Localisation &
Motion Planning + Control

"Are we there yet?"

– Anon

METR 4202: Advanced Control & **Robotics**

Dr Surya Singh -- Lecture # 9

September 24, 2014

metr4202@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~metr4202/

Schedule

Week	Date	Lecture (W: 11:10-12:40, 24-402)
1	30-Jul	Introduction
2	6-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	13-Aug	Robot Kinematics (& Ekka Day)
4	20-Aug	Robot Dynamics & Control
5	27-Aug	Robot Motion
6	3-Sep	Robot Sensing: Perception & Multiple View Geometry
7	10-Sep	Robot Sensing: Features & Detection using Computer Vision
8	17-Sep	Navigation (+ Prof. M. Srinivasan)
9	24-Sep	**Localization & Motion Planning + Control**
	1-Oct	*Study break*
10	8-Oct	State-Space Modelling
11	15-Oct	Shaping the Dynamic Response
12	22-Oct	Linear Observers & LQR
13	29-Oct	Applications in Industry & Course Review

Announcements: Lab 2 Extended

- **October 6: Holiday (?)**
- **Lab 2:**
  - Extended to October 6/8
  - Signup in order of Check off
- **IROS 2014:**
  - **CHARM was a Charm!**
  - **Top 2 Keywords**: Computer Vision, Motion Planning
  - Emphasis on pattern recognition and manipulation
  - Future is?
    - Force space
    - Haptics
- Cool Robotics Share Site
  ➔ http://metr4202.tumblr.com/

Cool Robotics Video Share

RECAP I: Static Forces [L4-Slide 48]

- We can also use the Jacobian to compute the joint torques required to maintain a particular force at the end effector
- Consider the concept of virtual work

$$F \cdot \delta \mathbf{X} = \tau \cdot \delta \theta$$

- Or

$$F^T \delta \mathbf{X} = \tau^T \delta \theta$$

- Earlier we saw that

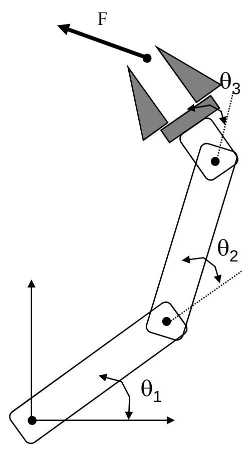
$$\delta \mathbf{X} = \mathbf{J} \delta \theta$$

- So that

$$F^T \mathbf{J} = \tau^T$$

- Or

$$\tau = \mathbf{J}^T F$$



RECAP II: Dynamics

- At the end of the day, you **probably** drive torques

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$$

where

- τ is a vector of joint torques
- Θ is the nx1 vector of joint angles
- $M(\Theta)$ is the nxn mass matrix
- $V(\Theta, \dot{\Theta})$is the nx1 vector of centrifugal and Coriolis terms
- $G(\Theta)$ is an nx1 vector of gravity terms

- Notice that all of these terms depend on Θ so the dynamics varies as the manipulator move

RECAP III: Mechanism

- Though not always…

Dynamixel Kit

http://www.tribotix.com/Products/Tribotix/Kits/UQ_Kits.htm

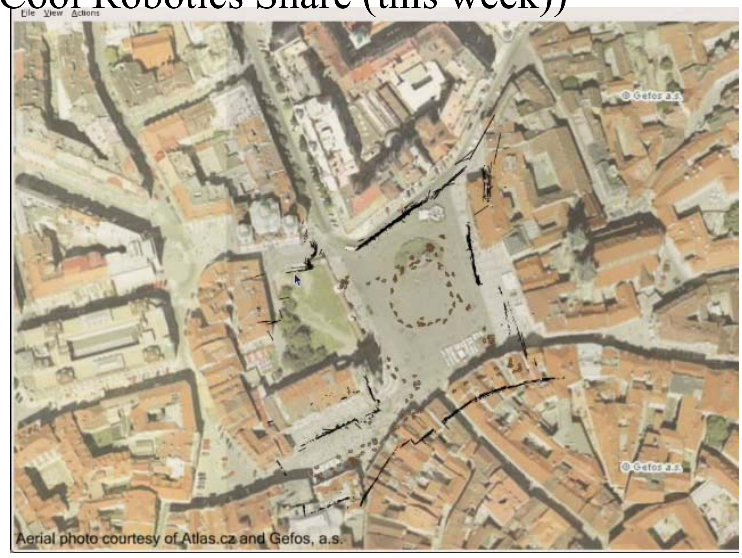
Kit includes:

Index	Part	Quantity
1	FP04-F1 Angles Hinge Bracket	2
2	FP04-F2 Stnd Hinge Bracket	4
3	FP04-F3 Bottom Bracket	5
4	FP04-F4 Large Hinge Bracket	2
5	FP04-F5 Wide Hinge Bracket	2
6	FP04-F6 Side Bracket	2
7	FP04-F7 Back Bracket	2
8	BNS-10 Bioloid Screw Set	1
9	Cable-3P Robot Cable-3P 200mm	1
10	SMPS2Dynamixel SMPS2Dynamixel	1
11	USB2Dynamixel USB2Dynamixel	1
12	AX-12A DYNAMIXEL AX-12A	3
13	DYNAMIXEL MX-12W	1

- Price: $250 (ex-GST)

From Last Week(s): SFM → Localisation

SFM: Structure from Motion
(& Cool Robotics Share (this week))



Structure [from] Motion

- Given a set of feature tracks,
  estimate the 3D structure and 3D (camera) motion.
- Assumption: orthographic projection
- Tracks: (u_{fp}, v_{fp}), f: frame, p: point
- Subtract out **mean** 2D position…

  $\mathbf{i}_f$: rotation, $\mathbf{s}_p$: position

$$u_{fp} = i_f^T s_p, v_{fp} = j_f^T s_p$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Structure from motion

- How many points do we need to match?
- 2 frames:
  - (R,t): 5 dof + 3n point locations ≤ …

    4n point measurements

    ⇒ n ≥ 5
- k frames:
  - 6(k–1)-1 + 3n ≤ 2kn
- always want to use many more

$$\hat{u}_{ij} = f(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$
$$\hat{v}_{ij} = g(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Measurement equations

- Measurement equations

$$u_{fp} = \boldsymbol{i}_f^T \boldsymbol{s}_p \qquad \boldsymbol{i}_f\text{: rotation, } \boldsymbol{s}_p\text{: position}$$
$$v_{fp} = \boldsymbol{j}_f^T \boldsymbol{s}_p$$

- Stack them up…

$$\boldsymbol{W} = \boldsymbol{R}\,\boldsymbol{S}$$
$$\boldsymbol{R} = (\boldsymbol{i}_1, \ldots, \boldsymbol{i}_F, \boldsymbol{j}_1, \ldots, \boldsymbol{j}_F)^T$$
$$\boldsymbol{S} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_P)$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Factorization

$$W = R_{2F\times 3}\, S_{3\times P}$$

SVD

$$W = U \Lambda V \quad \Lambda \text{ must be rank 3}$$

$$W' = (U \Lambda^{1/2})(\Lambda^{1/2} V) \qquad = U' V'$$

Make R orthogonal

$$R = QU', \; S = Q^{-1}V'$$

$$i_f^T Q^T Q i_f = 1 \ \ldots$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Results

- Look at paper figures…

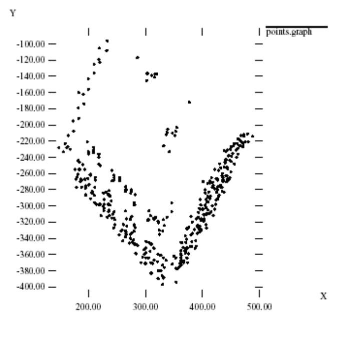


Figure 4.5: A view of the computed shape from approximately above the building (compare with figure 4.6).

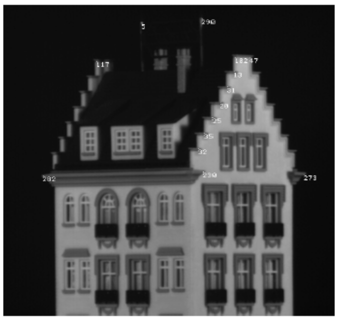


Figure 4.7: For a quantitative evaluation, distances between the features shown in the picture were measured on the actual model, and compared with the computed results. The comparison is shown in figure 4.8.

From Szeliski, *Computer Vision: Algorithms and Applications*

Bundle Adjustment

- What makes this non-linear minimization hard?
  - many more parameters: potentially slow
  - poorer conditioning (high correlation)
  - potentially lots of outliers
  - gauge (coordinate) freedom

$$\hat{u}_{ij} = f(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$
$$\hat{v}_{ij} = g(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$

From Szeliski, *Computer Vision: Algorithms and Applications*

From Last Week(s): SFM → Localisation → SLAM

Localization: SFM → SLAM

- ACFR View:
  - Treat as joint estimation problem
- [New] Oxford View:
  - Treat as (feature) placement optimization problem
  - Bundle Adjustment (borrow from computer vision)

What is SLAM?

- SLAM asks the following question:

  Is it possible for an autonomous vehicle to start at an unknown location in an unknown environment and then to incrementally build a map of this environment while simultaneously using this map to compute vehicle location?

- SLAM has many indoor, outdoor, in-air and underwater applications for both manned and autonomous vehicles.

- Examples
  - Explore and return to starting point (Newman)
  - Learn trained paths to different goal locations
  - Traverse a region with complete coverage (eg, mine fields, lawns, reef monitoring)
  - …

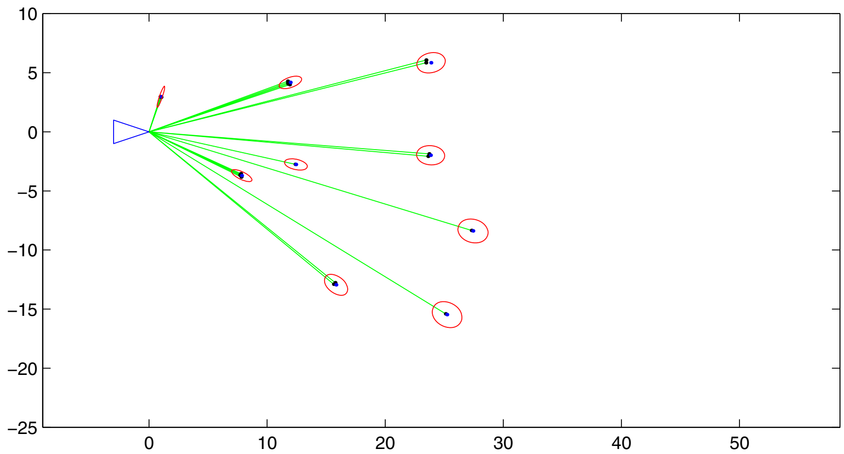
Components of SLAM

- Localisation
  - Determine pose given a priori map
- Mapping
  - Generate map when pose is accurately known from auxiliary source.
- SLAM
  - Define some arbitrary coordinate origin
  - Generate a map from on-board sensors
  - Compute pose from this map
  - Errors in map and in pose estimate are dependent.

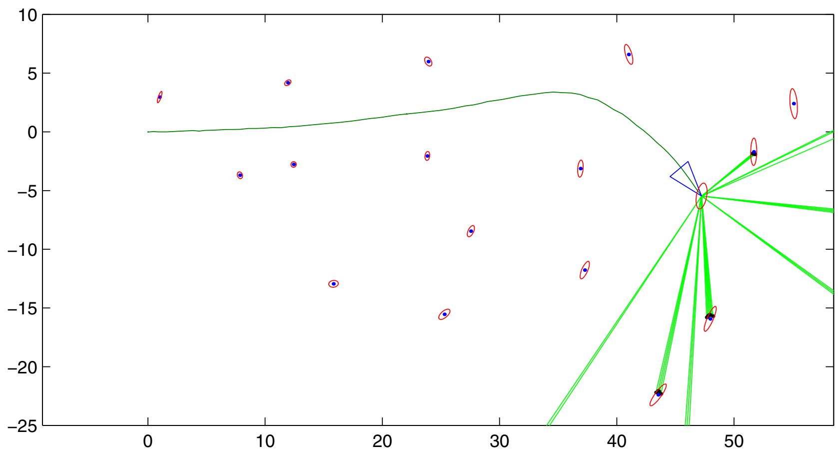
Basic SLAM Operation

Example: SLAM in Victoria Park

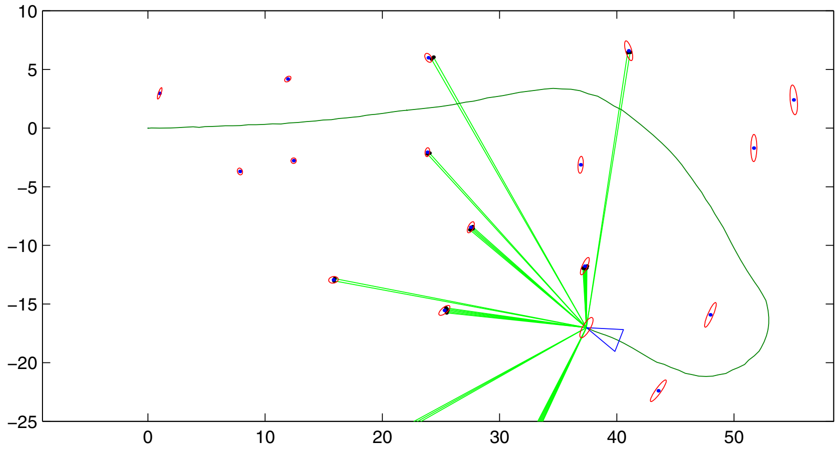
Basic SLAM Operation



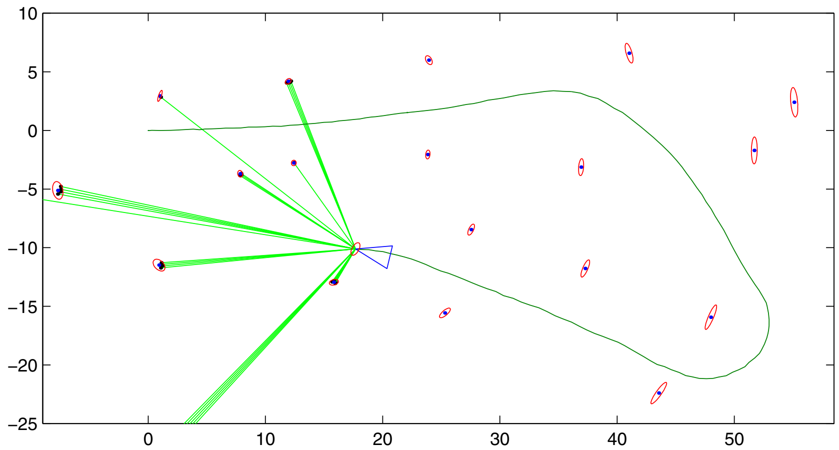
Basic SLAM Operation



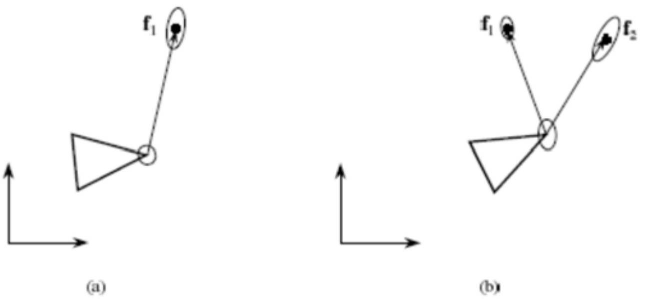
Basic SLAM Operation



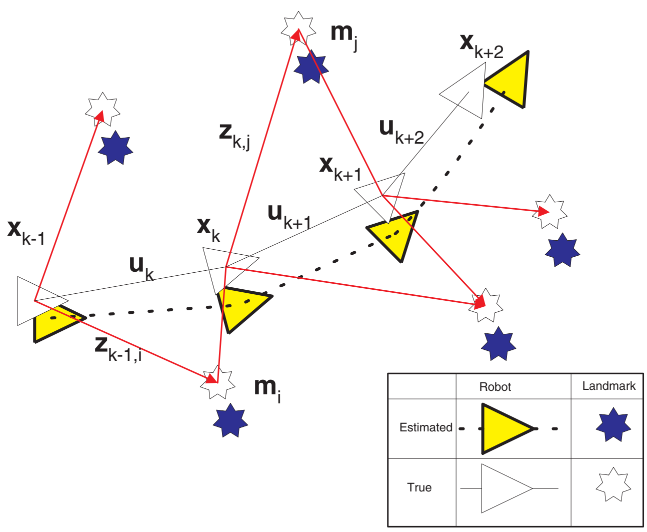
Basic SLAM Operation



Dependent Errors

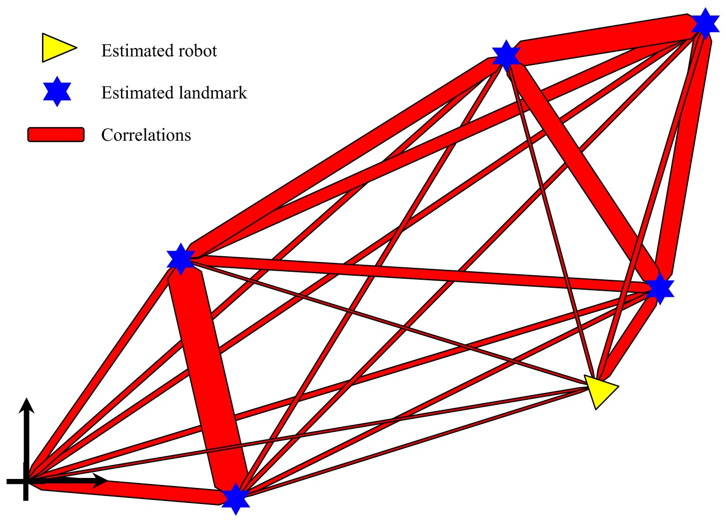


Correlated Estimates



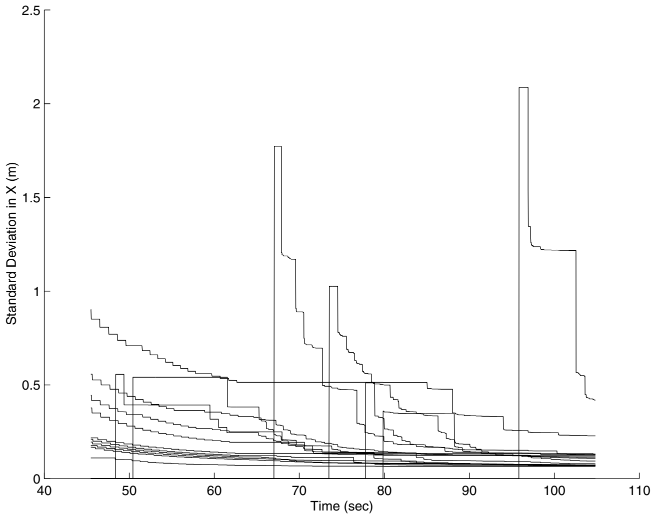
SLAM Convergence

- An observation acts like a displacement to a spring system
  - Effect is greatest in a close neighbourhood
  - Effect on other landmarks diminishes with distance
  - Propagation depends on local stiffness (correlation) properties
- With each new observation the springs become increasingly (and monotonically) stiffer.
- In the limit, a rigid map of landmarks is obtained.
  - A perfect *relative* map of the environment
- The location accuracy of the robot is bounded by
  - The current quality of the map
  - The relative sensor measurement



Monotonic Convergence

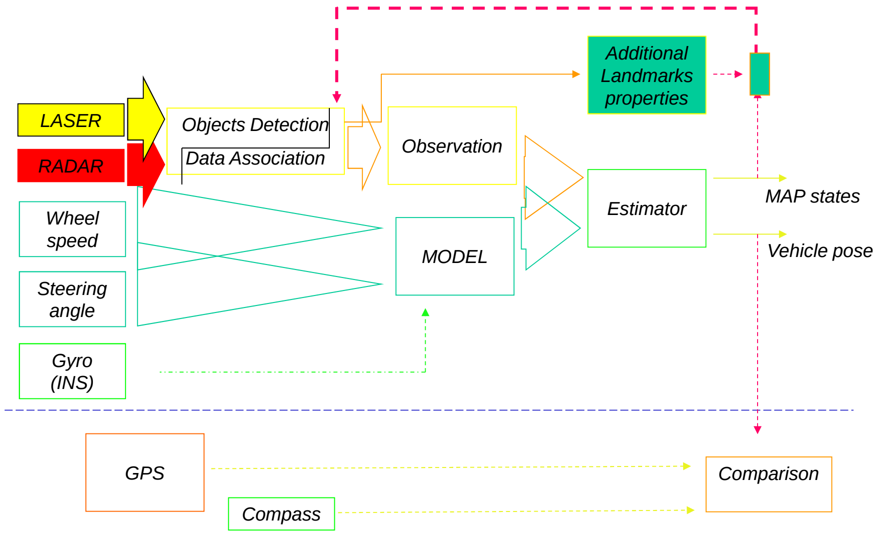
- With each new observation, the determinant decreases over the map and for any submatrix in the map.



Models

- Models are central to creating a representation of the world.
- Must have a mapping between sensed data (eg, laser, cameras, odometry) and the states of interest (eg, vehicle pose, stationary landmarks)
- Two essential model types:
  - Vehicle motion
  - Sensing of external objects

An Example System



States, Controls, Observations

Joint state with momentary pose

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{v_k} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

Joint state with pose history

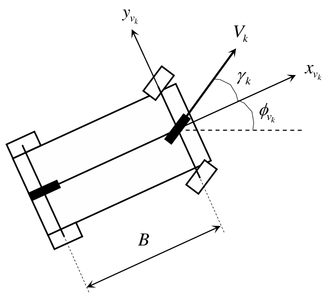
$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{v_k} \\ \mathbf{x}_{v_{k-1}} \\ \vdots \\ \mathbf{x}_{v_0} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

Control inputs: $\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k\} = \{\mathbf{U}_{0:k-1}, \mathbf{u}_k\}$

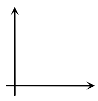
Observations: $\mathbf{Z}_{0:k} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k\} = \{\mathbf{Z}_{0:k-1}, \mathbf{z}_k\}$

Vehicle Motion Model

- Ackerman steered vehicles: Bicycle model



- Discrete time model:



$$\mathbf{x}_{v_k} = \mathbf{f}_v\left(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k\right) = \begin{bmatrix} x_{v_{k-1}} + V_k \Delta T \cos(\phi_{v_{k-1}} + \gamma_k) \\ y_{v_{k-1}} + V_k \Delta T \sin(\phi_{v_{k-1}} + \gamma_k) \\ \phi_{v_{k-1}} + \frac{V_k \Delta T}{B} \sin(\gamma_k) \end{bmatrix}$$

SLAM Motion Model

$$\mathbf{x}_{v_k} = \mathbf{f}_v\left(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k\right) = \begin{bmatrix} x_{v_{k-1}} + V_k \Delta T \cos(\phi_{v_{k-1}} + \gamma_k) \\ y_{v_{k-1}} + V_k \Delta T \sin(\phi_{v_{k-1}} + \gamma_k) \\ \phi_{v_{k-1}} + \frac{V_k \Delta T}{B} \sin(\gamma_k) \end{bmatrix}$$

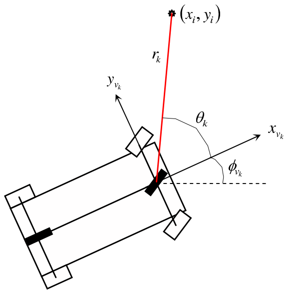
- Joint state: Landmarks are assumed stationary

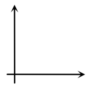
$$\mathbf{x}_k = \begin{bmatrix} \mathbf{f}_v(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k) \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix} \qquad \mathbf{x}_k = \begin{bmatrix} \mathbf{f}_v(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k) \\ \mathbf{x}_{v_{k-1}} \\ \vdots \\ \mathbf{x}_{v_0} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

Observation Model

- Range-bearing measurement

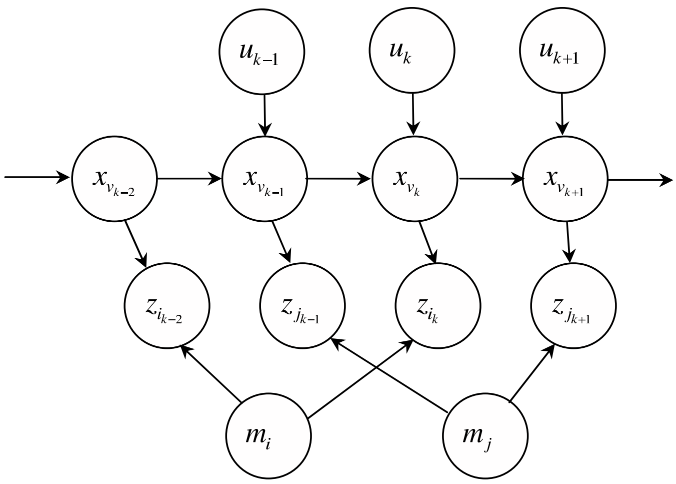
$$\mathbf{z}_{i_k} = \mathbf{h}_i(\mathbf{x}_k) = \begin{bmatrix} \sqrt{(x_i - x_{v_k})^2 + (y_i - y_{v_k})^2} \\ \arctan \frac{y_i - y_{v_k}}{x_i - x_{v_k}} - \phi_{v_k} \end{bmatrix}$$



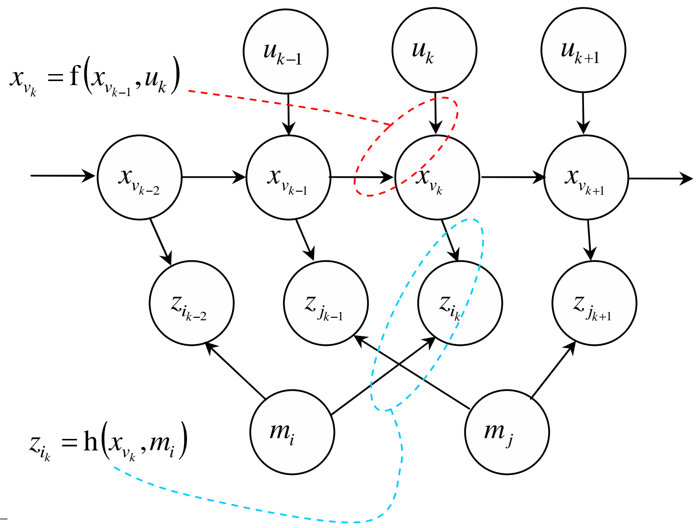


SLAM as Graphical Model

- An optimization framework



Models in Graphical Model



Perfect World: Deterministic

- Exact pose from motion model
- Global localisation by triangulation
  - Even if range-only or bearing-only sensors, can localise given enough measurements
  - Solve simultaneous equations: N equations for N unknowns

Real World: Uncertain

- All measurements have errors
- In SLAM, measurement errors induce dependencies in the landmark and vehicle pose estimates
  - Everything is correlated

Bayesian Estimation

- Standard theory for dealing with uncertain information in a consistent manner

Brief Overview of Probability Theory

- Probability density function (PDF) over N-D state space $\mathbf{x} \in \mathcal{X}$ is denoted $p(\mathbf{x})$

- Properties of a PDF

$$\mathbb{R}^N \mapsto \mathbb{R}$$

$$p(\mathbf{x}) \geq 0, \quad \forall\ \mathbf{x} \in \mathcal{X}$$

$$\int_{\mathcal{X}} p(\mathbf{x})\, d\mathbf{x} = 1$$

Brief Overview of Probability Theory

- State vector $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}$

- Joint PDF is $p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$

- Conditional PDF of x_1 *given* x_2 and x_3

$$p(\mathbf{x}_1 | \mathbf{x}_2, \mathbf{x}_3)$$

- *Conditional independence*: if x_1 is independent of x_2 given x_3 then

$$p(\mathbf{x}_1 | \mathbf{x}_2, \mathbf{x}_3) \stackrel{\text{indep}}{=} p(\mathbf{x}_1 | \mathbf{x}_3)$$

Two Essential Rules for Manipulating Probabilities

- Sum rule $$p(\mathbf{x}_1|\mathcal{H}) \triangleq \int p(\mathbf{x}_1, \mathbf{x}_2|\mathcal{H})\, d\mathbf{x}_2$$
- Product rule $$\begin{aligned} p(\mathbf{x}_1, \mathbf{x}_2|\mathcal{H}) &\triangleq p(\mathbf{x}_1|\mathbf{x}_2, \mathcal{H})\, p(\mathbf{x}_2|\mathcal{H}) \\ &\triangleq p(\mathbf{x}_2|\mathbf{x}_1, \mathcal{H})\, p(\mathbf{x}_1|\mathcal{H}) \end{aligned}$$

Implications of the Product Rule

- Conditionals $$p(\mathbf{x}_1|\mathbf{x}_2, \mathcal{H}) = \frac{p(\mathbf{x}_1, \mathbf{x}_2|\mathcal{H})}{p(\mathbf{x}_2|\mathcal{H})}$$
- Independence $$p(\mathbf{x}_1, \mathbf{x}_2|\mathcal{H}) \stackrel{\text{indep}}{=} p(\mathbf{x}_1|\mathcal{H})\, p(\mathbf{x}_2|\mathcal{H})$$
- Markov Models $$p(\mathbf{x}_1|\mathcal{H}) = \int p(\mathbf{x}_1|\mathbf{x}_2, \mathcal{H})\, p(\mathbf{x}_2|\mathcal{H})\, d\mathbf{x}_2$$
- Bayes theorem $$p(\mathbf{x}_1|\mathbf{x}_2, \mathcal{H}) = \frac{p(\mathbf{x}_2|\mathbf{x}_1, \mathcal{H})\, p(\mathbf{x}_1|\mathcal{H})}{p(\mathbf{x}_2|\mathcal{H})}$$

Marginalisation: Remove old states

- As per the sum rule

$$p(\mathbf{x}_1) = \int p(\mathbf{x}_1, \mathbf{x}_2)\, d\mathbf{x}_2$$
$$= \int p(\mathbf{x}_1|\mathbf{x}_2)\, p(\mathbf{x}_2)\, d\mathbf{x}_2$$

- Marginal says: what is PDF of x_1 when we don't care what value x_2 takes; ie, $p(x_1)$ regardless of x_2
- Important distinction: x_1 is still dependent on x_2, but $p(x_1)$ is not a *function* of x_2

Bayesian Update: Inverse probability

- Bayes theorem $p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{x})\, p(\mathbf{x})}{p(\mathbf{z})}$
- Observation model $\mathbf{z} = \mathbf{h}(\mathbf{x}, \mathbf{r})$
- Conditional probability

$$p(\mathbf{z}|\mathbf{x}) = \int p(\mathbf{z}|\mathbf{x}, \mathbf{r})\, p(\mathbf{r})\, d\mathbf{r}$$
$$= \int \delta(\mathbf{z} - \mathbf{h}(\mathbf{x}, \mathbf{r}))\, p(\mathbf{r})\, d\mathbf{r}$$

- Likelihood function $\Lambda(\mathbf{x}) = p(\mathbf{z} = \mathbf{z}_0|\mathbf{x})$

Bayes Update

- Update $$p(\mathbf{x}|\mathbf{z}=\mathbf{z}_0) = \frac{\Lambda(\mathbf{x})p(\mathbf{x})}{\int \Lambda(\mathbf{x})p(\mathbf{x})\,d\mathbf{x}}$$
- Denominator term often seen as just a normalising constant, but is important for saying how likely a model or hypothesis is
  - Used in FastSLAM for determining particle weights
  - Used in multi-hypothesis data association

Applying Bayes to SLAM: Available Information

- States $\mathbf{x}_k$ (Hidden or inferred values)
  - Vehicle poses
  - Map; typically composed of discrete parts called landmarks or features
- Controls $\mathbf{U}_{0:k}$
  - Velocity
  - Steering angle
- Observations $\mathbf{Z}_{0:k}$
  - Range-bearing measurements

Augmentation: Adding new poses and landmarks

- Add new pose

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{f}_v(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k) \\ \mathbf{x}_{v_{k-1}} \\ \vdots \\ \mathbf{x}_{v_0} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

- Conditional probability is a Markov Model

$$\begin{aligned} p\left(\mathbf{x}_{v_k} | \mathbf{x}_{k-1}\right) &= \int p\left(\mathbf{x}_{v_k} | \mathbf{x}_{k-1}, \mathbf{u}_k\right) p\left(\mathbf{u}_k\right) d\mathbf{u}_k \\ &= \int \delta\left(\mathbf{x}_{v_k} - \mathbf{f}_v\left(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k\right)\right) p\left(\mathbf{u}_k\right) d\mathbf{u}_k \\ &= p\left(\mathbf{x}_{v_k} | \mathbf{x}_{v_{k-1}}\right) \end{aligned}$$

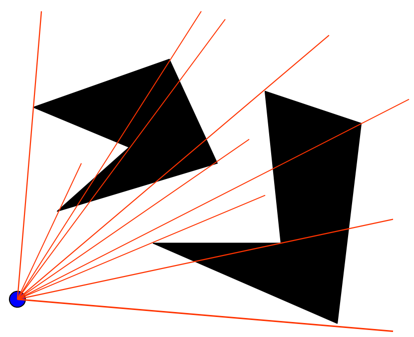
Now How Do We Use This To Get Somewhere?
Motion Planning

Path-Planning Approaches

- Roadmap
  Represent the connectivity of the free space by a network of 1-D curves
- Cell decomposition
  Decompose the free space into simple cells and represent the connectivity of the free space by the adjacency graph of these cells
- Potential field
  Define a function over the free space that has a global minimum at the goal configuration and follow its steepest descent

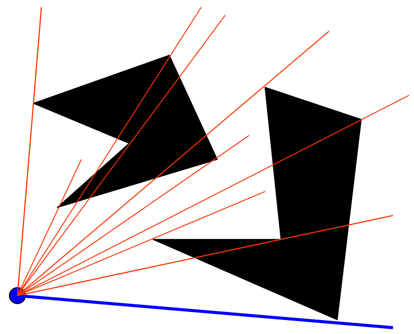
Slide from Latombe, CS326A

I. Rotational Sweep



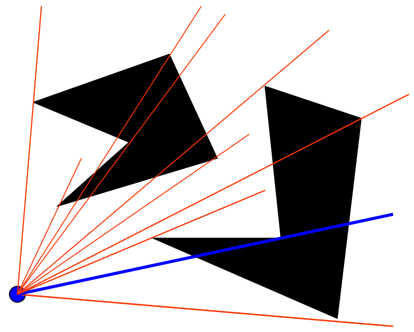
Slide from Latombe, CS326A

Rotational Sweep



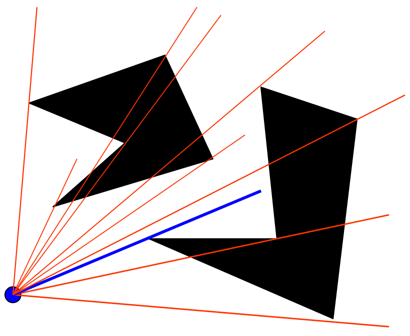
Slide from Latombe, CS326A

Rotational Sweep



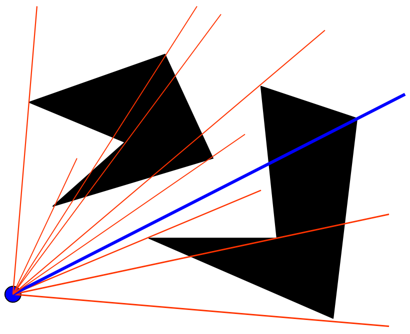
Slide from Latombe, CS326A

Rotational Sweep



Slide from Latombe, CS326A

Rotational Sweep



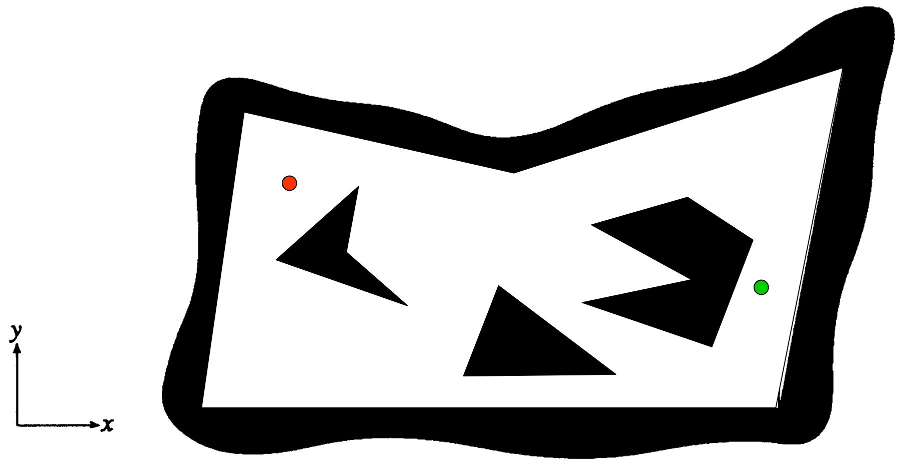
Slide from Latombe, CS326A

II. Cell-Decomposition Methods

Two classes of methods:

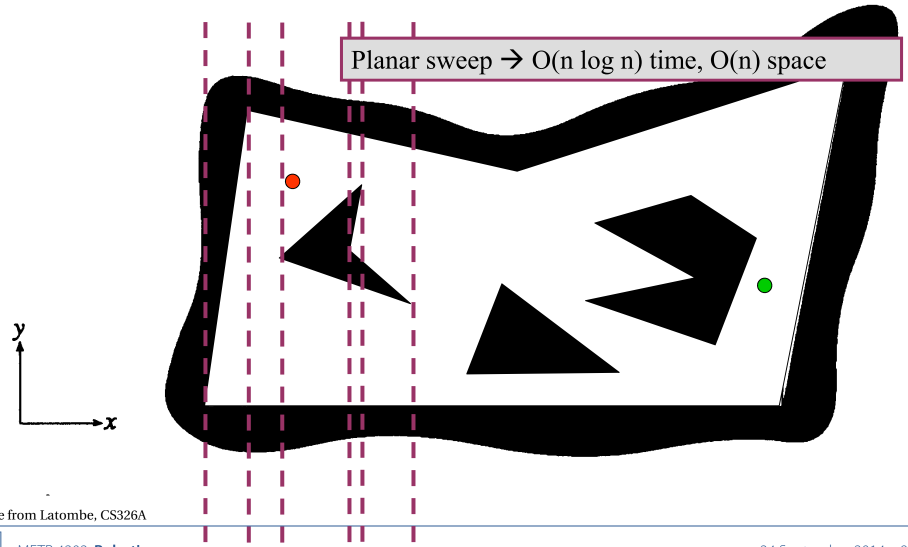
- Exact cell decomposition
  - The free space **F** is represented by a collection of non-overlapping cells whose union is exactly **F**
  - Example: trapezoidal decomposition
- Approximate cell decomposition
  - **F** is represented by a collection of non-overlapping cells whose union is contained in **F**
    Examples: quadtree, octree, 2n-tree

Trapezoidal decomposition



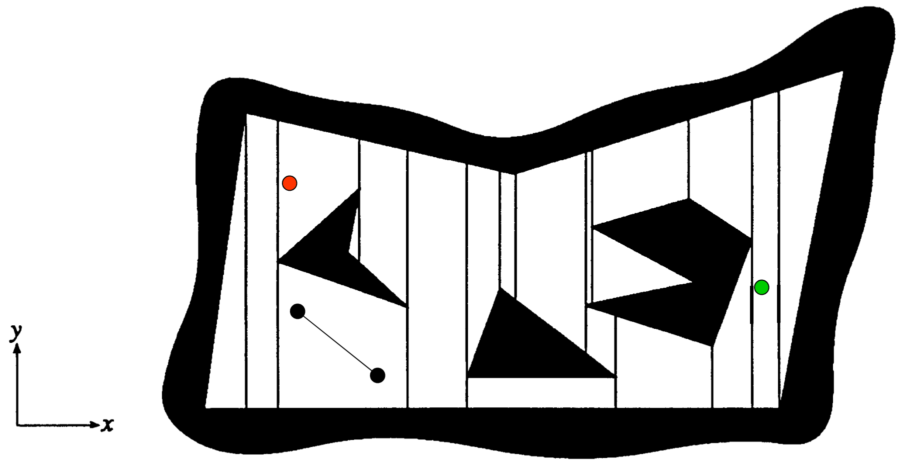
Slide from Latombe, CS326A

Trapezoidal decomposition



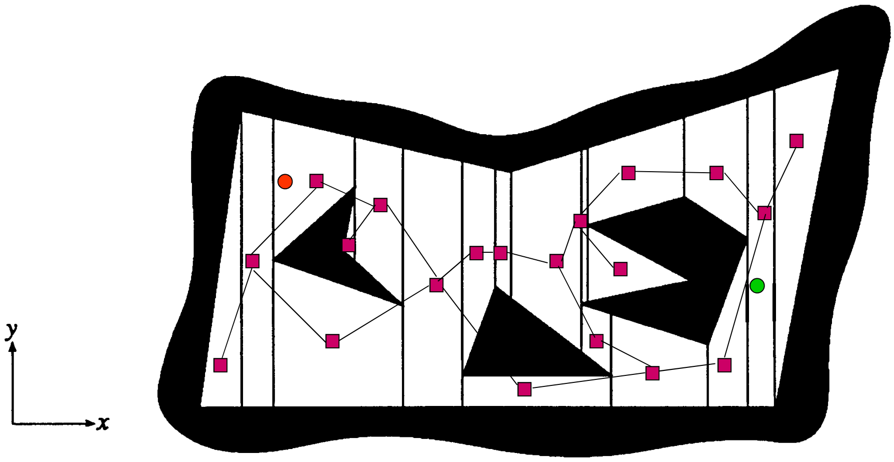
Slide from Latombe, CS326A

Trapezoidal decomposition



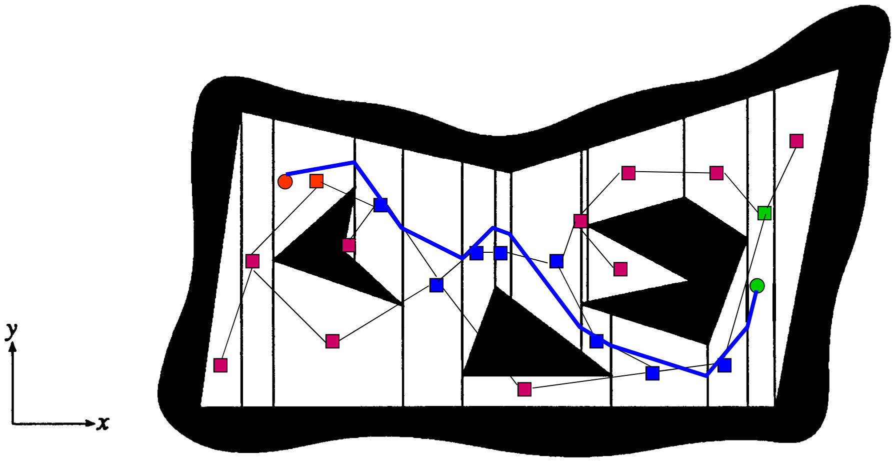
Slide from Latombe, CS326A

Trapezoidal decomposition



Slide from Latombe, CS326A

Trapezoidal decomposition



Slide from Latombe, CS326A

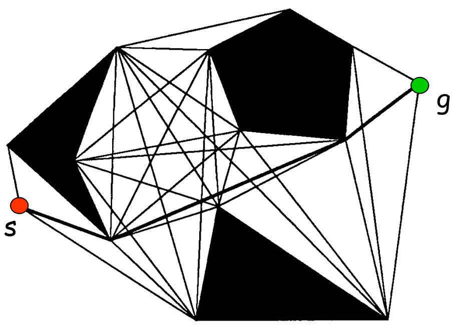
III. Roadmap Methods

- **Visibility graph**
- **Voronoi diagram**
- Silhouette
  First complete general method that applies to spaces of any dimension and is singly exponential in # of dimensions [Canny, 87]
- **Probabilistic roadmaps (PRMS)**
  and Rapidly-exploring Randomized Trees (RRTs)

Slide from Latombe, CS326A

Roadmap Methods

- **Visibility graph**
  Introduced in the Shakey project at SRI in the late 60s.
  Can produce shortest paths in 2-D configuration spaces

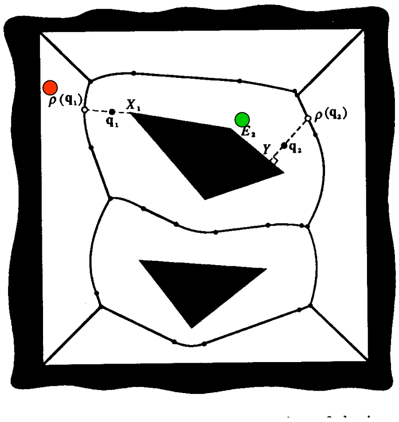


Slide from Latombe, CS326A

Roadmap Methods

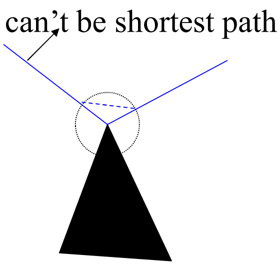
- Voronoi diagram
  Introduced by
  Computational
  Geometry researchers.
  Generate paths that
  maximizes clearance.

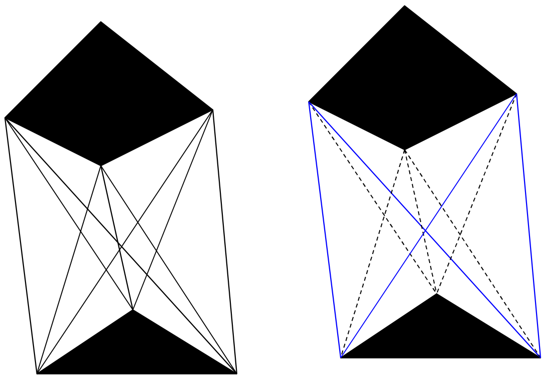
  O(n log n) time
  O(n) space



Slide from Latombe, CS326A

II.Visibility Graph



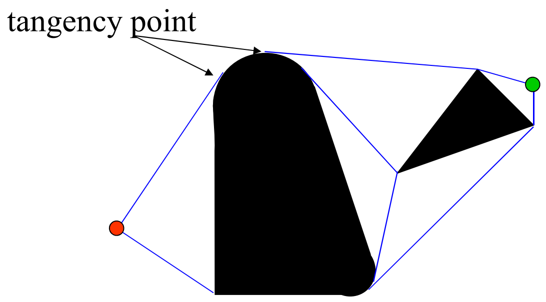


tangent segments

→ Eliminate concave obstacle vertices

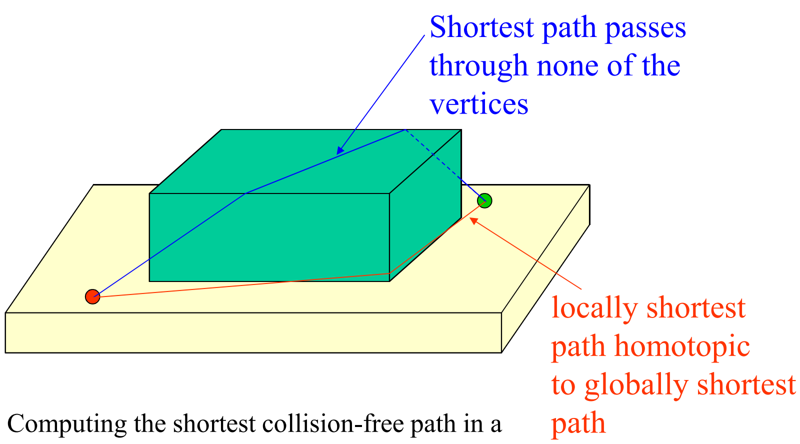
Slide from Latombe, CS326A

Generalized (Reduced) -- Visibility Graph



Slide from Latombe, CS326A

Three-Dimensional Space

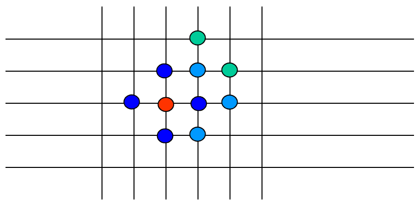


Computing the shortest collision-free path in a polyhedral space is NP-hard

Slide from Latombe, CS326A

Sketch of Grid Algorithm (with best-first search)

- Place regular grid G over space
- Search G using best-first search algorithm with potential as heuristic function



Slide from Latombe, CS326A

Simple Algorithm (for Visibility Graphs)

- Install all obstacles vertices in VG, plus the start and goal positions
- For every pair of nodes u, v in VG

```
    If segment(u,v) is an obstacle edge then
            insert (u,v) into VG
    else
    for every obstacle edge e
            if segment(u,v) intersects e
                    then go up to segment
  insert (u,v) into VG
```

- Search VG using A*

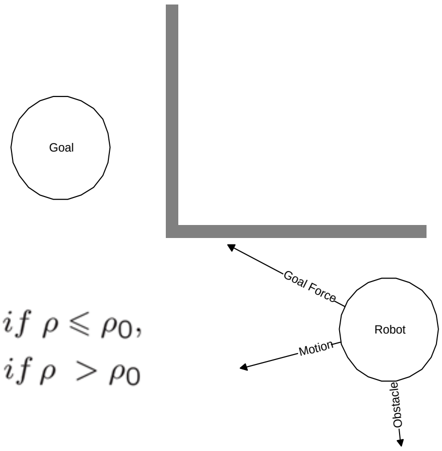
Slide based on Latombe, CS326A

IV. Potential Field Methods

- Approach initially proposed for real-time collision avoidance [Khatib, 86]

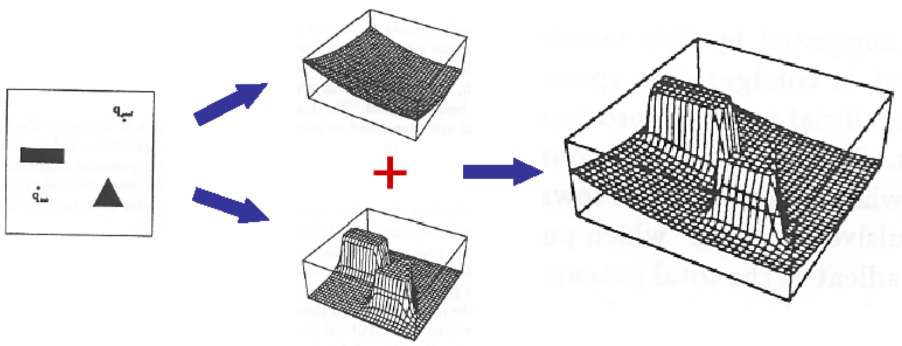
$$F_{Goal} = -k_p(x - x_{Goal})$$

$$F_{Obstacle} = \begin{cases} \eta \left(\frac{1}{\rho} - \frac{1}{\rho_0}\right) \frac{1}{\rho^2} \frac{\partial \rho}{\partial x} & if\ \rho \leqslant \rho_0, \\ 0 & if\ \rho > \rho_0 \end{cases}$$



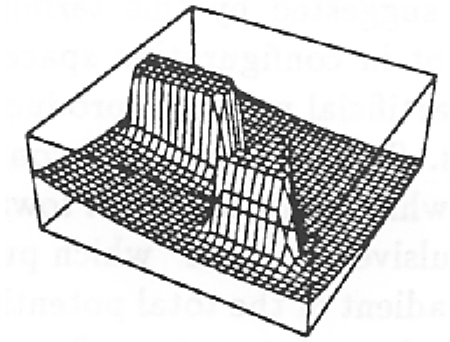
Slide based on Latombe, CS326A

Attractive and Repulsive fields



Slide from Latombe, CS326A

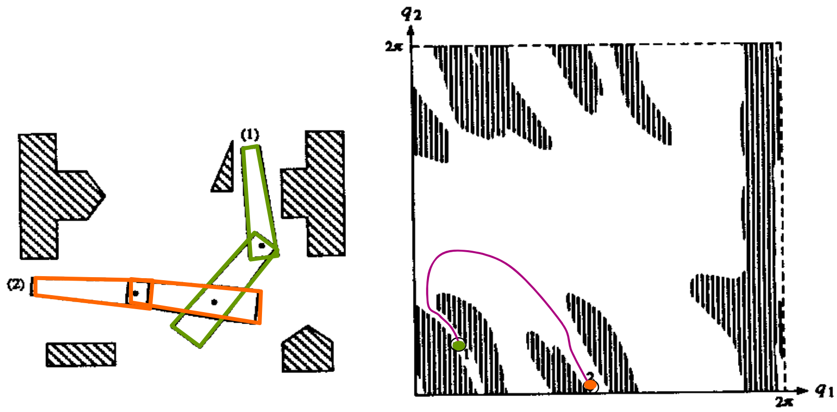
Local-Minimum Issue



- Perform best-first search (possibility of combining with approximate cell decomposition)
- Alternate descents and random walks
- Use local-minimum-free potential (navigation function)

Slide from Latombe, CS326A

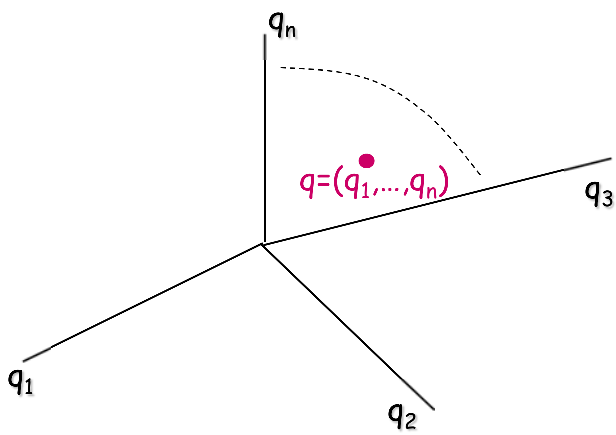
Configuration Space



- A robot configuration is a specification of the positions of all robot points relative to a fixed coordinate system
- Usually a configuration is expressed as a "vector" of position/orientation parameters

Slide from Latombe, CS326A

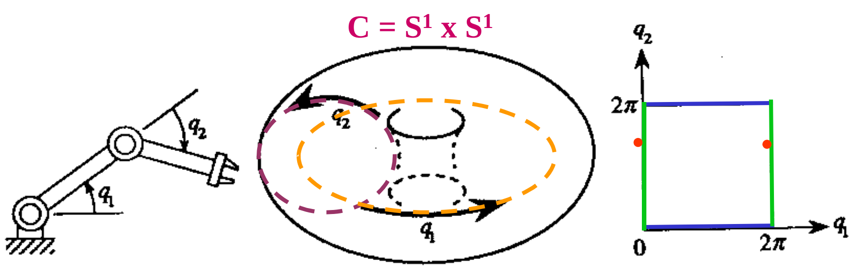
Motion Planning in C-Space



Slide from Latombe, CS326A

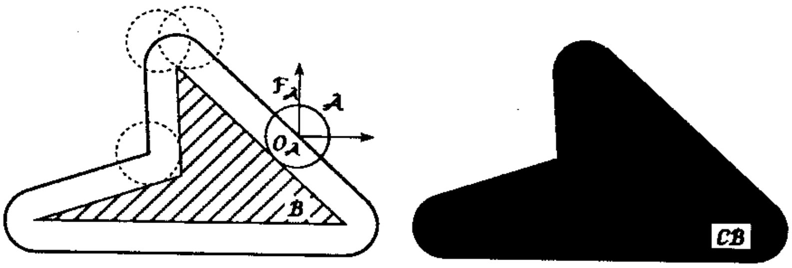
Configuration Space of a Robot

- Space of all its possible configurations
- But the topology of this space is usually not that of a Cartesian space



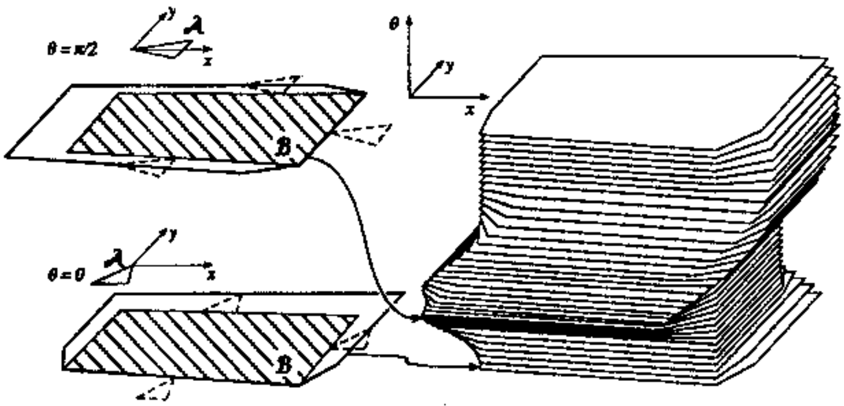
Slide from Latombe, CS326A

Disc Robot in 2-D Workspace



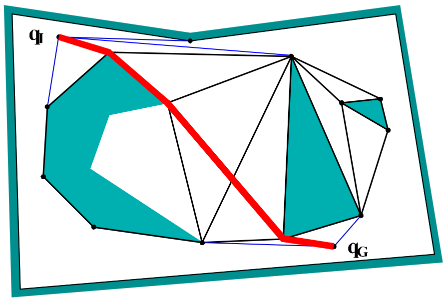
Slide from Latombe, CS326A

Rigid Robot Translating and Rotating in 2-D



Slide from Latombe, CS326A

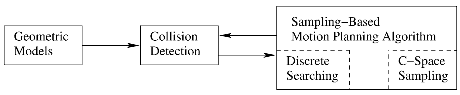
Geometric Planning Methods



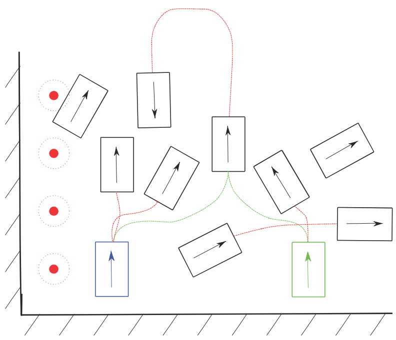
- Several Geometric Methods:
  - Vertical (Trapezoidal) Cell Decomposition
  - **Roadmap Methods**
    - **Cell (Triangular) Decomposition**
    - Visibility Graphs
    - Veroni Graphs

Artwork from LaValle, Ch. 6

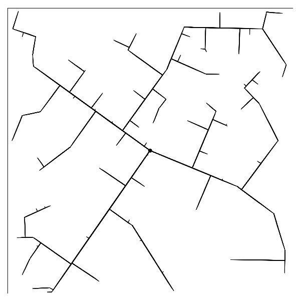
Sample-Based Motion Planning



- PRMs

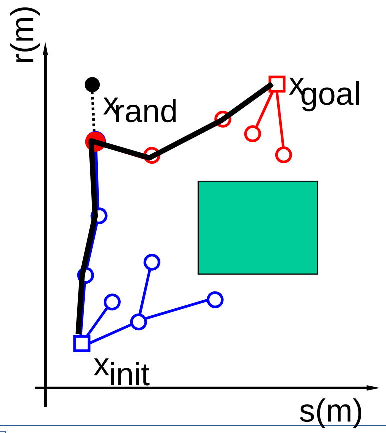


- RRTs

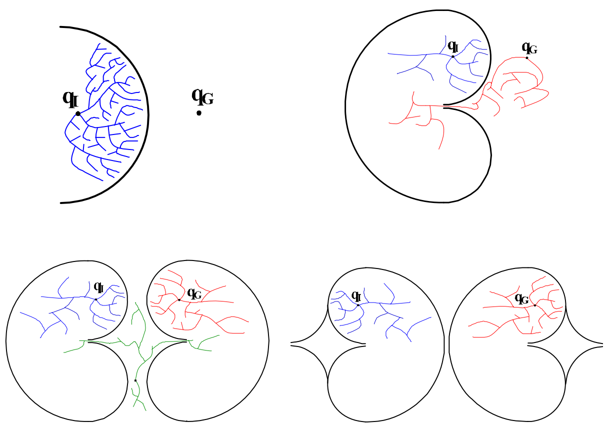


Artwork based on LaValle, Ch. 5

Rapidly Exploring Random Trees (RRT)

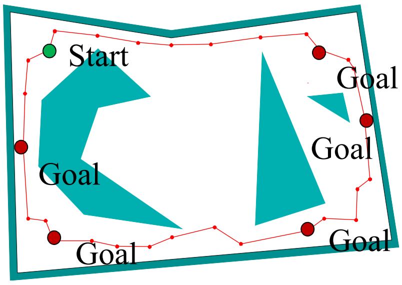


Sampling and the "Bug Trap" Problem



Artwork based on LaValle, Ch. 5

Multiple Points & Sequencing



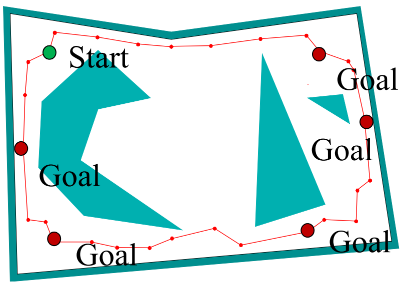
- Sequencing
  - Determining the "best" order to go in

➔ Travelling Salesman Problem

A salesman has to visit each city on a given list exactly once. In doing this, he **starts** from his home city and in the **end he has to return to his home** city. It is plausible for him to select the order in which he visits the cities so that the **total of the distances travelled** in his tour is as small as possible.

Artwork based on LaValle, Ch. 6

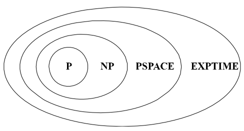
Travelling Salesman Problem



- Given a $n \times n$ distance matrix **C**=(c_{ij})
- Minimize:

$$c(\pi) = \sum_{i=1}^{n} c_{i\pi(i)}$$

- Note that this problem is NP-Hard

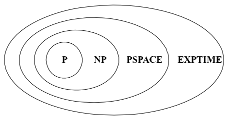


➔ BUT, Special Cases are Well-Solvable!

Artwork based on LaValle, Ch. 6

Travelling Salesman Problem [2]

- This problem is NP-Hard



➔ BUT,
Special Cases are Well-Solvable!

For the Euclidean case
(where the points are on the 2D Euclidean plane) :

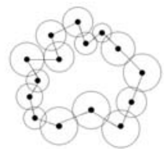
- The shortest TSP tour does not intersect itself, and thus geometry makes the problem somewhat easier.
- If all cities lie on the boundary of a convex polygon, the optimal tour is a cyclic walk along the boundary of the polygon (in clockwise or counterclockwise direction).

***The k*-line TSP**

- The a special case where the cities lie on *k* parallel (or almost parallel) lines in the Euclidean plane.
- EG: Fabrication of printed circuit boards
- Solvable in **O**(n^3) time by Dynamic Programming (Rote's algorithm)

The necklace TSP

- The special Euclidean TSP case where there exist *n* circles around the *n* cities such that every cycle intersects exactly two adjacent circles



Deliverance?
[State-Space] Control !
{Tune in Next Week ☺}