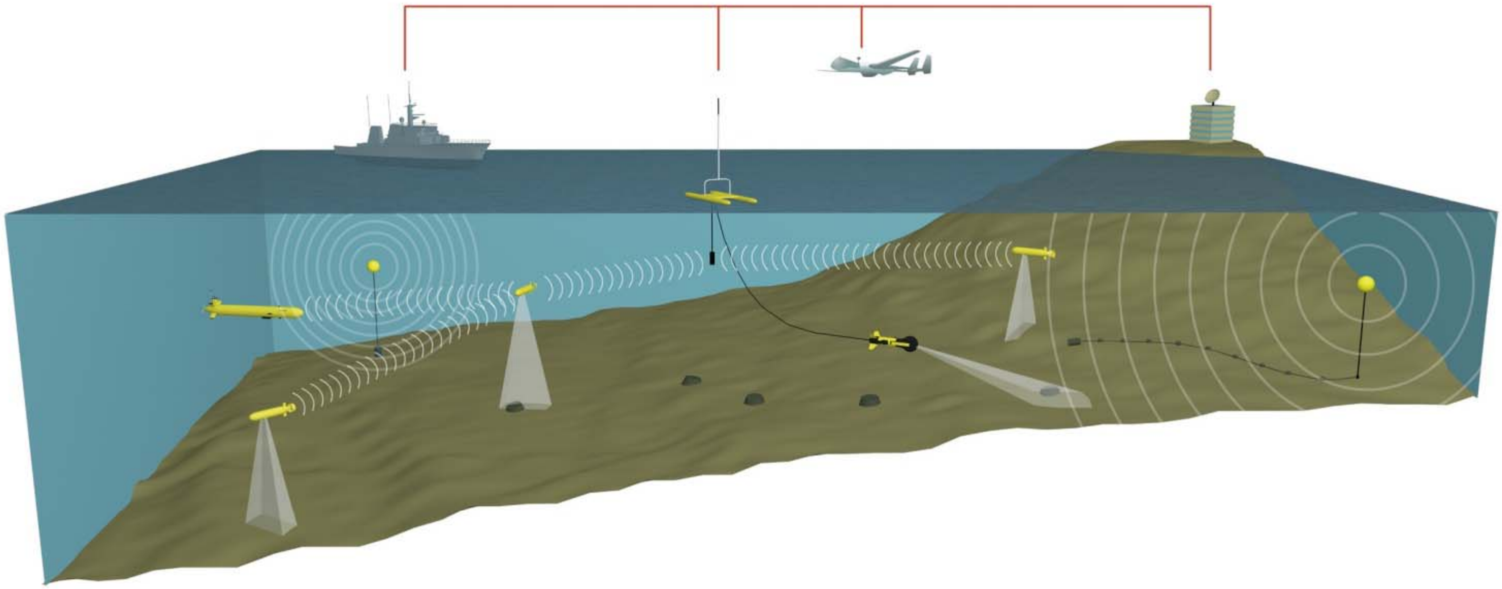
Autonomous Robotics 6905



Lecture 6: Simultaneous Localization and Mapping

Dalhousie University

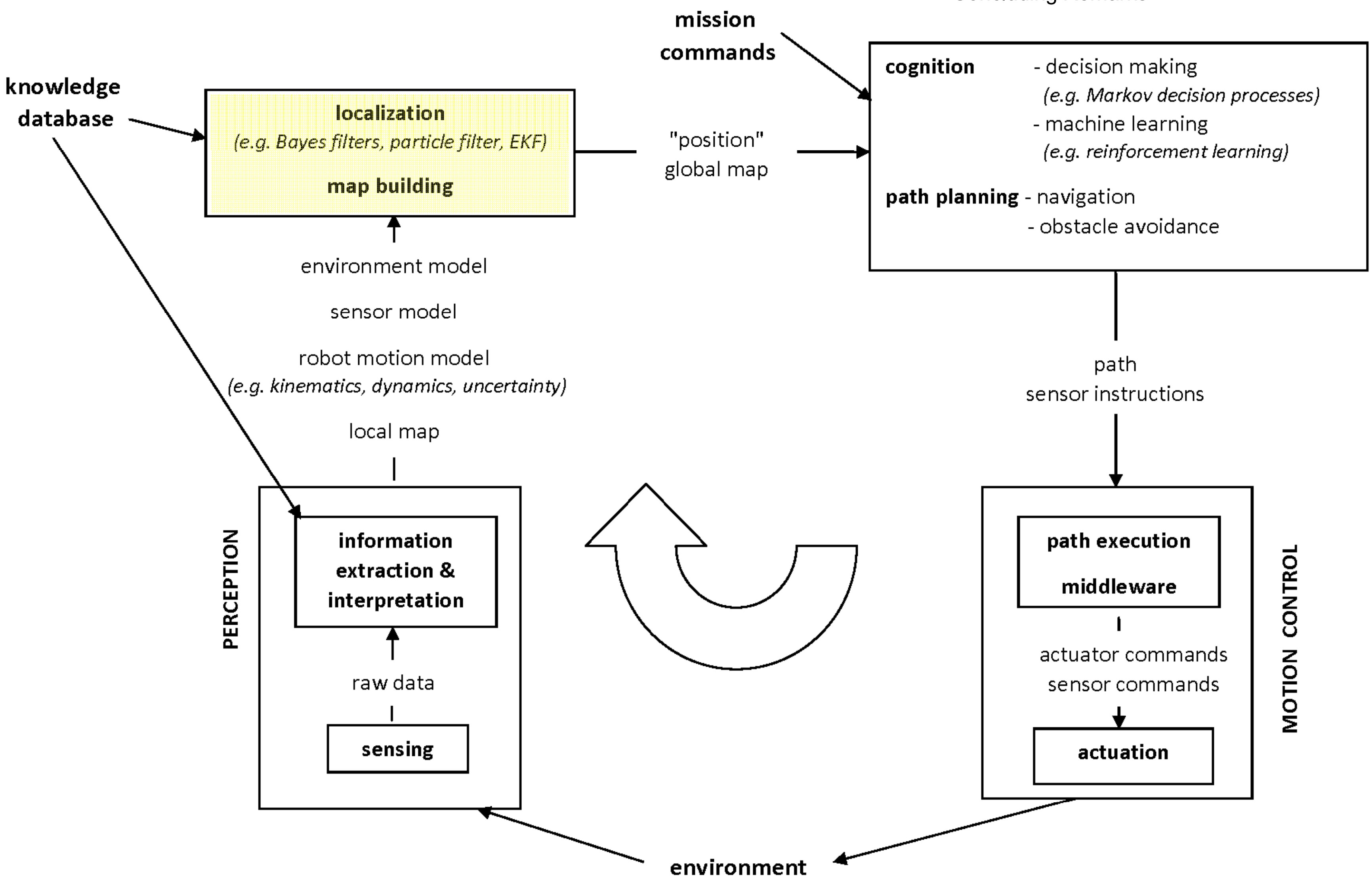
October 14, 2011

Lecture Outline

- Introduction
- Extended Kalman Filter
- Particle Filter
- Underwater SLAM
- Concluding Remarks

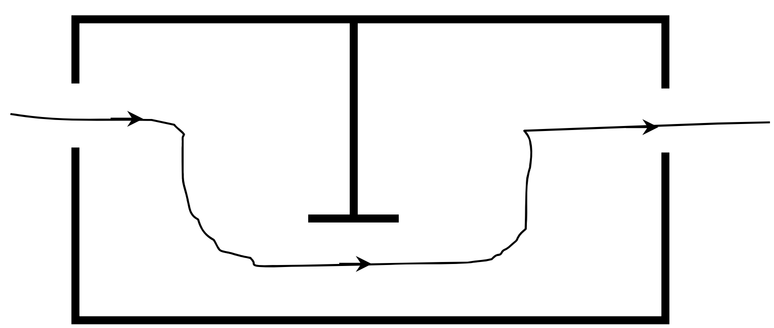
- based on diagrams and lecture notes adapted from:
  – Probabilistic Robotics (Thrun, et. al.)
  – Autonomous Mobile Robots (Siegwart, Nourbakhsh)

Control Scheme for Autonomous Mobile Robot



Plan for Class

- Thomas covered generalized Bayesian filters for localization last week
  – Kalman filter most useful outcome for localization
- Mae covers path-planning and navigation
- **Mae then follows on with Bayesian filters to do a specific example, SLAM**
- Thomas to follow with reinforcement learning after that



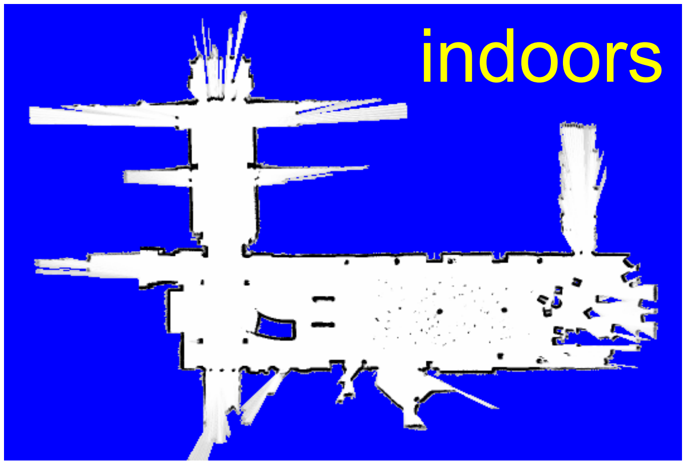
Robot Mapping
When it is Applied

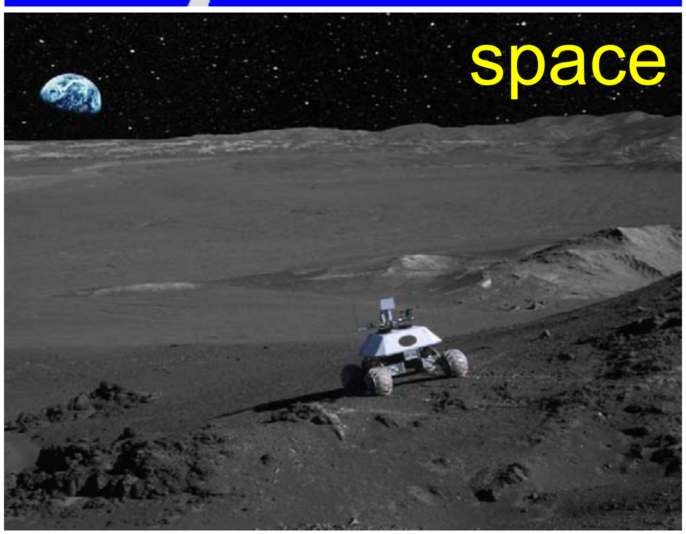
when is simultaneous localization and mapping (SLAM) needed?

- when a robot has to be truly autonomous with no human intervention (e.g. underwater vehicles beyond a few km, millions of miles away in space the operator has no situational awareness of the robot's environment)
- environment is unknown and there is no prior knowledge
- beacons and networks cannot be deployed or used (e.g. in GPS denied areas like underwater or under-ice)

Robot Mapping
Where it is Applied

- in all environments robots are in





Robot Mapping Problems Difficulty

- most difficult perceptual inference problem in mobile robots
- acquiring a spatial model of the robot's environment for navigation purposes
- robot must have sensors that enable it to perceive its environment e.g. cameras, range finders, sonar, laser, tactile sensors, compass and GPS
- sensors are subject to error (measurement noise)
- sensors have finite range (e.g. sound can't penetrate walls)
  - this means the robot has to navigate through its environment when map building
- motions commands (*controls*) issued during mapping carry information for building maps since they convey info about locations where different sensor measurements are taken

Markov Localization (Bayes Filter) Quick Review

- observation model: $P(z_t \mid x_t)$ or $P(z_t \mid x_t, m)$
  - probability of a measurement z_t given that the robot is at position x_t *and map* m
- motion model: $P(x_t \mid x_{t-1}, u_t)$
  - posterior probability that action u_t takes the robot from states x_{t-1} to x_t
- belief
  - posterior probability
  - conditioned on available data
  - $Bel(x_t) = p(x_t \mid z_t, u_t)$
- prediction
  - estimate before measurement: $\overline{Bel}(x_t) = p(x_t \mid z_t u_{t-1})$

Markov Localization(Bayes Filters) Quick Review

- prediction (prior):

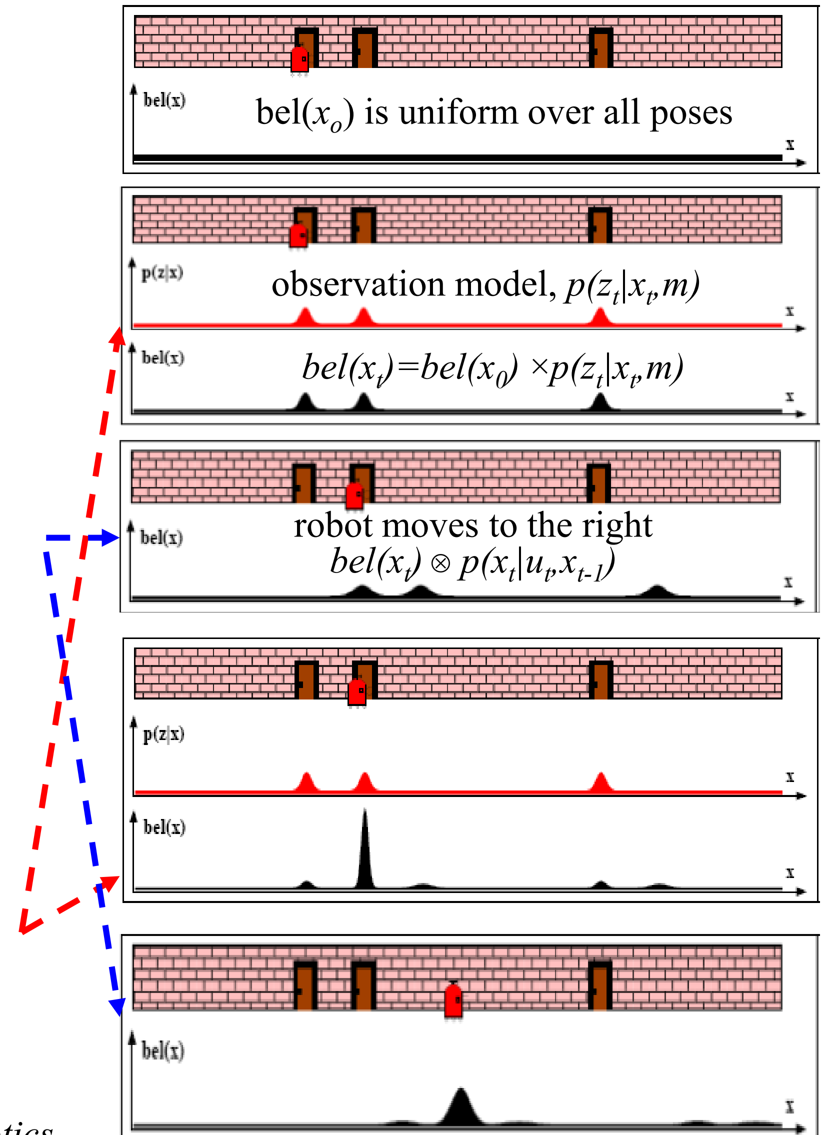
$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$$

(convolves motion model with belief from previous time step)

- update (posterior):

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

incorporates the measurement



Markov Localization (Bayes Filter) Quick Review

- for developing a range/bearing sensor model it is useful to introduce a correspondence variable between the feature f_t^i and the landmark m_j of the map
  - this variable is the correspondence and it is denoted c_t^i
  - c_t^i is the true identity of the observed feature f_t^i
- EKF localization assumes the map is represented by a collection of features and that the correspondences are known

Robot Mapping Challenges
1. Modelling Measurement Noise

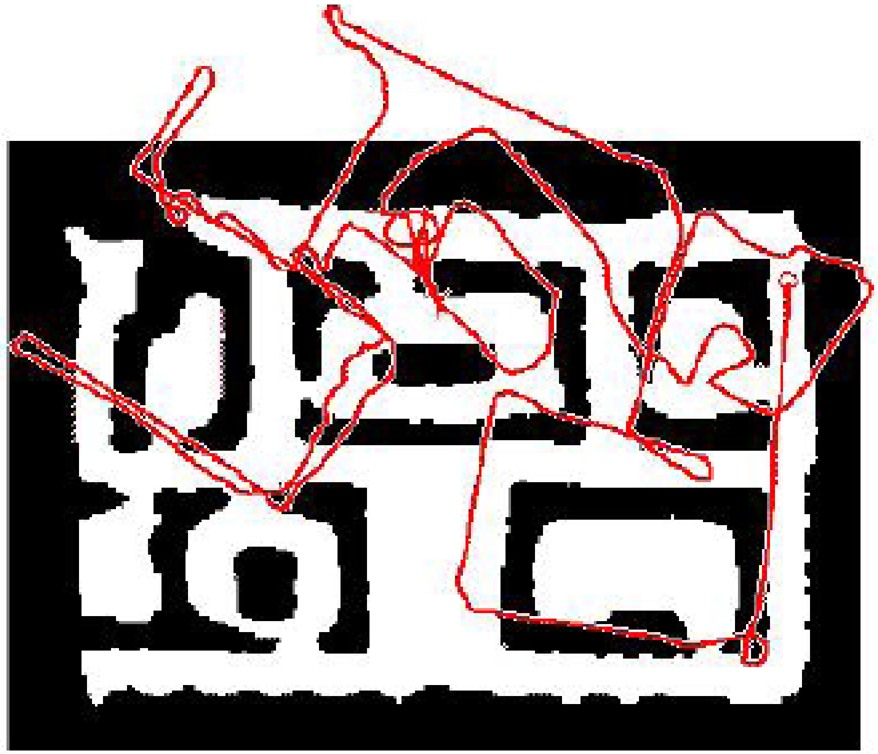
- robot motion itself is subject to errors and controls alone are insufficient to determine a robot's pose within its environment
- modelling measurement noise is a key challenge
  - robotic mapping would be relatively easy if the noise of different measurements are *statistically independent*
    - robot would just make more measurements to negate noise effects
  - unfortunately, with robotic mapping measurements errors are *statistically dependent*
    - errors in controls accumulate over time and affect the way sensor measurements are made

Robot Mapping Challenges Localization and Mapping

- mapping sometimes referred to in conjunction with localization (determine robot pose)
  - estimating where things are and determining where the robot is (both have uncertainty) – is solved in conjunction
  - allows the measurement and control noise to be independent in the robot state estimation
- thus the problem of mapping creates an inherent robot *localization* problem so robot mapping is also referred to as concurrent mapping and localization (CML)
- state-of-the-art algorithms in mapping are probabilistic due to the uncertainty and sensor noise

Robotic Mapping Challenges
1. Modelling Measurement Noise

cumulative effect of control errors on future sensor interpretations



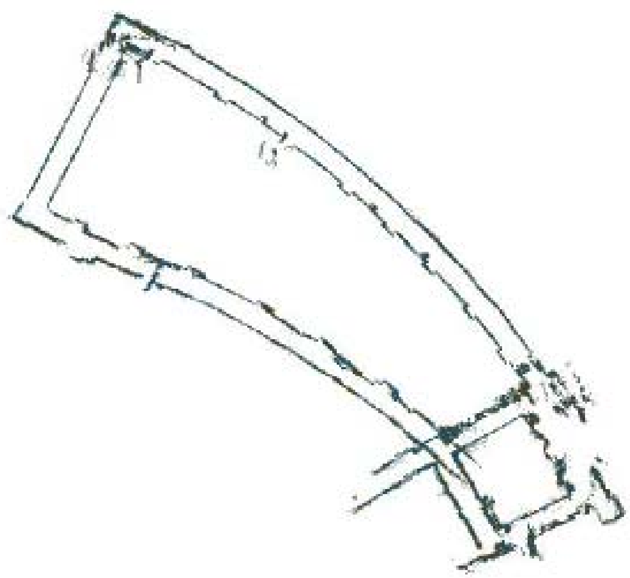
small rotation error at one end of a corridor cumulates to many meters of error at the other end relative to map for robot path obtained by odometry

Robotic Mapping Challenges
2. High Dimensionality of Entities

- consider the info to describe your home environment with just corridors, intersections, rooms, and doors
  - detailed 2D floor plan requires thousands of coordinates to define
  - 3D visual map would require millions of coordinates
  - from a statistical perspective, each coordinate is a dimension of the estimation problem

Robotic Mapping Challenges
3. Correspondence Problem

- also referred to as the *data association problem* – most difficult problem
  - determine if sensor measurements taken at different times correspond to the same physical object



robot trying to map a cyclic environment; when closing cycle robot has to localize itself relative to the previous map – by then, cumulated pose error may be unbounded

Robotic Mapping Challenges

4. Environment Changes with Time

- on scales that vary depending on the environment:
  - from a tree that changes very slowly
  - sea bottom that changes due to currents over days
  - location of a chair that could change on the order of minutes,
  - or people movement that changes constantly

- environment changes manifest as inconsistent sensor measurements (when they are not)
  - few algorithms that *learn* meaningful maps of dynamic environments (lots of room for research contributions here!)

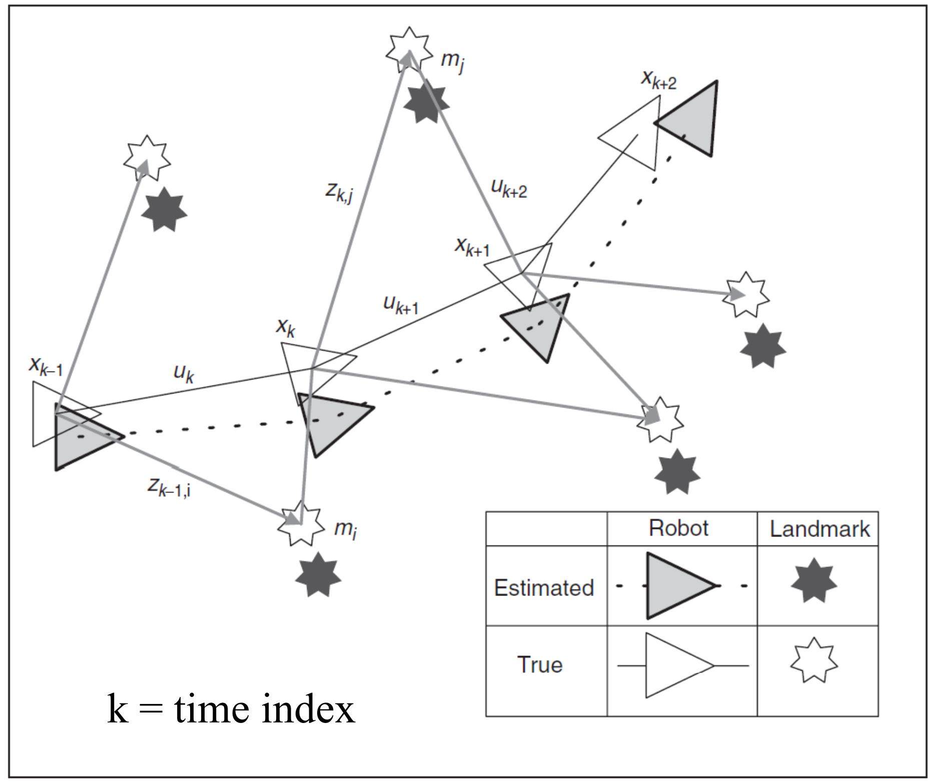
Robot Mapping Challenges
5. Path-Planning On-the-Fly

- robot must plan its path during mapping
- task of generating robot motion plans to build a map is referred to as *robotic exploration*
  - optimal path planning in a fully modelled environment is relatively well understood
  - robots in unknown environments has incomplete model
  - have to accommodate contingencies and surprises that arise during map building
    - generate plans in near real-time
    - where to move balanced against map information gain and time and energy to obtain info as well as possible loss of pose info along the way

The SLAM Problem

A mobile robot can build a map of an environment and at the same time use this map to deduce its location. The trajectory of the robot and the location of all landmarks are estimated on-line without the need for any a priori knowledge of location

- simultaneous estimate of both robot and landmark locations required
- true locations are never known or measured directly
- observations are made between the true robot and landmark locations.



Probabilistic SLAM
Recursive Solution

- compute the probability distribution for all times t

  $$p(X_{0:t}, m \mid Z_{0:t}, U_{0:t}, x_0) \qquad (*)$$

  this is the joint posterior density of the landmark location and vehicle state x_t given recorded observations Z & control inputs U (up to and including t) with initial vehicle pose x_o

- desire a *recursive* solution (i.e. calc from the same probability distribution from previous time step)
  - start with estimate for distribution

    $$p(x_{t-1}, m \mid Z_{0:t-1}, U_{0:t-1})$$

    at t-1, use Bayes theorem to determine the joint posterior, following control u_t and observation z_t

Probabilistic SLAM
Observation and Motion Models

- need motion (state transition) and observation models to describe the effect of the control input, u_t

- *observation* model when robot and landmark location known:

$$p(z_t \mid x_t, m)$$

- *motion* model for state transitions: $p(x_t \mid x_{t-1}, u_t)$

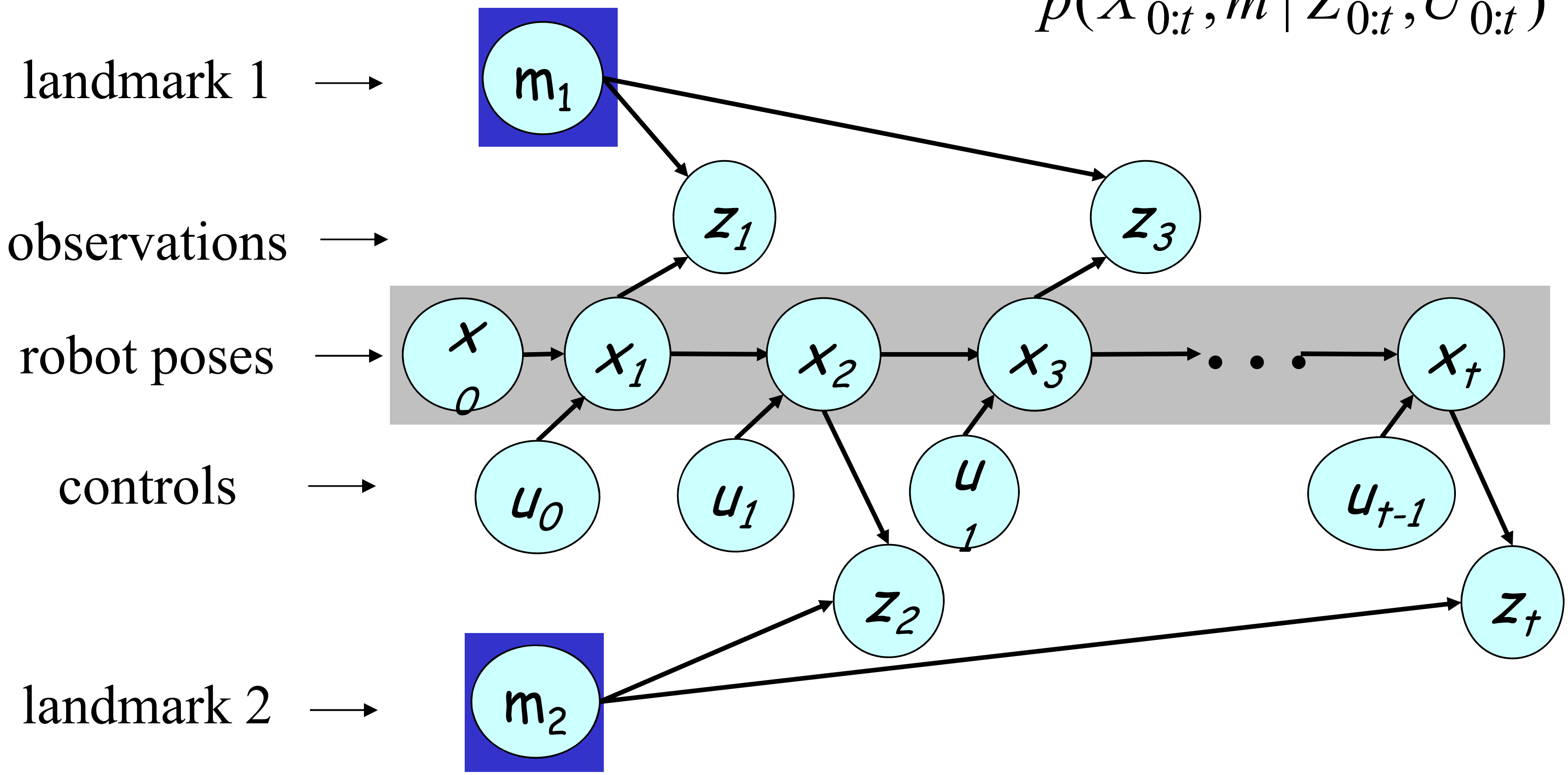
  state transition is assumed to be a Markov process where next state x_t depends only on the immediate state, $t\text{-}1$, before it and applied control u_t

  - independent of observation and map

SLAM
Problem Formulation

- no map available and no pose information

$$p(X_{0:t}, m \mid Z_{0:t}, U_{0:t})$$



Two Forms of SLAM

DALHOUSIE UNIVERSITY
Inspiring Minds
ACADIA UNIVERSITY

there are really two forms of the SLAM problem:

- full SLAM: estimates posterior for entire path (*0:t*) and map which is what is discussed so far (particle filter solution):

$$p(X_{0:t}, m \mid Z_{0:t}, U_{0:t})$$

- online SLAM: estimates posterior for current pose using most recent pose and map only (i.e. last time step) (EKF solution))

$$p(x_t, m \mid Z_{0:t}, U_{0:t}) = \int\int \ldots \int p(X_{0:t}, m \mid Z_{0:t}, U_{0:t})\, dx_0 dx_1 dx_2 \ldots dx_{t-1}$$

integrations typically done one at a time

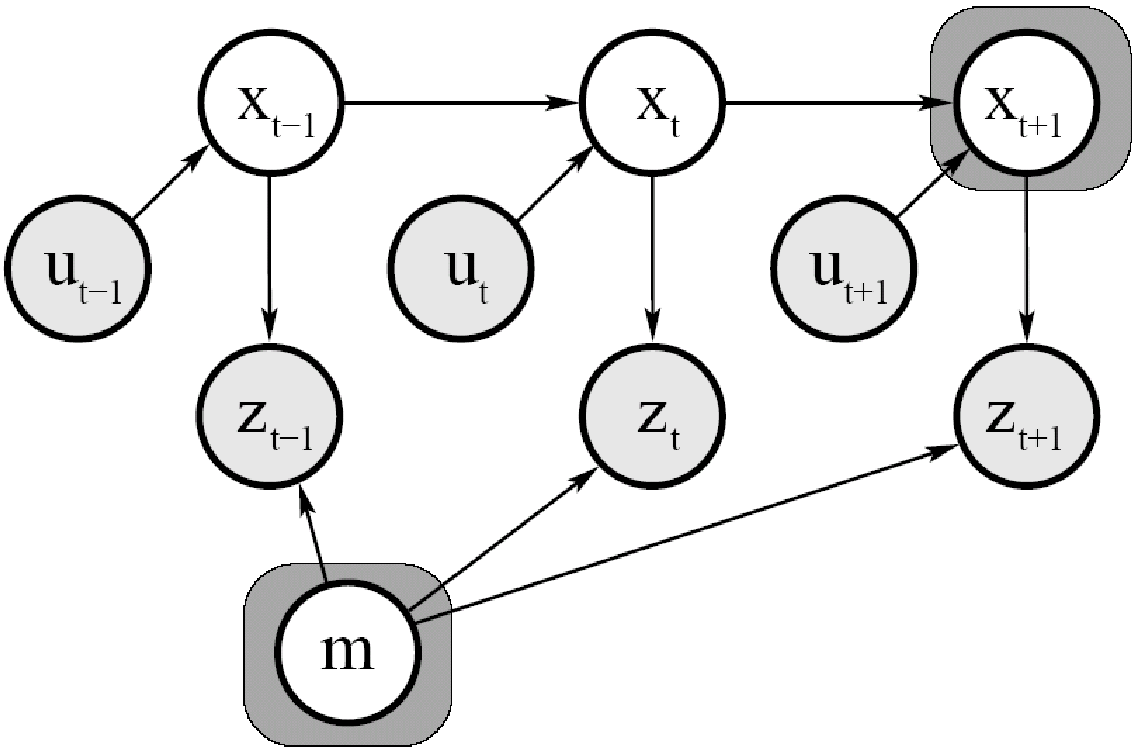
- discards past controls and measurements once processed since they are not used again

SLAM Feature

- a continuous and discrete component
- continuous
  - location of objects in the map and the robot pose
    - objects may be landmarks in the feature-based representation
    - object patches detected by range finders
- discrete (more on this later)
  - *correspondence* or *data association* between landmarks and measurements, i.e. how a newly detected object relates to previously detected ones
    - either the object was previously detected or it was not

On-line SLAM

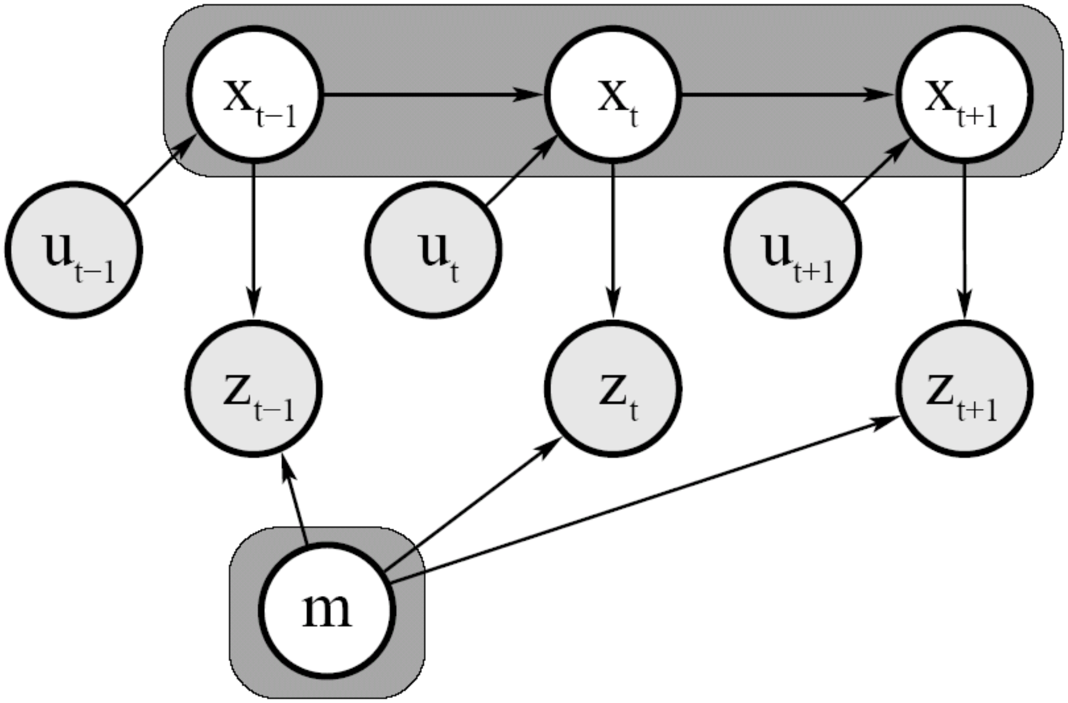
- graphical model of on-line SLAM (one pose at a time)



$$p(x_t, m \mid Z_{0:t}, U_{0:t}) = \int\int\ldots\int p(X_{0:t}, m \mid Z_{0:t}, U_{0:t})\, dx_0 dx_1 dx_2 \ldots dx_{t-1}$$

Full Blown SLAM

- graphical model of full blown SLAM



$$p(X_{0:t}, m \mid Z_{0:t}, U_{0:t})$$

Probabilistic SLAM

- SLAM implemented in standard 2-step recursive prediction (time update) correction (measurement update) form:

time update (prior distribution)

$$p(x_t, m \mid Z_{0:t-1}, U_{0:t}, x_0) = \int p(x_t \mid x_{t-1}, u_t) \times p(x_{t-1}, m \mid Z_{0:t-1}, U_{0:t-1}, x_0) dx_{t-1}$$

measurement update (posterior distribution)

$$p(x_t, m \mid Z_{0:t}, U_{0:t}, x_0) = \frac{p(z_t \mid x_t, m) p(x_t, m \mid Z_{0:t-1}, U_{0:t}, x_0)}{p(z_t \mid Z_{0:t-1}, U_{0:t})}$$

- now, have a recursive procedure for calculating

$$p(X_{0:t}, m \mid Z_{0:t}, U_{0:t}, x_0)$$

- for robot state x_t and map m at time t based on all control inputs U and observations Z as functions of the motion and observation models

Strength of SLAM

- the error between estimated & true landmark locations are common between landmarks and come from a single source: *errors in knowledge of where the robot is when the landmark observations were made*
  - landmark location error estimates are highly correlated
  - *relative* location between landmarks $m_i - m_j$ known with good accuracy even when absolute locations uncertain
  - correlations between landmark estimates increase monotonically as more and more observations are made
  - *knowledge of relative location of landmarks always improves and never diverges regardless of robot motion*
    - this is due to observations being nearly independent for *relative* locations between landmarks

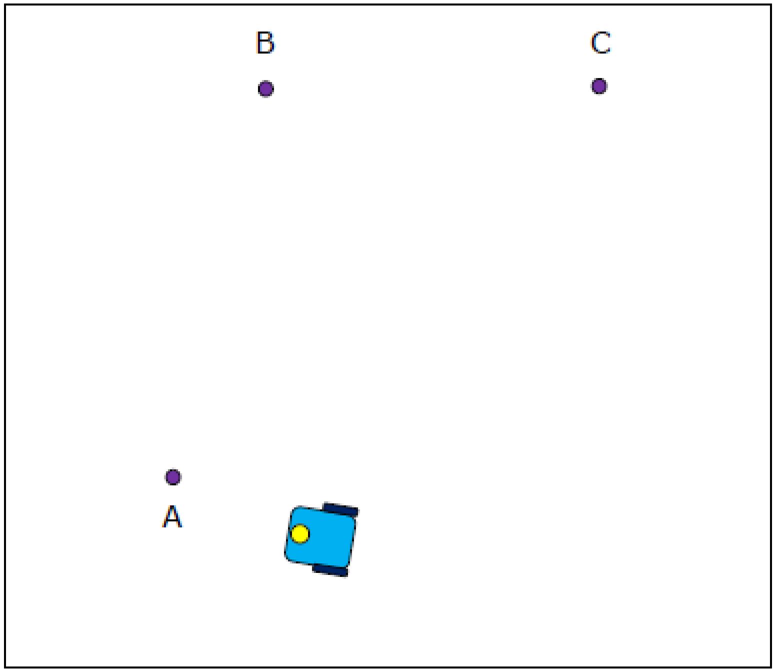
SLAM Solutions

- now require representations for:
  - motion model
  - observation model

  that allow efficient and consistent computation of the prior (time) and posterior (measurement) distributions
- most common representation is with state space model and additive Gaussian noise which leads to use of *extended Kalman filter* (EKF) solution
- alternative representation is to describe robot motion model as a set of samples of a more general non-Gaussian probability distribution which leads to the use of *particle filter* or FastSLAM as another solution
- there are many others but will only cover these two today

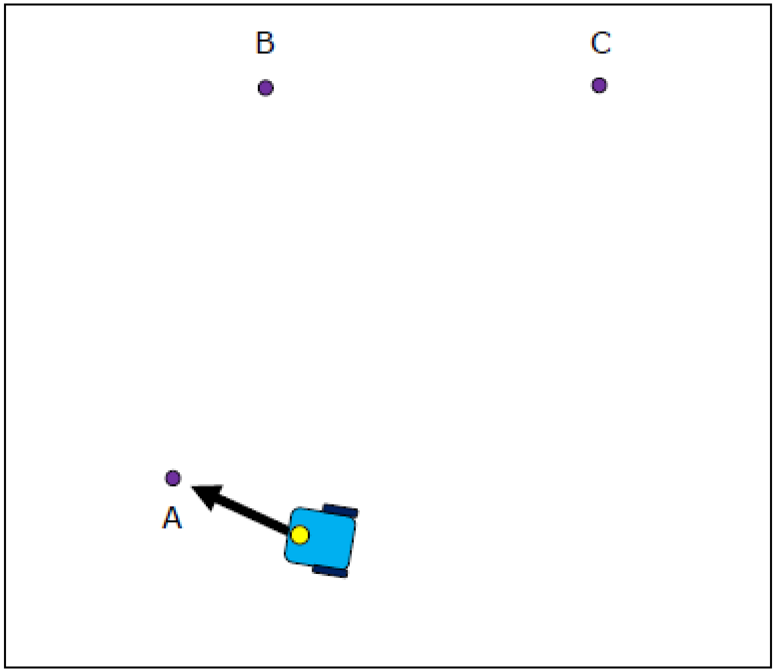
SLAM in Action – 1 / 9

- use internal representations for
  - positions of landmarks (map)
  - sensor parameters
- assume: robot uncertainty at start position is zero



start: robot has zero uncertainty

SLAM in Action – 2 / 9

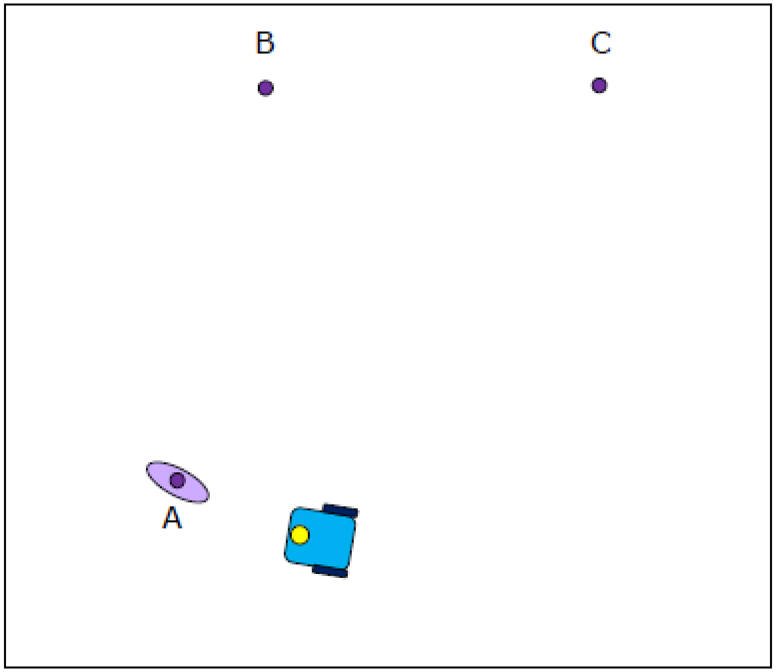


first measurement of feature A

- predict how the robot has moved
- **measure**
- update the internal representation

SLAM in Action – 3 / 9

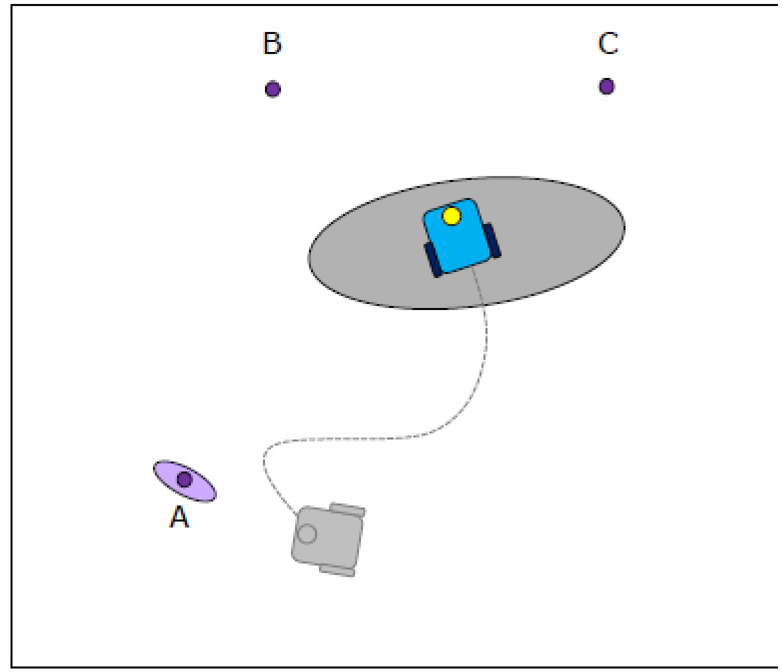
- robot observes a feature which is mapped with an uncertainty related to the sensor error model (i.e. *measurement model*)



- predict how the robot has moved
- measure
- **update** the internal representation

SLAM in Action – 4 / 9

- as robot moves (in response to the motion mode), its pose uncertainty increases

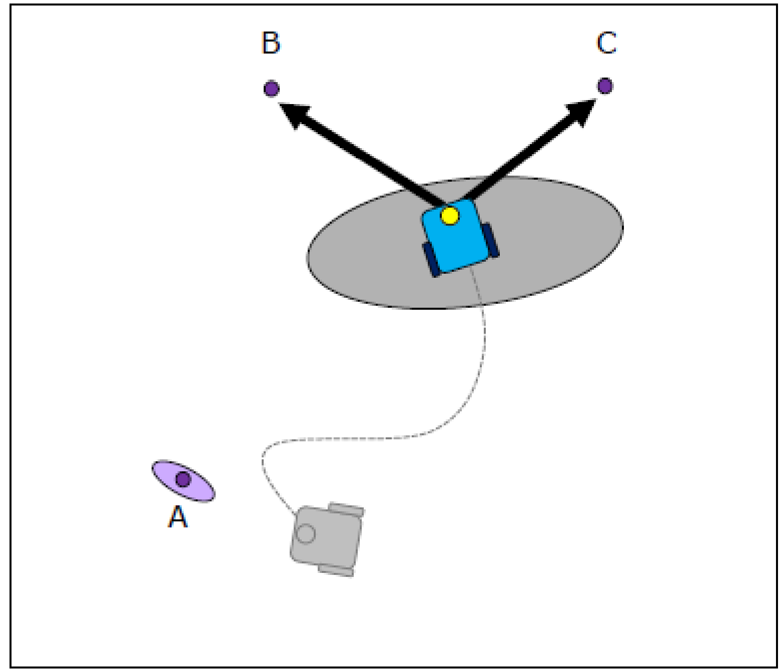


robot moves forwards: uncertainty grows

- **predict** how the robot has moved
- measure
- update the internal representation

SLAM in Action – 5 / 9

- robot observes two new features

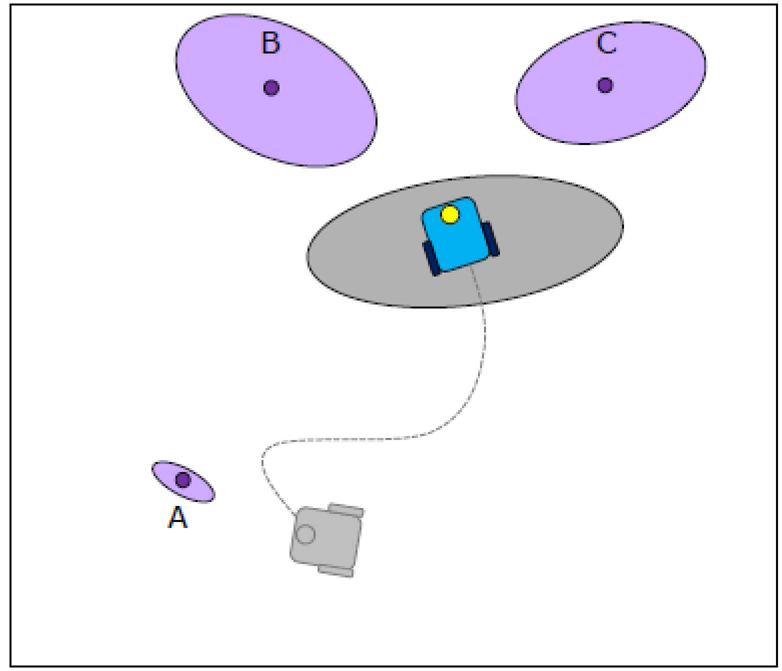


robot makes first measurements of B & C

- predict how the robot has moved
- **measure**
- update the internal representation

SLAM in Action – 6 / 9

- their position uncertainty results from the combination of the measurement error with the robot pose uncertainty
  - map becomes correlated with the robot position estimate

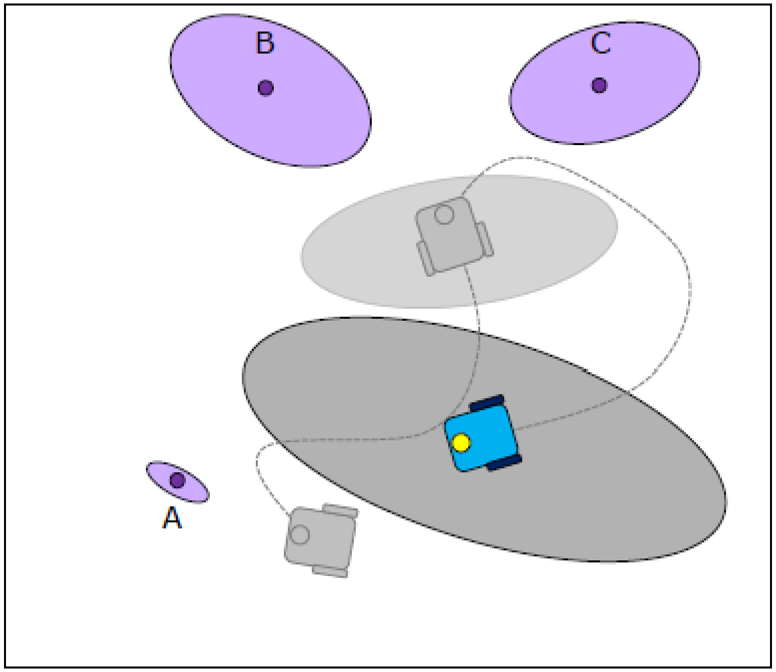


robot makes first measurement of B & C

- predict how the robot has moved
- measure
- **update** the internal representation

SLAM in Action – 7 / 9

- robot moves again and its uncertainty increases (motion model)

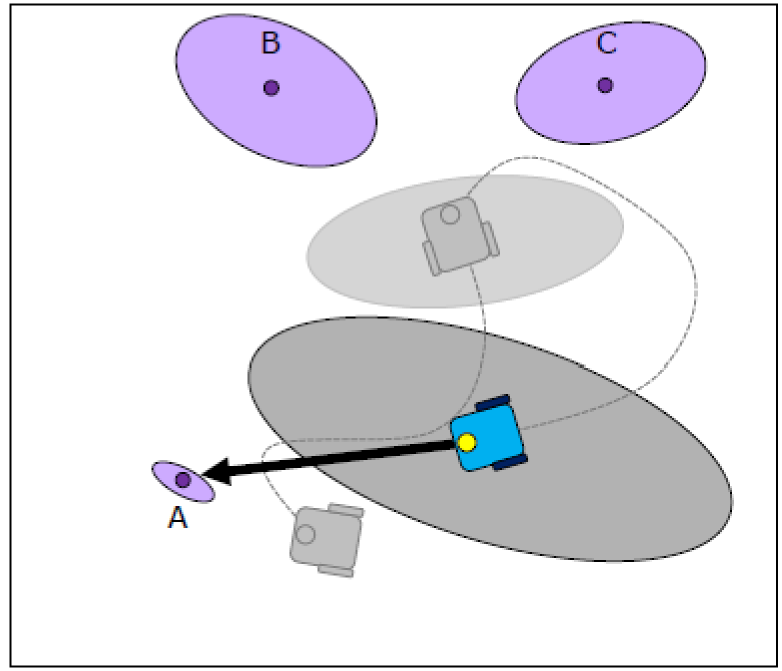


robot moves again: uncertainty grows still more

- **predict** how the robot has moved
- measure
- update the internal representation

SLAM in Action – 8 / 9

- robot re-observes an old feature
  - loop closure detection

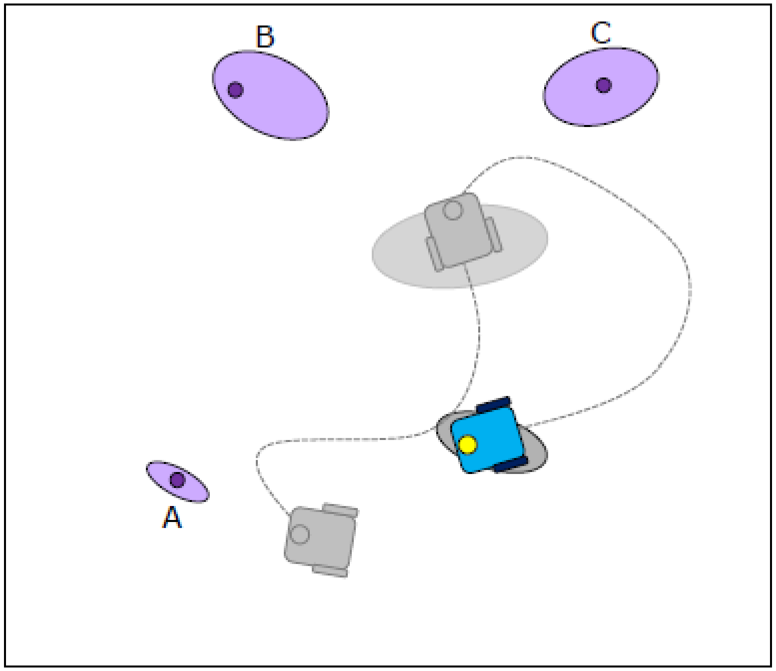


robot re-measures A: “loop closure”

- predict how the robot has moved
- **measure**
- update the internal representation

SLAM in Action – 9 / 9

- robot updates its position: the resulting position estimate becomes correlated with the feature location estimates
- robot's uncertainty decreases and so does the uncertainty in the rest of the map

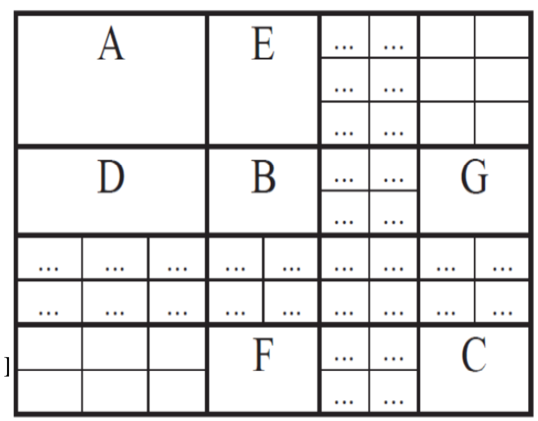


robot re-measures A: "loop closure"
uncertainty decreases

- predict how the robot has moved
- measure
- **update** the internal representation

About Covariance Matrix Σ

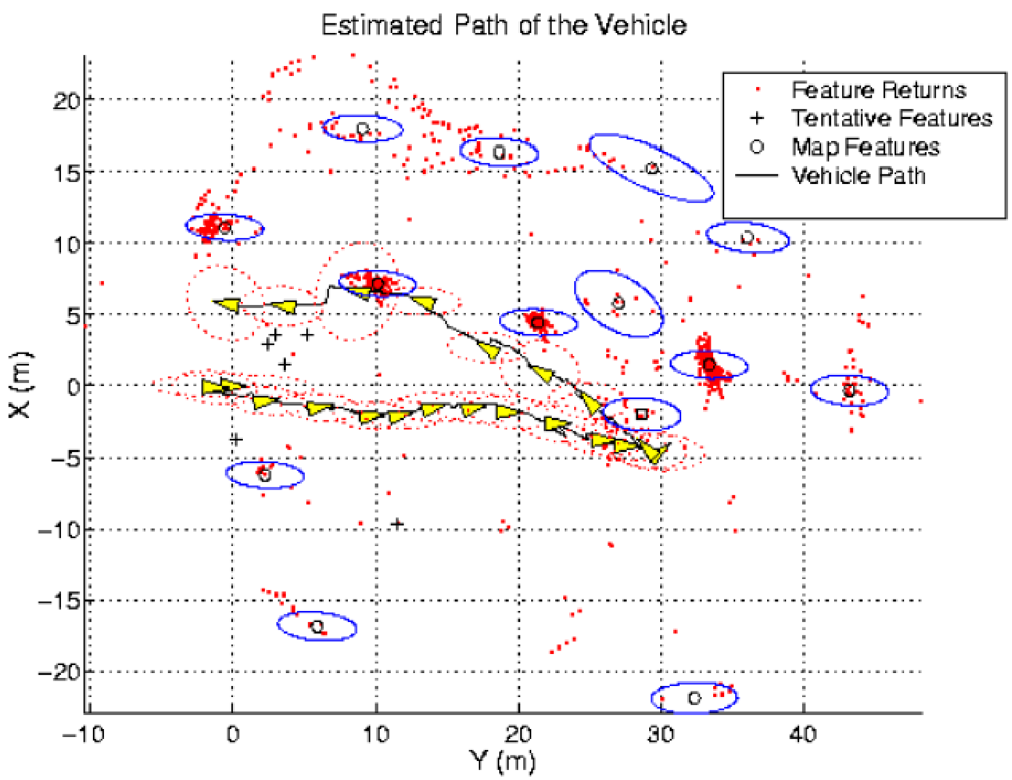
- correlation measures the degree of linear dependence between two variables
- covariance of two variables measure how strongly correlated two variables are
- covariance matrix Σ contains the covariance on: robot position, landmarks, between robot position and landmarks and between the landmarks
- cell A contains the covariance on robot position, a 3 by 3 matrix (x, y and θ)
- B is the covariance on the first landmark, a 2 by 2 matrix, since the landmark does not have orientation, θ ; C is covariance for the last landmark.
- D contains the covariance between the robot state and the first landmark; E contains the covariance between the first landmark and the robot state; E can be deduced from D by transposing sub-matrix D
- F contains the covariance between the last landmark and the first landmark, while G contains the covariance between the first landmark and the last landmark, which again can be deduced by transposing F
- $\mathrm{cov}(X, Y) = E\{[X - E(X)][Y - E(Y)]\}$
- $\mathrm{cor}(X, Y) = \mathrm{cov}(X, Y) / [\mathrm{sqrt}(\mathrm{var}(x)) * \mathrm{sqrt}(\mathrm{var}(Y))]$



covariance matrix Σ

EKF SLAM Implementation

- Kalman filters are Bayesian filters that represent posterior, $p(x_t, m \mid z_t, u_t)$ with Gaussians



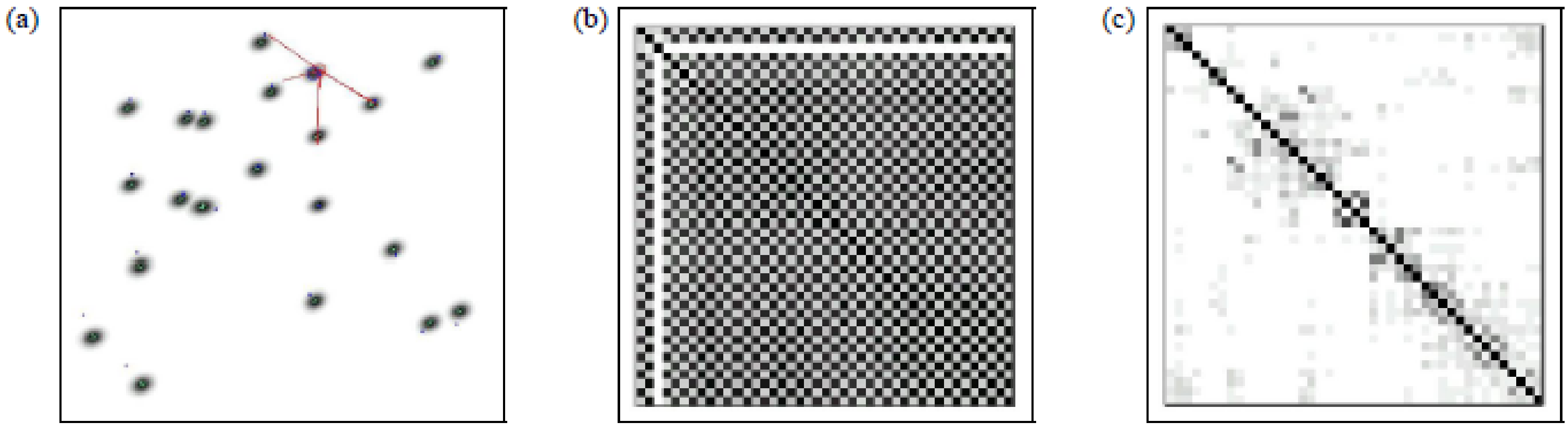
Example of Kalman filter estimation of the map and vehicle pose [1].

Shown is the path of an AUV with range measurements from a sonar. 14 features are identified from the sonar data.

Ellipse around features convey uncertainty that remains after mapping as specified by the covariance matrix

EKF SLAM Implementation
Results Mapped

(a) map of landmarks obtained in simulation (b) correlation matrix after 278 iterations of Kalman filter mapping. Checkerboard appearance verifies theoretical find that in the limit, all landmark location estimates are fully correlated (c) normalized inverse covariance matrix of the same estimate shows the dependencies are local.



Kalman Filtering
Assumptions

- three main ones:
  - (i) next state function (motion model) linear with added Gaussian noise
  - (ii) same is true of the perceptual model
  - (iii) the initial uncertainty must be Gaussian

Extended Kalman Filter (EKF)
State Model

- in a linear state function, robot pose x_t, and map m_t, at time t ~ linearly with previous pose x_{t-1}, map m_{t-1}, and control u_t
  - for map, obviously true since the map does not change
  - however, x_t usually governed by a trig function that varies nonlinearly with previous pose x_{t-1} and control u_t
    - to accommodate such nonlinearities Kalman filters approximate the robot motion model with a linear function obtained via Taylor series expansions to yield the extended Kalman Filter (EKF)
    - motion commands approximated by a series of smaller motion segment
      - usually works well for most robotic vehicles

EKF SLAM

State Motion Model

- $p(x_t \mid x_{t-1}, u_t) = Ax_{t-1} + Bu_t + w_t$
- A and B are matrices that implement linear mapping from state x_{t-1} and motion command u to state x_t
  – noise (assumed Gaussian) in motion is modeled via w_t which is assumed to be normally distributed with zero mean and covariance Q_t

more specifically,

- $p(x_t \mid x_{t-1}, u_t) \Leftrightarrow x_t = f(x_{t-1}, u_t) + w_t$

  where $f(..)$ models the robot dynamics / kinematics / odometry

EKF SLAM
Observation Model

DALHOUSIE UNIVERSITY *Inspiring Minds*

ACADIA UNIVERSITY

- sensor measurements usually nonlinear with non-Gaussian noise
- approximate through a first degree Taylor series expansion, i.e. $p(z_t | x_t, m) = Cx_t + v_t$
- C is a matrix (a linear mapping) and v_t is the normally distributed measurement noise with zero mean and covariance R_t

  more specifically, $p(z_t \mid x_t, m) \Leftrightarrow z_t = h(x_t, m) + v_t$
- where $h(..)$ describes the geometry of the observation

- these approximations work well for robots that can measure their ranges and bearings to landmarks

EKF SLAM
Overview

- similar to EKF implementation for robot localization
- EKF SLAM summarizes all past experience in an extended state vector, y_t compromising of robot pose x_t and the position of all map features m_t and an associated covariance matrix Σ_{yt} :

- for a MindStorm robot, size of $y_t = 3 + 2n$ since the n map feature have only 2 coordinates each
- size of $\Sigma_{yt} = (3+2n)^2$

$$y_t = \begin{bmatrix} x_t \\ m_t \\ \dots \\ m_{n-1} \end{bmatrix}, \quad \Sigma_{yt} = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xm_1} & .. & \Sigma_{xm_{n-1}} \\ \Sigma_{m_1 x} & \Sigma_{m_1 m_1} & .. & \Sigma_{m_1 m_{n-1}} \\ .. & .. & .. & .. \\ \Sigma_{m_{n-1} x} & \Sigma_{m_{n-1} m_1} & .. & \Sigma_{m_{n-1} m_{n-1}} \end{bmatrix}$$

- as robot moves and makes measurements, y_t and Σ_{yt} are updated with the standard EKF equations
- correlations are *important* for convergence, the more observations that are made the more correlations between the features will grow → better the SLAM solution

Compute Mean and Covariance
Time Update

- apply standard EKF method to calculate the mean

$$\begin{bmatrix} \hat{x}_{t|t} \\ \hat{m}_t \end{bmatrix} = E\begin{bmatrix} x_t \mid Z_{0:t} \\ m \mid Z_{0:t} \end{bmatrix}$$

- and covariance:

$$\Sigma_{t|t} = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{xm}^T & \Sigma_{mm} \end{bmatrix}_{t|t}$$

$$= E\left[\begin{pmatrix} x_t - \hat{x}_t \\ m - \hat{m}_t \end{pmatrix} \begin{pmatrix} x_t - \hat{x}_t \\ m - \hat{m}_t \end{pmatrix}^T |Z_{0:t} \right]$$

- of the joint posterior distribution $p(x_t, m|Z_{0:t}, U_{0:t}, x_0)$ from
- *time update*

$$\hat{x}_{t|t-1} = f(\hat{x}_{t-1|t-1}, u_t)$$

$$\Sigma_{xx,t|t-1} = \nabla f \, \Sigma_{xx,t-1|t-1} \nabla f^T + Q_t$$

such that ∇f is the Jacobian of f evaluated at the estimated $\hat{x}_{t\text{-}1|t\text{-}1}$

Compute Mean and Covariance Observation Update

DALHOUSIE UNIVERSITY *Inspiring Minds*

ACADIA UNIVERSITY

- *observation update*:

$$\begin{bmatrix} \hat{x}_{t|t} \\ \hat{m}_t \end{bmatrix} = \left[\hat{x}_{t|t} \hat{m}_{t-1}\right] + W_t \left[z_t - h(\hat{x}_{t|t-1}, \hat{m}_{t-1})\right]$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - W_t S_t W_t^T$$

such that

$$S_t = \nabla h \Sigma_{t|t\text{-}1} \nabla h^T + R_t$$

$$W_t = \Sigma_{t|t-1} \nabla h^T S_t^{-1}$$

and ∇h is the Jacobian of h evaluated at $\hat{x}_{t|t-1}$ and $\hat{m}_{t-1}$

http://www.youtube.com/watch?v=r-ogNDdHL34

EKF SLAM Drawbacks

convergence

- convergence of the map is based on the monotonic convergence of the determinant of the map covariance matrix ($\Sigma_{mm,t}$) and all landmark pair submatrices to zero

computational effort

- observation update step requires all landmarks and the covariance matrix be updated every time an observation is made → computation grows quadratically with # of landmarks, it is a little better than that with optimizations

EKF SLAM Drawbacks

DALHOUSIE UNIVERSITY *Inspiring Minds*

ACADIA UNIVERSITY

data association

- errors in associating observations with landmarks breaks it
  - loop-closure where a robot returns to re-observe landmarks after having been away a long time is difficult
  - especially difficult if landmarks are not simple points and look different from different directions (e.g. mines with side scan sonar images)

nonlinearity

- linearized versions of nonlinear model and observation models used
  - can result in huge inconsistencies in the solutions

FastSLAM

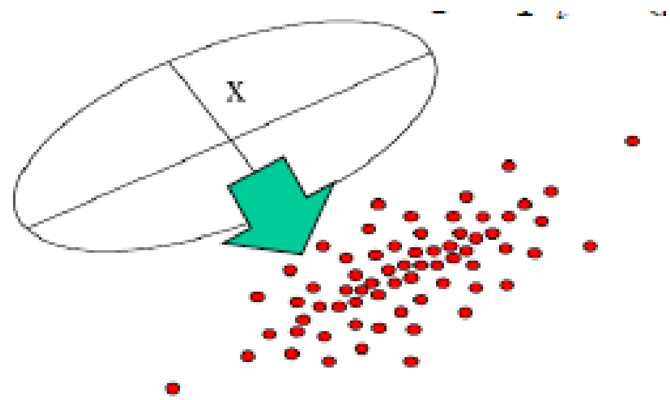
better solution: FastSLAM [2] using a *particle filter*

- fundamental shift in recursive probabilistic SLAM
- particle filter captures the nonlinear process model and non-Gaussian pose distribution for robot pose estimation
- Rao-Blackwellized method reduces computation effort (FastSLAM still linearizes observation model, like EKF)

Particle Filter SLAM Definitions

- particle filter: models that represent probability distributions as a set of discrete particles which occupy the state space
- particle: a point estimate of the state with an associated weight, w, $p_i = (y_{ti}, w_i)$

each particle defines a different vehicle trajecotry hypothesis



probability distribution (ellipse) as particle set (red dots)

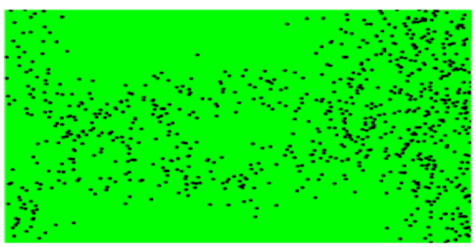
Particle Filters
Overview

- high dimensionality state-space of SLAM makes direct application of particle filters computationally infeasible
- it is possible to reduce the sample space by applying a particle filter where a joint space is partitioned according to product rule: $p(x_1, x_2) = p(x_2 \mid x_1)\, p(x_1)$
- if $p(x_2|x_1)$ can be represented analytically then only $p(x_1)$ need be sampled $x_1^{(i)} \sim p(x_1)$
- joint distribution is then represented by the set: $\left\{ x_1^{(i)}, p(x2 \mid x_1^{(i)} \right\}_i^N$ and statistics such as the marginal probability

$$p(x_2) \approx \frac{1}{N} \sum_i^N p\left(x_2 \mid x_1^{(i)}\right)$$

Particle Filters Overview

- recursive estimate performed by particle filtering for pose states and EKF for map states
- represents beliefs by random samples
- estimation of nonlinear, non-Gaussian processes
- Sampling Importance Re-Sampling (SIR) principle
  - draw the new generation of particles
  - assign an importance weight to each particle
  - re-sample as needed



weighted samples

after resampling

Implementation

DALHOUSIE UNIVERSITY *Inspiring Minds*

ACADIA UNIVERSITY

- as with EKF, the joint SLAM state may be factored into a robot component and a conditional map component:

$$p(x_{0:t}, m \mid Z_{0:t}, U_{0:t}, x_0)$$

$$= p(m \mid X_{0:t}, Z_{0:t})\, p(X_{0:t} \mid Z_{0:t}, U_{0:t}, x_0)$$

  - the probability distribution is on the trajectory $X_{0:t}$ rather than the single pose x_t
  - when conditioned on the trajectory the landmarks become independent – that is why particle filters are so fast
  - map is represented as a set of independent Gaussians

Implementation
Overview

- essential structure of FastSLAM is a Rao-Blackwellized (RB) state where the trajectory is modelled by weighted samples and the map is determined analytically
- joint distribution at time t, is represented by the set:

$$\left\{ w_t^{(i)}, X_{0:t}^{(i)}, p(m \mid X_{0:t}^{(i)}, Z_{0:t}) \right\}_i^N$$

where the map associated with each particle is composed of independent Gaussian distributions:

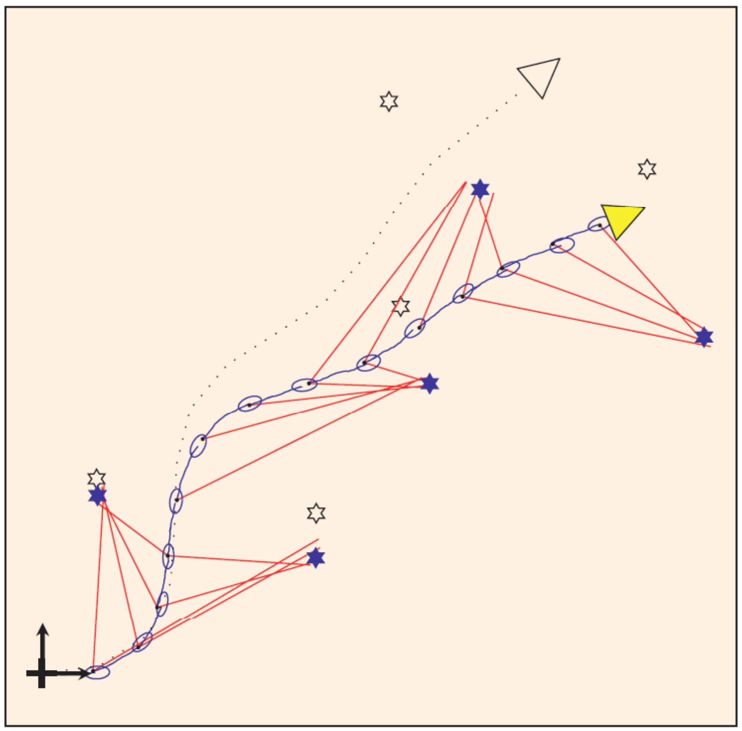
$$p(m \mid X_{0:Y}^{(i)}, Z_{0:t}) = \prod_j^M p(m_j \mid X_{0:t}^{(i)}, Z_{0:t})$$

- recursive estimation performed by particle filtering for the pose states and the EKF still for the map states

Implementation

Map

- updating map for given trajectory particle $X_{0:t}^{(i)}$ is trivial
- each observed landmark is processed individually as an EKF measurement update from a known pose
- unobserved landmarks are unchanged



A single realization of robot trajectory in the FastSLAM process. Ellipsoids show the proposal distribution for each update stage from which a robot pose is sampled, and, assuming this pose is perfect, the observed landmarks are updated. Thus, the map for a single particle is governed by the accuracy of the trajectory. Many of these trajectories provide a probabilistic model of robot location.

Implementation
Pose States

- propagating the pose particles is much more complex
- particle filter is derived from a recursive form of sample, *sequential important sampling* (SIS) which samples from a joint state history and 'telescopes' the joint into a recursion via the product rule:

  $$p(x_0, x_1, ..., x_T \mid Z_{0:T}) = p(x_0 \mid Z_{0:T})p(x_1 \mid x_0, Z_{0:T}), ..., p(x_T \mid X_{0:T-1}, Z_{0:T})$$

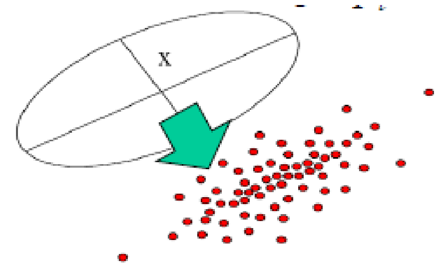
  at each time step t, particles are drawn from a *proposal distribution*: $\pi(x_t \mid X_{0:t-1}, Z_{0:t})$ which approximates the true distribution $p(x_t \mid X_{0:t-1}, Z_{0:T})$ and the samples are given *importance weights*
- approximation error grows with time increasing the variation in sample weights and thus degrade the statistical accuracy

Implementation
Pose States

- resampling step reinstates uniform weighting but causes loss of historical particle information
- SIS with resampling produces reasonable statistics only for systems that 'exponentially forget' their past
- general form for RB particle filter for SLAM:
  - assume at time *t-1* the joint state is represented by:

$$\left\{ w_{t-1}^{(i)}, X_{0:t-1}^{(i)}, p(m \mid X_{0:t-1}^{(i)}, Z_{0:t-1}) \right\}_i^N$$

Implementation Steps



probability distribution (ellipse) as particle set (red dots)

- **predict**
  - apply motion prediction to each particle
- **make measurements**
- **update**, for each particle:
  - compare particle's predictions of measurements with the actual measurements
  - assign weights such that particles with good predictions have higher weights
- **normalize** weight of particles to sum to 1
- **resample**: generate new set of M particles which all have equal weights $1/M$ reflecting probability density of last particle set

Particle Filter SLAM Format

1. for each particle, compute a proposal distribution, conditioned on the specific particle history, draw a sample from it: $x_t^{(i)} \sim \pi(x_t \mid X_{0:t-1}^{(i)}, Z_{0:t}, u_t)$
   this new sample is joined to the particle history $X_{0:t}^{(i)} = \left\{ X_{0:t=1}^{(i)}, x_t^{(i)} \right\}$
2. weight samples according to the importance function

$$w_t^{(i)} = w(i)_{t-1} \times \frac{p(z_t \mid X_{0:t-1}^{(i)}, Z_{0:t-1})}{\pi(x_t^{(i)} \mid X_{0:t-1}^{(i)}, Z_{0:t}, u_t)}$$

the numerator terms are the observation model and the motion model; the observation model differs because RB requires dependency on the map be marginalized away.

$$p(z_t \mid X_{0:t}, Z_{0:t-1}) = \int p(z_t \mid x_t, m)\, p(m \mid X_{0:t-1}, Z_{0:t-1}) dm$$

Particle Filter SLAM Format

3. If necessary, resample. When best to resample is an open problem. Resampling is accomplished by selecting particles, with replacement, from the set $\left\{X_{0:t}^{(i)}\right\}_N^{(i)}$ including associated maps, with probably of selection proportional to $w_t^{(i)}$. Selected particles are given uniform weight, $w_t^{(i)} = 1/N$

4. For each particle, perform an EKF update on the observed landmarks as a simple mapping operation with known vehicle pose.

Particle Filter SLAM Format

- several implementations of FastSLAM (particle filter), most complete is FastSLAM 2.0

- For FastSLAM 2.0, the proposal distribution includes the current observation: $x_t^{(i)} \sim p(x_t \mid X_{0:t}^{(i)}, u_t)$

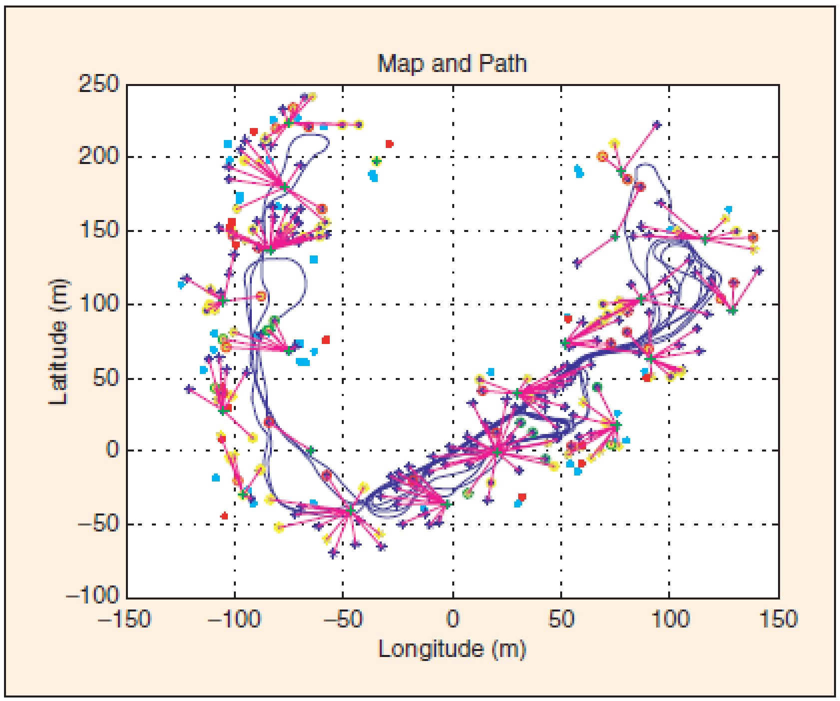
  such that:
  $$p(x_t \mid X_{0:t-1}^{(i)}, Z_{0:t}, u_t) = \frac{1}{C} p(z_t \mid x_t, X_{0:t-1}^{(i)}, Z_{0:t-1}) p(x_t \mid x_{t-1}^{(i)}, u_t)$$

  where C is a normalizing constant

  importance weight is $w_t^{(i)} = w_{t-1}^{(i)} C$

Particle Filter SLAM Format

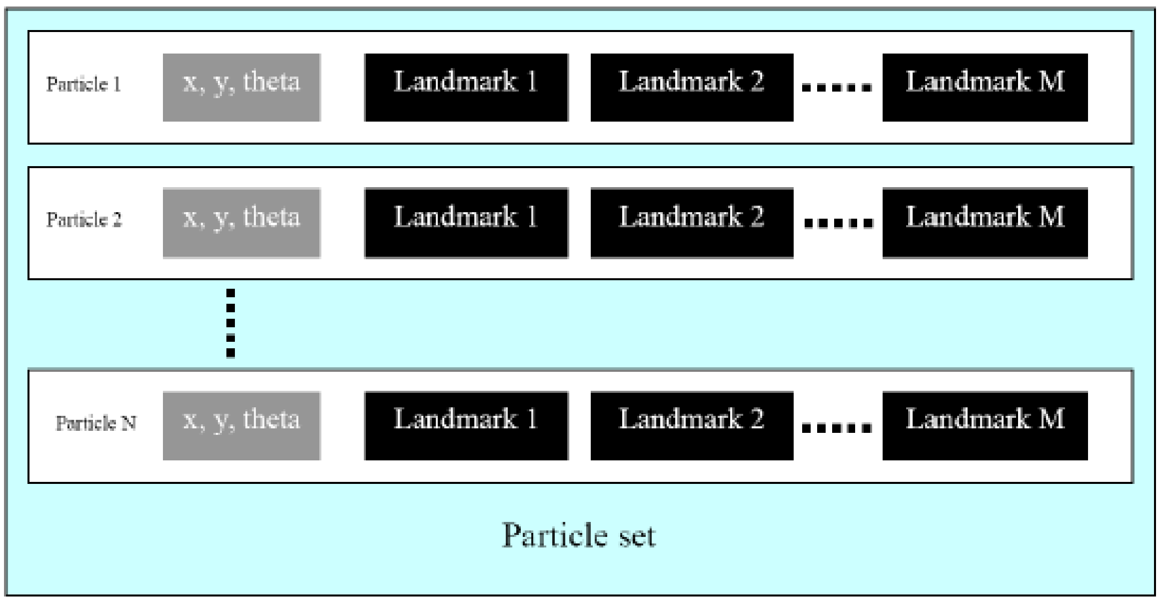
- proposal distribution is locally optimal – each particle gives the smallest possible variance in importance weight condition upon available information $X_{0:t-1}^{(i)}, Z_{0:t}$, and $U_{0:t}$



large scale outdoor SLAM [3]

Particle Filter SLAM FastSLAM Approach

- solve state posterior using Rao-Blackwellized Particle Filter
- each landmark estimate is represented by a 2x2 EKF
- each particle is independent (due to factorization) from the others and maintains the estimate of M landmark positions



http://www.youtube.com/watch?v=m3L8OfbTXH0

Underwater SLAM

- use natural features of environment for navigation important in applications where odometry and direction sensors are *unavailable*
- for e.g. ship hull inspection by an AUV where sonar imaging and range sensing present cost-effective alternatives to high precision inertial navigation, and u/w ops near a large steel structure means no compass, GPS, or long baseline acoustic tracking
- a planar marine vehicle using range and bearing measurements of a set of point features to traverse a path with time-varying controller and estimator gains

Navigation of AUV

- success of future AUVs lies in the ability to accurately localize itself within the underwater domain
- underwater world limits the types of sensor available compared to above water
- GPS is not available underwater
- however, if truly autonomous underwater vehicles are to be developed, good navigation sensory information is needed to achieve mission goals and provide safe operation

Navigation of AUV
Current AUV Navigation Schemes

- inertial navigation
  - uses gyroscopic sensors to detect the acceleration of the AUV
  - significant improvement over dead reckoning and is often combined with a Doppler velocity log which can measure the AUV's relative velocity
- acoustic navigation
  - uses transponder beacons to allow AUV to determine its position
  - most common method are long baseline which uses at least two, widely separated transponders and ultra-short baseline which uses GPS calibrated transponders on a single surface ship

Navigation of AUV
Current AUV Navigation Schemes

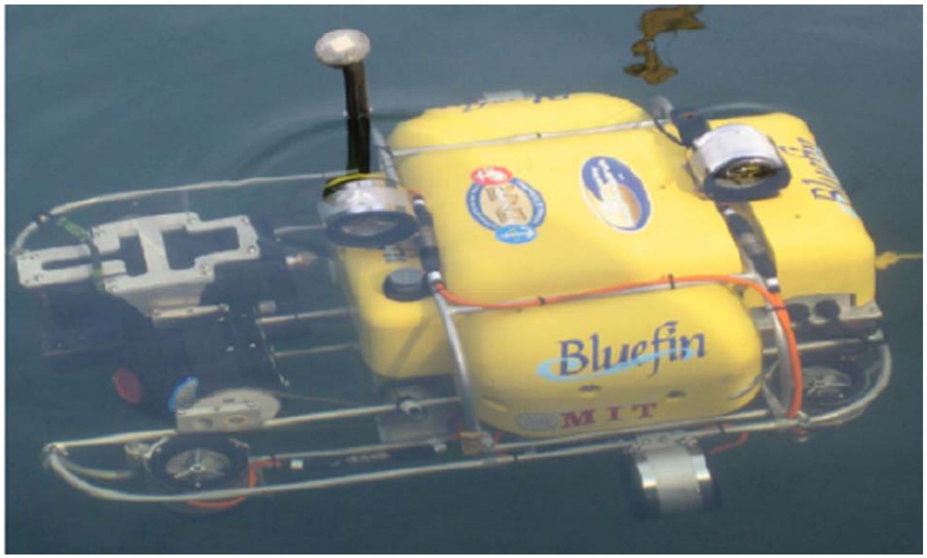
- geophysical navigation
  - uses physical features of the AUV's environment to produce an estimate of the AUV location
  - there can be pre-existing or purposefully deployed features
- most current AUV's are equipped with sensors which can make use of a combination of all three methods
  - different sensor data from each method needs to be processed together throughout a mission to obtain an optimal estimate of the AUV position

Navigation of AUV
Current AUV Navigation Schemes

- techniques currently used for deriving an estimate of the AUV's position from such sensor data are
  - Kalman filters
  - particle filters
  - SLAM

Ship Hull Monitoring

- SLAM applied to ship hull inspections

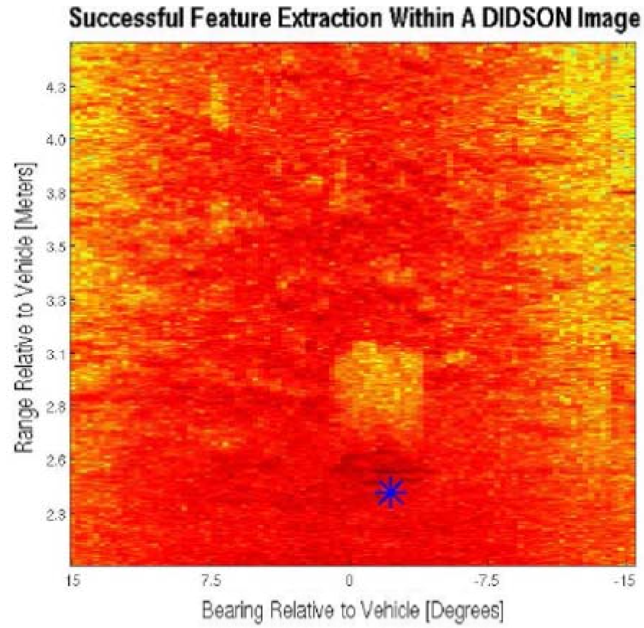


MIT Bluefin Hovering Autonomous Underwater Vehicle (HAUV) designed to perform autonomous ship hull inspections using SLAM. Identified mine-like objects using DIDSON imaging sonar in real-time.

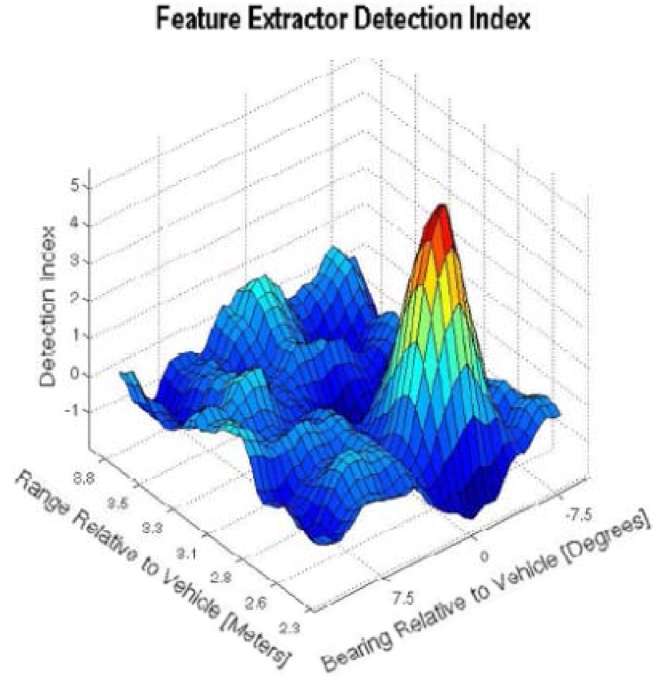
DALHOUSIE UNIVERSITY *Inspiring Minds*

ACADIA UNIVERSITY

Ship Hull Monitoring
Feature Extraction



Feature Extractor Detection Index

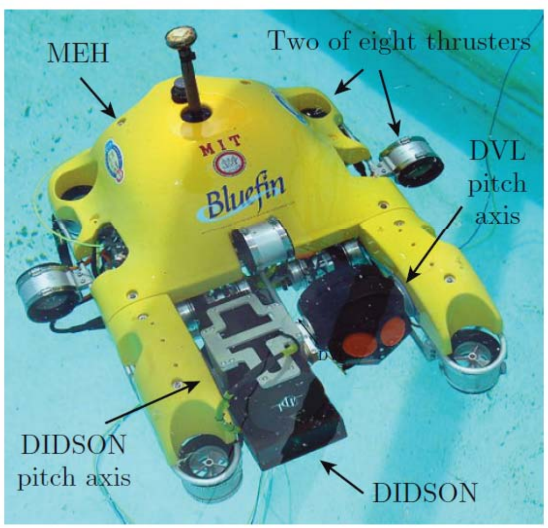


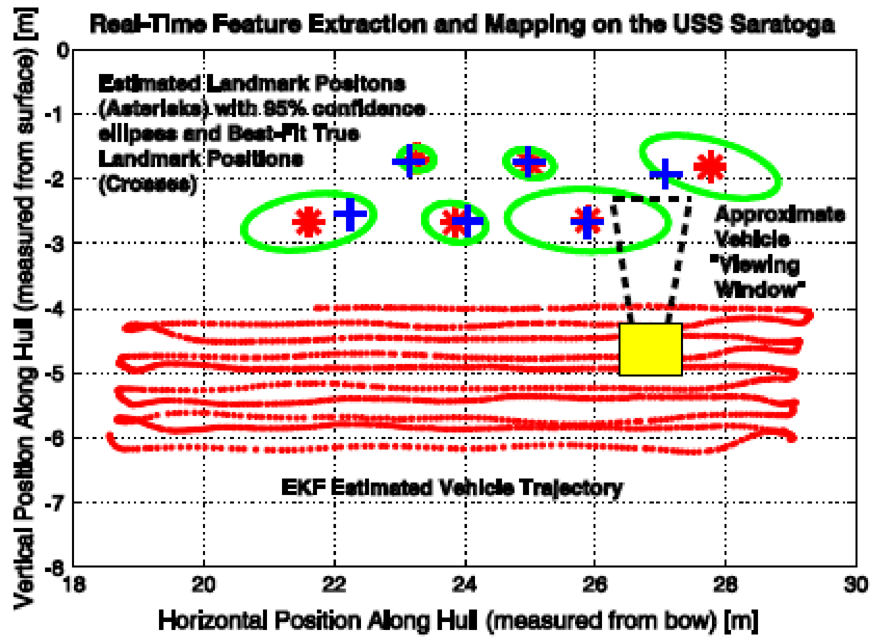
Performance of real-time feature extractor demonstrated using a DIDSON frame. Raw data (left) and the feature extractor detection index for each rectangular quadrant of image (right). Areas where features were identified (indicated by blue asterisk) correspond to high peaks in the feature detection index.

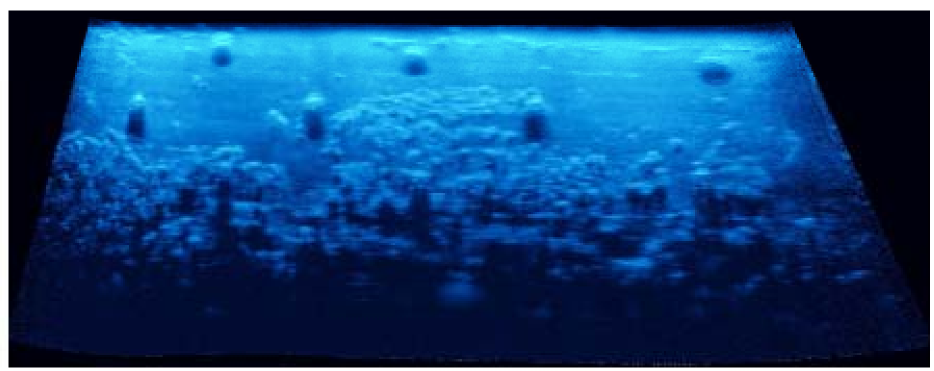
Ship Hull Monitoring
Feature Extraction

(a) Real-time map and vehicle localization data obstained from a survey of the USS Satatoga using an EKF.

(b) A sonar mosaic image of the targets placed on the ship hull







Multi-Vehicle SLAM

- group of unmanned surface vehicles (USV) for shallow water hydrographic missions more efficiently and reliably than a single one over a large environment
- issues of inter-vehicle map fusion and data association
- some level of collaboration required

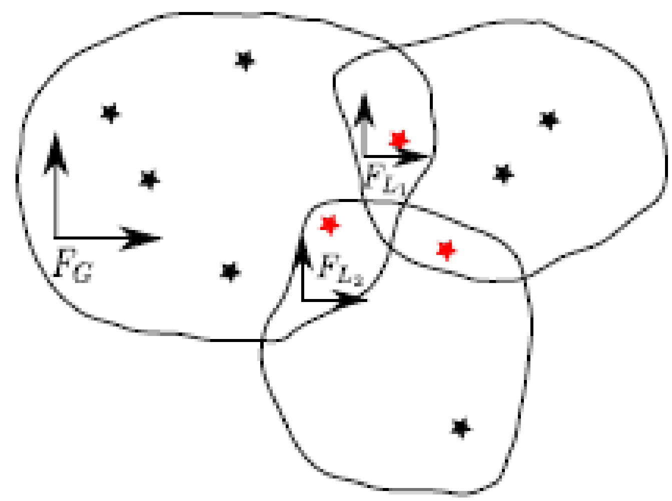
Multi-Vehicle SLAM

- multi-beam sonar scanners used to extract features and objects on the seabed
  - combine features with accurate positional information to build maps
- each USV performs SLAM independently over its local region and at specified times fuses these independent measurements to build an overall global map while improving each vehicle's position estimates
  - combining information from multiple USVs challenged by compounding positional errors of individual USVs and varying uncertainties and sensor noise characteristics
  - scalability for numbers of vehicles can be an issue

Multi-Vehicle SLAM

- local sub maps not only facilitate improved data association but significantly improved performance gains due to periodic map fusions (mosaicking)
  – achieved through identifying common features from overlapping areas

Two robots mapping independently with respect to local frames of reference. F_G refers to the global reference frame while F_{L1} and F_{L2} refers to the local reference frame of the two robots. Black stars in local frames of reference correspond to the features mapped by each vehicle and red ones correspond to the overlapping feature.



Concluding Remarks

- SLAM is one of the most difficult problems in robotics
- EKF and particle filter are the two most popular solutions for the SLAM problem
  - particle filter is a more robust solution but there are researchers in underwater SLAM that get good results with EKF
- underwater SLAM is an area that is receiving more attention

References

[1] S.Williams, G. Dissanayake, and H.F. Durrant-Whyte, "Towards terrain-aided navigation for underwater robotics," *Advanced Robotics, 15(5), 2001.*

[2] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fast-SLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. AAAI Nat. Conf. Artif. Intell., 2002, pp. 593–598.*

[3] J.E. Guivant and E.M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Trans. Robot. Automat., vol. 17, no. 3, pp. 242–257, 2001.*

hw #3, quest 2, part (ii)

- for range of 10 units, range res = 0.25, ang res = 2.5 deg