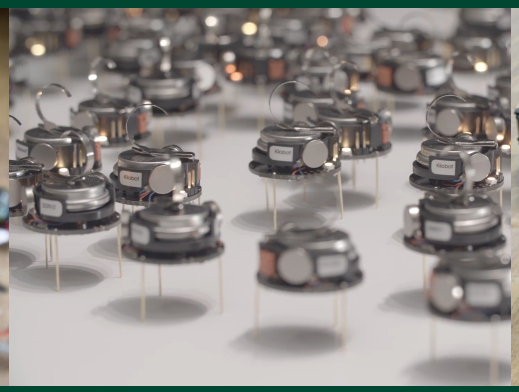
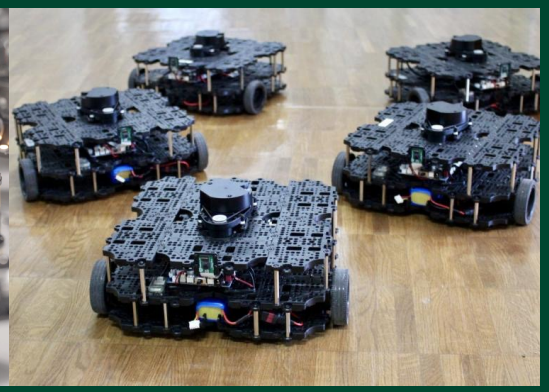
ECE693H, Spring 2025: Multi-robot System Design

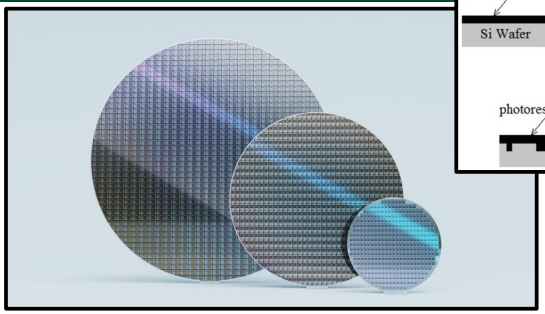
“Intelligence 1”

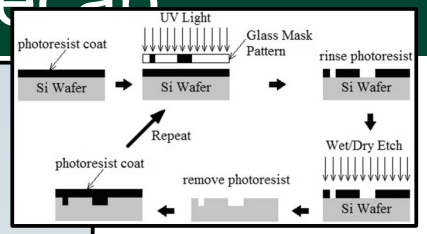



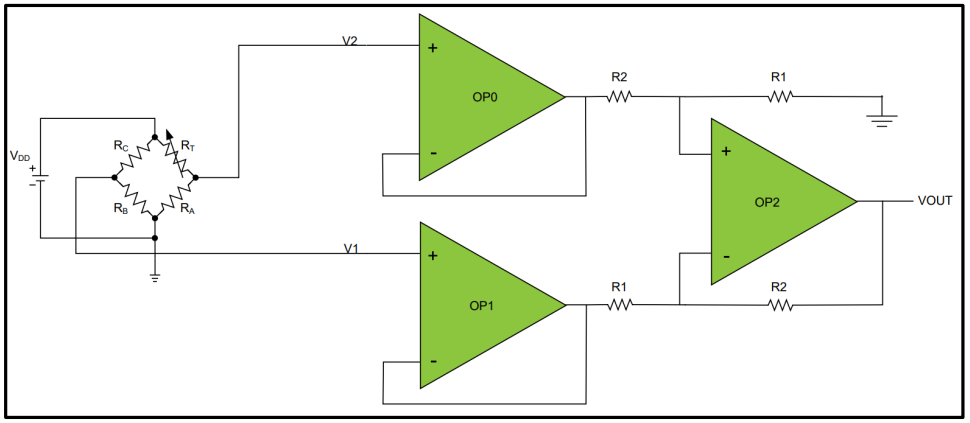
Dr. Daniel Drew

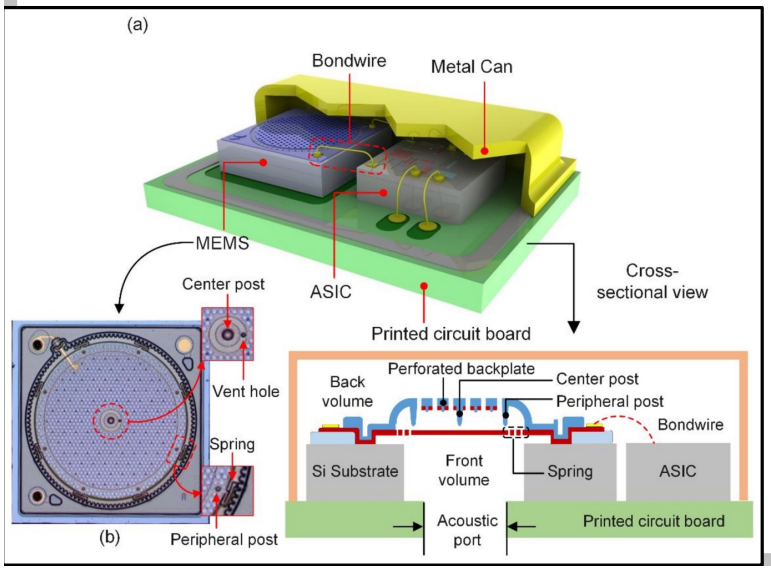
College of Engineering
University of Hawai'i at Mānoa

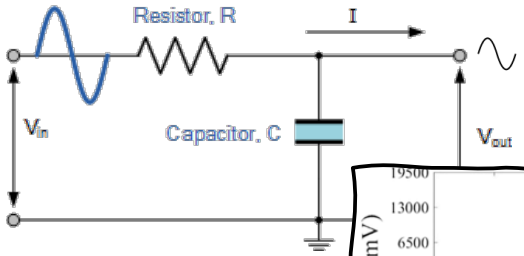
Perception Recap

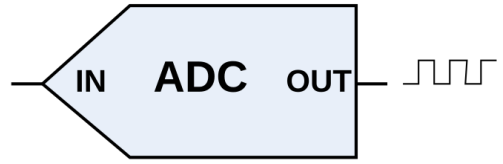


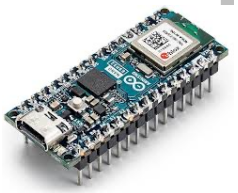


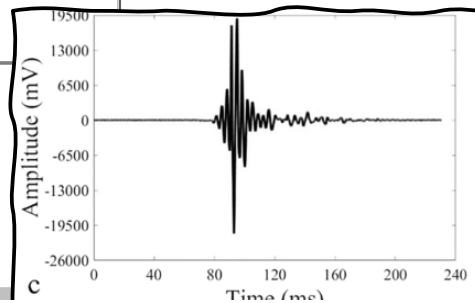


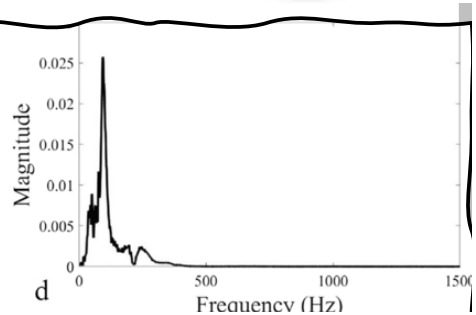












Questions?

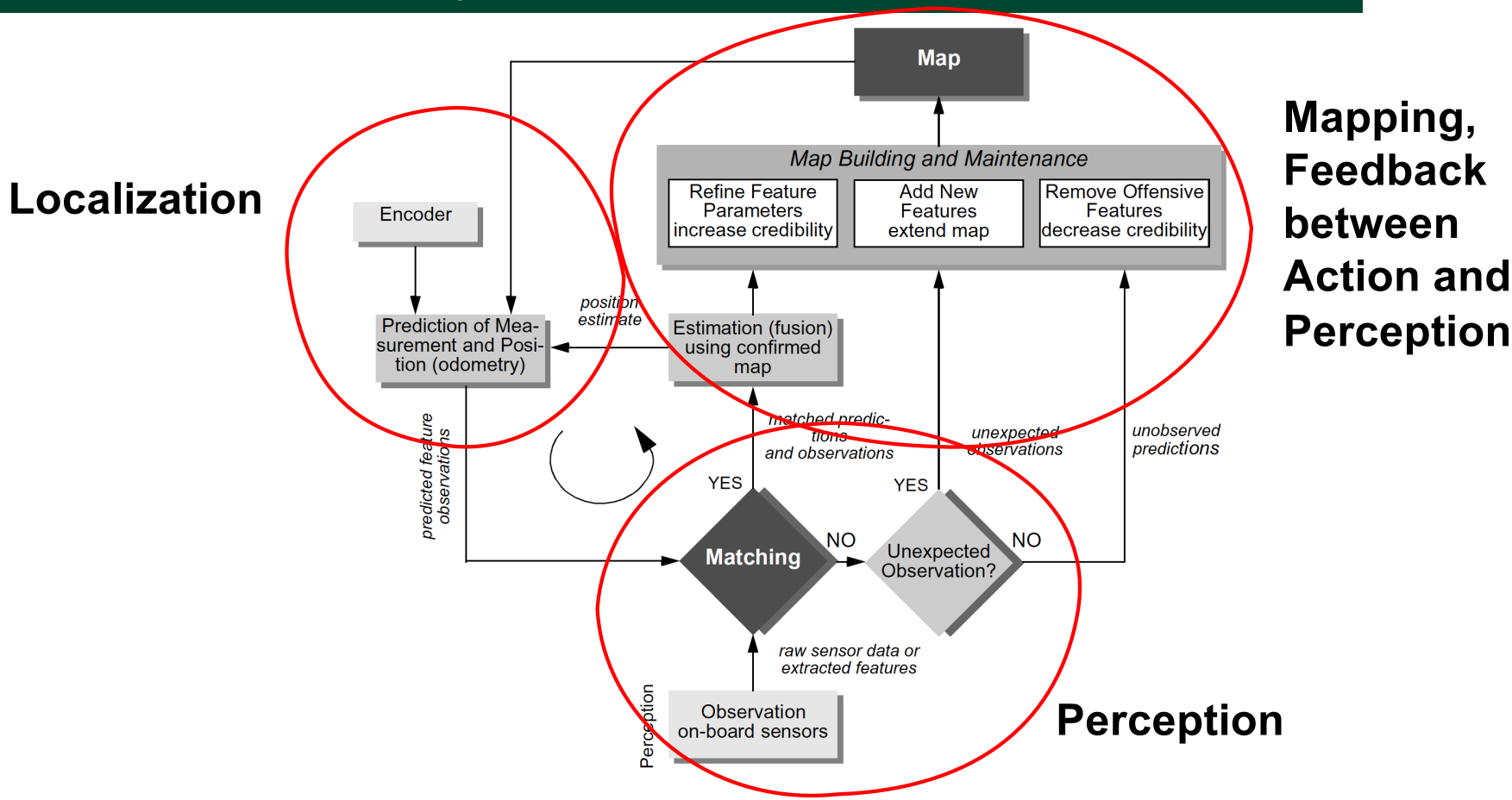
Class Overview

1. Lecture (~40 minutes)
2. Subsystem check-in (~20 minutes)

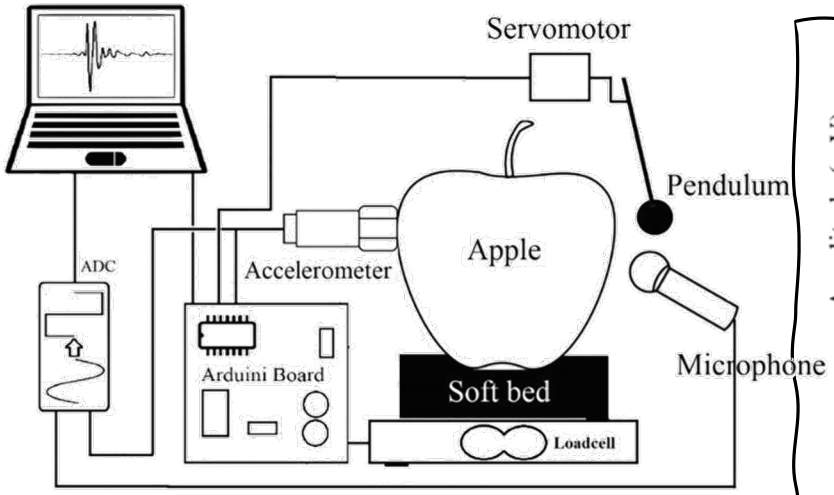
Intelligence: Lecture 1 of 2

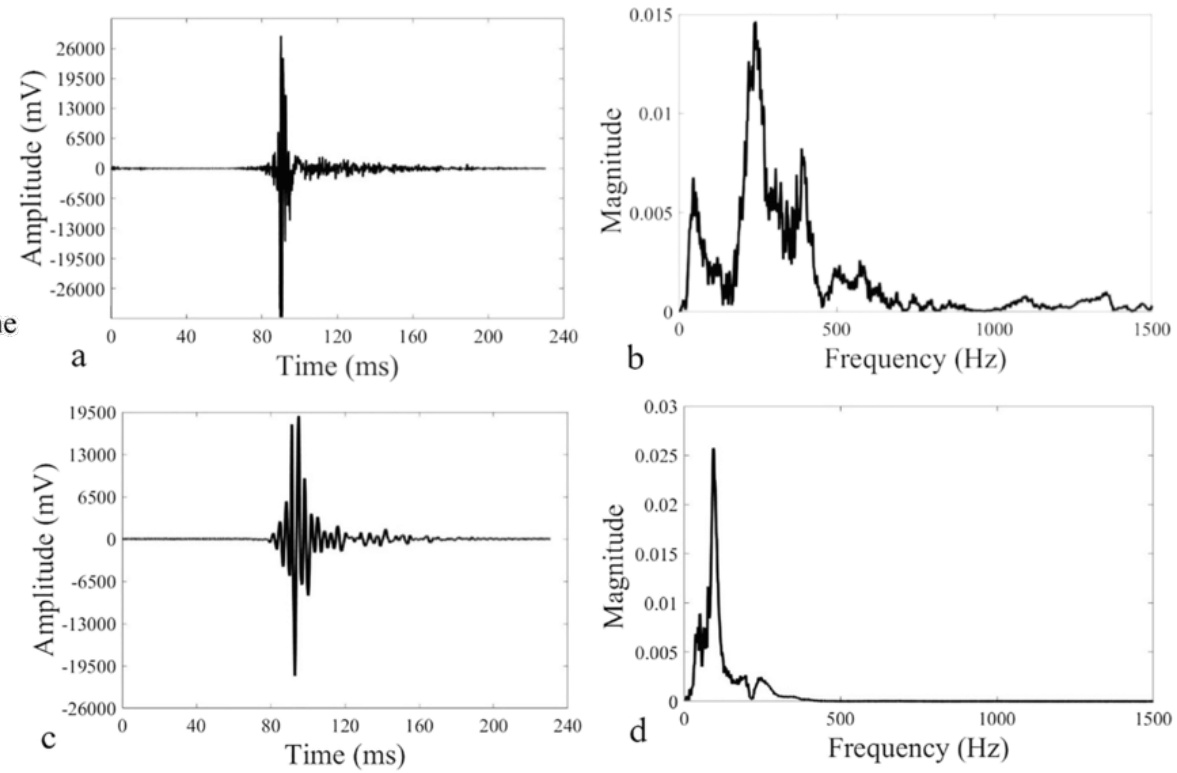
Lecture 1: features, transformations, and mapping
Lecture 2: SLAM

SLAM as an archetypical function



From Data to Features



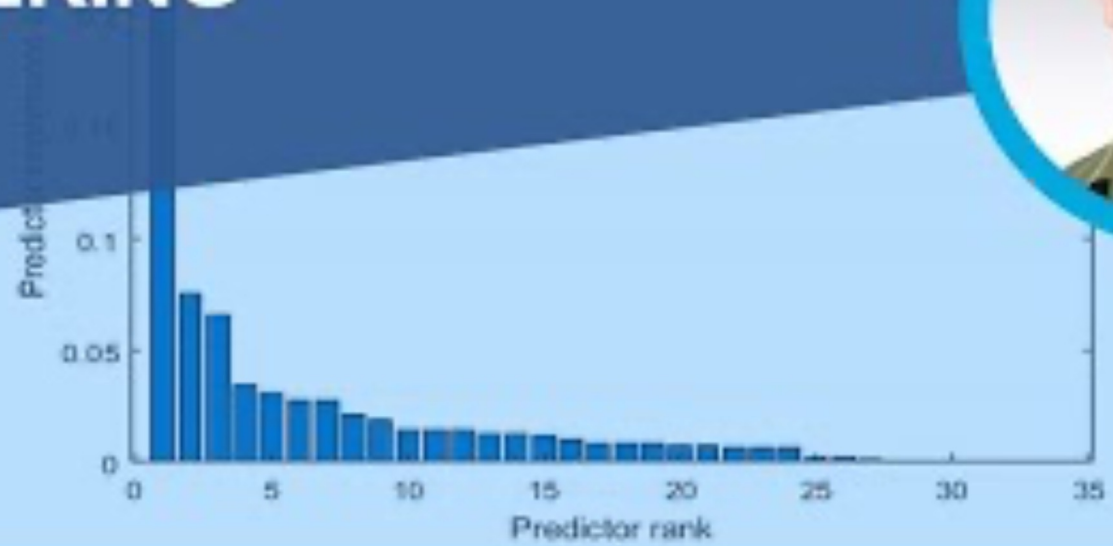


Apple Testing

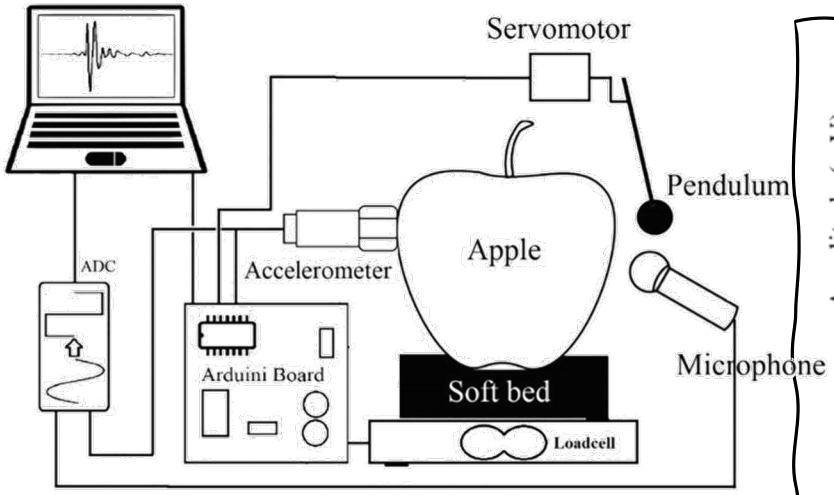
PART 1

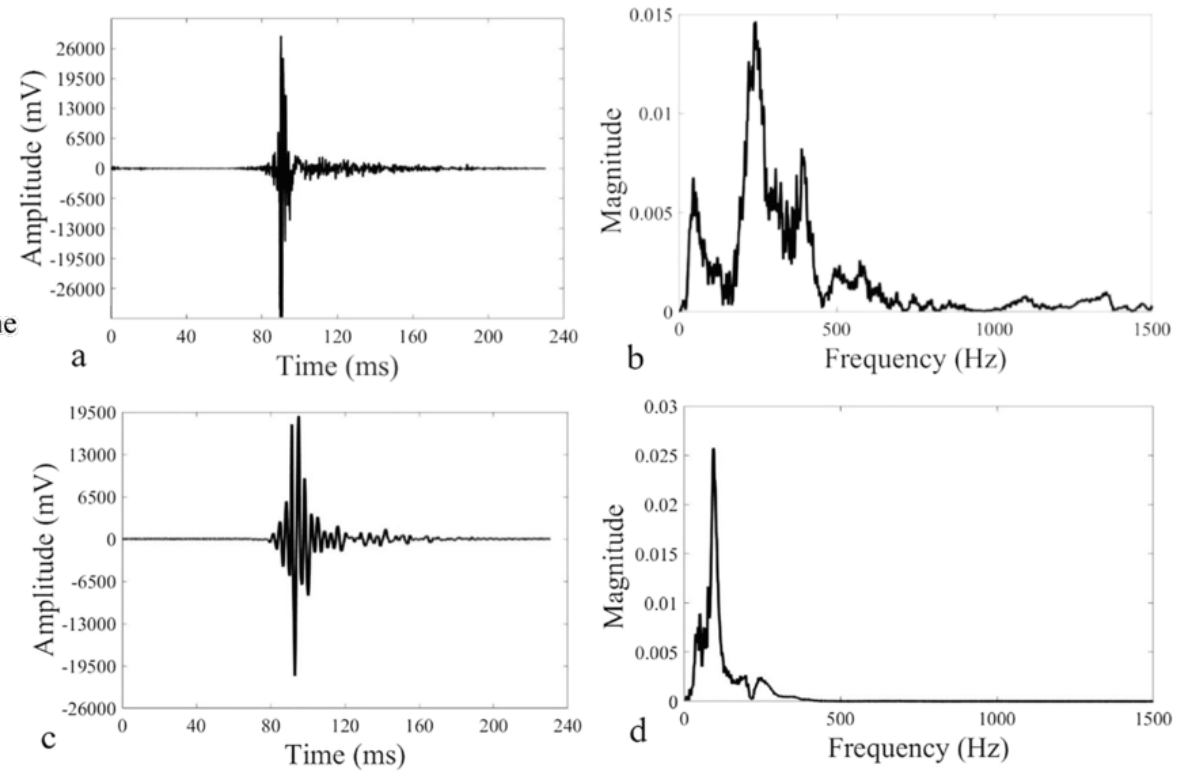
MACHINE LEARNING

FEATURE ENGINEERING



From Data to Features

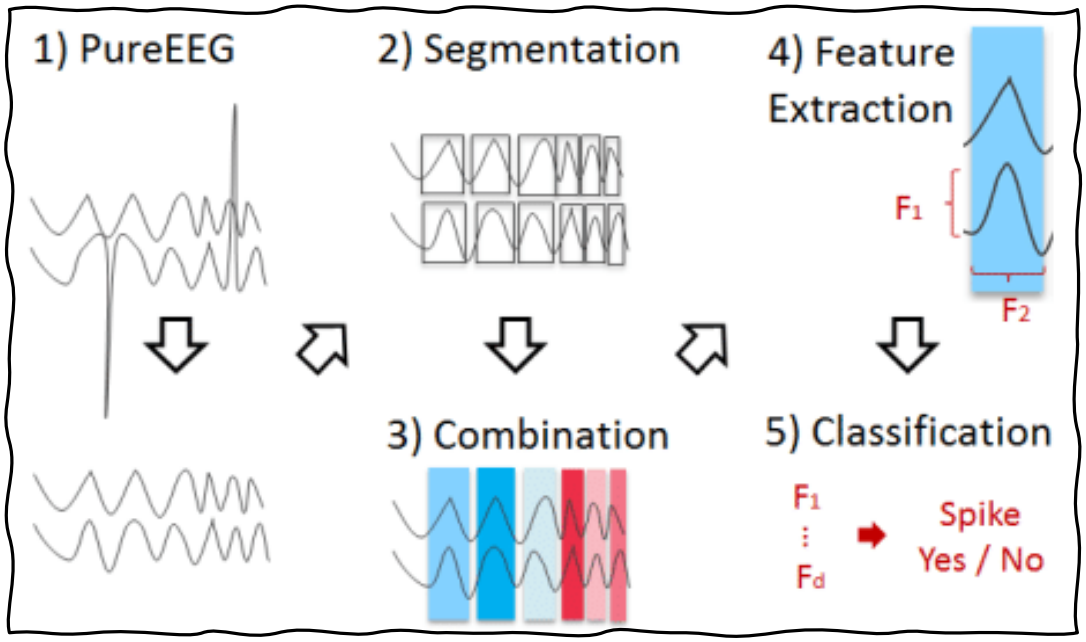




Spectral purity?
Max amplitude?
Spectrum centroid?
Harmonic amplitude?

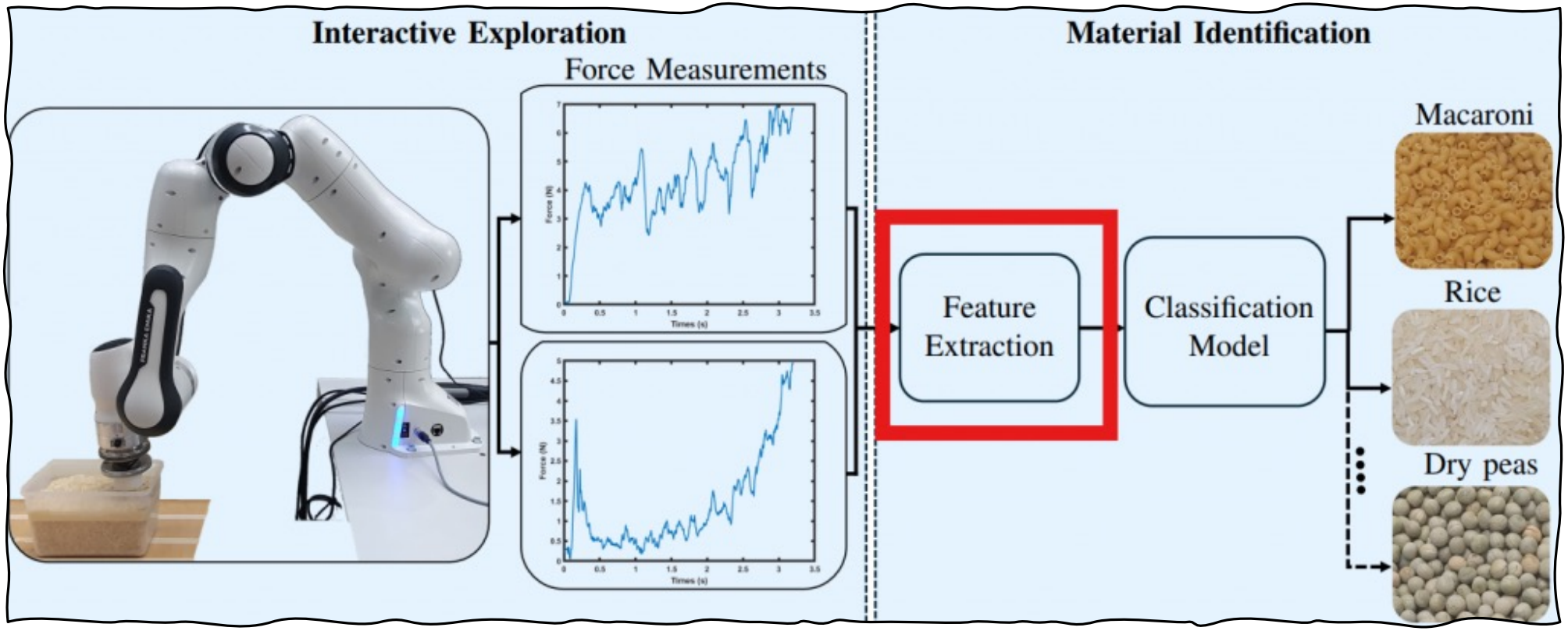
Apple Testing

From Data to Features

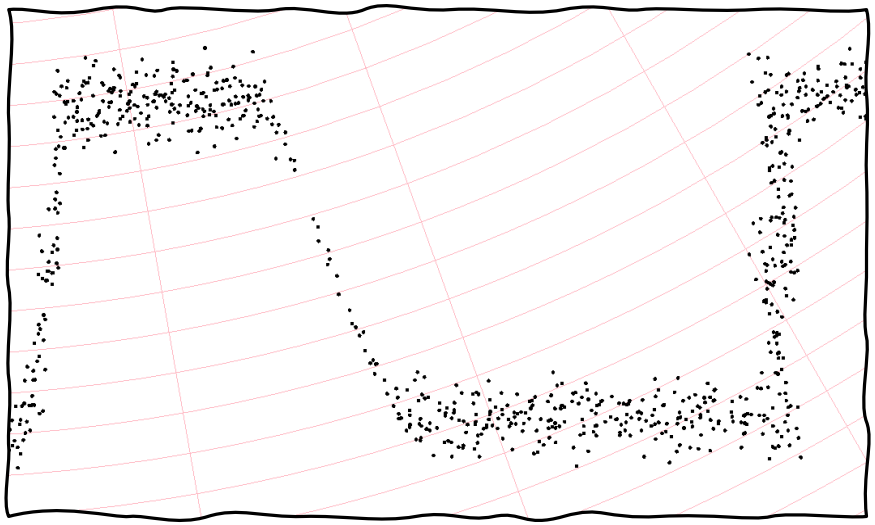


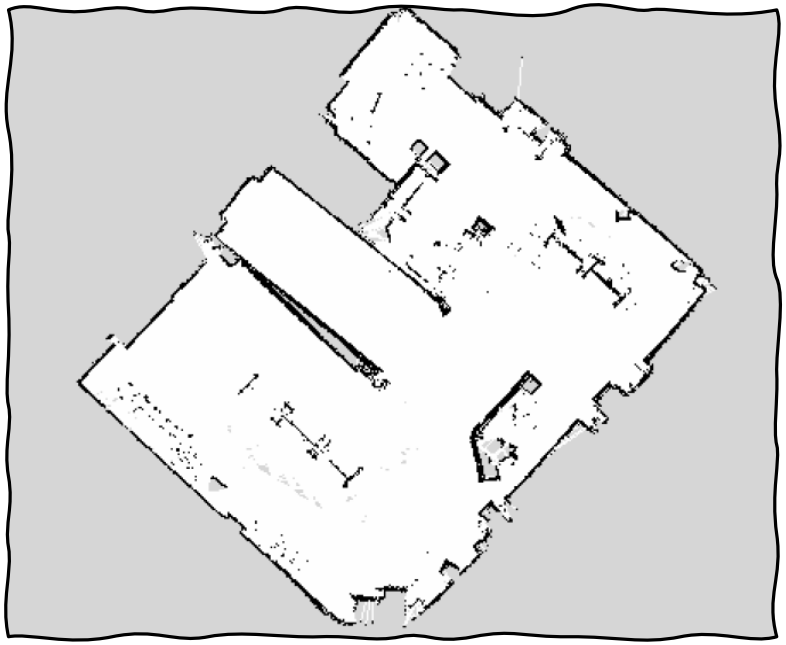
From Data to Features

From Data to Features



Features: Edges

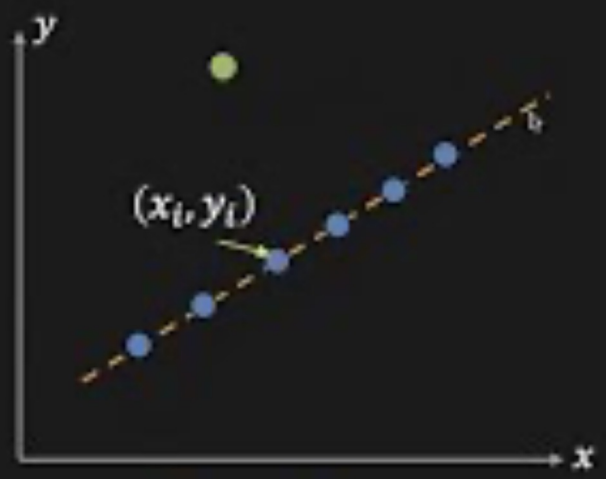




How do we turn this into “walls” -- algorithmically?

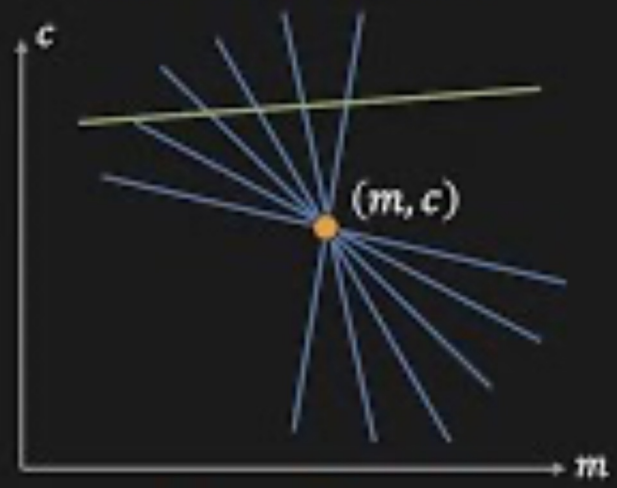
Hough Transform: Concept

Image Space



$$y_i = mx_i + c$$

Parameter Space



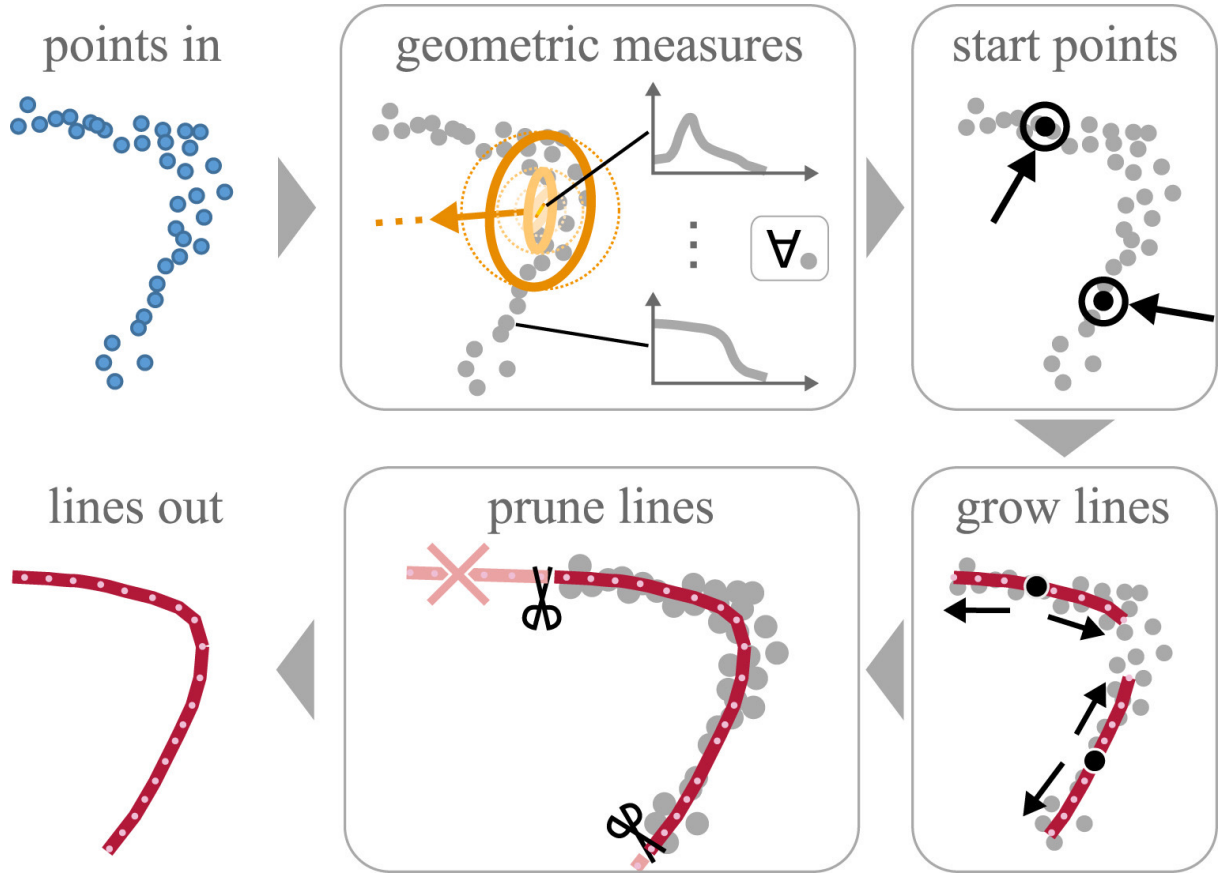
$$c = -mx_i + y_i$$

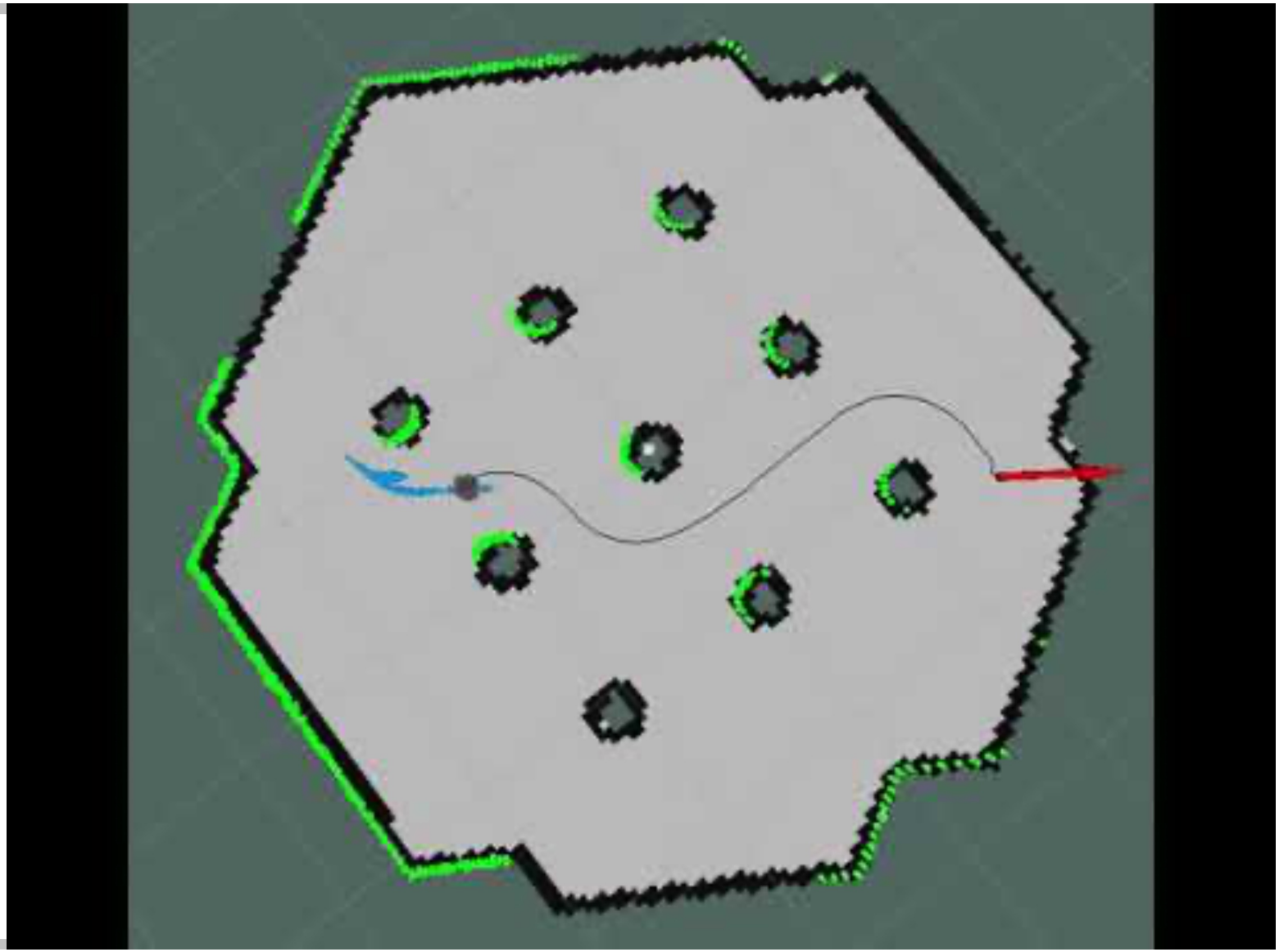
© 2020 Shree K. Nayar

Features: Curved Edges

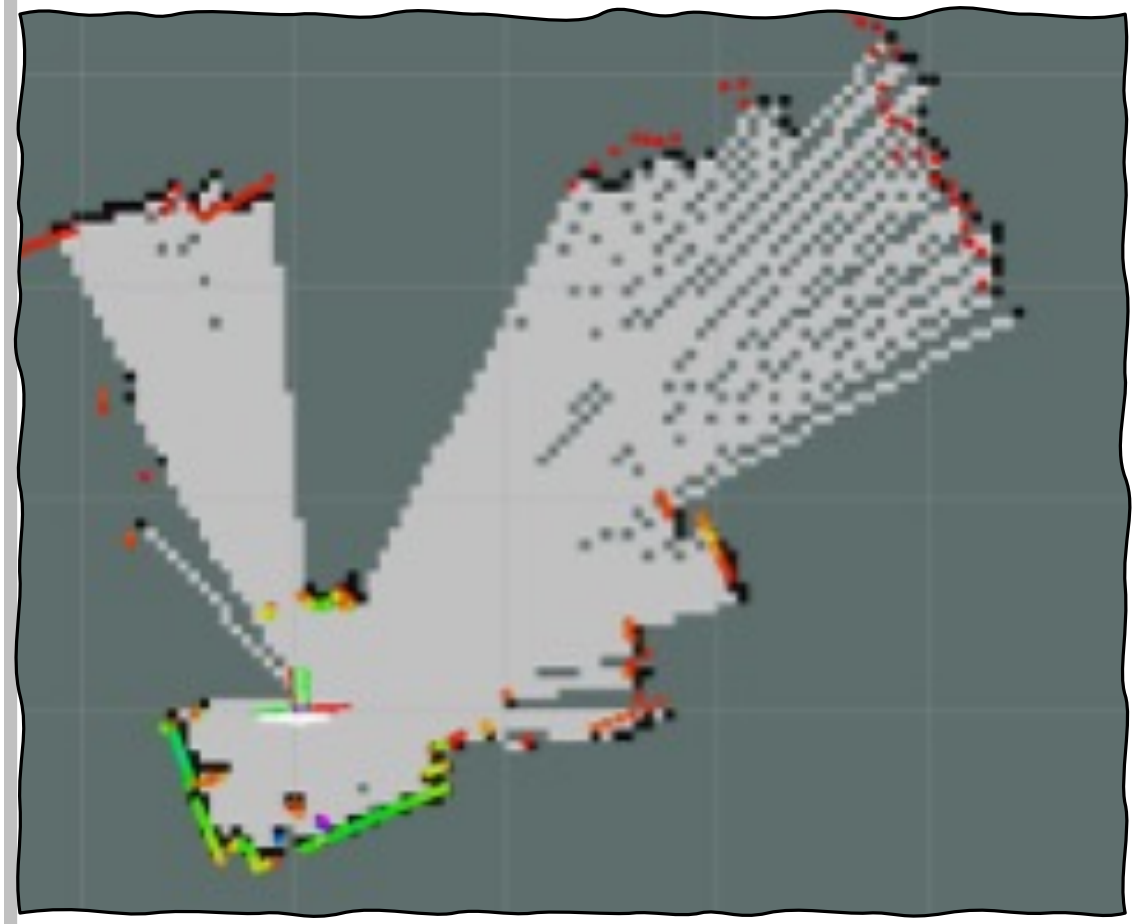
Generalized Hough Transform (GHT) works for curves – but expensive.

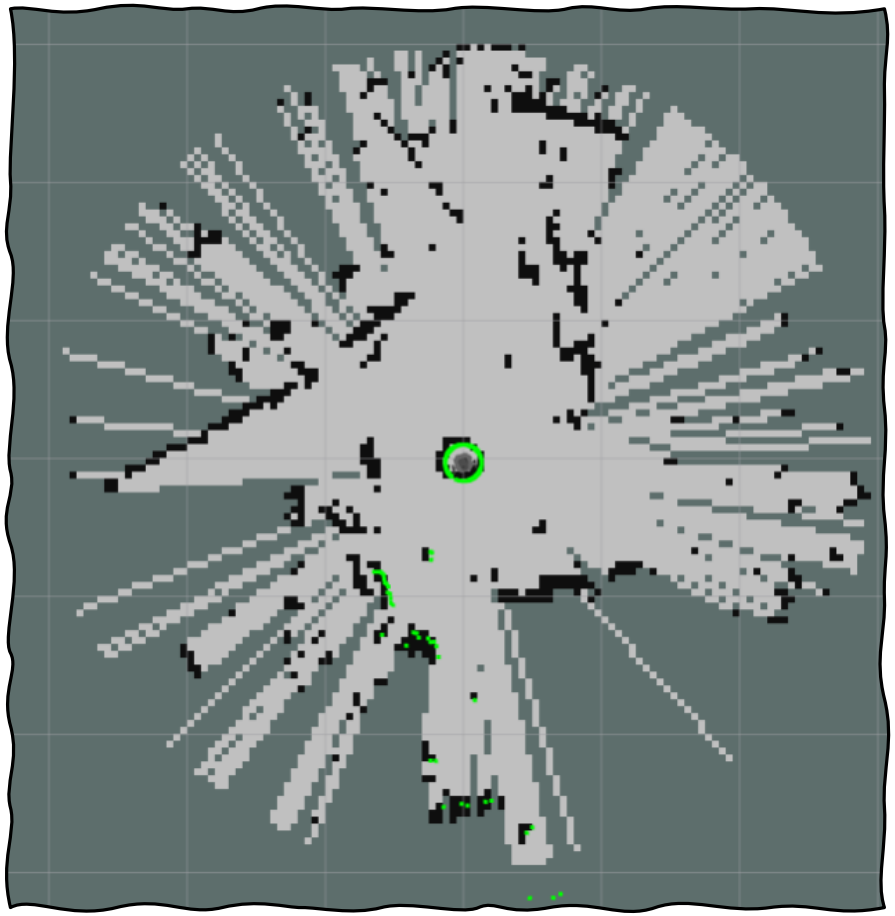
Alternatives are in research:



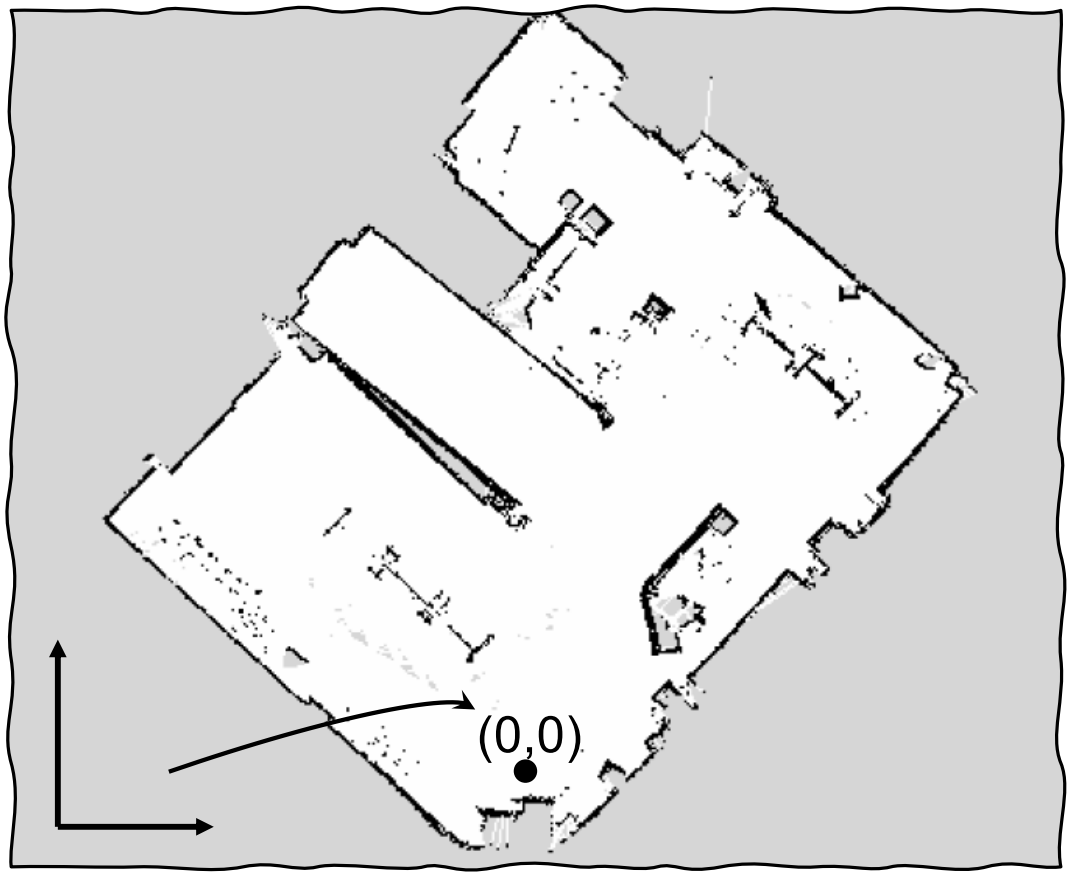


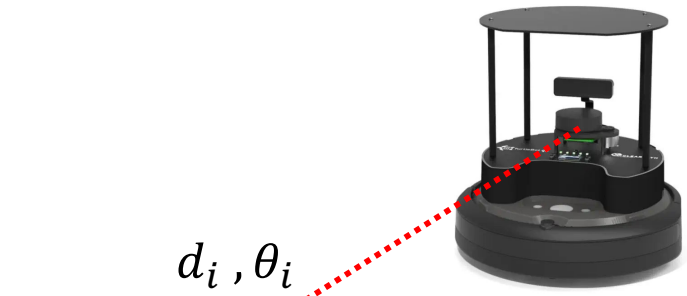
Features: Gone Wrong



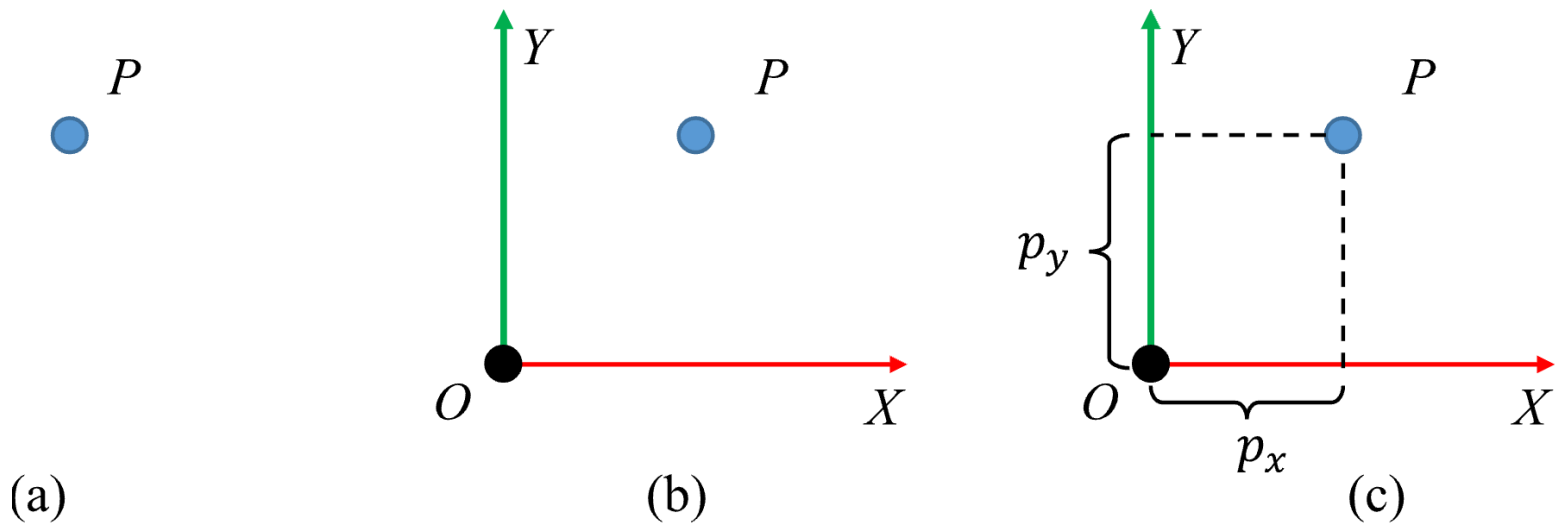


The World Frame



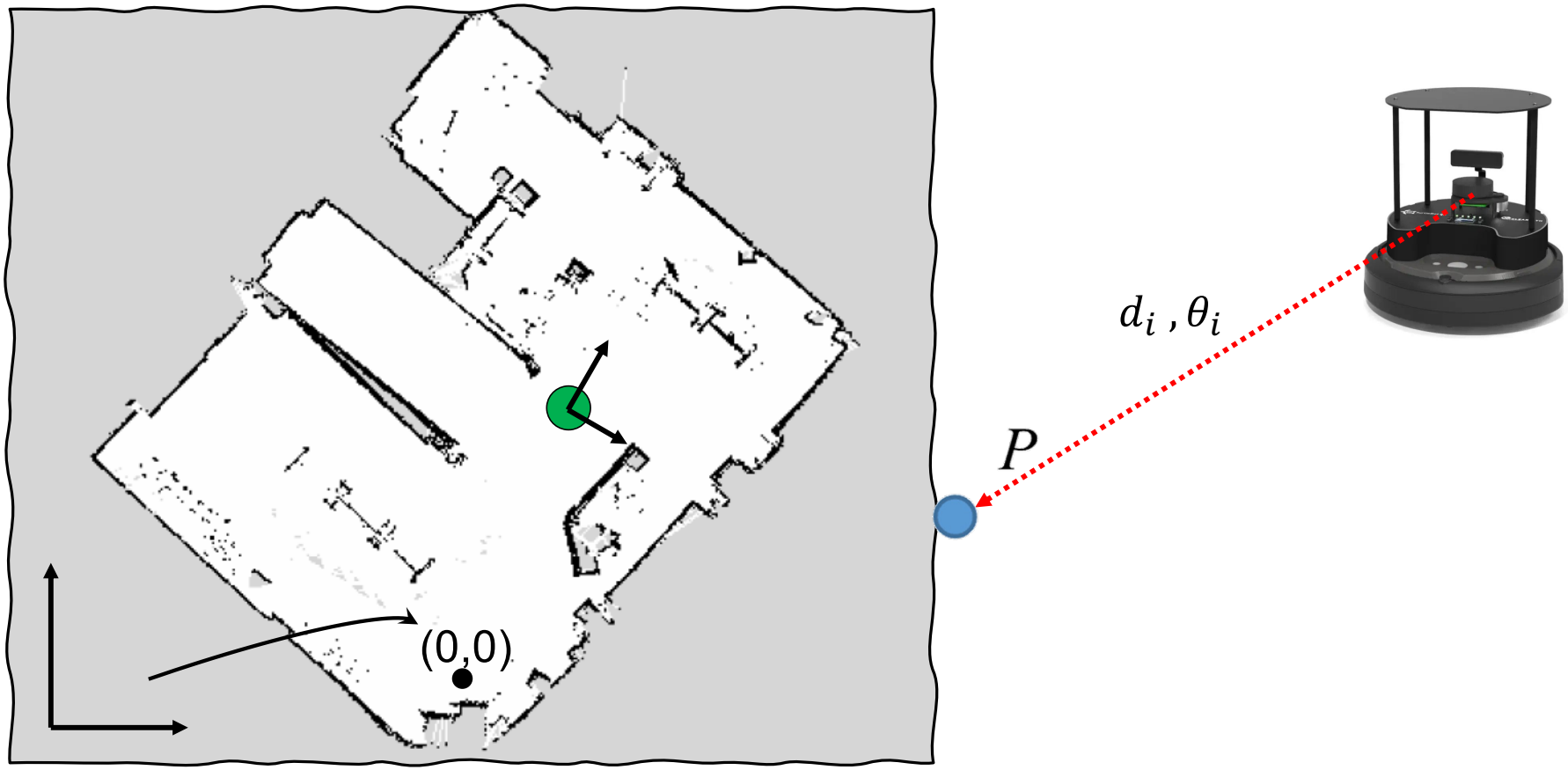


Coordinate Frames



A point means nothing without a reference coordinate frame

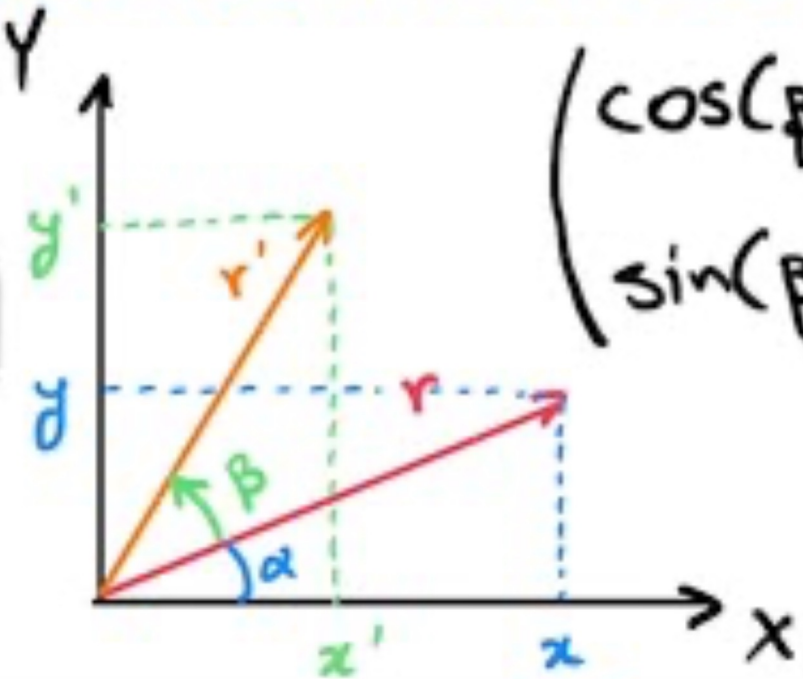
The World Frame



Rotation Matrix

-DERIVATION-

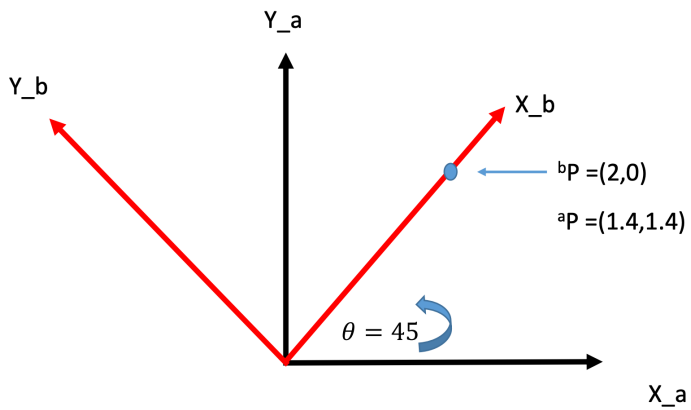
Step By Step Prove



$$\begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix}$$

Frame Transformations: Rotation

2D Rotation Matrix



$^aP = {}^aR_b\ {}^bP$

$$^aR_b = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} .7 & -.7 \\ .7 & .7 \end{pmatrix}$$

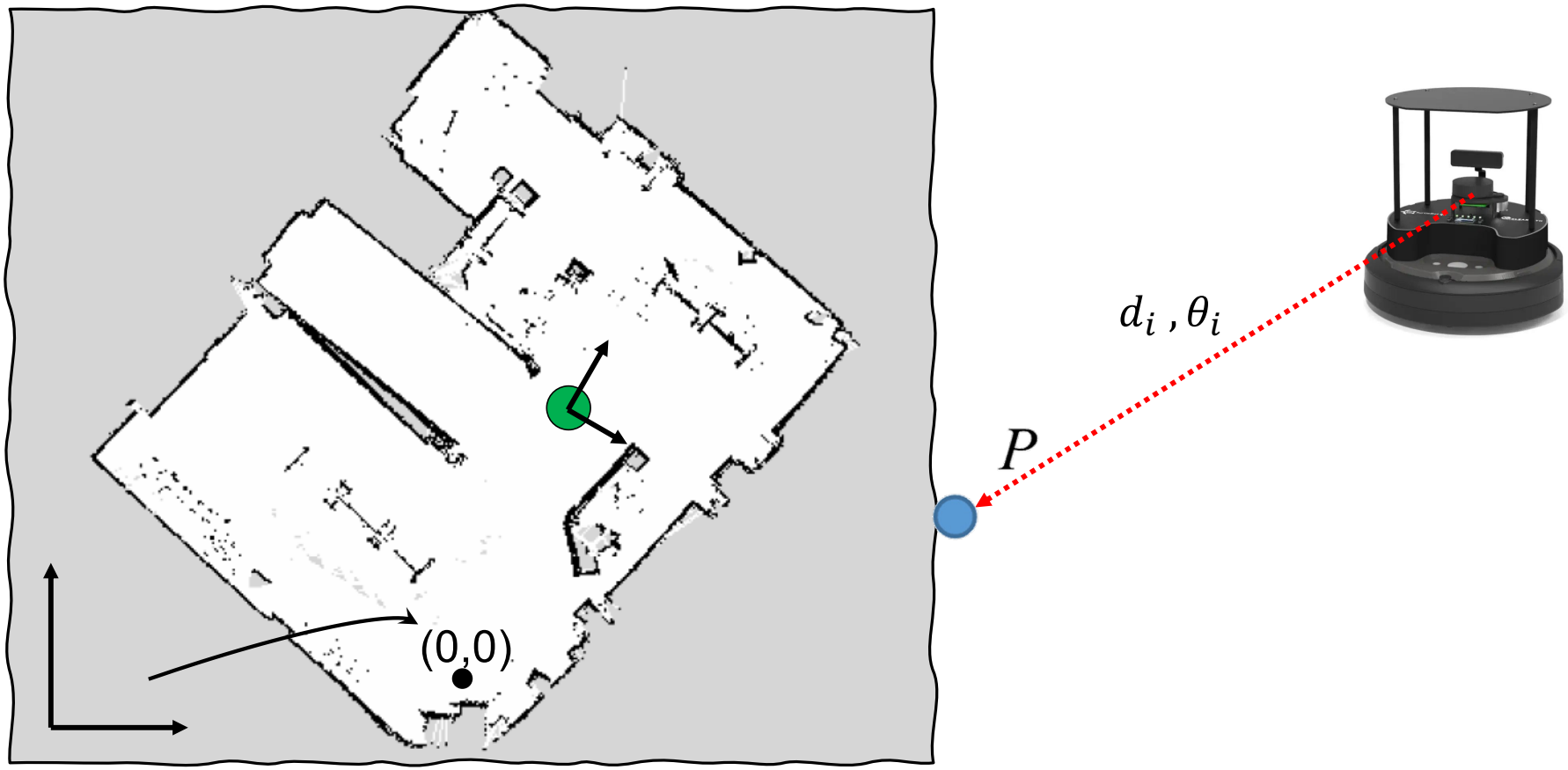
(rotates b to a)

$$(x_a, y_a)^T = \begin{pmatrix} cos45 & -sin45 \\ sin45 & cos45 \end{pmatrix} (x_b, y_b)^T$$

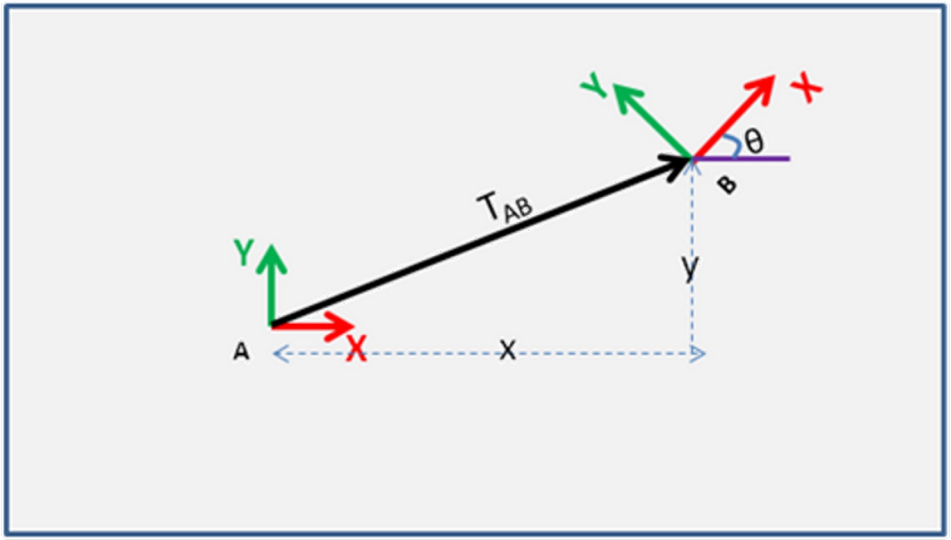
$$(x_a, y_a)^T = \begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix} \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

$$(x_a, y_a)^T = \begin{pmatrix} \frac{\sqrt{2}}{2} \times 2 + (-\frac{\sqrt{2}}{2} \times 0) \\ \frac{\sqrt{2}}{2} \times 2 + \frac{\sqrt{2}}{2} \times 0 \end{pmatrix} = \begin{pmatrix} \sqrt{2} \\ \sqrt{2} \end{pmatrix}$$

The World Frame



Frame Transformations: Adding Translation



$$ {}^{A}T_{B} = \begin{bmatrix} cos\theta & -sin\theta & P_x \\ sin\theta & cos\theta & P_y \\ 0 & 0 & 1 \end{bmatrix} $$

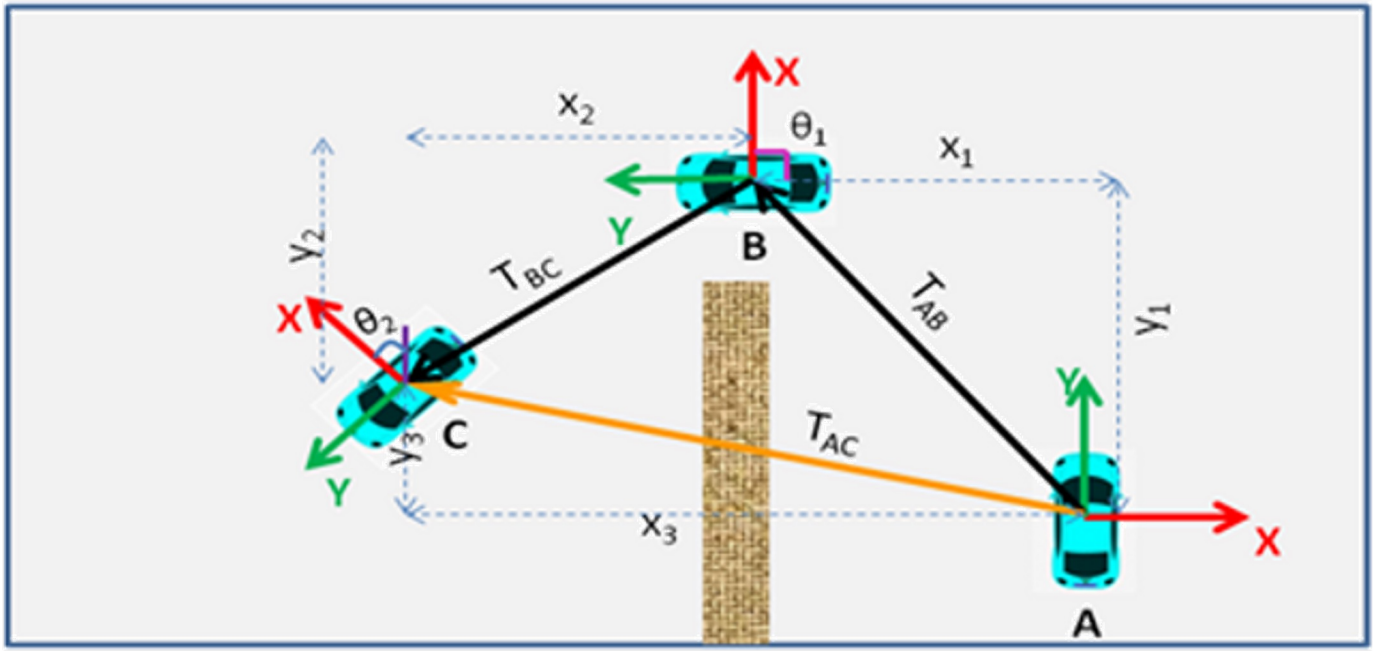
Translation in x and y

Consider a point $P = (3, 2)$ which we want to rotate 30 degrees counterclockwise, then translate by $(4, 5)$ in x, y

$$ T_1^2 = \begin{pmatrix} cos30° & -sin30° & 4 \\ sin30° & cos30° & 5 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.866 & -0.5 & 4 \\ 0.5 & 0.866 & 5 \\ 0 & 0 & 1 \end{pmatrix} $$

$$ P_2 = T_1^2 P_1 = \begin{pmatrix} 0.866 & -0.5 & 4 \\ 0.5 & 0.866 & 5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 5.598 \\ 8.232 \\ 1 \end{pmatrix} $$

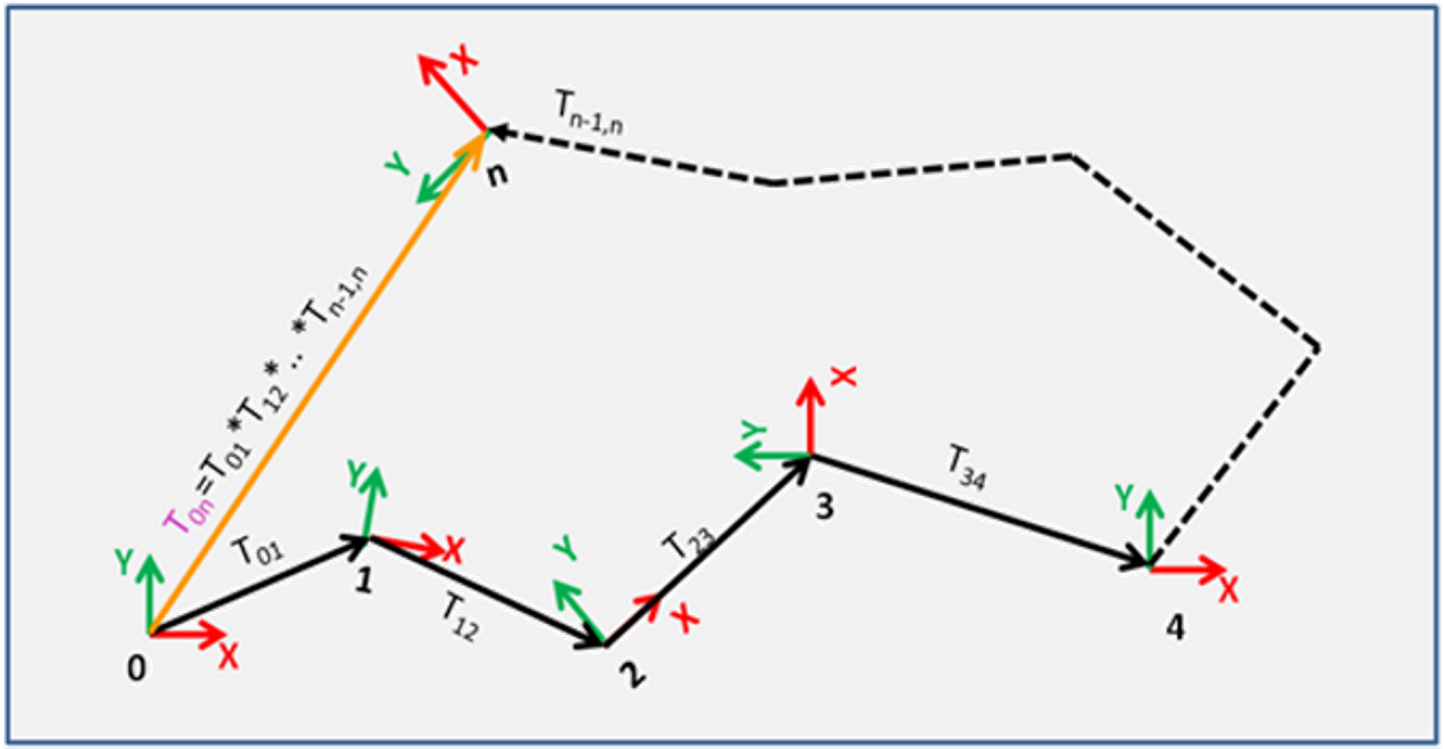
Frame Transformations: Successive Transforms



$$T_A^C = T_A^B T_B^C$$

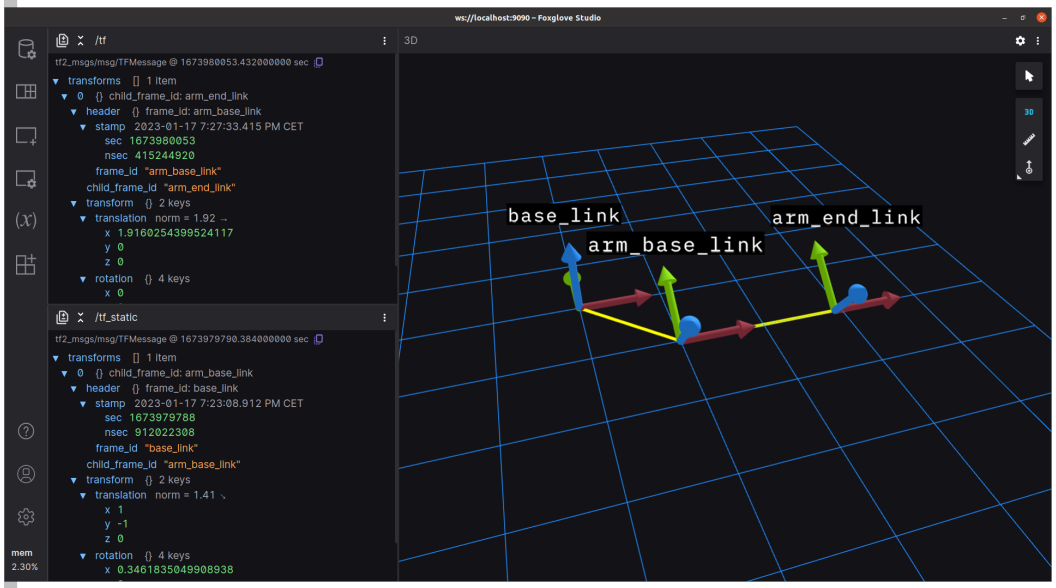
(transforms A to C)

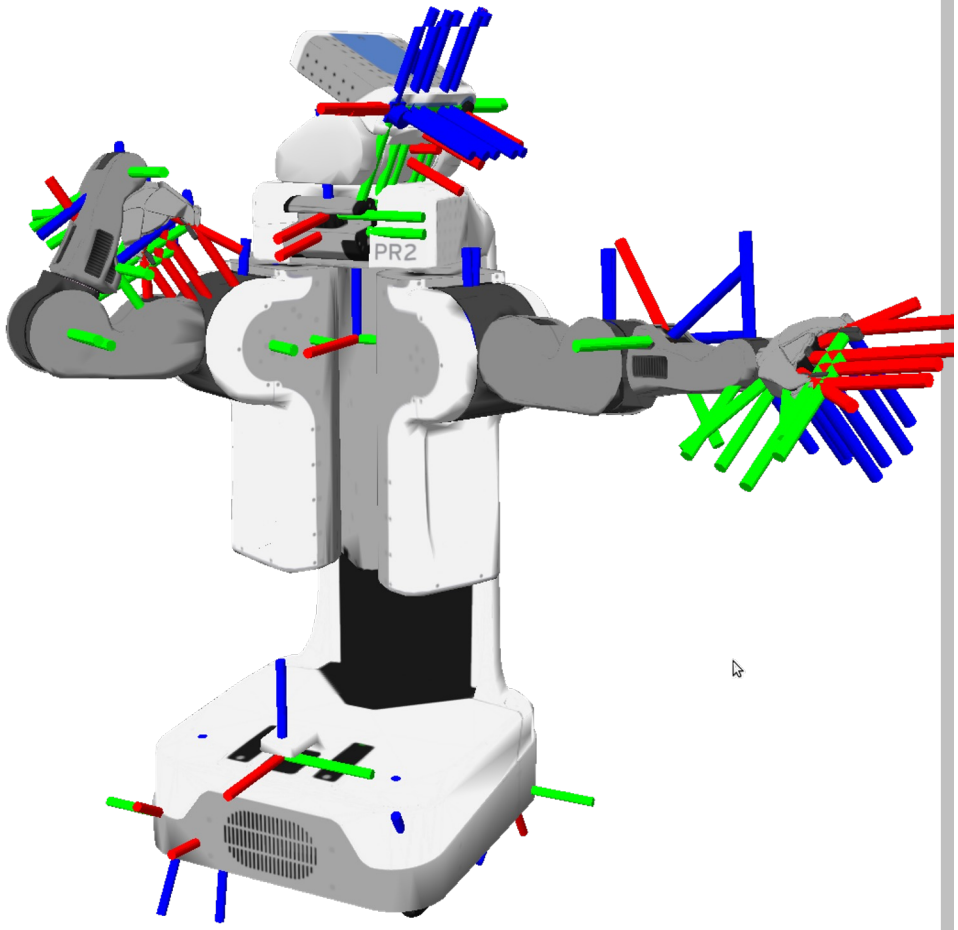
Frame Transformations: Successive Transforms

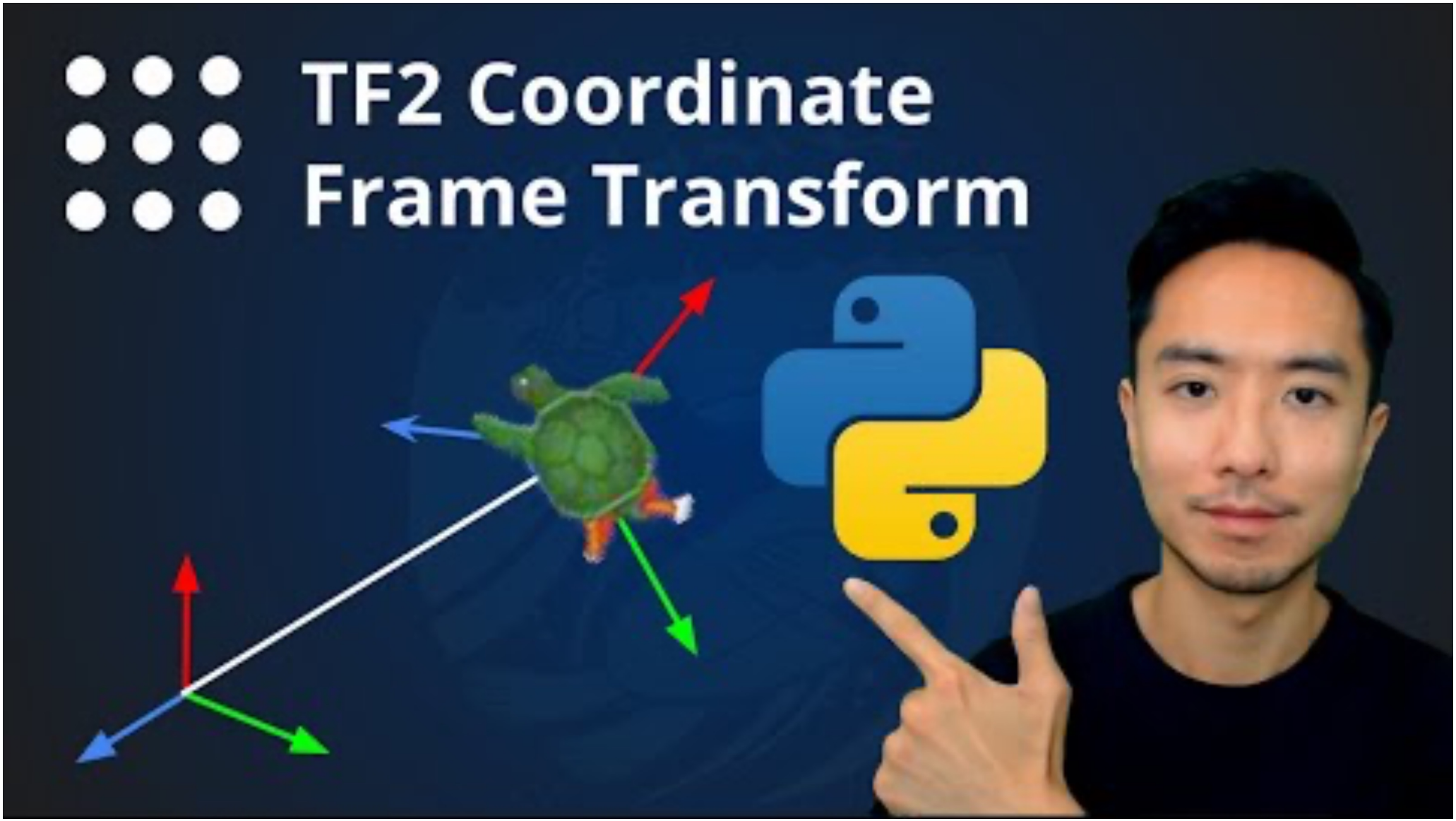


$$T_{0n}=T_{01}*T_{12}*T_{23}*.....T_{n-2,n-1}*T_{n-1,n}$$

Frame Transformations: ROS TF2



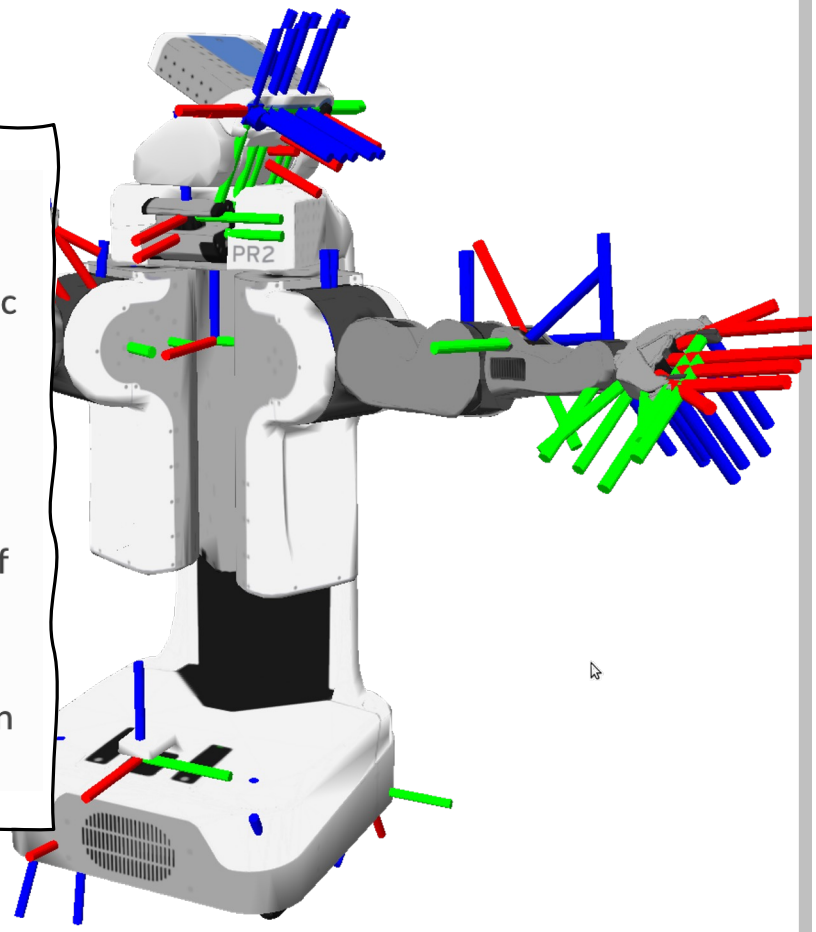



TF2 Coordinate
Frame Transform

Frame Transformations: ROS TF2

If you want to use tf2 to transform between coordinate frames, your nodes will need to listen for transforms. What you will do is receive and buffer all coordinate frames that are broadcasted in the system, and query for specific transforms between frames. Check out the “Writing a listener” tutorial (Python) (C++) to learn more.

To extend the capabilities of a robot, you will need to start broadcasting transforms. Broadcasting transforms means to send out the relative pose of coordinate frames to the rest of the system. A system can have many broadcasters that each provide information about a different part of the robot. Check out the “Writing a broadcaster” tutorial (Python) (C++) to learn more.



Mapping

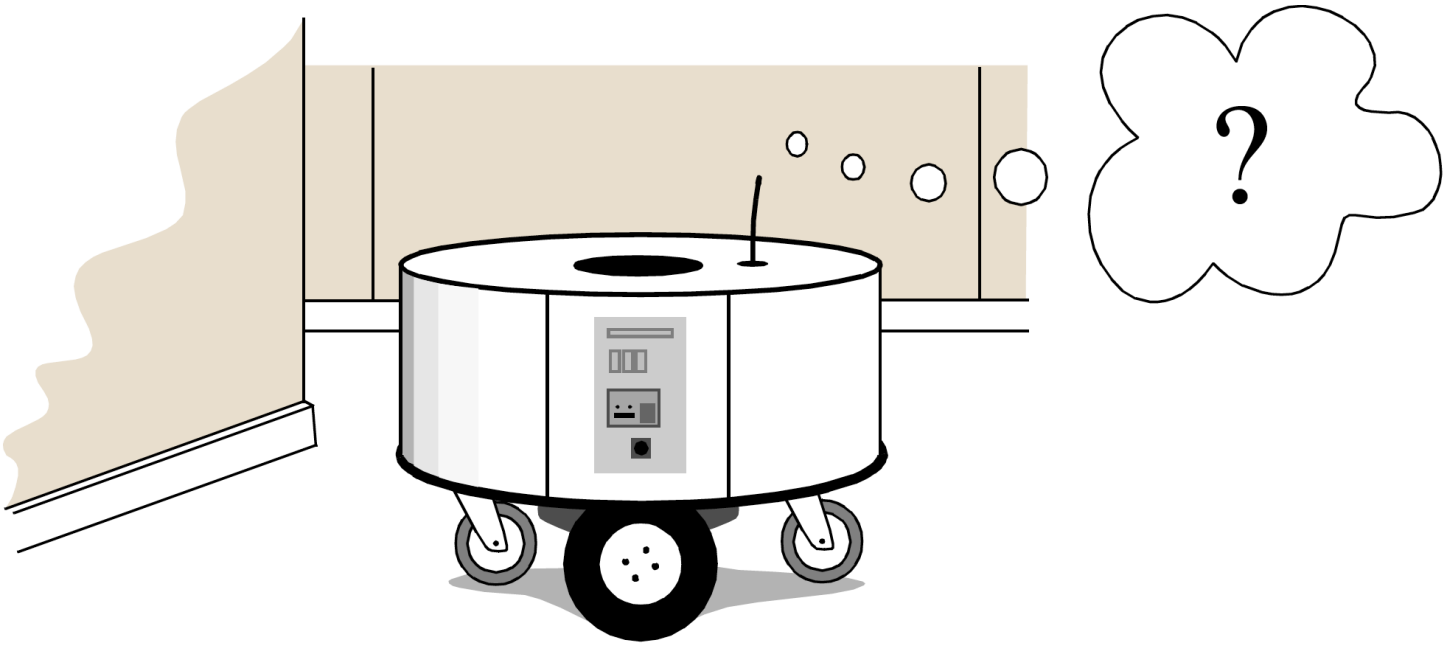
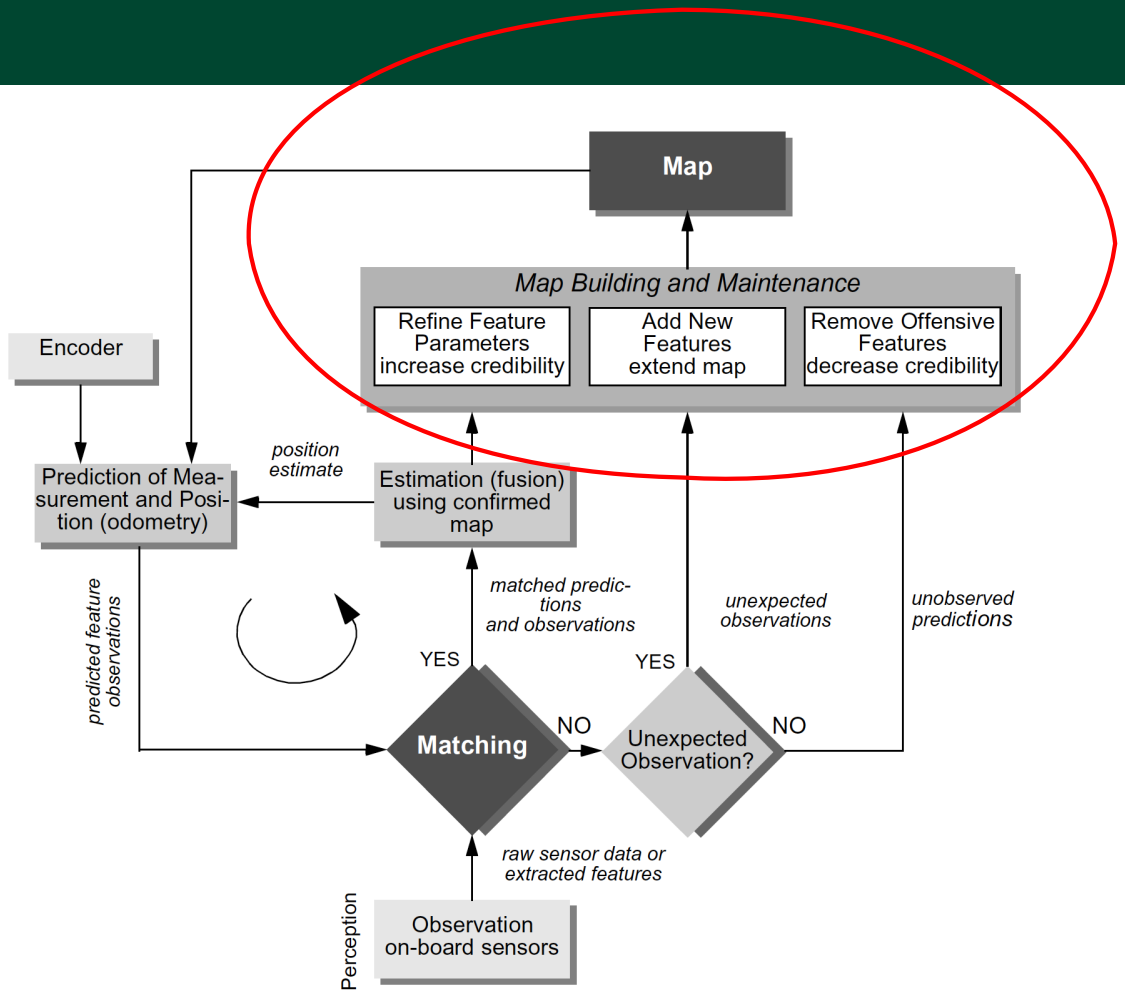
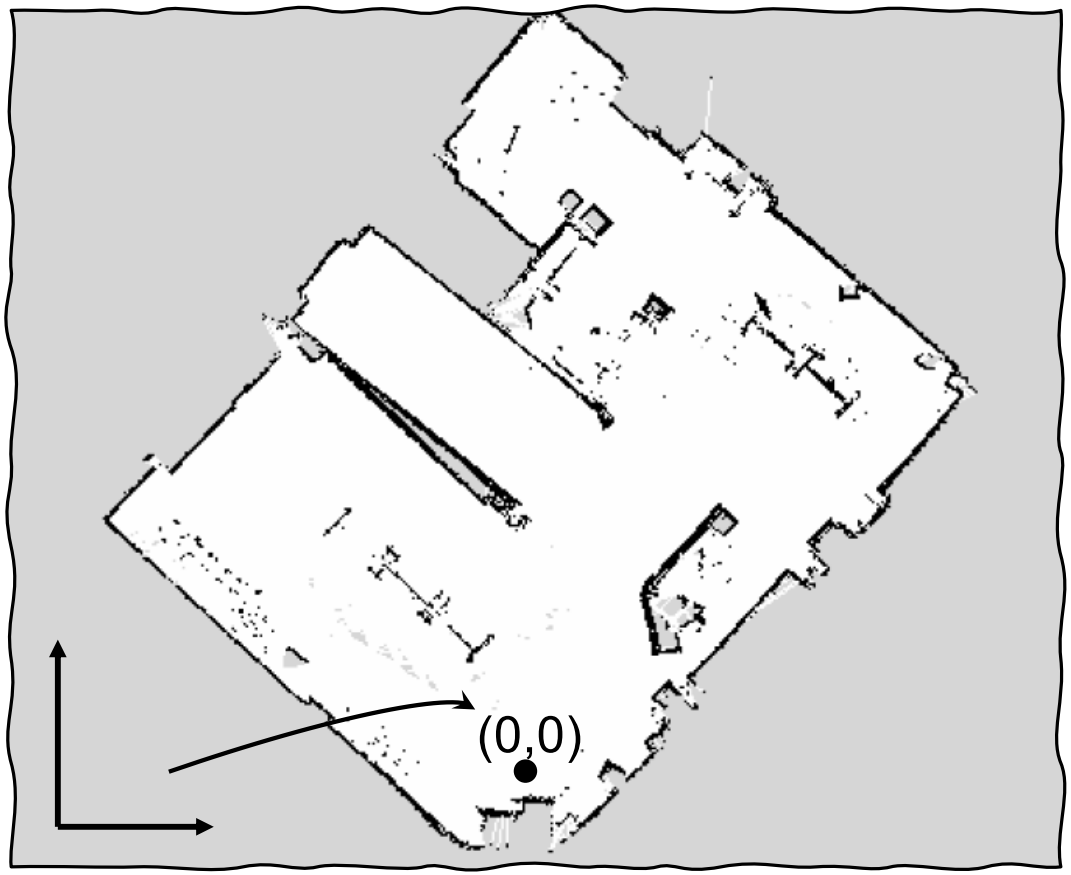


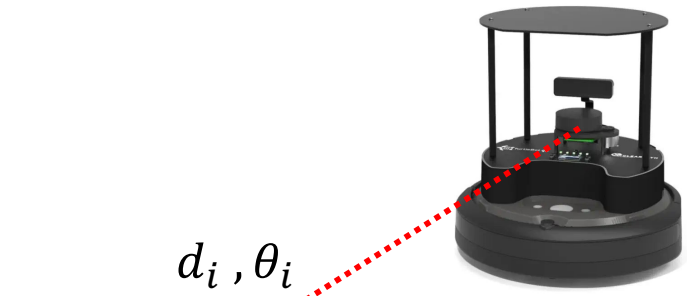
Figure 5.1
Where am I?

Mapping

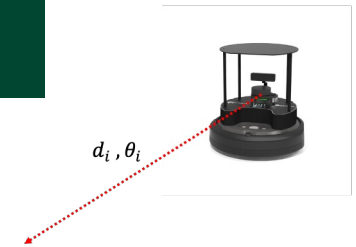

Map
Map Building and Maintenance
Refine Feature Parameters increase credibility
Add New Features extend map
Remove Offensive Features decrease credibility
Encoder
Prediction of Measurement and Position (odometry)
position estimate
Estimation (fusion) using confirmed map
predicted feature observations
matched predictions and observations
unexpected observations
unobserved predictions
YES
YES
Matching
NO
Unexpected Observation?
NO
raw sensor data or extracted features
Perception
Observation on-board sensors

The World Frame





Mapping: Adding to the World Frame



On Turtlebot:

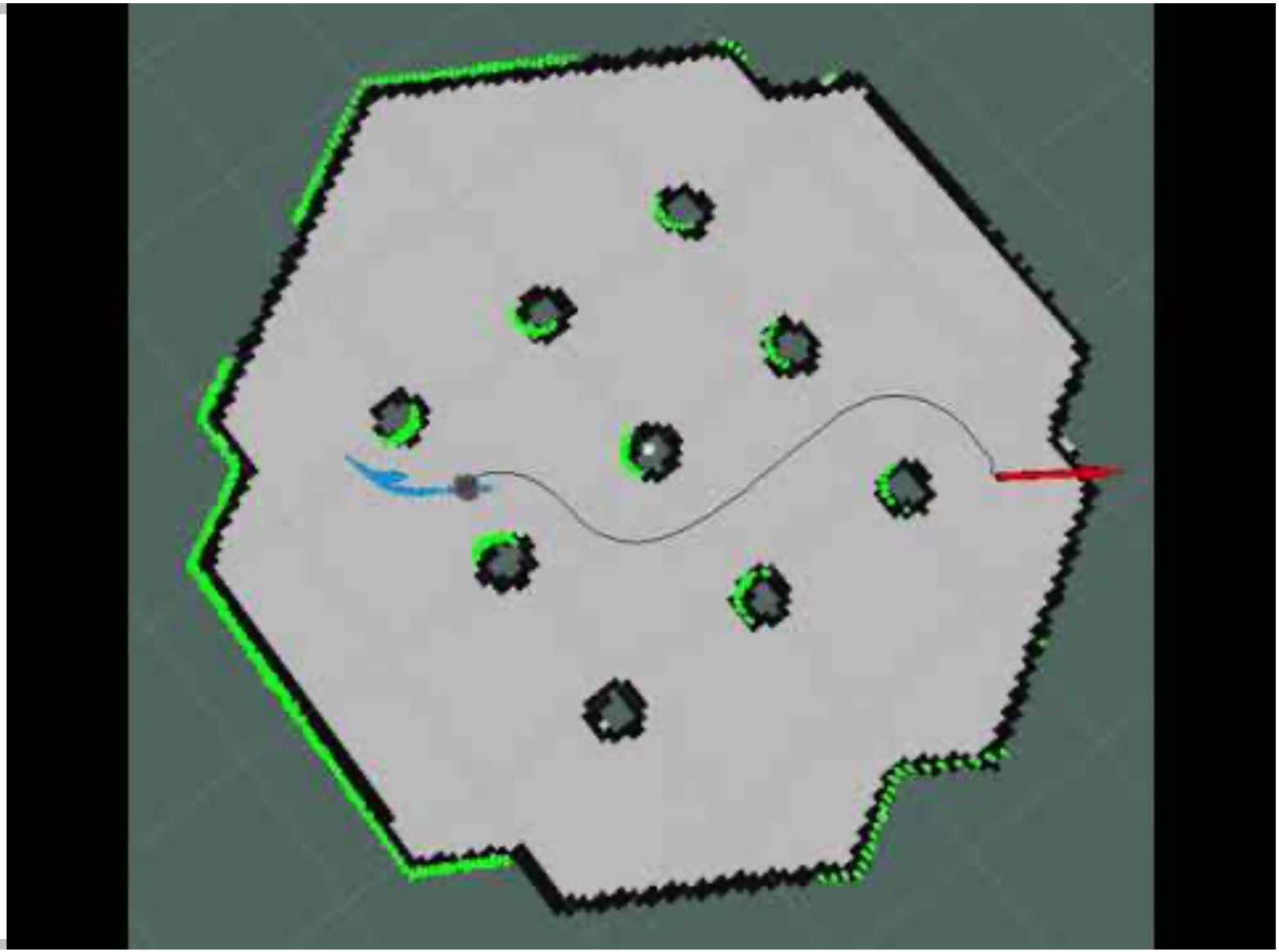
$p_2 = (dcos\theta, dsin\theta)$

$$p_1 = T_2^1 p_2$$

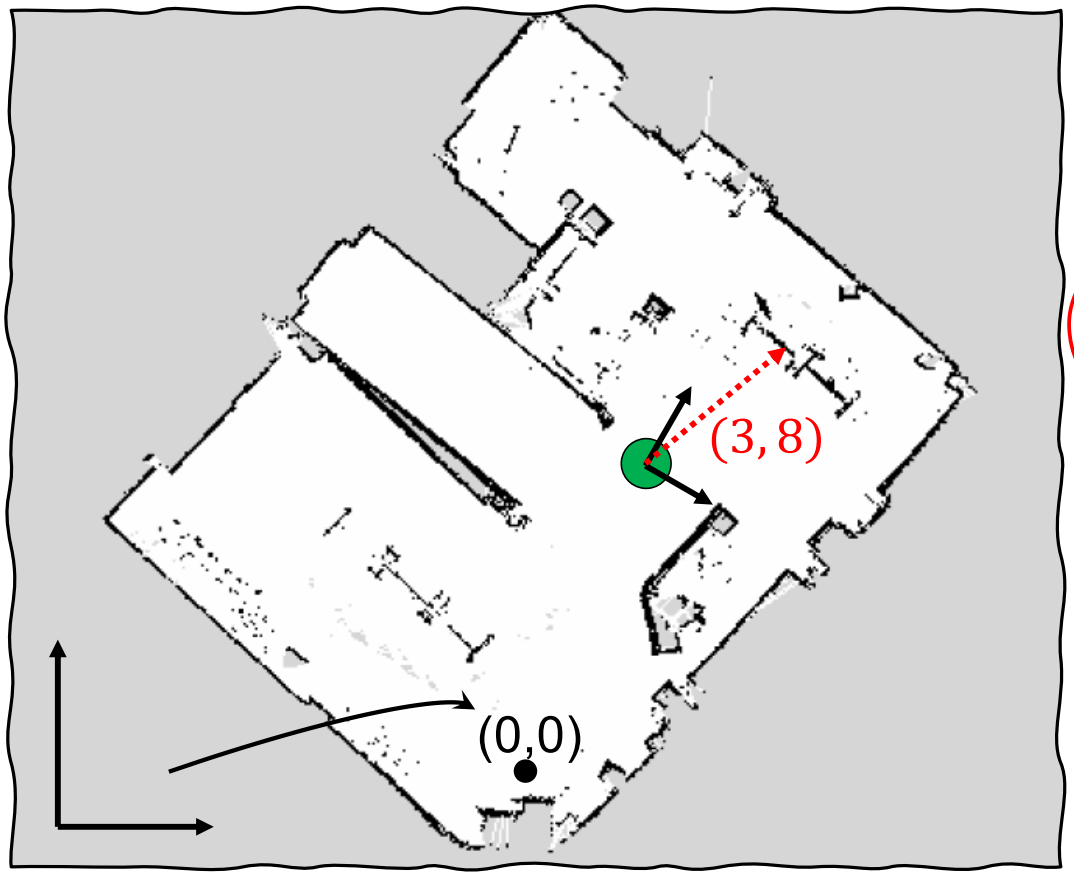
1) Find rigid transformation matrix from robot frame to world frame

2) Transform point in robot frame to world frame

3) Add to graph



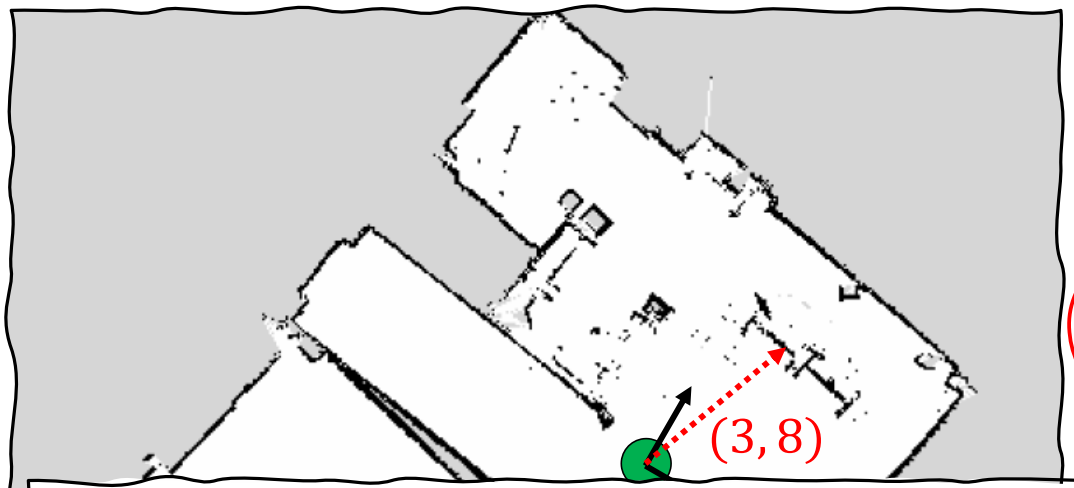
Mapping: Adding to the World Frame



$$p_1 = T_2^1 p_2$$

1) Find rigid transformation matrix from robot frame to world frame

Mapping: Adding to the World Frame

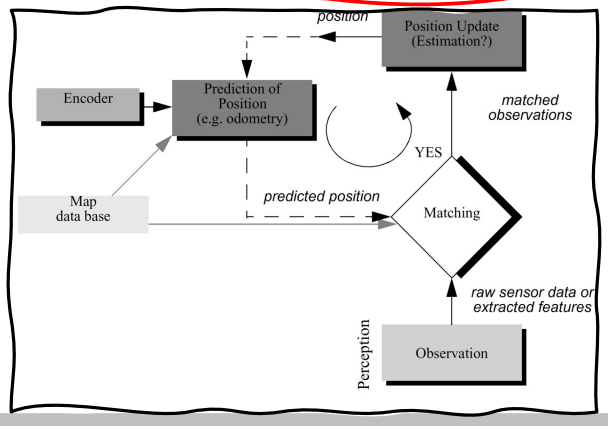


$$p_1 = T_2^1 p_2$$

1) Find rigid transformation matrix from robot frame to world frame

Localization
(aided by odometry)

$$p' = f(x, y, \theta, \Delta s_r, \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$



1. Lecture (~40 minutes)
2. Subsystem check-in (~20 minutes)

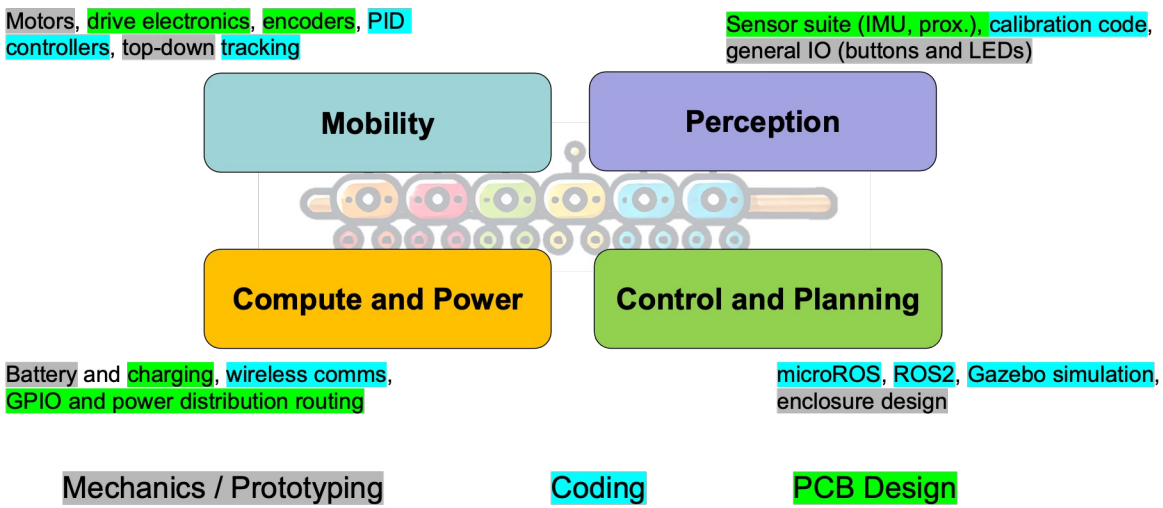
Questions?

Lecture 1: features, transformations, and mapping
Lecture 2: SLAM

693H Robotic Subsystems

First Design Review Presentations are 2/10, 2/12

- I will randomly select which two groups present which day. Be ready on 2/10.
- Link to presentation templates are **posted on Discord + Laulima**



693H Robotic Subsystems

Motors, drive electronics, encoders, PID controllers, top-down tracking

Sensor suite (IMU, prox.), calibration code, general IO (buttons and LEDs)

Mobility

Perception

Compute and Power

Control and Planning

Battery and charging, wireless comms, GPIO and power distribution routing

microROS, ROS2, Gazebo simulation, enclosure design

Mechanics / Prototyping

Coding

PCB Design