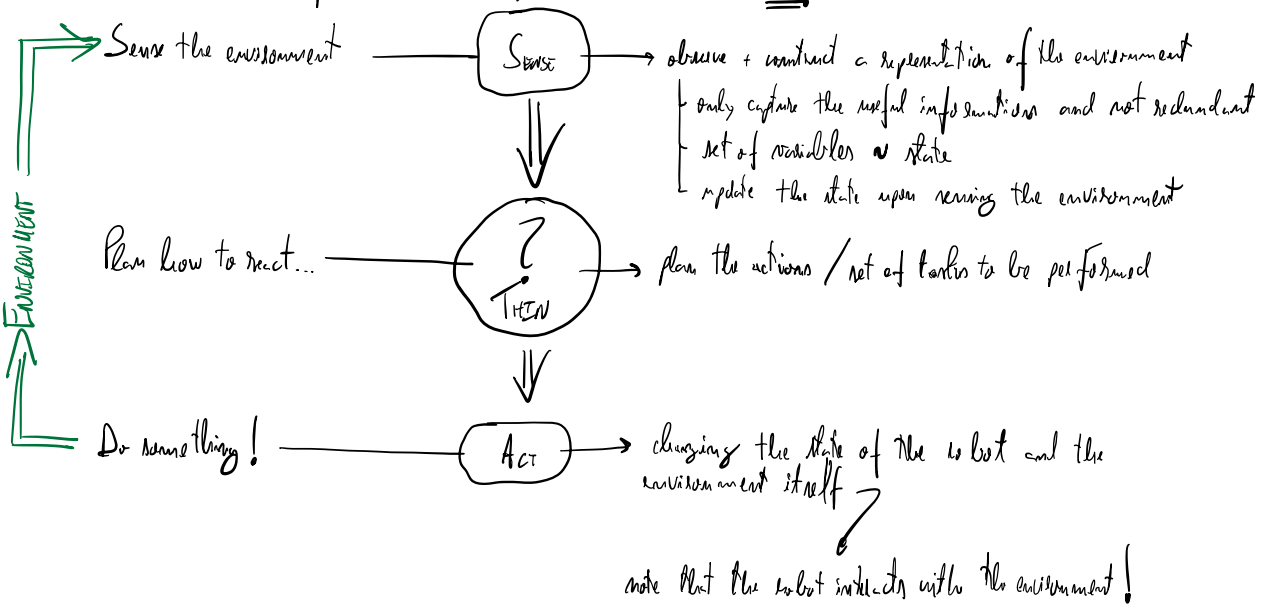
Probabilistic Robotics - Filtering

(RBS Notes) → based on Robotics @ Sapienza University of Rome (Prof. Dr. Giorgio Grisetti)

Introduction

"The more useful the machine is, the more human it can do!"



And when the state representation built from the perception does not correspond to the environment?

↳ wrong state/observation models

↳ erroneous data associations

...

↳ **State in Robotics** → one of the main goals of the probabilistic robotics is focused on state estimation

↓

Generic definition of state: all we need to predict in the future and forget in the past

- model of the world
  - geometry of the environment
  - other moving objects
  - traversability (where the robot can go through the environment, define which paths can traverse)
  - tracking other vehicles to avoid obstacles / collisions
  - ...
- robot configuration
  - Kinematics ⊕ dynamics
  - batteries
  - internal calibration parameters (useful to improve cheap devices performance)
  - ...

↳ **Probability and Robotics** — "the more complicated the model is, the more complicated is to use / configure / tune it..."

- model of a system is typically an abstraction of a more complex real entity
- disturbances affect the measurements and the system

> models of the systems themselves are the ones that produce errors!
> (if we had a perfect model → no errors)

⇓

Predicted Behavior ≠ Real

(errors in the model affect the real actions that the robot will take due to model errors)

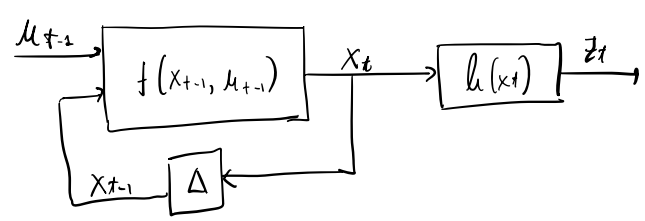
||

more light in camera:

- open the shutter (physically, may not be possible)
- increase exposure time
- increase a gain (but if noise in the camera, gains also increases noise)
  - there is a small amplifier on the chip to amplify the camera signal

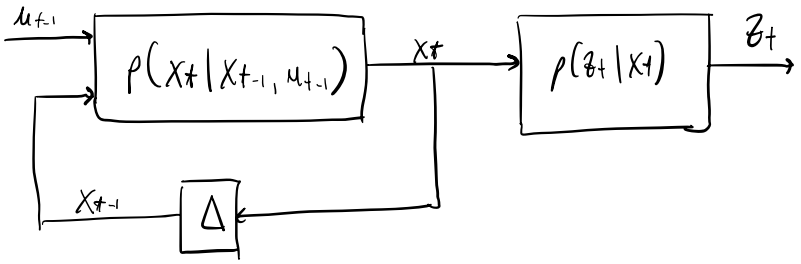
instead of single values, we have a set of possible solutions ⇒ PROBABILISTIC INFERENCE

SYSTEM MODEL



u: controls
x: state
z: measurement
f: transition function
h: observation function

Perfect knowledge of
- inputs
- measurements
- transition model
- observation model



PROBABILISTIC MODEL

$x_t \sim p(x_t | x_{t-1}, u_{t-1})$

$z_t \sim p(z_t | x_t)$

↳ all variables become stochastic (w/ uncertainty)

↳ transition, observation functions ⇒ models

↳ if no uncertainty is present the probabilistic models degenerate to the deterministic models

PROB LAB COURSE CONTENTS

FILTERING
estimate the distribution over the possible current states of a dynamic system

↓

access to all controls & measurements (all we know up to current time!)

not the future

DATA ASSOCIATION
PRE-PROCESSMENT to determine which state variables is responsible of a measurement

↓

multiple doors and see one of them at a certain time

⇓

we need to discriminate the position of the robot based on the knowledge where the doors are and current estimated pose of the robot (and its uncertainty)

MAXIMUM LIKELIHOOD ESTIMATION (MLE)
estimate most likely trajectory of the system state given all measurements so far

|||

figuring out the set/configuration of states which better explain the measurements

SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM)
Graph SLAM

||

minimization configuration of the system by redistributing the error in the graph

∿

most likely possible distribution of the graph

CALIBRATION
Kinematic parameters of a robotic systems

extrinsic parameters of sensors

TRACKING
Track the position of a moving sensor by estimating the egomotion

(e.g.: 3D RECONSTRUCTION)

LOCALIZATION
- online / offline
- local / global
- ....

TOOLS
- Linux
- Octave
- C/C++ (OS, g2o, V-REP)

Robots & Sensors

Sensors for Egomotion (relative) (absolute)

- Wheel encoders: measure the revolutions count / location of the motor's shaft
  - optical → light passes through the holes, square signals allow count of encoder ticks
  - magnetic → hall-effect
- Inertial Measurement Units (IMUs) → you can see the IMU of your phone w/ the app DEVICE INFO
  - accelerometers
  - gyroscopes
  - magnetometers (not all IMUs have them)

→ How to estimate egomotion?
- integrate sensor measurements
- INTEGRATION ⇒ ACCUMULATED DRIFT (due to the noise affecting the measurement)

only egomotion not possible to localize a robot for a long-time ...

⇓

external reference systems / observations allow to correct the error of the ego motion estimation!

Measuring the Environment

- Active: inject energy into the environment ⇒ measurement = return of the energy
  - ultrasound
  - laser range finder
  - structured-light cameras
  - infrared
- Passive: measure the energy of the environment itself (without adding/injecting any into it!)
  - RGB cameras
  - Tactiles

⇓

→ Sonars ≡ Sound Navigation and Ranging
- emit sound ⊕ measure echo
- sound not very directional specially on the axis (but water is a lot more directional!)
- sound disperses according to a conic form

⇓

v_{sound} @ air $\simeq 345\ m/s$ → to measure 3000 m ⇒ ν 0.5 Hz (max reading frequency)

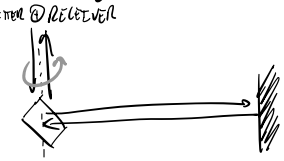
note that (device needs the signal and waits for its return)

extensive Field Of View (FOV)

(2D: Hokuyo UST-10LX, Sick, RPlidar | 3D: RoboSense, Ouster, Velodyne, ...)

→ Laser Range Finders

EMITTER ⊕ RECEIVER



- Beam of pulsing light
- emitter sends the light ⊕ receiver waits for its return
- ROTATING MIRROR!

⇓

if mirror tilts, we have a 3D scanner

- are approved in terms of security standards for collision detection

→ RGB Monocular Cameras

monocular limitability of depth

2D projection of a 3D structure ⊕ light intensity of each pixel

⇓

pin-hole cameras tend to curve straight lines especially in monocular cameras (especially in fish-hole cameras ν 180° FOV)

- EXPOSURE TIME: how long the image is being captured
  - ↑ Δt ⇒ ↑ blur
  - ↑ Δt (shutter open) ⇒ ↓ ν max. robot
  - ↓ Δt ⇒ ↓ light
- image sensitivity
- single frame, single camera ⇒ unobservable depth

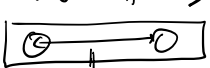
⇓

BUT multiple images → STRUCTURE FROM MOTION

→ RGB Stereo Cameras

allow to measure distance (from corresponding features)

- stereo cameras = 2 monocular cameras that allow Triangulation (epipolar geometry)
- error in depth depends on the distance
- textureless environments ⇒ (usually) cannot determine depth



this distance influences the minimum and maximum range of the sensor!

→ RGB-D Cameras – color ⊕ depth

- active light source to retrieve the depth:
  - stereo triangulation (emitter + receiver in different positions)
  - time-of-flight (ToF) (emitter + receiver SAME position)
- environment conditions should allow to sense emitted light
- typically, ok indoors

Mobile Bases

- Mechanical structure
  - Kinematic configuration / steering geometry (differential, tricycle, omnidirectional, ackermann)
  - robot domain (air, water, ground)
  - motion structures (wheels, fans, blades, ...)
  - ...
- Sensors ⊕ Actuators
  - proprioceptive (internal robot's state)
  - exteroceptive (environment info)
  - ...
  - (think the other categorizations for sensors)
- Electrical connections (batteries, drives to power motors, ...)

Ground mobile robots

autonomous navigation

↕

path planning + localization + control + ...

↕

linear / angular velocity

↕

joints space

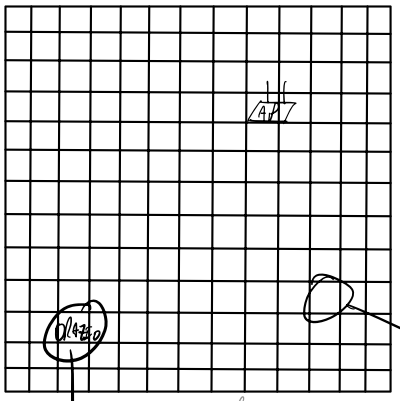
Introduction to Probabilities

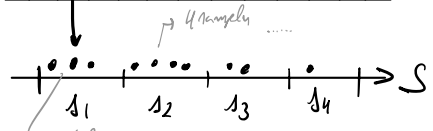
Scenario

Obelix = mobile platform
AP = wi-fi access point

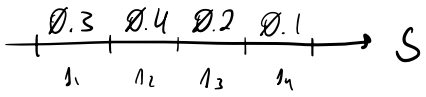
What is the wi-fi signal strength recorded by Obelix at different locations?

(4-bar indicator
4 = maximum strength
1 = weak signal
∅ = no signal at all)

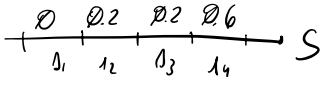




3 samples of l_j with $S = s_1$



(where $N = 10$)



...

⇒

1. put Obelix in different locations
2. start recording samples of measured signal strengths
   - gives us a statistic on the signal strength
3. define a set of intervals on the strengths
4. to each location, assign a number:

$$P_{l_j}(S = s_i) = \frac{1}{N} N_i$$

- l_j: different locations in the building
- $s_i|_{i=1,2,3,4}$: signal strength intervals considered
- N_i: number of times signal strength s_i was observed with the robot at the location l_j
- N: total number of experiments at the locations l_j

5. repeat the process for each location

Events and Probabilities

- Event ≜ signal strength S falls in the interval s_i (in the case of this example)
  - events characterize a SUBSET OF THE POSSIBLE OUTCOMES
  - events are either true or false!
- Event $S_i = S = s_i$ (to compact the notation) ⟶ $P_{l_j}(S_i)$ = probability of the event of observing a signal strength S_i given a location l_j
- Probability ≜ function going from an each measurable subset of the EVENT SPACE Ω to the interval $[\emptyset, 1]$

Axioms: probability does not need to fit into a certain shape ⟶ only need to obey certain axioms

1. $P(E) \geq \emptyset$: probability of an event is greater or equal than $\emptyset$
   - an event cannot have a negative probability
   - event is either true or false (occurred or not!)
2. $P(\bigcup_i E_i) = \sum_i P(E_i)$: probability of the union of a set of disjoint events is the sum of probabilities of the events
   - disjoint event ≜ mutually exclusive (if one occurs, the other cannot occur)
   - example: measuring signal strength s_1 is mutually exclusive from s_2
3. $P(\Omega) = 1$: probability of the outcome being in the set of possible outcomes is 1

Continuous Domains → probability of the outcome assuming exactly a specific value X is, in general, $\emptyset$ (in a continuous domain)

- e.g., signal strength in a continuous value
- recovering from a statistic would require infinitesimal small values ....

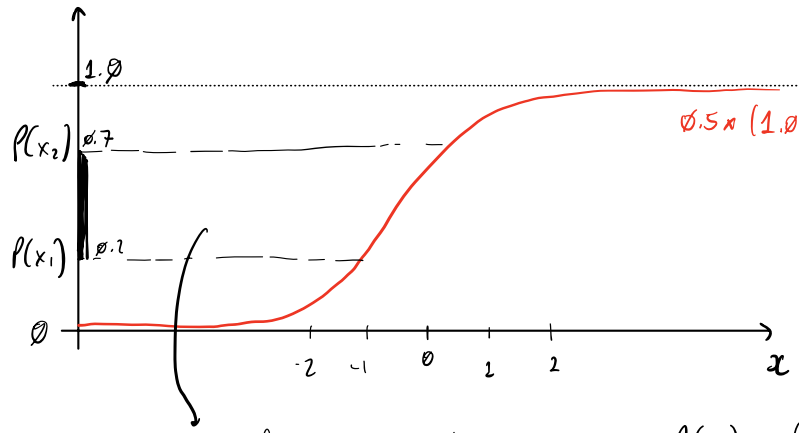
(and most of the bins with $\emptyset$ probability...)

So, how do we define an event in a continuous domain?

$P(X < x)$, where the event $X < x$ is TRUE when the outcome of the event X becomes lower than x

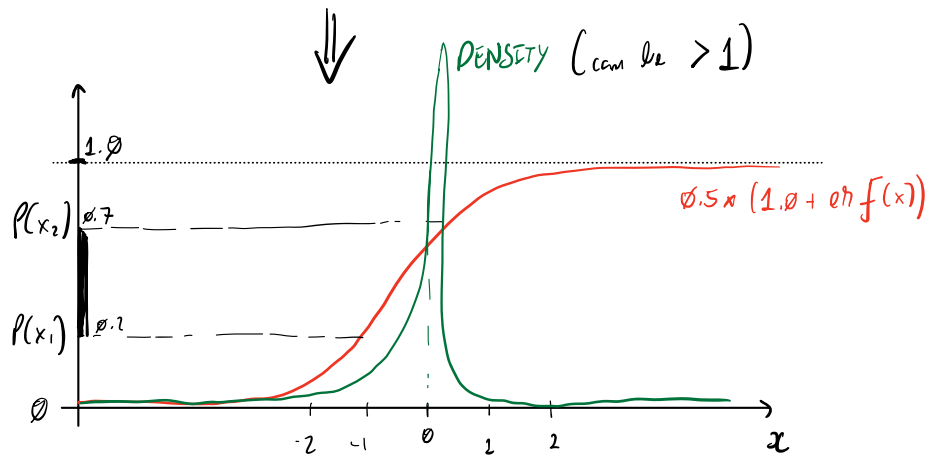
$\equiv$

$P(x)$ ≜ CUMULATIVE DENSITY FUNCTION → monotonically increasing



$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{\emptyset}^{x} e^{-t^2} dt$$

the probability an element falls in this interval is $P(x_2) - P(x_1)$ → as $x_2 \to x_1 \Rightarrow P(x_2) - P(x_1) \to \emptyset$



Probability Density $p(x)$

$\triangleq$

$$p(x) = \frac{\partial P(x)}{\partial x}$$

(nothing more than the derivative)

To compute the probability of a generic event $E \subseteq \Omega$ is by integration: $P(E) = \int_{x \in E} p(x)\, dx$

Ω ✓ $\equiv$

summing up infinitesimal disjoint events that cover E !

Back to our Scenario

→ let's say, similar to the signal strength experiment, we acquire statistics on how often the robot visits each location

↓

but, completely ignoring signal stats

→ results of the experiments : $P(L_i)$

l_1	l_2	l_3	....					....	l_{10}
l_{11}	l_{12}	l_{13}	....					....	l_{20}
...			⋱						⋮
⋮									
l_{91}	l_{92}	...							l_{100}

Conditional Probability

- We acquire independent statistics on signal strength at each location ≡ separate distribution over signal strength given the location

$$P_{l_j}(S = s_i) := P(S_i \mid L_j)$$

$$\triangleq$$

distribution over the signal strength, **GIVEN** the knowledge of the location !

- L_j is the part of the event that controls the shape of the probability function
- note that $P(S_i \mid L_j)$ **REMAINS** the probability distribution <u>OVER</u> the signal strength, <u>NOT</u> on location !

Chain Rule → how likely is that Oscar visits location L_2 and from there measures a signal strength S_3 ?

$$P(S_3, L_2)$$

- A <u>CERTAINTY</u> is that $P(S_3, L_2) \leq P(L_2)$
  - $P(L_2)$ = probability of visiting L_2 and measuring <u>ANY</u> signal strength
- it is as much likely of occurring the event $\{S_3, L_2\}$ as being at L_2 <u>AND</u> from L_2 measuring S_3

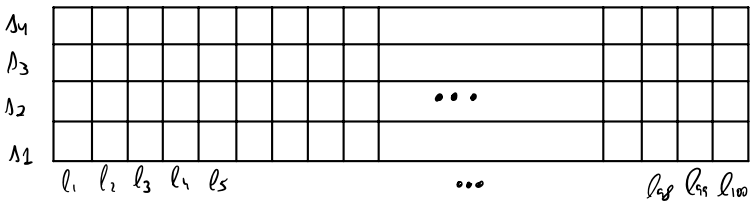
$$P(S_3, L_2) = P(L_2) \cdot P(S_3 \mid L_2)$$

- note that the probability of joint independent events is the product of occurring each independent event ($P(A, B) = P(A)\,P(B)$, when A and B are independent)
- experiments of obtaining $P(S_3 \mid L_2)$ were completely independent from estimating $P(L_2)$

Joint Distributions

$P(S_3, L_2)$ ≡ joint distribution

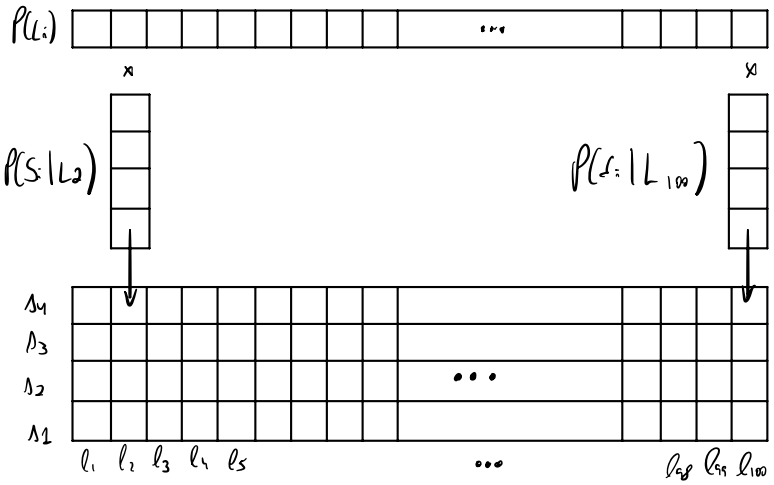
whose domain = all possible pairs of signal strengths and locations



GRID

(w/ all possible combinations)

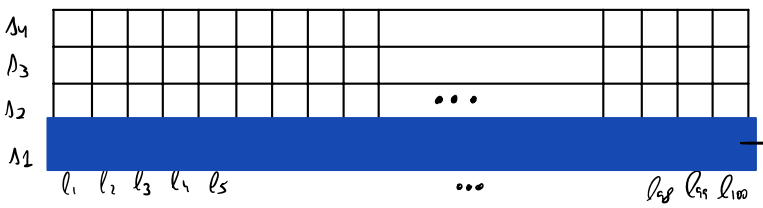
- limits the search space where we want to know the probability of measuring a certain signal strength and being at a location
- Can respond any specific question in the joint distribution domain



≡ Compute the joint distribution using the chain rule

↳ multiply conditional $\{S_1|L_j\}$ by the conditional $\{L_j\}$

Marginals → what is the probability of seeing a very weak signal strength s_1?

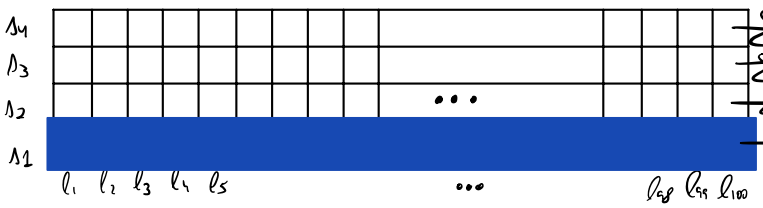


$P(S_1) = \sum_j P(S_1, L_j)$ ≡ probability of measuring s_1 from any locations (= row in the table)

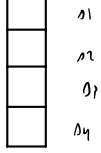
joints events $< S_1, L_j >$ are disjoint ! (see 2nd axiom)

We basically collapsed 1 variable in space / deleting it by summing up all possible events that include that variable !

Marginalization ≡ process of summing a variable from the joint distribution



$P(S)$



or

method that requires running over the possible values of one variable to determine the marginal contribution of another !

Independence

$P(A|B) = P(A)$ ⟶
- whatever the value of B, A remains
- Knowing B tells nothing about A ⟶ unconditional events!
- example: independent phenomena would be the floor at which the elevator is located and the signal strength (i.e., the position of the elevator does not affect the signal strength locally)

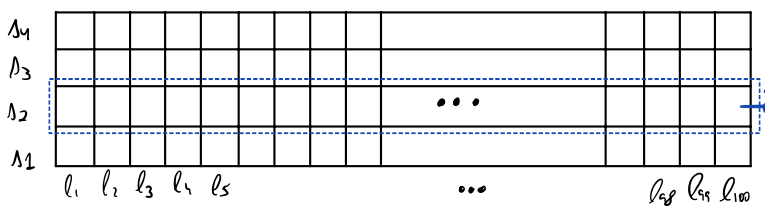
$\downarrow$

$P(A,B) = P(A|B)P(B) = P(A)P(B)$

$\Downarrow$

INDEPENDENCE TEST:
1. Compute the marginal over the first variable
2. Compute the marginal over the second variable
3. Compute the joint distribution from the marginals as if the variables were independent
   - if the resulting table is the same as the original one, variables are independent
   - otherwise, are not

Localizing Orazio



⟶ suppose we have the joint distribution over all locations and strengths

$\Downarrow$

⟶ then, we measure the signal strength s_2

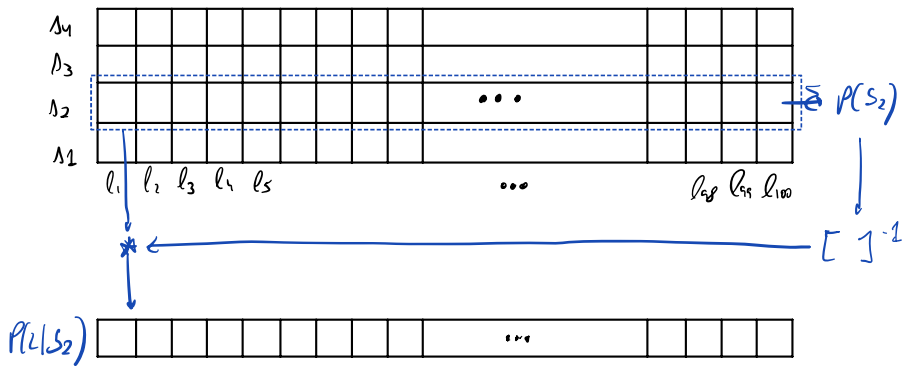
So, we want to localize Orazio, GIVEN the measurement s_2: $P(L|s_2)$

- given that we measure s_2, all other paths/signals become irrelevant
- HOWEVER, the row of s_2 is not a valid distribution (does not sum up to 1, but sums up to the marginal of s_2) — AXIOM 3
- EVEN SO, row of s_2 comprises all the knowledge we can conclude given the evidence

$\Downarrow$

$$P(l_j|s_2) = \frac{P(s_2, l_j)}{P(s_2)}$$

we divide all elements in row 2 by the (CONSTANT) value $P(s_2)$ to render the row s_2 as a valid probability distribution



Functions

X: random variable $\sim P(X)$

Let $y = f(x)$, where f is a generic function over all possible values of X

⇒ y is also a random variable

$$P(y) = \sum_{x_i : f^{-1}(y)} P(x_i)$$

Conditional Independence

- Now, suppose we have 2 access points at different locations (suppose they transmit at different frequencies NOT INTERFERING WITH EACH OTHER)
- Also, assume we can compute statistics on the strength for 1st and 2nd signals
- Let's assume the only element influencing the signal strength is the location (the signals do not interfere!)

$$\begin{bmatrix} P(S^A|L) \\ P(S^B|L) \end{bmatrix}$$

⇓

What about $P(S^A, S^B)$? Are these two variables independent? ⟹

- strength depends on the location ⇓ strength and location ARE CORRELATED!
- if S^A influences L, and S^B is influenced by L ⇓ S^A, S^B are correlated through the location

(note that $P(S^A, S^B)$ does not have the location)

≡

"if I know S^A, can I tell anything about S^B?"

But what if I know the location?

$$P(S_A, S_B|L) = P(S_A|L)\,P(S_B|L)$$ → the two signals are independent if I know the location

⇓

exploiting the structure of the problem (physical properties, work relations, etc...), we can figure it out if two variables are correlated or not

⇓

Summary:

- Marginalization: collapsing the variables from the joint probability
- Conditioning: slicing the joint distribution and selecting all the pts of the joint probability that are consistent from the evidence that I got
- Chain rule: probability of something happens is equal to that one of the things happened multiplied by the second thing probability given the first one

(NOT ASSUMING ANY PARTICULAR PROBABILITY DISTRIBUTION SHAPE)

Probabilities Identities

$$P(A,B) = P(A|B)\,P(B)$$

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

$$P(A) = \sum_i P(A, B_i)$$

$$P(A,B|C) = P(A|B,C)\,P(B|C)$$

$$\Rightarrow P(A|B,C) = \frac{P(A,B|C)}{P(B|C)}$$

$$P(A|C) = \sum_i P(A, B|C)$$

we can add the same conditioning term to *all* terms of the equation and it still holds.

Summary

- Axioms:
  1. $P(E) \geq 0$
  2. $P(\bigcup_i E_i) = \sum_i P(E_i)$
  3. $P(\Omega) = 1$
- Chain rule: $P(A,B) = P(A|B)\,P(B)$
- Marginalization: $P(A) = \sum_i P(A, B_i)$
- Conditioning: $P(A|B) = \frac{P(A,B)}{P(B)}$

BAYESIAN NETWORKS

DIMENSIONALITY

4 possible level of strength for each AP
2 APs
6 floors
100 locations

$\#\text{disjoint events} = 4 \times 4 \times 6 \times 100 = 9600$ possible configurations

$\#\text{disjoint events} = \#\text{signal strength}^{\#\text{APs}} = 4^2$

$\#S_i^A \times \#S_i^B \times \#\text{floors} \times \#L_j$

What if we "gift" Dorito with a camera with "poor VGA" (10x10) and 16 possible levels of grayscale?

$\#\text{disjoint events}: 4 \times 4 \times 6 \times 100 \times 16^{10\times 10} = 9600 \times 16^{100} = 24790 \times 10^{124}$ → well, ... that's a lot of configurations

⇓

Conditional independence helps reduce this!

Query: $P(S^A \mid S_2^B)$

Joint: $P(S^A, S_2^B, \cancel{I^{1,1}, \ldots, I^{10,10}}, L, E)$ ← elevation

eliminate all variables that are not relevant through MARGINALIZATION (lots of sums and collapsing the joint distribution "table"/space)

BAYES RULE (to obtain the 1D answer, we need to compute the 2D table that contains both S^A and S_2^B, THEN, we can do conditional probability)

But, if we know that signal strength is only affected by location, we can ignore a lot of variables

- signal strength only depends on the location
- pixels in the image depend only on the location (if nothing but Dorito moves ☺ if light conditions do not change)
- elevator light lives in his own world (as long as Dorito does not take it)

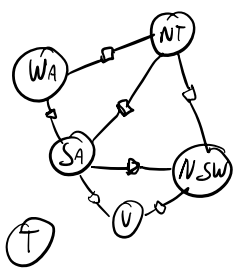
BAYESIAN NETWORKS

Probabilistic graphic models = graphs ≡ compact representations of joint probability distributions !

- NODES (=) random variables
- (LACK OF) ARCS (=) conditional independence assumptions

UNDIRECTED GRAPHIC MODELS ≡ MARKOV RANDOM FIELDS (MRF) ≡ MARKOV NETWORKS

- independence ≜ A and B conditionally independent given a third set C IF all paths between A and B are separated by C



WA, NT, SA, NSW, V = A

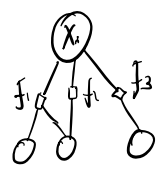
T = B

⇓

A ⫫ B (independent)

⇓

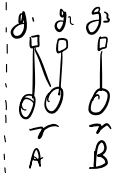
no edges between A and B



Conditioning $X_i = v$

Remove X_i and all factors that depend on X_i (f_1, f_2, f_3)

add $g_j = f_j(x \in U \mid X_i = v)$



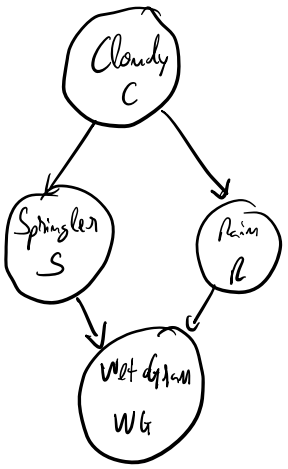
A and B are conditionally independent given C ($L = x_i$)

Directed Graphical Models ≡ Bayesian Networks

- more complicated notion of conditional independence
- gives indication of cause → effect (helpful guide to construct the graph)
- encode deterministic relations ⇒ easier to fit the data
- represents a set of variables and their conditional dependencies via a directed acyclical graph (DAG) with no loss of information!

(most simplest case)

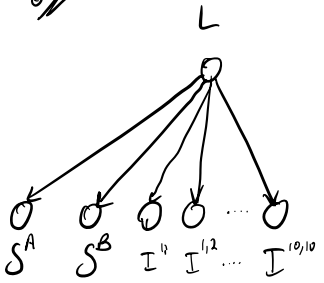
→ Node independent of its Ancestors given its Parents, IF ancestor – parent relation is with respect to some fixed topological ordering of the nodes



$$P(C,S,R,WG) = P(C)\,P(S|C)\,P(R|S,C)\,P(WG|R,S,C)$$

| conditional independence

$$= P(C)\cdot P(S|C)\cdot P(R|C)\cdot P(WG|S,R)$$



→ $L \rightarrow S^A$ (cause) (effect)

→ all we need to know to render a picture in the location of the robot

→ etc.

(rational thinking about the Oregon scenario)

pixels independent given the location

$$\#L + \#S^A|L + \#S^B|L + \#I^{ij}|L + \#p_x + \#E$$
$$= 100 + (4\times 100) + (4\times 100) + (16\times 100\times 100) + 6 = 160906$$

- much more less variables for the joint distribution
- captures the SAME information

Note: each node stores a conditional probability (NOT THE EDGE!)

Inference on Bayesian Networks

$P(S^A|I_7^{1,1}) = ?$ →

1. Compute the joint probability

$$P(S^A, S^B, I_7^{1,1}, I_1^{1,2},\dots,I^{10,10}, L, E) = P(S^A|L)\,P(S^B|L)\,P(I_7^{1,1}|L)\dots P(I^{10,10}|L)\,P(L)\,P(E)$$

2. marginalize the variable we do not need

$$P(S^A, I_7^{1,1}) = \sum_{S^B}\sum_{I^{1,2}}\dots\sum_{I^{10,10}}\sum_L\sum_E P(S^A, S^B, I_7^{1,1}, I^{1,2},\dots,I^{10,10}, L, E)$$

⇓ collapse the joint distribution dimension space to only 2 dimensions

3. use conditioning to get the answer: $P(S^A|I_7^{1,1}) = \dfrac{P(S^A, I_7^{1,1})}{\sum_{S_A} P(S_A, I_7^{1,1})} = P(I_7^{1,1})$!

→ would we get the same answer if elevation were not in the domain? YES

→ Knowing intensity of another pixel would provide more information? YES

- $P(I^{1,1}, I^{1,2}) \leq P(I^{1,1})$ → it would add more information to where the robot would be (↓ variables marginalized)

thus, S^A (L causes S^A)

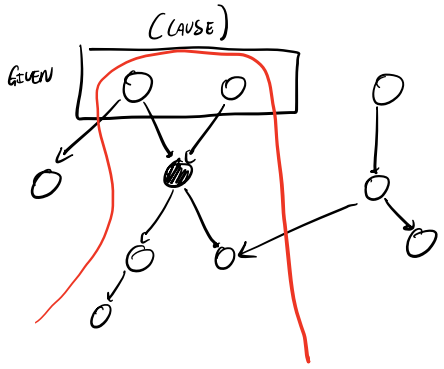
→ Does the same hold if we know the location? no

- Knowing the location makes the pixel intensity conditionally independent

⇓

Knowing $I^{1,2}$ given L does not bring more information than $I^{1,1}$ given L...

SEMANTICS

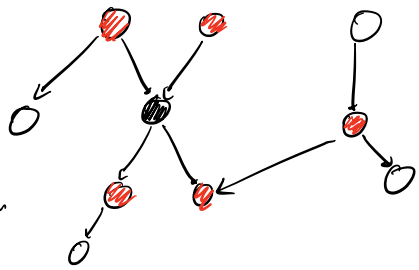


LOCALLY

each node conditionally independent from its non-descendant given its parents

GLOBALLY

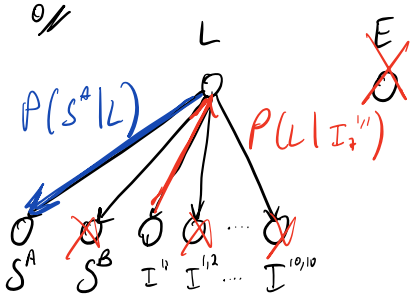
each node independent from the rest given
- its parents
- its children
- the parents of its children



e.g., elevation E is disconnect ⇒ INDEPENDENT

if we would know the location, it would be easy!

$L \; P(L \mid I_7^{1,1})$



↳ all other variables would be redundant!

$P(L, I^{1,1}) = P(I^{1,1} \mid L) P(L)$

1. claim rule on $P(I^{1,1} \mid L)$ and $P(L)$ to get $P(L, I)$
2. Conditioning on $P(L, I)$ to get $P(L \mid I)$

$P(L \mid I^{1,1}) = \frac{P(L, I^{1,1})}{P(I^{1,1})}$

$P(L \mid I^{1,1}) = \frac{P(L, I^{1,1})}{\sum_{L_i} P(L_i, I^{1,1})}$

⇓

$P(S^A \mid I^{1,1}) = P(S^A \mid L) P(L \mid I^{1,1})$

3. Claim rule on $P(S^A \mid L)$ and $P(L \mid I)$

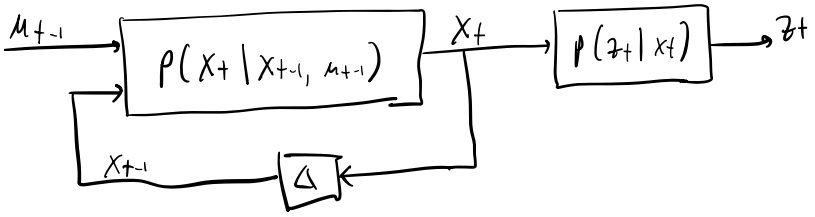
determine signal strength from improved location estimate $L \mid I_7^{1,1}$

Dynamic Bayesian Networks

(Temporal model) → generalizes Hidden Markov Models (HMM) and Linear Dynamical Systems (LDS)

→ by representing hidden (and observed) states in terms of state variables

Probabilistic Dynamical Systems



$p(x_t \mid x_{t-1}, u_{t-1})$: transition model

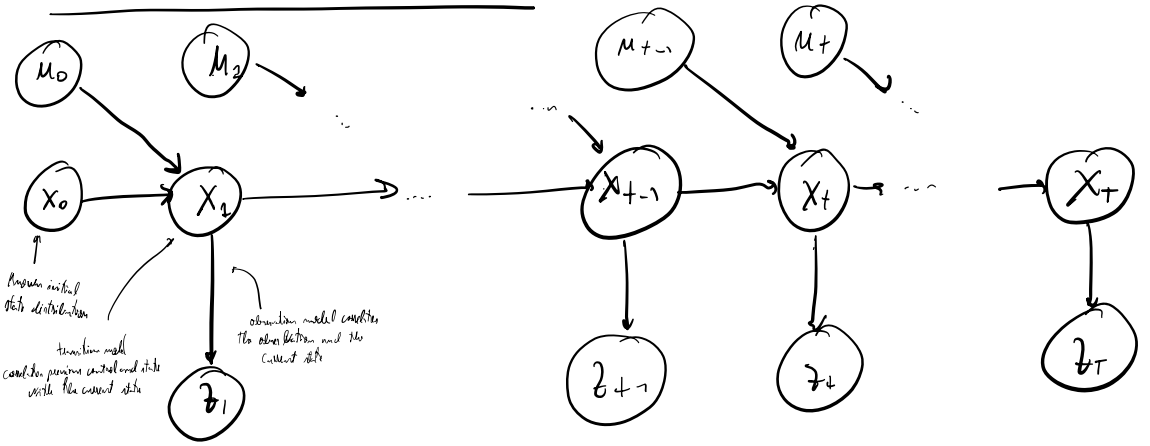
$p(z_t \mid x_t)$: observation model

x_{t-1} : previous state — z_t : current observation

x_t : current state — Δt : time delay

u_{t-1} : previous control / action

Dynamic Bayesian Networks (DBN)



→ Recurrent structure depending on time t

$x_{0:T} \equiv$ robot's trajectory

or

cities / things

(not restricted to, cool / obvious)

- States in a DBN
  - Robot localization with laser range finder
    - States $x_t \in SE(2)$
    - Observations $z_t \in \mathbb{R}^{\#\text{beams}}$
    - controls $u_t \in \mathbb{R}^2$ (translational and rotational speed)
  - HMM
    - States $x_t \in [x_1, \dots, x_{N_x}]$ finite states
    - Observations $z_t \in [z_1, \dots, z_{N_z}]$ finite observations
    - Controls $u_t \in [v_2, \dots, v_{N_u}]$ finite controls

using traditional tools of Bayes not not a good idea

But, recurrent structure may be exploited !

Inferences on a DBN

	Query	Known	
Filtering	$p(x_T \mid u_{0:T-1}, z_{1:T})$	$u_{0:T-1}$, $z_{1:T}$	→ Know the distribution of the current state knowing everything else (only most recent state)
Smoothing	$p(x_t \mid u_{0:T-1}, z_{1:T})$, $0 < t < T$	$u_{0:T-1}$, $z_{1:T}$	→ estimate a location in the past "generally", provides better estimate than filtering (because we have more info)
Max a Posteriori	$\underset{x_{0:T}}{\text{argmax}}\, p(x_{0:T} \mid u_{0:T-1}, z_{1:T})$	$u_0 : T-1$, $z_{1:T}$	→ given all the knowledge, compute most likely trajectory over the state

- **Belief:** $b(x_t) \triangleq p(x_t \mid \dots)$ = our current belief on the current state
  - algorithms inferencing on DBN usually work by updating a belief
  - discrete state $x \in \{X_{1:n}\}$ ⟶ array of n floats = belief
    (where each cell contains the probability of that state)
  - continuous state + distributed according to a Gaussian ⟶ belief = mean + covariance matrix
  - continuous state + unknown distribution ⟶ approximate representation (e.g., weighted samples of state values)

⇒ **FILTERING:** $p(x_T \mid u_{0:T-1}, z_{1:T})$

Known:
- $z_{1:T}$
- $u_{0:T-1}$
- $p(x_t \mid x_{t-1}, u_{t-1})$ = transitional model
- $p(z_t \mid x_t)$ = observation model
- $b(x_{t-1}) = p(x_{t-1} \mid u_{0:t-2}, z_{1:t-1})$ (current belief about previous state)

both functions

$$p(x_T \mid u_{0:T-1}, z_{1:T}) =$$

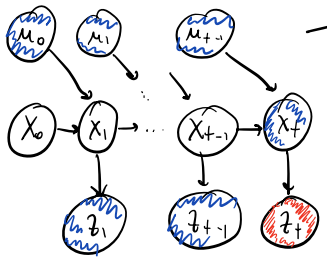
$$= p(x_t \mid z_t, u_{0:t-1}, z_{1:t-1})$$
(A = x_t, B = z_t, C = $u_{0:t-1}, z_{1:t-1}$)

$$p(A \mid B, C) = \frac{p(B \mid A, C)\, p(A \mid C)}{p(B \mid C)}$$

$$= \frac{p(z_t \mid x_t, u_{0:t-1}, z_{1:t-1}) \cdot p(x_t \mid u_{0:t-1}, z_{1:t-1})}{p(z_t \mid u_{0:t-1}, z_{1:t-1})}$$

let the denominator $\eta_t = \frac{1}{p(z_t \mid u_{0:t-1}, z_{1:t-1})}$
- does not depend on the state x
- JUST A NORMALIZER FACTOR !

$$= \eta_t \cdot p(z_t \mid x_t, u_{0:t-1}, z_{1:t-1}) \cdot p(x_t \mid u_{0:t-1}, z_{1:t-1}) =$$



⇒ if we know x_t, we do not need to know $u_{0:t-1}$, $z_{1:t-1}$

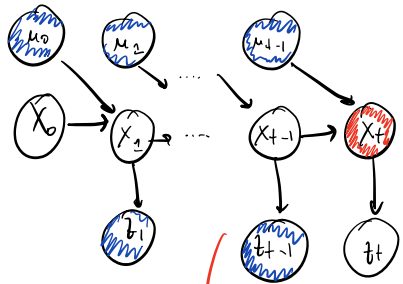
⇕

x_t encodes all the knowledge about the past (MARKOV ASSUMPTION)

$$= \eta_t \cdot p(z_t \mid x_t) \cdot p(x_t \mid u_{0:t-1}, z_{1:t-1})$$

OBSERVATION MODEL

? still no meaning... ?



if we would know x_{t-1}, through the Markov assumption, it would make our life a lot easier.....

⟶ $p(x_t \mid x_{t-1}, u_{0:t-1}, z_{1:t-1}) = p(x_t \mid x_{t-1}, u_{t-1})$

$\Longleftarrow$ marginalization: $p(A|C) = \sum_B p(A,B|C)$; chain rule: $p(A,B|C) = p(A|B,C)\, p(B|C)$ $\Rightarrow p(A|C) = \sum_B p(A|B,C)\, p(B|C)$

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

↱ the last control u_{t-1} has no influence on x_{t-1} if we do not know x_t !

$$= \eta_t \cdot \underbrace{p(z_t|x_t)}_{\text{OBSERVATION MODEL}} \cdot \sum_{x_{t-1}} \underbrace{p(x_t|x_{t-1}, u_{t-1})}_{\text{TRANSITION MODEL}} \cdot \underbrace{p(x_{t-1}|u_{0:t-2}, z_{1:t-1})}_{b(x_{t-1})}$$

$$= \eta_t \cdot p(z_t|x_t) \cdot \sum_{x_{t-1}} p(x_t|x_{t-1}, u_{t-1}) \cdot b(x_{t-1})$$

↑ is only a constant to ensure that $b(x_t)$ remains a probability distribution

$$\eta_t = \frac{1}{p(z_t|u_{0:t-1}, z_{1:t-1})} = \frac{1}{\sum_{x_t} \left(p(z_t|x_t) \cdot \left(\sum_{x_{t-1}} p(x_t|x_{t-1}, u_{t-1}) \cdot b(x_{t-1}) \right) \right)}$$

this must be inside the sum, otherwise would not make any sense ....

⊕ checks out with the Octave code!

MARGINALIZATION ⊕ CHAIN RULE

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

(A = z_t, C = $u_{0:t-1}, z_{1:t-1}$)

① $B = \{x_t, x_{t-1}\}$ $\quad p(A|C) = \sum_B p(A|B,C)\, p(B|C)$

MARKOV ASSUMPTION

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

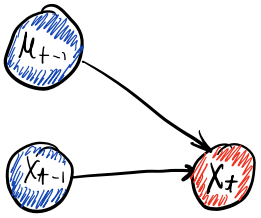
$p(z_t|x_t)$ ↓ does not depend on x_{t-1}

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

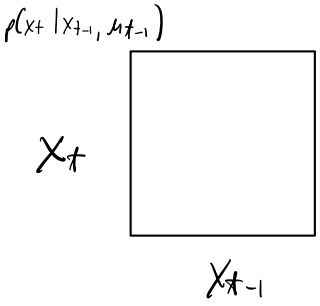
$$= \sum_{x_t} p(z_t|x_t) \left[\sum_{x_{t-1}} p(x_t|x_{t-1}, u_{t-1}) \cdot b(x_{t-1}) \right]$$

qed

FILTERING: ALTERNATIVE FORMULATION → Belief in a different view from the classical formulation



- from previous Bayesian network, we know u_{t-1} and x_{t-1}
- we want to shift all the previous knowledge to x_t
- marginalizing ≈ collapsing the variables space
- conditioning ≈ slicing



- **PREDICT**: incorporate in the last belief $b_{t-1|t-1}$ the most recent control u_{t-1}

ingredients
- transition model — $p(x_t | x_{t-1}, u_{t-1})$
- prior belief — $b_{t-1|t-1}$
- control — u_{t-1}

$$p(x_t, x_{t-1} | t-1) = \underbrace{p(x_t | x_{t-1}, u_{t-1})}_{\text{MOTION MODEL}} \cdot \underbrace{p(x_{t-1} | t-1)}_{b_{t-1|t-1}}$$

↓ remove x_{t-1} through MARGINALIZATION

$$p(x_t | t-1) = \sum_{x_{t-1}} p(x_t, x_{t-1} | t-1)$$

$b_{t|t-1}$

```
BeliefType b_pred = BeliefType::Zero;
for (x_i : X)
  for (x_j: X)
     b_pred[x_j] += b[x_i]*transitionModel(x_j,x_i,u);
```

- **UPDATE**: incorporate in the predicted belief $b_{t|t-1}$ the new measurement z_t

$$p(x_t, z_t | t) = p(z_t | x_t) \cdot p(x_t | t-1)$$

$b_{t|t-1}$

$$p(x_t | t) = \frac{p(x_t, z_t | t)}{p(z_t | t)}$$

$b_{t|t}$

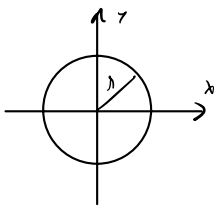
$$p(z_t | t) = \sum_{x_t} p(x_t, z_t | t) \quad \text{≈ normalizer!}$$

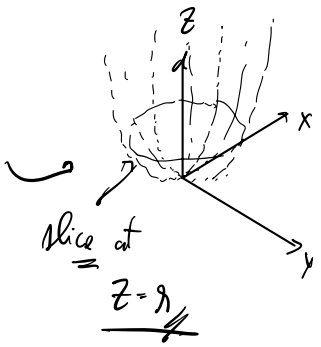
```
float normalizer=0;
for (x_i : X) {
   b[x_i] = b_pred[x_i] * observationModel(z,x_i);
   normalizer += b[x_i];
}
b *= 1./normalizer;
```

Gaussian Distribution

Elementary Exercises

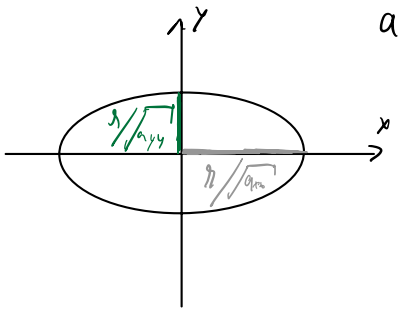
$x^2 + y^2 = R^2$ ≡ slice of paraboloid





slice at $z = R$

Circle w/ scaling factors:



$a_{xx} x^2 + a_{yy} y^2 = R^2$

↓ a_{xx}, a_{yy}

note that these values affect the shape of the paraboloid

⇓

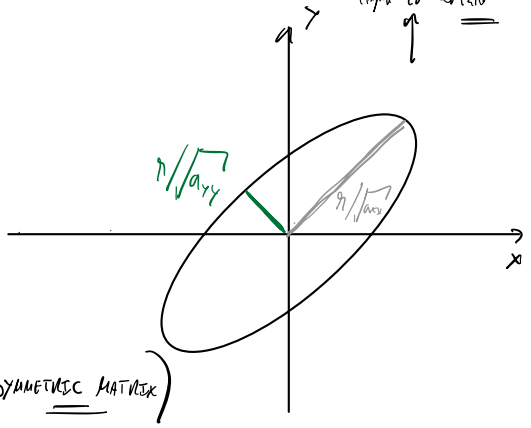
- **Slanted Circles:** $a_{xx} x^2 + a_{xy} xy + a_{yy} y^2 = R^2 \Leftrightarrow$

$$\Leftrightarrow \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a_{xx} & a_{xy}/2 \\ a_{xy}/2 & a_{yy} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = R^2$$

cross-correlation coefficient

(the reason why ellipse is rotated)

a_{xy} causes the ellipse to rotate



A ⟶ admits an eigen-value decomposition (SYMMETRIC MATRIX)

$$A = R^T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

EIGEN VALUES

EIGEN VECTORS ≡ orthogonal matrices that form a rotation matrix

In addition to rotation, ellipses may also be translated by a vector $[x_c \; y_c]^T$:

$$a_{xx} (x-x_c)^2 + a_{xy} (x-x_c)(y-y_c) + a_{yy} (y-y_c)^2 = R^2 \Leftrightarrow$$

$$\Leftrightarrow \begin{bmatrix} x-x_c & y-y_c \end{bmatrix} \begin{bmatrix} a_{xx} & a_{xy}/2 \\ a_{xy}/2 & a_{yy} \end{bmatrix} \begin{bmatrix} x-x_c \\ y-y_c \end{bmatrix} = R^2 \Leftrightarrow$$

$$\Leftrightarrow \left[R \cdot \begin{bmatrix} x-x_c \\ y-y_c \end{bmatrix} \right]^T \cdot \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \cdot \left[R \cdot \begin{bmatrix} x-x_c \\ y-y_c \end{bmatrix} \right] = R^2$$

⟶ general equation of an ellipse (w/ rotation R and translation $\begin{pmatrix} x_c \\ y_c \end{pmatrix}$)

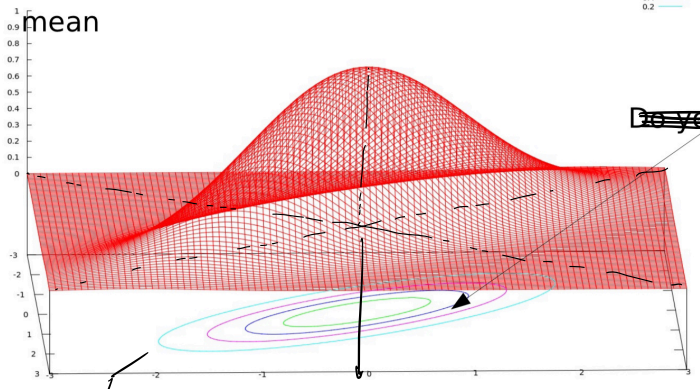
$\Vert\Vert$

$$\begin{bmatrix} x-x_c & y-y_c \end{bmatrix} R^T$$

Gaussian and Parameterizations

Probability density function (PDF) of a Gaussian distribution: $p(x) = \mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$

μ = MEAN, Σ = COVARIANCE



2D ellipses ≡ slices of the Gaussian PDF (w/ $n = 2$)

$\mu \equiv$ point of higher value in the PDF of the Gaussian dist.

Cov-correlation factors in Σ make the sliced ellipses to be rotated

→ NORMALIZER!

to ensure that the integral of the 2nd part of the PDF function sums up to 1

$p(x) = 0.3 \Leftrightarrow$

$\Leftrightarrow 0.3 = \frac{1}{(2\pi)^{n/2}\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right) \Leftrightarrow$

$\Leftrightarrow \ln(0.3) = \ln\left(\frac{1}{(2\pi)^{n/2}\sqrt{\det(\Sigma)}}\right) - \frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)$

↑ ellipse geometric form

→ example:

- **Why using Gaussians?** these distributions are closed under
  - sums
  - affine transformations $(Ax+b)$
  - conditioning
  - marginalization

  → the end-result still is a Gaussian described by μ and Σ (only 2 parameters!)

  ≡ have closed-form equations to describe the distribution

- **Moment Parameterization** → parameters μ and Σ can be computed from a large set of samples

$$\mu = \frac{1}{N}\sum x^{(i)}$$

$$\Sigma = \frac{1}{N-1}\sum (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

↑ this estimator is called **sample variance** (s^2)

$s^2 = \frac{N}{N-1}\hat{\sigma}^2$, where it differs from $\hat{\sigma}^2$ by having $(N-1)$ in the denominator

≡ BESSEL'S CORRECTION

(to correct the bias in the estimation of the population variance)

↳ has more impact with lower number of samples

(for ↑N ↘ $\frac{N}{N-1} \to 1$)

↑ EQUIVALENT

Discrete case, where we previously were using an histogram!

EXPECTED VALUE

$$\mu = \int_{\mathbb{R}} x\, p(x)\, dx = \mathbb{E}[x]$$

≡ 1-st ORDER MOMENT

$\mathbb{R}$ = EVENT SPACE

$$\Sigma = \int_{\mathbb{R}} (x-\mu)(x-\mu)^T p(x)\, dx = \mathbb{E}\left[(x-\mu)(x-\mu)^T\right]$$

≡ 2nd ORDER MOMENT

⇓

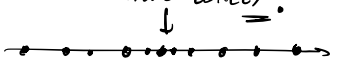
these parameters obtained experimentally with large set of samples

↳ but assuming that we know the process is governed by a Gaussian distribution

IF $\Sigma = \emptyset$ ⇒ each sample is exactly the same as the mean!

$p(x) \equiv$ PDF function

in the discrete case, a sample is highly likely if it occurs more often...



- **Canonical Parameterization** → useful for conditioning

$\Omega = \Sigma^{-1}$: INFORMATION MATRIX

$\nu = \Omega \mu$: INFORMATION VECTOR

$$\longrightarrow \mathcal{N}^{-1}(x; \nu, \Omega) = \frac{\exp\left(-\frac{1}{2}\nu^T \Omega^{-1} \nu\right)\sqrt{\det(\Omega)}}{(2\pi)^{n/2}} \cdot \exp\left(-\frac{1}{2} x^T \Omega x + x^T \nu\right)$$

How to prove that $\mathcal{N}^{-1}$ equation is correct?

$$p(x) = \mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

MEAN ↑ ↑ COVARIANCE

$\Omega = \Sigma^{-1}$, $\nu = \Omega\mu \Leftrightarrow \Omega^{-1}\nu = \mu$, $\det(\Omega) = 1/\det(\Sigma)$

$$= \frac{1}{(2\pi)^{n/2}\sqrt{1/\det(\Omega)}} \cdot \exp\left(-\frac{1}{2}(x - \Omega^{-1}\nu)^T \Omega (x - \Omega^{-1}\nu)\right) = \frac{1}{(2\pi)^{n/2}\sqrt{1/\det(\Omega)}} \cdot \exp\left(-\frac{1}{2}\left(x^T - (\Omega^{-1}\nu)^T\right)(\Omega x - \Omega\Omega^{-1}\nu)\right) =$$

$(\Omega^{-1}\nu)^T = \nu^T(\Omega^{-1})^T = \nu^T\Omega^{-1}$ (SYMMETRIC MATRIX), $(\Omega^{-1})^T = \Omega^{-1}$, $\frac{1}{\sqrt{1/\det(\Omega)}} = \sqrt{\det(\Omega)}$

$$= \frac{1}{(2\pi)^{n/2}\sqrt{1/\det(\Omega)}} \cdot \exp\left(-\frac{1}{2}\left(x^T - \nu^T\Omega^{-1}\right)(\Omega x - \nu)\right) =$$

$$= \frac{\sqrt{\det(\Omega)}}{(2\pi)^{n/2}} \cdot \exp\left(-\frac{1}{2}\left(x^T\Omega x - x^T\nu - \nu^T\Omega^{-1}\Omega x + \nu^T\Omega^{-1}\nu\right)\right) = \frac{\sqrt{\det(\Omega)}}{(2\pi)^{n/2}} \cdot \exp\left(-\frac{1}{2}\left(x^T\Omega x - 2x^T\nu + \nu^T\Omega^{-1}\nu\right)\right) = \frac{\exp\left(-\frac{1}{2}\nu^T\Omega^{-1}\nu\right)\sqrt{\det(\Omega)}}{(2\pi)^{n/2}} \cdot \exp\left(-\frac{1}{2}x^T\Omega x + x^T\nu\right)$$

INNER PRODUCT $= \nu^T x = x^T\nu$; $\nu^T\Omega^{-1}\Omega x = \nu^T x$ ↑

cqd

- **Partitioned Gaussian Densities** → the space may be splitted into 2 subspaces density is over a JOINT DISTRIBUTION:

$$X = \begin{pmatrix} x_a \\ x_b \end{pmatrix} \longrightarrow \mu = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} \qquad \nu = \begin{pmatrix} \nu_a \\ \nu_b \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} \qquad \Omega = \begin{pmatrix} \Omega_{aa} & \Omega_{ab} \\ \Omega_{ba} & \Omega_{bb} \end{pmatrix}$$

Note that for Σ to remain symmetric, $\Sigma_{ba} = \Sigma_{ab}^T$ ✓

⇒ **AFFINE TRANSFORMATION**

$x_a \sim \mathcal{N}(x_a; \mu_a, \Sigma_a)$

$x_b = f(x_a) = A x_a + c$ ≡ affine transformation of x_a

↓

x_b is still a Gaussian!

$$\Rightarrow \mu_b = A\mu_a + c$$

$$\Sigma_b = A \Sigma_a A^T$$

$\mu_b = \mathbb{E}[x_b] = \mathbb{E}[A x_a + c] = A\,\mathbb{E}(x_a) + \mathbb{E}(c) = A\mu_a + c$ cqd

$\Sigma_b = \mathbb{E}\left[(x_b - \mu_b)(x_b - \mu_b)^T\right] = \mathbb{E}\left[(A x_a + c - A\mu_a - c)(A x_a + c - A\mu_a - c)^T\right] =$ ($x_b = A x_a + c$, $\mu_b = A\mu_a + c$)

$= \mathbb{E}\left[(A(x_a - \mu_a))(A(x_a - \mu_a))^T\right] = \mathbb{E}\left[A(x_a - \mu_a)(x_a - \mu_a)^T A^T\right] = A\Sigma_a A^T$ cqd

What about non-linear transformations? APPROXIMATE THE FUNCTION AROUND A LINEARIZATION POINT x_0 (≡ Taylor expansion)

$$f(x) \simeq f(x_0) + \underbrace{\frac{\partial f(x)}{\partial x}}_{A}(x - x_0) =$$

$$= A x + \underbrace{f(x_0) - A x_0}_{b} \equiv 1^{st}\text{-order Taylor expansion}$$

- the farther f is from being linear, the worse the approximation!
- approximation holds ONLY around the linearization point
- reduces a non-linear transformation to an affine one

MARGINALIZATION → "kills a dimension" (collapses the variables)

$$X = \begin{pmatrix} X_a \\ X_b \end{pmatrix} \sim \mathcal{N}(x; \mu, \Sigma), \text{ where } \Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}$$

$$p(X_a) = \int_{X_b} p(X_a, X_b)\, dX_b \sim \mathcal{N}(X_a; \mu_a, \Sigma_{aa})$$

$p(X_a, X_b)$: collapses X_b from the joint distribution $X = (X_a, X_b)$

⇒ MARGINALIZATION DOES NOT CHANGE THE MEAN!

CONDITIONING → "slices the original distribution" → usually gives a SMALLER uncertainty distribution!

$$X = \begin{pmatrix} X_a \\ X_b \end{pmatrix} \sim \mathcal{N}(x; \mu, \Sigma), \text{ where } \Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}$$

CROSS-CORRELATION FACTOR: if $\Sigma_{ab} = \emptyset$ ⇒ no matter what I do, the MEAN DOESN'T CHANGE! (ellipse not rotated)

$$p(X_a \mid X_b) = \frac{p(X_a, X_b)}{\int_{X_a} p(X_a, X_b)\, dX_a} \sim \mathcal{N}(X_a; \mu_{a|b}, \Sigma_{a|b})$$

X_a: continues to be a distribution over X_a!

⇒

$$\mu_{a|b} = \mu_a + \Sigma_{ab} \Sigma_{bb}^{-1} (X_b - \mu_b)$$

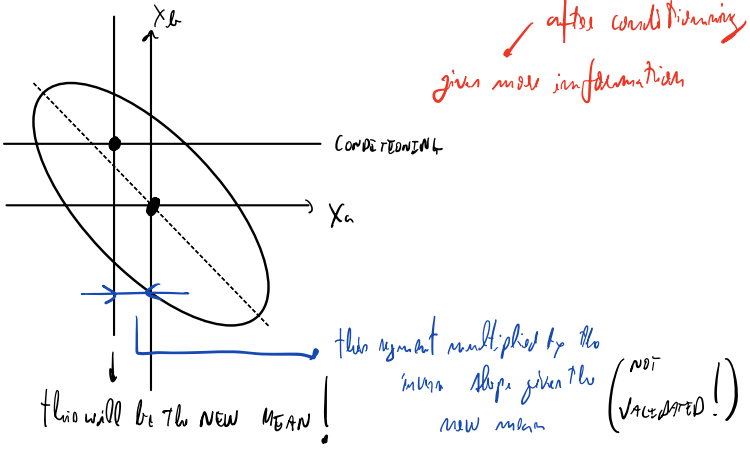
$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab} \Sigma_{bb}^{-1} \Sigma_{ba}, \quad \Sigma_{ba} = \Sigma_{ab}^T$$

or

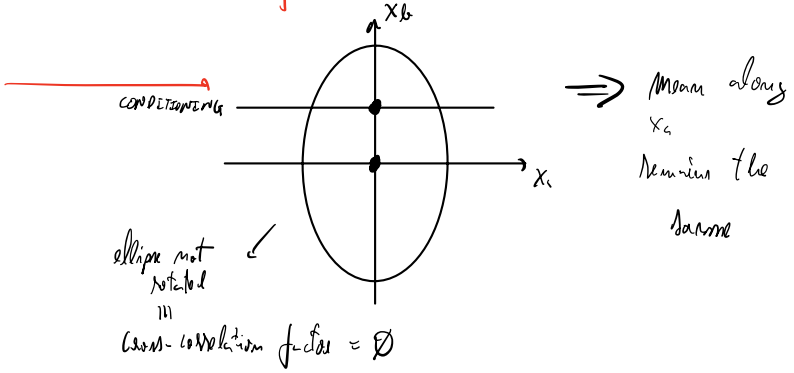
$$\nu_{a|b} = \nu_a - \Omega_{ab} X_b$$

$$\Omega_{a|b} = \Omega_{aa}$$

the more correlated the 2 variables are ⇒ ↓↓ uncertainty after conditioning ⇒ gives more information



if two variables are not correlated:



CHAIN RULE

Known:

$$p(X_a) = \mathcal{N}(X_a; \mu_a, \Sigma_a)$$

$$p(X_b \mid X_a) = \mathcal{N}(X_b; A X_a + c, \Sigma_{b|a})$$

$A X_a + c = \mu_{a|b}$

Query:

$$p(X_a, X_b) = \mathcal{N}(X_{a,b}; \mu_{a,b}, \Sigma_{a,b}) = p(X_b \mid X_a)\, p(X_a)$$

$$\mu_{a,b} = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} = \begin{pmatrix} \mu_a \\ A\mu_a + c \end{pmatrix}$$

$$\Sigma_{a,b} = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} = \begin{pmatrix} \Sigma_a & \Sigma_a A^T \\ A\Sigma_a & \Sigma_{b|a} + A\Sigma_a A^T \end{pmatrix}$$

CANONICAL REPRESENTATION

OR

A is acting as a Cross-correlation factor

$$\mu_{a,b} = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} = \begin{pmatrix} \mu_a \\ A\mu_a + c \end{pmatrix}$$

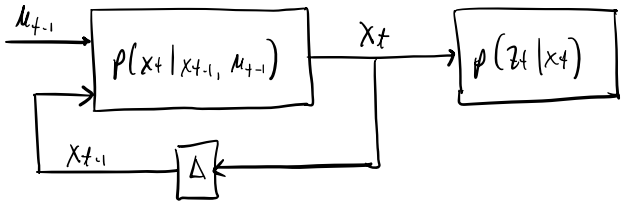
→ the mean usually has more meaning / useful than the information vector !

$$\Omega_{a,b} = \begin{pmatrix} A^T \Omega_{b|a} A + \Omega_a & -A^T \Omega_{b|a} \\ -\Omega_{b|a} A^T & \Omega_{b|a} \end{pmatrix}$$

NOTE: these formulas may be deducted by the PDF function of a Gaussian ! (see appendix of additional notes on lecture 06)

KALMAN FILTERS

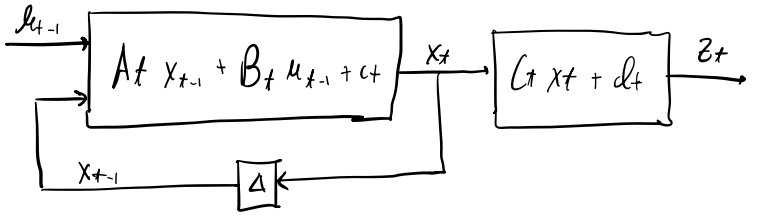
(STOCHASTIC) SYSTEM MODELS



$p(x_t | x_{t-1}, u_{t-1})$: TRANSITION MODEL

$p(z_t | x_t)$: OBSERVATION MODEL

LINEAR SYSTEMS ≡ transition & observations are affine transformations



A: STATE TRANSITION MATRIX

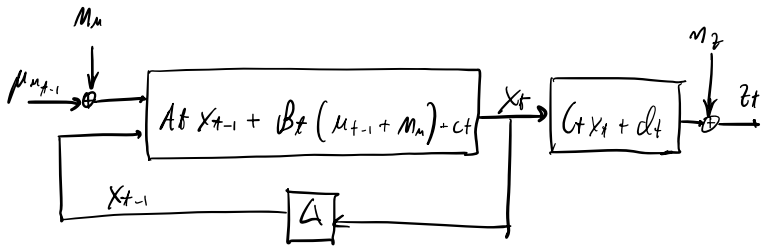
B: INPUT MATRIX

C: OBSERVATION MATRIX

LINEAR SYSTEM W/ GAUSSIAN NOISE

$$p(x_t | x_{t-1}, u_{t-1}) = \mathcal{N}(x_t; A_t x_{t-1} + B_t u_{t-1} + c_t, \Sigma_x)$$

$$p(z_t | x_t) = \mathcal{N}(z_t; C_t x_t + d_t, \Sigma_z)$$



inputs and observations are affected by $\emptyset$-MEAN Gaussian noise

$$u_{t-1} \sim \mathcal{N}(u_{t-1}; \mu_{u,t-1}, \Sigma_{u,t-1})$$

← FOR COMPACTNESS EMBBED THE NOISE IN CONTROL

$$n_u \sim \mathcal{N}(n_u; \emptyset, \Sigma_u)$$

$$n_z \sim \mathcal{N}(n_z; \emptyset, \Sigma_z)$$

FILTERING

Considerations: initial belief is Gaussian; noise is Gaussian; all formulations are affine } Gaussian distributions have closed form operations: affine transformations, chain rule, marginalization, conditioning ⇒ BELIEF REMAINS GAUSSIAN

NOTE: Kalman filters ≡ Bayes filters for [linear systems, Gaussian noise]

- **PREDICT** — goal is to incorporate the control by computing the Gaussian distribution of the next state given the input

$$x_t = \begin{pmatrix} A_t & B_t \end{pmatrix} \begin{pmatrix} x_{t-1} \\ u_{t-1} \end{pmatrix} + c_t$$ ≡ AFFINE TRANSFORMATION

→ 2 variables are independent (cross-correlation factor = $\emptyset$)

PREVIOUS STATE: $\begin{pmatrix} x_{t-1|t-1} \\ u_{t-1} \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mu_{t-1|t-1} \\ \mu_{u,t-1} \end{pmatrix}; \begin{pmatrix} \Sigma_{t-1|t-1} & \emptyset \\ \emptyset & \Sigma_{u,t-1} \end{pmatrix} \right]$

↑ JOINT DISTRIBUTION

⇓

1. 2 distributions as input ⇒ transition model ⇒ next state (predict)
2. apply affine transformation theorems (where $x_b = f(x_a) = A x_a + c$, then $\mu_b = A \mu_a + c$, $\Sigma_b = A \Sigma_a A^T$)

⇓

$p(x_{t|t-1})$ IS STILL GAUSSIAN!

$$p(x_{t|t-1}) \sim \mathcal{N}\left(x_{t|t-1};\ \mu_{t|t-1},\ \Sigma_{t|t-1}\right)$$

$$\mu_{t|t-1} = A_t\,\mu_{t-1|t-1} + B_t\,\mu_{u,t-1} + c_t$$

$$\Sigma_{t|t-1} = A_t\,\Sigma_{t-1|t-1}\,A_t^T + B_t\,\Sigma_{u,t-1}\,B_t^T$$

$\longrightarrow$ note that $A_{affine} = (A_t \ B_t)$ ✗

- **UPDATE** $\longrightarrow$ carry the information of observations to refine our state estimation

① — chain rule ($p(z_t|x_t)$)

② — compute the joint ($p(z_t, x_t)$)

③ — conditioning the joint on the measurement itself ($p(x_{t|t}|z_t)$)

Know:

$$p(z_t|x_t) = \mathcal{N}(z_t;\ C_t x_t + d_t,\ \Sigma_z)$$

$$p(x_t) = p(x_{t|t-1}) = \mathcal{N}(x_{t|t-1};\ \mu_{t|t-1},\ \Sigma_{t|t-1})$$

$$\Downarrow$$

$$p(z_t, x_t) = \mathcal{N}\left(\begin{pmatrix}\mu_{t|t-1}\\ \mu_z\end{pmatrix};\ \begin{pmatrix}\Sigma_{t|t-1} & \Sigma_{t|t-1}\,C_t^T\\ C_t\,\Sigma_{t|t-1} & C_t\,\Sigma_{t|t-1}\,C_t^T + \Sigma_z\end{pmatrix}\right), \text{ where } \mu_z = C_t\,\mu_{t|t-1} + d_t$$

note that $z_t = C_t x_t + d_t$

(!) APPINE TRANSFORMATION

$$\Downarrow$$

$$p(x_{t|t}\,|\,z_t) = \mathcal{N}\left(x_{t|t};\ \mu_{t|t},\ \Sigma_{t|t}\right) \equiv \text{integrate the information of the measurement in the with estimation}$$

$$= \frac{p(x_t, z_t)}{\int_{x_t} p(x_t, z_t)\,dx_t}$$

$$\mu_{t|t} = \mu_{t|t-1} + \underbrace{\Sigma_{t|t-1}\,C_t^T \cdot \left(C_t\,\Sigma_{t|t-1}\,C_t^T + \Sigma_z\right)^{-1}}_{K_t} \cdot \left(z_t - \mu_{z_t}\right)$$

K_t = KALMAN GAIN: expresses how much difference between measurement and expected values influence

$$\Sigma_{t|t} = \Sigma_{t|t-1} - \underbrace{\Sigma_{t|t-1}\,C_t^T \cdot \left(C_t\,\Sigma_{t|t-1}\,C_t^T + \Sigma_z\right)^{-1}}_{K_t} \cdot C_t\,\Sigma_{t|t-1}$$

(OPTIMAL ESTIMATOR)

↳ if all assumptions are expected, we have a perfect estimation

Non-linear Systems: KF → EKF [to deal w/ non-linear systems, NOT OPTIMAL]

⇓

linearize the system: (linearization point (most likely position ≡ mean))

$$f(x,u) \approx f(x_0,u_0) + \underbrace{\frac{\partial f(x,u)}{\partial x}}_{A}(x-x_0) + \underbrace{\frac{\partial f(x,u)}{\partial u}}_{B}(u-u_0)$$

$$= Ax + Bu + \underbrace{f(x_0,u_0) - Ax_0 - Bu_0}_{c}$$

A: JACOBIAN $\partial f/\partial x$; B: JACOBIAN $\partial f/\partial u$

$$h(x,u) \approx h(x_0) + \underbrace{\frac{\partial h(x)}{\partial x}}_{C}(x-x_0)$$

$$= Cx + \underbrace{h(x_0) - Cx_0}_{d}$$

⇓

PREDICT:

$$\mu_{t|t-1} = f(\mu_{t-1|t-1}, \mu_{u,t-1})$$

$$A_t = \frac{\partial f(x,u)}{\partial x}\Big|_{x=\mu_{t-1|t-1}}$$

$$B_t = \frac{\partial f(x,u)}{\partial u}\Big|_{u=\mu_{u,t-1}}$$

$$\Sigma_{t|t-1} = A_t \Sigma_{t-1|t-1} A_t^T + B_t \Sigma_u B_t^T$$

UPDATE:

$$\mu_z = h(\mu_{t|t-1})$$

$$C_t = \frac{\partial h(x)}{\partial x}\Big|_{x=\mu_{t|t-1}}$$

$$K_t = \Sigma_{t|t-1} C_t^T \left(\Sigma_z + C_t \Sigma_{t|t-1} C_t^T\right)^{-1}$$

$$\mu_{t|t} = \mu_{t|t-1} + K_t(z_t - \mu_z)$$

$$\Sigma_{t|t} = (I - K_t C_t)\cdot \Sigma_{t|t-1}$$

EKF Simultaneous Localization and Mapping (SLAM)

(w/ known association)

→ Map not known

→ Landmarks [still able to recognize a specific ID for the landmark
but position of the landmark not known] → State grows over time

Scenario

- Robot [2D plane
  $\Delta\rho, \Delta\theta$ (translational and rotational velocities)]
- Sensor measurements: uniquely distinguishable landmarks
  2D landmark sensor ⟶ landmark position observation relative to the robot's local coordinate frame

⇓

position landmark NOT KNOWN ⟶ NO PRIOR KNOWLEDGE OF THE MAP

⇓

- EKF SLAM:
  1. predict: incorporate new control
  2. correct/update: incorporate new measurements ⟶ $\left(z_t - h(\mu_{t|t-1})\right)$ ≡ INNOVATION
  3. SLAM: add new landmarks to state

Domains

- Robot: $X_t^{[R]} = [R_t \mid t_t] \in SE(2) \Rightarrow x_t^{[R]} = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} \in \mathbb{R}^3$
- Landmarks: $x_t^{[m]} = \begin{pmatrix} x_t^{[m]} \\ y_t^{[m]} \end{pmatrix} \in \mathbb{R}^2, \; m = 1, \dots, N$

⇒ Full State Vector: $X_t = \begin{pmatrix} x_t^{[R]} \\ x_t^{[1]} \\ \vdots \\ x_t^{[N]} \end{pmatrix} \in \mathbb{R}^{(3+2\times N)}$

- Controls: $u_t = \begin{pmatrix} u_t^{[1]} \\ u_t^{[2]} \end{pmatrix} \in \mathbb{R}^2$ (translational, rotational) (DISPLACEMENTS)
- Measurements: $z_t^{[m]} = \begin{pmatrix} x_t^{[m]} \\ y_t^{[m]} \end{pmatrix} \in \mathbb{R}^2, \; m = 1, \dots, M$

N landmarks in the map
M landmark OBSERVATIONS
(M may be ≠ N)

Transition Function

$$x_t = f(x_{t-1}, u_{t-1}) = \begin{pmatrix} x_{t-1} + u_t^{[1]} \cdot \cos(\theta_{t-1}) \\ y_{t-1} + u_t^{[1]} \cdot \sin(\theta_{t-1}) \\ \theta_{t-1} + u_t^{[2]} \\ x_{t-1}^{[1]} \\ \vdots \\ x_{t-1}^{[N]} \end{pmatrix}$$

pose update (the robot moves)

landmarks do not move!

Measurement Function

1 measurement function for each landmark $(x_t^{[m]}, m = 1, \dots, N)$

$$z_t^{[m]} = h^{[m]}(x_t) = R_t^T \left(x_t^{[m]} - t_t\right) = \begin{pmatrix} \cos\theta_t \cdot (x_t^{[m]} - x_t) + \sin\theta_t \cdot (y_t^{[m]} - y_t) \\ -\sin\theta_t \cdot (x_t^{[m]} - x_t) + \cos\theta_t \cdot (y_t^{[m]} - y_t) \end{pmatrix}$$

relative position of the landmark [m] w.r.t robot at time t (with respect to)

Control Noise

$$u_{t,t} \sim \mathcal{N}\left(u_{t,t}; \; \emptyset, \; \begin{pmatrix} \sigma_R^2 + \sigma_T^2 & \emptyset \\ \emptyset & \sigma_R^2 + \sigma_R^2 \end{pmatrix}\right) \quad \Sigma_u$$

$\sigma_T = u_t^{[2]}$

$\sigma_R = u_T^{[2]}$ = proportional to velocity!

Measurement Noise

$$n_z \sim \mathcal{N}\left(n_z; \; \emptyset, \; \begin{pmatrix} \sigma_\theta^2 & \emptyset \\ \emptyset & \sigma_\theta^2 \end{pmatrix}\right)$$ — common constant deviation (e.g., laser scan distance usually describes the σ_{dist})

2D ✓

4D Jacobians

$$A_t = \left.\frac{\partial f(\cdot)}{\partial x}\right|_{x=\mu_{t-1|t-1}} = \begin{pmatrix} \frac{\partial f(\cdot)}{\partial x^{[1]}} & \frac{\partial f(\cdot)}{\partial x^{[2]}} & \cdots & \frac{\partial f(\cdot)}{\partial x^{[N]}} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & -\Delta\cos\theta_{t-1}\cdot\mu_t^{[1]} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -\Delta\sin\theta_{t-1}\cdot\mu_t^{[2]} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

$$B_t = \left.\frac{\partial f(x,u)}{\partial u}\right|_{u=\mu_{t-1}} = \begin{pmatrix} \Delta\cos\theta_{t-1} & 0 \\ \Delta\sin\theta_{t-1} & 0 \\ 0 & \Delta \\ 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

R_t^T does not depend on $x_t^{[n]}, y_t^{[n]}$

t_t " " " θ_t

$$\frac{\partial l^{[n]}}{\partial t_t} = -R_t^T$$

$$\frac{\partial l^{[n]}}{\partial \theta_t} = \frac{\partial R_t^T}{\partial \theta_t}\left(x_t^{[n]} - t_t\right)$$

$$C_t^{[n]} = \left.\frac{\partial l^{[n]}(\cdot)}{\partial x}\right|_{x=\mu_{t|t-1}} = \frac{\partial}{\partial x}\left[R_t^T\left(x_t^{[n]} - t_t\right)\right] = \begin{pmatrix} \frac{\partial l^{[n]}}{\partial x^{[1]}} & 0 & \cdots & \frac{\partial l^{[n]}}{\partial x^{[n]}} & \cdots & 0 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\partial l^{[n]}}{\partial t} & \frac{\partial l^{[n]}}{\partial \theta} & 0 & \cdots & \frac{\partial l^{[n]}}{\partial x^{[n]}} & \cdots & 0 \end{pmatrix} =$$

$$= \begin{pmatrix} -R_t^T & \frac{\partial R_t^T}{\partial \theta_t}\cdot\left(x_t^{[n]} - t_t\right) & 0 & \cdots & \frac{\partial l^{[n]}}{\partial x^{[n]}} & \cdots & 0 \end{pmatrix}$$

Data Association

↓

$j(m) \in [1, \dots, N]$

↳ e.g., $j(3) = 5$

measurement 3 ⇐ landmark 5

Yt originated from a SUBSET of landmarks in the state

Goal: determine association between observations and state

something I've seen before OR something new?

determine which measurement is generated by which landmark in the state

→ for now, we assume we know the assignment $j(m)$

$$\Downarrow$$

$$h(x_t) = \begin{pmatrix} h^{[j(1)]} \\ \vdots \\ h^{[j(M)]} \end{pmatrix} \qquad C_t = \frac{\partial h}{\partial x} = \begin{pmatrix} \frac{\partial h^{[j(1)]}}{\partial x} \\ \vdots \\ \frac{\partial h^{[j(M)]}}{\partial x} \end{pmatrix}$$

Populating the Map → happens every time we see a new landmark

id_to_state_map = (-1, ..., -1) → at the initial iteration

state_to_id_map = (-1, ..., -1)

⋮

landmark 1 is represented in the state as index 3

id_to_state_map = (3, -1, -1, -1, -1, -1, 1, -1, 2, -1, ..., -1)

state_to_id_map = (7, 9, 1, -1, -1, -1, -1, -1, -1, -1, ..., -1)

↓

// element 1 in state vector represents landmark 7

Updating the Map

- already known landmarks (part of the state) → we plan to the update step of EKF
- new landmarks → add them to state AFTER EKF correction

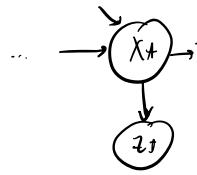
(initialized w/ coordinate of the measurement that originated the landmark

$\begin{pmatrix} \sigma_z^2 & 0 \\ 0 & \sigma_\theta^2 \end{pmatrix}$)

Summary

1. Prediction
2. Correction
   - FOR EACH OBSERVATION
     - id_to_state_map
     - IF unobserved landmark
       - $\hat{z} = h(x_t)$
       - $C_t^{[j(m)]} = \dots$
     - update filter
3. Update map with observed landmarks

Data Association



... → note that this x_t may include the landmarks here

So, indeed the causality relation between landmark → observation is present in the DBN

Problem Definition

Until now, landmarks were UNIQUELY IDENTIFIABLE! (we assume that we know which landmark caused the observation)

What if we do not know how to discriminate this?

- must deal with distributions having multiple maximas
  - Multi-modal distributions
  - no EKF/UKF
- Choose best possible association → ↓ robustness (e.g., if we do wrong association ⇒ VERY BAD!)
  - however, we may still use an EKF/UKF!

current estimate of the state →
history of measurements →

⇓

DATA ASSOCIATION → state possible responsible for a measurement

→ e.g., if we want to use the history of the measurements (Bearing-only observations ...)

$$\#\text{combinations} = \binom{n}{m} = \frac{n!}{m!(n-m)!} \quad \Big| \; n > m$$

(= number of possible associations when we have n landmarks and m observations)

Association Problem

$q^{[j(1)]}(x_t)$ (of the j assignment) ← JOINT DISTRIBUTION ← probability of assignment j

$$j^* = \underset{j}{\operatorname{argmax}} \left\{ p\left(\hat{z}^{[j(1)]} = z^{[1]}, \dots, \hat{z}^{[j(M)]} = z^{[M]} \right) \right\}$$

↳ probability of assignment j

where $j(i) \in [\emptyset, 1, \dots, N]$ and $i = 1, \dots, M$ → $j(n) = \emptyset \equiv$ no correspondence

$X = [x^{[1]}, x^{[1]}, \dots, x^{[N]}] \in \mathbb{R}^{3+2\times N}$ → STATE (robot, landmarks)

$z = z^{[1]}, \dots, z^{[m]}$ → MEASUREMENTS

$\hat{z} = z^{[1]}, \dots, z^{[N]}$ → PREDICTIONS

Naive algorithm → generate all possible associations $\binom{M}{m}$ = # associations

① evaluate the previous formula for each associations

$$p\left(\hat{z}^{[j(1)]} = z^{[1]}, \dots, \hat{z}^{[j(m)]} = \hat{z}^{[m]} \right) =$$

→ CHAIN RULE + MARGINALIZATION

→ = JOINT DISTRIBUTION OF ALL PREDICTIONS GIVEN AN ASSIGNMENT IF WE KNOW

$$= \int_X p\left(\hat{z}^{[j(1)]} = z^{[1]}, \dots, \hat{z}^{[j(m)]} = \hat{z}^{[m]} \mid x\right) \cdot p(x)\, dx =$$ ← THE STATE ...

$$= \int_{\substack{x \\ \in \Omega}} \prod_{m=1}^{M} p\left(\hat{z}^{[j(1)]} = z^{[1]}, \dots, \hat{z}^{[j(m)]} = \hat{z}^{[m]} \mid x^{[j(m)]}, x_n\right) \cdot p\left(x^{[j(m)]}, x_n\right) dx$$

↑ MEASUREMENT INDEPENDENCE

Knowing x_n makes the observation of the landmarks independent of each other

⇓

we would have to repeat the previous formulation FOR EACH ASSOCIATION !!

CAN WE DO SOMETHING MORE SMART ? YES → we can compute the distribution of the prediction ONLY ONCE !

↙ the distribution of the prediction is NOT DEPENDENT on the specific assignment

$$p(\hat{z}) = \int_X p\left(\hat{z} \mid x\right) \cdot p(x)\, dx = \quad \leftarrow \text{chain rule} \oplus \text{marginalization}$$

$$= \int_{x \in \Omega} \prod_{m=1}^{N} p\left(\hat{z}^{[m]} \mid x^{[m]}, x^{n}\right) \cdot p(x)\, dx \quad \leftarrow \text{measurement independence}$$

↑ N blocks (as many as the landmarks) ≡ PREDICTION MEASUREMENT VECTOR $\hat{z}$ → only compute once their distribution evaluate them for each assignment

⇒ INDEPENDENCE LOSS — marginalizing out the states renders the landmark observations NOT INDEPENDENT

⇓ CANNOT "estimate" the likelihood of a single landmark assignment as the product of single likelihoods .....

$$p\left(\hat{z}^{[j]} = z\right) \neq \prod_{m=1}^{M} p\left(\hat{z}^{[j(m)]} = z^{[m]}\right)$$

⇒ INVERSE ASSIGNMENT

$$j(m) = n \Leftrightarrow j^{-1}(n) = m$$

- returns invalid values if landmark does not appear in the measurements (∅: Octave, -1: C++)
- some landmarks may not appear in the observations
- mapping not invertible !

$$z^{[j^{-1}]} = \begin{pmatrix} z^{[j^{-1}[1]]} \\ z^{[j^{-1}[2]]} \\ \vdots \\ z^{[j^{-1}[N]]} \end{pmatrix}$$ → reshuffle measuring vector by reordering the measurements according to the inverse assignment j^{-1}

⇒ $z^{[j]} = \begin{pmatrix} z^{[j^{-1}[1]]} \\ \\ z^{[j^{-1}[3]]} \\ \\ \dots \\ z^{[j^{-1}[N]]} \end{pmatrix}$

Even if mapping not invertible, we put "HOLES" for landmarks not seen or unmatched measurements !

↓ but would have invalid values (NOT INVERTIBLE j^{-1})

PREDICTION VECTOR SAME MEASURING VECTOR !

Under the usual Gaussian assumptions, apply marginalization ⊕ chain rule:

$$p(z|x) = \mathcal{N}(z;\ Cx + d,\ \Sigma_{z|x})$$ ← conditioned over all landmarks

$$p(x) = \mathcal{N}(x;\ \mu_x,\ \Sigma_x)$$ ← prior on pose and landmarks

$$\mu = \begin{pmatrix} \mu^{[R]} \\ \mu^{[1]} \\ \mu^{[2]} \\ \vdots \\ \mu^{[N]} \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \Sigma^{[RR]} & \Sigma^{[R1]} & \Sigma^{[R2]} & \cdots & \Sigma^{[RN]} \\ \Sigma^{[1R]} & \Sigma^{[11]} & \Sigma^{[12]} & & \Sigma^{[1N]} \\ \Sigma^{[2R]} & \Sigma^{[21]} & \Sigma^{[22]} & \cdots & \Sigma^{[2N]} \\ \vdots & \vdots & & \ddots & \vdots \\ \Sigma^{[NR]} & \Sigma^{[N1]} & \Sigma^{[N2]} & \cdots & \Sigma^{[NN]} \end{pmatrix}$$

$$\mu_z = C\mu_x + d$$

$$\Sigma_{zz} = C\Sigma_x C^T + \Sigma_{z|x} = \begin{pmatrix} \Sigma_z^{[11]} & \Sigma_z^{[12]} & \cdots & \Sigma_z^{[1N]} \\ \Sigma_z^{[21]} & \Sigma^{[22]} & & \Sigma_z^{[2N]} \\ \vdots & & & \vdots \\ \Sigma_z^{[N1]} & \Sigma^{[N2]} & \cdots & \Sigma^{[NN]} \end{pmatrix}$$

global C of all landmarks

← joint covariance of the measurements

cross-correlation between measurements

With previous operation that leads to independance loss, all these elements WILL BECOME CORRELATED!

$$z^{[j]} = \begin{pmatrix} z^{[j^{-1}(1)]} \\ \text{HOLE} \\ z^{[j^{-1}(3)]} \\ \vdots \\ z^{[j^{-1}(m)]} \\ \vdots \\ z^{[j^{-1}(N)]} \end{pmatrix}$$

SAME SIZE AS LANDMARKS ⊕ RESHUFFLE VECTOR BY REORDERING ACCORDING TO INVERSE ASSIGNMENT j^{-1}

→ surpass "HOLES" in
- measurements
- prediction
- covariance matrix

⟺ MARGINALIZATION → compressed the joint distribution over all potential measurements to match the size of the measurements

$$\tilde{z}^{[j]} = \begin{pmatrix} z^{[j^{-1}(1)]} \\ z^{[j^{-1}(3)]} \\ \vdots \\ z^{[j^{-1}(m)]} \\ \vdots \\ z^{[j^{-1}(N)]} \end{pmatrix}$$

→ evaluate the Gaussian in the point of the measurement

$$p\left(\hat{z}^{[j(1)]} = z^{[1]}, \dots, \hat{z}^{[j(M)]} = z^{[M]}\right) \propto$$

$$\propto \frac{1}{\dots} \cdot \exp\left(-\left(\tilde{z}^{[j]} - \mu_z\right)^T \tilde{\Sigma}_z^{-1} \cdot \left(\tilde{z}^{[j]} - \tilde{\mu}_z\right)\right)$$

$\tilde{z}$ = estimations!

BUT, WE HAVE AN ISSUE → in all possible associations, there will be a possible association that none of the landmarks in our data are producing our measurement
- will have a probability associated with it
- will probably have the highest probability → element is never the one that $\tilde{z}^{[j]}$ perfectly matches the mean $\tilde{\mu}_z$

favors associations w/ Ø or few associations!

ASSIGN PROBABILITY PENALTY TERM OF NOT OBSERVING A LANDMARK (= event of missing a landmark)

$$p\left(\hat{z}^{[j(1)]} = z^{[1]}, \dots, \hat{z}^{[j(M)]} = z^{[M]}\right) \propto$$

$$\propto \frac{1}{\dots} \cdot \exp\left(-\left(\tilde{z}^{[j]} - \mu_z\right)^T \tilde{\Sigma}_z^{-1} \cdot \left(\tilde{z}^{[j]} - \tilde{\mu}_z\right)\right) \cdot p_{MISS}^{K}$$

number decreases the number of unmatched measurements K

$p_{MISS} < 1$

by playing w/ p_{miss} → favor / prefer unmatched landmarks (w/event of mismatch INDEPENDENT)

→ explain measurement with old landmarks

e.g., if $p_{miss} = 1.0 \Rightarrow$ favor creation of new landmarks over time !

p_{miss} = PARAMETER OF OUR DATA ASSOCIATION ENGINE

INDEPENDENCE ASSUMPTION — non-independence of measurements is $\emptyset$ → NOT EXACT ANYMORE !

(if the landmark are FAR from each other, indeed would tend to $\emptyset$)

⇓

measurement independent from each other (no cross-correlation between them)

$$\mu_z = \begin{pmatrix} \mu_z^{[1]} \\ \mu_z^{[2]} \\ \vdots \\ \mu_z^{[N]} \end{pmatrix} \quad ; \quad \Sigma_z = \begin{pmatrix} \Sigma_z^{[11]} & \emptyset & \cdots & \emptyset \\ \emptyset & \Sigma_z^{[22]} & \cdots & \emptyset \\ \vdots & \vdots & & \\ \emptyset & \emptyset & \cdots & \Sigma_z^{[NN]} \end{pmatrix}$$

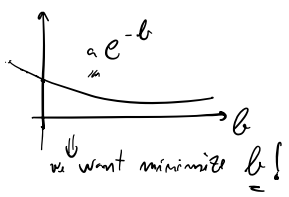
→ ignoring these entries ⇒ NO NEED TO COMPUTE THEM (APPROXIMATION)

⇓

$$p\left(z = \hat{z}^{[j]}\right) = \prod_{m=1}^{M} p\left(z^{[m]} = \hat{z}^{[j(m)]}\right) \longrightarrow \text{MAXIMIZE}$$

$$\log\left(p\left(z = \hat{z}^{[j]}\right)\right) = \sum_{m=1}^{M} \log\left(p\left(z^{[m]} = \hat{z}^{[j(m)]}\right)\right) \quad \Leftrightarrow \quad \text{MINIMIZE}$$

$$\alpha \propto \sum_{m=1}^{M} \left(z^{[m]} - \mu_z^{[m]}\right)^T \left(\Sigma^{[j(m),j(m)]}\right)^{-1} \left(z^{[m]} - \mu_z^{[m]}\right)$$



↓ much more efficient

IF we have m and n (association $n \to m$) $\Rightarrow a_{mn} = \left(z^{[m]} - \mu_z^{[n]}\right)^T \Omega^{[m,n]} \left(z^{[m]} - \mu_z^{[n]}\right)$

↓ we can immediately compute the log-likelihood of an association:

$$\sum_m a_{m,j(m)}$$

ASSIGNMENT PROBLEM

Cost matrix: $A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & & a_{2N} \\ \vdots & & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{pmatrix}$

→ GOAL: find exactly one element for each column
- sum of all entries minimized
- no column selected more than once (assignment)
- matrix made square ($M < N$) by padding with a default value p_{miss}

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & & a_{2N} \\ \vdots & & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \\ l_{MISS} & l_{MISS} & & l_{MISS} \\ l_{MISS} & l_{MISS} & \cdots & l_{MISS} \end{pmatrix}$$

$$l_{MISS} = -\log(p_{MISS})$$

Nearest Neighbour → pick measurement w/ closest prediction

assign to each landmark the closest measurement $\iff a_{mn} = (z^{[m]} - \mu_z^{[m]})^T (z^{[m]} - \mu_z^{[m]})$

if the eigen values of the measurement covariances are equal $\Rightarrow$ greedy association ≡ nearest neighbour

EVEN nearest neighbor can be very expensive w/ very large M, N

Greedy Association

- select for each column, the row w/ smallest value
- might result in undesired configurations [multiple measurements can be assigned to the same landmark
  a landmark can be assigned to a far measurement because nothing better is found

Pruning Heuristics → Why? Bad associations are EVIL !!! → pruning potential wrong associations

① **Gating**: ignore all associations whose cost is higher than a threshold

```
FOR m = 1:N
    [a_min, min_idx] = min(A(m, :))
    IF (a_min < γ)
        associations = [
            associations;
            m, min_idx, a_min
        ];
```

② **Best Friends**: an association should be the best (i.e., the minimum) of both row and column

get rid of ambiguities

```
FOR i = 1: #gated_associations
    a_min                = associations(i, 3);
    proposed_landmark_id = associations(i, 2);
    min_on_column        = min(A(:, proposed_landmark_id))
    IF (a_min ≠ min_on_column)
        associations(i, 2) = ∅        → ≡ invalid association
```

(3) Lonely but friends: one measurement should be CLEARLY only assigned

difference between minimum and second minimum of row / column should be higher than threshold

```
FOR i = 1 : #gated associations
    a_mm                  = associations(i, 3)
    measurement_id        = associations(i, 1)
    proposed_landmark_id  = associations(i, 2)

    IF proposed_landmark_id == ∅
        CONTINUE

    ordered_row           = minigue(A(measurement_id, :))
    second_row_min_value  = ordered_row(2)

    ordered_col           = minigue(A(:, proposed_landmark_id))
    second_col_min_value  = ordered_col(2)

    IF ((second_row_min_value < τ_LONELY) || (second_col_min_value < τ_LONELY))
        associations(i, 2) = ∅
```

EKF SLAM w/ UNKNOWN ASSOCIATION → in this case, we do not know the ID of the observations

Scenario: Omnio — translational + rotational velocities
senses a set of NON DISTINGUISHABLE landmarks = relative position of the landmarks relative to the robot frame but unknown ID

⇓

KF-based algorithm to track
- Omnio pose (LOCALIZATION)
- position of observed plant-landmarks (MAPPING)

location of the landmarks in the world NOT known!

While performing data association ⇒ but no prior KNOWLEDGE OF THE MAP

EKF SLAM

1. PREDICT: incorporate new control

$$\mu_{t|t-1} = f(\mu_{t-1|t-1}, u_{t-1})$$

$$A_t = \left.\frac{\partial f(x,u)}{\partial x}\right|_{x=\mu_{t-1|t-1}}$$

$$B_t = \left.\frac{\partial f(x,u)}{\partial u}\right|_{u=\mu_{u,t-1}}$$

$$\Sigma_{t|t-1} = A_t \cdot \Sigma_{t-1|t-1} \cdot A_t^\top + B_t \cdot \Sigma_u \cdot B_t^\top$$

PROJECT PREVIOUS COVARIANCE OF THE STATE — UNCERTAINTY OF THE CONTROLS

2. CORRECT: incorporate new measurement

$$C_t = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\mu_{t|t-1}}$$

$$K_t = \Sigma_{t|t-1} \cdot C_t^\top \cdot \left(\Sigma_z + C_t \cdot \Sigma_{t|t-1} \cdot C_t^\top\right)^{-1}$$

$$\mu_{t|t} = \mu_{t|t-1} + K_t \cdot \left(z_t - h(\mu_{t|t-1})\right)$$

INNOVATION

$$\Sigma_{t|t} = (I - K_t C_t) \cdot \Sigma_{t|t-1}$$

3. ADD: extend state w/ new landmarks

DOMAINS

state space:

$$X_t^{[n]} = [R_t | t_t] \in SE(2) \longrightarrow X_t^{[n]} = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} \in \mathbb{R}^3$$

landmarks in the state ← $X^{[m]} = \begin{pmatrix} x_t^{[m]} \\ y_t^{[m]} \end{pmatrix} \in \mathbb{R}^2$, $m = 1, \dots, N$

$$\longrightarrow X_t = \begin{pmatrix} X_t^{[n]} \\ X^{[1]} \\ \vdots \\ X^{[N]} \end{pmatrix} \in \mathbb{R}^{(3+2\cdot N)}$$

control inputs space:

$$u_t = \begin{pmatrix} u_t^x \\ u_t^\theta \end{pmatrix} \in \mathbb{R}^2$$

space of observations (measurements):

$$z_t^{[m]} = \begin{pmatrix} x_t^{[m]} \\ y_t^{[m]} \end{pmatrix} \in \mathbb{R}^2, \quad m = 1, \dots, M \longrightarrow Z_t = \begin{pmatrix} z_t^{[1]} \\ \vdots \\ z_t^{[M]} \end{pmatrix} \in \mathbb{R}^{(2\times M)}$$

TRANSITION MODEL

$$X_t = f(x_{t-1}, u_{t-1}) = \begin{pmatrix} x_{t-1} + u_{t-1}^x \cdot \cos(\theta_{t-1}) \\ y_{t-1} + u_{t-1}^x \cdot \sin(\theta_{t-1}) \\ \theta_{t-1} + u_{t-1}^\theta \\ X_t^{[1]} \\ \vdots \\ X_t^{[N]} \end{pmatrix}$$

Kinematic model of UNICYCLE after 1st-order Euler integration

landmarks do not move!

Control Noise:

$$n_{u,t} \sim \mathcal{N}\left(n_{u,t};\ \emptyset,\ \begin{pmatrix} (u_t^x)^2 + \sigma_w^2 & \emptyset \\ \emptyset & (u_t^\theta)^2 + \sigma_w^2 \end{pmatrix}\right)$$

$$A_t = \left.\frac{\partial f(\cdot)}{\partial x}\right|_{x=\mu_{t-1|t-1}} = \begin{pmatrix} 1 & 0 & -u_{t-1}^x \cdot \sin(\theta_{t-1}) & \emptyset_{3\times(2\cdot N)} \\ 0 & 1 & u_{t-1}^x \cdot \cos(\theta_{t-1}) & \\ 0 & 0 & 1 & \\ \hline \emptyset_{(2\cdot N)\times 3} & & & \begin{matrix} 1 & \emptyset \\ \emptyset & 1 \end{matrix} \end{pmatrix}_{N\times N}$$

$$B_t = \left.\frac{\partial f(\cdot)}{\partial u}\right|_{u=\mu_{u,t-1}} = \begin{pmatrix} \cos(\theta_{t-1}) & \emptyset \\ \sin(\theta_{t-1}) & \emptyset \\ \emptyset & 1 \\ \emptyset_{(2\cdot N)\times 2} & \end{pmatrix}$$

Observation Model

$$z_t^{[m]} = h^{[m]}(x_t) = R_t^\top \cdot (x_t^{[m]} - t_t) \longrightarrow C_t^{[m]} = \frac{\partial h^{[m]}(\cdot)}{\partial x_t} = \begin{pmatrix} \frac{\partial h^{[m]}}{\partial x_t^{\wedge}} & \emptyset & \dots & \frac{\partial h^{[m]}}{\partial x_t^{[m]}} & \dots & \emptyset \end{pmatrix}$$

↓ usually, $M < N$ measurement functions, one for each observed landmark that is part of the state (N)

↑ POSE BLOCK

↑ LANDMARK

$$\frac{\partial h^{[m]}}{\partial x_t^{\wedge}} = \begin{pmatrix} -R_t^\top & \frac{\partial R_t^\top}{\partial \theta_t} \cdot (x_t^{[m]} - t_t) \end{pmatrix}$$

$$\frac{\partial h^{[m]}}{\partial x_t^{[m]}} = R_t^\top$$

Data Association

[we do not observe landmark IDs
when new landmark appears $\Rightarrow$ it is our responsibility to assign a new UNIQUE ID

$$\text{ID LAND.} \longrightarrow \text{IDX STATE} \equiv \text{id\_to\_state\_map} = (-1, \dots, -1)$$

$$\text{IDX STATE} \longrightarrow \text{ID LAND.} \equiv \text{state\_to\_id\_map} = (-1, \dots, -1)$$

At each time step, compute LIKELIHOOD of the associations for each MEASUREMENT — LANDMARK:

$$a_{m,n} = \left(z^{[m]} - h^{[n]}(x_t)\right)^\top \cdot \Omega_{m,n} \cdot \left(z^{[m]} - h^{[n]}(x_t)\right)$$

$h^{[n]}(x_t)$: predicted landmark; $\Omega_{m,n}$: information matrix

$\equiv$ MAHALANOBIS DISTANCE (takes into consideration the uncertainty of our measurement)

$$\Omega_{m,n} = \Sigma_{m,n}^{-1}$$

$$\Sigma_{m,n} = C_t^{[m]} \cdot \Sigma_x \cdot \left(C_t^{[m]}\right)^\top + \Sigma_{z|x}$$

1st order PROPAGATION: project the covariance into the measurement state

↓ assemble the cost MATRIX

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \dots & a_{MN} \end{pmatrix}$$

Pseudo-code

1. Prediction
2. Associate landmark IDs
3. Correction
4. Add new landmarks

UNSCENTED TRANSFORM → alternative way of parametrizing a Gaussian Distribution

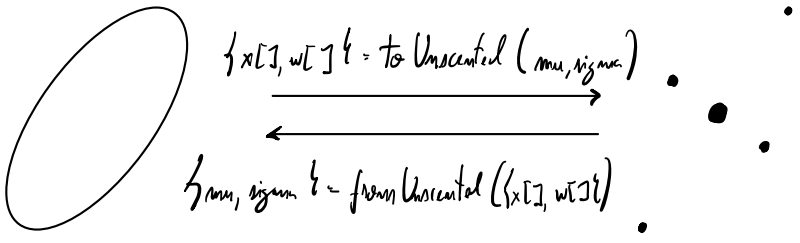
Gaussian: $p(x) = \mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}\sqrt{\det \Sigma}} \cdot \exp\left(-\frac{1}{2}(x-\mu)^T \cdot \Sigma^{-1} \cdot (x-\mu)\right)$ → can be determined experimentally

probability density function

⇓

⇒ **UNSCENTED GAUSSIAN**

- control points ≡ samples selected deterministically
- weights → the heavier a control point would be, the more the mean would lean towards the point



→ samples w/ weights

⇓

$$\text{SIGMA POINTS } \chi^{(i)} = \begin{cases} \text{position} & \chi^{(i)} \in \Omega \\ \text{weight for mean} & W_m^{(i)} \in \mathbb{R}^+ \\ \text{weight for covariance} & W_c^{(i)} \in \mathbb{R}^+ \end{cases}$$

2 scalars (to reconstruct mean and covariance from sample pts, respectively)

$$\mu = \sum_i W_m^{(i)} \chi^{(i)}$$

$$\Sigma = \sum_i W_c^{(i)} \cdot (\chi^{(i)} - \mu) \cdot (\chi^{(i)} - \mu)^T$$

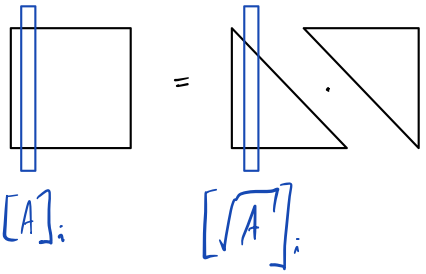
similar experimental & computation of the Gaussian parameters

⇒ **CHOLESKY DECOMPOSITION** — CHOL (in MATLAB/Octave)

$A = LL^T$

↑ ↳ lower triangular matrix

Hermitian positive-definite matrix

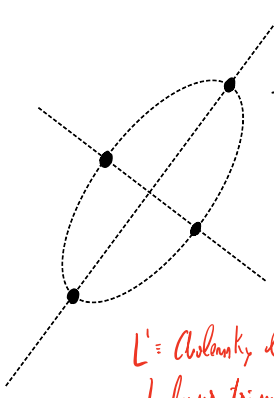


→ can be seen as $L \cong$ square root of A

$L = \sqrt{A}$ ABUSE OF NOTATION

↓

computed by using a variant of Gaussian elimination



→ to put these points along the axes along the ellipse would require the computation of the EIGEN VALUES

(very costly for ++ dimensionality)

⇓

CHOLESKY DECOMPOSITION → in a way to choose the sigma points

n — dimension of the state → # sigma points = $2n+1$ → more DOF than the Gaussian parametrization

$\lambda = \alpha^2 n$

$L = \sqrt{(n+\lambda)\Sigma}$

L' = Cholesky decomposition of $(n+\lambda)\cdot A$

L lower triangular matrix of VECTORS

$\alpha \in]0, 1]$ (influences how far the points are from the mean)

$\alpha = 10^{-3}$, $\beta = 2$ → HOW TO SELECT ?

the important thing is when going forward and backward we still have the SAME Gaussian distribution

$K = \emptyset$

determine how to select the sigma points, given that we have selected the set of directions

$$\mathcal{X}^{(0)} = \mu$$

$$\mathcal{X}^{(i)} = \mu + [L']_i \quad , \text{ for } i \in [1, \dots, n]$$

$$\mathcal{X}^{(i)} = \mu - [L']_{n-i} \quad , \text{ for } i \in [n+1, \dots, 2n]$$

↳ order does not matter!
(sum operation destroys the order)

$$W_m^{(0)} = \frac{\lambda}{n+\lambda}$$

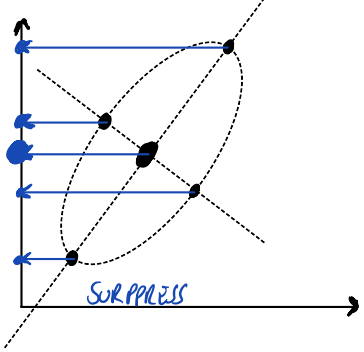
$$W_c^{(0)} = W_m^{(0)} + (1 - \alpha^2 + \beta)$$

$$W_c^{(i)} = W_m^{(i)} = \frac{1}{2 \cdot (n+\lambda)}$$

WHAT IS THE GOAL OF UNSCENTED TRANSFORM? Simple operations when using the Gaussians!

- Jacobians can be error prone
- Unscented transform does not require Jacobians

UNSCENTED MARGINALIZATION



- Keep sigma points & suppress dimensions
- Weights remain the same

→ but we end up with more sigma points than what we need → ⋮ ⇒ μ, Σ ⇒ ⋮

$$\mathcal{X} = \begin{pmatrix} \mathcal{X}_a \\ \mathcal{X}_b \end{pmatrix} \text{ be a random variable}$$

→ marginal $p(x_a) = \int_{x_b} p(x_a, x_b)\, dx_b$

$$\mathcal{X} \sim \mathcal{UT}(x;\ \mathcal{X}^{(i)}, W_m^{(i)}, W_c^{(i)})$$

$$\mathcal{X}_a \sim \mathcal{UT}(x_a;\ \mathcal{X}_a^{(i)}, W_m^{(i)}, W_c^{(i)})$$

UNSCENTED FUNCTIONS

$$\mathcal{X}_a \sim \mathcal{UT}(x_a;\ \mathcal{X}_a^{(i)}, W_m^{(i)}, W_c^{(i)})$$

$$\mathcal{X}_b = f(\mathcal{X}_a)$$

→

$$\mathcal{X}_b \sim \mathcal{UT}(x_b;\ \mathcal{X}_b^{(i)}, W_m^{(i)}, W_c^{(i)})$$

$$\mathcal{X}_b^{(i)} = f(\mathcal{X}_a^{(i)}) \equiv \mathcal{X}_b^{(i)} = f(\mathcal{X}_a^{(i)})$$

WEIGHTS ARE PRESERVED!

- may be NON-LINEAR (but not guaranteed if it is still a Gaussian)

but IF AFFINE, $\mathcal{X}_b$ still represents Gaussian

Unscented Chain Rule

$$x_a \sim \mathcal{UT}(x_a;\ x_a^{(i)},\ w_m^{(i)},\ w_c^{(i)})$$

$$p(x_b \mid x_a) = \mathcal{N}(x_b;\ f(x_a),\ \Sigma_{b|a}) \xrightarrow{\text{WE WANT TO COMPUTE ...}} p(x_a, x_b) = \mathcal{N}(x_{a,b};\ \mu_{a,b},\ \Sigma_{a,b})$$

$$\mu_{a,b} = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} = \begin{pmatrix} \mu_a \\ f(\mu_a) \end{pmatrix}$$

$$\Sigma_{a,b} = \begin{pmatrix} \Sigma_a & \Sigma_{a,b} \\ \Sigma_{b,a} & \Sigma_{b|a} + \Sigma_{b,b} \end{pmatrix}$$

CROSS-CORRELATION COEFFICIENTS

covariance due to projection through $f(x_a)$

$$p(x_a, x_b) = p(x_b \mid x_a) \cdot p(x_a)$$

$$\begin{pmatrix} x_a \\ x_b \end{pmatrix} \sim \begin{pmatrix} x_a \\ f(x_a) \end{pmatrix} \Leftarrow$$

JOINT DISTRIBUTION

$$x_b^{(i)} = f(x_a^{(i)})$$

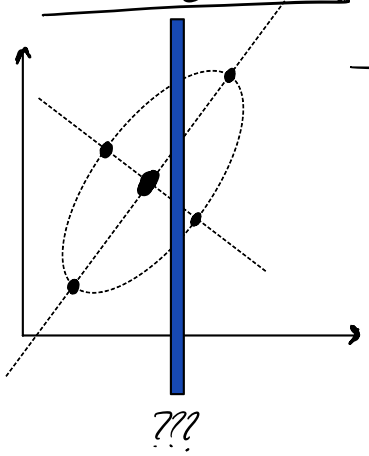
$$\mu_b = \sum_i w_m^{(i)} \cdot f(x_a^{(i)})$$

UNSCENTED (CROSS) CORRELATIONS:

$$\Sigma_{ab} = \sum_i w_c^{(i)} \cdot (x_a^{(i)} - \mu_a)(x_b^{(i)} - \mu_b)^T$$

$$\Sigma_{bb} = \sum_i w_c^{(i)} \cdot (x_b^{(i)} - \mu_b)(x_b^{(i)} - \mu_b)^T$$

Unscented Conditioning



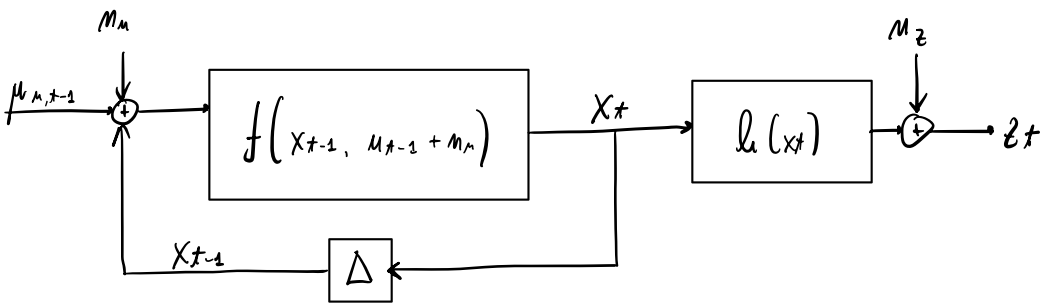
???

→ With sigma points, we do not have a continuous $\Rightarrow$ Go back to Gaussian

↓ Conditioning

↓ Unscented Transform

Unscented Filtering



GAUSSIAN NOISE

$$m_m \sim \mathcal{N}(m_m \mid \emptyset, \Sigma_m)$$

$$m_z \sim \mathcal{N}(m_z \mid \emptyset, \Sigma_z)$$

$$p(x_0) = \mathcal{N}(x_0 \mid \mu_0, \Sigma_0)$$

↑ INITIAL BELIEF ~ GAUSSIAN

- Unscented transform behaves better under non-linear transformations

⇓

REPLACE
- unscented projection through function to compute the prediction
- unscented chain rule to compute the joint distribution over measurements and predicted states ⊕ NO JACOBIANS
- Conditioning remains the same as in (E)KF

1. PREDICT : $x_t = f(x_{t-1}, \mu_{t-1})$

$$\begin{pmatrix} x_{t-1|t-1} \\ \mu_{t-1} \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mu_{t-1|t-1} \\ \mu_{\mu,t-1} \end{pmatrix}, \begin{pmatrix} \Sigma_{t-1|t-1} & \emptyset \\ \emptyset & \Sigma_{\mu,t-1} \end{pmatrix} \right]$$

$$\Downarrow$$

$\chi^{(i)}_{t-1|t-1}$ — $\dim(x) + \dim(\mu)$

project the point through transition function: $\chi^{(i)}_{t|t-1} = f(\chi^{(i)}_{t-1|t-1})$

$\dim(x)$

$[\text{sigma\_pts}, W\_m, W\_c] = \text{PREDICT}(\text{mu}, \text{sigma}, \text{controls})$

1. gaussian parameters $\longrightarrow$ sigma points $\chi^{(i)}_{t-1|t-1}$
2. FOR EACH $\chi^{(i)}_{t-1|t-1}$
   - $\chi^{(i)}_{t|t-1} = f(\chi^{(i)}_{t-1|t-1})$

↓

but the weights remain the SAME !

2. UPDATE :

$p(z_t|x_t) = \mathcal{N}(z_t ; h(x_t), \Sigma_{z|x})$

$p(x_t) = \mathcal{UT}(x_t ; \chi^{(i)}_{t|t-1})$

WE WANT TO COMPUTE ... $\longrightarrow$ $\mu_{x,z} = \begin{pmatrix} \mu_x \\ \mu_z \end{pmatrix}$

$$\Sigma_{x,z} = \begin{pmatrix} \Sigma_x & \Sigma_{xz} \\ \Sigma_{zx} & \Sigma_{z|x} + \Sigma_{zz} \end{pmatrix}$$

Project $x_{t|t-1} \sim \mathcal{UT}(x_t ; \chi^{(i)}_{t|t-1})$

$z_t^{(i)} = h(\chi^{(i)}_{t|t-1})$

$\mu_z = \sum_i w_m^{(i)} \cdot z_t^{(i)}$

$\Rightarrow$

$\Sigma_{xz} = \sum_i w_c^{(i)} \cdot (\chi^{(i)}_{t|t-1} - \mu_{t|t-1}) \cdot (z_t^{(i)} - \mu_z)^T$

$\Sigma_{zz} = \sum_i w_c^{(i)} \cdot (z_t^{(i)} - \mu_z) \cdot (z_t^{(i)} - \mu_z)^T$

Conditioning

$\mu_{x,z} = \begin{pmatrix} \mu_x \\ \mu_z \end{pmatrix}$

$\Sigma_{x,z} = \begin{pmatrix} \Sigma_x & \Sigma_{xz} \\ \Sigma_{zx} & \Sigma_{z|x} + \Sigma_{zz} \end{pmatrix}$

$\Rightarrow$

$\mu_{t|t} = \mu_{t|t-1} + \Sigma_{xz} \cdot (\Sigma_{z|x} + \Sigma_{zz})^{-1} \cdot (z_t - \mu_z)$

$\Sigma_{t|t} = \Sigma_{t|t-1} - \Sigma_{xz} \cdot (\Sigma_{z|x} + \Sigma_{zz})^{-1} \cdot \Sigma_{zx}$

$[\text{mu}, \text{sigma}] = \text{UPDATE}(\text{sigma\_pts}, W\_m, W\_c, \text{landmarks}, \text{observations})$

1. $z_t^{(i)} = h(\chi^{(i)}_{t|t-1})$
2. $z_t^{(i)} \longrightarrow \mu_z, \Sigma_z$
3. $\chi^{(i)}_{t|t-1} \longrightarrow \mu_{t|t-1}, \Sigma_{t|t-1}$
4. $\Sigma_{xz} = \sum_i w_c^{(i)} \cdot (\chi^{(i)}_{t|t-1} - \mu_{t|t-1}) \cdot (z_t^{(i)} - \mu_z)^T$
5. UPDATE THE FILTER :
   - $\mu_{t|t} = \mu_{t|t-1} + \Sigma_{xz} \cdot (\Sigma_{z|x} + \Sigma_z)^{-1} \cdot (z_t - \mu_z)$
   - $\Sigma_{t|t} = \Sigma_{t|t-1} - \Sigma_{xz} \cdot (\Sigma_{z|x} + \Sigma_z)^{-1} \cdot \Sigma_{zx}$

↑ uncertainty introduced by the measurement

* UKF is a non-linear extension of EKF
* Reputed to better deal with non-linear transition and observation functions
* However, still suffers the Gaussian assumption and uses pure Gaussian conditioning
* DERIVATIVE FREE !

Particle Distribution & Particle Filters

In general, in computer science RANDOMNESS does not exist → all deterministic !

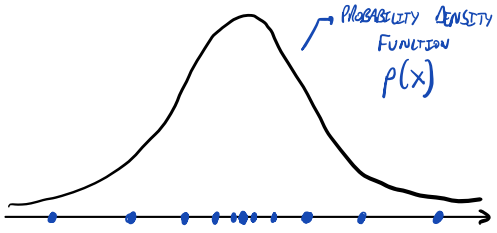
⇒ But there are programs to generate samples from uniform distribution (UNIFORM between 0 and 1)

- state machine → initialized with a seed
- example: <stdlib.h> → drand48()

↓

by altering the initial state / seed, we can decide how to play the random number

Sampling from a distribution

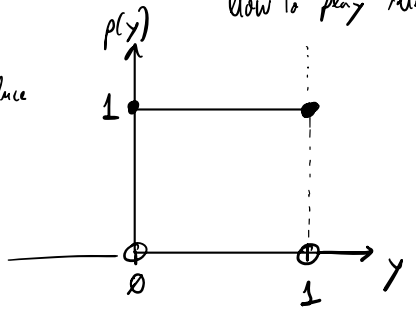


$x^{(i)} \sim p(x)$

↓

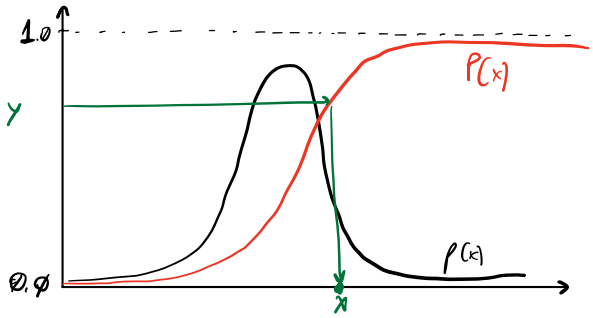
While most random generators produce from uniform distribution

$y^{(i)} \sim U(0, 1)$



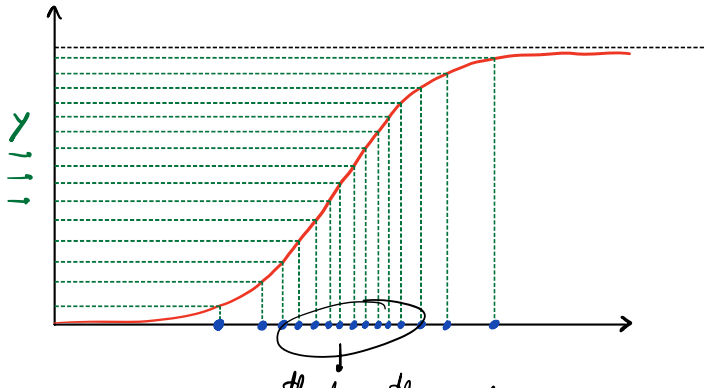
- Assuming we have a closed-form for $p(x)$:

1. Compute cumulative distribution: $P(x) = \int_{-\infty}^{\infty} p(x')\,dx'$ → monotonic growing function



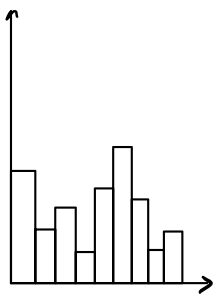
2. Draw a sample from $y^{(i)} \sim U(0, 1)$

3. Compute the inverse of the cumulative at $y^{(i)}$ $\left(x^{(i)} = P^{-1}(y^{(i)})\right)$

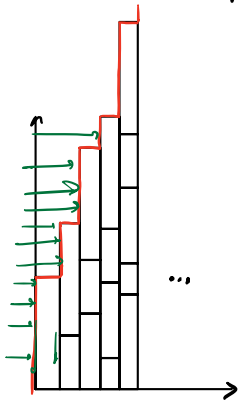


the denser the samples are, the higher the probability of $p(x)$ is !

- Discrete case:



CUMULATIVE ⇒



However,

Computation problem: sampling strategy from unknown distributions

computations ~ millions of samples

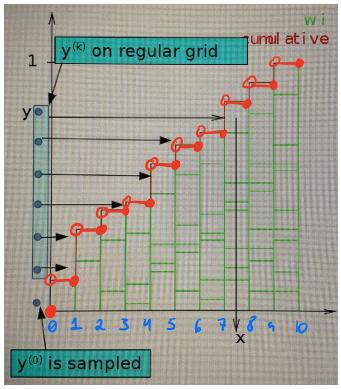
⇓

INEFFICIENT !

Uniform Sampling — generate N samples drawn from a discrete distribution at random by invoking 1 only once the random generator

1. Sample a value $y^{(0)}$ between 0 and $1/N$

$$y^{(0)} \sim \mathcal{N}\left(0, \frac{1}{N}\right)$$



2. Pick the remaining $y^{(i)}$ samples in a regular grid

$$y^{(K)} = y^{(0)} + \frac{K}{N}$$

——— doing the inversion, this way we are only moving forward in searching the cumulative distribution

(instead of log N search time)

```
FUNCTION sampled_indices = uniformSample (weights, num_desired_samples)
    normalized = 1./sum(weights)
    sampled_indices = zeros(num_desired_samples, 1)
    step = 1./num_desired_samples
    y0 = rand() * step
    yi = y0
    cumulative = 0
    sample_index = 1 ——— in Octave, one-based!
    FOR (weight_index = 1:size(weights,1))
        cumulative += normalized * weights(weight_index)
        WHILE (cumulative > yi)
            sampled_indices(sample_index) = weight_index
            sample_index ++
            yi += step
        END
    END
END
```

Importance Sampling ——— sometimes we do not know the sampling distribution $\Rightarrow$ CANNOT COMPUTE INVERSE CUMULATIVE (but still able to evaluate POINT-WISE)

1. Sample from a known distribution $\tilde{\Pi}(x)$ possibly close to $p(x)$

↳ do not know it in a close equation!

$$x^{(i)} \sim \tilde{\Pi}(x) \equiv \text{PROPOSAL DISTRIBUTION}$$

↑ supports samples (uniform, Gaussians, ...)

→ $\tilde{\Pi}(x)$ should cover all the relevant portion of the TARGET DISTRIBUTION $p(x)$ ≡ goal is to entirely cover the full domain of the target distribution

↳ otherwise, some feasible samples might not be generated...

(+)

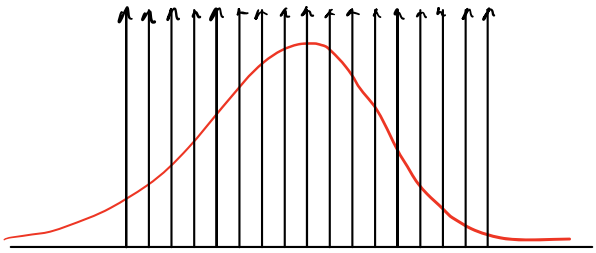
$$p(x) > 0 \implies \Pi(x) > 0$$

——— $\Pi(x)$ must be non-zero in the areas $p(x)$ is non-zero !

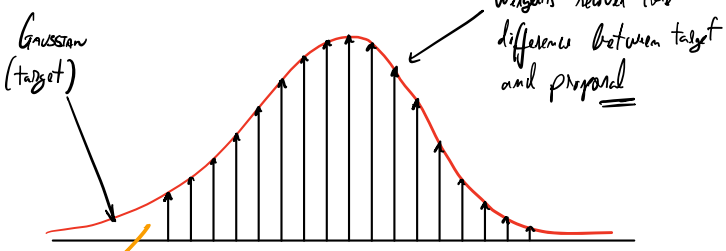
2. Compute a weight $w^{(i)} = \frac{p(x^{(i)})}{\pi(x^{(i)})}$ — TARGET DISTRIBUTION / PROPOSAL DISTRIBUTION

note that we are still able to evaluate point-wise $p(x)$

↳ if the proposal distribution $\pi(x) = p(x)$ target distribution ⟹ ALL THE WEIGHTS WOULD BE UNIFORM !



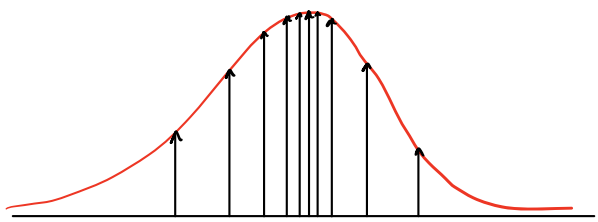
⟹



but we may be over-representing irrelevant samples

↳ RESAMPLING — in order to turn a weighted sample set into an unweighted one (uniform), we need:
- to repeat samples having high weights
- suppress samples with low weight

⇓ RESAMPLING



Why using re-sampling ? new generation after resampling allows the filter to focus computation on likely regions of the state space, while suppressing non-meaningful samples !

How to do resampling?

1. input weight sample set $\{\langle w^{(i)}, x^{(i)} \rangle\}$

2. sample a set of indices w/ probability proportional to the weights

$$\{j^{(i)}\} = \text{uniform\_sample}(\{w^{(i)}\})$$

3. new sample in position i is old sample in position $j[i]$

$$\tilde{x}^{(i)} = x^{(j(i))}$$

⇓

After resampling, all particles have the same weight !

Particles

- particle densities → another way to represent probability distributions
- particles → DOES NOT RESTRICT the distribution to a Gaussian

denser samples in the region, higher will be probability of that region

each sample ||| discrete event ⇒

$$x^{[i]} \sim p(x) \quad \text{(Dirac centered at } x^{(i)}\text{)}$$

$$p(x) \cong \sum_i w^{(i)} \cdot \delta\left(x - x^{[i]}\right)$$

$$\int_E p(x)\,dx \cong \sum_{x^{[i]} \in E} w^{(i)}$$

→ probability that x falls in a region E can be obtained by SUMMING THE WEIGHTS in the region

Why particles are cool?

- can represent arbitrary distributions
- easy to "visualize"
- easy to manipulate
- good for small state spaces

Transformation

$$p(x_a) \cong \sum_i w^{(i)}\, \delta\left(x_a - x_a^{(i)}\right)$$

$$x_b = f(x_a)$$

↑ function of a random variable

→

$$p(x_b) \cong \sum_i w^{(i)}\, \delta\left(x_b - f(x_a^{(i)})\right)$$

$$x_b = f(x_b)$$

↑ weights remain the same!

||| TRANSFORMING EACH SAMPLE WITH $f(\cdot)$

Marginalization

$$p(x_a, x_b) \cong \sum_i w^{(i)}\, \delta\left(\begin{pmatrix} x_a \\ x_b \end{pmatrix} - \begin{pmatrix} x_a^{(i)} \\ x_b^{(i)} \end{pmatrix}\right)$$

→

$$p(x_a) = \int p(x_a, x_b)\,dx_b$$

$$\cong \sum_i w^{(i)}\, \delta\left(x_a - x_a^{(i)}\right)$$

≡ DELETES FROM SAMPLE SET THE COORDINATES OF THE MARGINALIZED COMPONENT

Chain Rule

$$p(x_a) \cong \sum_i w^{(i)}\, \delta\left(x_a - x_a^{(i)}\right)$$

$p(x_b \mid x_a)$ ↷ $p(x_a, x_b) = p(x_b \mid x_a) \cdot p(x_a)$

→

1. generate a sample from the conditional, for each sample in the conditionings

$$x_b^{(i)} \sim p\left(x_b \mid x_a^{(i)}\right)$$

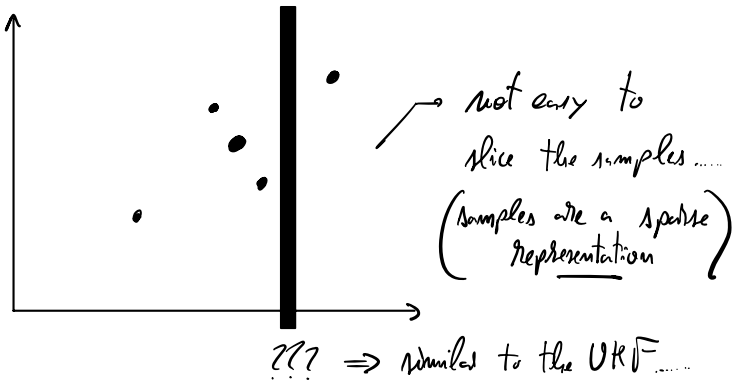
2. stack the samples to get a PARTICLE from the joint distribution

$$p(x_a, x_b) \cong \sum_i w^{(i)} \cdot \delta\left(\begin{pmatrix} x_a \\ x_b \end{pmatrix} - \begin{pmatrix} x_a^{(i)} \\ x_b^{(i)} \end{pmatrix}\right)$$

CONDITIONING

$$p(x_a, x_b) \simeq \sum_i w^{(i)} \cdot \delta\left(\begin{pmatrix} x_a \\ x_b \end{pmatrix} - \begin{pmatrix} x_a^{(i)} \\ x_b^{(i)} \end{pmatrix}\right)$$

$$p(x_a \mid x_b) = \frac{p(x_a, x_b)}{p(x_b)}$$



not easy to slice the samples ..... (samples are a sparse representation)

??? ⇒ similar to the UKF .....

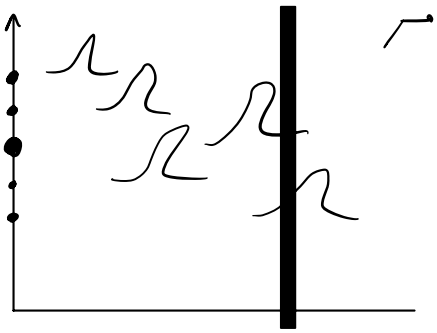
$$\Downarrow$$

IS ONLY A NUMBER

$$p(x_a, x_b) \simeq \sum_i w^{(i)} \delta\left(x_a - x_a^{(i)}\right) \cdot p\left(x_b \mid x_a^{(i)}\right)$$

SAMPLE — CONDITIONAL OVER THE SAMPLE

(assuming we are able to sampling the distribution $x_b \mid x_a^{(i)}$)



a lot more simple if we would have for each sample a conditional distribution on x_b

$$\Downarrow$$

$$p(x_a \mid x_b) = \frac{p(x_a, x_b)}{p(x_b)} = \frac{p(x_a, x_b)}{\int p(x_a, x_b)\, dx_a}$$

$$\simeq \frac{\sum_i w^{(i)} \delta(x_a - x_a^{(i)})\, p(x_b \mid x_a^{(i)})}{\int \left[\sum_i w^{(i)} \delta(x_a - x_a^{(i)})\, p(x_b \mid x_a^{(i)})\right] dx_a} =$$

$p(x_b \mid x_a^{(i)})$ ↑ it is just a number

$$= \frac{\sum_i w^{(i)} \delta(x_a - x_a^{(i)})\, p(x_b \mid x_a^{(i)})}{\sum_i w^{(i)} p(x_b \mid x_a^{(i)}) \cdot \int \delta(x_a - x_a^{(i)})\, dx_a} = \frac{1}{\sum_i w^{(i)} p(x_b \mid x_a^{(i)})} \cdot \sum_i w^{(i)} \delta(x_a - x_a^{(i)})\, p(x_b \mid x_a^{(i)})$$

$\int \delta(\cdot) = 1$

$\Vert$ NORMALIZER (η)

$$\Downarrow$$

only affect the weights

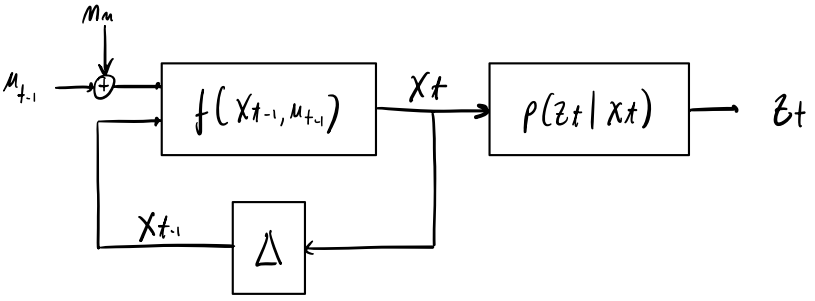
$$\Downarrow$$

CONDITIONING = multiply each weight by the conditional of the sample evaluated at the conditioning variable

$$w_{a|b}^{(i)} \propto w_a^{(i)} \cdot p\left(x_b \mid x_a^{(i)}\right)$$

Particle Filter

- Dynamic System



$x_t = f(x_{t-1}, u_{t-1})$ — TRANSITION FUNCTION

$p(x) \approx \sum_i w^{(i)} \delta(x - x^{(i)})$ — STATE PDF = SET OF WEIGHED SAMPLES

$n_m \sim p(n_t)$ — ADDITIVE NOISE DISTRIBUTED ACCORDING TO $p(n_m)$ WE CAN SAMPLE FROM

$p(z_t|x_t)$ — WE CAN EVALUATE POINT-WISE THE OBSERVATION MODEL

1. Generate noise samples $\left(n_m^{(i)} \sim p(n_m)\right)$

2. Predict — particles over time start to spread over space (NOT NECESSARILY A BAD THING)

PRIOR: $p(x_{t-1|t-1}) \approx \sum_i w_{t-1|t-1}^{(i)} \cdot \delta(x_{t-1|t-1} - x_{t-1|t-1}^{(i)})$

a) Transform each sample with its noise from motion model: $p(x_{t|t-1}) \approx \sum_i w_{t-1|t-1}^{(i)} \cdot \delta(x_{t|t-1} - f(x_{t|t-1}^{(i)}, u_{t-1} + n_m^{(i)}))$

- $x_{t|t-1}^{(i)} = f(x_{t-1|t-1}^{(i)}, u_{t-1} + n_m^{(i)})$ — samples of PDF after prediction
- $w_{t|t-1}^{(i)} = w_{t-1|t-1}^{(i)}$ — weights of PDF after prediction (UNCHANGED)

3. Update

a) Data association for each particle

b) Update particles: $p(x_{t|t}) \approx \sum_i w_{t|t-1}^{(i)} \cdot p(z_t|x_t^{(i)}) \cdot \delta(x_{t|t-1} - x_{t|t-1}^{(i)})$ (CONDITIONED) $\equiv$ Weights update by multiplying by the measurement likelihood

- $w_{t|t}^{(i)} = w_{t|t-1}^{(i)} \cdot p(z_t|x_{t|t-1}^{(i)})$
- $x_{t|t}^{(i)} = x_{t-1}^{(i)}$ — samples after update (UNCHANGED)

4. Resampling — to focus computation on likely regions of the state space

(⊕ normalization that is also carried on by resampling)

FINDING NEIGHBORS

generic formulation
(not necessarily Euclidean distances)

NEAREST SEARCH

- collection of vectors $P = \{p_m\}_{m=1:N}$, $p_m \in \mathbb{R}^K$
- query vector $p_q \in \mathbb{R}^K$
- distance metric $d(p_m, p_q) \in \mathbb{R}^+$

$\Rightarrow$ GOAL:

- point closest to the query, according to a metric
$$p_i = \underset{p_m \in P}{\operatorname{argmin}}\ d(p_m, p_q)$$
OR
- points in the collection whose distance from the query is smaller than a value ϵ
$$P' = \{p_i \in P,\ d(p_q, p_i) < \epsilon\}$$

$\Downarrow$

EXAMPLES OF DISTANCE METRICS $\longrightarrow$

- Squared Norm: $\|p_i - p_j\|^2 = (p_i - p_j)^T \cdot (p_i - p_j)$ (EUCLIDEAN)
- Omega Norm: $\|p_i - p_j\|^2_\Omega = (p_i - p_j)^T \cdot \Omega \cdot (p_i - p_j)$ (MAHALANOBIS)
- Hamming Distance: # different bits

e.g.: HAMMING $(10\mathbf{0}1\,00, 10\,1\,000) = 2$

What about BRUTE FORCE SEARCH?

- compute distance metric between query point and each point in vector
- update the minimum along the way

$\Rightarrow$ USE AUXILIARY SEARCH STRUCTURES

$O(n \times cost\_distance\_metric)$ $\longrightarrow$ can be very expensive in multiple queries....

DISTANCE MAP

IF $\dim(\text{vectors}) < 3$ and spread in small region of space

pre-computation of GRID LOOKUP TABLE $\longrightarrow$ each cell of the grid contains
- distance from the closest point
- identity of the closest point

HOW TO COMPUTE THE DISTANCE MAP?

- modify version of Dijkstra algorithm
- each grid cell c stores $\begin{cases} d : \text{distance from nearest point} \\ parent : \text{pointer to the nearest pointer} \end{cases}$

$\Downarrow$

- initialize grid cells which correspond to the points in P as $\begin{cases} p.d = 0 \\ p.parent = p \end{cases}$ $\longrightarrow$ the closest point to the current cell is the cell itself!

1. expand each initialized grid cell p, e.g., taking the 8 neighbors
2. for each of these expanded cells c_i

$d_{p,c} = \text{distance}(p, c_i)$

IF (c_i.isEmpty())
- $c_i.d = d_{p,c}$
- $c_i.parent = p.parent$

ELSE IF ($c_i.d > d_{p,c}$)
- $c_i.d = d_{p,c}$
- $c_i.parent = p.parent$

END

3. continue iteratively for all the expanded cells

COMPUTATION

$\longrightarrow O(grid\_size \times \log(grid\_size))$

QUERY

direct access $\leftarrow O(1)$

$\Downarrow$

Good when:
- low dimensional points
- small grid
- tolerate small connection points

+

- 2D NAVIGATION
  - distance map for obstacle avoidance
  - initial cost function to path planning

$\Vert$

traversability evaluation $\rightarrow$

What about heuristics?

- GATING: expands up to a maximum distance
- BEST FRIENDS: construct a distance map also for the measurements and cross-check
- LOWELY BEST FRIENDS: looking up for a region of the map instead of a single point

BUT

IF cost(construct distance map) $>$ cost(brute force search)
- brute force is better ....

END

KD-TREES

What if points are K-dimensional, with K large?
(distance map does not scale well.....)

$\Longrightarrow$ **KD-TREE**

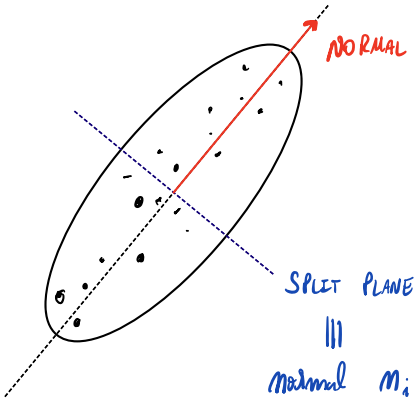
memory grows with K, so the time to construct it....

- search structure that partitions the database according to their spatial distribution
- if the tree is balanced, a search takes $\log(N)$ with N the number of points

$\Downarrow$

[constructing KD-trees can be done with a simple recursion
AT EACH TIME, set is split in two parts until #points for leaf is smaller than a threshold

How to Split?

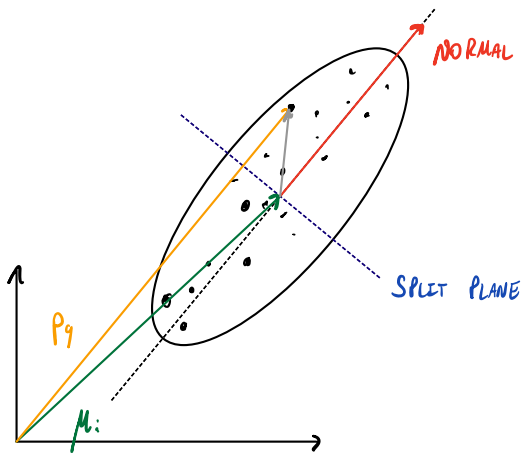


- consider as if K-dimensional pts are normally distributed
- compute covariance matrix
- choose splitting (hyper) plane that passes through the MEAN and has the NORMAL aligned with the longest / dominant axis of the covariance matrix

SPLIT PLANE
$\Updownarrow$
normal n_i
mean μ_i

→ check if a point lies on one side of the plane by evaluating:

$$n_i^\top (p_q - \mu_i) > 0 \quad (= \text{INNER PRODUCT})$$



↘ equivalent to say if

$$\cos\left(\sphericalangle(n_i, p_q - p_i)\right) > 0$$

Each intermediate node of the tree contains:
- normal of splitting plane
- mean
- pointers to children nodes

→ QUERY ≡ traversing tree from top to bottom and, at each time, go left or right

↓

IS AN APPROXIMATION !

(tree ~ heuristic that might return not the real minimum, depending on how it was built...)

↓

Use efficient randomized variants, e.g., ANN C++ Library

SPLITTING MIGHT NOT BE GOOD ⊕
- once we got to a branch of the tree no going back !

⇩ OTHER TOOLS

- PROJECTION TO LOWER DIMENSIONS
  - reduce dim (pts) (e.g., projection to 2D)
  - use some easy heuristic to perform data annotation in lower dimension space
- BAG OF WORDS (BoW)
  - used to determine the appearance similarity of multiple points
  - an item of the search collection contains many points

DIFFERENTIATION

Computation of the derivatives of complicated multi-variate functions may be complex...

↓

By hand ← tedious, error prone, requires lot of time

↳ NUMERICAL DIFFERENTIATION

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

↓ given a set of distinct from vectors... $\epsilon_1 = \begin{pmatrix} \epsilon \\ 0 \\ \vdots \\ 0 \end{pmatrix}$, $\epsilon_2 = \begin{pmatrix} 0 \\ \epsilon \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ (...) $\epsilon_n = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \epsilon \end{pmatrix}$

centered differentiation (↓ error)

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x_i + \epsilon_i) - f(x_i - \epsilon_i)}{2\epsilon}$$

↓

BUT, HOW TO SELECT ϵ?

- too small leads to machine precision errors
- too large poor derivative
- computations might be lowered

vs

- easy to implement
- most of the times works well
- can be used to check (round-) computed derivatives

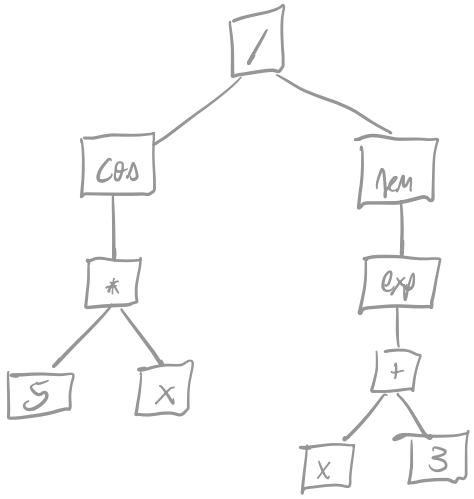
↳ AUTOMATIC DIFFERENTIATION → make the code compiler do it for us

⇓

PARSING TREE

$\cos(5 * x) / (\text{rem}(\exp(x+3)))$

- NODES: operation, constant, variable
- LEAF NODES: ALWAYS constants or variables
- ROOT & INTERMEDIATE: operations NODES



CHAIN RULE:

We want to evaluate the derivative of a nested function $f(g(x))$ in a point $\breve{x}$

⇒ FORMULA OF DERIVATIVE OF F: $\frac{\partial f(y)}{\partial y} = f'(y)$ → $\left.\frac{\partial f(g(x))}{\partial x}\right|_{x=\breve{x}} = f'(y) \cdot y'$

- Value of argument of f: $y = g(\breve{x})$
- Value of derivative of argument: $y' = g'(\breve{x})$

instead of having single float / double...

$\Rightarrow$ pairs (u, u') $\oplus$ redefine the operations in C++

↳ atoms
- constant : $[c, \emptyset]$
- variable : $[x, 1]$

↳ transcendental functions
- $\sin([u,u']) = [\sin(u), \cos(u)\cdot u']$
- $\cos([u,u']) = [\cos(u), -\sin(u)\cdot u']$
- $\exp([u,u']) = [\exp(u), \exp(u)\cdot u']$
- ....

↳ binary functions
- $[u,u'] + [v,v'] = [u+v, u'+v']$
- $[u,u'] - [v,v'] = [u-v, u'-v']$
- $[u,u'] * [v,v'] = [u*v, u'v + uv']$
- $[u,u'] / [v,v'] = [u/v, \frac{u'v - uv'}{v^2}]$

...