

# AAE2004 Introduction to Aviation Systems

## AAE

### Design of Path Planning Algorithm for Aircraft Operation

#### Week 4: Path Planning Algorithm and Python Robotics

---

Dr Li-Ta Hsu and Dr Weisong Wen

Assisted by

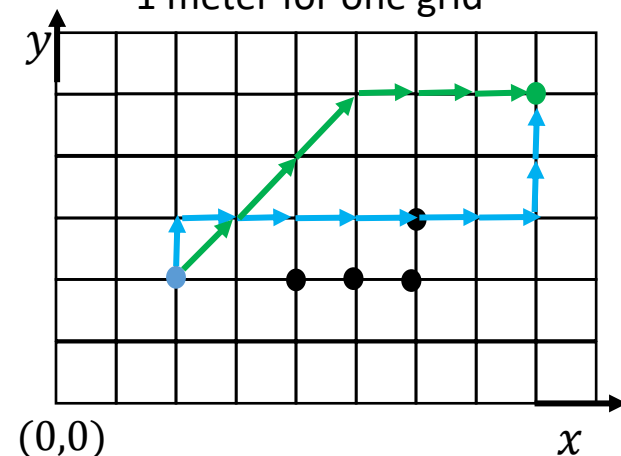
Man Hei CHENG (Melvin), Miss Hiu Yi HO (Queenie), Miss Yan Tung LEUNG (Nikki)

# A\* Path Planning Algorithm

---

# Definition of Path Planning

1 meter for one grid



● Start node

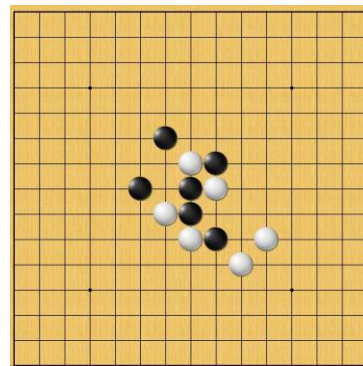
● Goal node

→ Route 1

→ Route 2

• **Node** — All potential position you can go across with a unique position  $(x, y)$

• **Search Space** — A collection of nodes, like all board positions of a board game.



Gobang

• **Objective of path planning**— Find the shortest routes with smallest cost from start node to goal node.

How to find the shortest route!

# Path Planning using A Star Method

Definition of cost:

$$f(x, y) = g(x, y) + h(x, y)$$

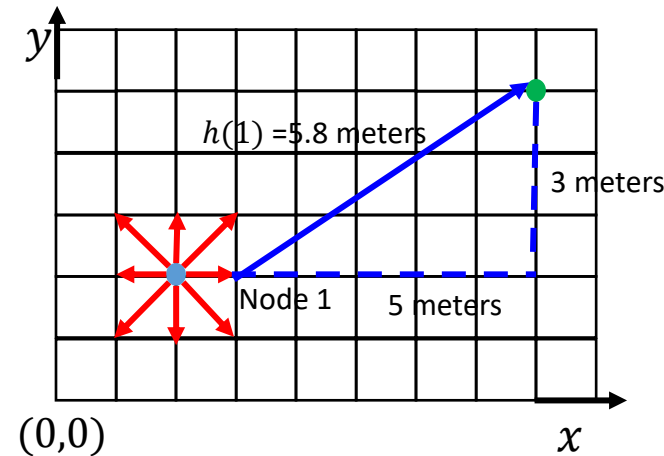
- $g(x, y)$  — this represents the **exact cost** of the path from the **starting node** to node  $(x, y)$
- $h(x, y)$  — this represents the heuristic **estimated cost** from node  $(x, y)$  to the goal node.
- $f(x, y)$  — cost of the neighboring node  $(x, y)$

8 neighboring node and the cost can be calculated as follows!

Node 1:

$$f(3, 2) = g(3, 2) + h(3, 2) = 6.8 \text{ meters}$$

with  $g(3, 2) = 1 \text{ meter}$  and  $h(3, 2) = 5.8 \text{ meters}$



● Start node

● Goal node

1 meter for one grid

# Path Planning using A Star Method

Definition of cost:

$$f(x, y) = g(x, y) + h(x, y)$$

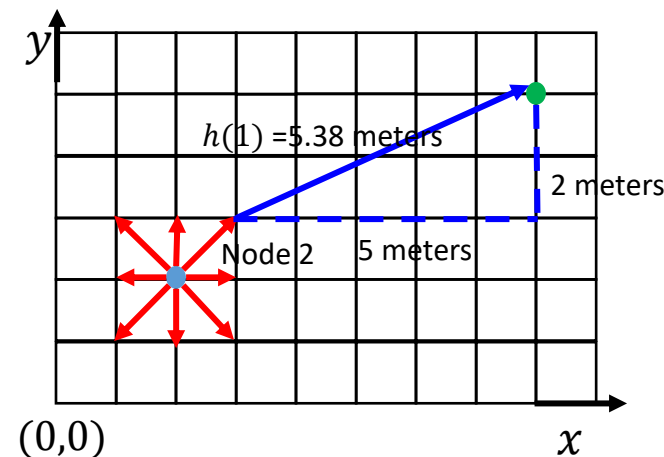
- $g(x, y)$  — this represents the **exact cost** of the path from the **starting node** to node  $(x, y)$
- $h(x, y)$  — this represents the heuristic **estimated cost** from node  $(x, y)$  to the goal node.
- $f(x, y)$  — cost of the neighboring node  $(x, y)$

8 neighboring node and the cost can be calculated as follows!

Node 2:

$$f(3,3) = g(3,3) + h(3,3) = 6.79 \text{ meters}$$

with  $g(3,3) = \sqrt{2}$  meter and  $h(3,3) = 5.38$  meters



● Start node

● Goal node

1 meter for one grid

# Path Planning using A Star Method

Definition of cost:

$$f(x, y) = g(x, y) + h(x, y)$$

- $g(x, y)$  — this represents the **exact cost** of the path from the **starting node** to node  $(x, y)$
- $h(x, y)$  — this represents the heuristic **estimated cost** from node  $(x, y)$  to the goal node.
- $f(x, y)$  — cost of the neighboring node  $(x, y)$

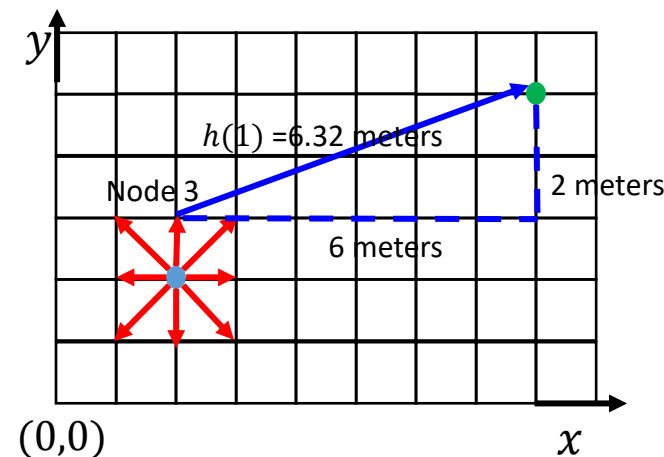
8 neighboring node and the cost can be calculated as follows!

Node 3:

$$f(2,3) = g(2,3) + h(2,3) = 7.32 \text{ meters}$$

with  $g(2,3) = 1 \text{ meter}$  and  $h(2,3) = 6.32 \text{ meters}$

Similar cost calculation method for other 5 nodes



● Start node

● Goal node

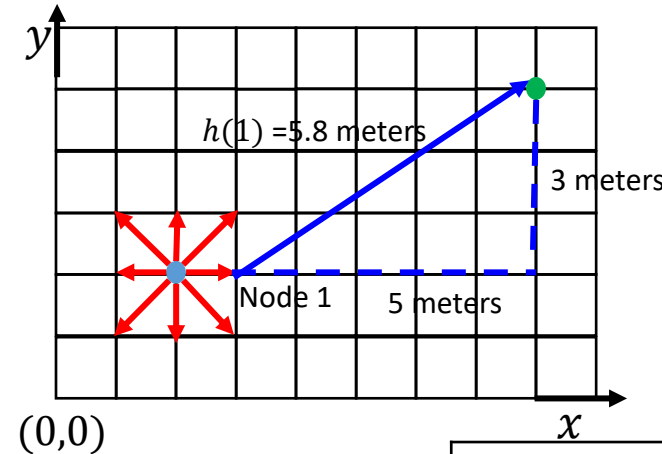
1 meter for one grid

# Path Planning using A Star Method

Definition of cost:

$$f(x, y) = g(x, y) + h(x, y)$$

- $g(x, y)$  — this represents the **exact cost** of the path from the **starting node** to node  $(x, y)$
- $h(x, y)$  — this represents the heuristic **estimated cost** from node  $(x, y)$  to the goal node.
- $f(x, y)$  — cost of the neighboring node  $(x, y)$



● Start node

● Goal node

1 meter for one grid

Node $(x, y)$	Node 1 $(x, y)$	Node 2 $(x, y)$	Node 3 $(x, y)$	Node 4 $(x, y)$	Node 5 $(x, y)$	Node 6 $(x, y)$	Node 7 $(x, y)$	Node 8 $(x, y)$
$g(x, y)$	1	1.414	1	1.414	1	1.414	1	1.414
$h(x, y)$	5.8	5.38	6.32	7.28	7.62	8.06	7.21	6.40
$f(x, y)$	6.8	6.79	7.32	8.694	8.62	9.474	8.21	7.814

# Path Planning using A Star Method

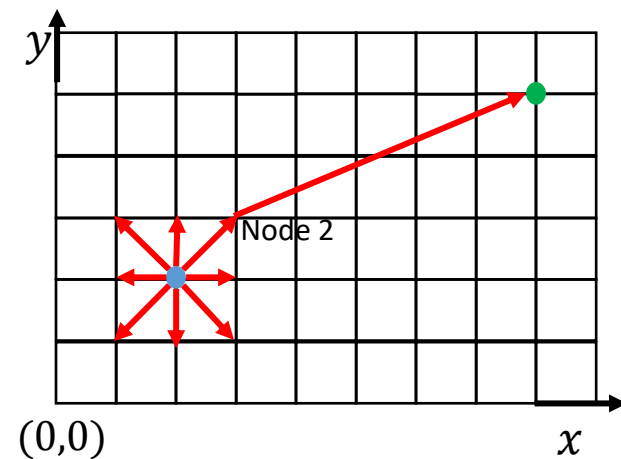
Definition of cost:

$$f(x, y) = g(x, y) + h(x, y)$$

- $g(x, y)$  — this represents the **exact cost** of the path from the **starting node** to node  $(x, y)$
- $h(x, y)$  — this represents the heuristic **estimated cost** from node  $(x, y)$  to the goal node.
- $f(x, y)$  — cost of the neighboring node  $(x, y)$

8 neighboring node and the cost can be calculated as follows!

Node 2 leads to smallest cost



● Start node

● Goal node

1 meter for one grid



# Calculate the cost of node

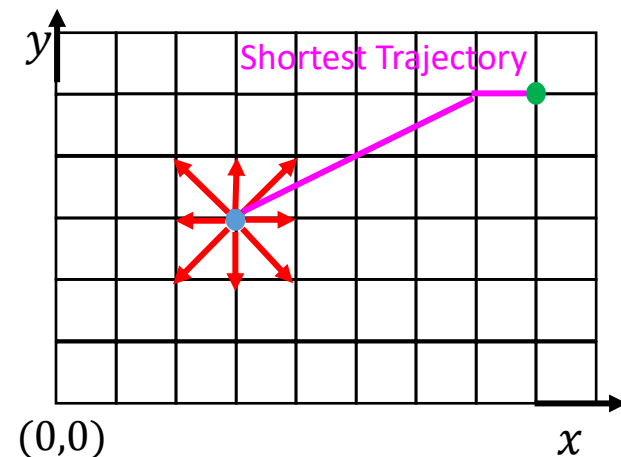
Definition of cost:

$$f(x, y) = g(x, y) + h(x, y)$$

- $g(x, y)$  — this represents the **exact cost** of the path from the **starting node** to node  $(x, y)$
- $h(x, y)$  — this represents the heuristic **estimated cost** from node  $(x, y)$  to the goal node.
- $f(x, y)$  — cost of the neighboring node  $(x, y)$

8 neighboring node and the cost can be calculated as follows!

Search from the neighbouring node with smallest cost until reaching the goal!



● Start node

● Goal node

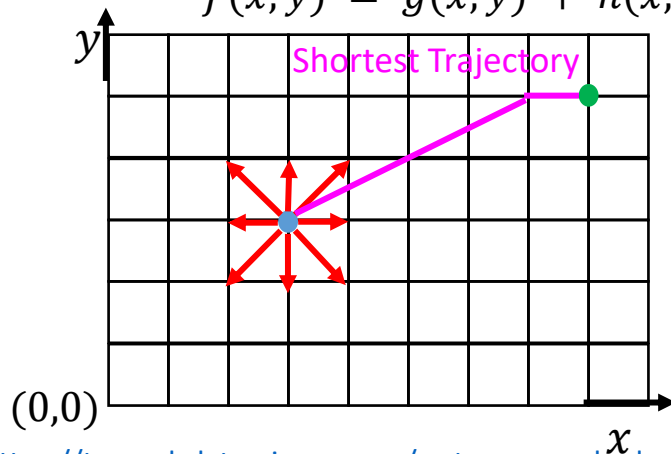
1 meter for one grid

# A star method example

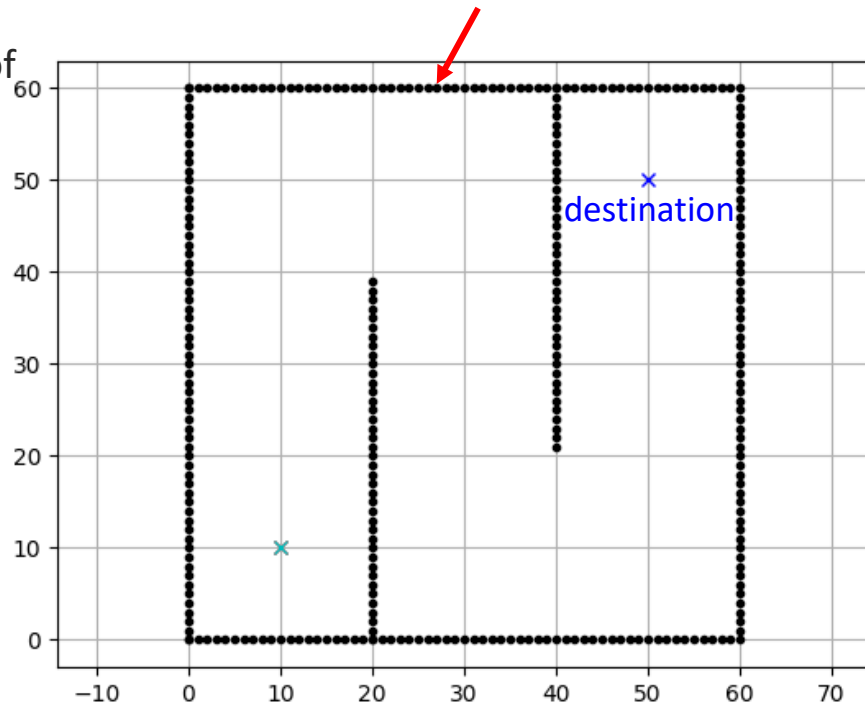
Each time A\* enters a node, it calculates the cost,  $f(n)$  ( $n$  being the neighboring node), to travel to all of the neighboring nodes, and then enters the node with the lowest value of  $f(n)$ .

These values we calculate using the following formula:

$$f(x, y) = g(x, y) + h(x, y)$$

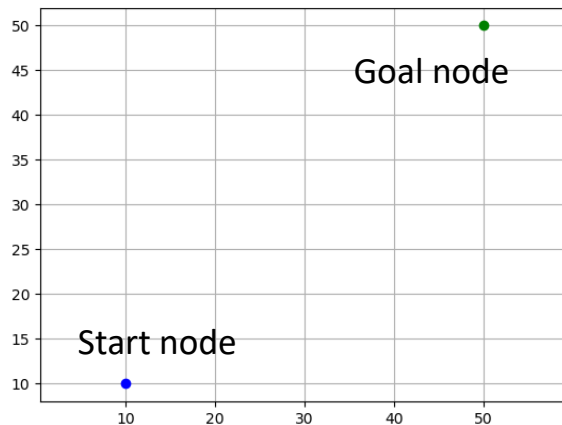


Wall (obstacles) cannot  
be touched!



Source: PythonRobotics

# Code: set up start and goal node



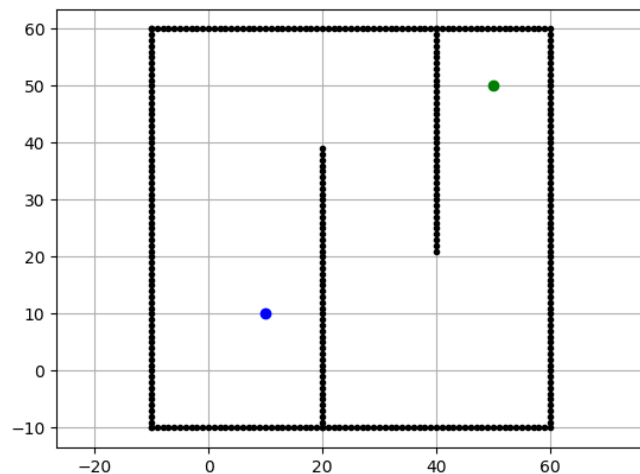
Set up the start and goal  
nodes using the code

```
# start and goal position  
sx = 10.0 # [m]  
sy = 10.0 # [m]  
gx = 50.0 # [m]  
gy = 50.0 # [m]  
grid_size = 2 # [m]
```

● Start node

● Goal node

# Code: set up obstacle



● Start node

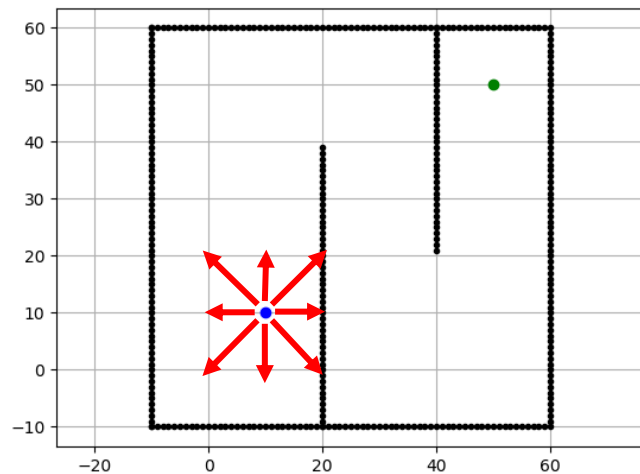
● Goal node

█ Obstacle (wall)

Set up the obstacle using the  
code

```
# set obstacle positions
ox, oy = [], []
for i in range(-10, 60): # draw the button border
    ox.append(i)
    oy.append(-10.0)
for i in range(-10, 60):
    ox.append(60.0)
    oy.append(i)
for i in range(-10, 61):
    ox.append(i)
    oy.append(60.0)
for i in range(-10, 61):
    ox.append(-10.0)
    oy.append(i)
for i in range(-10, 40):
    ox.append(20.0)
    oy.append(i)
for i in range(0, 40):
    ox.append(40.0)
    oy.append(60.0 - i)
```

# Code: neighboring node search



neighboring node search

```
def get_neighbouring_node(): # the cost of the surrounding 8 points
    # dx, dy, cost
    motion = [[1, 0, 1],
               [0, 1, 1],
               [-1, 0, 1],
               [0, -1, 1],
               [-1, -1, math.sqrt(2)],
               [-1, 1, math.sqrt(2)],
               [1, -1, math.sqrt(2)],
               [1, 1, math.sqrt(2)]]

    return motion
```

● Start node

● Goal node



Obstacle (wall)

# Code: cost calculation

Heuristic cost  $g(x, y)$  calculation

```
def calc_heuristic(n1, n2):  
    w = 1.0 # weight of heuristic  
    d = w * math.hypot(n1.x - n2.x, n1.y - n2.y)  
    return d
```

exact cost  $g(x, y)$  calculation

```
node = self.Node(current.x + self.motion[i][0],  
                 current.y + self.motion[i][1],  
                 current.cost + self.motion[i][2], c_id)
```

# Code: calculation of final path

```
def calc_final_path(self, goal_node, closed_set):  
    # generate final course  
    rx, ry = [self.calc_grid_position(goal_node.x, self.min_x)], [  
        self.calc_grid_position(goal_node.y, self.min_y)] # save the goal node as the first point  
    parent_index = goal_node.parent_index  
    while parent_index != -1:  
        n = closed_set[parent_index]  
        rx.append(self.calc_grid_position(n.x, self.min_x))  
        ry.append(self.calc_grid_position(n.y, self.min_y))  
        parent_index = n.parent_index  
  
    return rx, ry
```

