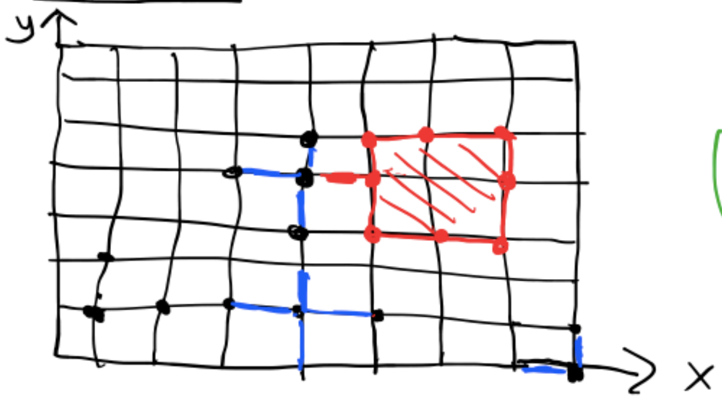
Monday, November 11, 2024 7:41 PM

## Chapter 5 : Grid search motion planning → Static obstacle avoidance




2d grid → Graph $G=(V,E)$

If node $q \in C_{free}$ add to graph

If node $q \in C_{obs}$ then discard

$C_{obs}$ obstacles

⇒

Graph $G=(V,E)$

All nodes $q \in C_{free}$

— Edge that connects 2 nodes/vertices

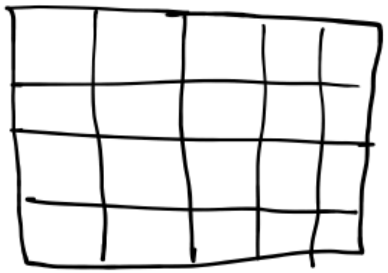
• Vertex / node

x,y coordinate → Vertex (V)

OneNote

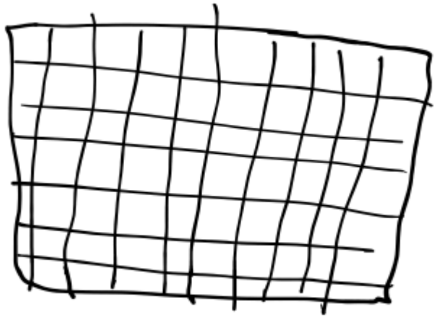
Home Insert Draw View Tell me Share

Monday, November 11, 2024 8:18 PM



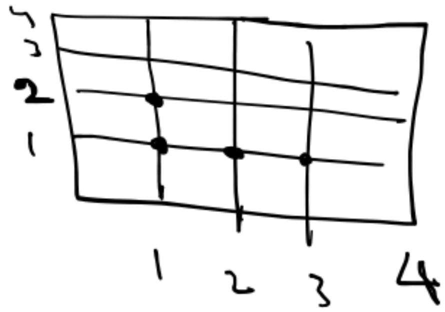
$\Delta x = 1$

$\Delta y = 1$



$\Delta x = 0.1$

$\Delta y = 0.1$




Large discretization

$\Rightarrow$ smaller graph

lower computation cost

$\Rightarrow$ less accuracy

small discretization

$\Rightarrow$ larger graph

much higher computation

higher accuracy

$\begin{cases} x = 1.7 \\ y = 2.6 \end{cases}$

$\begin{cases} x = 2.24 \\ y = 2.56 \end{cases}$

$\boxed{\text{tradeoff between accuracy \& cost}}$

SS

OneNote

Home Insert Draw View Tell me

Share

# Grid search method

Step 1: 2d Grid $\longrightarrow$ create graph

Step 2: Run a search method on the graph to find the best path

(1) depth first search
(2) breadth first search
(3) Dijkstra
(4) $A^*$

SS

Monday, November 11, 2024 8:28 PM

Grid based search methods have static obstacle avoidance built in (due $C_{obs}$ space)

Aside

Compare to differential flatness (DF)

Real time piecewise DF trajectory calculation

• Intermediate Point

— DF trajectory

OneNote

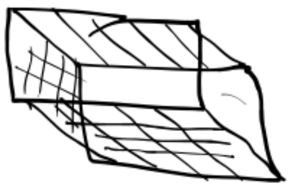
Home Insert Draw View Tell me

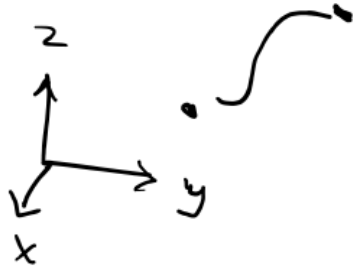
Share

Monday, November 11, 2024 8:34 PM

## Aside

## Quadrotor in 3d space






Grid search method in 3D => very large graph

Much higher computation

Differential Flatness in 3d

Lower computation

OneNote

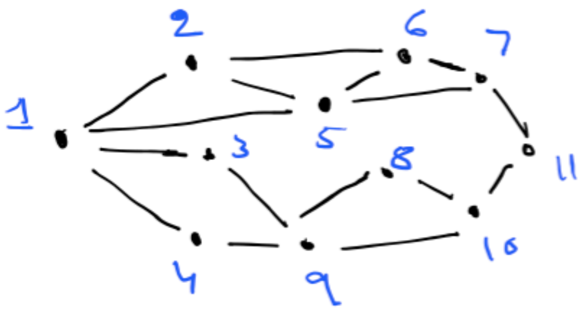
Home Insert Draw View Tell me Share

Monday, November 11, 2024 8:39 PM

**Step 2** : Graph search algorithms

Starting node $q_I$ , Goal node $q_G$

**Objective** : Find best path from $q_I$ to $q_G$




$q_I = 1$

$q_G = 11$

All edges are equal cost.

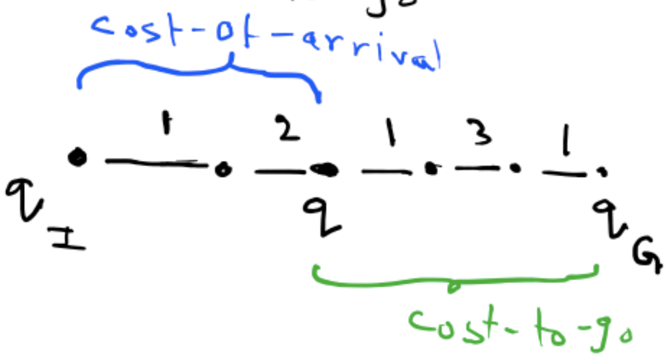
$1 \to 5 \to 7 \to 11$

Shortest path

SS

Define costs to go from one node to another node

- cost-of-arrival
- cost-to-go




$q_I$ start node

$q$ intermediate

$q_G$ goal node

<u>Cost-to-go</u>

The cost $c(q)$ for a node $q$ is the cost to go from $q$ to $q_G$

<u>Cost-of-arrival</u>

The cost $c(q)$ for a node $q$ is the cost to go from $q_I$ to $q$

$c(q) = 1 + 2 = 3$

OneNote

Home Insert Draw View Tell me

Share

Monday, November 11, 2024 8:51 PM

Depth first search

Breadth first search $\longrightarrow$ Finds the path with minimum cost-of-arrival

Dijkstra's method

✓ $A^*$ $\longrightarrow$ Finds the path with minimum cost-of-arrival + cost-to-go

SS