Autonomous Mobile Robots

Module 20: Unit III - Localization & Mapping
Introduction

Unit-III

LOCALIZATION AND MAPPING: Introduction - Bayes filter – Kalman Filter - Extended Kalman Filter - Information Filter - Histogram Filter - Particle Filter – Challenges of Localization- Map Representation- Probabilistic Map based Localization-Monte carlo localization- Landmark based navigation-Globally unique localization- Positioning beacon systems- Route based localization – Mapping - Metrical maps - Grid maps - Sector maps – Hybrid Maps – SLAM.

SLAM - Simultaneous Localization and Mapping

- Navigation : to go from one place to another autonomously
- Has 4 building blocks
- Perception: interpret sensor data
- Localization: determine position in environment
- Cognition: decide how to act to achieve goals
- Motion Control: modulate motor output

Recursive State Estimation

State: Collection of all aspects of a robot and its environment that can impact the future course of the robot

- State can change over time: dynamic state - position, velocity of moving robot, car
- State can be static - position of wall in a room
- Typical states we will deal with:
  - Mobile robot pose: position and orientation
  - For planar robot: pose is given by x,y,Ψ
  - Drone robot pose: (x,y,z) : 3 positions, (Φ,Θ,Ψ) : 3 orientations
  - Locations and features of surrounding objects in environment
  - Locations and velocities of moving objects and people
- State variable is represented by x

Environment Interaction

- Robot can influence the state of its environment through its actuators **- Control actions**
- Robot can gather information about the state through its sensors - **Sensor measurements**

- In practice, a robot continuously executes control action and measurements concurrently
- Control actions are represented by *u*
- Sensor measurements are represented by *z*
- Sensing measurements provide information about environment's state, hence it increases robot's knowledge
- Motion leads to a loss of knowledge due to uncertainty in actuation

Recursive State Estimation

Estimate current value of state based on previous value

Value of state at time 't' = $x(t) = x_t$

Value of state at previous time instant 't-1' = x(t-1) = x_{t-1}

x_{t-1} --> x_t : *update*

p(x): Random variable (RV) X

p(y): Random variable Y

Joint distribution of 2 RVs X & Y

$p(x,y) = p(X=x, Y=y)$

Independence: $p(x,y) = p(X=x, Y=y) = p(X=x).p(Y=y)=p(x)p(y)$

Recursive State Estimation - Conditional Probability

Suppose we know RV Y has value y, we want to know probability that RV X has value x

This makes use of Conditional probability *p(x|y) = p(X=x| Y=y)*

Assuming p(y)>0 $$p(x|y) = \frac{p(x,y)}{p(y)}$$

If X & Y are independent $$p(x|y) = \frac{p(x)p(y)}{p(y)} = p(x)$$

If X & Y are independent, it means that Y tells us nothing about X

Theorem of total probability: **Discrete case -** $$p(x) = \Sigma p(x|y)p(y)$$

Continuous case - $$p(x) = \int p(x|y)p(y)dy$$

Recursive State Estimation - Bayes Rule

Bayes Rule: $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$

$$p(x|y) = \frac{p(y|x)p(x)}{\int p(y|x')p(x')dx'} = \eta . p(y|x)p(x)$$

where $\eta = \frac{1}{p(y)}$

In general, *x* is a quantity we want to infer from *y*.

p(x) is called as the prior probability distribution

It captures knowledge we have regarding X prior to incorporating data Y

p(x|y) is called as the posterior probability distribution

Probabilistic Generative Laws

- Solution of state and measurements is governed by a probabilistic law
- Value of state *x* at time t : x_t
- May depend on all past states, measurements and controls
- Hence probability distribution may look like $p(x_t | x_{0:t-1}, z_{1:t-1}, u_{1:t})$

previous states **measurements** **controls**

- We may sometimes assume that x_{t-1} captures all the effects of $z_{1:t-1}, u_{1:t-1}$
- In that case: $p(x_t | x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t | x_{t-1}, u_t)$
- We call this as the **state transition probability**

Probabilistic Generative Laws

- Suppose we want to model the process by which measurements are generated

$$p(z_t | x_{0:t}, z_{1:t-1}, u_{1:t})$$

- Can sometimes assume

$$p(z_t | x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t | x_t)$$

- We call this as the measurement probability

Belief Distributions

- A belief reflects the robot's internal knowledge about the state of the environment
- Robot's true pose might be ***x=(2,3,1),*** but the robot will not know it
- Instead it must infer its pose from data

- Understand the difference between true state and internal belief
- Probabilistic robots represent beliefs through conditional probability distributions
- A belief distribution assigns a probability to each possible hypothesis with regard to true state

Belief Distributions

- Belief over a state variable x_t

$$bel(x_t) = p(x_t | z_{1:t}, u_{1:t})$$

Conditioned on all measurements **all controls**

- Can have another belief as

$$\overline{bel(x_t)} = p(x_t | z_{1:t-1}, u_{1:t})$$

- This probability distribution is referred to as a prediction, it predicts state at time t based on previous knowledge before incorporating measurement at time t

Autonomous Mobile Robots

Module 21: Bayes Filter

Bayes filter

- Input to Bayes filter: $bel(x_{t-1}), u_t, z_t$

Output of Bayes filter: $bel(x_t)$

- Bayes filter has 2 steps:
- **Prediction step:** processes control
- **Measurement update:** uses sensor information
- Bayes filter is the most general algorithm for calculating beliefs
- Algorithm calculates belief distribution from measurement and control data
- Bayes filter is recursive, that is *bel*(x_t) at time t is calculated from belief *bel*(x_{t-1})
- Input is belief at time t-1, most recent control u_t and most recent measurement z_t, output is belief at time t, update rule is applied recursively.

Bayes filter - update rule algorithm

1: **Algorithm Bayes_filter**$(bel(x_{t-1}), u_t, z_t)$**:**
2: for all x_t do
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx$
4: $bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$
5: endfor
6: return $bel(x_t)$

Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Bayes filter - update rule algorithm

- Algorithm is a 2 step process
- Step 1 : Line 3 processes control

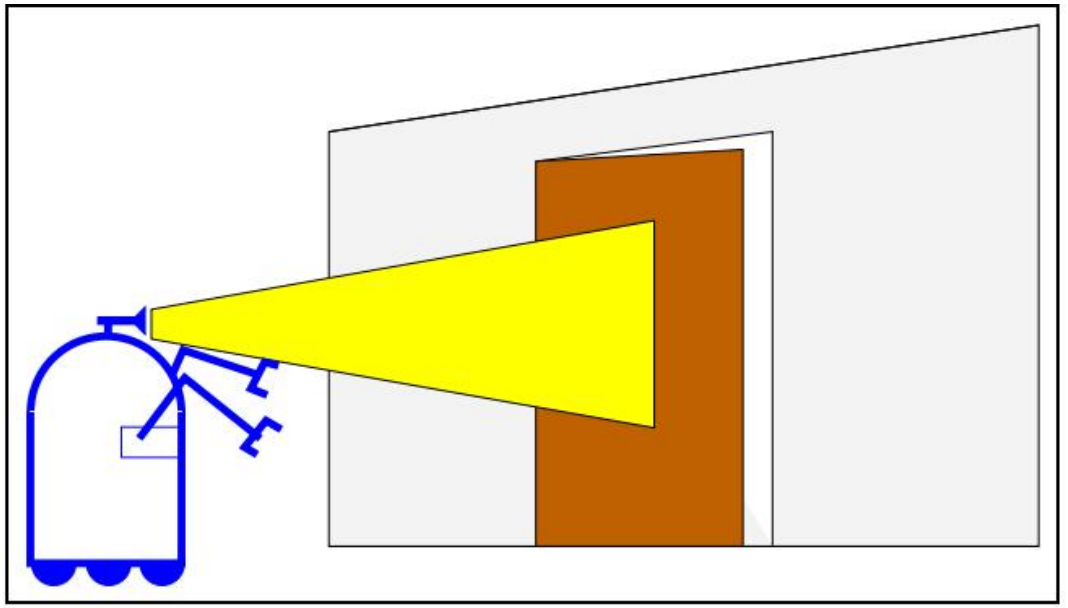
$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx$$

- Step 2 : Line 4 performs measurement update

$$bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$$

- η is a normalization constant to ensure that probability distribution satisfies typical rules
- bel(x_t) --> bel(x_{t-1}) --> bel(x_{t-2}) --....-->bel(x_1)-->bel(x_0)

Bayes filter - Example



- Example Scenario:
- Robot estimates state of a door using its camera
- Door can be in open state or closed state
- Assume robot does not know state of the door initially
- Assume equal prior probability to possible door states

$$bel(X_0 = open) = 0.5$$

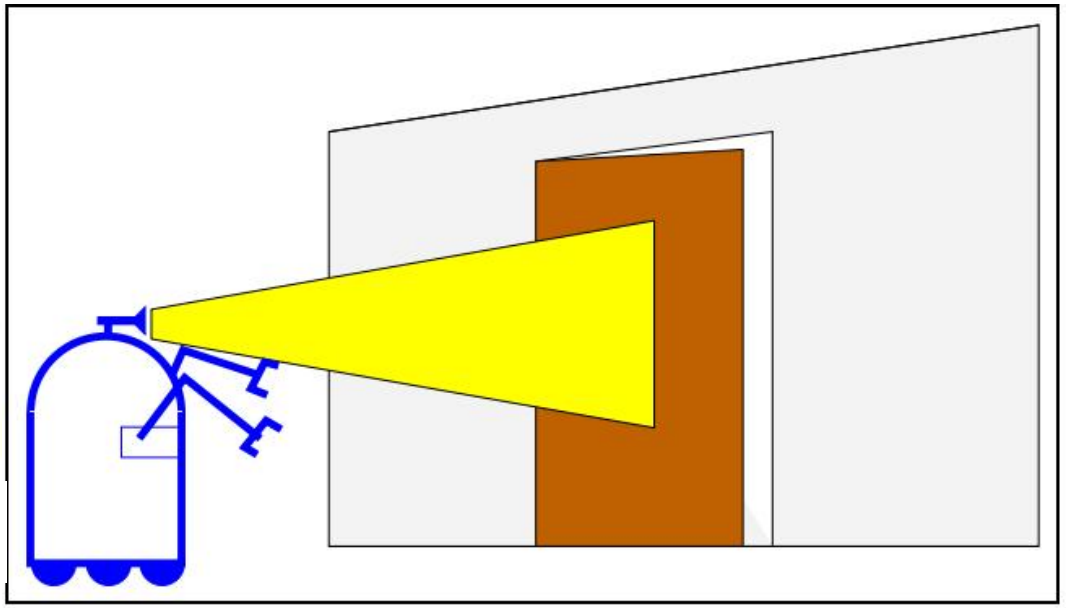
$$bel(X_0 = closed) = 0.5$$

Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Bayes filter - Example

Assume robot sensors are noisy

$$p(Z_t = \textbf{sense\_open} \mid X_t = \textbf{is\_open}) = 0.6$$
$$p(Z_t = \textbf{sense\_closed} \mid X_t = \textbf{is\_open}) = 0.4$$
$$p(Z_t = \textbf{sense\_open} \mid X_t = \textbf{is\_closed}) = 0.2$$
$$p(Z_t = \textbf{sense\_closed} \mid X_t = \textbf{is\_closed}) = 0.8$$



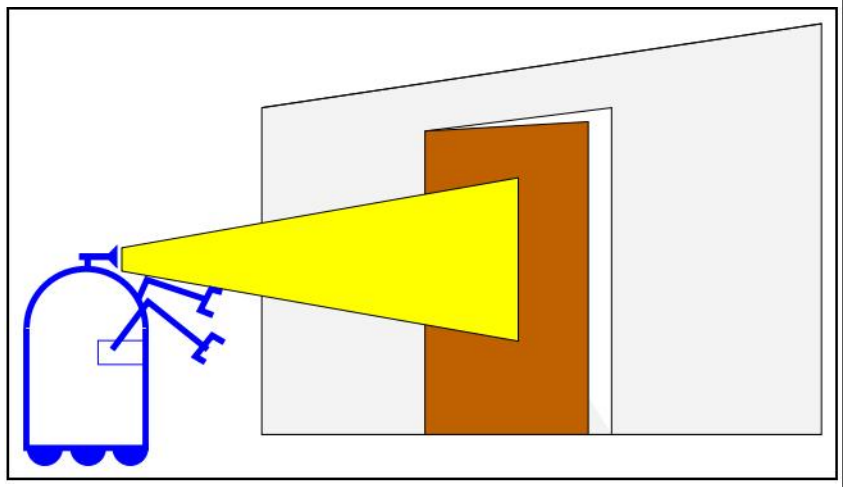
Based on sensor probabilities we can infer:

Sensor more reliable in detecting closed door

When door is open, error probability is 0.4

Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Bayes filter - Example



$$p(X_t = \textbf{is\_open} \mid U_t = \textbf{push}, X_{t\_1} = \textbf{is\_open}) = 1$$
$$p(X_t = \textbf{is\_closed} \mid U_t = \textbf{push}, X_{t\_1} = \textbf{is\_open}) = 0$$
$$p(X_t = \textbf{is\_open} \mid U_t = \textbf{push}, X_{t\_1} = \textbf{is\_closed}) = 0.8$$
$$p(X_t = \textbf{is\_closed} \mid U_t = \textbf{push}, X_{t\_1} = \textbf{is\_closed}) = 0.2$$

$$p(X_t = \textbf{is\_open} \mid U_t = \textbf{do\_nothing}, X_{t\_1} = \textbf{is\_open}) = 1$$
$$p(X_t = \textbf{is\_closed} \mid U_t = \textbf{do\_nothing}, X_{t\_1} = \textbf{is\_open}) = 0$$
$$p(X_t = \textbf{is\_open} \mid U_t = \textbf{do\_nothing}, X_{t\_1} = \textbf{is\_closed}) = 0$$
$$p(X_t = \textbf{is\_closed} \mid U_t = \textbf{do\_nothing}, X_{t\_1} = \textbf{is\_closed}) = 1$$

Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Bayes filter - update rule algorithm

1: **Algorithm Bayes_filter**$(bel(x_{t-1}), u_t, z_t)$**:**
2: for all x_t do
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx$
4: $bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$
5: endfor
6: return $bel(x_t)$

Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Bayes filter example

- Suppose at time t, the robot takes no control action but it senses an open door.
- Use control u1 = do nothing, and the measurement **sense open** as input

$$\overline{bel}(x_1) = \int p(x_1 \mid u_1, x_0)\, bel(x_0)\, dx_0$$

$$= \sum_{x_0} p(x_1 \mid u_1, x_0)\, bel(x_0)$$

$$= p(x_1 \mid U_1 = \textbf{do\_nothing}, X_0 = \textbf{is\_open})\, bel(X_0 = \textbf{is\_open})$$
$$+ p(x_1 \mid U_1 = \textbf{do\_nothing}, X_0 = \textbf{is\_closed})\, bel(X_0 = \textbf{is\_closed})$$

Bayes filter example

$$\overline{bel}(X_1 = \textbf{is\_open})$$

$$= \quad p(X_1 = \textbf{is\_open} \mid U_1 = \textbf{do\_nothing}, X_0 = \textbf{is\_open})\ bel(X_0 = \textbf{is\_open})$$
$$+p(X_1 = \textbf{is\_open} \mid U_1 = \textbf{do\_nothing}, X_0 = \textbf{is\_closed})\ bel(X_0 = \textbf{is\_closed})$$

$$= \quad 1 \cdot 0.5 + 0 \cdot 0.5 \quad = \quad 0.5$$

$$\overline{bel}(X_1 = \textbf{is\_closed})$$

$$= \quad p(X_1 = \textbf{is\_closed} \mid U_1 = \textbf{do\_nothing}, X_0 = \textbf{is\_open})\ bel(X_0 = \textbf{is\_open})$$
$$+p(X_1 = \textbf{is\_closed} \mid U_1 = \textbf{do\_nothing}, X_0 = \textbf{is\_closed})\ bel(X_0 = \textbf{is\_closed})$$

$$= \quad 0 \cdot 0.5 + 1 \cdot 0.5 \quad = \quad 0.5$$

Bayes filter example

- Incorporate the measurement

$$bel(x_1) \quad = \quad \eta\, p(Z_1 = \textbf{sense\_open} \mid x_1)\, \overline{bel}(x_1)$$

$$bel(X_1 = \textbf{is\_open})$$
$$= \quad \eta\, p(Z_1 = \textbf{sense\_open} \mid X_1 = \textbf{is\_open})\, \overline{bel}(X_1 = \textbf{is\_open})$$
$$= \quad \eta\, 0.6 \cdot 0.5 \;=\; \eta\, 0.3$$

$$bel(X_1 = \textbf{is\_closed})$$
$$= \quad \eta\, p(Z_1 = \textbf{sense\_open} \mid X_1 = \textbf{is\_closed})\, \overline{bel}(X_1 = \textbf{is\_closed})$$
$$= \quad \eta\, 0.2 \cdot 0.5 \;=\; \eta\, 0.1$$

$$\eta \quad = \quad (0.3 + 0.1)^{-1} \;=\; 2.5$$

$$bel(X_1 = \textbf{is\_open}) \quad = \quad 0.75$$
$$bel(X_1 = \textbf{is\_closed}) \quad = \quad 0.25$$

Bayes filter example

for $u_2 = \textbf{push}$ and $z_2 = \textbf{sense\_open}$

$$\begin{aligned}
\overline{bel}(X_2 = \textbf{is_open}) &= 1 \cdot 0.75 + 0.8 \cdot 0.25 = 0.95 \\
\overline{bel}(X_2 = \textbf{is_closed}) &= 0 \cdot 0.75 + 0.2 \cdot 0.25 = 0.05
\end{aligned}$$

$$\begin{aligned}
bel(X_2 = \textbf{is_open}) &= \eta\, 0.6 \cdot 0.95 \approx 0.983 \\
bel(X_2 = \textbf{is_closed}) &= \eta\, 0.2 \cdot 0.05 \approx 0.017
\end{aligned}$$

Autonomous Mobile Robots

Module 22: Kalman Filter

Kalman filter

- Gaussian filter
  - Kalman filter -> Linear Gaussian
  - Extended Kalman filter -> Linearized Gaussian
  - Information filter
- Typical Gaussian Probability Density Function

Gaussian Distribution for Multivariate case

- We have n state variables X_1, X_2, ..., X_n
- We have m control inputs: u_1, u_2, ..., u_m

$$X = \begin{bmatrix} X_1 \\ X_2 \\ . \\ . \\ X_n \end{bmatrix}$$

- We need a Normal Distribution, but multivariate

$$p(X) = (det(2\pi\Sigma))^{-1/2}.exp(\frac{-1}{2})(X-\mu)^T\Sigma^{-1}(X-\mu)$$

$$x : \mathbb{R}^{n\mathbf{x}1} \qquad (x-\mu)^T : \mathbb{R}^{1\mathbf{x}n}$$

$$\mu : \mathbb{R}^{n\mathbf{x}1} \qquad \Sigma : \mathbf{covariance\ matrix} : \mathbb{R}^{n\mathbf{x}n}$$

$$\Sigma^{-1} : \mathbb{R}^{n\mathbf{x}n}$$

$$(X-\mu)^T\Sigma^{-1}(X-\mu) : \mathbb{R}^{1\mathbf{x}n}.\mathbb{R}^{n\mathbf{x}n}.\mathbb{R}^{n\mathbf{x}1}$$

Scalar Value

Kalman Filter

- Kalman filter is a Linear Gaussian Filter, introduced by Rudolf Kalman in 1950
- It performs belief computation for continuous states (not discrete)
- At time t, belief is represented by a mean at time t and covariance at time t:
  μ_t, Σ_t

Assumptions:

1. Next state probability must be a linear function $x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$

$$x_t = \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ . \\ . \\ x_{n,t} \end{bmatrix} \qquad u_t = \begin{bmatrix} u_{1,t} \\ u_{2,t} \\ . \\ . \\ u_{m,t} \end{bmatrix}$$

A_t is a matrix of size nxn
B_t is a matrix of size nxm
ϵ_t is Gaussian noise

Kalman Filter - Assumptions

Assumptions:

2. Measurement probability must be linear with added noise

$$z_t = C_t x_t + \delta_t$$

$$C_t \text{ is a matrix}$$

3. Initial belief bel(X_0) must be normal distributed

$$\text{Mean of belief} : \mu_0$$
$$\text{Covariance of belief} : \Sigma_0$$

Kalman Filter - Algorithm

Inputs : $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$ $\quad bel(x_{t-1}) : \mu_{t-1}, \Sigma_{t-1}$

Output : $bel(x_t) : \mu_t, \Sigma_t$

Algorithm Kalman_filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**

$\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$

$\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$

$K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$

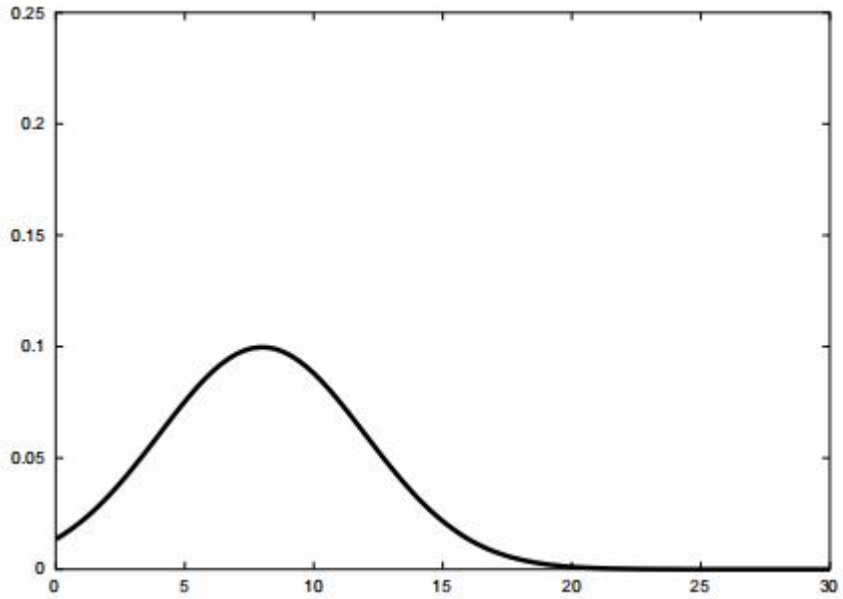
$\mu_t = \bar{\mu}_t + K_t(z_t - C_t\ \bar{\mu}_t)$

$\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$

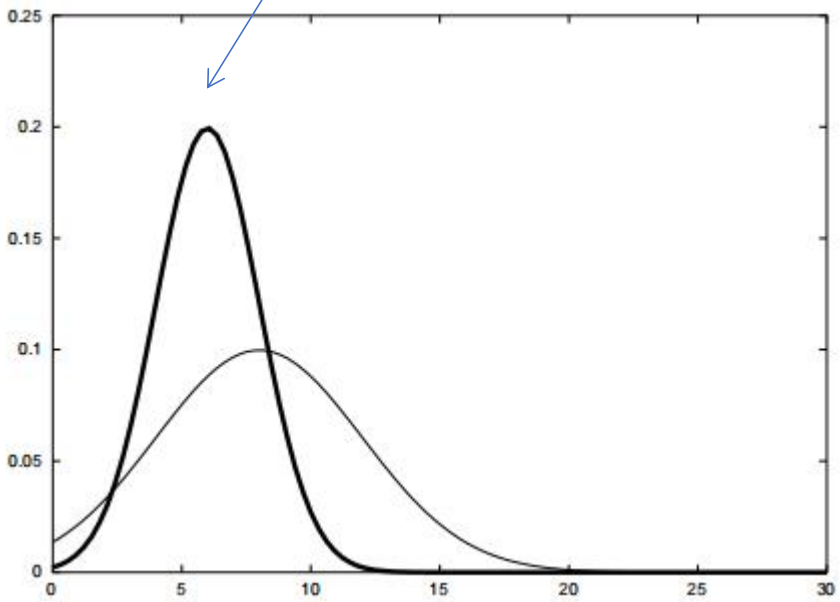
return μ_t, Σ_t

Kalman Filter - Algorithm

Initial Belief, prior



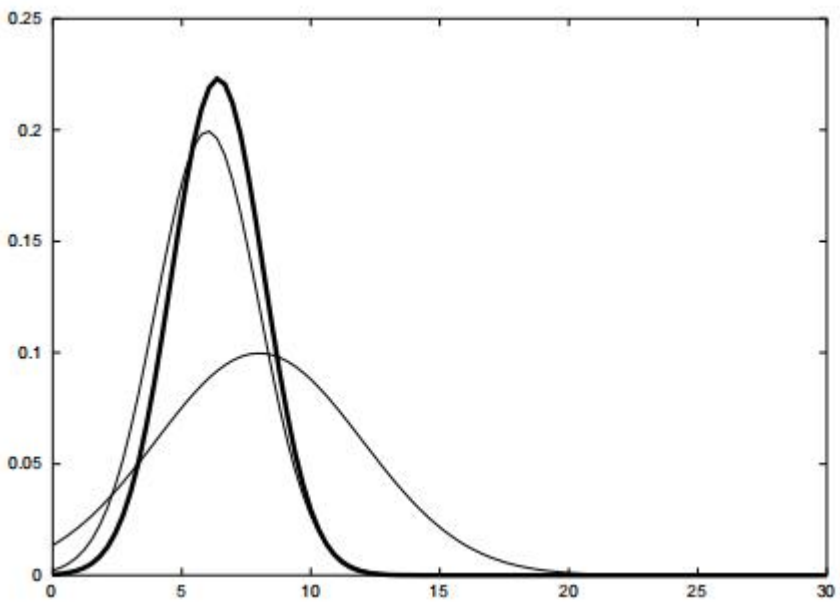
Measurement



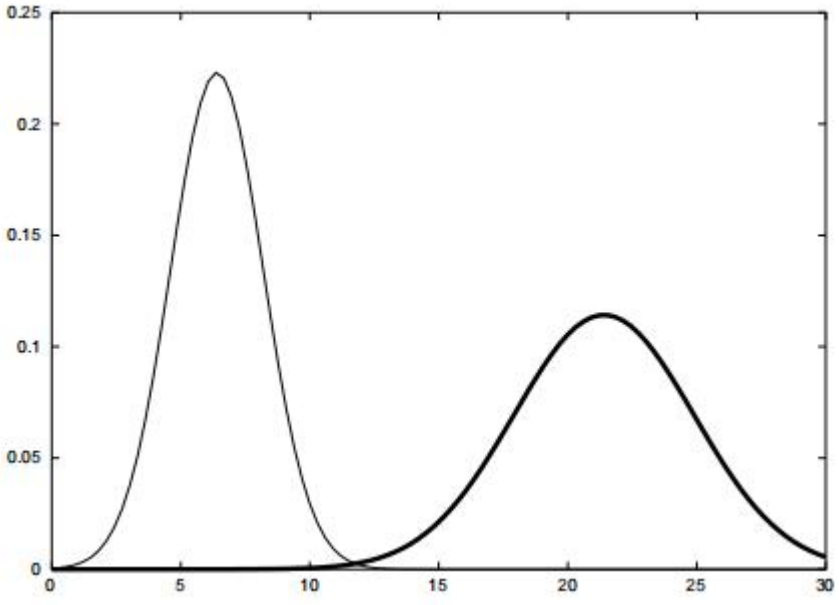
Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Kalman Filter - Algorithm

Belief after integrating measurement using KF



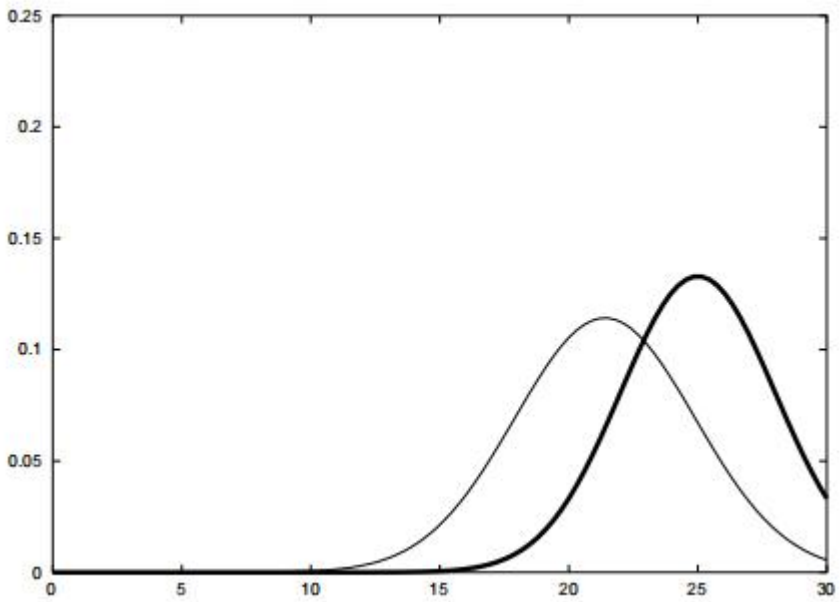
Belief after motion to the right



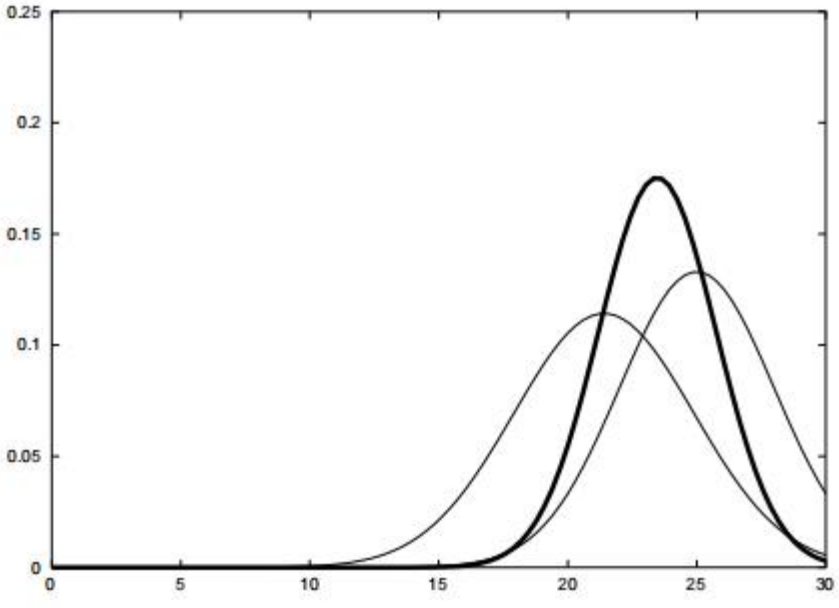
Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Kalman Filter - Algorithm

New measurement with associated uncertainty



Resulting belief



Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Autonomous Mobile Robots

Module 23: Extended Kalman Filter

Extended Kalman filter

- Linearized Gaussian filter
- Relaxes assumptions of linear state transitions
- Relaxes assumptions of linear measurements

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t$$
$$z_t = h(x_t) + \delta_t \ .$$

g : replaces matrices A_t, B_t

h : replaces matrix C_t

- Gaussian aspects are retained, but linear aspects are relaxed
- belief bel(x_t) represented by mean and covariance

Extended Kalman filter - algorithm

1: **Algorithm Extended_Kalman_filter**$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3: $\bar{\Sigma}_t = G_t \, \Sigma_{t-1} \, G_t^T + R_t$
4: $K_t = \bar{\Sigma}_t \, H_t^T (H_t \, \bar{\Sigma}_t \, H_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6: $\Sigma_t = (I - K_t \, H_t) \, \bar{\Sigma}_t$
7: return μ_t, Σ_t

- Input to EKF filter: $bel(x_{t-1}), u_t, z_t$ $bel(x_{t-1})$ given by μ_{t-1}, Σ_{t-1}

Output of EKF filter: $bel(x_t)$ $bel(x_t)$ given by μ_t, Σ_t

Lines 2&3 are predictions, Lines 5&6 are measurement updates

Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Extended Kalman filter - algorithm

1: **Algorithm Extended_Kalman_filter**$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**

2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$

3: $\bar{\Sigma}_t = G_t \, \Sigma_{t-1} \, G_t^T + R_t$

4: $K_t = \bar{\Sigma}_t \, H_t^T (H_t \, \bar{\Sigma}_t \, H_t^T + Q_t)^{-1}$

5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$

6: $\Sigma_t = (I - K_t \, H_t) \, \bar{\Sigma}_t$

7: return μ_t, Σ_t

	Kalman filter	EKF
state prediction (Line 2)	$A_t \, \mu_{t-1} + B_t \, u_t$	$g(u_t, \mu_{t-1})$
measurement prediction (Line 5)	$C_t \, \bar{\mu}_t$	$h(\bar{\mu}_t)$

Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Autonomous Mobile Robots

Module 24: Information Filter

Information filter

- Information filter is a Gaussian filter which is a dual of a Kalman filter
- Information filter represents belief by a Gaussian
- Instead of representing beliefs by mean and covariance, the information filter uses canonical representation based on information matrix and information vector
- This leads to different update equation

$$\text{Information matrix} : \Omega = \Sigma^{-1} : \text{inverse of the covariance matrix}$$
$$\text{Information vector} : \xi = \Sigma^{-1}\mu$$

Information filter - algorithm

Algorithm Information_filter$(\xi_{t-1}, \Omega_{t-1}, u_t, z_t)$**:**

$\bar{\Omega}_t = (A_t \, \Omega_{t-1}^{-1} \, A_t^T + R_t)^{-1}$

$\bar{\xi}_t = \bar{\Omega}_t (A_t \, \Omega_{t-1}^{-1} \, \xi_{t-1} + B_t \, u_t)$

$\Omega_t = C_t^T \, Q_t^{-1} \, C_t + \bar{\Omega}_t$

$\xi_t = C_t^T \, Q_t^{-1} \, z_t + \bar{\xi}_t$

return ξ_t, Ω_t

- Information filter is a Gaussian filter which is a dual of a Kalman filter

Autonomous Mobile Robots

Module 25: Histogram Filter, Particle filter

Non-parametric filters

- These filters do not rely on a fixed function form for probability density function
- Histogram filters and Particle filters come under this category
- Posterior probability is approximated by a finite number of values
- The continuous state space is decomposed into a finite number of regions
- The number of parameters which are used to approximate the posterior can be varied
- The quality of approximation depends on the number of parameters uesd to represent the posterior

Histogram filter - example

- Age that a person can have is represented as a continuous state space
- Ex: 1.5, 2.5, 23.8
- Examples of a discrete space:
- **Coin Flip**: H,T: 2 outcomes
- **Gear Position:** N, R, 1, 2, 3, 4: 6 outcomes
- The posterior distribution is approximated by a finite number of values
- When this is applied to continous state spaces, we call it the histogram filter
- When this is applied to discrete state spaces, we call it the Discrete Bayes Filter

Histogram filter - example

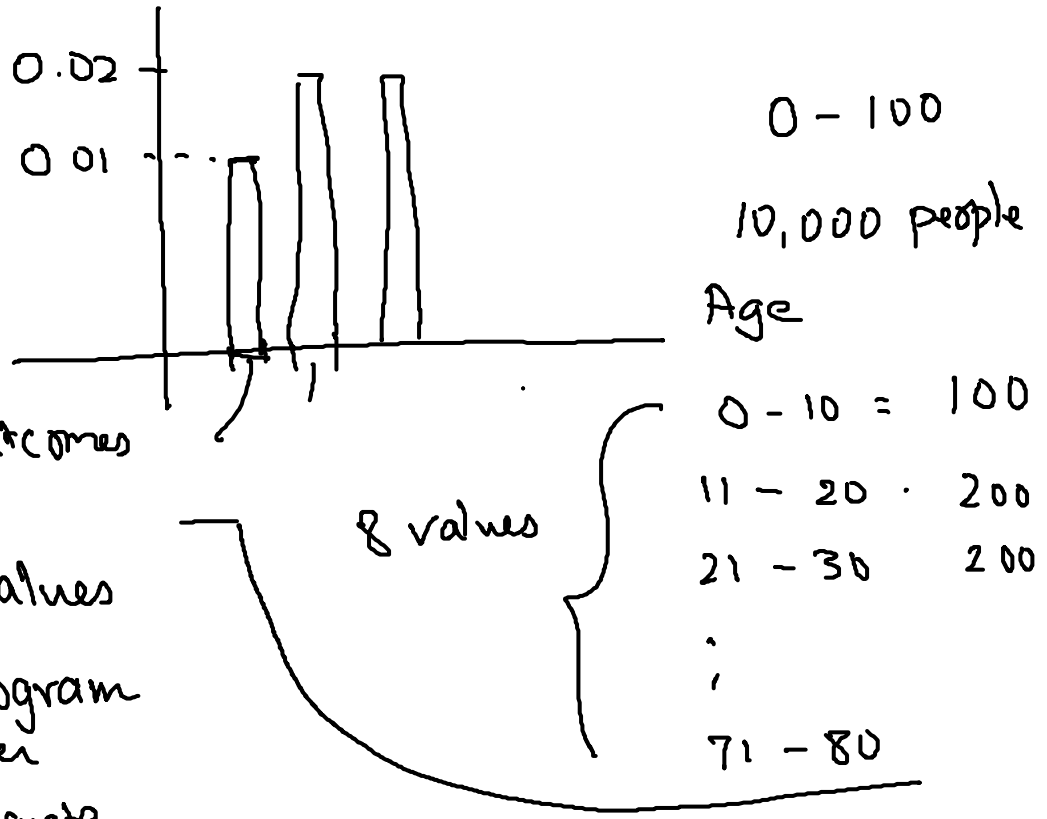
Age that a person can have is a continuous state space

Ex 1.5, 2.5, 23.8

Discrete space

Ex: Coin flip H, T : 2 outcomes

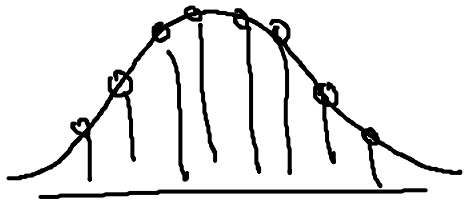
gear position N, R, 1, 2, 3, 4 : 6 outcomes



Approximate posteriors by finite number of values

When applied to continuous state spaces : histogram filter

When " " discrete " " : discrete Bayes filter



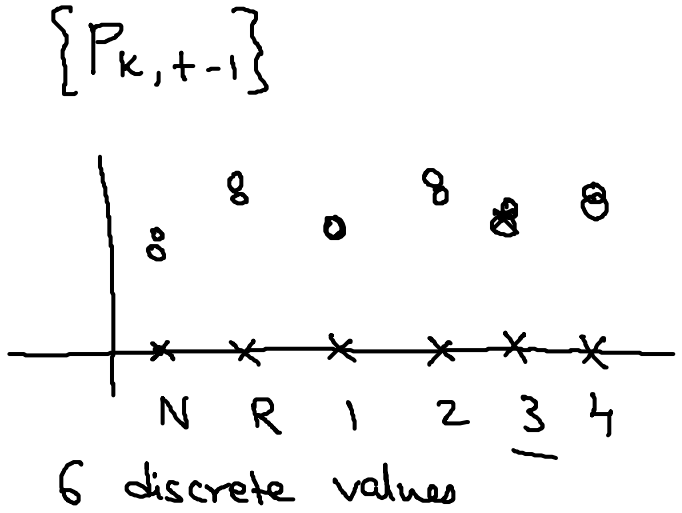
Histogram filter example

Algorithm Discrete_Bayes_filter($\{p_{k,t-1}\}, u_t, z_t$):

1: **Algorithm Discrete_Bayes_filter**($\{p_{k,t-1}\}, u_t, z_t$):
2: for all k do
3: $\bar{p}_{k,t} = \sum_i p(X_t = x_k \mid u_t, X_{t-1} = x_i)\, p_{i,t-1}$
4: $p_{k,t} = \eta\, p(z_t \mid X_t = x_k)\, \bar{p}_{k,t}$
5: endfor
6: return $\{p_{k,t}\}$

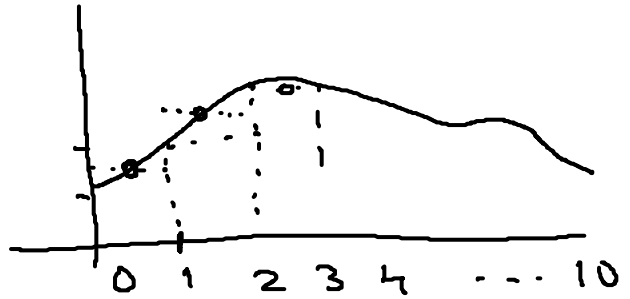
Prediction
measurement
bel
bel
Output
$p(x_k \mid u_t, x_i)$ ← line 3
$p(z_t \mid x_k)$ ← line 4

Discrete State Space



Continuous State Space

Break state space into regions and maintain finite number of probability values



Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Histogram filter

- The state space is decomposed into finite number of regions and the posterior is represented by a histogram in this case
- This is also called a Discrete Bayes Filter

Algorithm Discrete_Bayes_filter$(\{p_{k,t-1}\}, u_t, z_t)$**:**

for all k do

$\bar{p}_{k,t} = \sum_i p(X_t = x_k \mid u_t, X_{t-1} = x_i)\, p_{i,t-1}$

$p_{k,t} = \eta\, p(z_t \mid X_t = x_k)\, \bar{p}_{k,t}$

endfor

return $\{p_{k,t}\}$

Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Histogram filter - example

- The posterior is represented as a single cumulative probability value for each region
- The random variable X_t can only take so many values: $\mathbf{x}_{1,t}$, $\mathbf{x}_{2,t}$, …, $\mathbf{x}_{k,t}$

$$\text{range}(X_t) = \mathbf{x}_{1,t} \cup \mathbf{x}_{2,t} \cup \ldots \mathbf{x}_{K,t}$$

- Posterior becomes a piecewise constant PDF with uniform probability to each state $\mathbf{x}_t$ within each region $\mathbf{x}_{k,t}$
- **Mean state:** $\hat{x}_{k,t}$

$$p(z_t \mid \mathbf{x}_{k,t}) \approx p(z_t \mid \hat{x}_{k,t})$$

$$p(\mathbf{x}_{k,t} \mid u_t, \mathbf{x}_{i,t-1}) = \eta \, |\mathbf{x}_{k,t}| \, p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1})$$

Histogram filter - example

1: **Algorithm Discrete_Bayes_filter**($\{p_{k,t-1}\}, u_t, z_t$):
2: for all k do
3: $\bar{p}_{k,t} = \sum_i p(X_t = x_k \mid u_t, X_{t-1} = x_i)\, p_{i,t-1}$
4: $p_{k,t} = \eta\, p(z_t \mid X_t = x_k)\, \bar{p}_{k,t}$
5: endfor
6: return $\{p_{k,t}\}$

Prediction → 3:
measurement → 4:

$\overline{bel}$

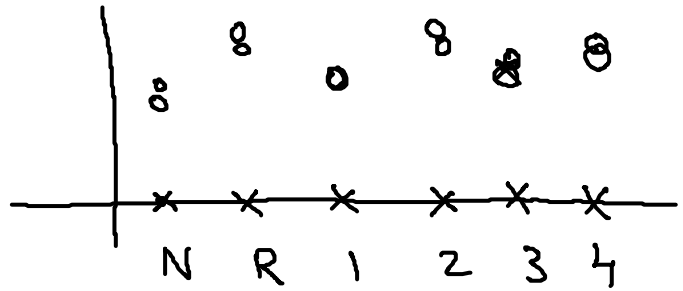
bel

Output

$p(x_k \mid u_t, x_i)$ ← line 3

$p(z_t \mid x_k)$ ← line 4

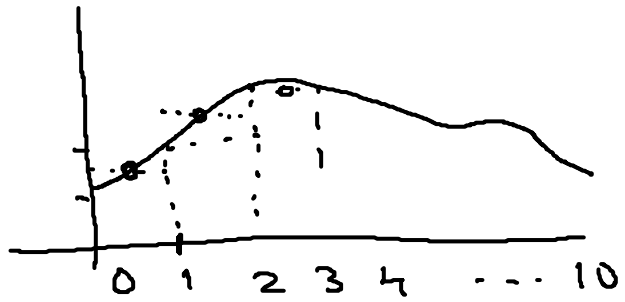
$\{p_{k,t-1}\}$



6 discrete values

0 - 1 : p

1 - 2 : p



Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Particle filter

- A particle filter provides a flexible method to approximate a function
- This flexibility gives it the capability to deal with arbitrary distributions
- The posterior is approximated by a finite number of parameters
- The key idea is that the posterior bel(xt) is represented by a set of random state samples drawn from the distribution
- The samples of the posterior distribution are called particles are denoted as

$$\mathcal{X}_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}$$

- Each particle $\mathbf{x_t^{[m]}}$ is a hypothesis of what the true world state may be at time t, M is the total number of particles

Particle filter - illustrative example for navigation

https://www.youtube.com/watch?v=NrzmH_yerBU&t=799s&pp=ygUPcGFydGljbGUgZmlsdGVy

Particle filter

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

for $m = 1$ to M do

  sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$

  $w_t^{[m]} = p(z_t \mid x_t^{[m]})$

  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ to M do

  draw i with probability $\propto w_t^{[i]}$

  add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Particle filter

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

for $m = 1$ to M do

  sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$ ← prediction, control

  $w_t^{[m]} = p(z_t \mid x_t^{[m]})$ ← measurement update (importance weight)

  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ to M do

  draw i with probability $\propto w_t^{[i]}$

  add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

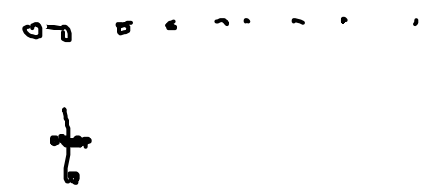
} Resampling. — Change particle distribution based on weights

Input $\mathcal{X}_{t-1}, u_t, z_t$

Output $\mathcal{X}_t$

$\bar{\mathcal{X}}_t$: temporary particle set

$x_t = 0.5$



Reproduced from: Sebastian Thrun, Wolfram Burgard, Dieter Fox, —Probabilistic Robotics, MIT Press, 2005

Autonomous Mobile Robots

Module 26: Challenges of Localization,
Map Representation
Probabilistic Map-based Localization
Monte Carlo Localization

Challenges of Localization

- **Localization:** Need the robot to know its exact position, indoors and outdoors
- GPS network provides location with accuracy to within several meters, not suitable for mobile robots
- It also does not work indoors
- Robot may also need to know its relative position to objects in the environment
- Robot may need to build a map and then identify the position of the robot relative to the map

Challenges of Localization

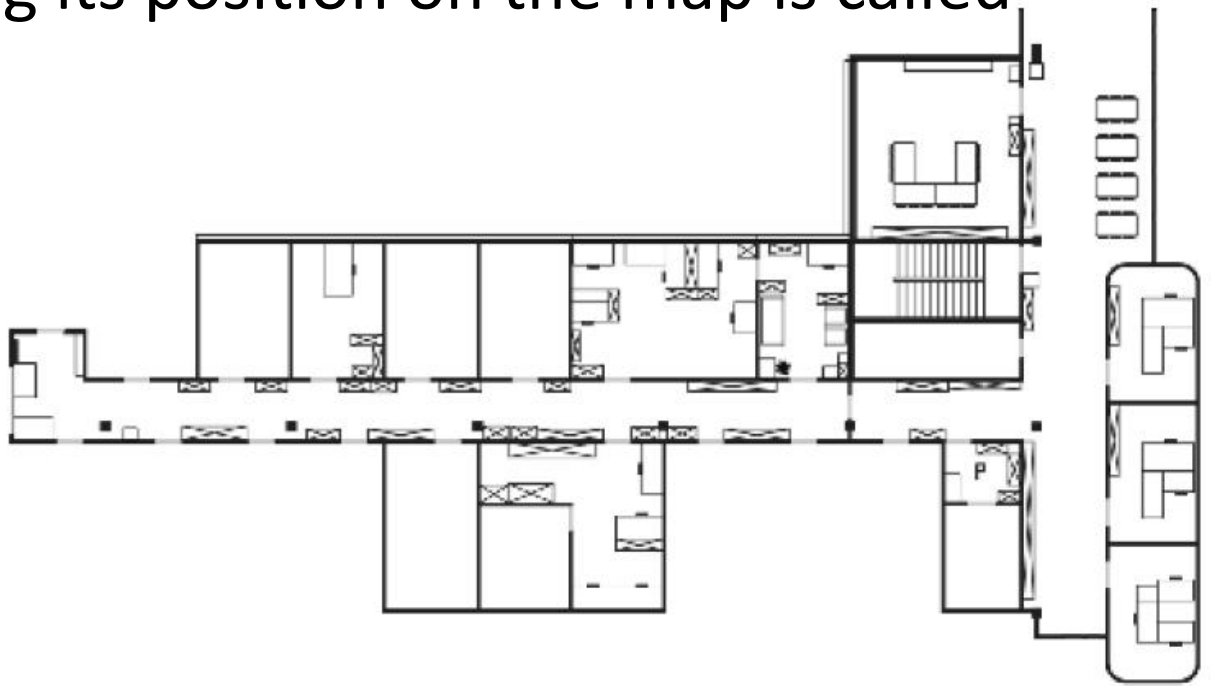
- Robot's sensors and actuators play a major role in localization
- But both components have issues
- Sensor has noise and aliasing issues
- Aliasing refers to non-uniqueness of sensor readings
- There is a many to one mapping from environmental states to robot's perceptual inputs

Challenges of Localization

- Robot's sensors and actuators play a major role in localization
- But both components have issues
- Sensor has noise and aliasing issues
- Aliasing refers to non-uniqueness of sensor readings
- There is a many to one mapping from environmental states to robot's perceptual inputs
- Actuator noise: An action taken by a robot may have several possible results.
- Moving tends to increase the uncertainty of a mobile robot
- To improve robot performance in the presence of actuator noise, we can try to model the source of the error and account for it

Map Representation

- Representation of the environment of a robot is called Map representation
- Representation of its belief regarding its position on the map is called as Belief representation



Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Map Representation

- 3 relationships must be understood when choosing a map representation

  1) Precision of the map must match the precision with which the robot needs to achieve goals

  2) Precision of the map and type of features represented must match the precision and data types returned by the robot's sensors

  (Camera v/s Laser rangefinder)

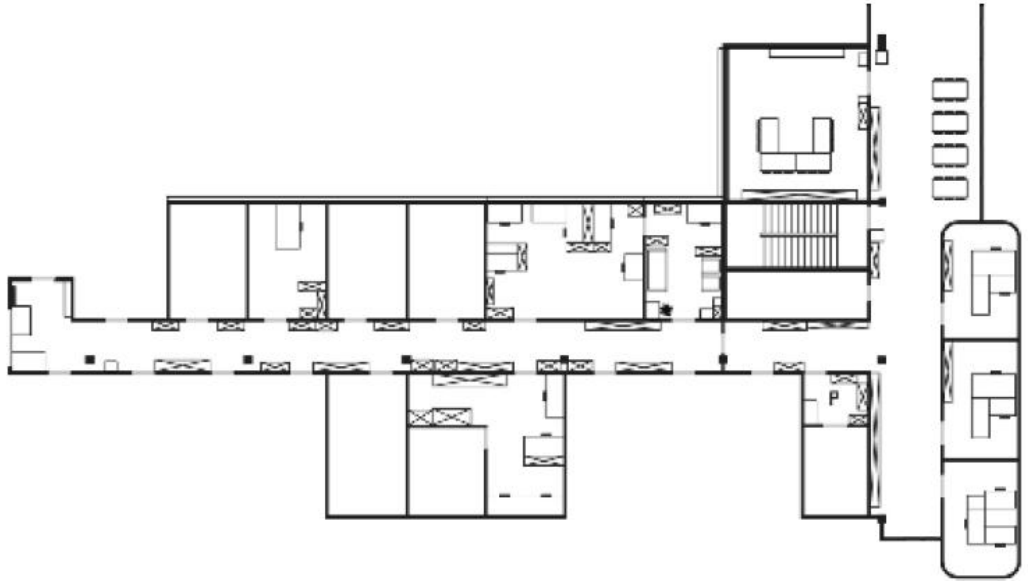
  3) Complexity of map representation has direct impact on computational complexity of reasoning about mapping, localization and navigation (3D v/s 2D representation)

Map Representation

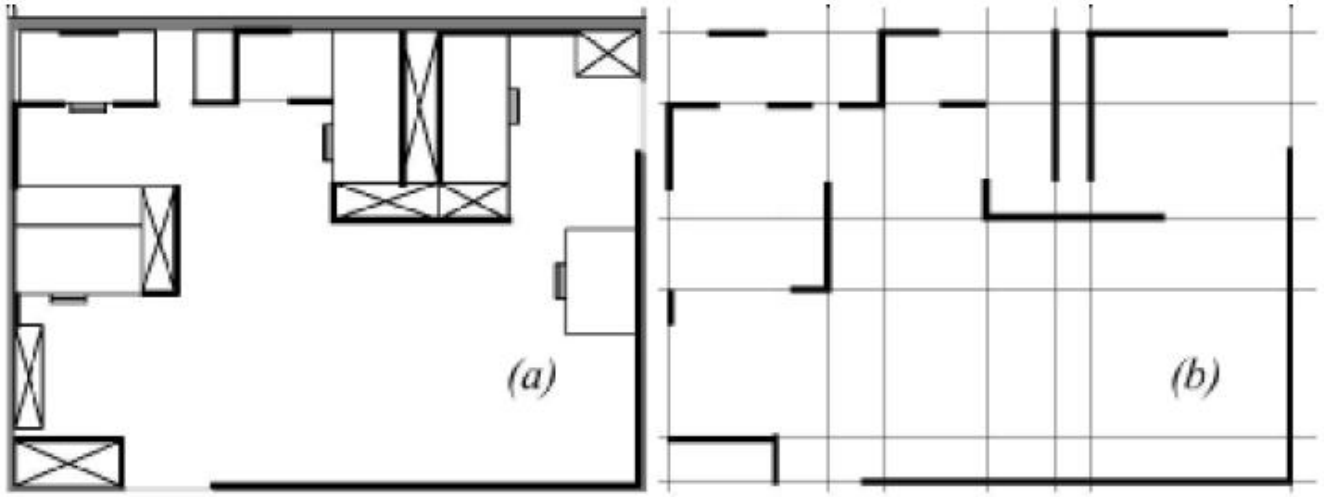
- Map representation can be classified into two major categories
  - Continuous representation
  - Discrete representation

- **<u>Continuous representations</u>** tend to be exact
- Can use geometric primitives to represent all objects in the environment. Example – lines, polygons
- Has the potential for high accuracy
- Map maybe computationally costly

Map Representation



Representation of Map with polygons



Real Map

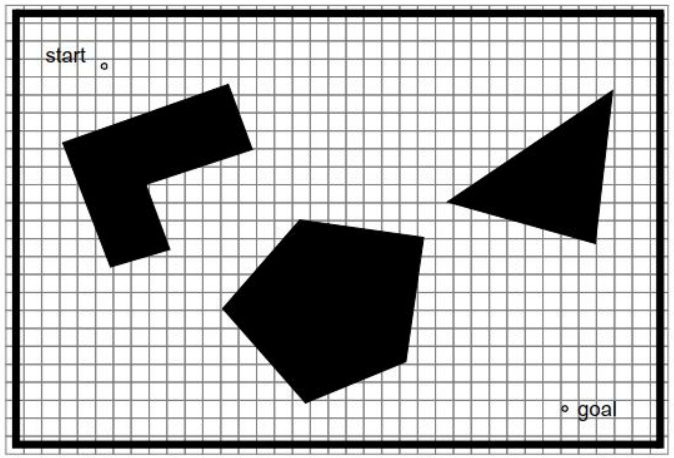
Representation of Map with lines

- Representing objects as polygons is not directly implementable
- In comparison, mobile robots use laser rangefinders, which give data from which lines can be fit

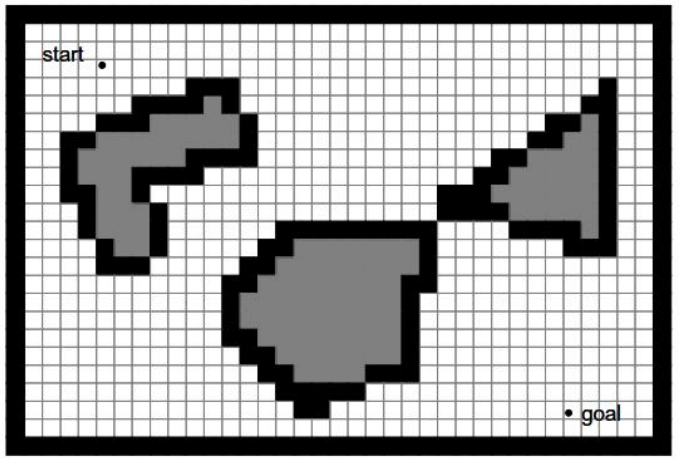
Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Map Representation - Discrete

- Fixed decomposition of space – transforming the continuous real environment into a discrete approximation for the map
- Represents map as free area, obstacle or unexplored
- Discrete representation can make it inexact, small gaps may not be identified



Actual World

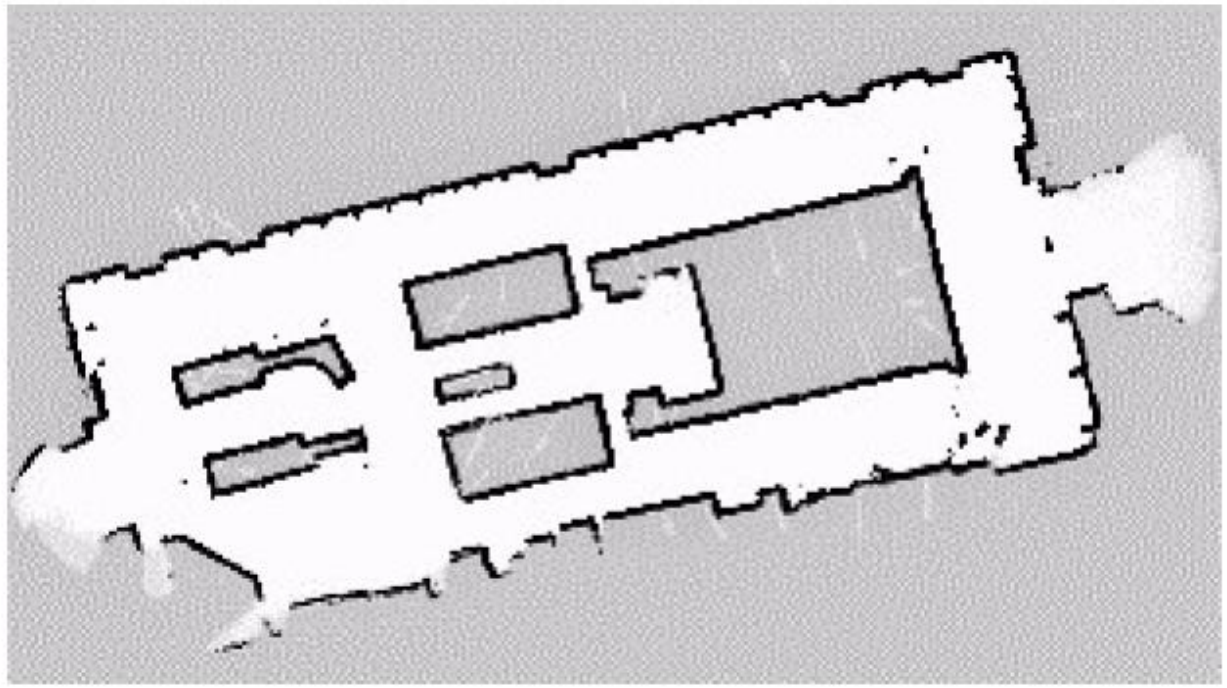


Fixed decomposition of world to create map

Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Map Representation - Discrete

- Most common map representation uses fixed decomposition method and is called as the ***occupancy grid*** representation



Occupancy Grid representation of an environment

Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Probabilistic Map-based Localization

- Robot moves in an environment for which Map is available
- Robot is considered to be moving in a known environment
- Robot's belief state will change over time so as to be consistent with motor outputs and perceptual inputs
- Probabilities identified for possible robot positions
- As robot starts to move from a known location, its motion can be kept track of using odometry
- Due to odometry uncertainty, after some movement, the robot will become very uncertain about its position
- To localize itself, robot must make observations using sensors

Probabilistic Map-based Localization

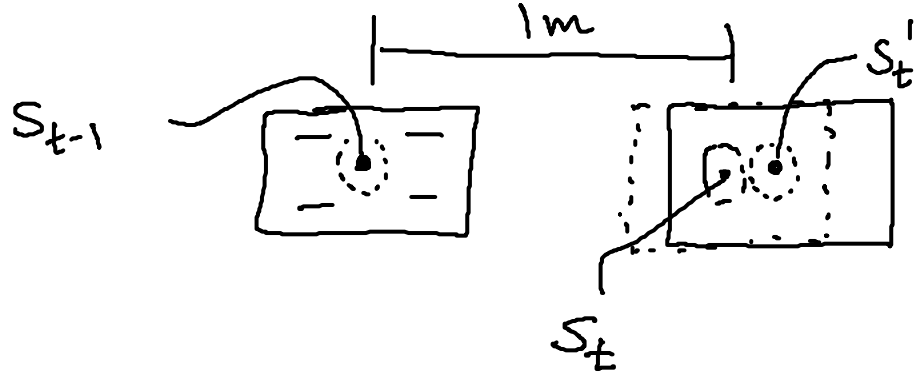
- Two step process used for updating
  - Action update
  - Perception update

- Action update represents application of some action model *Act* to mobile robot's proprioceptive encoder measurements o_t and prior belief state s_{t-1} and yields s'_t
- Perception update represents application of some perception model *See* to mobile robot's exteroceptive sensor inputs i_t and s'_t to yield belief state s_t

Probabilistic Map-based Localization

- Action update

$$S_t' = Act(O_t, S_{t-1})$$



- Perception update

$$S_t = See(i_t, S_t')$$

0.9 m

Probabilistic Map-based Localization

Monte Carlo Localization – based on Particle Filter

Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t$):

$\mathcal{X}_t = \mathcal{X}_t = \emptyset$

for $m = 1$ to M do

$x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$

$w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$

$\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ to M do

draw i with probability $\propto w_t^{[i]}$

add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

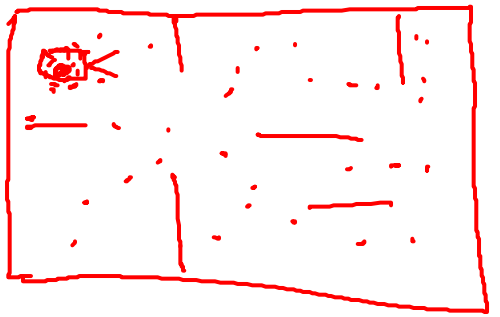
Probabilistic Map-based Localization

Monte Carlo Localization – based on Particle Filter

1: **Algorithm MCL**($\mathcal{X}_{t-1}, u_t, z_t$):
2: $\mathcal{X}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Resampling

M=1000



Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Autonomous Mobile Robots

Module 27:Landmark Based Navigation

Globally Unique Localization

Positioning Beacon Systems

Route-based localization

Landmark-based navigation

- Landmarks are defined as passive objects in the environment that provide a high degree of localization accuracy when they are within the robot's field of view
- Generally done using artificial markers that have been placed by the robot's designers to make localization easy
- The control system for a landmark-based navigator consists of two discrete phases.

Landmark-based navigation

- When a landmark is in view, the robot localizes frequently and accurately, using action update and perception update to track its position without cumulative error.
- But when the robot is in no landmark “zone,” then only action update occurs, and the robot accumulates position uncertainty until the next landmark enters the robot’s field of view.
- The robot must ensure that each motion between landmarks is sufficiently short, given its motion model, that it will be able to localize successfully upon reaching the next landmark

Landmark-based navigation

- The primary disadvantage of landmark-based navigation is that in general it requires significant environmental modification.

- Landmarks are local, and therefore a large number are usually required to cover a large factory area or research laboratory.

Landmark-based navigation

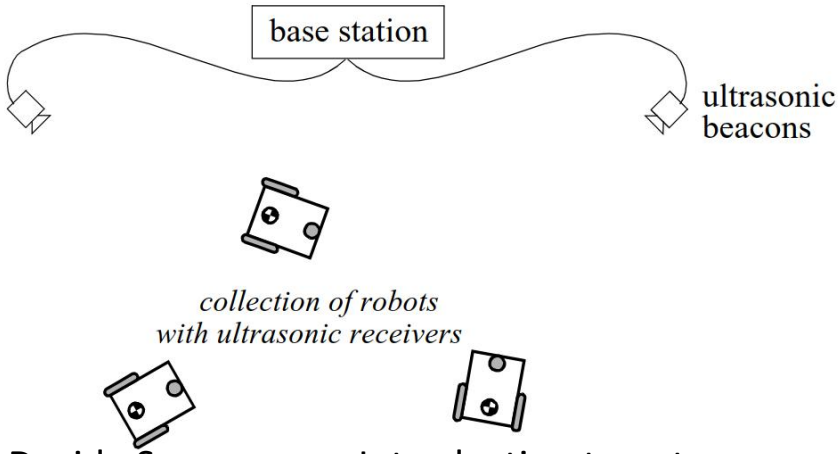
- What if every location has a unique landmark which lets the robot localize -> Globally unique localization
- Such a localization system would need to use a sensor that collects a very large amount of information. Vision sensors are used for this purpose
- One method for globally unique localization is called mosaic-based localization
- Takes advantage of fine-grained floor texture
- Robot is fitted with a high-quality high-speed CCD camera pointed toward the floor

Landmark-based navigation

- Once the complete image mosaic is stored, the robot can travel any trajectory on the floor while tracking its own position without difficulty.
- Localization is performed by simply recording one image and matching the image to the mosaic database using simple techniques based on image database matching.
- Results show robot can localize repeatedly with 1 mm precision while moving at 25 km/hr
- Main disadvantage of globally unique localization is that there will always be cases where local sensory information is truly ambiguous
- Compare with human abilities

Positioning beacon system

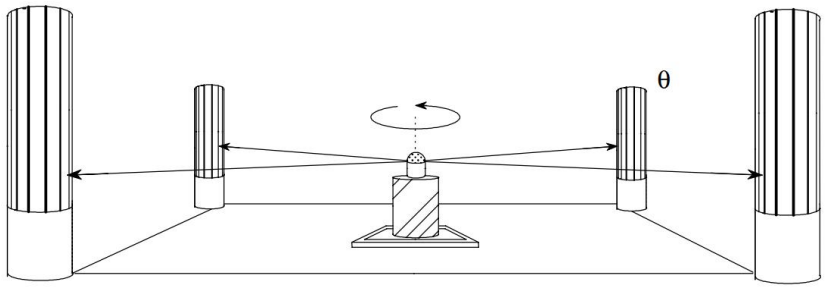
- Another way to solve the localization problem is to use an active beacon system specifically for the target environment
- GPS is considered as an example of such a system
- Based upon geometric principles to effect localization.
- In image shown below, the robots must know the positions of the two active ultrasonic beacons in the global coordinate frame in order to localize themselves to the global coordinate frame.



Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Positioning beacon system

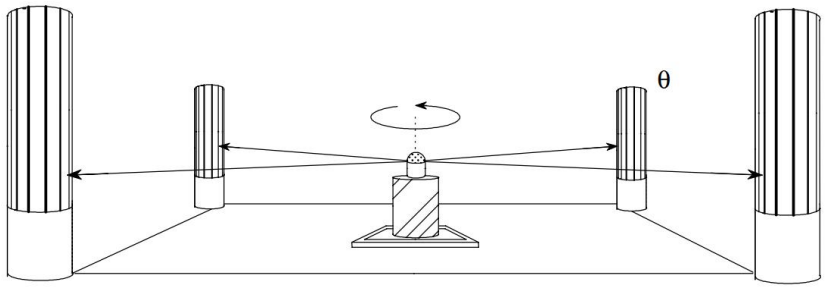
- Optical beacons can also be used which use retroreflective markers that can be easily detected by a mobile robot based on their reflection of energy back to the robot.
- Given known positions for the optical retroreflectors, a mobile robot can identify its position whenever it has three such beacons in sight simultaneously.
- Disadvantage is that significant engineering usually surrounds the installation of such a system, therefore, moving the robot to a different factory floor will be both, time consuming and expensive.



Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Route-based Localization

- In this case, the route of the robot is explicitly marked so that it can determine its position relative to the specific path it is allowed to travel.
- For example, high ultraviolet-reflective, optically transparent paint can mark the route such that only the robot, using a specialized sensor, easily detects it.
- Robot is forced to follow a prescribed path



Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Autonomous Mobile Robots

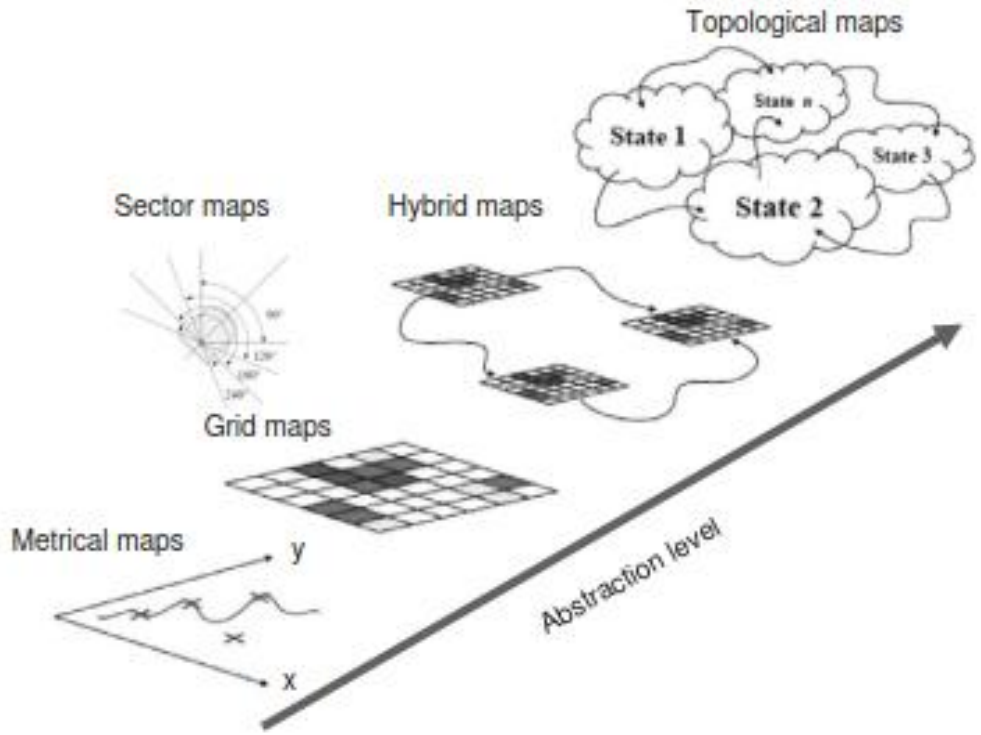
Module 28: Mapping

- Metrical maps
- Grid maps
- Sector maps
– Hybrid Maps

Mapping

- There are numerous ways to model the environment of a vehicle. The data structures can be broadly divided into

    1) Metrical Maps
    2) Grid Maps & Sector Maps
    3) Topological Maps
    4) Hybrid Maps



Reproduced from: Berns, Karsten and von Puttkamer, Ewald. Autonomous Land Vehicles. Springer, 2009.

Mapping - metrical maps

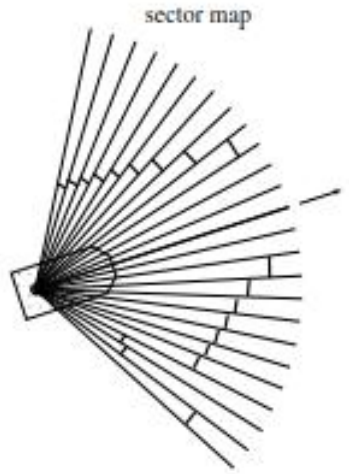
- In metrical maps, the environment is described with the help of geometrical features such as 2D or 3D points, lines, polygons or 2D areas like rectangles.
- For indoor scenarios, lines could describe the walls of rooms. In outdoor scenarios, lines could denote streets or highways.
- An advantage is that metrical feature maps offer a more compact description of the surroundings than grid maps. Hence they are superior especially in a scenario with a rather structured environment.
- We can have line-based metrical maps and plane-based metrical maps

Mapping - Grid maps

- Grid maps model the environment as a regular grid of cells with constant areas.
- The cells are filled with information extracted from noisy sensors.
- The sensors used are predominantly ranging sensors like sonar or laser scanners.
- High occupancy regions indicate objects, while lower values most likely represent free space.
- The advantage of this specific kind of map is the robustness and easy implementation.
- Their disadvantage is that they rely on exact pose estimation.
- Example: Occupancy Grid Maps

Mapping - Sector maps

- A sector map widens the rigid structure of a grid map and allows to partition space in a more flexible way
- A sector map is divided up into one or more sectors
- The most common separations consist of uniform polar or rectangular sectors



Reproduced from: Berns, Karsten and von Puttkamer, Ewald. Autonomous Land Vehicles. Springer, 2009.

Mapping - Topological maps

- Compared to metrical maps, topological maps are more abstract descriptions of large-scale structures of the environment.
- Topological maps are typically represented as graphs in which navigation-relevant places are modeled as graph nodes and connections between places are indicated by graph edges.
- Often, some metrical information is also stored in a topological map, such as the coordinates of a place or the metrical length of the topological edge.

Mapping - Topological maps

- Typical examples of topological maps are:
    - bus lines and bus stops in a town,
    - a highway network of a country,
    - a network of stations and railway lines for a subway or railway system,
    - a grid of the high voltage transmission lines of a country,
    - the sewage system of a town.
- Typical questions to be answered with help of a topological map are:
    - Where do I change buses between stations A and B?
    - Can I also drive from A to B via C or via D?
    - How many stops are there on the way from A to B?
    - If one transmission line fails, are there lines to circumvent the failed one?

Mapping - Hybrid maps

- Hybrid approaches try to combine both types, allowing localization and map building with the high precision of metrical maps while retaining the compactness of topological maps.

Autonomous Mobile Robots

Module 29: Simultaneous Localization and Mapping (SLAM)

SLAM

- Building a map & locating the robot in the map at the same time
- Cannot decouple one from the ~~re~~ other
- Need to know position of robot to know map of landmarks & Vice Versa.

Given:

1. The robot's controls:
$u_{1:T} = \{u_1, u_2, u_3, \ldots u_T\}$ → raw commands such as velocity commands (or) odometry readings.

2. Observations:
$z_{1:T} = \{z_1, z_2, z_3, \ldots z_T\}$ → Laser rangefinder, Camera

Want to find:

1. map of the environment, m (or landmarks)
2. Path of the robot
$x_{0:T} = \{x_0, x_1, x_2, \ldots x_T\}$

Online SLAM: Only interested in x_T, not previous values.

SLAM

3 paradigms of SLAM algorithms

- Kalman filter
- Particle filter
- Graph-based

Recall Bayes filter:

Prediction step

$$\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) \cdot bel(x_{t-1})\, dx_{t-1}$$

Correction step

$$bel(x_t) = \eta\, p(z_t | x_t) \cdot \overline{bel}(x_t)$$

for KF	PF
$\bar{\mu}, \bar{\Sigma}$	$\bar{X}_t$
$\mu_t \Sigma_t$	X_t

SLAM

Extended Kalman filter – for Online SLAM.

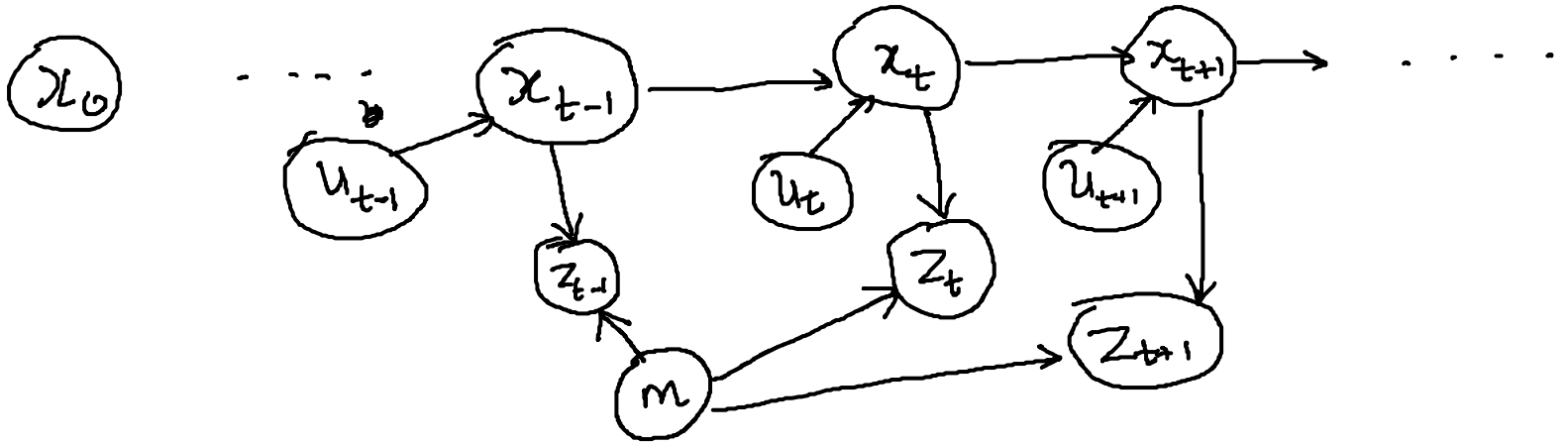
— Estimate robot's pose & locations of landmarks in the environment.

— Interested only in Current pose.

— Consider the KF as a solution to the Online SLAM problem

x_t : x
y
θ
m :

$$p(x_t, m \mid z_{1:t}, u_{1:t})$$



SLAM

Algorithm Extended_Kalman_filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**

$\bar{\mu}_t = g(u_t, \mu_{t-1})$

$\bar{\Sigma}_t = G_t \, \Sigma_{t-1} \, G_t^T + R_t$

$K_t = \Sigma_t \, H_t^T (H_t \, \Sigma_t \, H_t^T + Q_t)^{-1}$

$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$

$\Sigma_t = (I - K_t \, H_t) \, \bar{\Sigma}_t$

return μ_t, Σ_t

bel μ, Σ

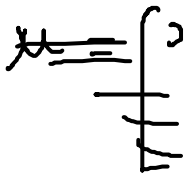
$g \xrightarrow{\text{Jacobian}} G_t$

$h \longrightarrow H_t$

μ_t, Σ_t

Localization : Robot's pose : States : x, y, θ

Localization & mapping : States : $\{x, y, \theta, m\}$



SLAM

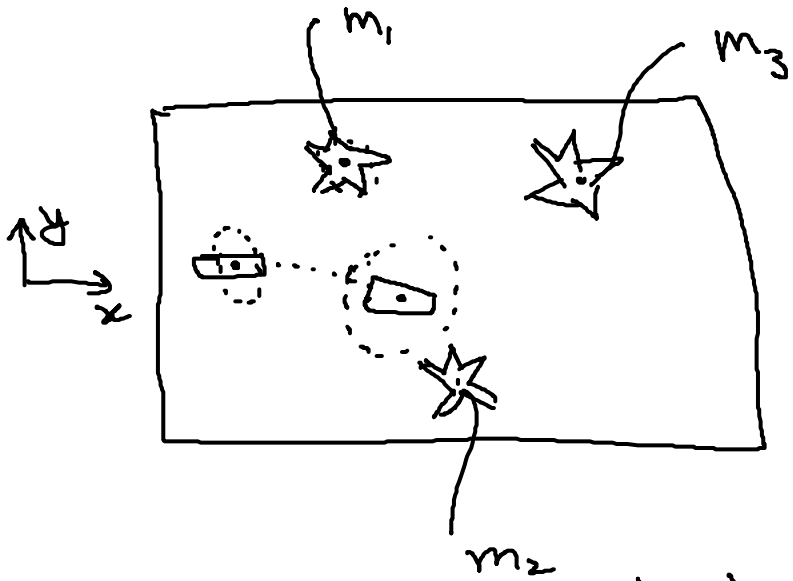
Each landmark introduces 2 state variables

$m_1 : (m_{1x}, m_{1y})$

$m_2 : (m_{2x}, m_{2y})$

$\vdots$

$m_n : (m_{nx}, m_{ny})$



Assume n - landmarks

$m_1, m_2, \ldots m_n$

$x_t \rightarrow x, y, \theta$

$m \rightarrow m_{1x}, \ldots$

State space for SLAM

$$\underline{x_t} = (\underbrace{x, y, \theta}, \underbrace{m_{1x}, m_{1y}, m_{2x}, m_{2y}, \ldots, m_{nx}, m_{ny}})$$

$3 + 2n$ dimensional Gaussian

$$\mu = \begin{bmatrix} x \\ y \\ \theta \\ m_{1x} \\ m_{1y} \\ m_{2x} \\ \vdots \\ m_{ny} \end{bmatrix}$$

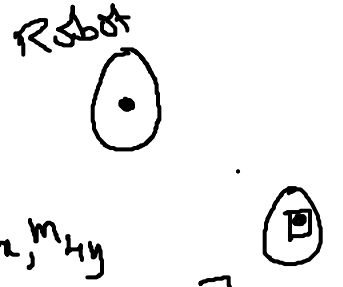
$\Sigma : (3+2n) \times (3+2n)$

This μ, Σ represent belief for EKF

SLAM

Ekf SLAM Steps

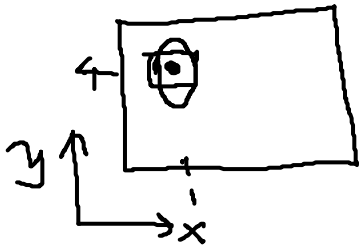
1. State prediction
2. Measurement prediction
3. Measurement
4. Data association
5. Update

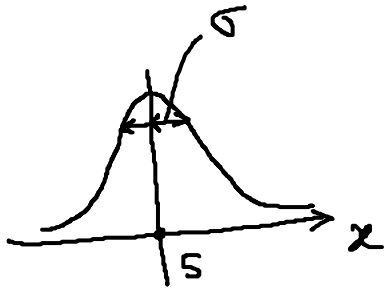


3+2n

11 states $x, y, \theta, m_{1x}, m_{1y}, m_{2x}, m_{2y} \ldots m_{4x}, m_{4y}$

$$\begin{cases} \mu_t = \begin{bmatrix} \mu_x, \mu_y, \mu_\theta, \mu_{m_{1x}}, \mu_{m_{1y}}, \ldots \mu_{m_{4x}}, \mu_{m_{4y}} \end{bmatrix} \\ \Sigma_t = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & \vdots & \\ \sigma_{yx} & \sigma_y^2 & \sigma_{y\theta} & 3\times 8 & \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_\theta^2 & \vdots & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 8\times 3 & & & \vdots & 8\times 8 \end{bmatrix} \end{cases} \longrightarrow 11 \times 11 \text{ matrix}$$





SLAM

1) State prediction: Predicted motion. : g
   When robot moves forward, only x, y, θ are updated
   $m_1, m_2, \ldots m_n$ remain the same

2) Measurement prediction: predicted observation
   Given robot is at a location, what should be the observation it should get h

3) Obtained measurement. z_t

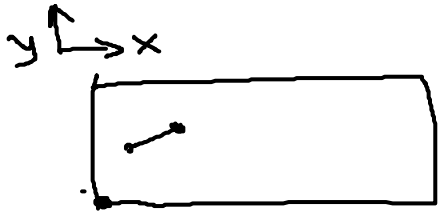
4) Data association. Compare your predicted measurement & actual measurement & associate landmarks appropriately.

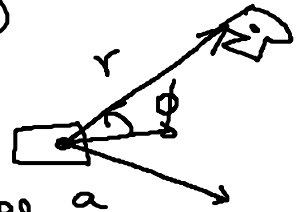
5) Update mean vector, kalman gain, covariance matrix
   belief of EKF at time t

SLAM

Example:

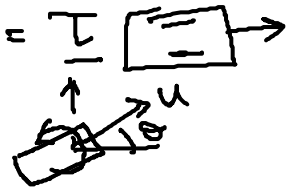
- Robot moves in 2D plane
- We have a motion model for robot (velocity-based)
- Robot observes point landmark (x, y)
- Range-bearing sensor
- Known data association: when we see a LM, we know which LM it is
- Known numbers of Landmarks.





Kinematic model

$$\dot{x} = u\cos\theta$$
$$\dot{y} = u\sin\theta$$
$$\dot{\theta} = \frac{I}{}$$



Initialize $\mu_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ $\Sigma_0 = \begin{bmatrix} 0.1 & & & & \\ & 0 & & & \\ & & 0.01 & & \\ & & & \infty & \\ & & & & \infty \\ & & & & & \ddots \\ & & & & & & \infty \end{bmatrix}$

SLAM

EKF algo :

Motion model g

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) \quad : \rightarrow \text{update only robot pose \& not LM locns}$$

$$\bar{\Sigma}_t \quad : \text{need } G_t$$

$$K_t = \bar{\Sigma}_t H_t^T \left(H_t \bar{\Sigma}_t H_t^T + \theta_t\right)^{-1} \quad : \text{need Jacobian of } h,\ H_t$$

Update $\mu_t = \bar{\mu}_t + K_t \left(z_t - h(\bar{\mu}_t)\right)$

z_t ↑ actual ; $h(\bar{\mu}_t)$ ← predicted measurement

$$\Sigma_t = (I - K_t H_t)\, \bar{\Sigma}_t$$

Autonomous Mobile Robots

Module 30: Unit IV Introduction

Path Planning Overview

UNIT – IV **13 Periods**

PLANNING AND NAVIGATION: Introduction-Path planning overview- Global path planning – A* Algorithm - local path planning - Road map path planning- Cell decomposition path planning-Potential field path planning-Obstacle avoidance – Path control.

Introduction to Navigation and Planning

- We have covered so far
  - Mobility
  - Sensors
  - Techniques for localization and mapping

- Now we look at purposeful decision making aspects of robots – Cognitive level
- Navigation abilities of the robot are directly linked with the cognitive processes that the robot employs
- To demonstrate navigation, two key competencies are required for the robot
  - Path Planning
  - Obstacle Avoidance

Introduction to Navigation and Planning

- ***Path Planning:*** Given a map and a goal location, path planning involves identifying a trajectory that will cause the robot to reach the goal location when executed
- Robot must decide what to do over the long term to achieve goals

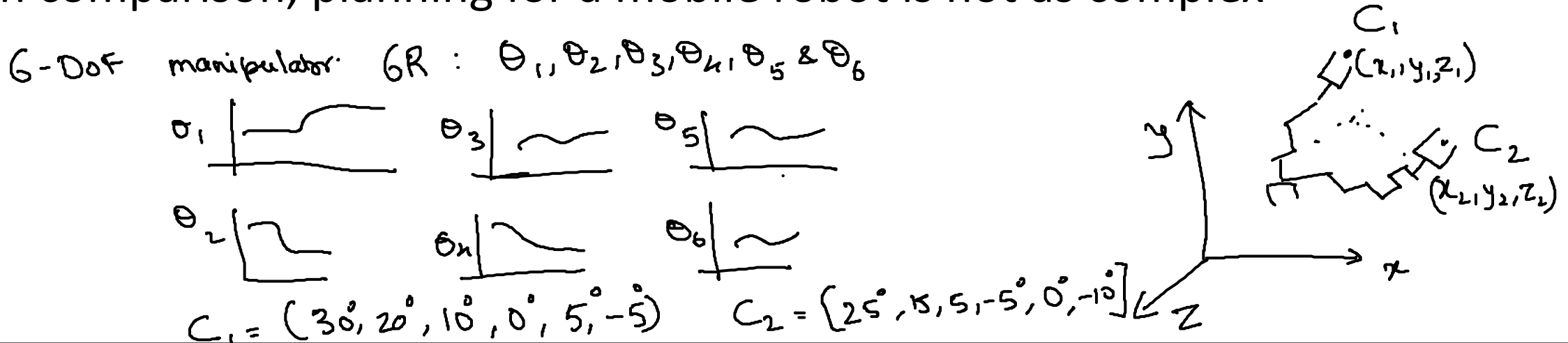
- ***Obstacle Avoidance:*** Given real-time sensor readings, obstacle avoidance is the process of modifying the robot's trajectory to avoid collisions
- There is a *planning* aspect and a *reacting* aspect
- Both aspects have many different approaches

Introduction to Navigation and Planning

- Environment changes as the robot moves and sensor's gather new information
- During these times, the reaction part becomes important
- The planner needs to incorporate new information and create plans
- Robot system is called *complete* if and only if for all problems, when there exists a trajectory to the goal, the system will achieve the goal
- If system is not *complete*, then even if a trajectory exists to the goal, system does not find it.

Introduction to Navigation and Planning

- Achieving completeness is hard, involves high computational complexity.
- Even before use of path planners for mobile robots, it was used for manipulators
- Path planning for manipulators arms tend to involve planning of all degrees of freedom.
- In comparison, planning for a mobile robot is not as complex



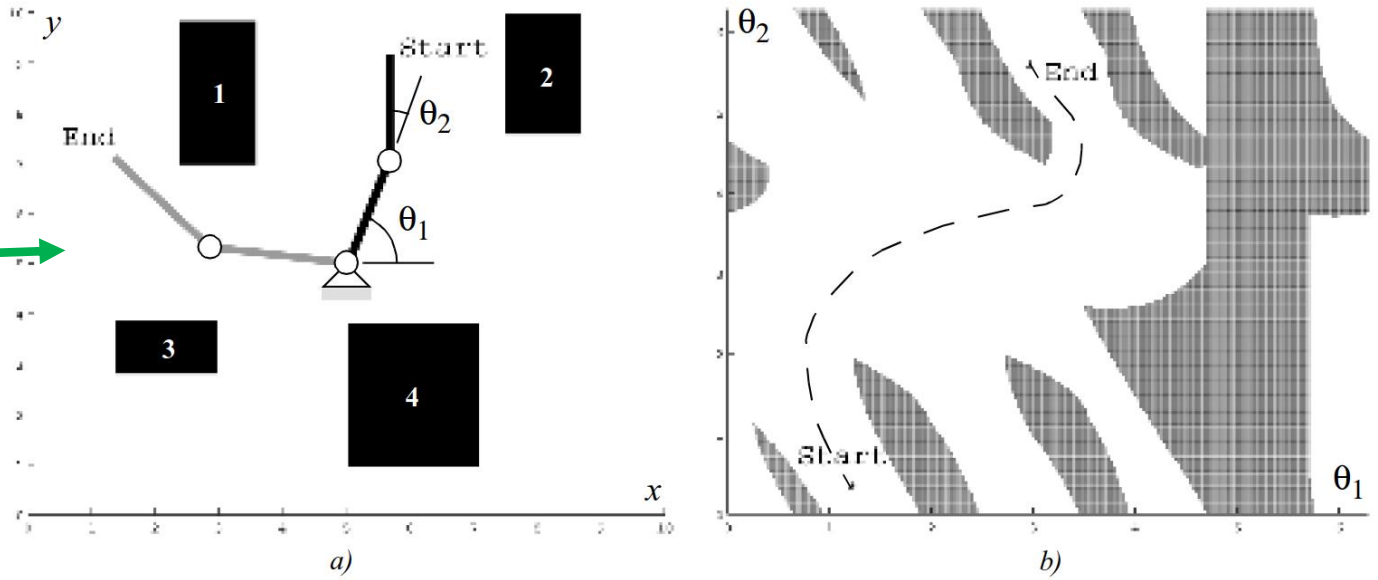
Path Planning

- Configuration Space: Path planning for manipulator robots and some times for mobile robots is done in configuration space
- If robot arm has *k-degrees of freedom*, every state or configuration of the robot can be described with *k real values:* $q_1, q_2, \ldots q_k$
- The k-values can be considered as a point p in k-dimensional space, called the configuration space *C* of the robot

Path Planning

- Consider the robot arm moving in an environment where the workspace contains known obstacles
- Goal of path planning is to find a path in the physical space from the initial position of the arm to the goal position, avoiding all collisions with obstacles
- This problem is more straightforward in configuration space

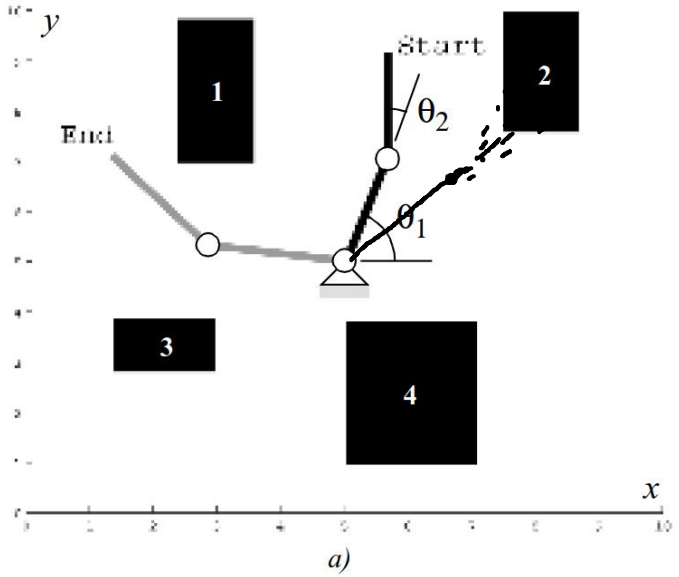
Physical Space



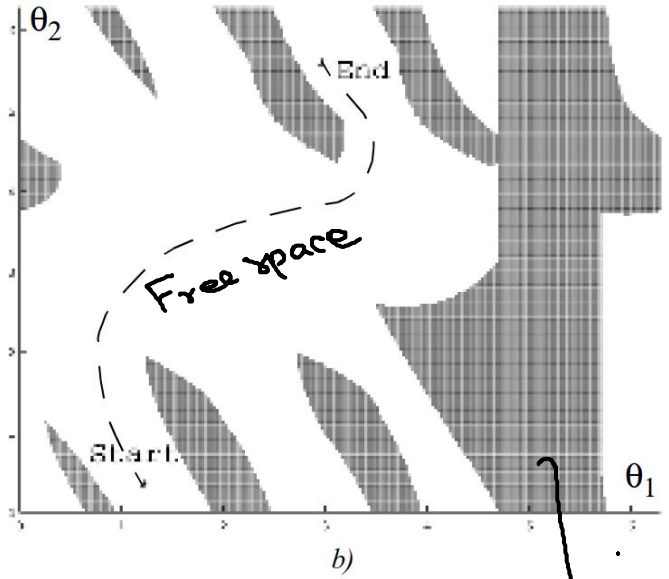
Configuration Space

Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Path Planning



a)



b)

Obstacle space

Define Configuration Space Obstacle O as a subspace of C where the robot arm collides with an object

$\theta_1 = 45^\circ$, $\theta_2 = 0$. falls on obst 2.

$\theta_1 = 45^\circ$, $\theta_2 = -5$ to -10

Physical Space (x, y) (x, y, z)

Configuration Space (θ_1, θ_2) $(\theta_1, \theta_2, \theta_3)$, K deg. of freedom

Planning in K-dim. Config space is easier compared to planning in Physical Space

Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Path Planning Considerations for Mobile Robots

- For mobile robots operating on flat ground, we generally represent robot configuration with three variables – x, y, θ
- Often plans are done under the assumption that the robot is simply a point
- We can further reduce the configuration space for mobile robot path planning to a 2D representation with just x- and y-axes
- The result of all this simplification is that the configuration space looks essentially identical to a 2D version of the physical space
- Because the robot is reduced to a point, we must inflate each obstacle by the size of the robot's radius to compensate.

Path Planning Overview

- The *global path planning* deals with finding a suitable path from a starting point to a goal point using a given representation of the environment.
- The *local path planning* defines path points (waypoints) taking into account the vehicle dimensions and kinematic constraints.
- *Path control* describes the task of generating suitable steering commands for following a precomputed path represented by reference points.

Path Planning Overview

- The robot's environment representation can be
  - a continuous geometric description
  - decomposition- based geometric map
  - topological map
- *A* or A-STAR:* search for a path from start to goal
- *Road Map*: identify a set of routes within free space
- *Cell decomposition*: discriminate between free and occupied cells
- *Potential field*: impose a mathematical function over space

Autonomous Mobile Robots

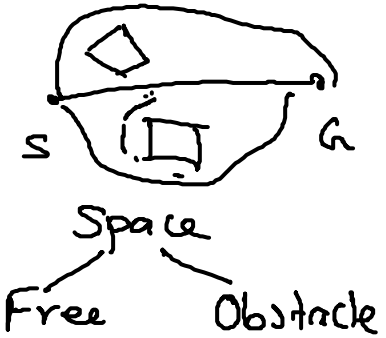
Module 31: Global Path Planning, A* Algorithm

Global Path Planning

- The *global path planning* deals with finding a suitable path from a starting point to a goal point using a given representation of the environment.
- The aim of global path planning is to describe a path through the environment from a starting point S to a goal point Z, fulfilling some conditions.
- One is to find the path with the lowest costs. This can depend on the length of the way, energy consumption for travelling, required time or a balanced mixture of that.
- Another condition can be that a path must be found to cover all free space, e. g. for cleaning vehicles.

A* Algorithm

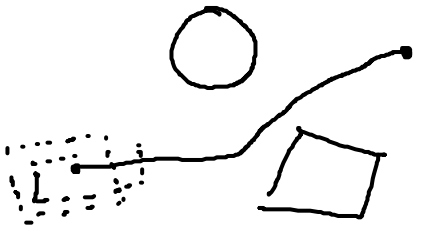
How does robot make decision to go from one place to another?



→ Goal :- Collision free trajectory.
- reach goal as fast as possible

Search-based algorithm

→ Challenges :
- Calculate optimal path
- Take potential uncertainties into account
- Quickly generate action in case of unforseen object
- Deal with moving obstacles



For motion planning problems:

Inputs
- Start pose of robot
- Desired goal pose
- Description of robot (geometric) — Size
- Representation of environment (geometric)

A* Algorithm

Output – A path that moves robot from start to goal while never colliding with obstacles

→ Perform planning in configuration space

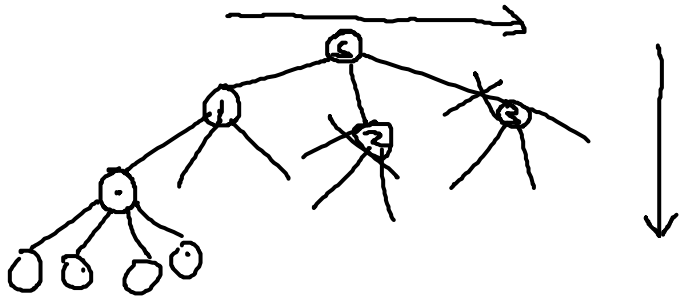
q_I : Start config.

q_G : goal config.

→ Discretize C-space use occupancy grid

→ Search configuration space for path

→ Search problem : find a sequence of actions (a path) that leads to a desirable state (goal).



Search
- Uninformed
  - Breadth first
  - Depth first
- Informed.
  - A*

A* Algorithm

Uninformed search is a blind search.

Informed search : based on information about the domain

Heuristics (or costs) are used to give info

Speed up the search ←

A* finds optimal path : can say that a particular node is more promising than another node

4 criteria for planning algorithms :

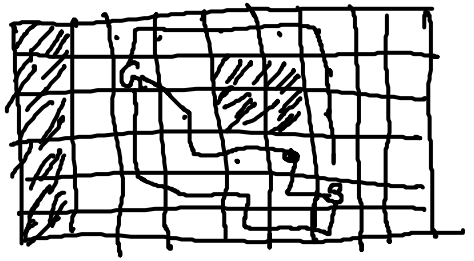
1) Completeness : does algorithm find a solution when there is one available
2) Optimality : is the solution obtained the best possible one.
3) Time complexity : how long does it take to find a solution
4) Space complexity : how much memory is needed to perform the search.

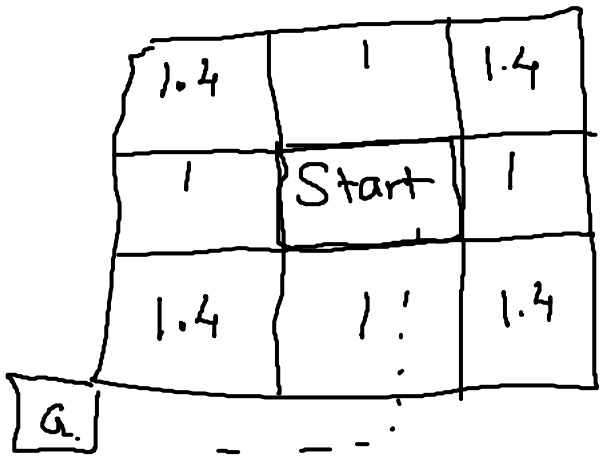
A* Algorithm

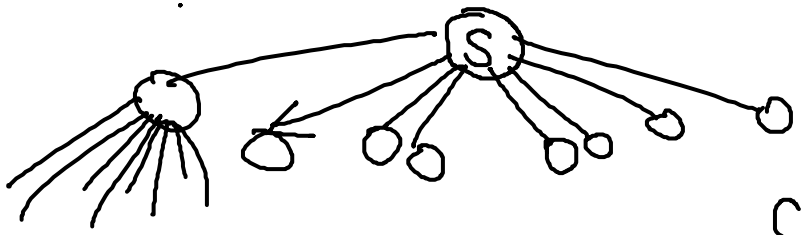
Assumptions :
1. Robot knows its own location
2. Robot computes its path based on a correct map.
3. The correct motion commands are executed

→ Map of environment is available & it is an occupancy grid

→ Each discretized unit (grid) can be thought of as a node







$F = G + H$

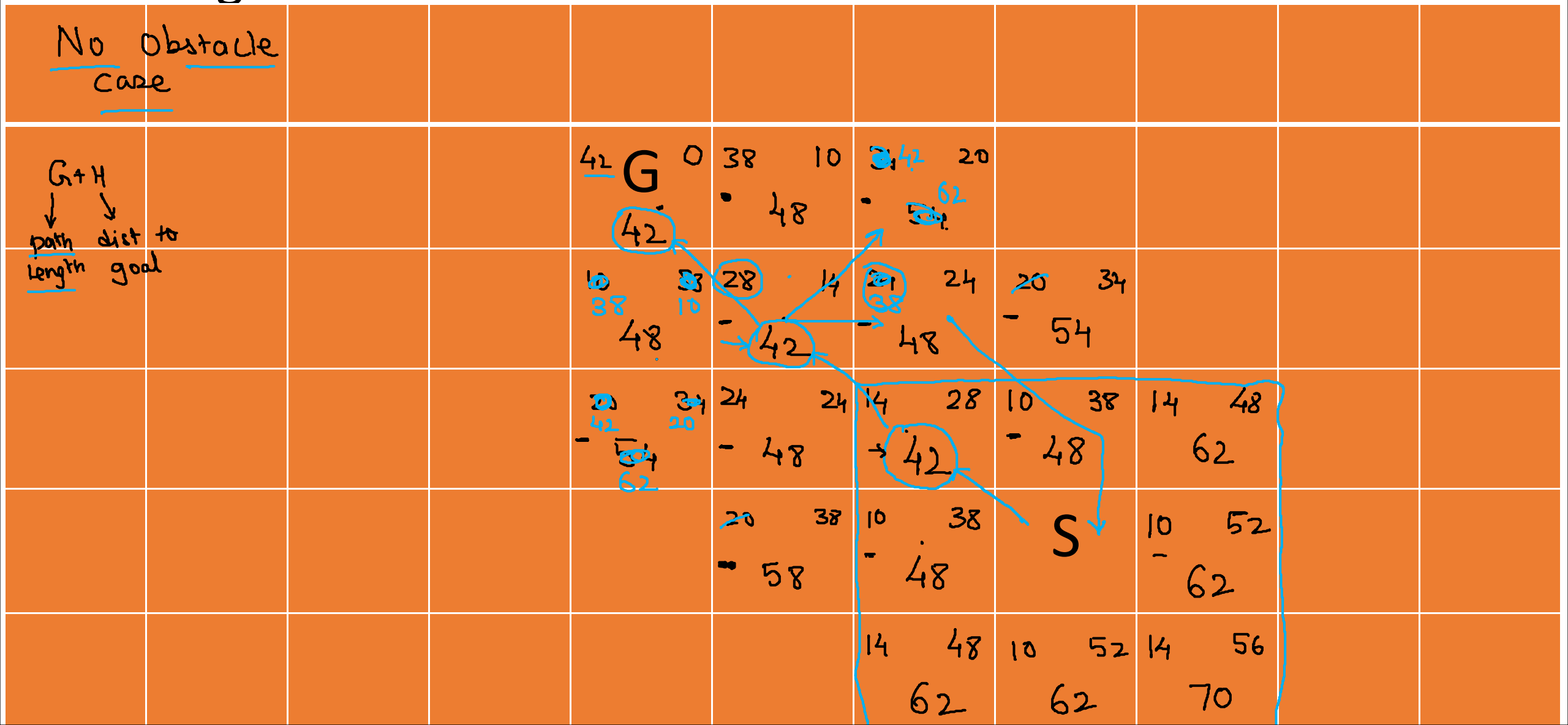
G — path cost

H — dist to goal.

Cost of moving up, down, left right = 10

Cost of moving on diagonal = 14

A* Algorithm


No Obstacle Case
G+H
path length
dist to goal
42 G 0
38 10
42 20
62
48
42
10 38
28 14
24
20 34
38
10
48
42
48
54
20 34
42
20
24 24
14 28
10 38
14 48
62
48
42
48
62
20 38
10 38
S
10 52
58
48
62
14 48
10 52
14 56
62
62
70

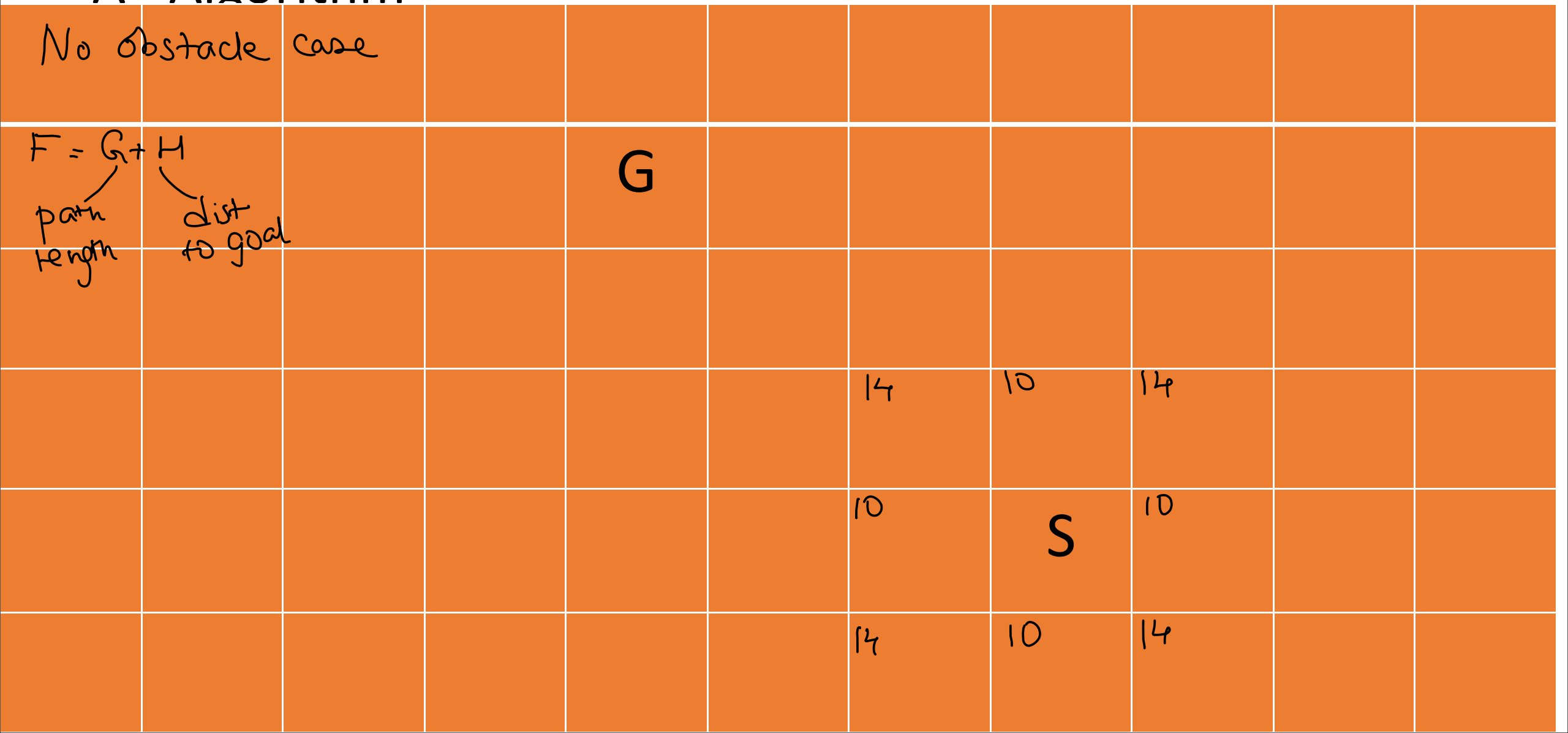
A* Algorithm

No Obstacle case

$F = G + H$

G — path length

H — dist to goal



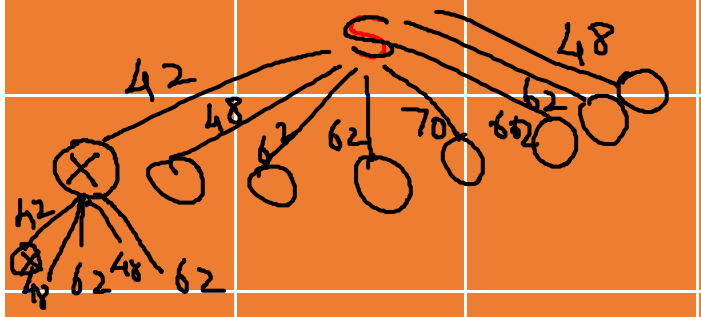
A* Algorithm

No Obstacle case

F = G + H

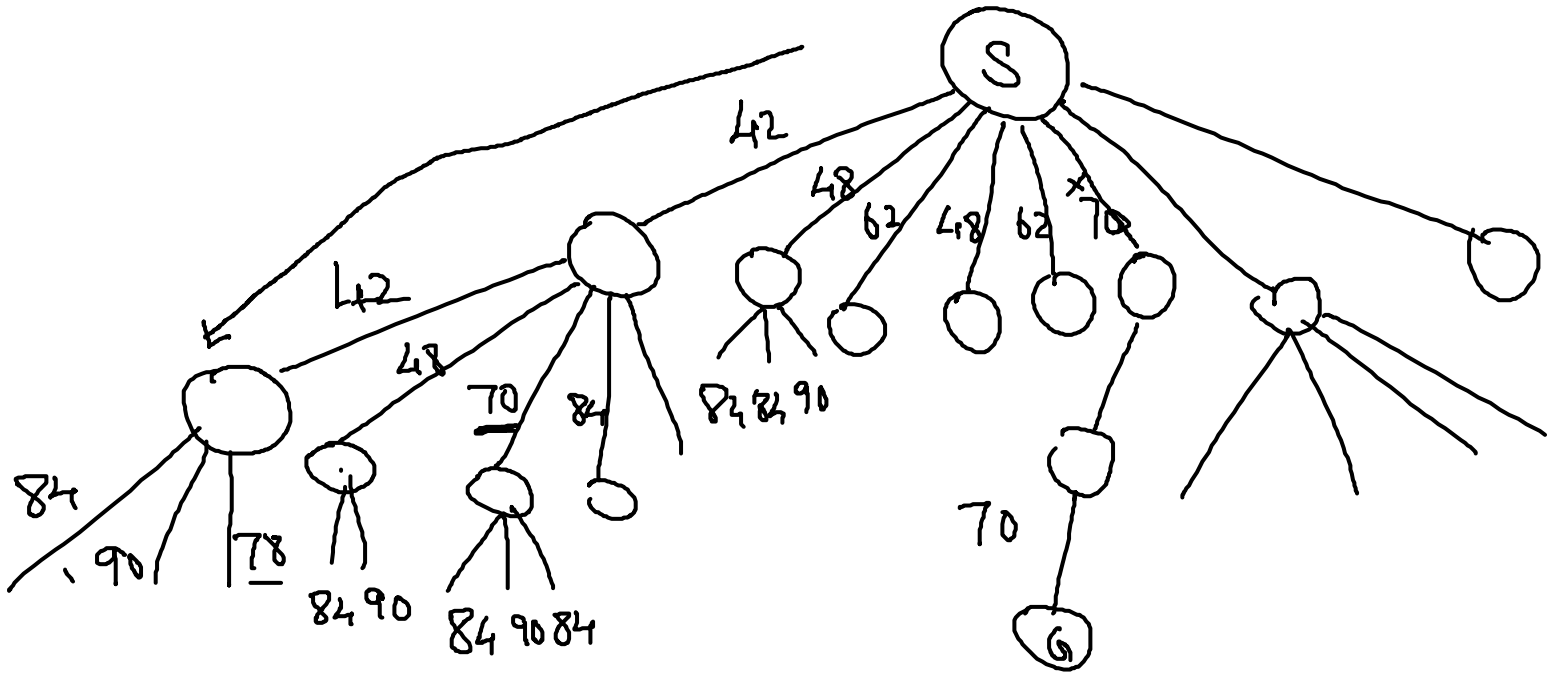
G: path length

H: dist to goal

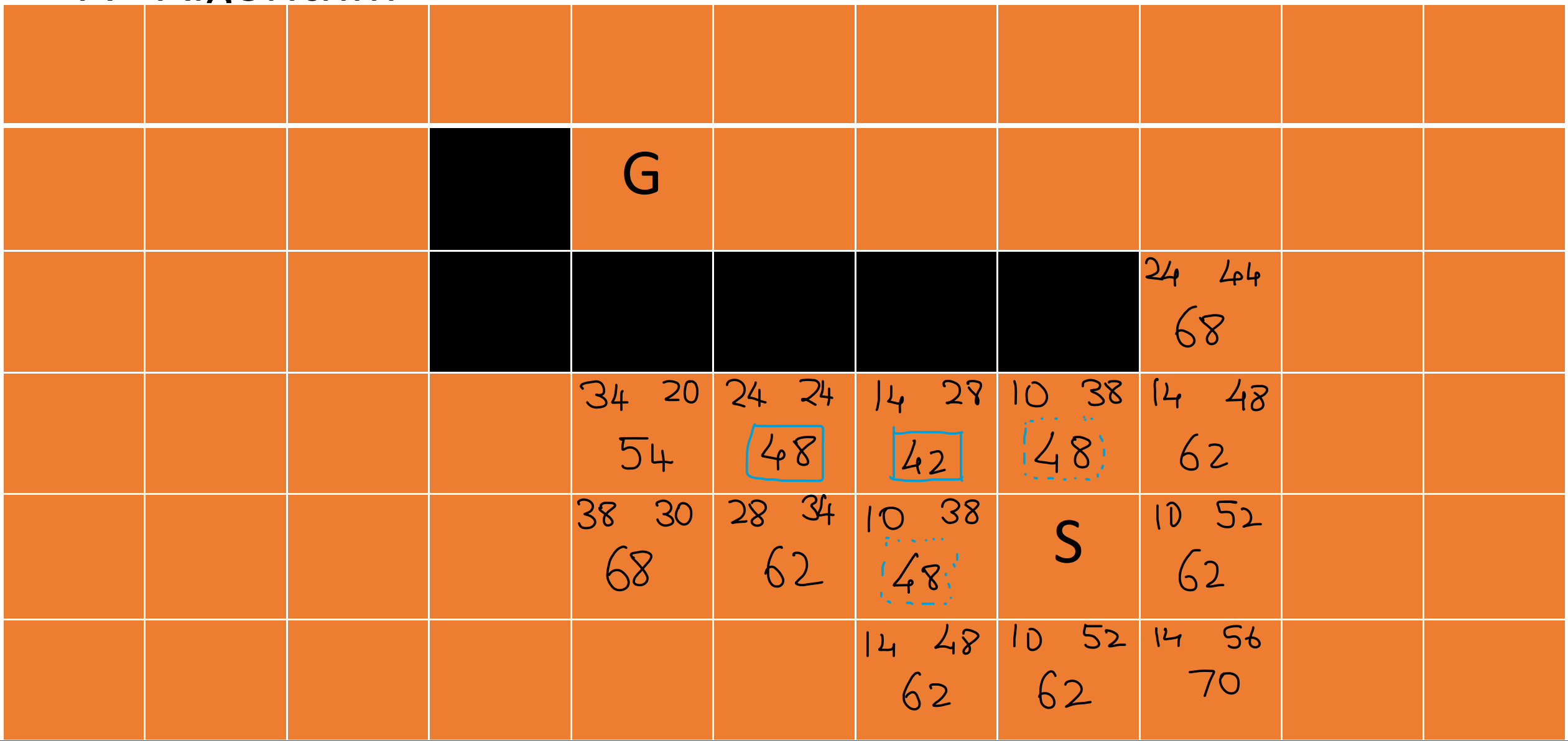


42 0 **G** 42	38 10 48	42 20 62				
38 10 48	28 14 42	24 24 48	28 34 62			
42 20 62	24 24 48	14 28 42	10 38 48	14 48 62		
	28 34 62	10 38 48	**S**	10 52 62		
		14 48 62	10 52 62	14 56 70		

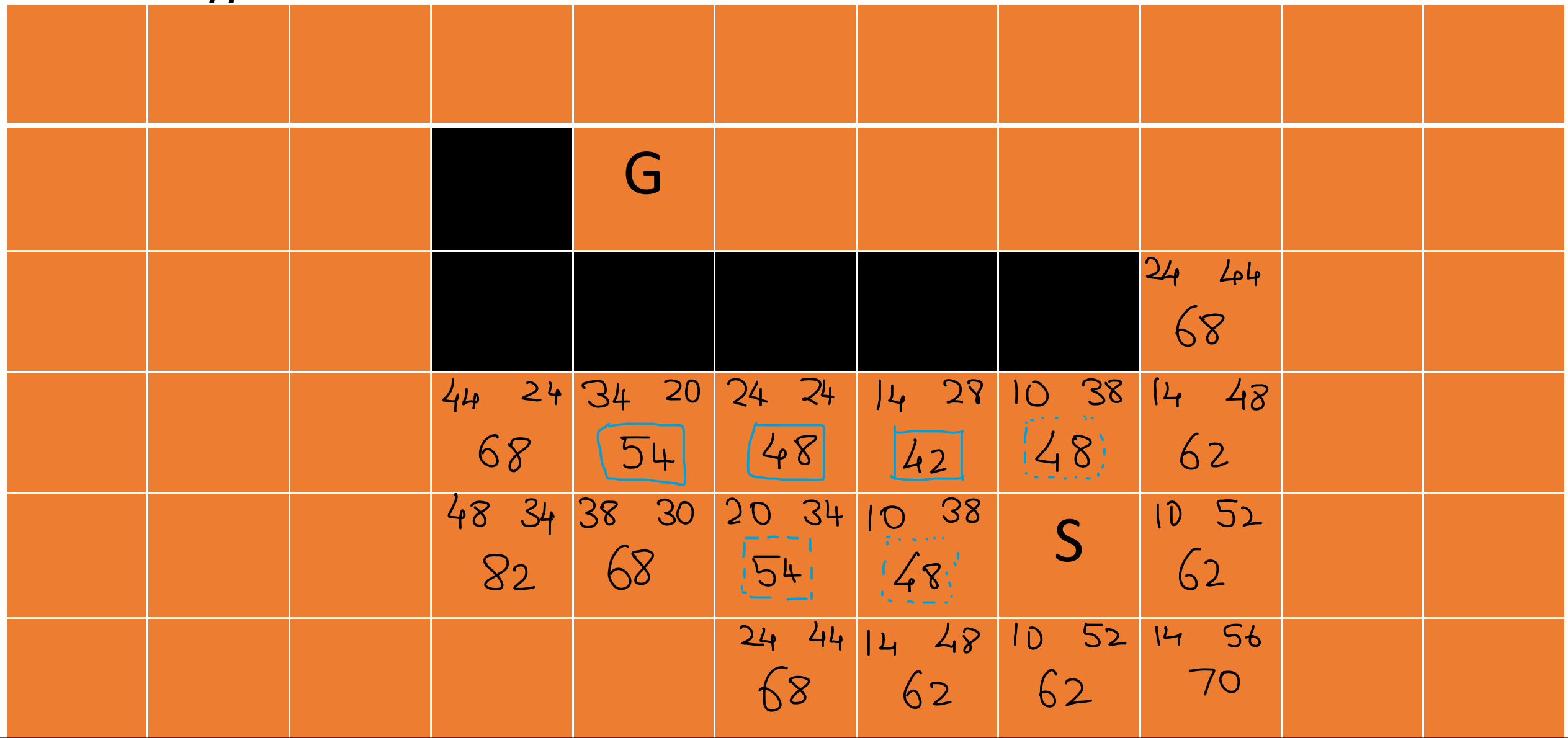
A* Algorithm



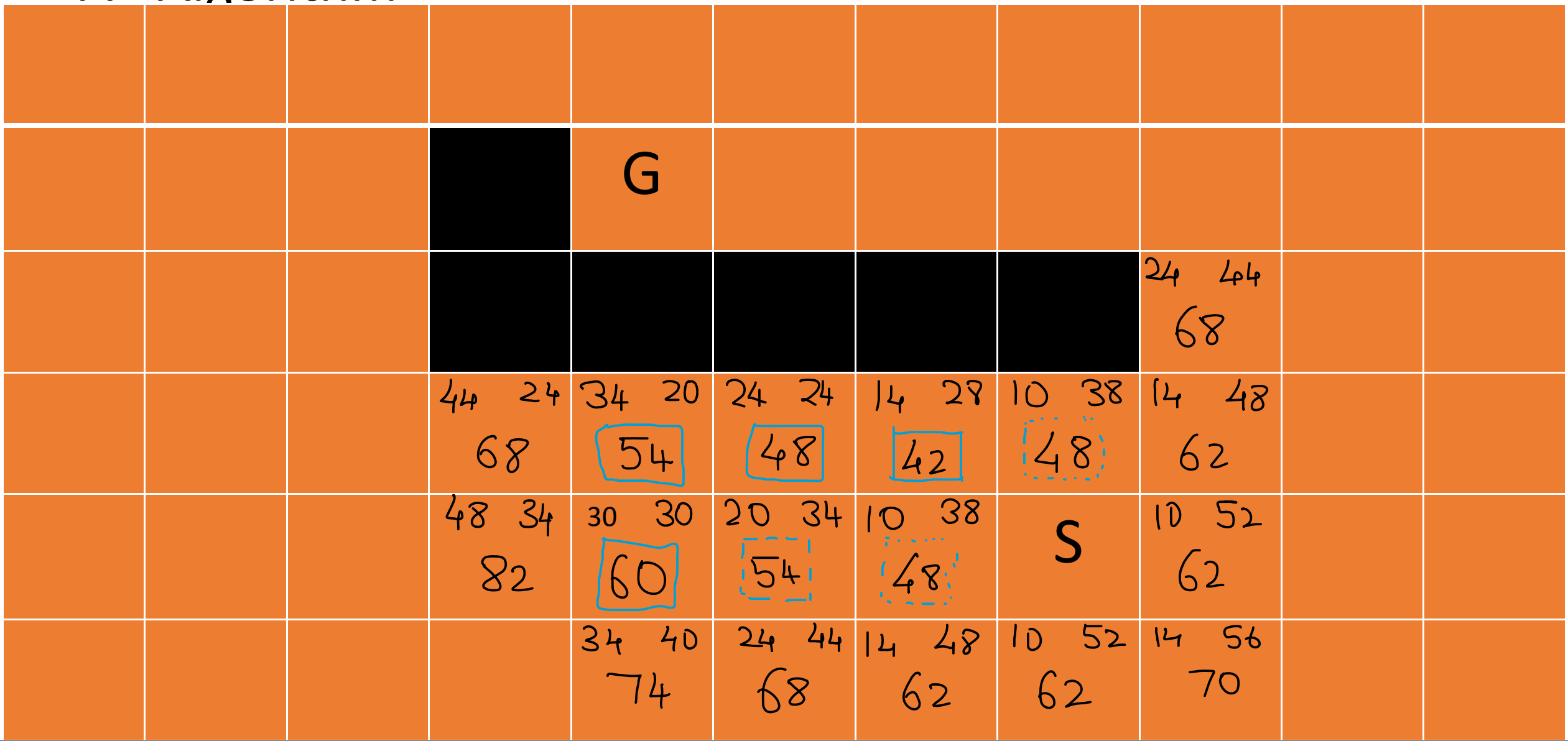
A* Algorithm


G
24 44
68
34 20
54
24 24
48
14 28
42
10 38
48
14 48
62
38 30
68
28 34
62
10 38
48
S
10 52
62
14 48
62
10 52
62
14 56
70

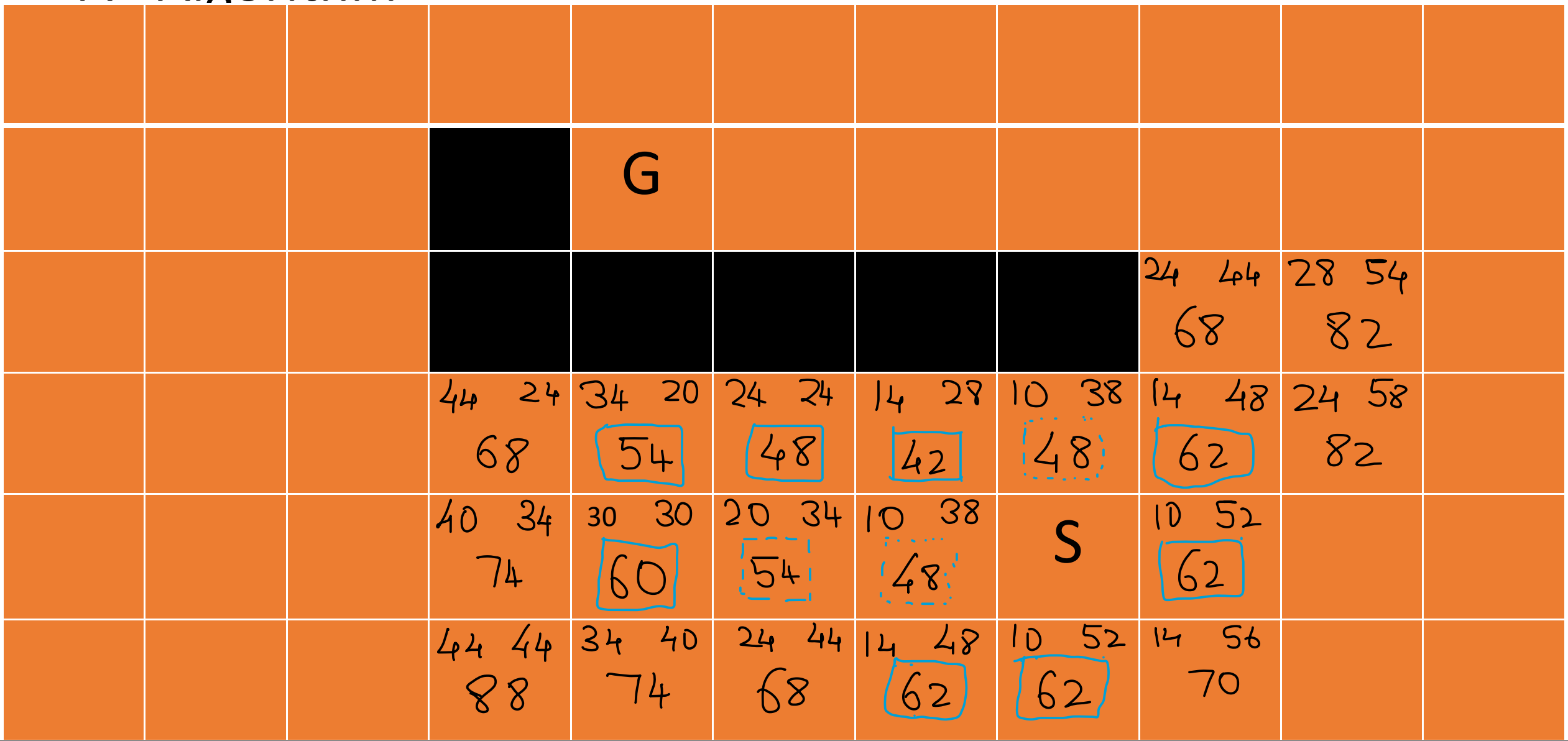
A* Algorithm


G
24 44
68
44 24
68
34 20
54
24 24
48
14 28
42
10 38
48
14 48
62
48 34
82
38 30
68
20 34
54
10 38
48
S
10 52
62
24 44
68
14 48
62
10 52
62
14 56
70

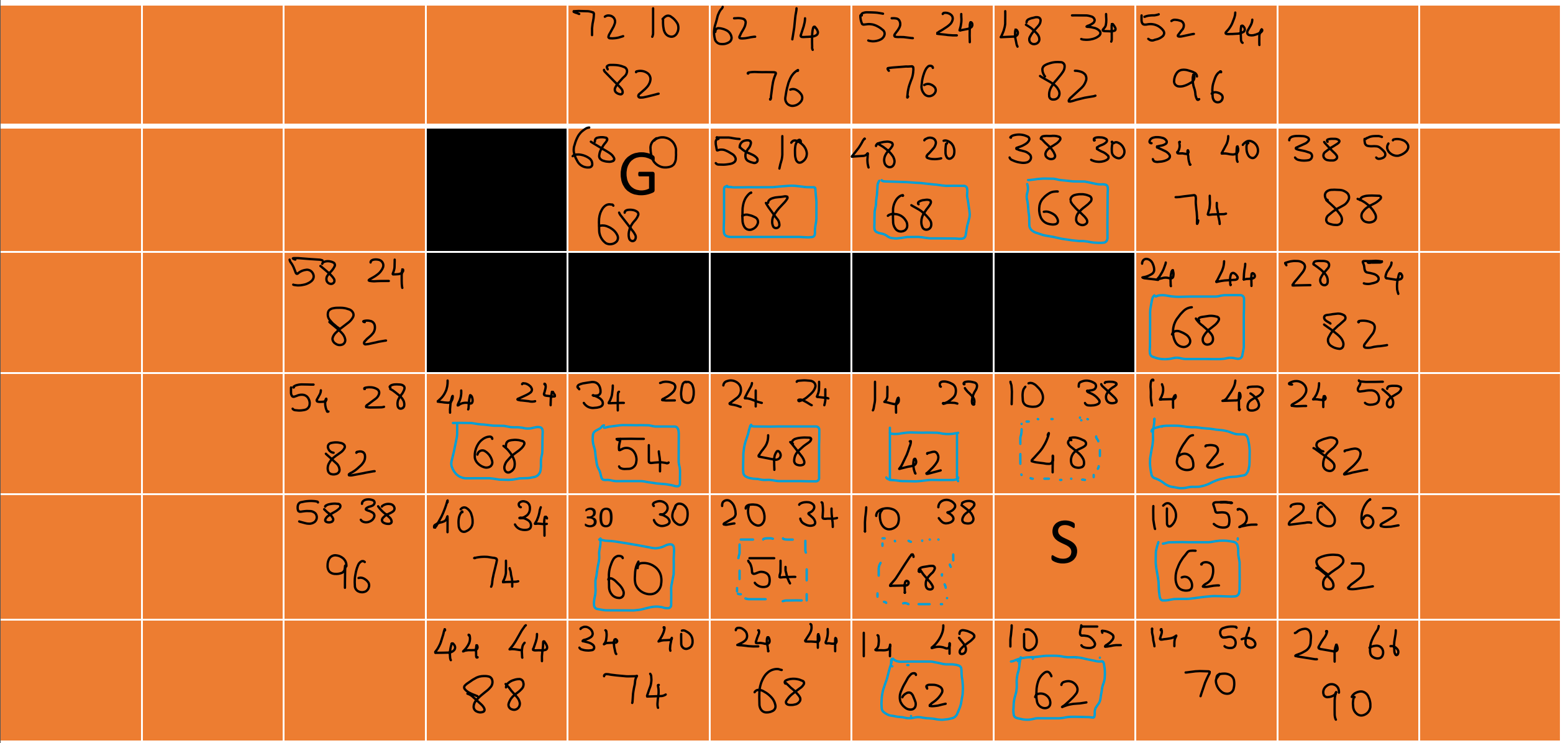
A* Algorithm


G
24 44
68
44 24
68
34 20
54
24 24
48
14 28
42
10 38
48
14 48
62
48 34
82
30 30
60
20 34
54
10 38
48
S
10 52
62
34 40
74
24 44
68
14 48
62
10 52
62
14 56
70

A* Algorithm


G
24 44
68
28 54
82
44 24
68
34 20
54
24 24
48
14 28
42
10 38
48
14 48
62
24 58
82
40 34
74
30 30
60
20 34
54
10 38
48
S
10 52
62
44 44
88
34 40
74
24 44
68
14 48
62
10 52
62
14 56
70

Autonomous Mobile Robots
MCT 308



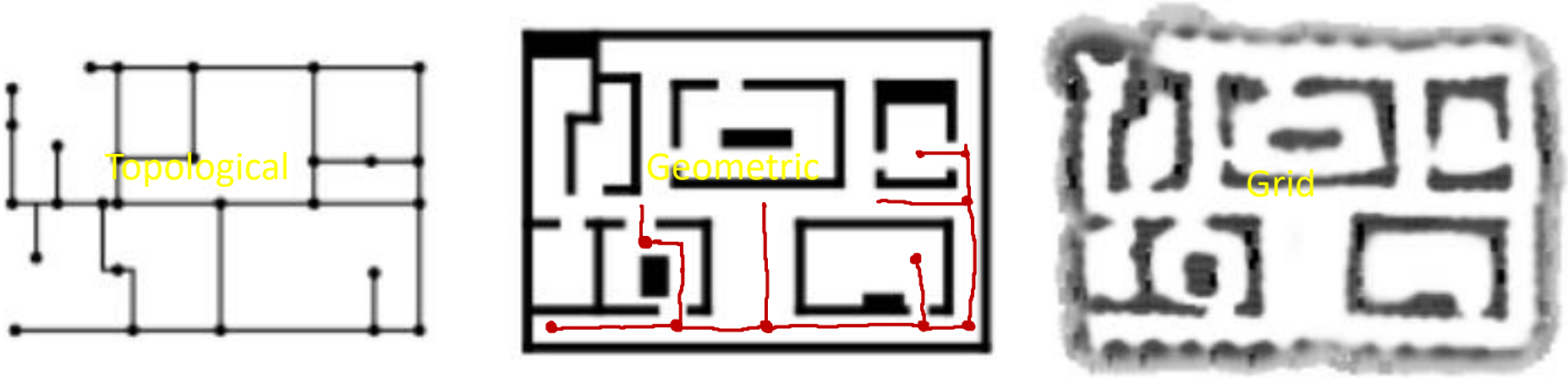
Autonomous Mobile Robots

Module 32: Road map path planning

Roadmap Path Planning

- If we knew that many paths were to be planned in the same environment, then it would make sense to construct a data structure once and then use that data structure to plan subsequent paths more quickly.
- *Roadmap Path Planning* is based on *topological* maps (in comparison to A* which was carried out on Occupancy grids)
- Topological representations represent environments with graph-like structures
- Nodes correspond to “something distinct” and edges represent relationship between nodes.

Roadmap Path Planning



- For example, a roadmap node corresponds to a specific location and an edge corresponds to a path between neighboring locations
- Robots use roadmaps in much the same way people use highway systems.

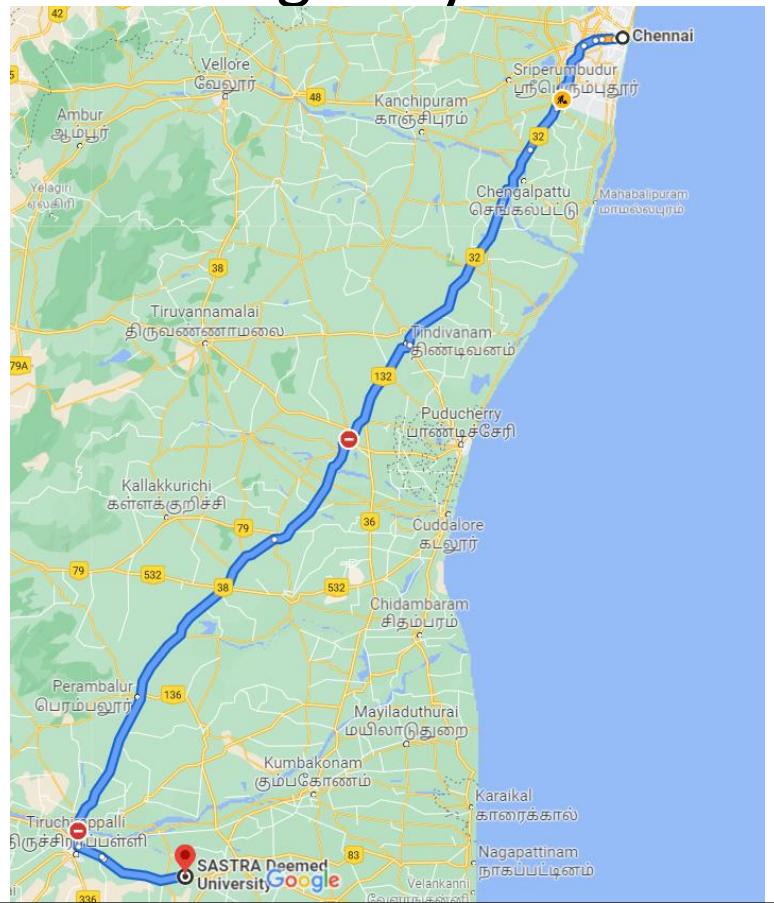
Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

Roadmap Path Planning

- Using a roadmap, the planner can construct a path between any two points in a connected component of the robot's free space by first finding a collision-free path onto the roadmap, traversing the roadmap to the vicinity of the goal, and then constructing a collision-free path from a point on the roadmap to the goal.

Roadmap Path Planning

- Instead of planning every possible side-street path to a destination, path to network of highways is planned, then along the highway system, and finally from the highway to the destination.



Google Map Result

Roadmap Path Planning

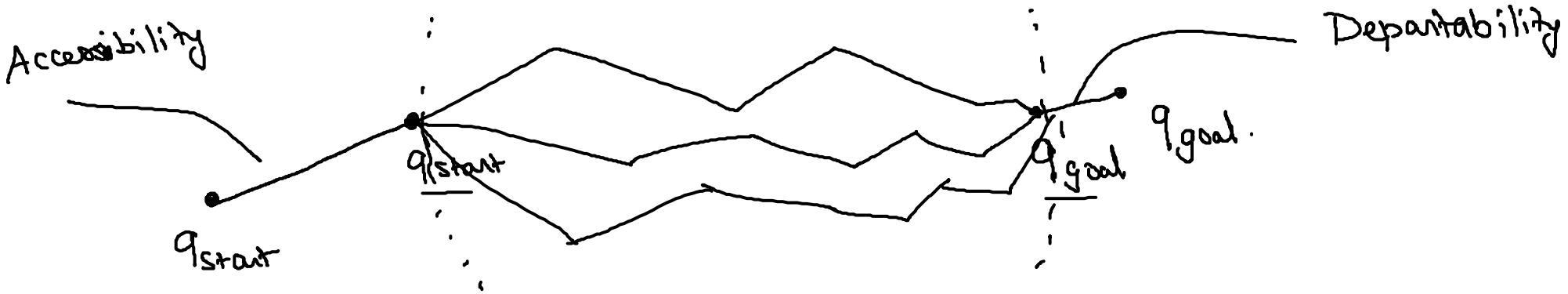
- *Definition of Roadmap:* A union of one-dimensional curves is a roadmap RM if for all q_{start} and q_{goal} in Q_{free} that can be connected by a path, the following properties hold:

1. **Accessibility**: *there exists a path from* $q_{\text{start}} \in Q_{\text{free}}$ *to some* $q'_{\text{start}} \in RM$,
2. **Departability**: *there exists a path from some* $q'_{\text{goal}} \in RM$ *to* $q_{\text{goal}} \in Q_{\text{free}}$, *and*
3. **Connectivity**: *there exists a path in* RM *between* q'_{start} *and* q'_{goal}.

Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

Roadmap Path Planning

1. **Accessibility**: *there exists a path from* $q_{start} \in Q_{free}$ *to some* $q'_{start} \in RM$,
2. **Departability**: *there exists a path from some* $q'_{goal} \in RM$ *to* $q_{goal} \in Q_{free}$, *and*
3. **Connectivity**: *there exists a path in RM between* q'_{start} *and* q'_{goal}.



Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

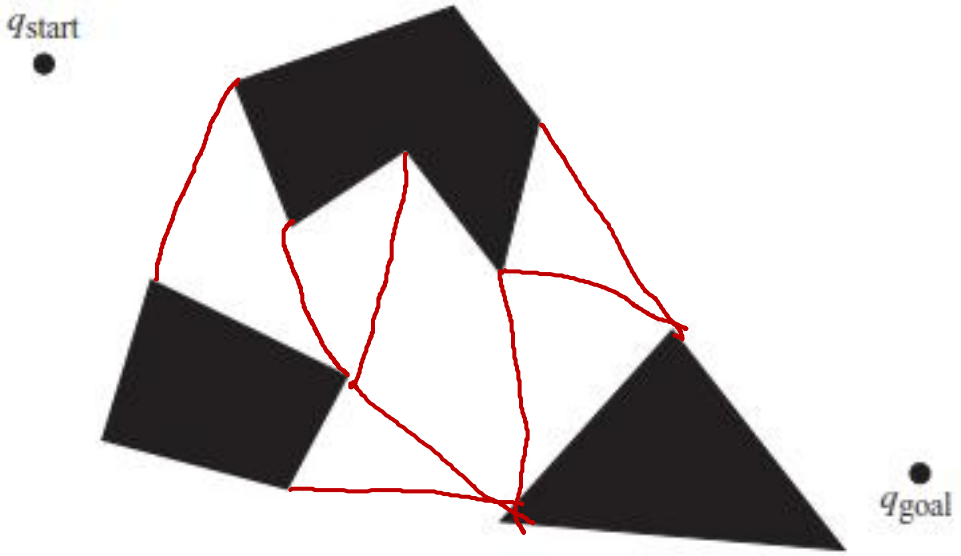
Roadmap Path Planning

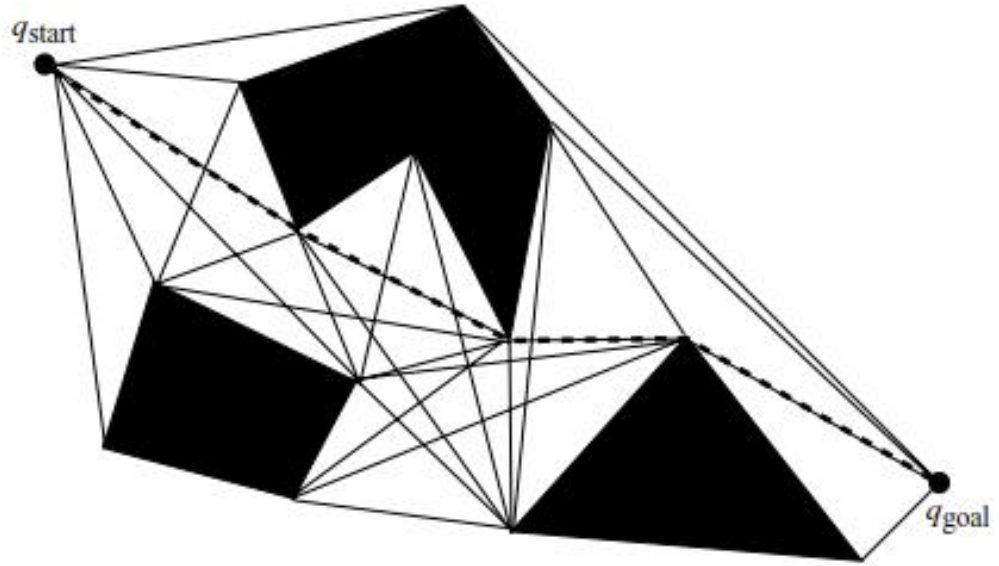
- Different types of Roadmaps exist:
- *Visibility maps, Deformation retracts, retract-like structures, piecewise retracts and silhouettes*
- We will focus on *Visibility maps*
- Visibility maps tend to apply to configuration spaces with polygonal obstacles.
- Nodes of the map are the vertices of the polygons and for visibility maps we can use the terms node and vertex interchangeably.
- Two nodes of a visibility map share an edge if their corresponding vertices are within line of sight of each other.

Roadmap Path Planning

- Nodes of the map are the vertices of the polygons and for visibility maps we can use the terms node and vertex interchangeably.
- Two nodes of a visibility map share an edge if their corresponding vertices are within line of sight of each other.

Q_{free}

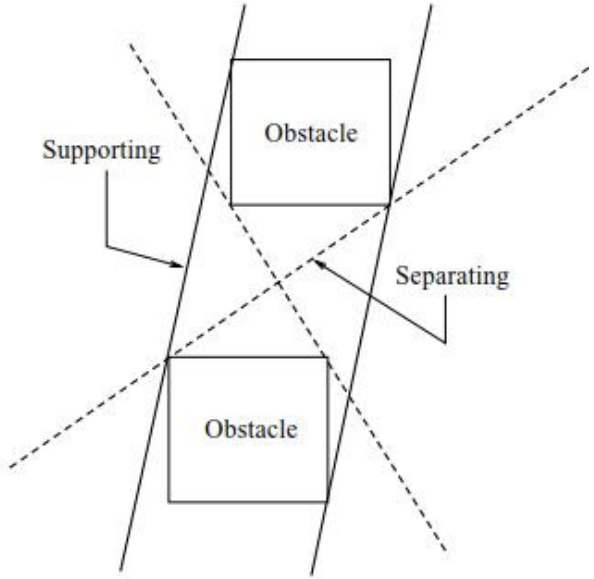




Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

Roadmap Path Planning

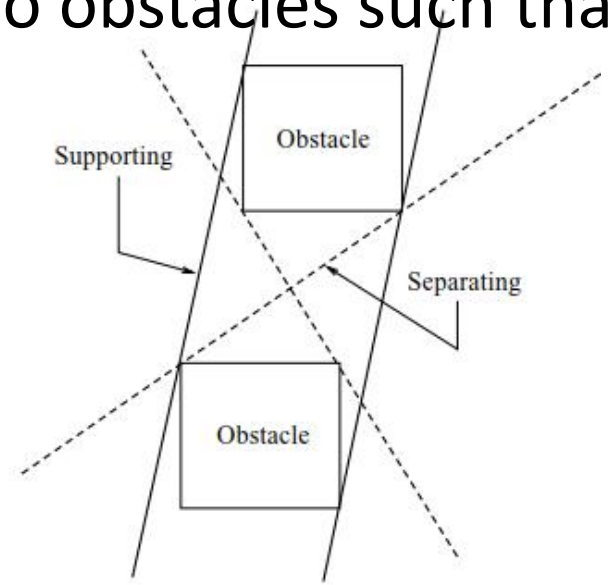
- All points in the robot's free space are within line of sight of at least one node on the visibility map
- This implies that visibility maps, by definition, possess the properties of accessibility and departability.
- We need to understand the concept of supporting lines and separating lines



Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

Roadmap Path Planning

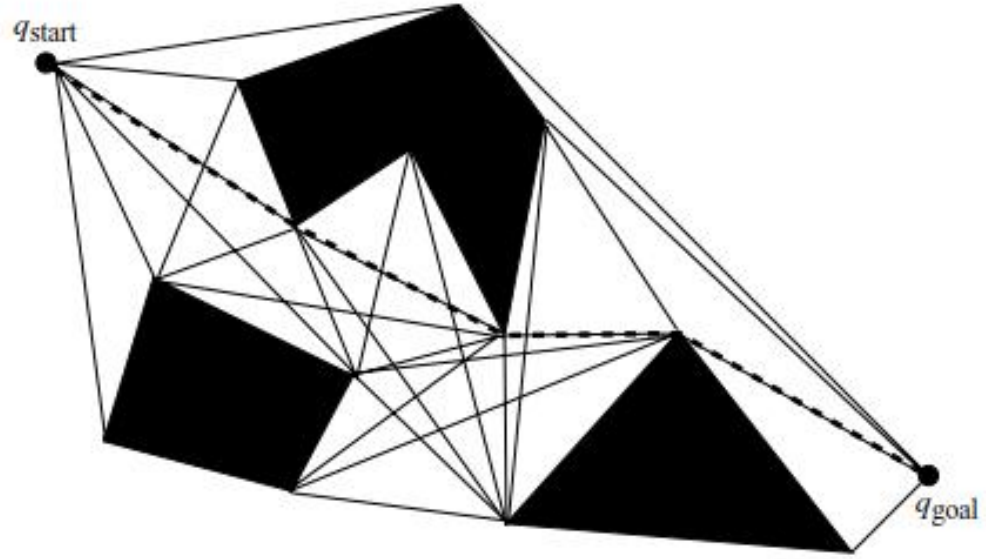
- A visibility graph has many edges, it needs to be reduced for computational purposes
- This is done using *Supporting lines* and *Separating lines*
- *Supporting line:* A supporting line is tangent to two obstacles such that both obstacles lie on the same side of the line.
- *Separating line:* A separating line is tangent to two obstacles such that the obstacles lie on opposite sides of the line



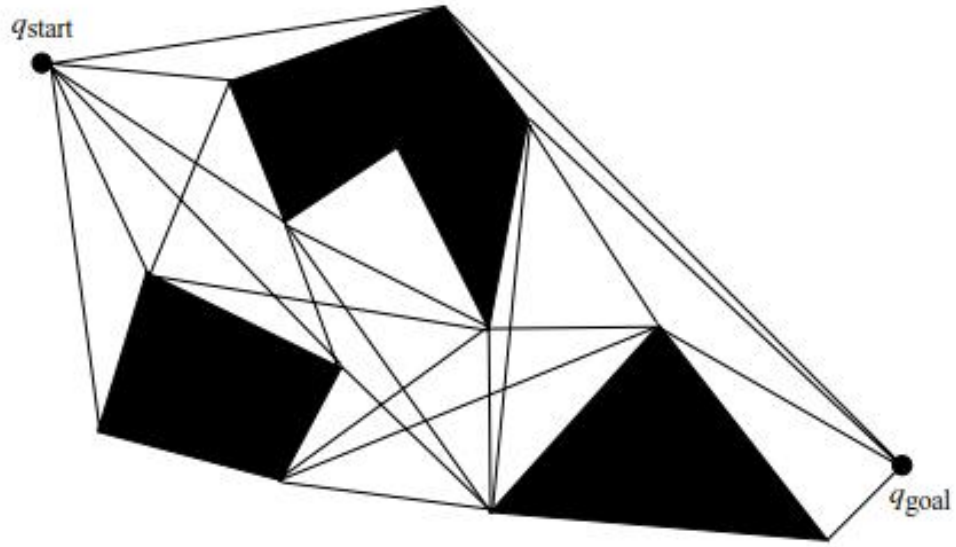
Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

Roadmap Path Planning

- The *reduced visibility graph* is solely constructed from supporting and separating lines.
- In other words, all edges of the original visibility graph that do not lie on a supporting or separating line are removed.



Visibility Graph



Reduced Visibility Graph

Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

Roadmap Path Planning

1. **Build the roadmap**

   a) nodes are points in Q_{free} (or its boundary)

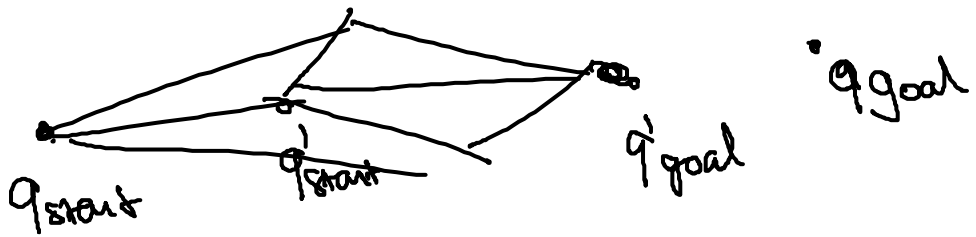
   b) two nodes are connected by an edge if there is a free path between them

2. **Connect start and goal points to the road map**

   at point q'_{start} and q'_{goal}, respectively

3. **Find a path on the roadmap betwen q'_{start} and q'_{goal}**

   The result is a path in Q_{free} from start to goal

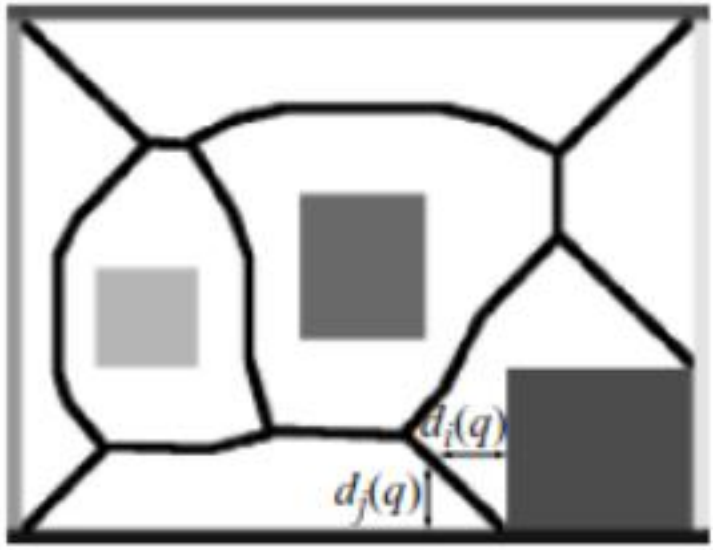


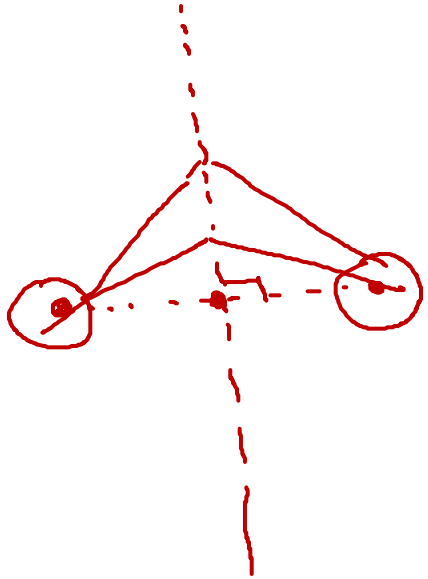
Roadmap Path Planning

- The visibility graph is constructed by first determining whether the vertices are in line-of-sight
- This is a computationally intensive process and there are different algorithms available to perform this in an efficient manner. E.g., plane-sweep algorithm
- The *visibility graph-based* approach moves very close to the objects and corners
- To overcome this issue, *Voronoi diagrams* are instead made use of in *Deformation retracts* approach

Roadmap Path Planning

- The *generalized Voronoi diagram (GVD)* is the set of points where the distance to the two closest obstacles is the same





Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

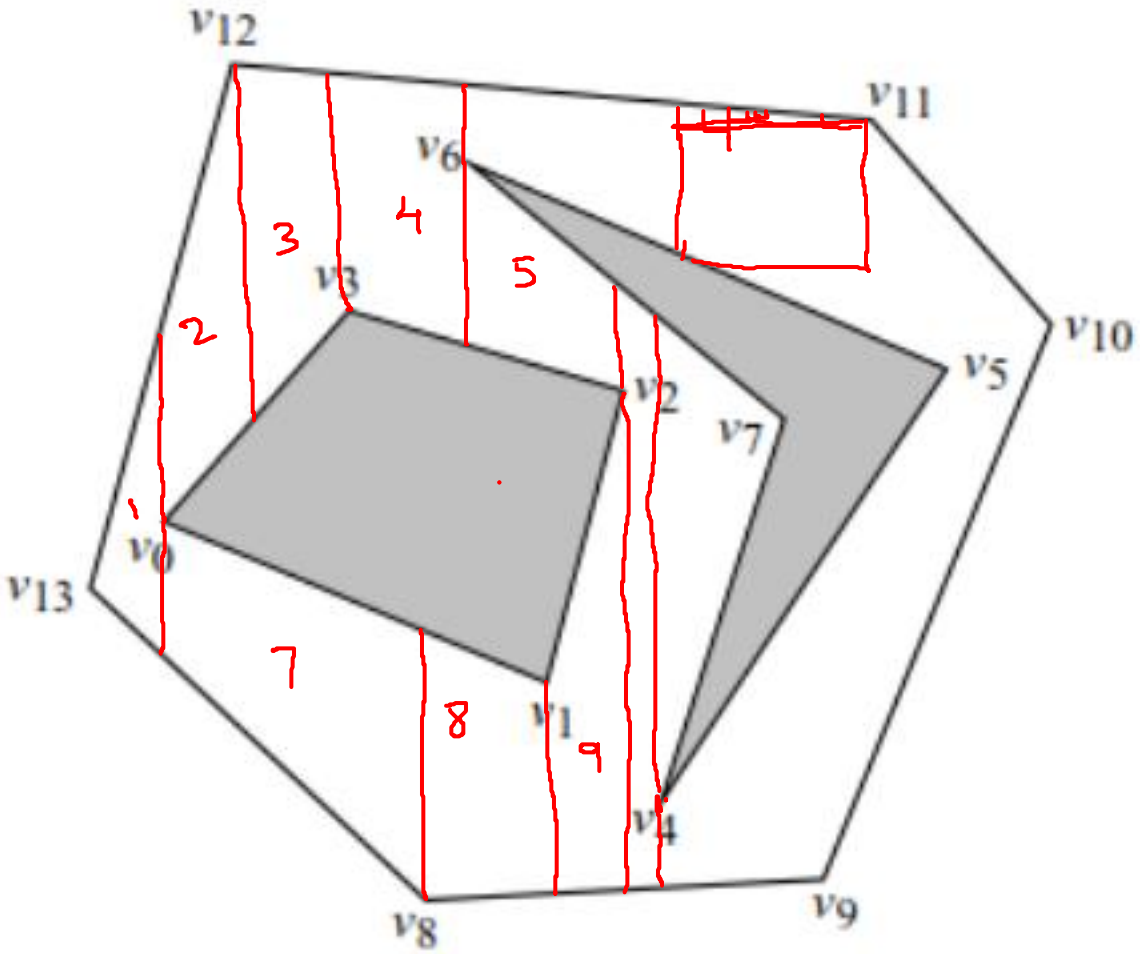
Autonomous Mobile Robots

Module 33: Cell Decomposition
Path Planning

Cell Decomposition Path Planning

- In this method, we decompose the free space into cells
- If we are able to decompose the free space in an exact manner, it is called as *Exact Cell Decomposition*
- If we can only decompose the free space in an approximate manner, it is called as *Approximate Cell Decomposition*
- Exact Cell Decomposition uses variable trapezoid shapes, while Approximate cell decomposition uses fixed square-like cells
- Union of the cells creates the entire free space

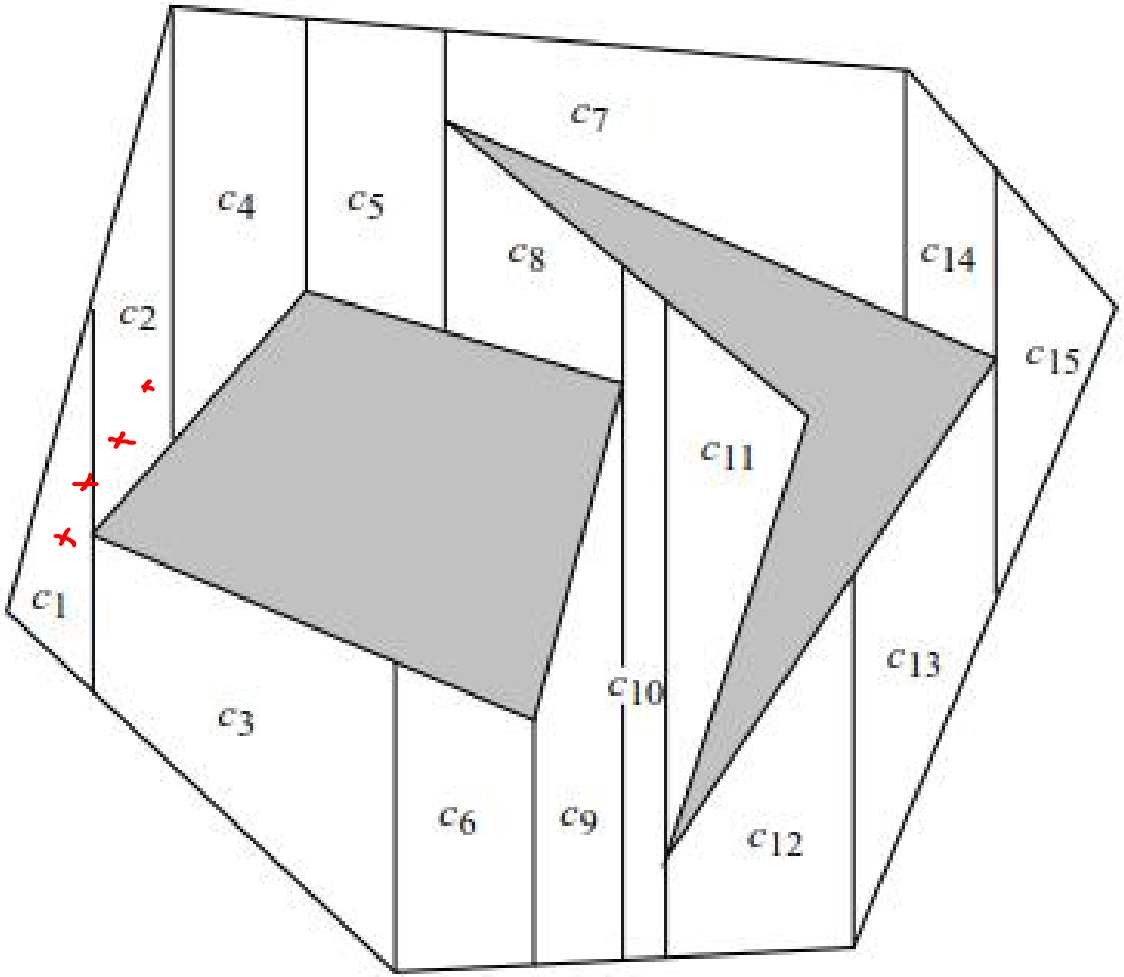
Exact Cell Decomposition



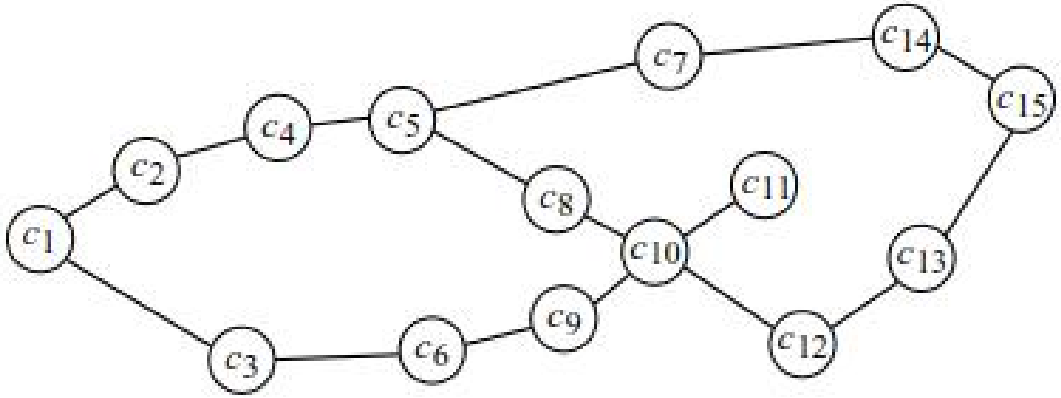
- Draw vertical lines wherever corners are present and divide free space
- Number the cells that are created in this manner

Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

Exact Cell Decomposition



Adjacency Graph

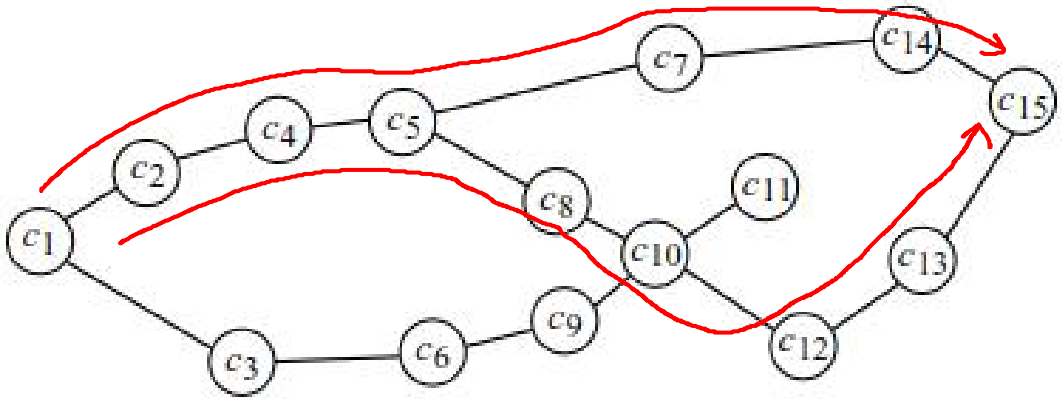


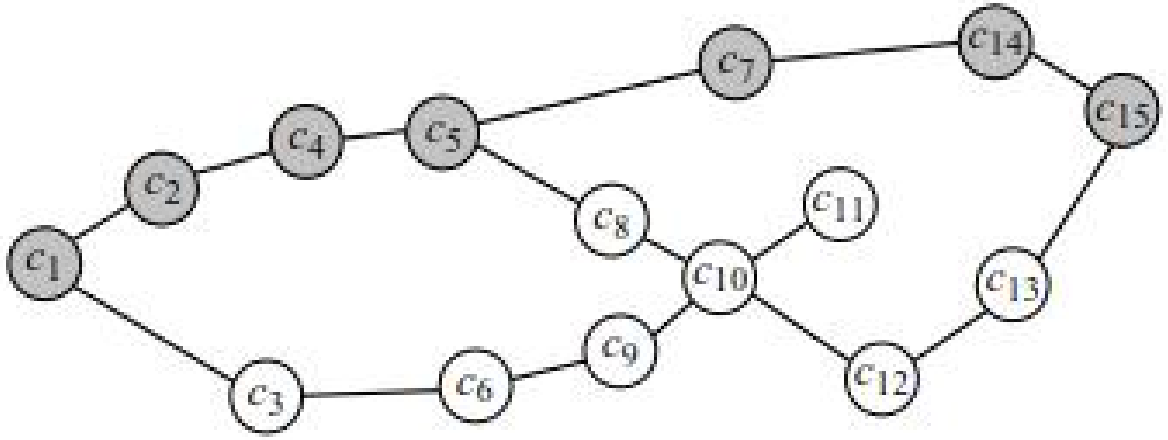
Search for path in Adjacency Graph using graph search techniques

Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

Exact Cell Decomposition

Suppose we need to plan a path from c_1 to c_{15}

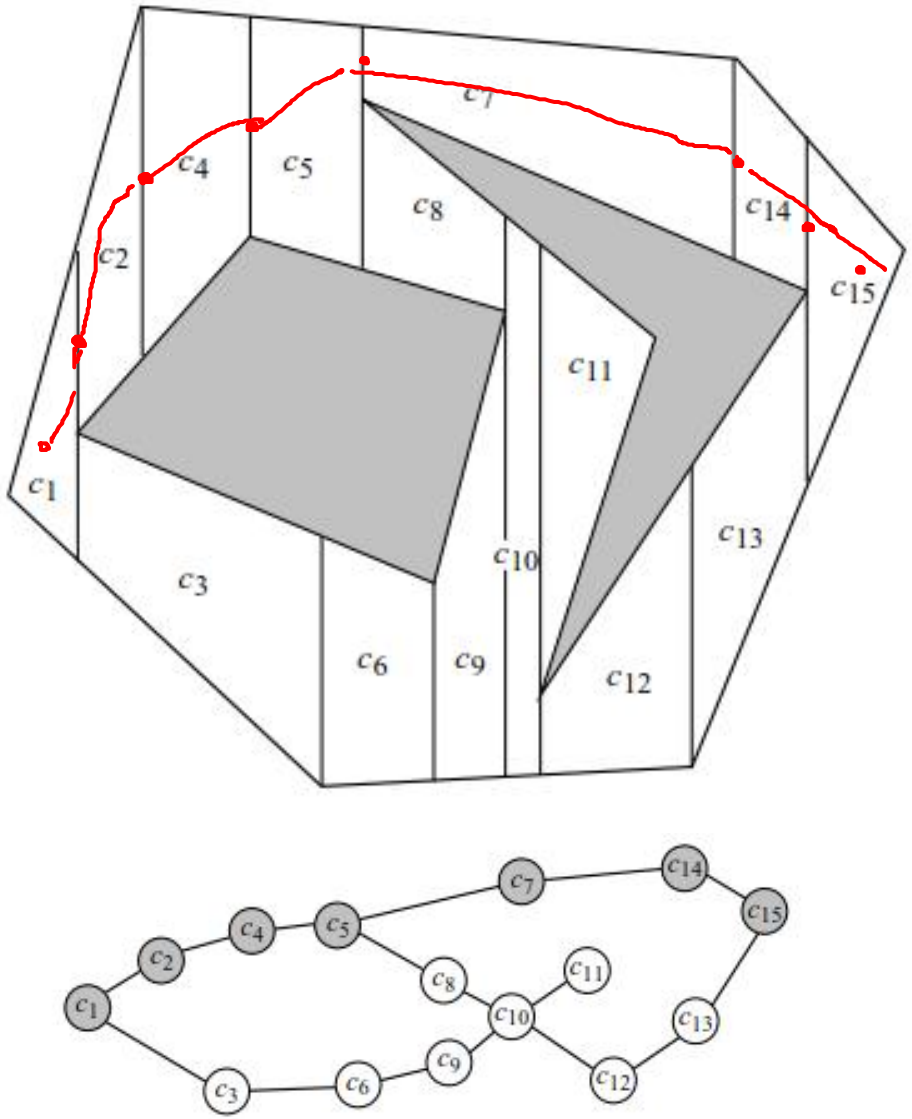


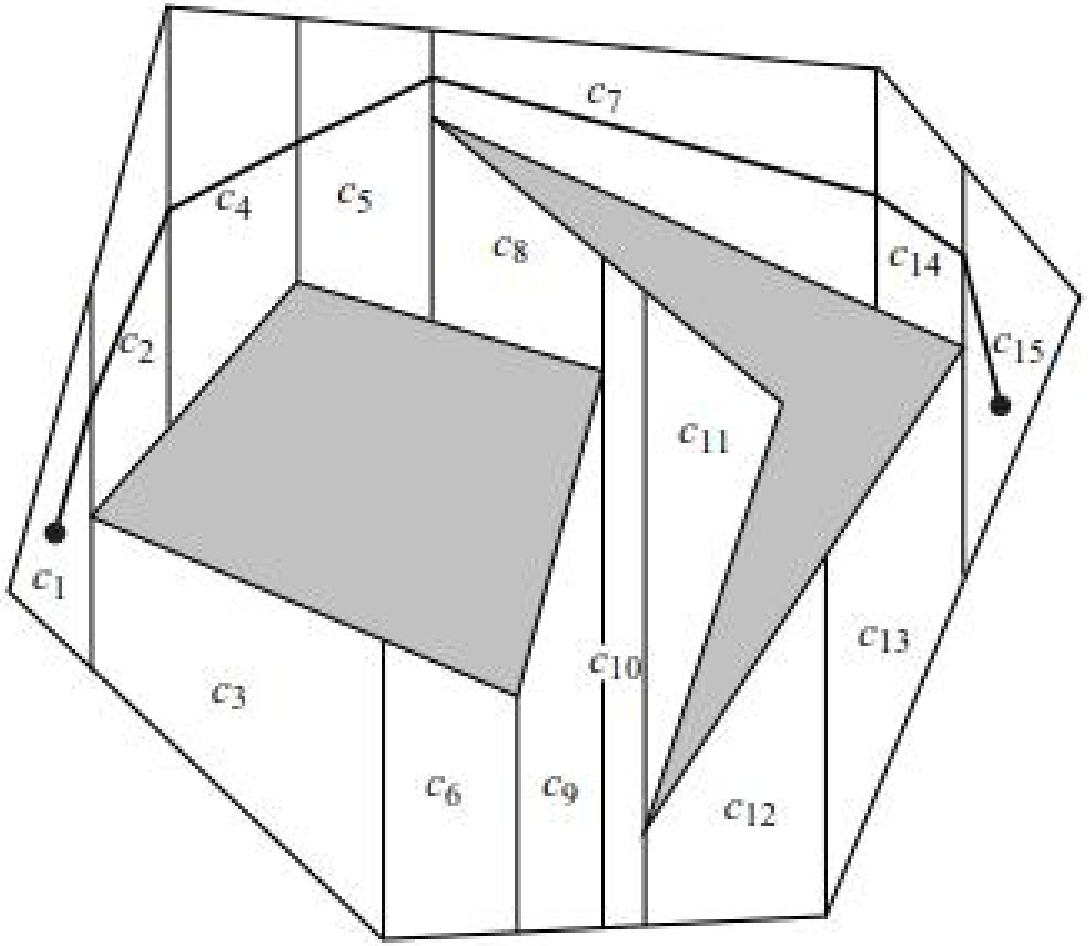


- Search for path in Adjacency Graph using graph search techniques
- Search solution will only give a sequence of cells, but does not provide path points for robot to move
- Midpoint of the boundary line is considered as the path point for the robot

Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

Exact Cell Decomposition

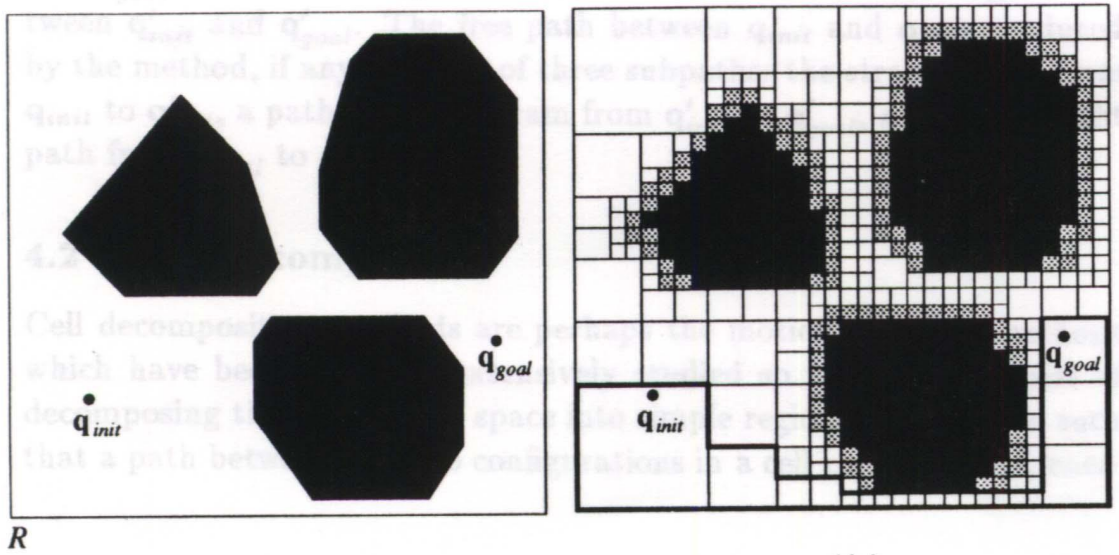




Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

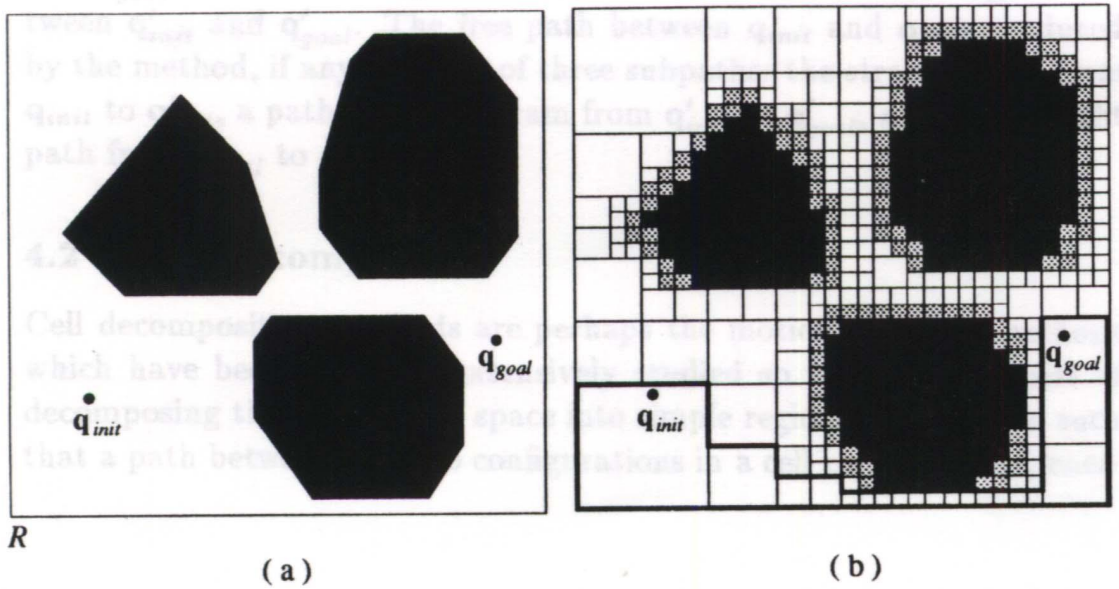
Exact Cell Decomposition

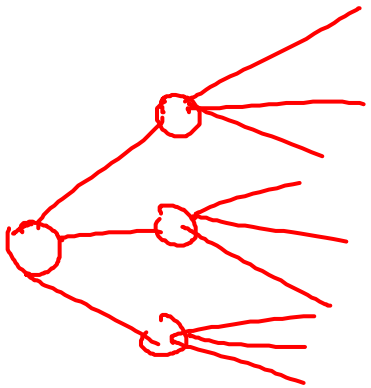
- Exact Cell Decomposition requires an algorithm for decomposing free space into trapezoids
- Approximate Cell Decomposition methods can also be used, where the cell shape is fixed as a square, but size can be changed



Source: https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/basicmotion.html

Exact Cell Decomposition

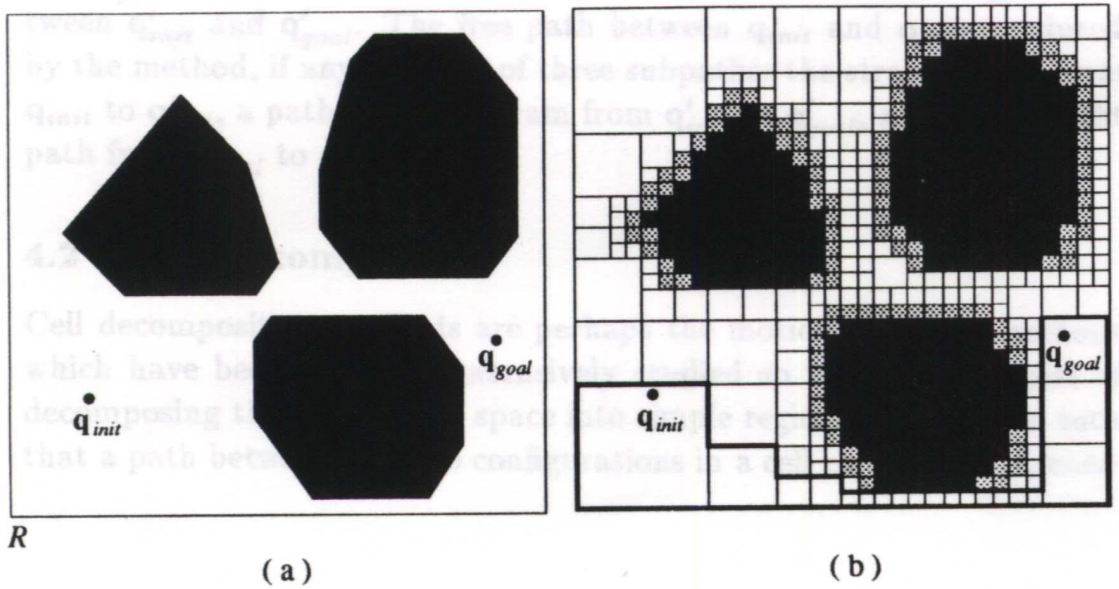


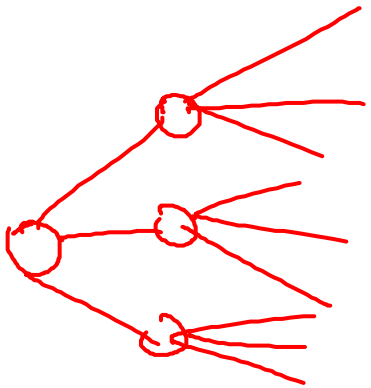


- More number of cells will be used in this case, in comparison to Exact method
- More computations will therefore be required
- However, the decomposition process can be simpler

Source: https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/basicmotion.html

Exact Cell Decomposition





- More number of cells will be used in this case, in comparison to Exact method
- More computations will therefore be required
- However, the decomposition process can be simpler

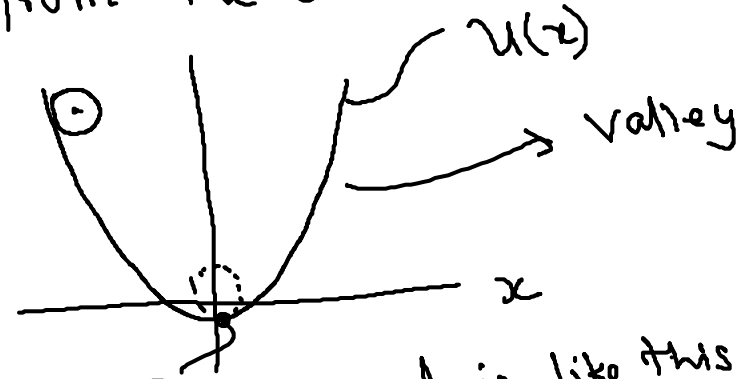
Source: https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/basicmotion.html

Autonomous Mobile Robots

Module 34: Potential Field Path Planning

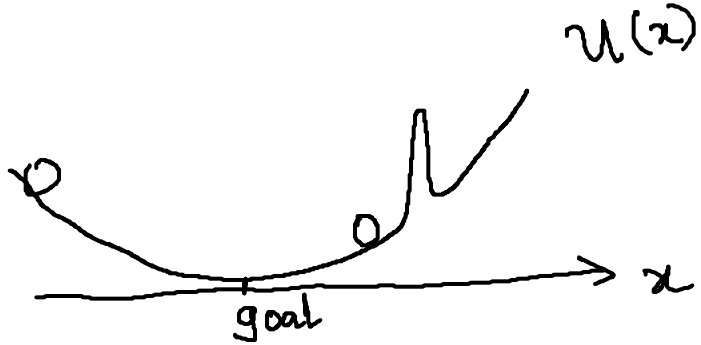
Potential Field Path Planning

Simple idea: Configuration space is assumed to be an artificial potential that draws the robot towards the goal & also drives it away from the obstacles



Suppose goal is like this configuration : goal is attracting the robot.

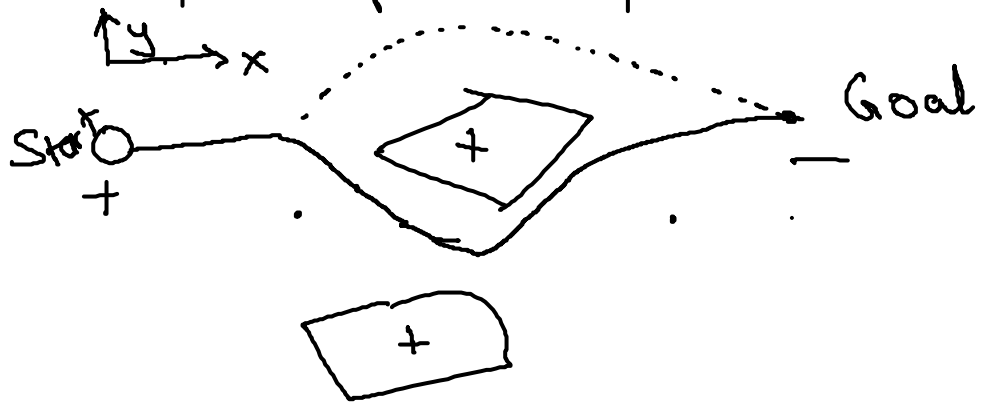
Think of obstacles as peaks which prevent robot from going near it.



Potential Field Path Planning

Here we are thinking of movement from a higher point in space to a lower point

But in reality it is moving from a point of higher potential energy to a point of lower potential energy.



$z = U(x,y)$; $y = U(x)$

Visualize potential functions for 2D space

- Can also think of interaction as charge based
- Robot & Obstacles have +ve charge & goal has −ve charge
- Essentially we are thinking of potential energy

Potential Field Path Planning

Definition: A potential function is a differentiable real-valued function U

$U: \mathbb{R}^m \rightarrow \mathbb{R}$. For 2D space, $m = 2$

$U \cdot \mathbb{R}^2 \rightarrow \mathbb{R}$

$(x,y) \xrightarrow{U}$

$(3,5) \qquad 17$

The value of the potential function can be thought of as the energy.

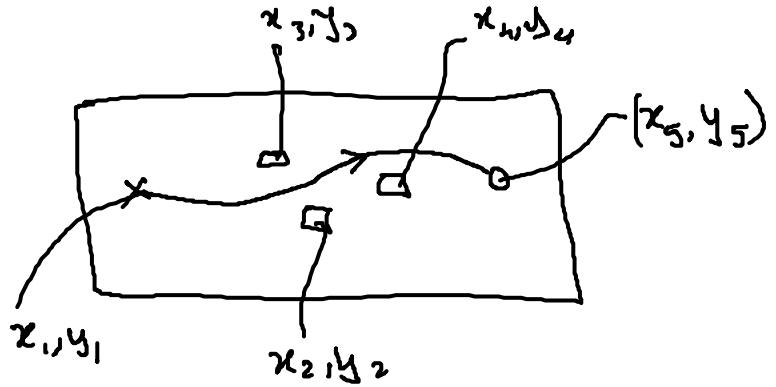
The gradient of potential energy is like force

Gradient is a vector $\nabla U(q) = D U(q)$

$$= \left[\frac{\partial U}{\partial q_1}, \frac{\partial U}{\partial q_2}, \ldots \frac{\partial U}{\partial q_m} \right]$$

for $m = 2$ $\quad \nabla U(x,y) = \left[\frac{\partial U}{\partial x}, \frac{\partial U}{\partial y} \right]$

$$\frac{\partial U}{\partial x}\hat{i} + \frac{\partial U}{\partial y}\hat{j}$$



$U = U(x) \qquad U(x,y)$

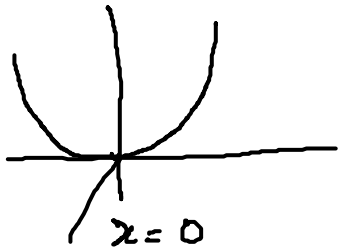
$\frac{du}{dx} \qquad \frac{\partial U}{\partial x} \qquad \frac{\partial U}{\partial y}$

$q = (q_1, q_2, q_3 \ldots q_m)$

$= (x,y)$

Potential Field Path Planning

$m = 1$ $\quad U = kx^2$

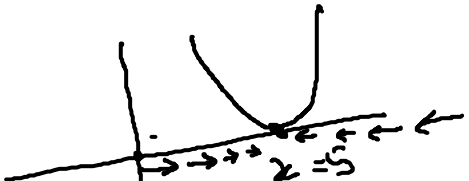


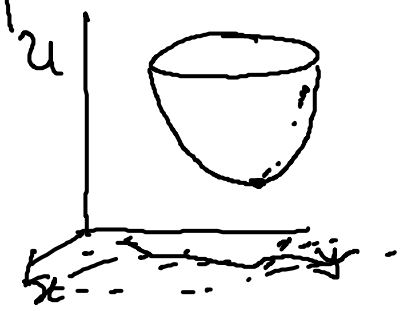
goal is at origin

Gradient : $\frac{du}{dx} = \frac{\partial u}{\partial x} = 2kx \sim$ Force

If x is large → Force is large
x is small → " " small

Suppose goal at $x = 5$, consider $U = k(x-5)^2$

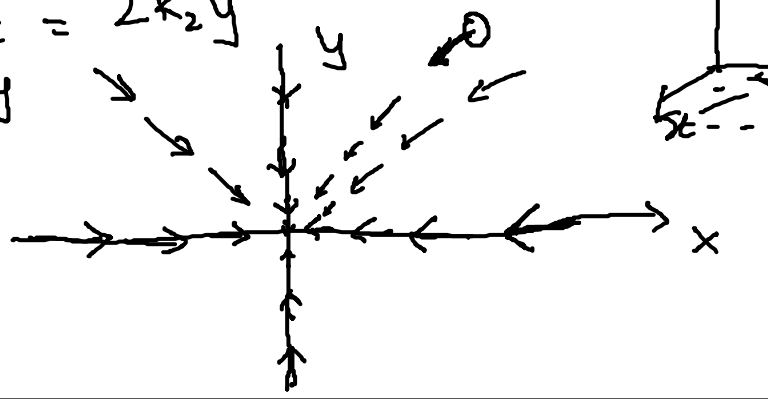




Case : $m = 2$ $\quad U = k_1 x^2 + k_2 y^2$

Gradient : $\frac{\partial u}{\partial x} = 2k_1 x$; $\frac{\partial u}{\partial y} = 2k_2 y$

$2k_1 x \hat{i} + 2k_2 y \hat{j}$



Potential Field Path Planning

Potential function approach directs a robot as though it is a particle moving in a gradient vector field.

As the robot follows a path towards lower energy, it goes downhill on the potential curve & towards the goal in 2D world

This type of path following is called as gradient - descent where robot's movement depends on the negative gradient value computed at any particular point

Attractive and Repulsive Potential Field

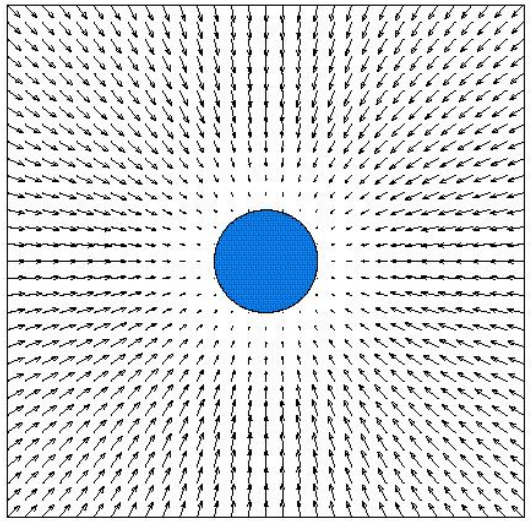
Attractive Potential Field

- Pulls towards itself
- Attracts objects farther away strongly

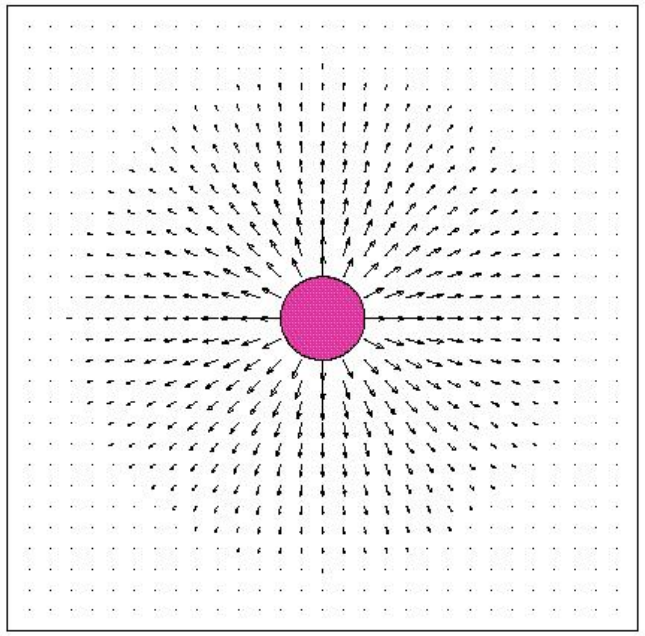
Repulsive Potential Field

- Repels strongly nearby
- Repels weakly farther away

Attractive Potential



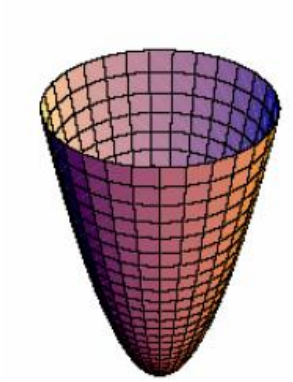
Gradient Vector Field



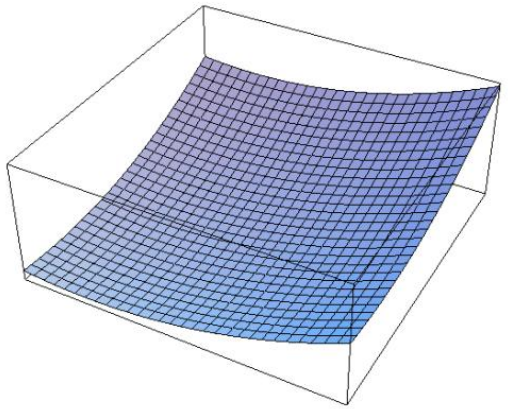
Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

Attractive Potential Field

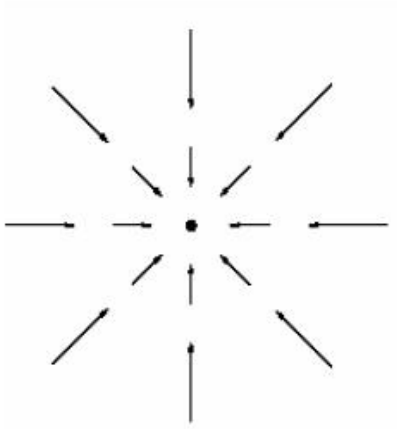
Attractive Potential Function (Quadratic)



Slow-varying quadratic

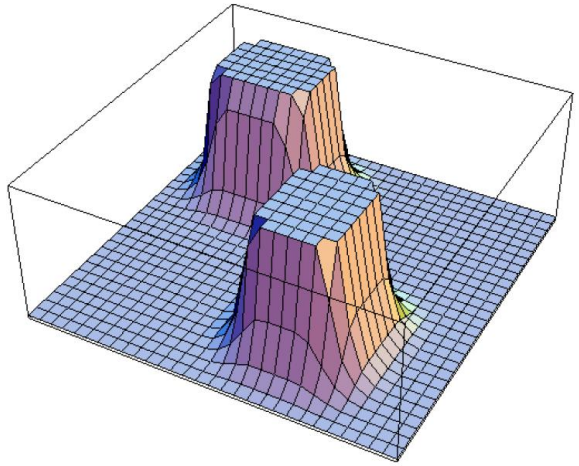


Gradient



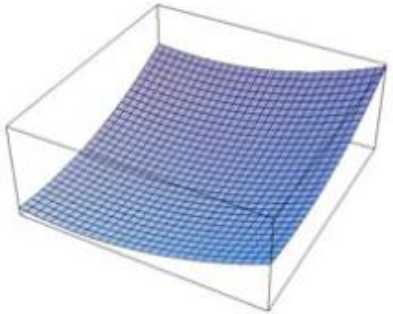
Reproduced from: Choset, Howie et al. Principles of Robot Motion Theory, Algorithms, and Implementation . Bradford Books, 2005.

Repulsive Potential Field

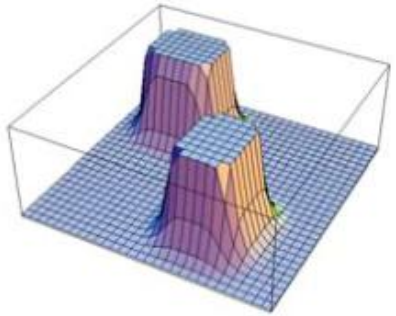


- Attractors are like valleys
- Repellers are like hills
- Add both potentials and compute the gradient

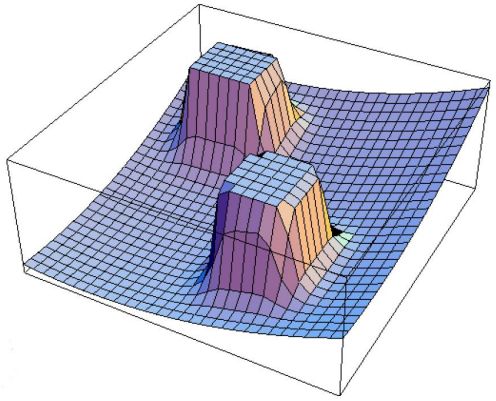
Total Potential given as $U(q) = U_{att}(q) + U_{rep}(q)$



+



=

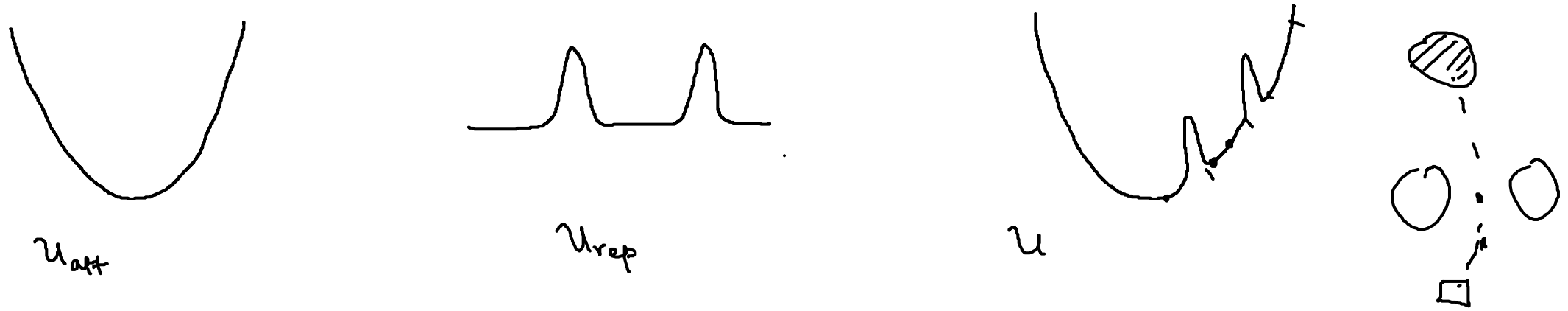


Adapted from slides by Howie Choset, with slides from Ji Yeong Lee, G.D. Hager and Z. Dodds

Potential Field Path Planning

- The potential field path planning method depends on identification of location of the goal point and obstacles
- This should be possible based on the perception scheme employed by the robot
- An artificial potential function is then created based on the locations, with an attractive potential created at the goal and a repulsive potential created at the obstacles
- The sum of the potentials is calculated
- The gradient of the total potential is calculated and robot is moved based on a gradient descent scheme which will take it to the goal
- One issue with this scheme is the formation of local minima in the space which will prevent the robot from reaching the goal

Potential Field Path Planning



- Unlike other roadmap, cell decomposition or A* methods, this method can operate fully online, with no need for offline exploration and computation

Autonomous Mobile Robots

Module 35: Obstacle Avoidance & Path control

Obstacle Avoidance

- Local obstacle avoidance focuses on changing the robot's trajectory based on sensor readings during robot motion
- The resulting robot motion is both a function of the robot's current or recent sensor readings and its goal position and relative location to the goal position.
- The obstacle avoidance algorithms depend to varying degrees on the existence of a global map and on the robot's precise knowledge of its location relative to the map.
- We will look at some simple algorithms that have been successfully used on robots, such as
- Bug 0
- Bug 1
- Bug 2
- Vector Field Histogram (VFH)

Obstacle Avoidance

- The Bug algorithm represents such a technique in that only the most recent robot sensor values are used, and the robot needs, in addition to current sensor values, only approximate information regarding the direction of the goal.
- Other algorithms may take into account recent sensor history, robot kinematics, and even dynamics.

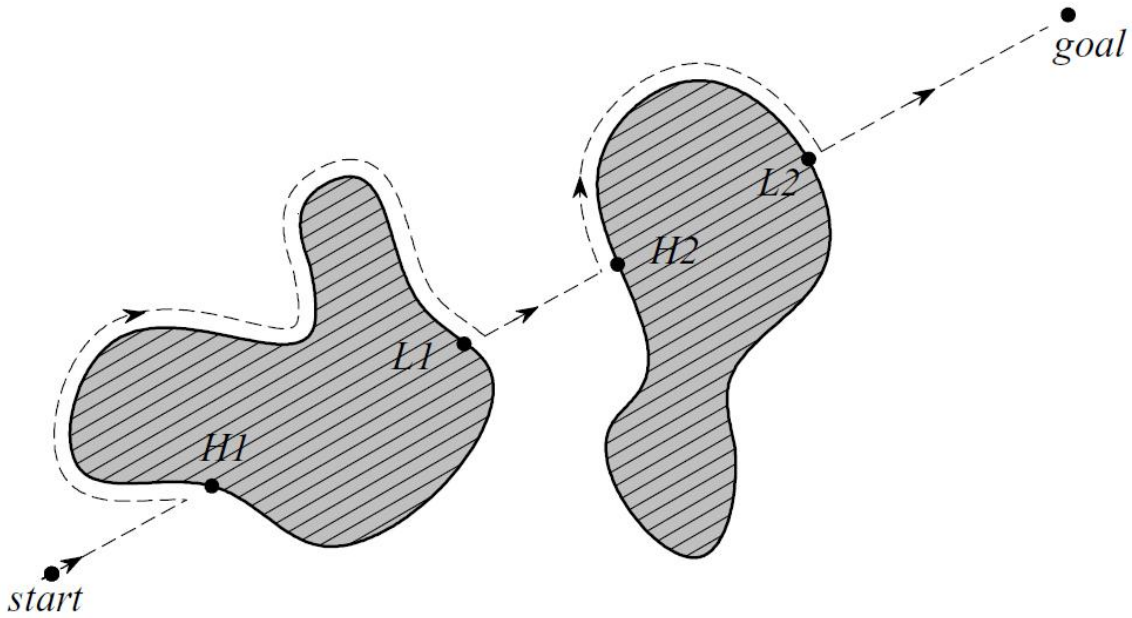
Obstacle Avoidance - Bug algorithms

- Some planning algorithms require global knowledge of the environment (full maps), and make plans based on full information

- Bug algorithms use only local knowledge of the environment along with information about global goal
- These algorithms are insect inspired
- Bugs execute simple behaviors
- 1) Follow a wall (right or left)
- 2) Move in a straight line toward goal
- Usually assume that only right wall following or left wall following is possible

Common Definitions for Bug algorithms

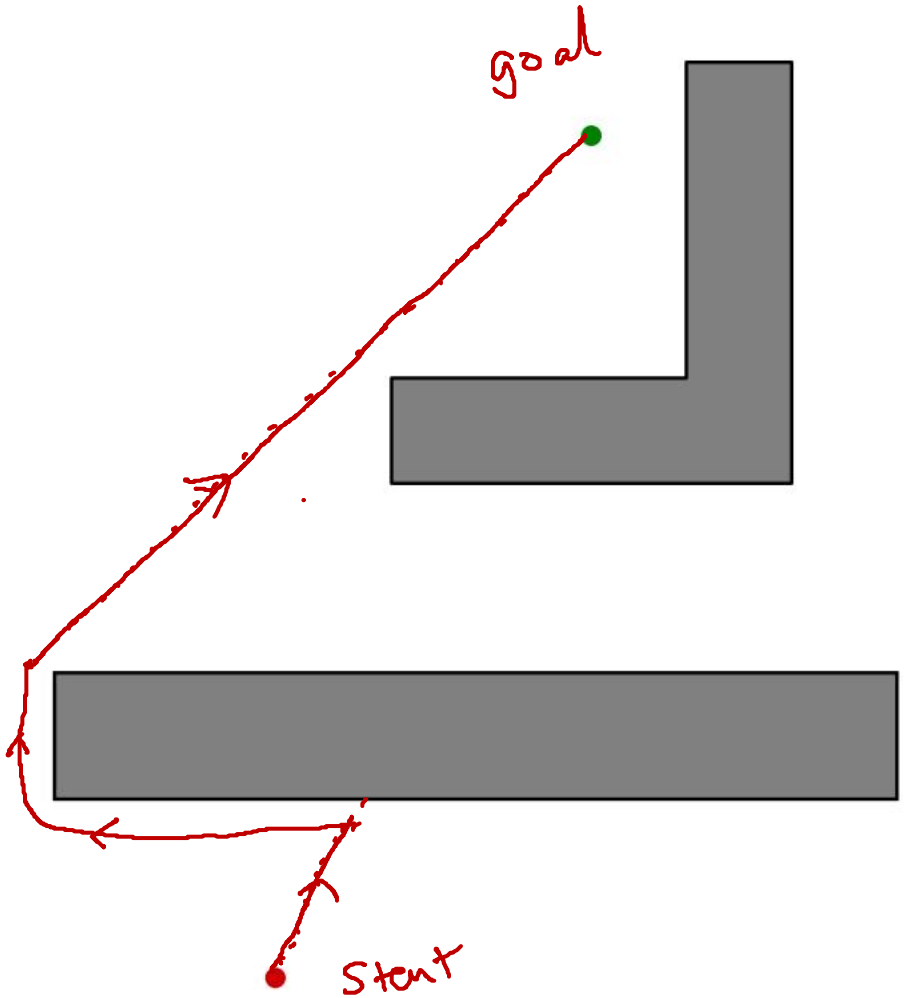
- Start point
- Goal point
- Hit point – H1, H2
- Leave point – L1, L2
- Right wall following



Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Bug 0 Algorithm

- Assume you know direction to goal
- Sense local information about walls/obstacles



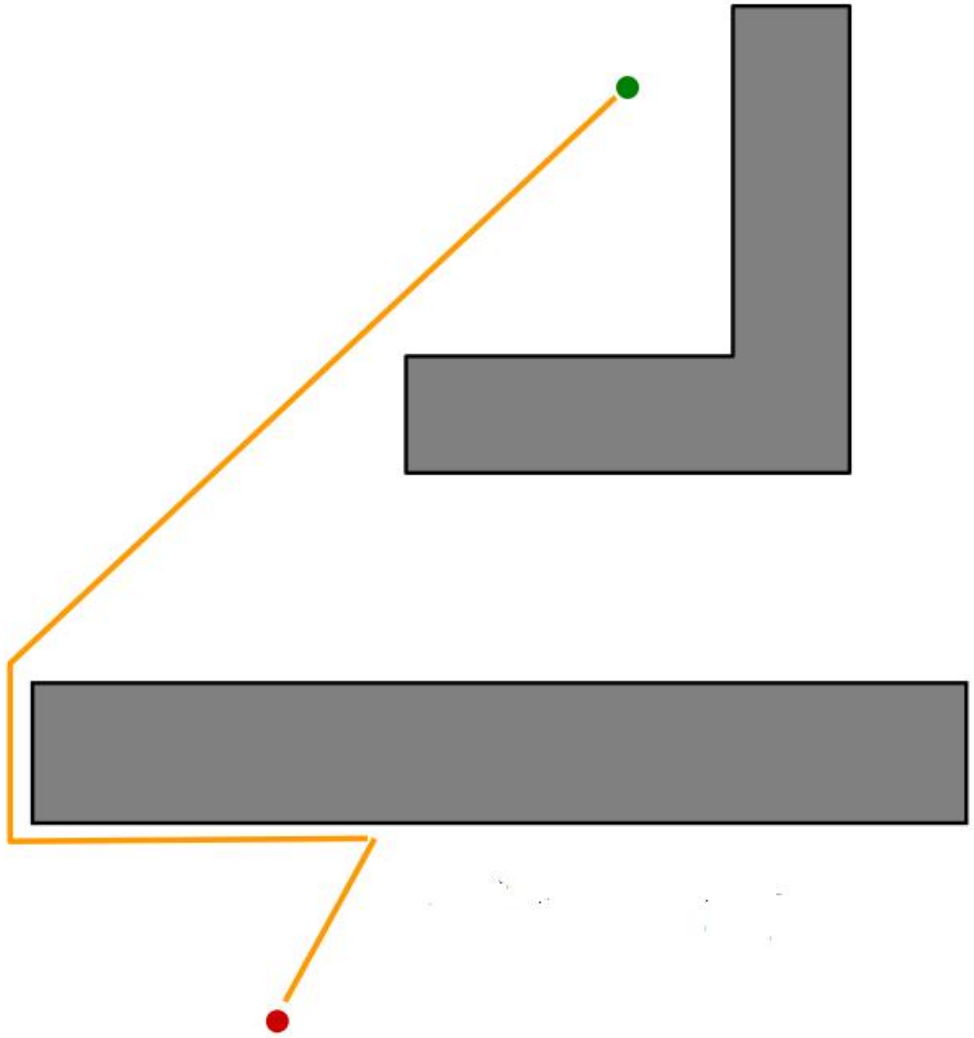
Algorithm

- Head from start point towards goal
- If you encounter obstacle start following it till you can get a clear path to goal
- Continue on clear path till goal is reached or next obstacle is encountered
- Repeat

https://www.cs.cmu.edu/~motionplanning/ - Slides from Howie Choset, G.D. Hager and Z. Dodds

Bug 0 Algorithm

- Assume you know direction to goal
- Sense local information about walls/obstacles



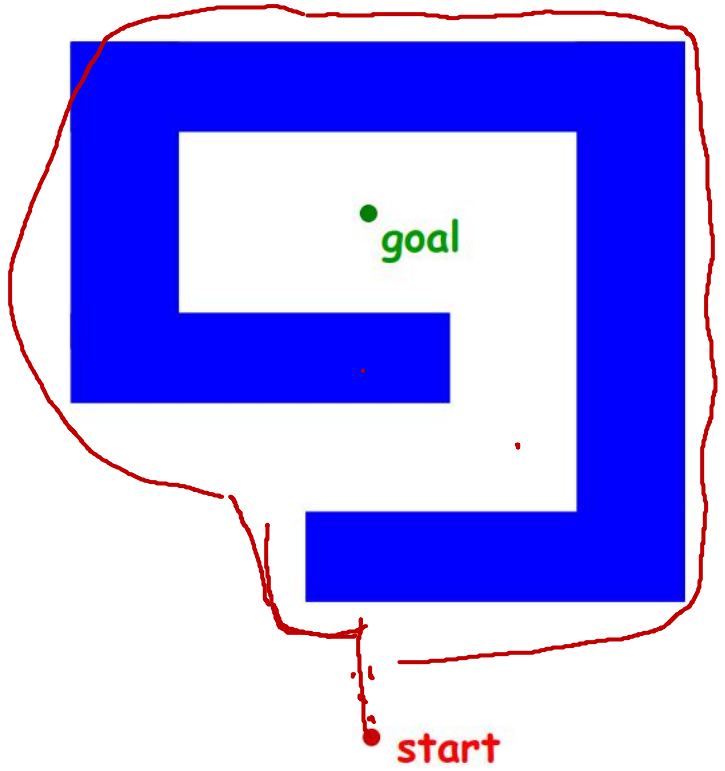
Algorithm

- Head from start point towards goal
- If you encounter obstacle start following it till you can get a clear path to goal
- Continue on clear path till goal is reached or next obstacle is encountered
- Repeat

https://www.cs.cmu.edu/~motionplanning/ - Slides from Howie Choset, G.D. Hager and Z. Dodds

Bug 1 Algorithm

- Bug 0 is a no-memory approach, does not guarantee that goal will be reached
- Can have situations where robot will be looping same path without reach goal



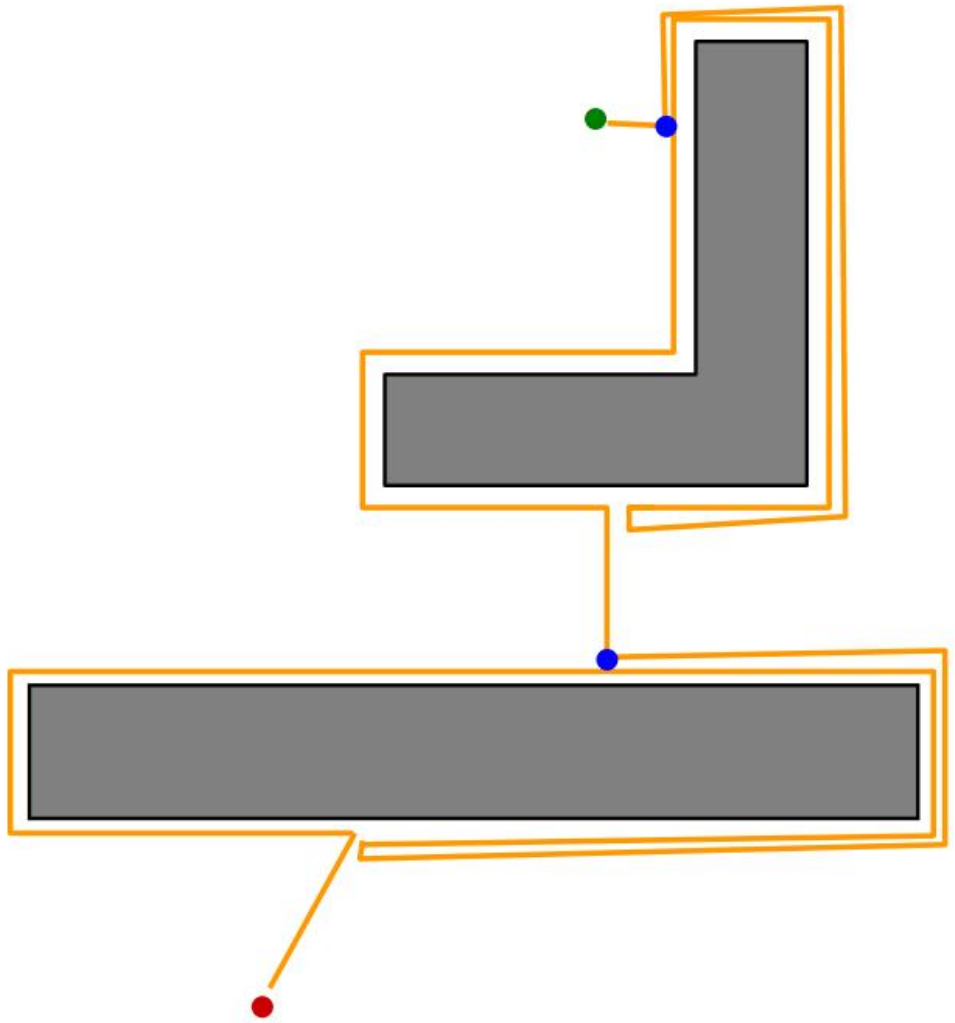
- Use some memory – remember point closest to goal

https://www.cs.cmu.edu/~motionplanning/

Bug 1 Algorithm

- Head from start point towards goal
- If you encounter obstacle circumnavigate it (fully circle it once), while doing this, remember point closest to goal
- After completing circumnavigation, return to that point by shortest path
- Leave at that point and head towards goal
- Continue

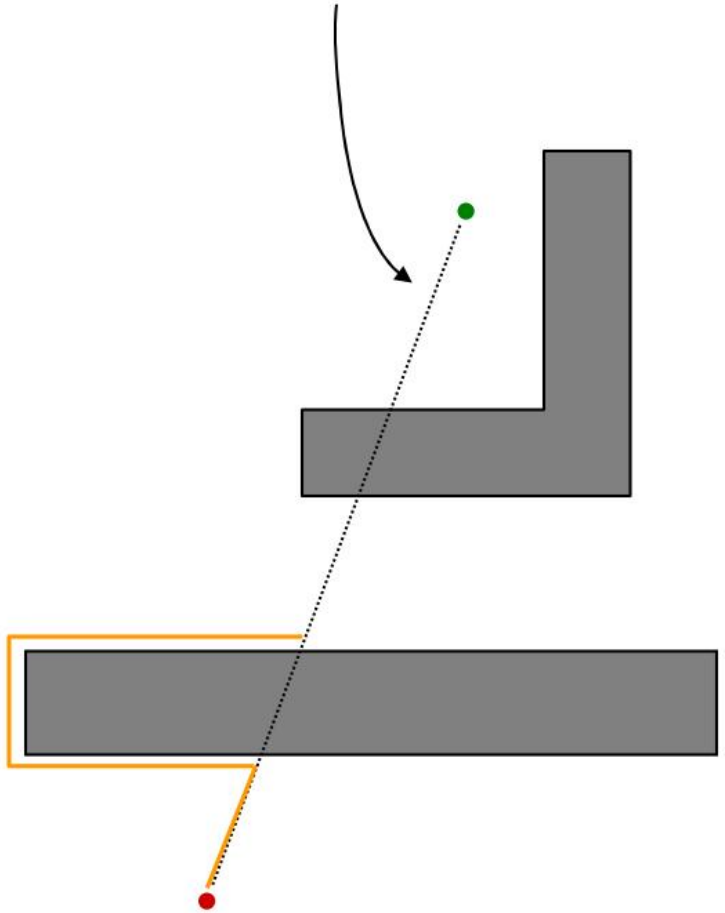
Ensures reaching goal, but is inefficient



https://www.cs.cmu.edu/~motionplanning/ - Slides from Howie Choset, G.D. Hager and Z. Dodds

Bug 2 Algorithm

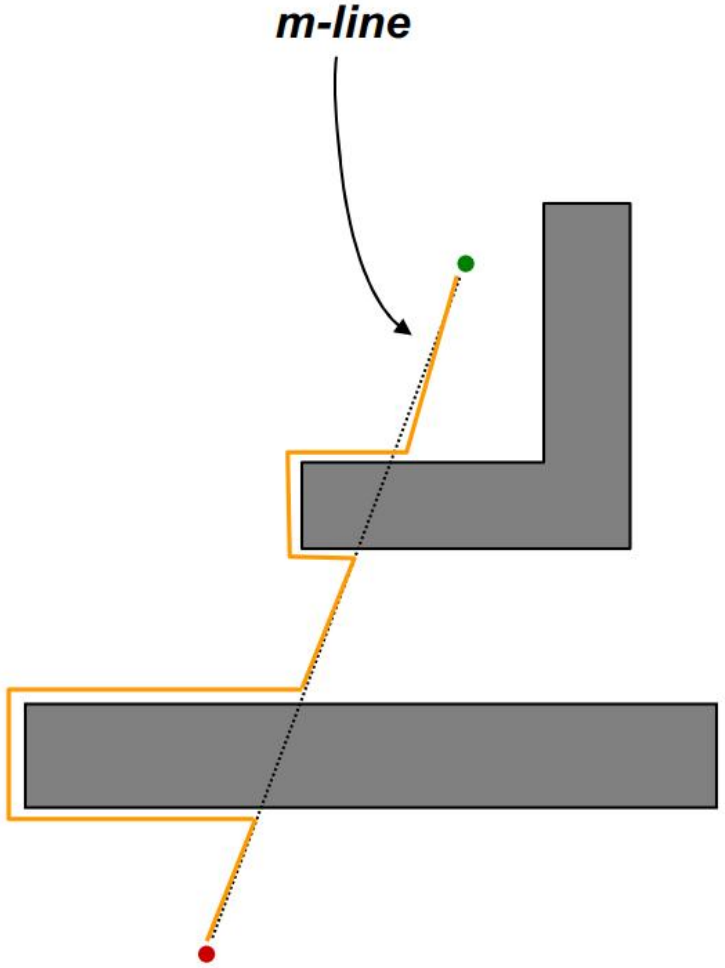
Call the line from the starting point to the goal the ***m-line***



- Head from start point towards goal on the m–line
- If you encounter obstacle start following it till you get back to the m-line
- Leave obstacle and head towards goal on m-line
- Repeat

https://www.cs.cmu.edu/~motionplanning/

Bug 2 Algorithm

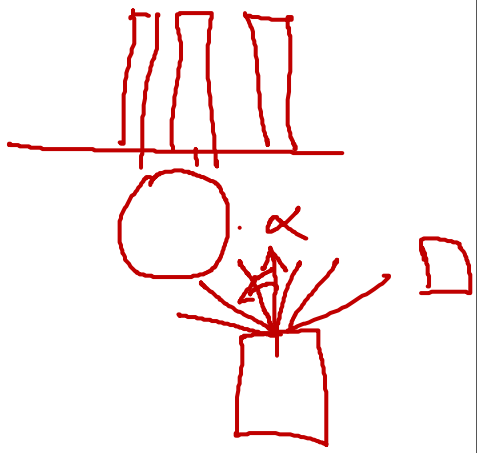


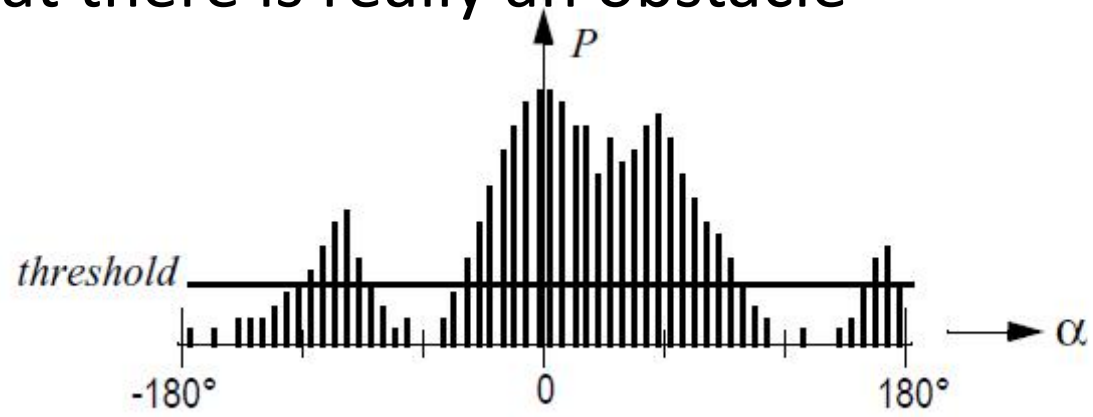
- Head from start point towards goal on the m–line
- If you encounter obstacle start following it till you get back to the m-line
- Leave obstacle and head towards goal on m-line
- Repeat

https://www.cs.cmu.edu/~motionplanning/

Vector Field Histogram

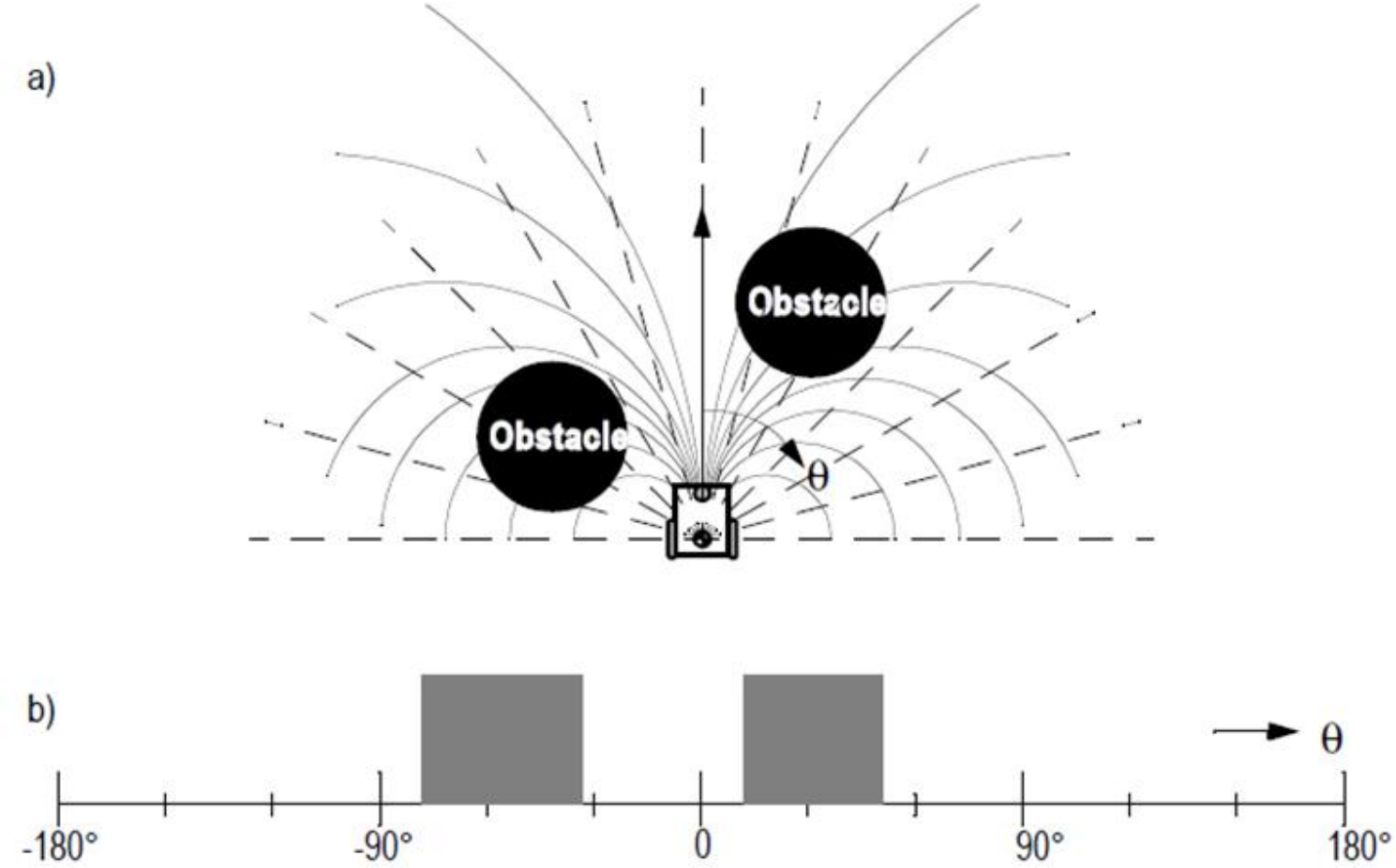
- Takes into account sensor readings collected over a period of time, instead of using latest sensor readings
- Bug algorithms only use latest sensor readings
- Instantaneous sensor readings can be noisy => not robust
- VFH creates a local map of the environment around the robot
- Generate a polar histogram
- α is the angle and P is the probability that there is really an obstacle in that direction





Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

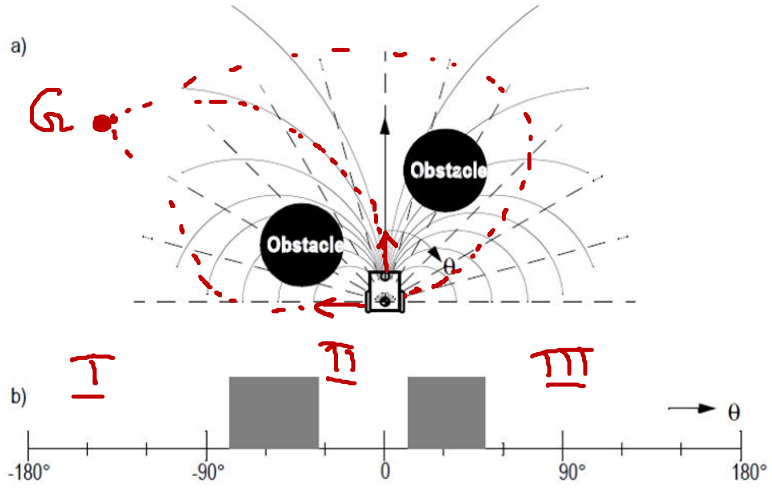
Vector Field Histogram



Reproduced from: Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

Vector Field Histogram

- From the histogram, steering direction is determined
- All openings large enough for the vehicle to pass are identified
- Cost function applied to every such opening
- Opening with lowest cost is chosen



$$G = a \cdot \text{target\_direction} + b \cdot \text{wheel\_orientation} + c \cdot \text{previous\_direction}$$

target_direction = alignment of the robot path with the goal;

wheel_orientation = difference between the new direction and the current wheel orientation;

previous_direction = difference between the previously selected direction and the new direction.