COMP417
Course review

McGill

MRL Mobile Robotics Lab at McGill University

Applications

To be discussed at the eif time permits

- Self-driving vehicles: cars, trucks, airplanes, boats, submarine
- Helping around the house
- Factory automation
- Warfare
- Scientific exploration
- Commercial exploration
- Medical applications
    - Surgery
    - Diagnosis
- Assistive robotics:
    - The elderly
    - The infirm
    - Everybody

Drawbacks of grid-based planners

- Grid-based planning works well for grids of up to 3-4 dimensions
- State-space discretization suffers from combinatorial explosion:
- If the state is $\mathbf{x} = [x_1, ..., x_D]$ and we split each dimension into N bins then we will have N^D nodes in the graph.
- This is not practical for planning paths for robot arms with multiple joints, or other high-dimensional systems.

(Sub)Sampling state-space

- Need to find ways to reduce the continuous domain into a sparse representation: graphs, trees etc.

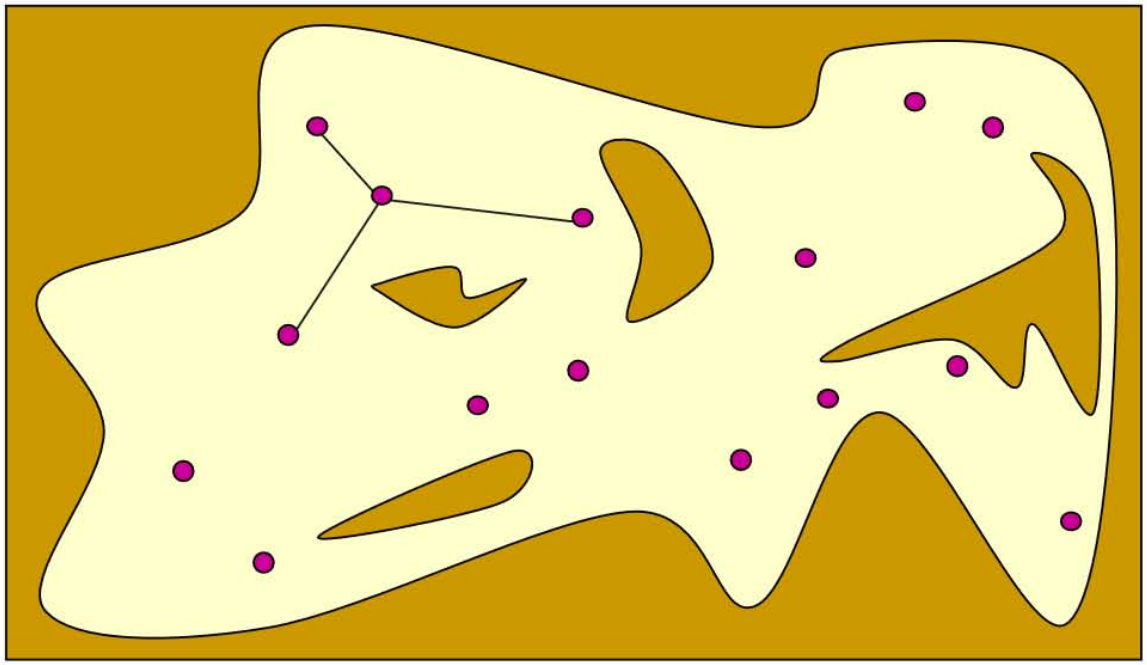
- Roadmaps:
  - Rapidly-exploring Random Tree (RRT),
  - Probabilistic RoadMap (PRM)
  - Visibility Planning
  - Smoothing Planned Paths

Probabilistic RoadMaps (PRMs)

- RRTs were good for single-query path planning
- You need to re-plan from scratch for every query A → B

- PRM addresses this problem
- It is good for multi-query path planning

PRM



Each node is connected to its neighbors (e.g. within a radius)

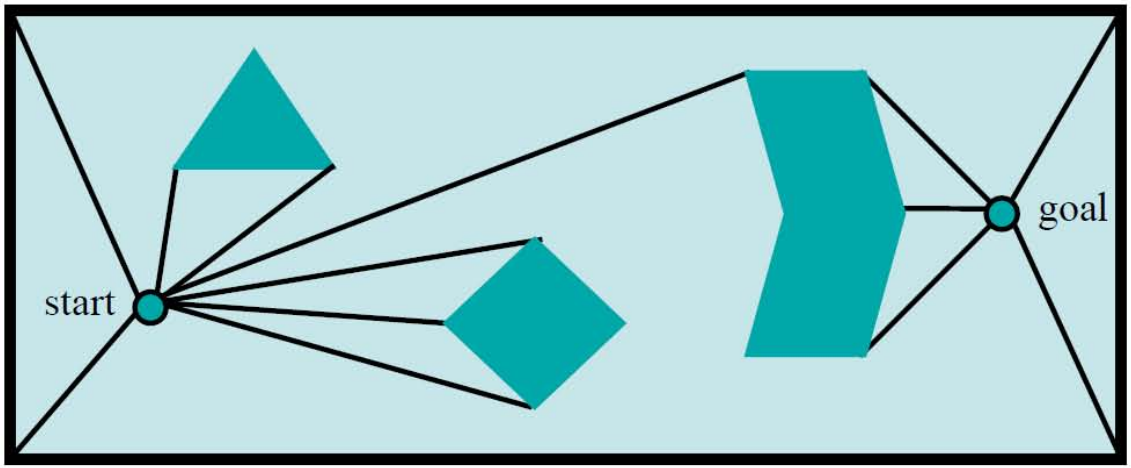
PRM

$V \leftarrow \{x_{\text{init}}\} \cup \{\texttt{SampleFree}_i\}_{i=1,\ldots,n}$; $E \leftarrow \emptyset$;
foreach $v \in V$ **do**
  $U \leftarrow \texttt{Near}(G = (V, E), v, r) \setminus \{v\}$;
  foreach $u \in U$ **do**
    if $\texttt{CollisionFree}(v, u)$ **then** $E \leftarrow E \cup \{(v, u), (u, v)\}$
return $G = (V, E)$;

To perform a query (A->B) we need to connect A and B to the PRM.
We can do this by nearest neighbor search (kd-trees, hashing etc.)

Visibility Graph Path Planning

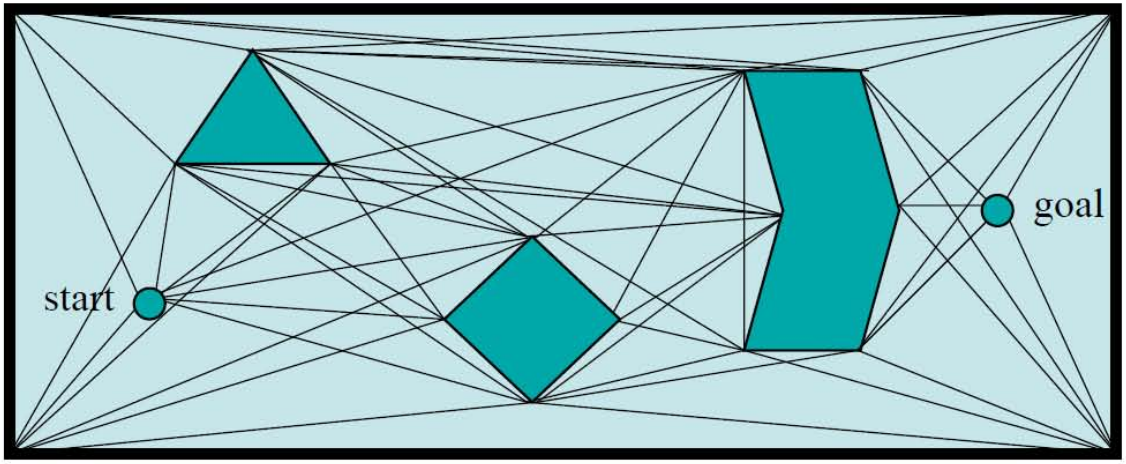
- First, draw lines of sight from the start and goal to all "visible" vertices and corners of the world.



Visibility Graph Path Planning

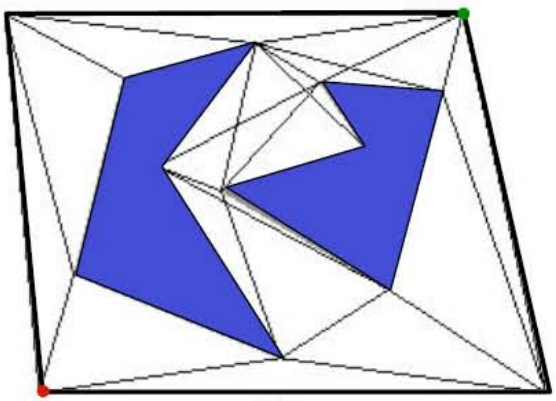
- Repeat until you're done.

Visibility graph

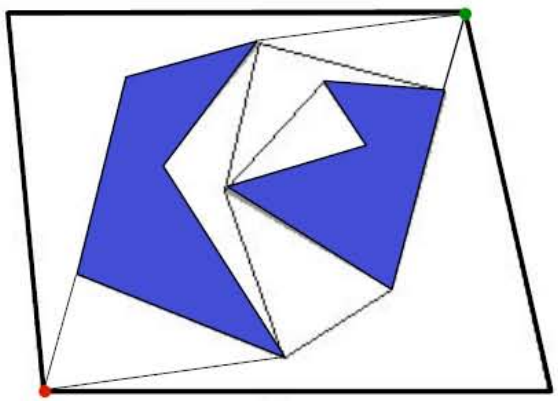


Can use graph search on visibility graph to find shortest path

Visibility Graph Path Planning



Full visibility graph



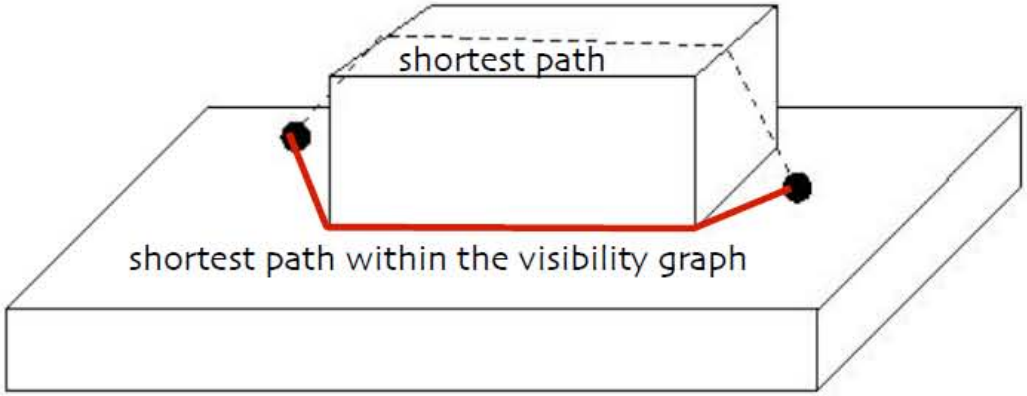
Reduced visibility graph, i.e., not including segments that extend into obstacles on either side.

(but keeping endpoints' roads)

Potential problem: shortest path touches obstacle corners. Need to dilate obstacles.

Visibility Graph Path Planning

Visibility graphs do not preserve their optimality in higher dimensions:



Path smoothing

- Plans obtained from any of these planners are not going to be smooth
- A plan is a sequence of states: $\pi = (\mathbf{x}_{\text{src}}, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N, \mathbf{x}_{\text{dest}})$

- We can get a smoother path $\text{smooth}(\pi) = (\mathbf{x}_{\text{src}}, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N, \mathbf{x}_{\text{dest}})$ by minimizing the following cost function

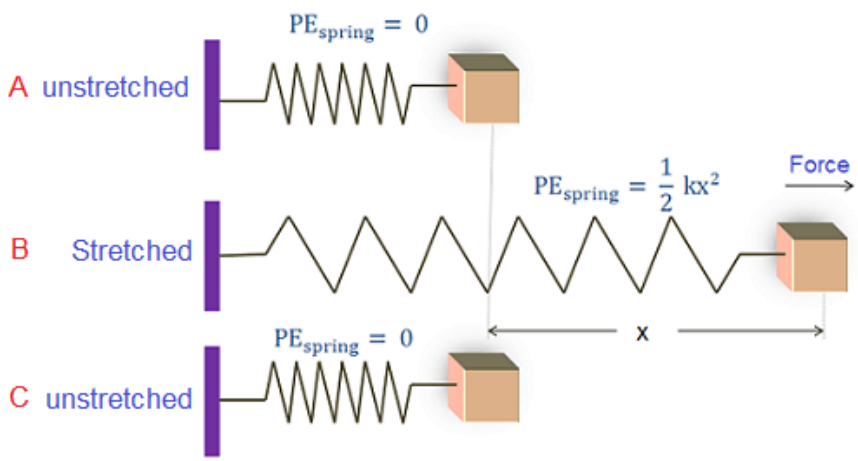
$$f(\mathbf{y}_1, \ldots, \mathbf{y}_N) = \sum_{t=1}^{N} ||\mathbf{y}_t - \mathbf{x}_t||^2 + \alpha \sum_{t=1}^{N} ||\mathbf{y}_t - \mathbf{y}_{t-1}||^2$$

Stay close to the old path

Penalize squared length

- May need to stop smoothing when smooth path comes close to obstacles.

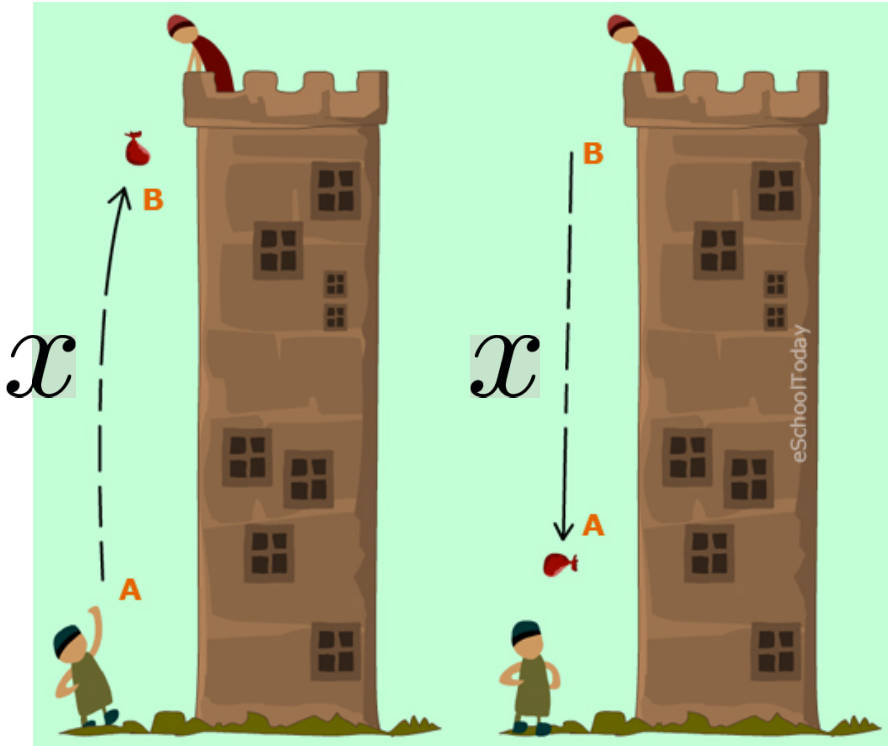
Background: potential energy and forces



$$U(x) = \frac{1}{2}kx^2$$

$$F(x) = -kx$$

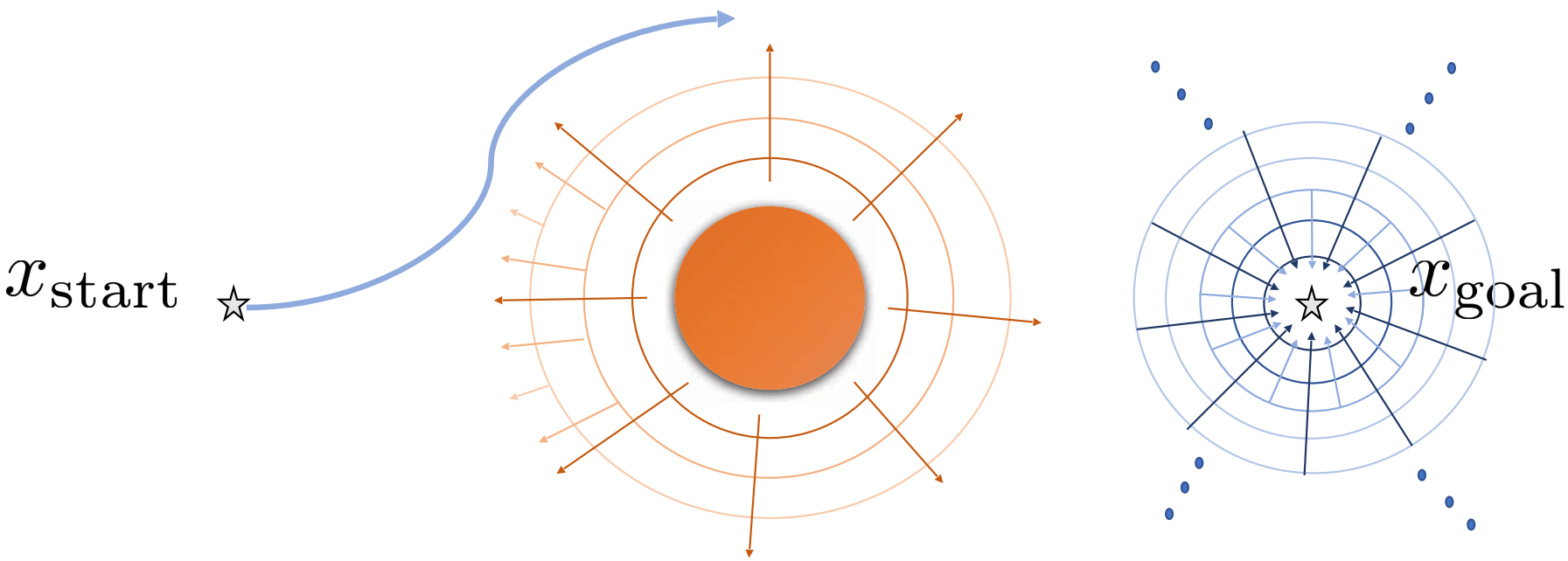
Background: potential energy and forces



$$U(x) = mgx$$

$$F(x) = -mg$$

Artificial Potential Fields



Q: How do you control the robot to reach the goal state while avoiding the obstacle?

A: Place a repulsive potential field around obstacles
and an attractive potential field around the goal

Combining Attractive and Repulsive Forces

Potential energy

$$U_{\text{total}}(x) = \alpha U_{\text{attractive}}(x) + \beta U_{\text{repulsive}}(x)$$

results in forces

$$F_{\text{total}}(x) = \alpha F_{\text{attractive}}(x) + \beta F_{\text{repulsive}}(x)$$

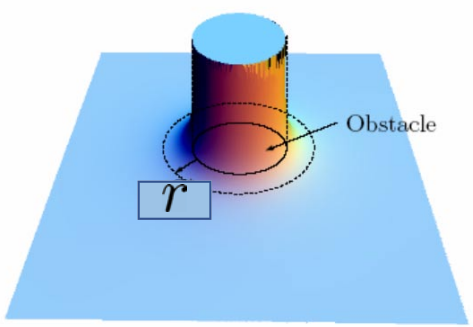
makes robot accelerate

$$\dot{x}_{t+1} = \dot{x}_t + \delta t F(x_t)$$

From Potential Fields to Forces

Make the robot move by applying forces resulting from potential fields

$$U_{\text{repulsive}}(x) = \begin{cases} \left(\frac{1}{d(x,\text{obs})} - \frac{1}{r}\right)^2 & \text{if} \quad d(x,\text{obs}) < r \\ 0 & \text{if} \quad d(x,\text{obs}) \geq r \end{cases}$$



$$F_{\text{repulsive}}(x) = \begin{cases} 2\left(\frac{1}{d(x,\text{obs})} - \frac{1}{r}\right)\frac{\nabla_x d(x,\text{obs})}{d(x,\text{obs})^2} & \text{if} \quad d(x,\text{obs}) < r \\ 0 & \text{otherwise} \end{cases}$$

Repulsive force makes state x go away from the obstacle
to lower potential energy states. Free space = {low-energy states}

Move the robot using F=ma, for m=1:

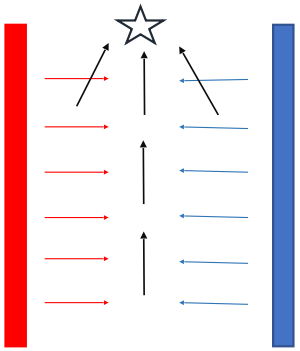
$$\dot{x}_{t+1} = \dot{x}_t + \delta t F(x_t)$$

Gradient descent until obstacle is cleared

Drawbacks of potential fields

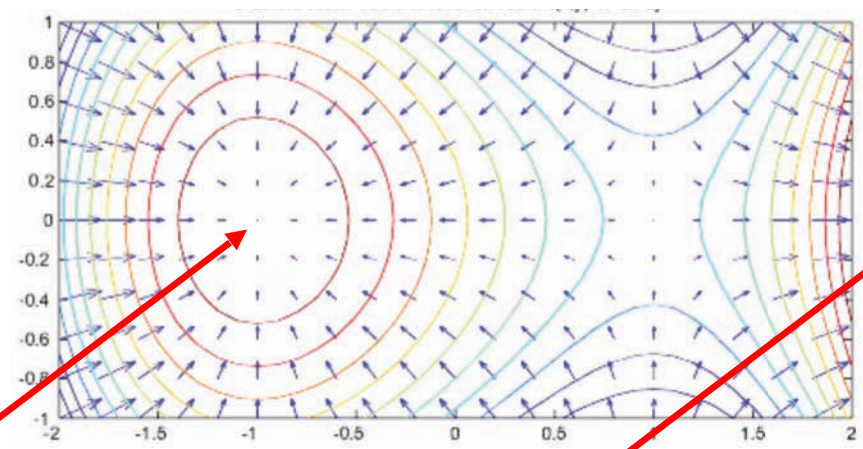
- *Local minima*
  - Attractive and repulsive forces can balance, so robot makes no progress.
  - Closely spaced obstacles, or dead end.

- *Unstable oscillation*
  - The dynamics of the robot/environment system can become unstable.
  - High speeds, narrow corridors, sudden changes



Local Minima on the Potential Field: Getting Stuck

States of zero total force correspond to local minima in the potential function:

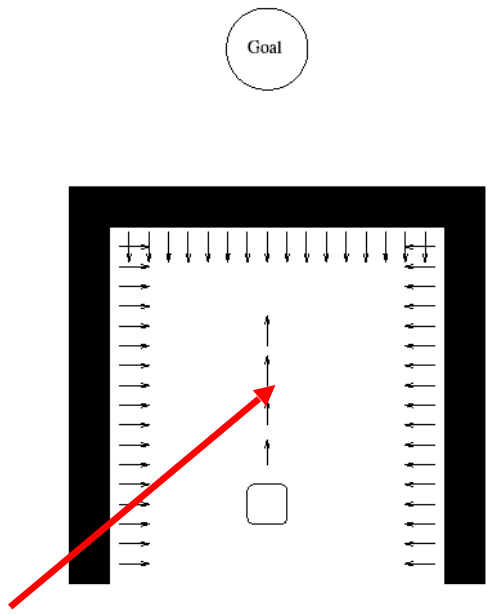


You start/end up here

Goal

Local Minima on the Potential Field: Getting Stuck

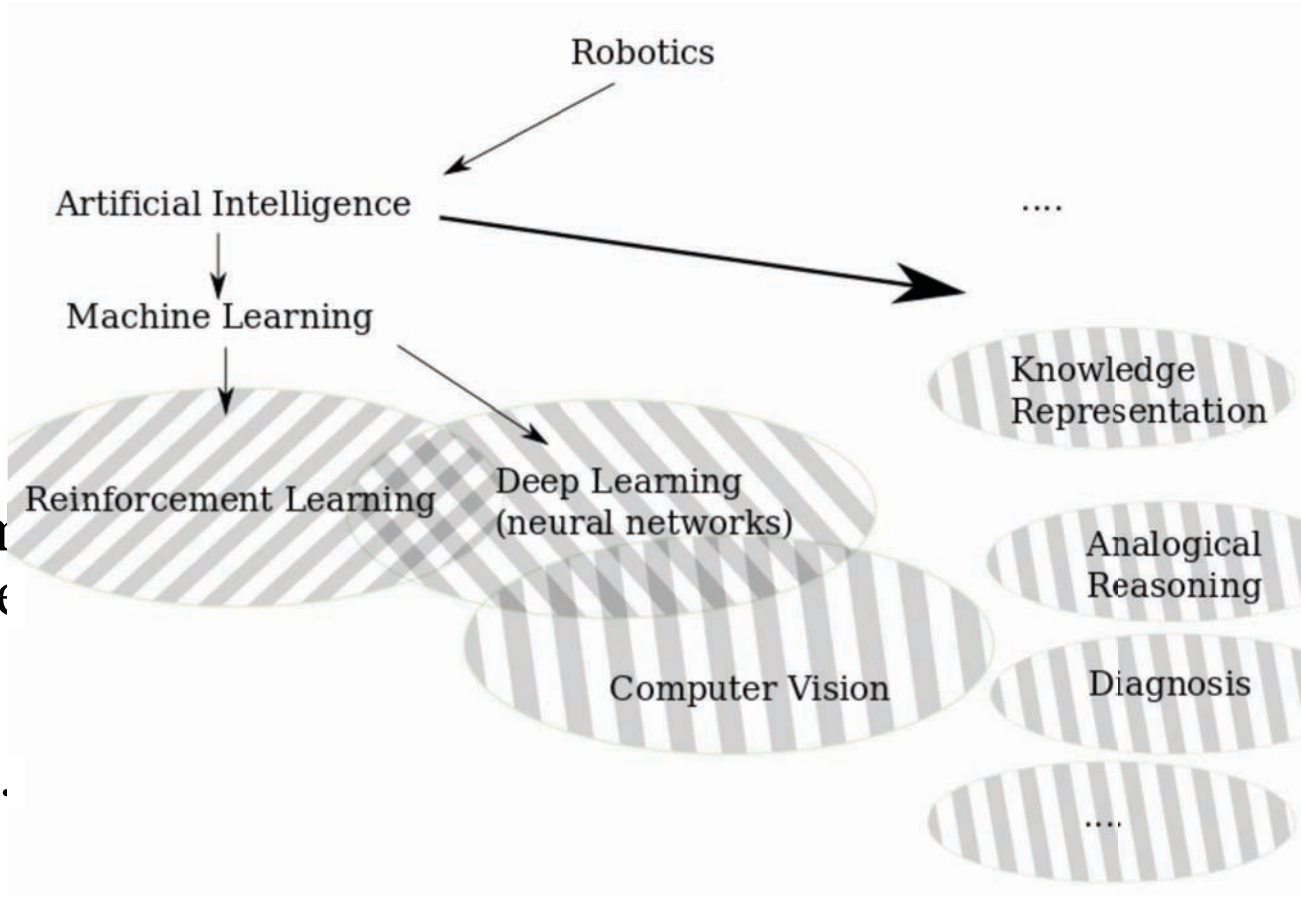
States of zero total force correspond to local minima in the potential function:



If you end up here gradient descent can't help you. All local moves seem identical in terms of value → local min

Reinforcement Learning

- A specific class of learning that is especially good for robotics.
- Captures way a long term plan produces eventual payoff.
- E.g. How a path on a roadn (only) eventually gets to the goal.
  - Deals with the issue of not knowing how to get "closer".



Markov Decision Processes (MDP)

Frameworks for optimal decision-making under uncertainty

Discrete states x_t
Discrete actions u_t

Uncertain dynamics $p(x_t|x_{t-1}, u_{t-1})$
State is directly observed at each step, without uncertainty

Agent receives instantaneous reward $R(x_t, u_t, x_{t+1}) = R_t$ after each transition.

Goal: find a policy $p(u_t|x_t) = \pi^*(u_t|x_t)$ that maximizes cumulative expected reward

Time horizon for optimization.
Can be infinite.

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \ \mathbb{E}[\sum_{t=0}^{H} \gamma^t R_t \mid \pi]$$

Discount factor in [0,1) specifies later rewards are less valued than earlier ones.

Partially-Observable Markov Decision Processes (POMDP)

Frameworks for optimal decision-making under uncertainty

Like MDP, but state is not directly observed. Instead, observation model $p(z_t|x_t)$

Agent knows action-observation history $h_t = (u_0, z_0, u_1, z_1, ..., u_t, z_t)$

Maintains and updates belief $b_t(x_t|h_t)$ using Bayes' filter

Receives instantaneous reward $R(x_t, u_t, x_{t+1}) = R_t$ after each transition, but when planning it can only weigh the reward by the probability of being at a state (since it does not observe it), so we define

$$r_t(h_t, u_t) = \sum_{x_t} b_t(x_t|h_t) R_t$$

Goal: find a policy $p(u_t|h_t) = \pi^*(u_t|h_t)$ that maximizes cumulative expected reward

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \ \mathbb{E}[\sum_{t=0}^{H} \gamma^t r_t \mid \pi, b_0]$$

With respect to future states
and future observations

Idea 1: bang-bang control

$$\omega = \begin{cases} \omega_{\max} & \text{if} \quad \text{CTE} > 0 \\ -\omega_{\max} & \text{if} \quad \text{CTE} < 0 \\ 0 & \end{cases}$$

What's wrong with this?

Idea 2: proportional (P-)control

$$\omega = K_p e(t)$$

Will the car reach the target line?

Will the car overshoot the target line?

Is the asymptotic (steady-state) error zero?

PID controller

$$\omega(t) = K_p e(t) + K_d \dot{e}(t) + K_i \int_{\tau=0}^{\tau=t} e(\tau) d\tau$$

Perhaps the most widely used controller in industry and robotics.
Perhaps the easiest to code.

You will also see it as:

$$\omega(t) = K_p [e(t) + T_d \dot{e}(t) + \frac{1}{T_i} \int_{\tau=0}^{\tau=t} e(\tau) d\tau]$$

Tips: how to implement PID

- Approximate the integral of error by a sum

- Approximate the derivative of error by:
  - Finite differences $\dot{e}(t_k) \approx \frac{e(t_k) - e(t_{k-1})}{\delta t}$
  - Filtered finite differences, e.g. $\dot{e}(t_k) \approx \alpha \dot{e}(t_{k-1}) + (1-\alpha)\frac{e(t_k) - e(t_{k-1})}{\delta t}$

- Limit the computed controls
- Limit or stop the integral term when detecting large errors and windup

Tips: how to tune the PID

- Ziegler-Nichols heuristic:
  - First, use only the proportional term. Set the other gains to zero.
  - When you see consistent oscillations, record the proportional gain K_u and the oscillation period T_u

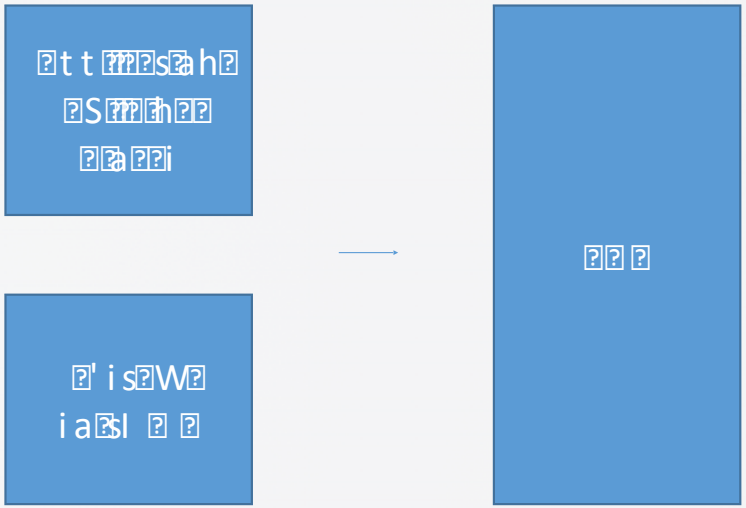
Ziegler–Nichols method[1]			
Control Type	K_p	T_i	T_d
P	$0.5K_u$	-	-
PI	$0.45K_u$	$T_u/1.2$	-
PD	$0.8K_u$	-	$T_u/8$
classic PID[2]	$0.6K_u$	$T_u/2$	$T_u/8$

[illegible]

- [illegible]
  - [illegible]
  - [illegible]
- [illegible]
  - [illegible]
- [illegible]
- [illegible]

[illegible]

- [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
- [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]

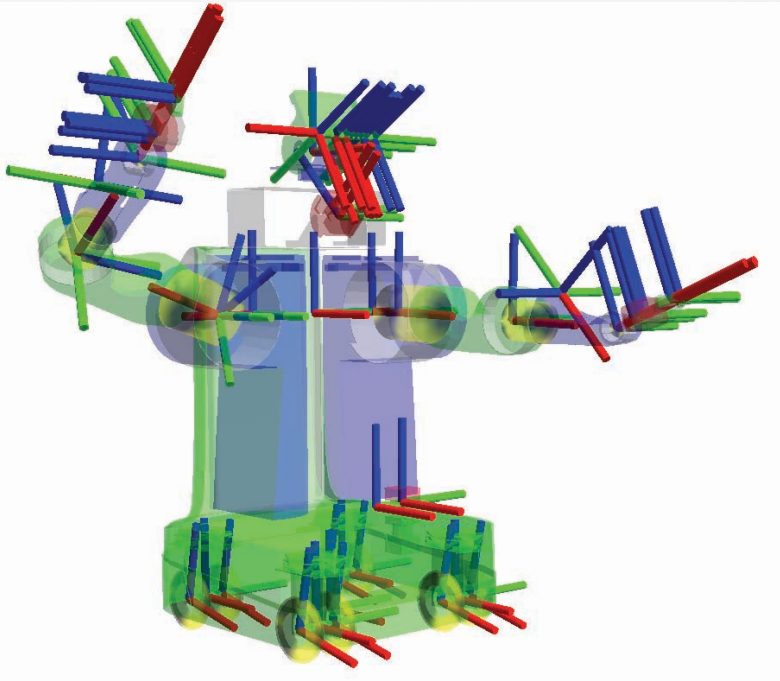


Types of sensors

- General classification:
  - contact vs. non-contact
  - active vs. passive
  - sampling rate: fast vs. slow
  - local vs. non-local

- General examples:
  - vision
  - laser
  - radar
  - sonar
  - compass, gyroscope, accelerometer
  - touch (tactile)
  - infrared

□□ □□ □hi□a W□s□a hi



- “TF” = Name of Transform package
  “Tully Foote” == Person/Developer

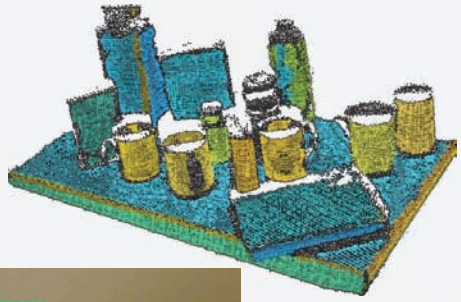
- TF Handles transforms between coordinate frames : space + time

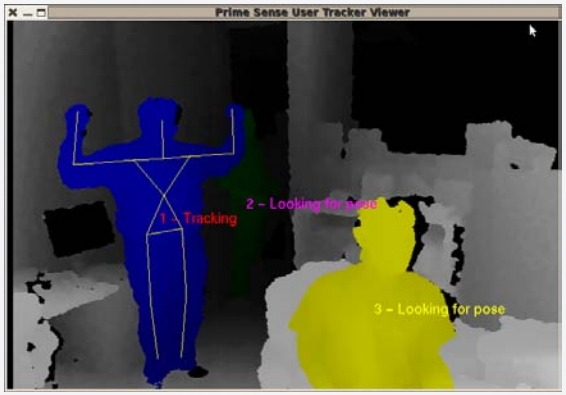
- tf_echo : print updated transforms in console

Example:
rosrun tf tf_echo [reference_frame] [target_frame]

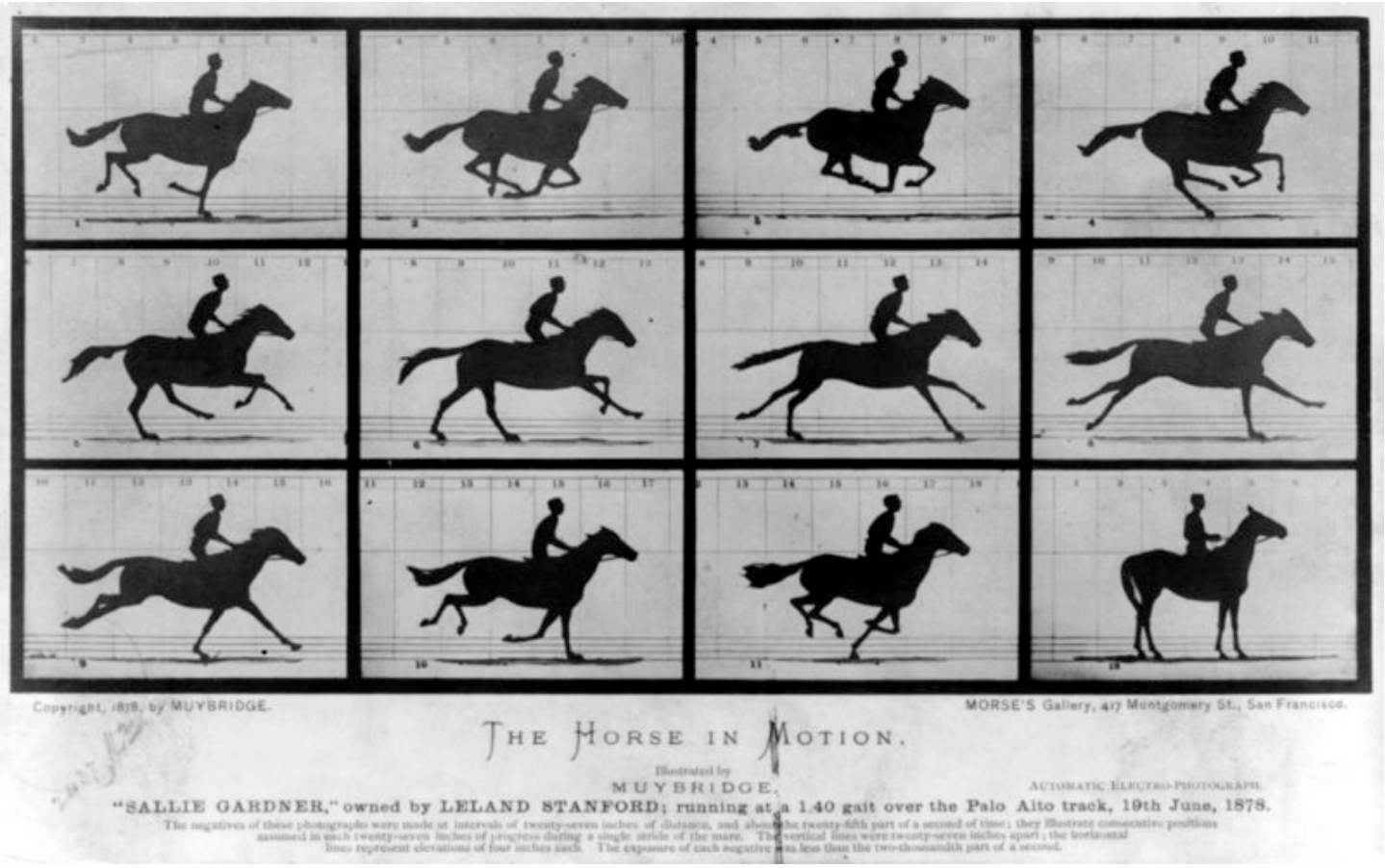
□□□□□□□i

- □□ □□t s□a h□
  - □a□h s□□□a S□□□□ □ ' □d□□□)
  - □ t □h□□□
  - □□h□□s(□ t □h□□





Legged Locomotion



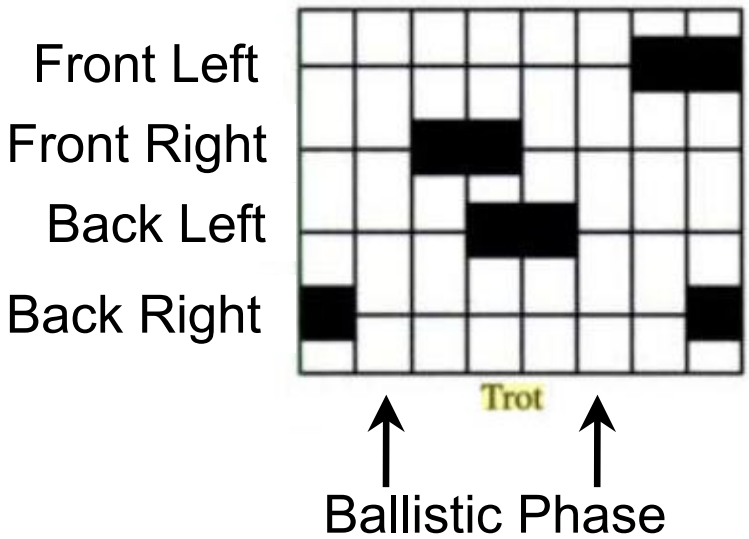
Stability Analysis

- Stability (in the sense of balance). Under what condition can we assure an object (vehicle, animal) will not fall over?

- Two classes of analysis (and behavior).
  - Static stability
  - Dynamic stability.

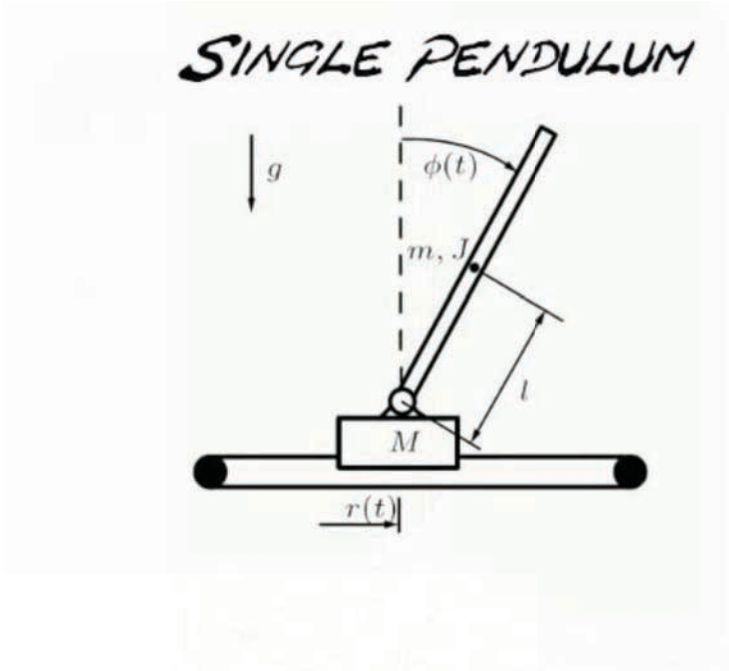
Hildebrand Gait Diagrams

Trot



Inverted pendulum

- The inverted pendulum uses the same equation, but with a minus sign in front of g relative to the above.
- Move base to maintain balance.
- Same idea to balance walking!



Dynamically Stable Gaits

- Robot is not always statically stable
- Must consider energy in limbs and body
- Much more complex to analyze
- E.G. Running:
  - Energy exchange:
    - Potential (ballistic)
    - Mechanical (compliance of springs/muscle)
    - Kinetic (impact)

Zero momentum point walk

- Force over foot modeled by reaction force at single point (ZMP): center of pressure.
- Point can be found by analysis or measurement.
- 1.Leg motion precomputed
- 2.Upper body motion chosen to keep ZMP within foot

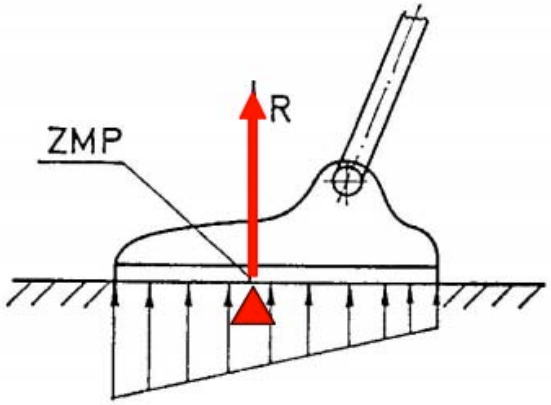


FIG. 1. Zero-moment point (ZMP).

Vukobratovic and Stepanenco,1972

Wagon (Kingpin) steering

- "Wagon steering": like the old covered wagons of yore.
- Two front wheels on common axle.

- Simple, but inefficient.
  - Small bumps effect steering angle

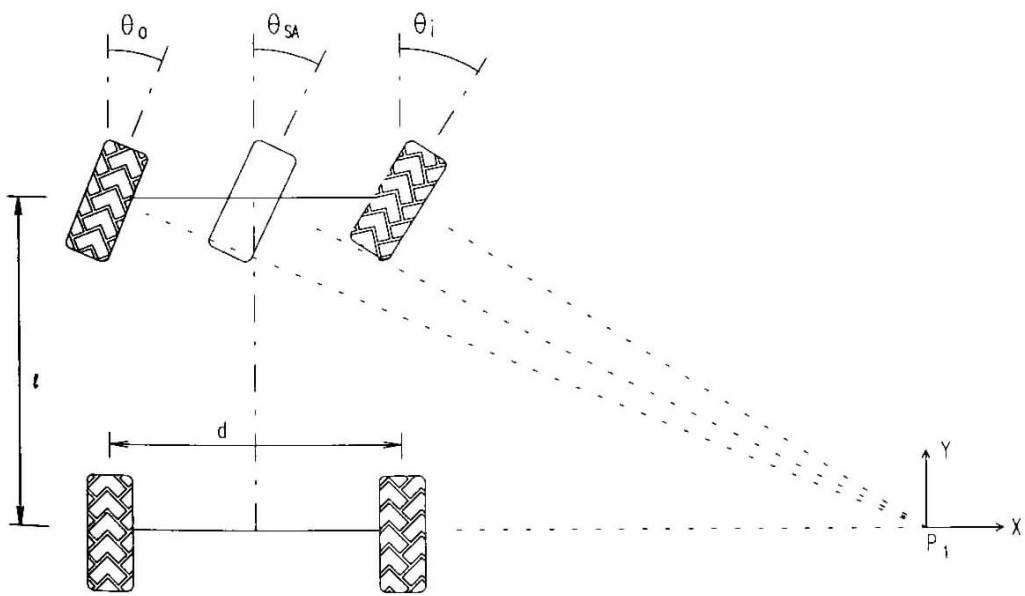
Ackerman Steering

- Car-like steering
  - "Double pivot"
- Original design: each wheel turns by same amount.
  - Different centers of curvature: leads to slip
    - Energy inefficient, hard to control (physically)
  - Jeantaud modification: slightly different steering angles.
- Front wheels turn by different amounts
- Back wheels do not turn
- ICC

$$\cot\theta_i - \cot\theta_o = \frac{d}{l}$$

where:

θ_i = relative steering angle of inner wheel
θ_o = relative steering angle of outer wheel
l = longitudinal wheel separation
d = lateral wheel separation.



COMP417
Introduction to Robotics and Intelligent Systems

Sensors and Actuators

McGill

MRL Mobile Robotics Lab at McGill University

Sensors: general characteristics

- Sensitivity: (change of output) ÷ (change of input)
- Linearity: constancy of (output ÷ input)
- Measurement range: [min, max] or {min, max}
- Response time: time required for input change to cause output change
- Accuracy: difference between measurement and actual
- Repeatability/Drift: difference between repeated measures
- Resolution: smallest observable increment
- Bandwidth: required rate of data transfer
- SNR: signal-to-noise ratio

Gyroscopes

- Measure angular velocity in the body frame
- Often affected by noise and bias

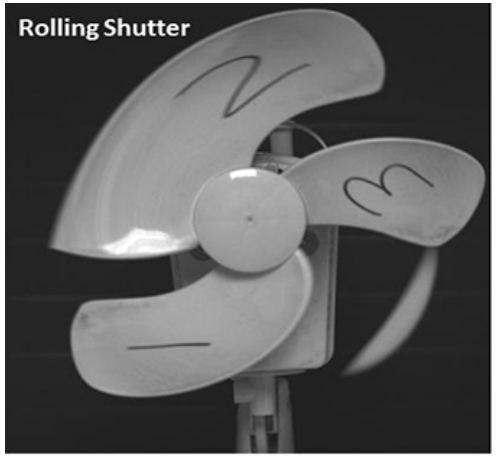
$$\omega_{\text{measured}}(t) = \omega_{\text{true}}(t) + b_g(t) + n_g(t)$$

- We integrate it to get 3D orientation (Euler angles, quaternions rotation matrices), but there is drift due to noise and bias

Cameras: Global vs. Rolling Shutter

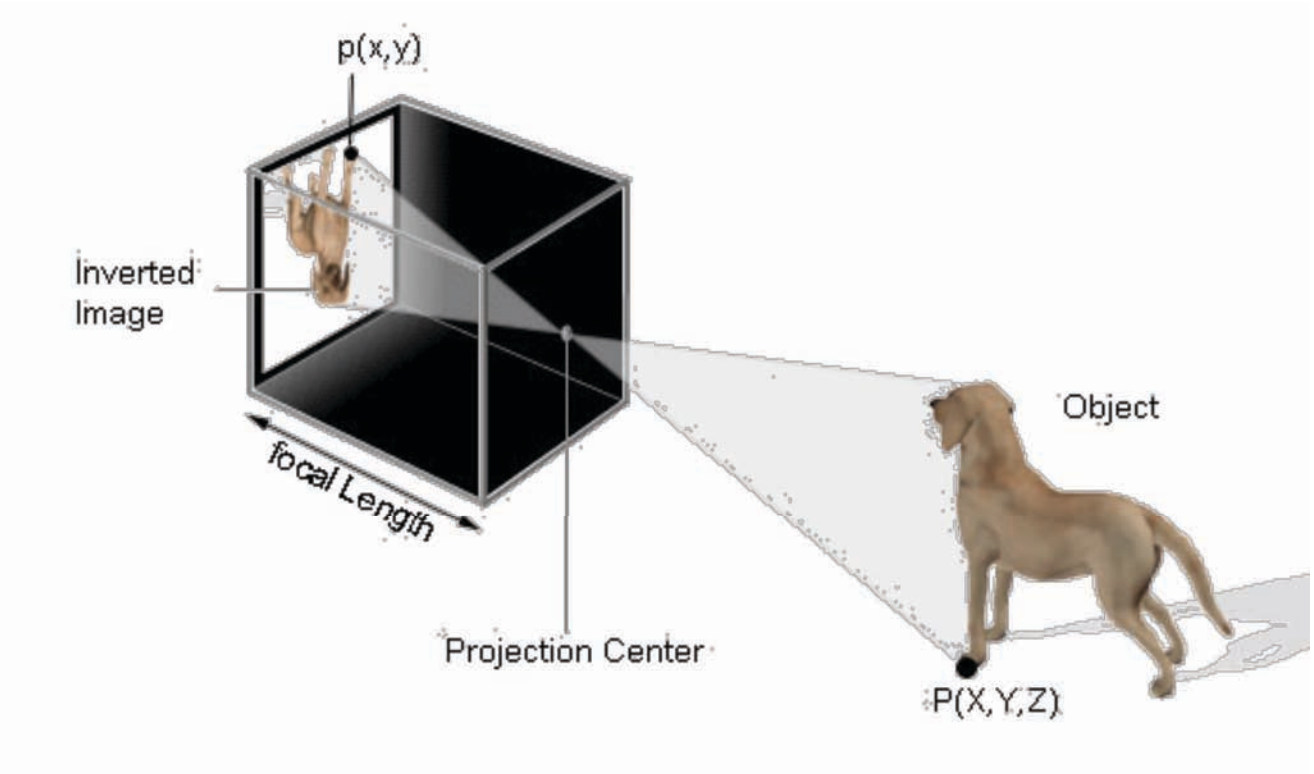
Shutter = mechanism that allows light to hit the imaging sensor

Shutter “speed” = Exposure time = time duration in which the sensor is exposed to light



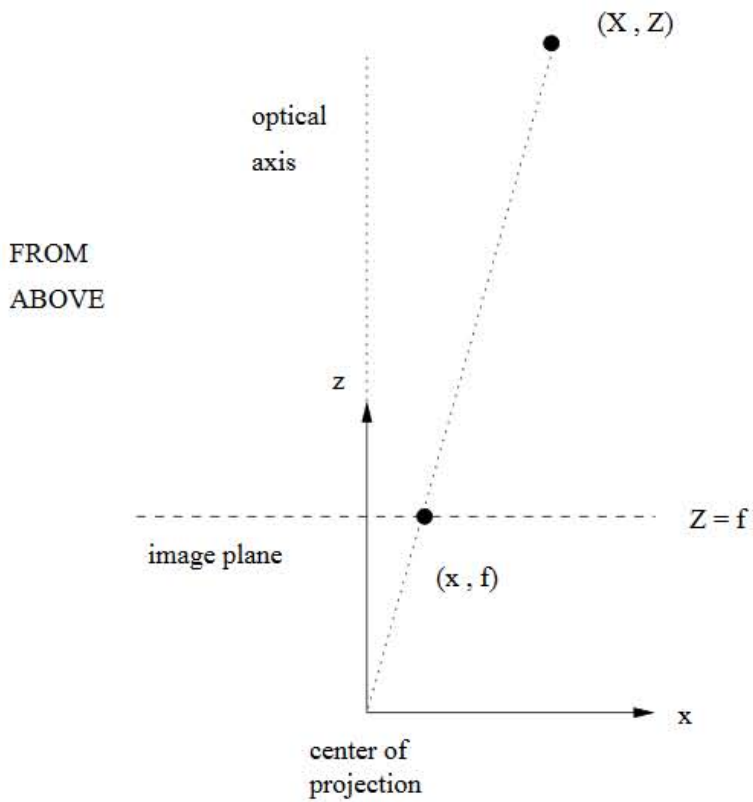
Rolling shutter

Pinhole Camera Model



We know **approximately** how a 3D point (X,Y,Z) projects to pixel (x,y)
We call this the ***pinhole projection model***

(1) Perspective projection
$[x,y] = \pi(X,Y,Z)$



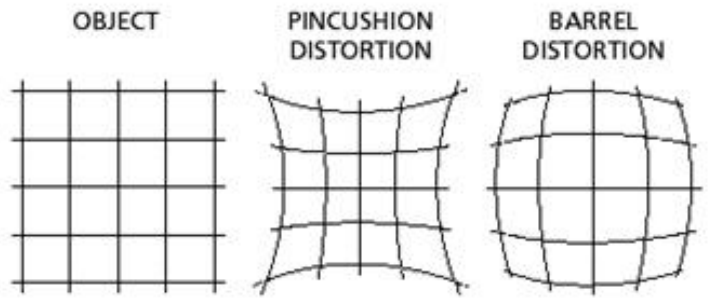
By similar triangles: $x/f = X/Z$

So, $x = f * X/Z$ and similarly $y = f * Y/Z$

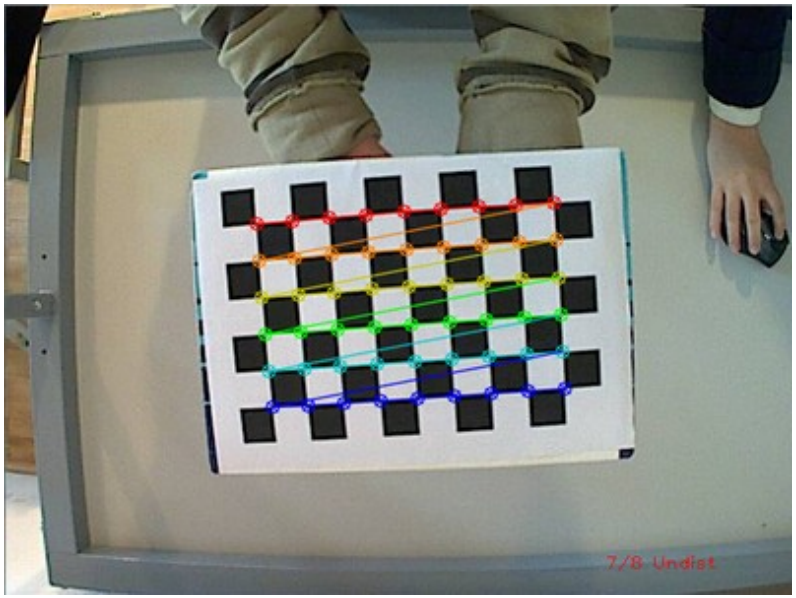
Problem: we just lost depth (Z) information by doing this projection, i.e. depth is now unknown.

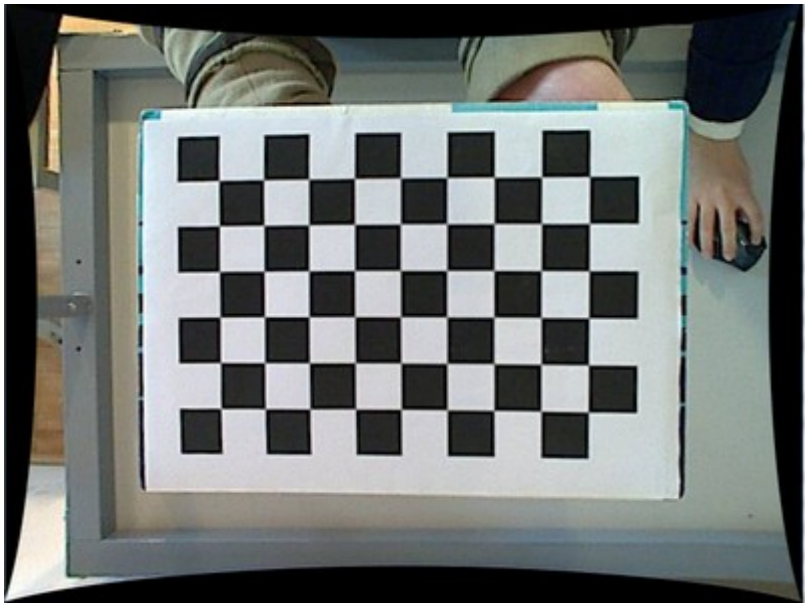
Going from 3D ->image: forward problem
Going from image -> 3D: inverse problem

(2) Lens distortion
$[x^*, y^*] = D(x,y)$



(2) Estimating parameters of lens distortion: [x*, y*] = D(x,y)





$$x^* = x\frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} \quad \text{where} \quad r = x^2 + y^2$$

Just an illustrative example:
you don't need to remember the details

Beyond the visible spectrum: RGBD cameras

Drawbacks:

- Does not work outdoors, sunlight saturates its measurements
- Maximum range is [0.5, 8] meters

Advantages:

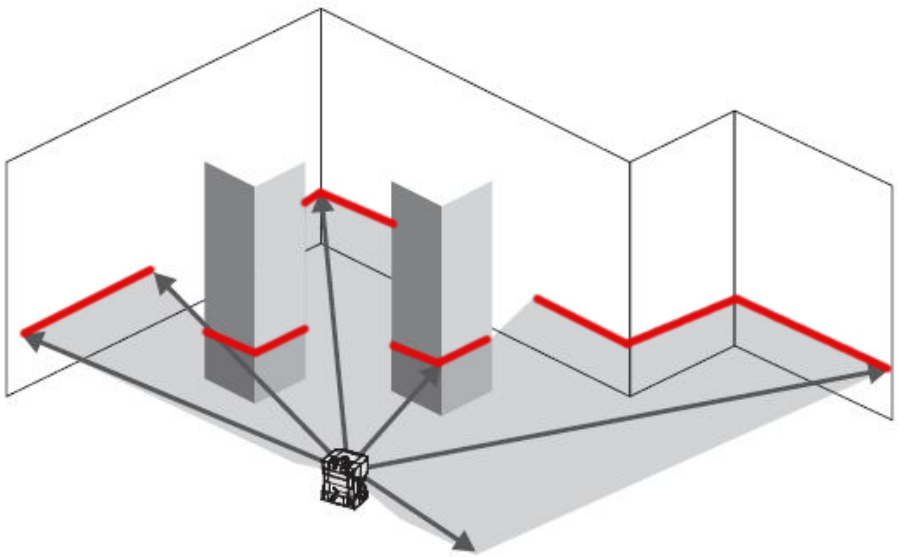
- Real-time depth estimation at 30Hz
- Cheap

2D LIDAR (Light detection and ranging)

Produces a scan of 2D points and intensities

- (x,y) in the laser's frame of reference
- Intensity is related to the material of the object that reflects the light

Certain surfaces are problematic for LIDAR: e.g. glass

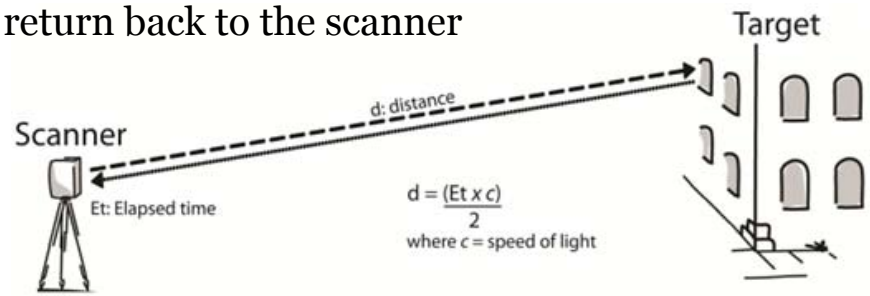


3D LIDAR (Light detection and ranging)

Produces a pointcloud of 3D points and intensities

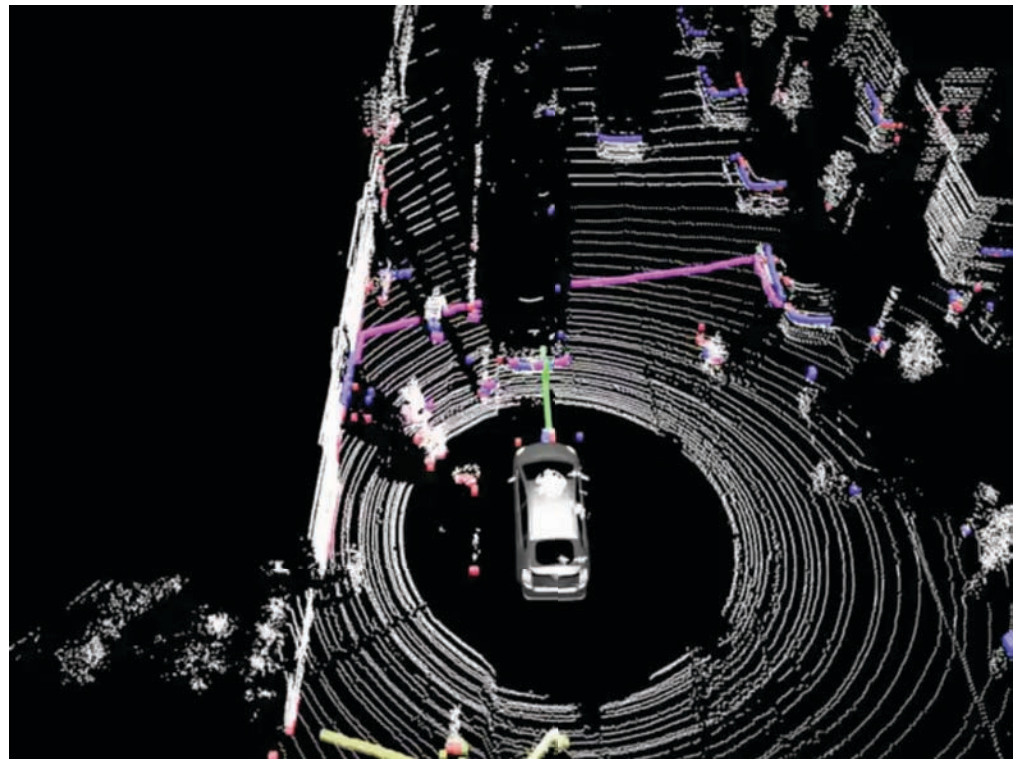
- (x,y,z) in the laser's frame of reference
- Intensity is related to the material of the object that reflects the light

Works based on time-of-flight for each beam to return back to the scanner



Not very robust to adverse weather conditions: rain, snow, smoke, fog etc.

Used in most self-driving cars today for obstacle detection. Range < 100m.



Usually around 1million points in a single pointcloud

RADAR

- Radar usually uses electromagnetic energy in the 1 - 12.5 GHz frequency range
  - this corresponds to wavelengths of 30 cm - 2 cm
    - microwave energy
  - unaffected by fog, rain, dust, haze and smoke
- It may use a pulsed time-of-flight methodology of sonar and lidar, but may also use other methods
  - continuous-wave phase detection
  - continuous-wave frequency modulation
- Continuous-wave systems make use of Doppler effect to measure relative velocity of the target

Inertial Sensors

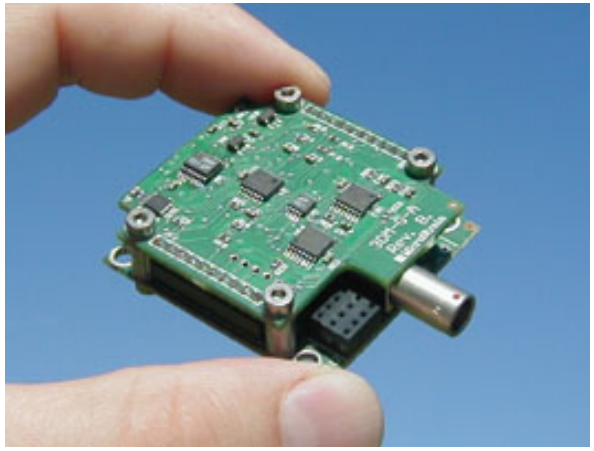
- Gyroscopes, Accelerometers, Magnetometers
- Inertial Measurement Unit (IMU)

- Perhaps the most important sensor for 3D navigation, along with the GPS

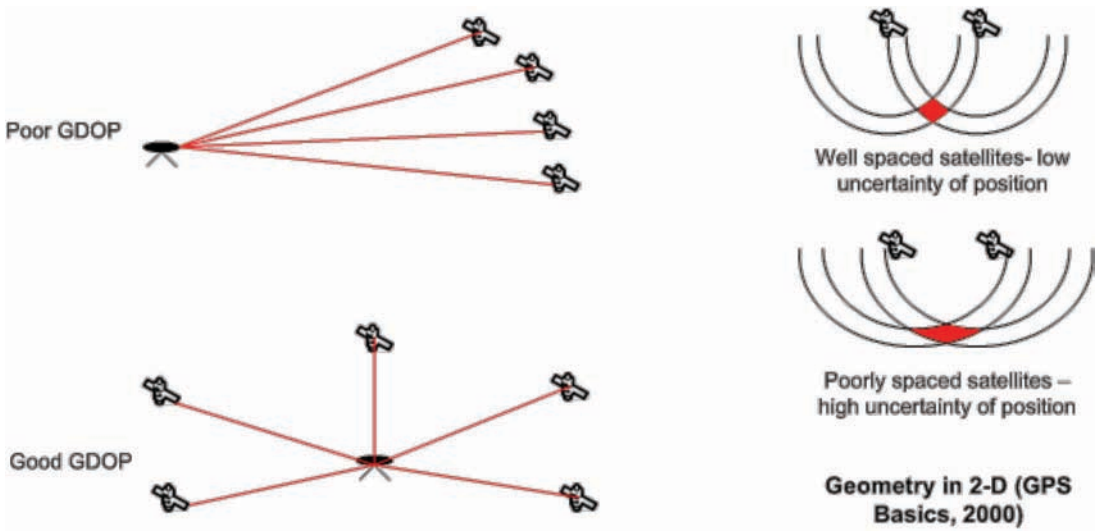
- Without IMUs, plane autopilots would be much harder, if not impossible, to build

IMU's

- Gyro, accelerometer combination.
- Typical designs (e.g. 3DM-GX1™) use triaxial gyros to track dynamic orientation and triaxial DC accelerometers along with the triaxial magnetometers to track static orientation.
- The embedded microprocessors contains a programmable filter algorithms, which blends these static and dynamic responses in real-time.



Global Positioning System: Receivers and Dilution of Precision



??hi a i

- ?????????????????????????

???s ??????????????????????a

  - ?? ?s?????????o ??(???a
  - ? ?????????????????

- ?????????????????????????

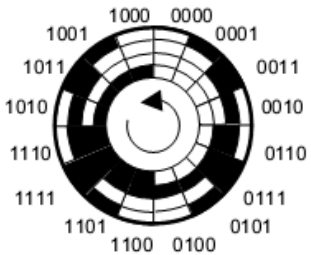
???s ??????????????????? ???a

  - ??? ?????s??????????????????? ??h? ?????t a
  - ?????????????????
  - ? ? ??????
  - ??????
  - ??????????t

Rotary Encoder

- Contains an analog to digital converter for encoding the angle of a shaft/motor/axle

- Usually outputs the discretized absolute angle of the sh motor/axle

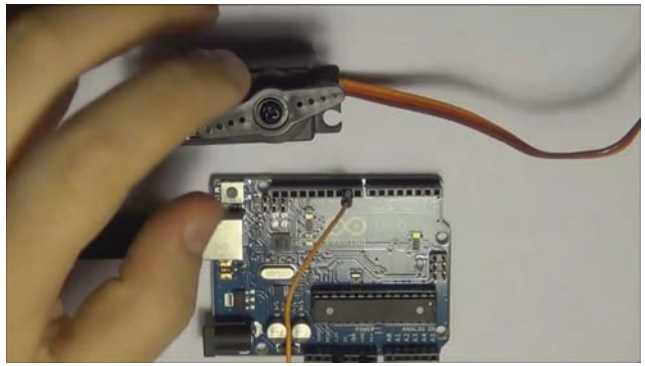


Gray code

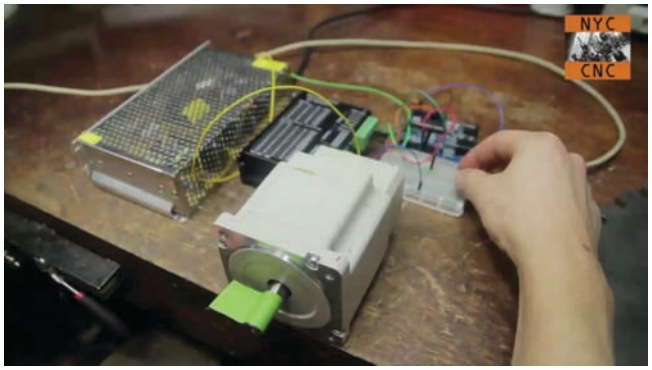
- Useful in order to know where different shafts are relative to each other.

Actuators

DC (direct current) motor
They turn continuously at high RPM (revolutions per minute) when voltage is applied. Used in quadrotors and planes, model cars etc.



Servo motor
Usually includes: DC motor, gears, control circuit, position feedback
Precise control without free rotation (e.g. robot arms, boat rudders)
Limited turning range: 180 degrees



Stepper motor
Positioning feedback and no positioning errors.
Rotates by a predefined step angle.
Requires external control circuit.
Precise control without free rotation.
Constant holding torque without powering the motor (good for robot arms or weight-carrying systems).

Connection between methods

- Sensors: measure the world
  with output depending on position $p(z|x)$
- Probabilistic (Bayes) filter: **State estimation.**
- Combine observations and prior beliefs (i.e. what we expect) to form current belief.
  - Only assumed a very abstract notion of maps, measurements, and beliefs: m, $z=h_m(x)$, $p(x|z,u)$
- SLAM:
  - combine current location estimate and measurement model to infer current position (e.g using Bayes filter)
  - invert measure model to infer map
- Global localization complexity MDL: select where to go next to most efficiently reduce uncertainty

- Occupancy grid: a specific form of map that is easy to use
  - General but not efficient
- Kalman filter: a specific state estimation filter that works well in practice
  - OK, it works really well if certain unrealistic assumptions hold
- Extended Kalman filter
  - More general, fewer assumptions, fewer guarantees

Defining the problem and example

- The occupancy grid map is a binary random variable

$$\mathbf{m} = \{m_{ij}\} \in \{0,1\}^{W \times H}$$

width = #columns
height = #rows
of the occupancy grid

- The path of the robot up to time t is a sequence of random variables $\mathbf{x}_{1:t} = \mathbf{x}_1, \ldots, \mathbf{x}_t$ with $\mathbf{x}_i = (x_i, y_i, \theta_i)$

Odometry coordinates

- At each time step the robot makes a measurement (sonar/laser). Measurements up to time t are a sequence of random variables $\mathbf{z}_{1:t} = \mathbf{z}_1, \ldots, \mathbf{z}_t$ with $\mathbf{z}_i = \{(r_i, \psi_i)\}^K$

(range, angle) in the laser's local coordinates

K = #beams, or #points in the scan

Evaluating the occupancy of a map cell

- How do we evaluate $p(m_{ij} = 1 | \mathbf{z}_{1:t}, \mathbf{x}_{1:t})$?
- Using conditional Bayes' rule we get

$$p(m_{ij} = 1 | \mathbf{z}_{1:t}, \mathbf{x}_{1:t}) = \frac{p(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t}, m_{ij} = 1) p(m_{ij} = 1 | \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t})}{p(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t})}$$

Current measurement only depends on current state and map cell

Current state without current measurement provides no additional information about the occupancy of the map cell

- And we simplify

$$\boxed{p(m_{ij} = 1 | \mathbf{z}_{1:t}, \mathbf{x}_{1:t})} = \frac{p(\mathbf{z}_t | \mathbf{x}_t, m_{ij} = 1) \boxed{p(m_{ij} = 1 | \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1})}}{p(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t})}$$

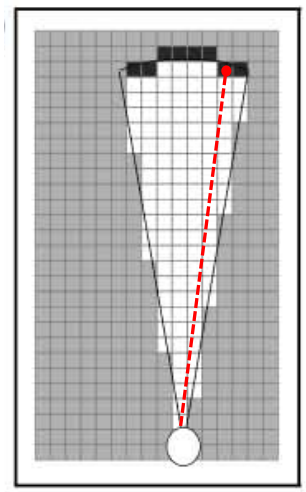
Note the similarity

The log-odds trick for binary random variables

- We showed $belief_t(m_{ij} = 1) = \eta\, p(\mathbf{z}_t|\mathbf{x}_t, m_{ij} = 1)\, belief_{t-1}(m_{ij} = 1)$ (1)
- Define the log odds ratio $l_t^{(ij)} = \log \frac{p(m_{ij} = 1|\mathbf{z}_{1:t}, \mathbf{x}_{1:t})}{p(m_{ij} = 0|\mathbf{z}_{1:t}, \mathbf{x}_{1:t})} = \log \frac{belief_t(m_{ij} = 1)}{belief_t(m_{ij} = 0)}$
- Then (1) becomes $l_t^{(ij)} = \log \frac{p(\mathbf{z}_t|\mathbf{x}_t, m_{ij} = 1)}{p(\mathbf{z}_t|\mathbf{x}_t, m_{ij} = 0)} + l_{t-1}^{(ij)}$
- We can recover the original belief as

$$belief_t(m_{ij} = 1) = 1 - \frac{1}{1 + \exp(l_t^{(ij)})}$$

Inverse sensor measurement model



$p(m_{ij} = 1 | \mathbf{z}_t, \mathbf{x}_t)$

Given map cell (i, j), the robot's state $\mathbf{x} = (x, y, \theta)$, and beams $\mathbf{z} = \{(r_k, \psi_k)\}$

Find index k of sensor beam that is closest in heading to the cell (i, j)

If the cell (i, j) is sufficiently farther than r_k or out of the field of view

// We don't have enough information to decide whether cell is occupied

Return prior occupation probability $p(m_{ij} = 1)$

If the cell (i, j) is nearly as far as the measurement r_k

// Cell is most likely occupied

Return p_{occupied} that is well above 0.5

If the cell (i, j) is sufficiently closer than r_k

// Cell is most likely free

Return p_{occupied} that is well below 0.5

[illegible]a[illegible]Rah

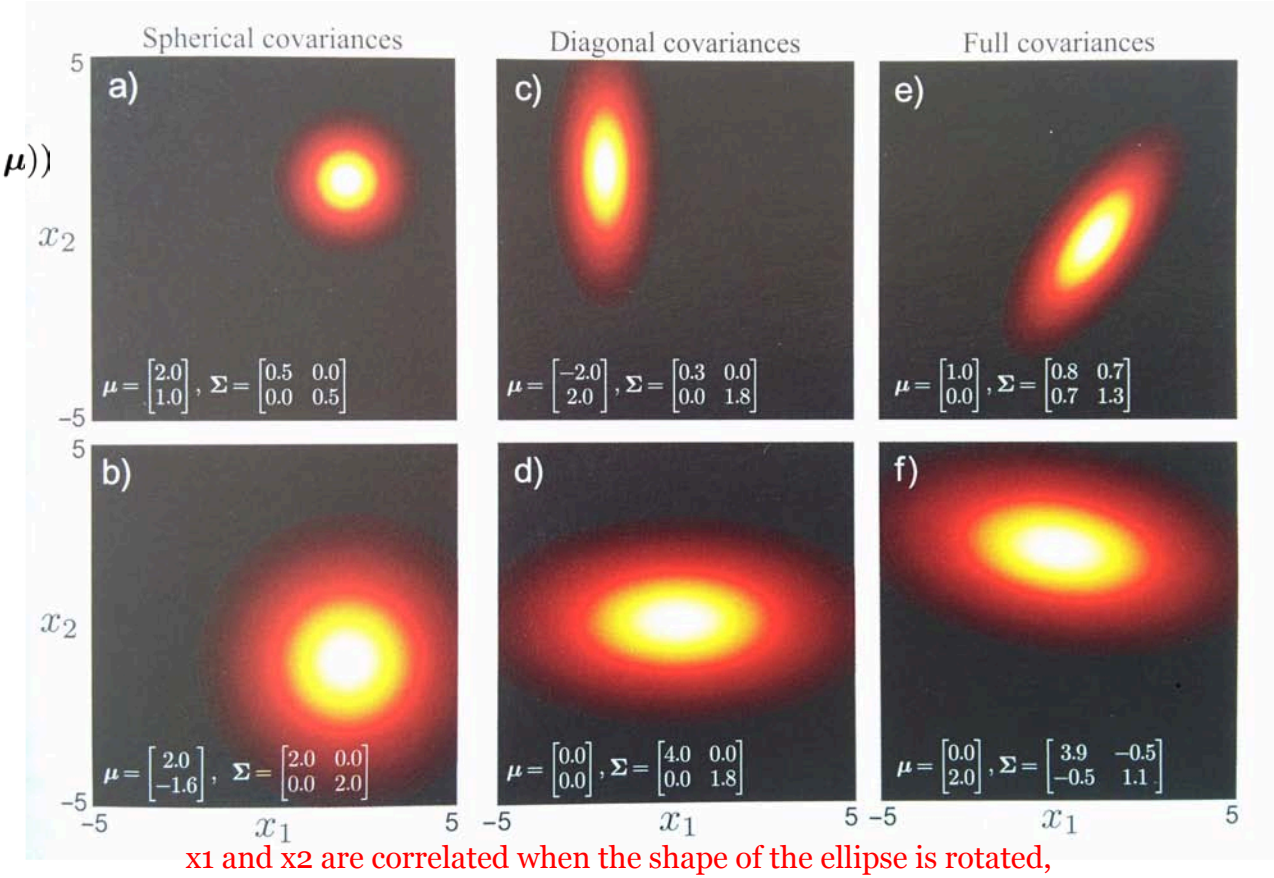
- [illegible]
- [illegible]
- [illegible]
  - s[illegible] a

Background: Multivariate Gaussian Distribution

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}\det(\boldsymbol{\Sigma})^{1/2}}\exp(-0.5(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}\det(\boldsymbol{\Sigma})^{1/2}}\exp(-0.5||\mathbf{x}-\boldsymbol{\mu}||^2_{\boldsymbol{\Sigma}})$$

Shortcut notation: $||\mathbf{x}||^2_{\boldsymbol{\Sigma}} = \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}$



x1 and x2 are correlated when the shape of the ellipse is rotated,
i.e. when there are nonzero off-diagonal terms in the covariance matrix.
In this example, (e) and (f)

From "Computer Vision: Models, Learning, and Inference" Simon Prince

Confidence regions

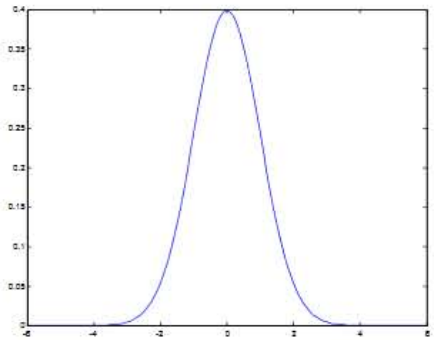
• To quantify confidence and uncertainty define a confidence region R about a point x (e.g. the mode) such that at a confidence level c $\leq$ 1

$$p(x \in R) = c$$

• we can then say (for example) there is a 99% probability that the true value is in R

• e.g. for a univariate normal distribution N(μ,σ^2)

$$\begin{aligned} p(|x-\mu| < \sigma) &\approx 0.67 \\ p(|x-\mu| < 2\sigma) &\approx 0.95 \\ p(|x-\mu| < 3\sigma) &\approx 0.997 \end{aligned}$$



Expectation (mean)

- Expected value of a random variable X:

$$E_{p(X)}[X] = \int_x xp(X = x)dx$$

Probability-weighted average

- E is linear: $E_{p(X)}[X + c] = E_{p(X)}[X] + c$

$$E_{p(X)}[AX + b] = AE_{p(X)}[X] + b$$

If every student's grade goes up by 10, the class average goes up by 10

- If X,Y are independent then [Note: inverse does not hold]

$$E_{p(X,Y)}[XY] = E_{p(X)}[X]E_{p(Y)}[Y]$$

Covariance Matrix

- How two (or more) variables X and Y (random variables) are related. Measures linear dependence between random variables X, Y.
  Does **not** measure independence.

$$\mathrm{Cov}[X, Y] = E[XY] - E[X]E[Y]$$

- Variance of X

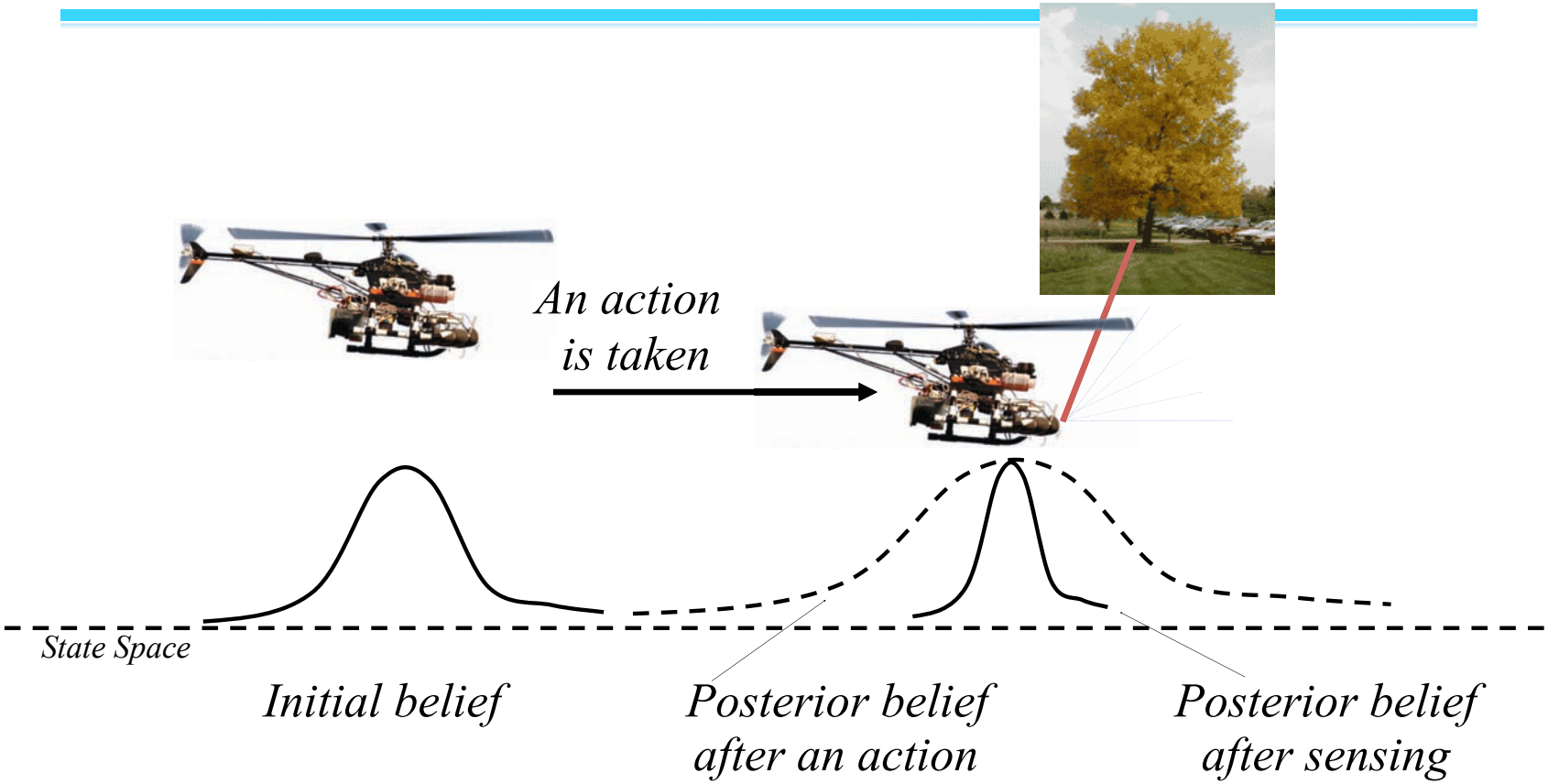
$$\mathrm{Var}[X] = \mathrm{Cov}[X] = \mathrm{Cov}[X, X] = E[X^2] - E[X]^2$$

Terminology: ***variance*** *of a random multi-dimensional variable is also known as* ***covariance*** *matrix.*

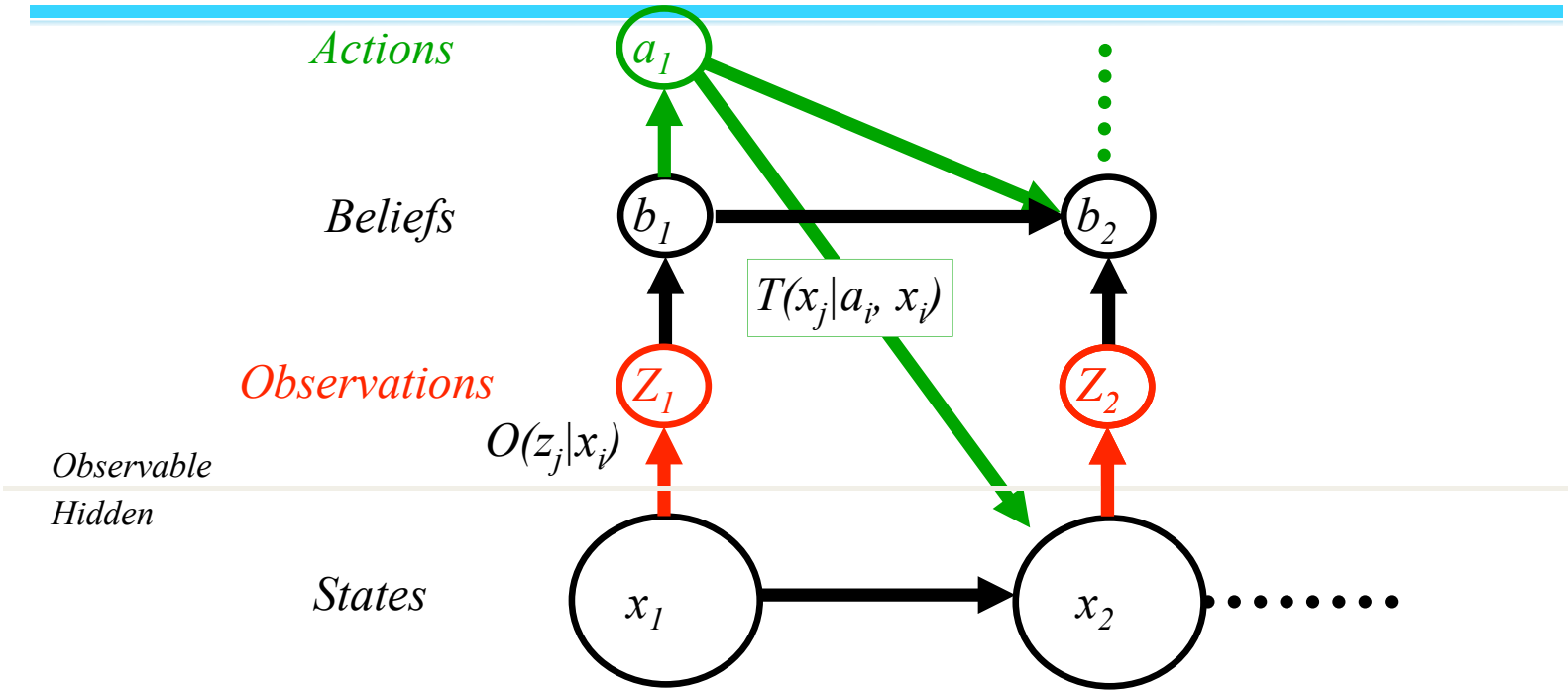
$$\mathrm{Cov}[AX + b] = A\mathrm{Cov}[X]A^T$$

$$\mathrm{Cov}[X + Y] = \mathrm{Cov}[X] + \mathrm{Cov}[Y] - 2\mathrm{Cov}[X, Y]$$

??' ?¡ ??????


An action
is taken
State Space
Initial belief
Posterior belief
after an action
Posterior belief
after sensing

[illegible]



Bayes rule : $p(x \mid y) = \dfrac{p(y \mid x)\, p(x)}{p(y)}$

Markov : $p(x_t \mid x_{t-1}, a_t, a_0, z_0, a_1, z_1, \ldots, z_{t-1}) = p(x_t \mid x_{t-1}, a_t)$

Bayes' Filter

- A generic class of filters that make use of Bayes' rule and assume the following:
  - **<u>Markov Assumption</u> For Dynamics**: the state x_t is conditionally independent of past states and controls, given the previous state x_{t-1}. In other words, the dynamics model is assumed to satisfy

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

  - **Static World Assumption:** the current observation is conditionally independent of past observations and controls, given the current state

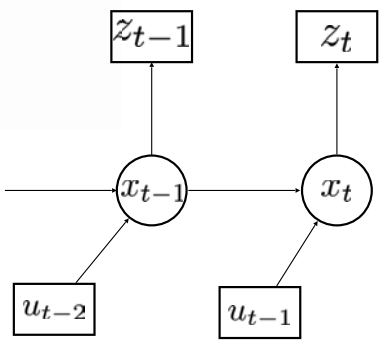
$$p(z_t | x_t, u_{0:t-1}, z_{0:t-1}) = p(z_t | x_t)$$

Note: the Markov assumption is more general than what we have presented here.

Bayes' Filter: Derivation

$$
\begin{aligned}
bel(x_t) &= p(x_t | u_{0:t-1}, z_{0:t}) \\
&= \eta\, p(z_t | x_t, u_{0:t-1}, z_{0:t-1})\; p(x_t | u_{0:t-1}, z_{0:t-1}) \\
&= \eta\, p(z_t | x_t)\; p(x_t | u_{0:t-1}, z_{0:t-1}) \\
&= \eta\, p(z_t | x_t) \int p(x_t, x_{t-1} | u_{0:t-1}, z_{0:t-1})\; dx_{t-1} \\
&= \eta\, p(z_t | x_t) \int p(x_t | u_{0:t-1}, z_{0:t-1}, x_{t-1})\; p(x_{t-1} | z_{0:t-1}, u_{0:t-1})\; dx_{t-1} \\
&= \eta\, p(z_t | x_t) \int p(x_t | u_{t-1}, x_{t-1})\; p(x_{t-1} | z_{0:t-1}, u_{0:t-1})\; dx_{t-1} \\
&= \eta\, p(z_t | x_t) \int p(x_t | u_{t-1}, x_{t-1})\; p(x_{t-1} | z_{0:t-1}, u_{0:t-2})\; dx_{t-1}
\end{aligned}
$$

This is the belief at the previous time step!
This means we can perform filtering recursively.



Control at time t-1 only affects state at time t

Bayes' Filter: Derivation

$$
\begin{aligned}
bel(x_t) &= p(x_t | u_{0:t-1}, z_{0:t}) \\
&= \underbrace{\eta \, p(z_t | x_t) \underbrace{\int p(x_t | u_{t-1}, x_{t-1}) \, bel(x_{t-1}) \, dx_{t-1}}_{\text{Belief after prediction step}}}_{\text{Belief after update step}}
\end{aligned}
$$

Kalman Filter: an instance of Bayes' Filter

$$
\begin{aligned}
bel(x_t) &= p(x_t | u_{0:t-1}, z_{0:t}) \\
&= \eta \, p(z_t | x_t) \int p(x_t | u_{t-1}, x_{t-1}) \, bel(x_{t-1}) \, dx_{t-1}
\end{aligned}
$$

Linear dynamics with Gaussian noise

$$
\begin{aligned}
x_t &= A x_{t-1} + B u_{t-1} + G w_{t-1} \\
&\quad \text{with noise } w_{t-1} \sim \mathcal{N}(0, Q)
\end{aligned}
$$

Linear observations with Gaussian noise

$$
\begin{aligned}
z_t &= H x_t + n_t \\
&\quad \text{with noise } n_t \sim \mathcal{N}(0, R)
\end{aligned}
$$

✚ Initial belief is Gaussian

$bel(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$

Kalman Filter

Linear dynamics with Gaussian noise

$$\begin{aligned} x_t &= Ax_{t-1} + Bu_{t-1} + Gw_{t-1} \\ &\quad \text{with noise } w_{t-1} \sim \mathcal{N}(0, Q) \end{aligned}$$

Linear observations with Gaussian noise

$$\begin{aligned} z_t &= Hx_t + n_t \\ &\quad \text{with noise } n_t \sim \mathcal{N}(0, R) \end{aligned}$$

Initial belief is Gaussian

$$bel(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$$

Kalman Filter with 1D state: the update step

Oct 26 end

Take-home message: new observations, no matter how noisy, always **reduce uncertainty** in the posterior. The mean of the posterior, on the other hand, only changes when there is a nonzero prediction residual.

Simple version: **extra data is always useful, no matter how unreliable the source.**

Controllers can be designed with full state knowledge:

$$\mathbf{x}_{t+1} = A\mathbf{x} + B\mathbf{u}$$

$$\mathbf{z} = C\mathbf{x}$$

Observers can be designed to **estimate** the full state from the output:

$$\hat{\mathbf{x}}_{\mathbf{t+1}} = A\hat{\mathbf{x}} + B\mathbf{u} + \mathbf{K}(\mathbf{z} - C\hat{\mathbf{x}}_{t+1})$$

Innovation term.
Why not just use $C^{-1}z$ to get an estimate of x?
Because it might not be invertible.

The controller and observer can be designed independently!

Fundamental Theorem of Estimation Theory -> Kalman filter optimality

- The minimum mean square error estimator equals the expected (mean) value of x conditioned on the observations Z
- The minimum mean square error term is quadratic:

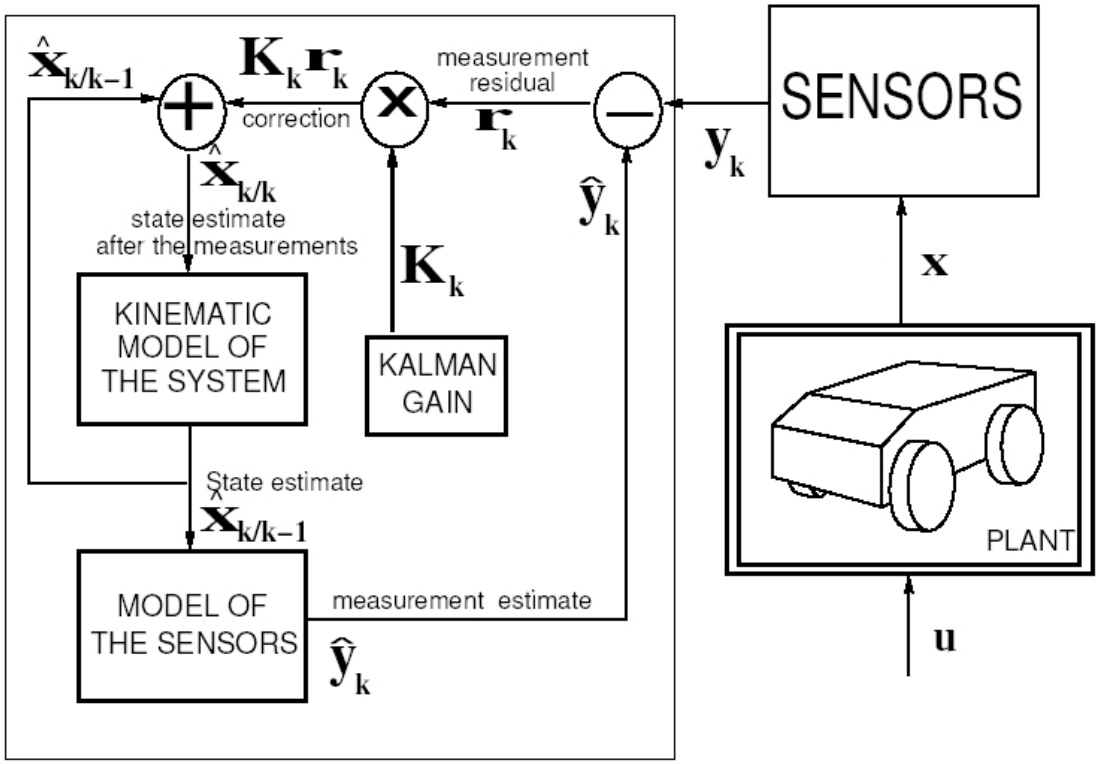
$$E[(x-\hat{x})^2 \mid Z_t]$$

  - Its minimum can be found by taking the derivative of the function w.r.t x and setting that value to 0.

$$\nabla_x (E[(x-\hat{x})^2 \mid Z]) = 0$$

- When they use the Gaussian assumption, Maximum A Posteriori estimators and MMSE estimators find the same value for the parameters.
  - This I can be explained (outside the scope of the course) because mean and the mode of a Gaussian distribution are the same.

Kalman Filter Block Diagram



Kalman Filter: an instance of Bayes' Filter

Assumptions guarantee that if the prior belief before the prediction step is Gaussian

$$
\begin{aligned}
bel(x_t) &= p(x_t | u_{0:t-1}, z_{0:t}) \\
&= \eta \, p(z_t | x_t) \int p(x_t | u_{t-1}, x_{t-1}) \, bel(x_{t-1}) \, dx_{t-1}
\end{aligned}
$$

then the prior belief after the prediction step will be Gaussian

and the posterior belief (after the update step) will be Gaussian.

Linear observations with Gaussian noise

$$
\begin{aligned}
z_t &= h(x_t) + n_t \\
&\quad \text{with noise } n_t \sim \mathcal{N}(0, R)
\end{aligned}
$$

Linear dynamics with Gaussian noise

$$
\begin{aligned}
x_t &= f(x_{t-1}, u_{t-1}) + G w_{t-1} \\
&\quad \text{with noise } w_{t-1} \sim \mathcal{N}(0, Q)
\end{aligned}
$$

+ Initial belief is Gaussian

$$bel(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$$

Suppose you replace the linear models with nonlinear models.

Does the posterior $bel(x_t)$ remain Gaussian? NO

EKF Summary

As in KF, inverting the covariance of the residual is O(k^2.376)

- Efficient: Polynomial in measurement dimensionality k and state dimensionality n:

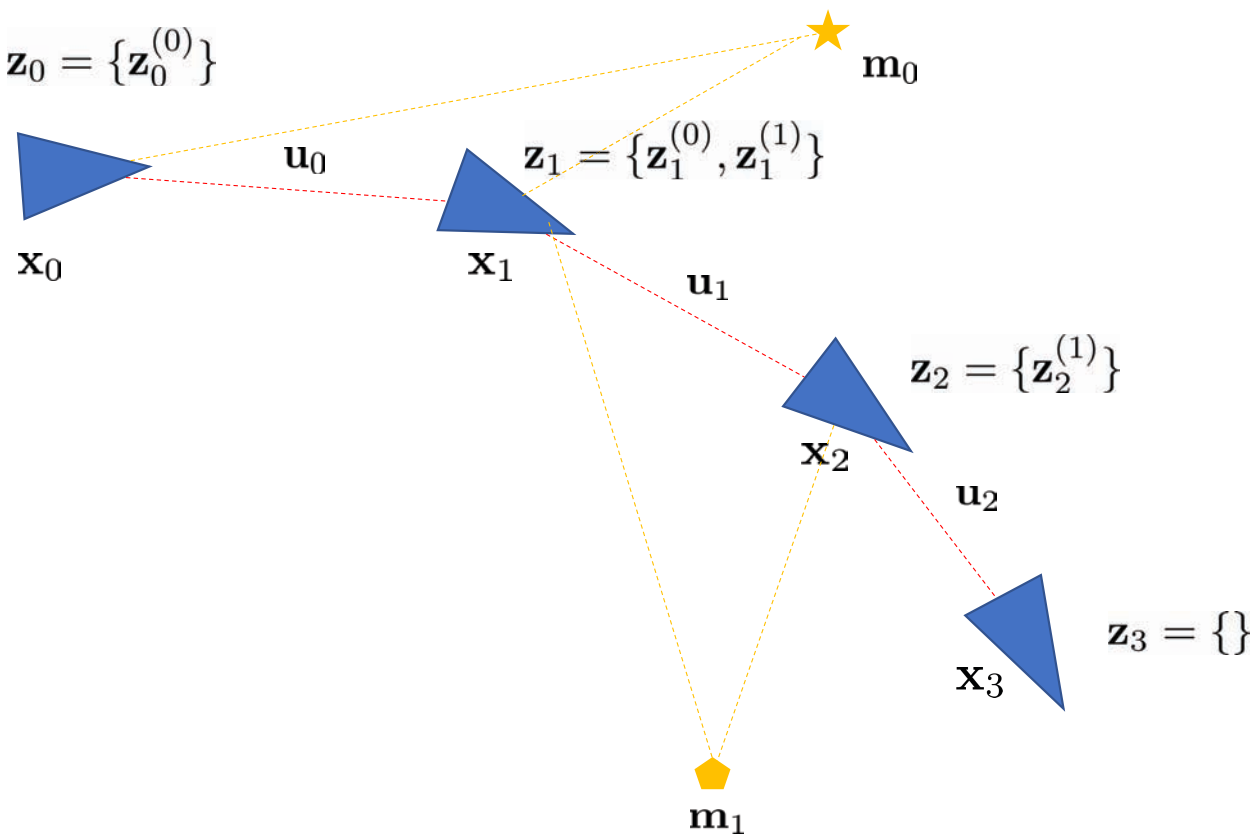
  $$O(k^{2.376} + n^2)$$

- Not optimal (unlike the Kalman Filter for linear systems)

- Can diverge if nonlinearities are large

- Works surprisingly well even when all assumptions are violated

SLAM: graph representation



Map $\mathbf{m} = \{\mathbf{m}_0, \mathbf{m}_1\}$ consists of landmarks that are easily identifiable and cannot be mistaken for one another.

SLAM: possible problem definitions

- Smoothing/Batch/Full SLAM

Initial state

$$p(\mathbf{x}_{1:T}, \mathbf{m} \mid \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}, \mathbf{x}_0)$$

Sequence of states

Map

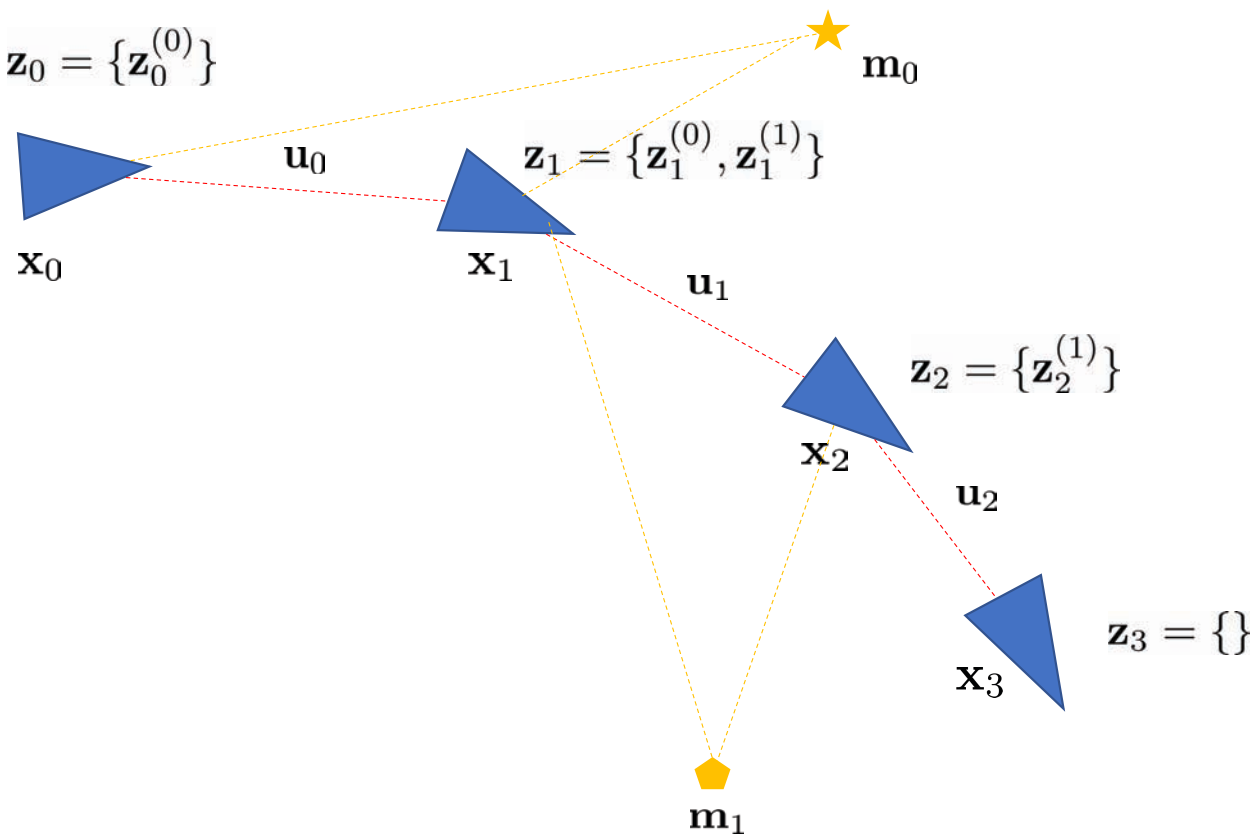
Sequence of sensor measurements

Sequence of commands

- Filtering SLAM

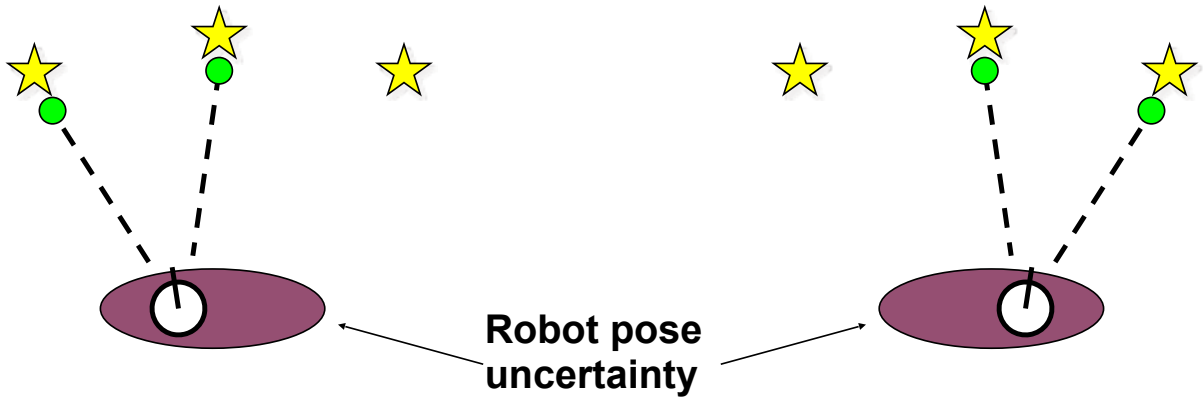
$$p(\mathbf{x}_t, \mathbf{m}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1}, \mathbf{x}_0)$$

SLAM: graph representation



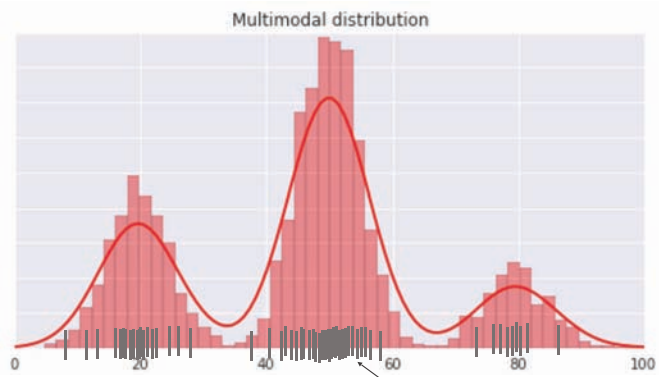
Map $\mathbf{m} = \{\mathbf{m}_0, \mathbf{m}_1\}$ consists of landmarks that are easily identifiable and cannot be mistaken for one another.

Why is SLAM a hard problem?



- In the real world, the mapping between observations and landmarks is unknown
- Picking wrong data associations can have catastrophic consequences
- Pose error correlates data associations

How can we represent multimodal distributions?



https://www.opsclarity.com

Higher density of particles means higher probability mass

Idea #1: Histograms

Idea #3: Weighted Particles $\{(x^{[1]}, w^{[1]}), ..., (x^{[M]}, w^{[M]})\}$

Advantages: easy to predict/update by treating each particle as a separate hypothesis whose weight is updated.

Disadvantages: need enough particles to "cover" the distribution

Want particles to be drawn from the belief at time t:

$$x_t^{[m]} \sim p(x_t | z_{0:t}, u_{0:t-1})$$

Using particles to represent a distribution

- Each particle is a hypothesis
- Each particle moves forward using your dynamics model (motion model)
- Each particle is "confirmed" by comparing the actual observation to what the particle pose would predict
- These predictions are used to either:
  - change the particle density (unweighted particles)
  - change the particle weight

Resampling Particles: Consequences

- Weak particles very likely do not survive.

⇩

- Variance among the set of particles **decreases**, due to mostly sampling strong particles (i.e. loss of particle diversity).

⇩

- Loss of particle diversity implies **increased variance** of the approximation error between the particles and the true distribution**.**

⇩

- Particle deprivation: there are no particles in the vicinity of the correct state

Particle Filter Algorithm

ParticleFilter($\bar{z}_t, u_{t-1}$)

$\bar{S}_t = \{\}$ $\quad \bar{W}_t = \{\}$

for particle index $m = 1 \ldots M$

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

$w_t^{[m]} = p(\bar{z}_t | x_t^{[m]})$

Weight computation as measurement likelihood.
For each particle we compute the probability of the actual observation given the state is at that particle.

$\bar{S}_t$.append($x_t^{[m]}$)

$\bar{W}_t$.append($w_t^{[m]}$)

$S_t = \{\}$

for particle index $m = 1 \ldots M$

sample particle i from $\bar{S}_t$ with probability $\propto w_t^{[i]}$

S_t.append($x_t^{[m]}$)

return S_t

Applications

- Self-driving vehicles: cars, trucks, airplanes, boats, submarine
- Helping around the house
- Factory automation
- Warfare
- Scientific exploration
- Commercial exploration
- Medical applications
    - Surgery
    - Diagnosis
- Assistive robotics:
    - The elderly
    - The infirm
    - Everybody

Lethal robots

- Little evidence that lethal robots autonomous systems are considered differently (by the military) than any other weaponry
- Often more about safety