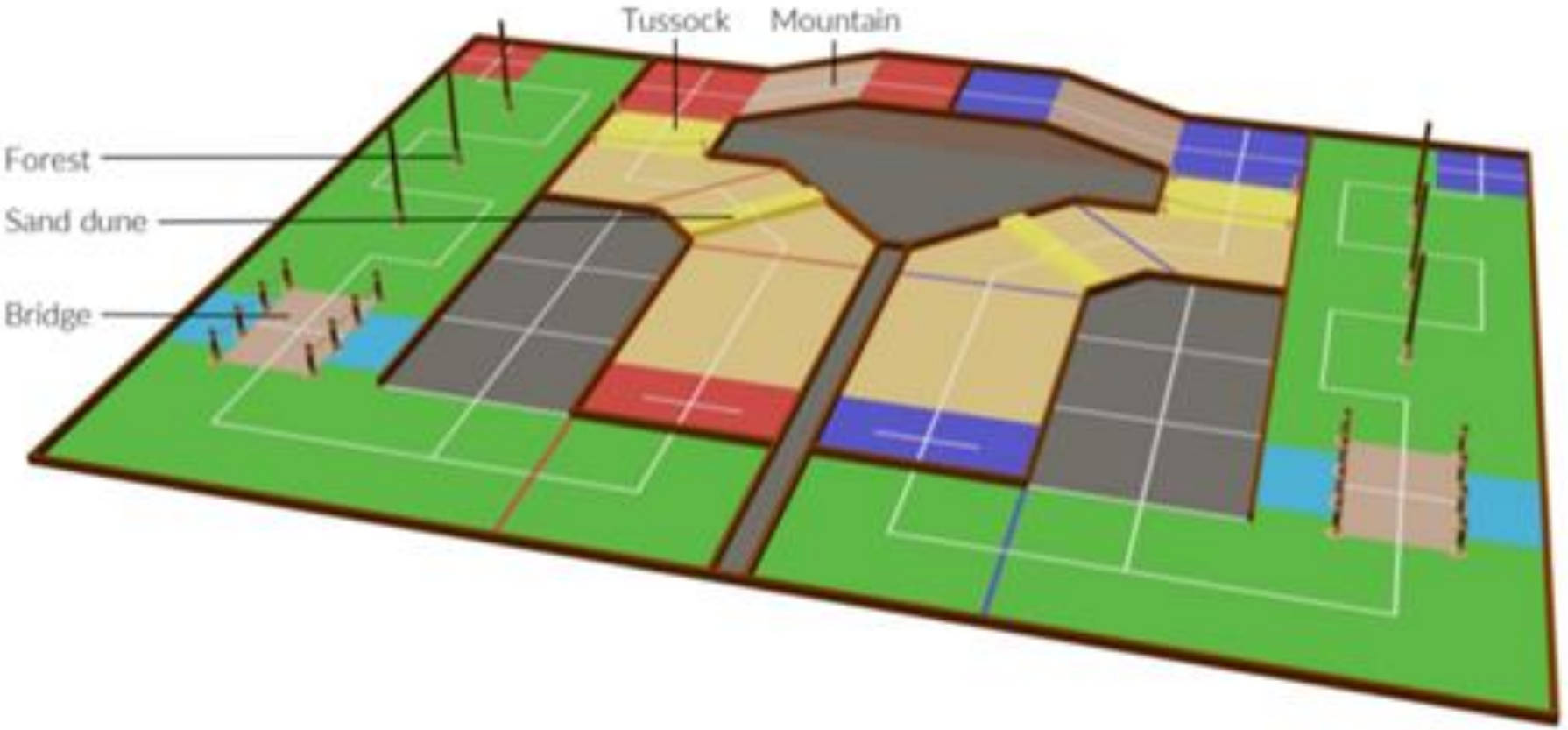
A-Star in Path Planning



A-Star in Path Planning

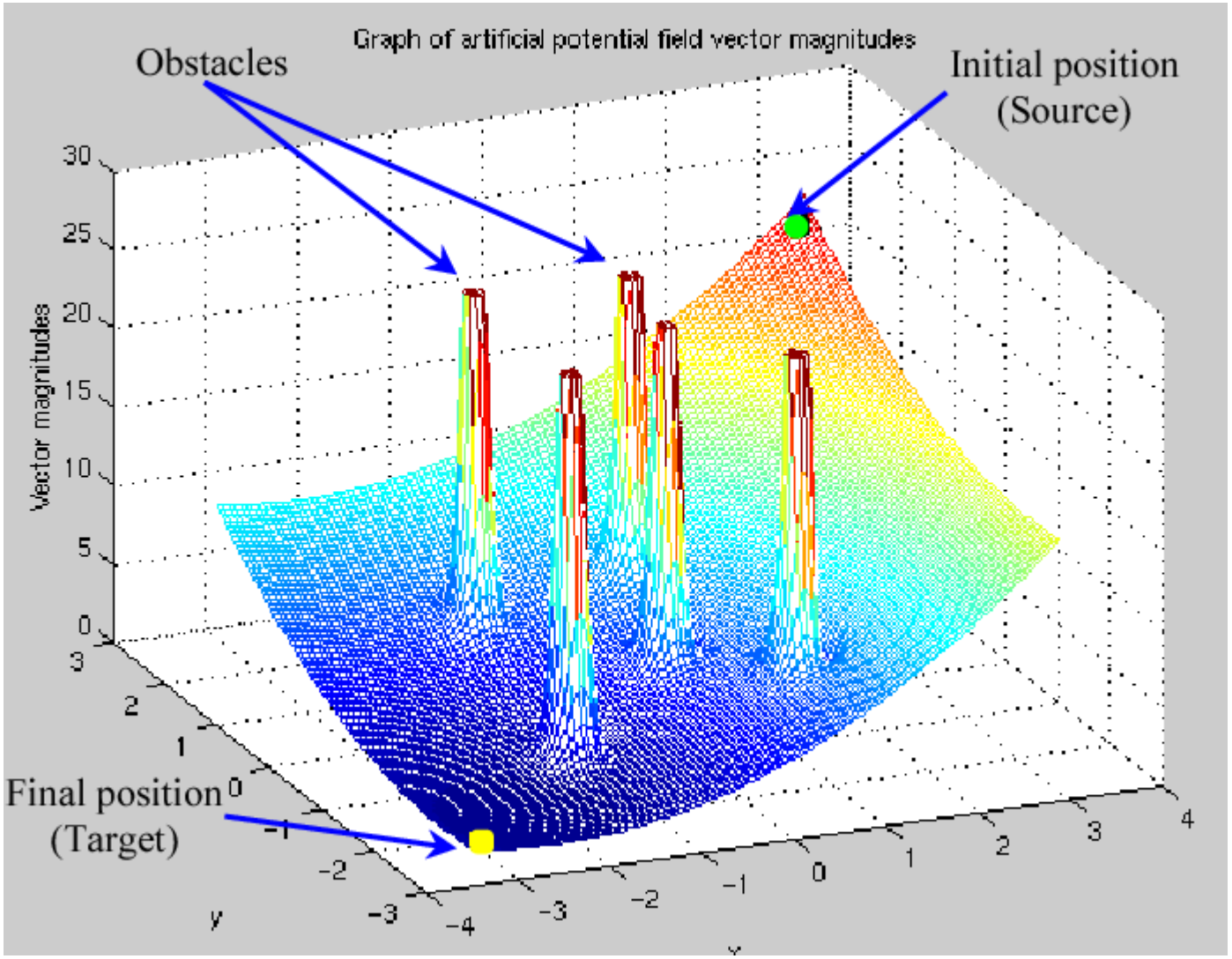


Global Path Planning of ABU Robocon 2019

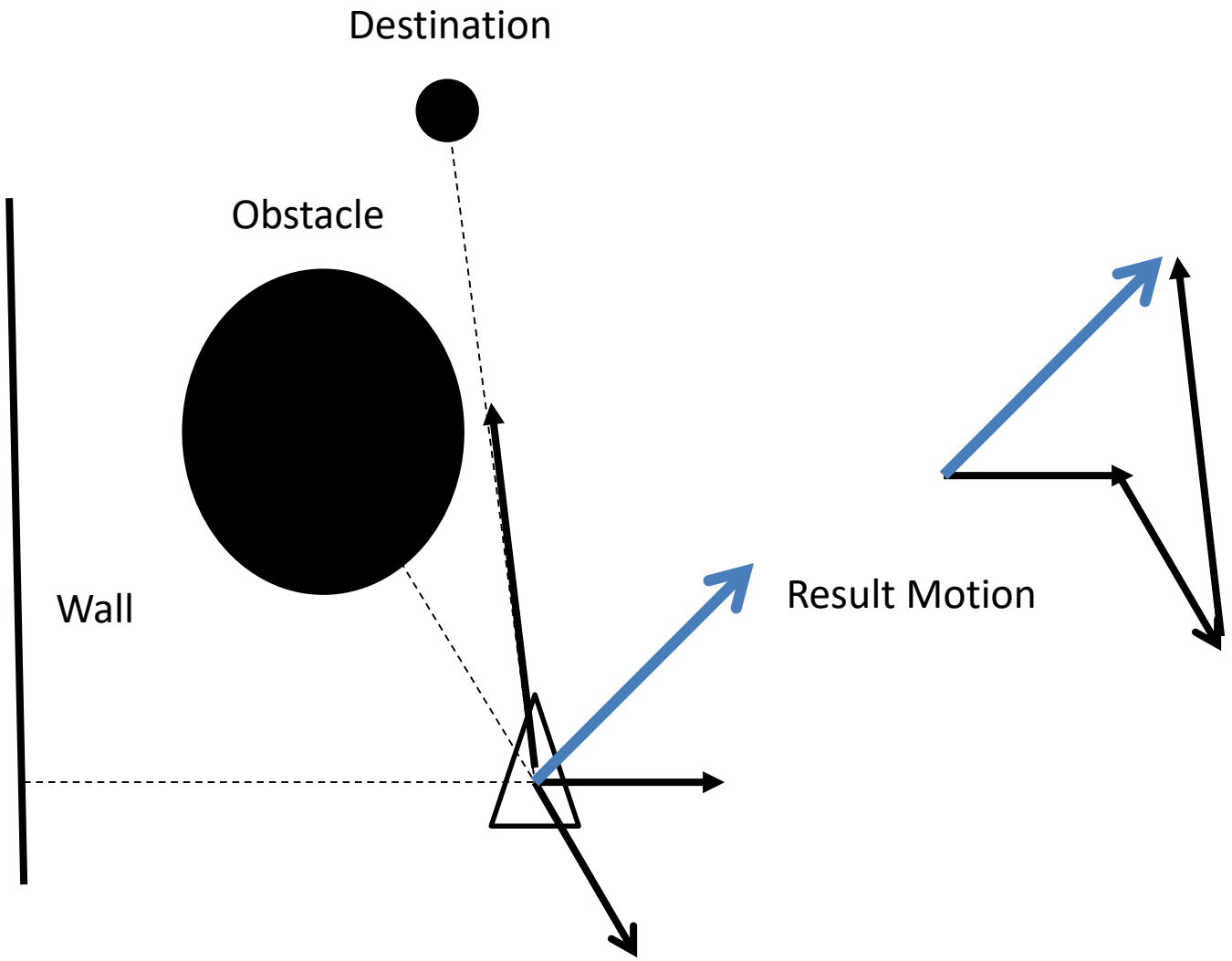


Local Path Planning

Potential Field

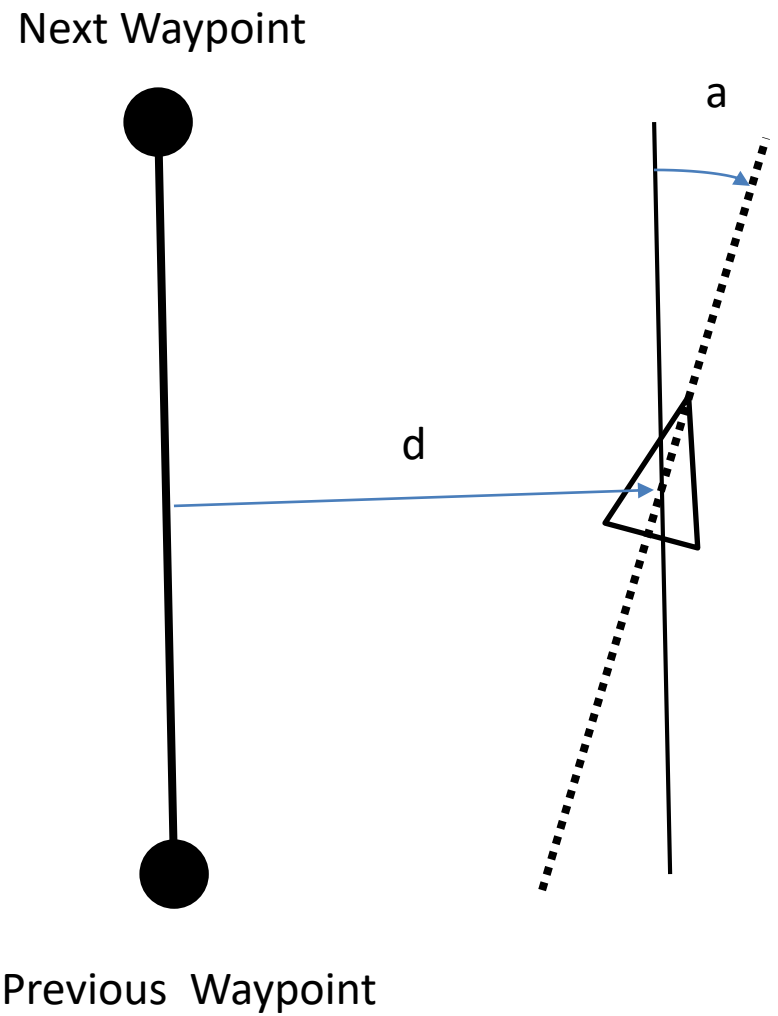


Virtual Forces



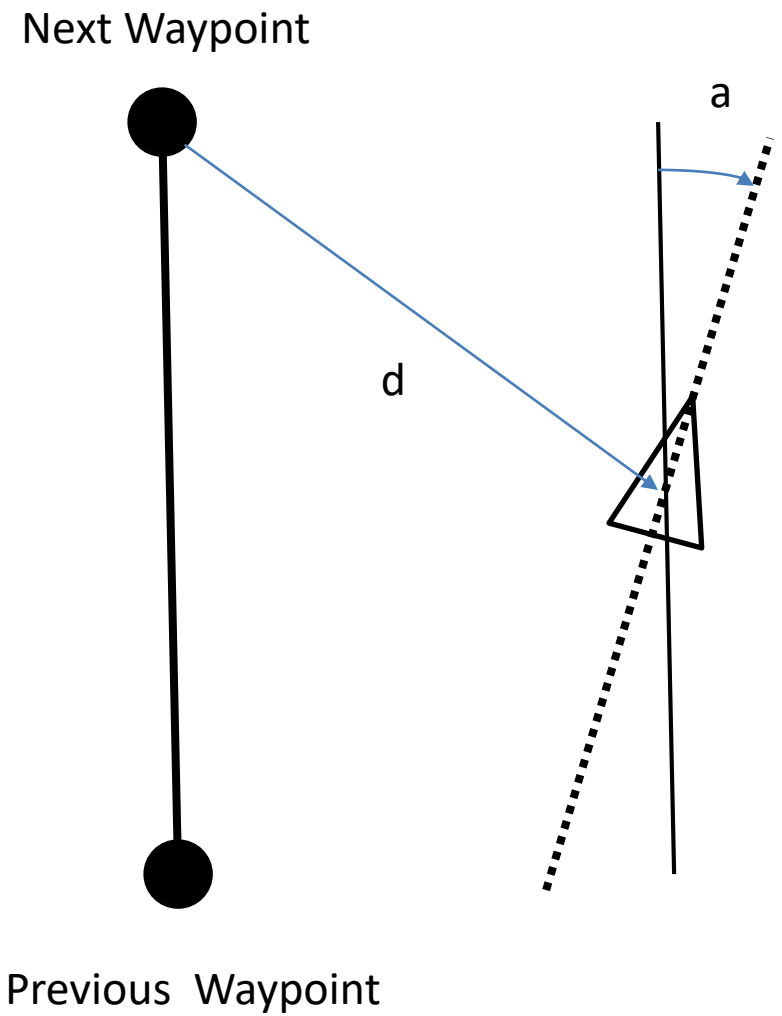
Virtual Force = Virtual Spring Force + Virtual Damping Force

Waypoint Tracking (Road)



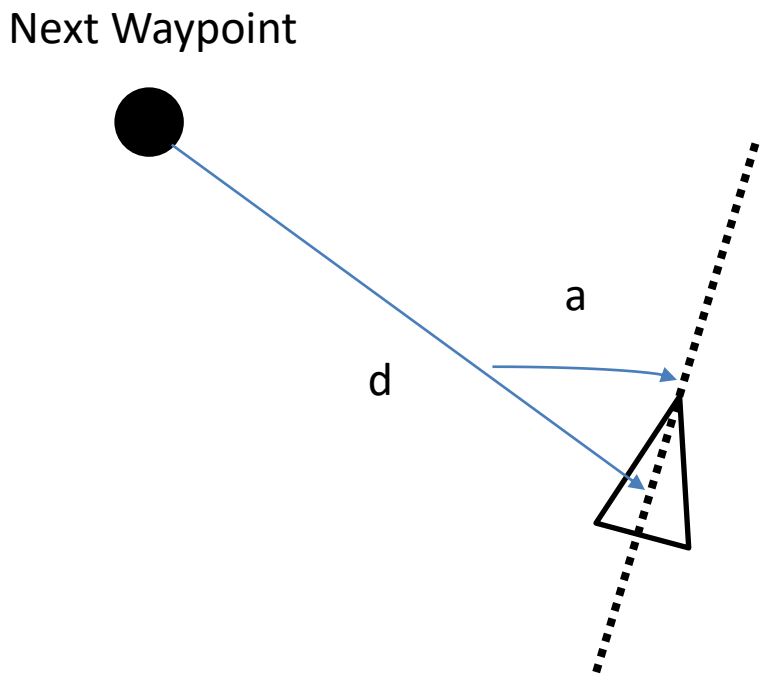
$$Steering = K_{dis\_P}(d) + K_{dis\_D}(\dot{d}) + K_{head\_P}(a) + K_{head\_d}(\dot{a})$$

Waypoint Tracking (Field)



$$Steering = K_{dis\_P}(d) + K_{dis\_D}(\dot{d}) + K_{head\_P}(a) + K_{head\_d}(\dot{a})$$

Waypoint Tracking (Field)

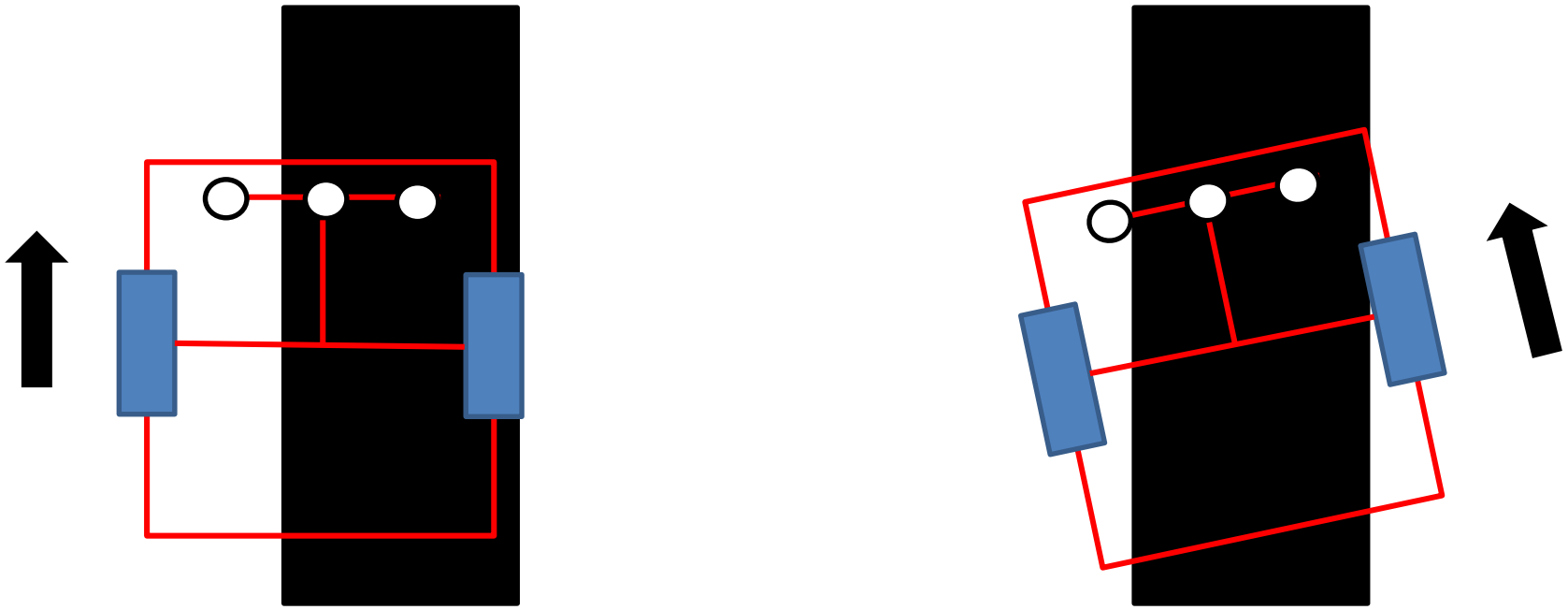


$$Steering = K_{head\_P}(a) + K_{head\_d}(\dot{a})$$

$$Speed = K_{speed\_P}(d) + K_{speed\_d}(\dot{d})$$

Rule-based Control

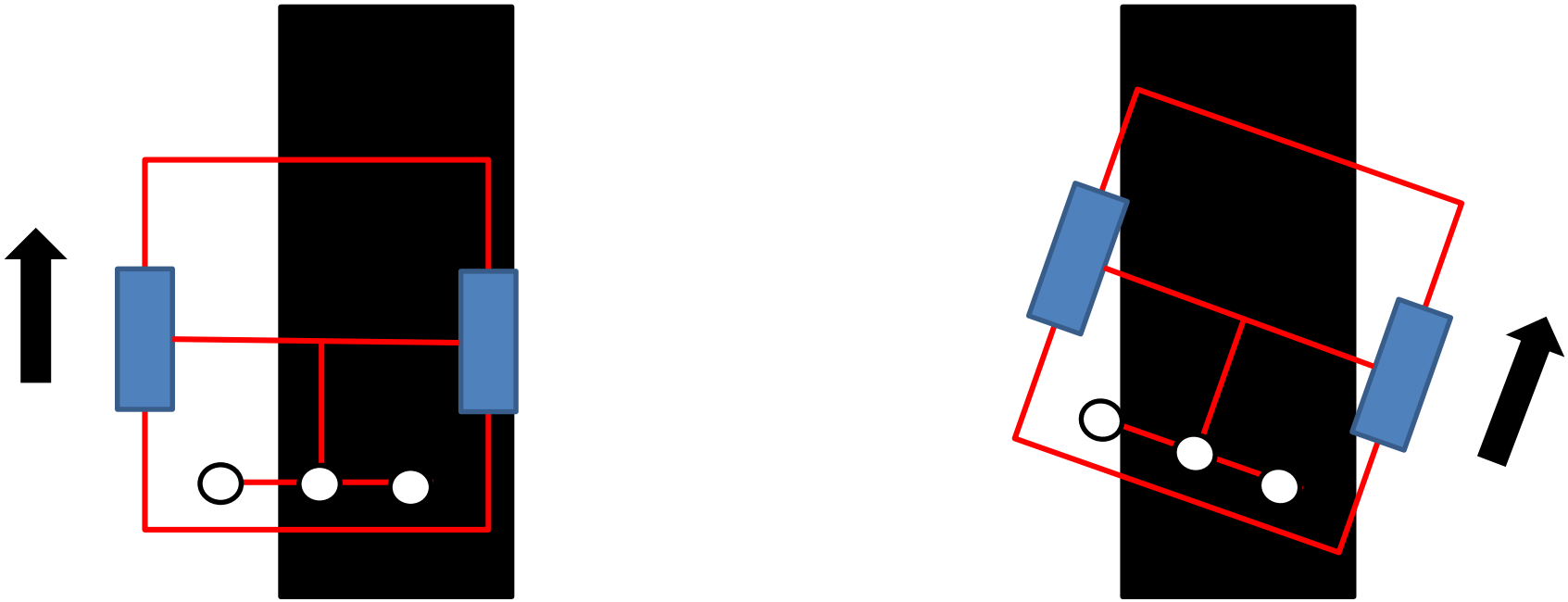
- For line tracking, sensor array should be installed in front of the robot center so that the actions of translation adjustment rule and rotation adjustment rule are not contradict and sensed by the same sensor.



- The robot should turn right for both translation and rotation adjustments to make the left sensor change the state.

Rule-based Control

- If sensor array is installed behind the robot center the actions of translation adjustment rule and rotation adjustment rule are contradict as sensed by the sensor.

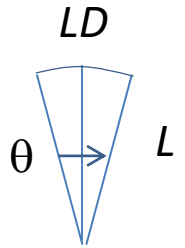


- The robot should turn right for translation adjustment but turn left for rotation adjustment.
- For translation adjustment, the left sensor doesn't change its state.

Mobile Robot Localization

- Dead Reckoning System
  - Encoder
  - Compass
  - Line Tracking Sensor (Ir, LDR, etc.)
- Positioning System
  - Vision (Landmark, SLAM)
  - Laser, Sonar
  - GPS

4-Leg Locomotion Mechanism



– Leg Motion Distance (*LD* (m))

$$LD = \frac{2N\pi L}{R}$$

N = Measured Pulse Number during Ground Contact
L = Leg Length (m)
R = Encoder Pulse per Revolution

– Robot Distance (*RD* (m))

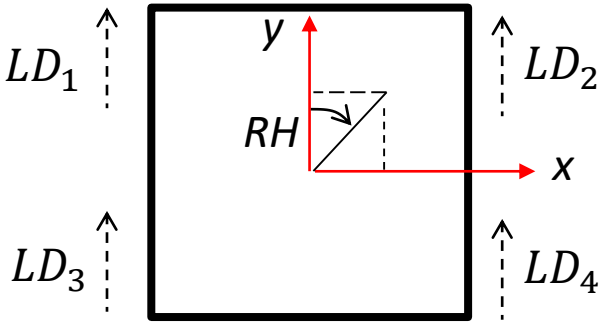
$$RD = \frac{\sum_{i=1}^{n} LD_i}{n}$$

LD_i = Leg no i Motion Distance during Ground Contact
n = 1, 2, 3, or 4 depending on Number of Legs contacting Ground

– Robot Heading (*RH* (degree) positive in clockwise direction))

$$RH = \frac{180(LD_1 - LD_2 + LD_3 - LD_4)}{n\pi D}$$

D = Perpendicular Distance between LD vector and Robot Center



– Robot Coordinate (*x* (m), *y* (m))

$x = \sum(RD \times \sin(\sum RH))$ $y = \sum(RD \times \cos(\sum RH))$

Differential Wheeled Locomotion Mechanism

– Wheel Distance (*WD* (m))

$$WD = \frac{N\pi D}{R}$$

N = Measured Pulse Number
D = Wheel Diameter (m)
R = Encoder Pulse per Revolution

– Robot Distance (*RD* (m))

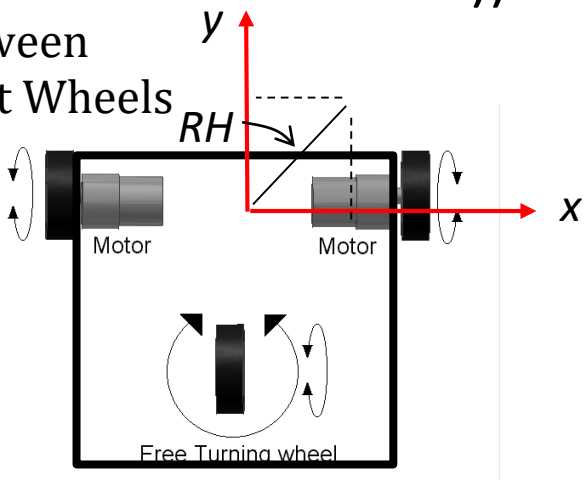
$$RD = \frac{WD_L + WD_R}{2}$$

WD_L = Left Wheel Distance
WD_R = Right Wheel Distance

– Robot Heading (*RH* (degree) positive in clockwise direction))

$$RH = \frac{180(WD_L - WD_R)}{\pi L}$$

L = Distance between Left and Right Wheels



– Robot Coordinate (*x* (m), *y* (m))

$x = \sum(RD \times \sin(\sum RH))$ $y = \sum(RD \times \cos(\sum RH))$

Omni Wheeled Locomotion Mechanism

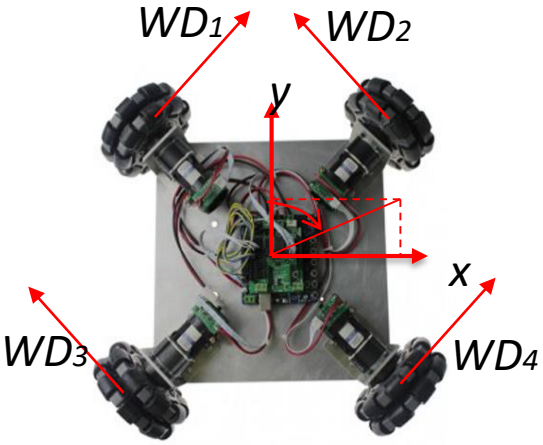
– Wheel Distance (*WD* (m))

$$WD = \frac{N\pi D}{R}$$

N = Measured Pulse Number
D = Wheel Diameter (m)
R = Encoder Pulse per Revolution

– Wheel Coordinate (x_i (m), y_i (m))

$$x_i = \frac{WD_i}{\sqrt{2}} \qquad y_i = \frac{WD_i}{\sqrt{2}}$$



– Robot Coordinate (*x* (m), *y* (m))

$$x = \frac{x_1 - x_2 - x_3 + x_4}{4} \qquad y = \frac{y_1 + y_2 + y_3 + y_4}{4}$$

– Robot Heading (*RH* (degree) positive in clockwise direction))

$$RH = \frac{180(WD_1 - WD_2 + WD_3 - WD_4)}{4\pi S}$$

S = Perpendicular Distance between WD vector and Robot Center

Mecanum Wheeled Locomotion Mechanism

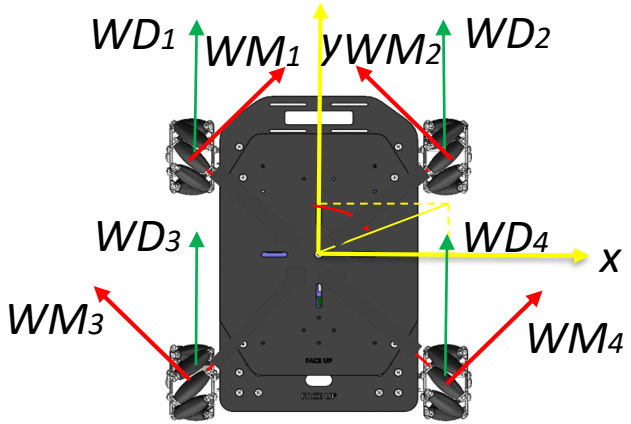
– Wheel Distance (*WD* (m))

$$WD = \frac{N\pi D}{R}$$

N = Measured Pulse Number
D = Wheel Diameter (m)
R = Encoder Pulse per Revolution

– Wheel Real Motion (*WM* (m))

$$WM_i = \frac{WD_i}{\sqrt{2}}$$



– Robot Coordinate (*x* (m), *y* (m))

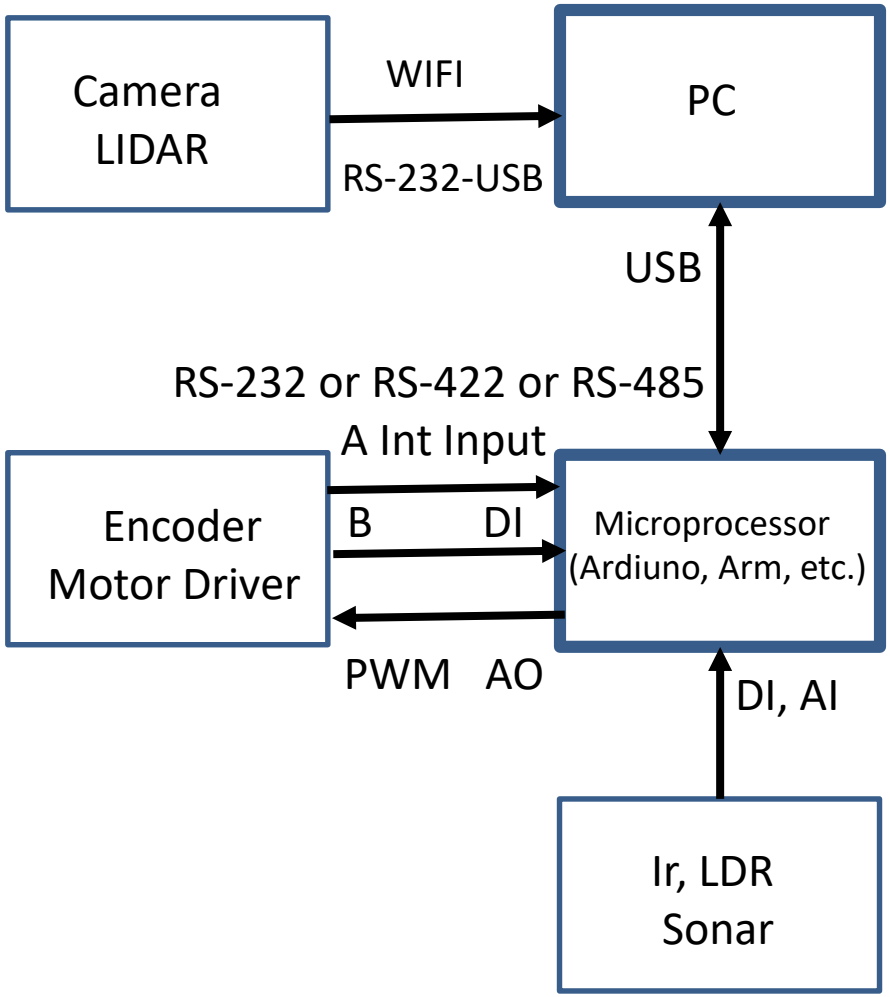
$$x = \frac{WM_1 - WM_2 - WM_3 + WM_4}{4\sqrt{2}} \qquad y = \frac{WM_1 + WM_2 + WM_3 + WM_4}{4\sqrt{2}}$$

– Robot Heading (*RH* (degree) positive in clockwise direction))

$$RH = \frac{180(WM_1 - WM_2 + WM_3 - WM_4)}{4\pi C}$$

C = Perpendicular Distance between WM vector and Robot Center

Robot System Architecture



For GUI,
- Global Path Planning
- Local Path Planning
- Camera, LIDAR

C#, C++ on Visual Studio or
Python on Raspberry Pi

For Low Level Control
- Motor Driver
- Encoder
- Ir, LDR, Sonar

C on Ardiuno or
C on mbed.com -> Hardware ->
Boards -> STMicroelectonics ->
NUCLEO411RE (for Cortex M4)

Q & A