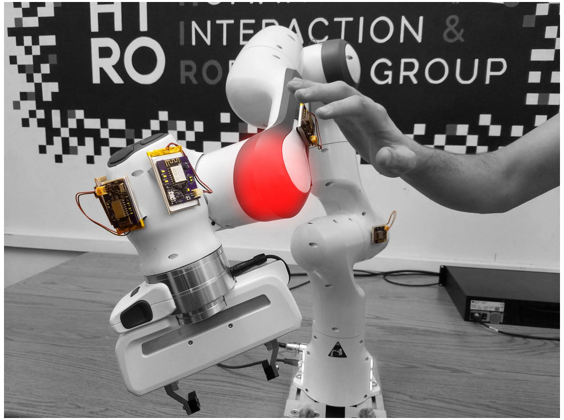
A Gentler Introduction to Robotics

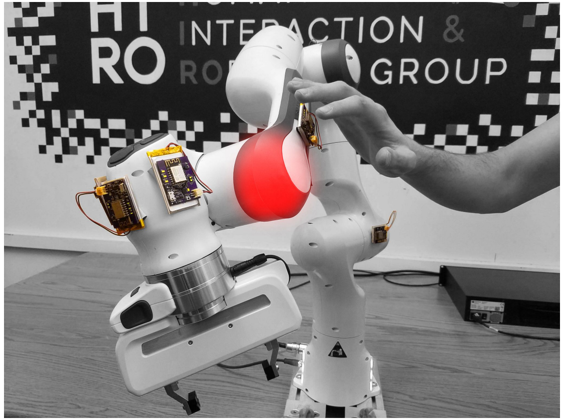


Matt **Strong**

Self-Introduction

- Former HIRO Group Researcher
- CRA Outstanding Undergraduate Researcher Award: **Honorable Mention**
- College of Engineering **Research Award**
- SWE@**Microsoft**
- Accepted to PhD programs at:
  - **Stanford**
  - **Cornell**
  - **UPenn**
  - **UT Austin**
  - Morale of the story: join the HIRO Group

1. My Research at a High Level (stop me if it gets confusing)
2. Which Fundamentals Got Me There



Matt **Strong**

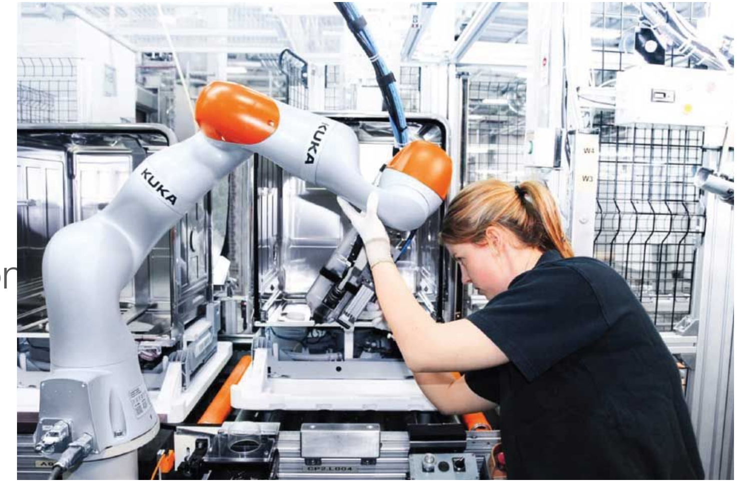
Introduction: The Problem

- Robotics currently exist at a large scale in **industrial/manufacturing** environments
- Humans work around robotics, robots don't work around humans
- But, we need to drive the **transition** from industrial to environments with people

Robots working on a car

Introduction: Nearby Space Perception

- In these environments, **extended**, **close proximity** human robot collaboration is essential
- To achieve this, first step: **Perception.** But there's some problems.
    - External, sparse, high-resolution sensing -- occlusion problem
    - Onboard, contact-based sensors
- Solution (and **Contribution**): Whole Body Distributed Sensing is **key**

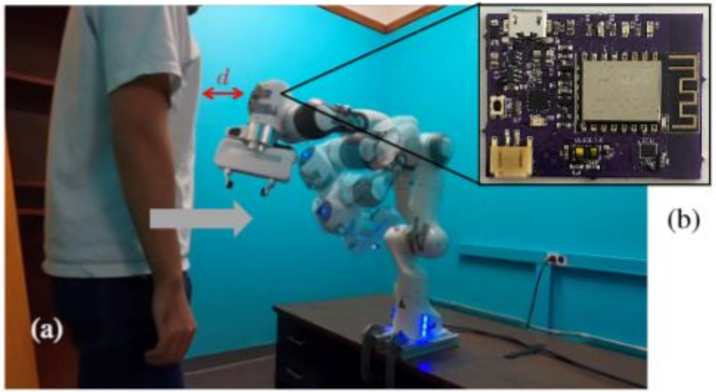


A human and robot collaborate

Contribution 1

- **Problem 1**: Current Whole Body Sensing lacks a certain degree of modularity, accuracy, and ease of use
- **Solution 1**: A new plug-and-play robotic skin system for calibration, demonstrated on a real avoidance example

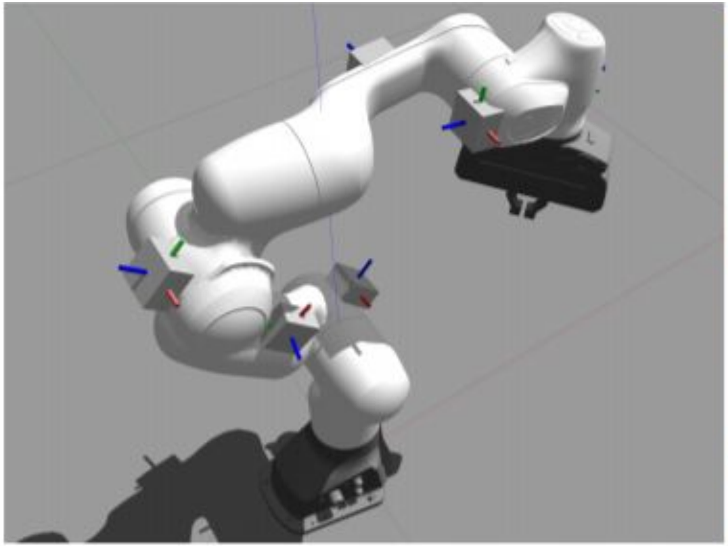
A robot avoids a human with the calibrated skin units.



A Plug and Play Robotic Skin

- Goal: Automatically calibrate the skin units along a robot's body

Result of Calibration



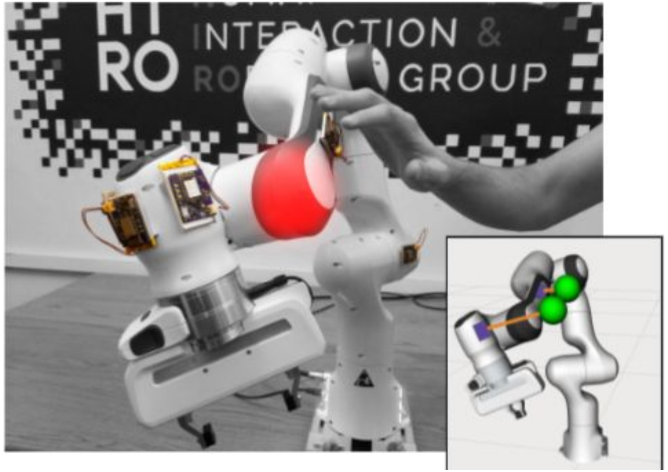
Actual skin unit poses

A Plug and Play Robotic Skin

Contribution 2

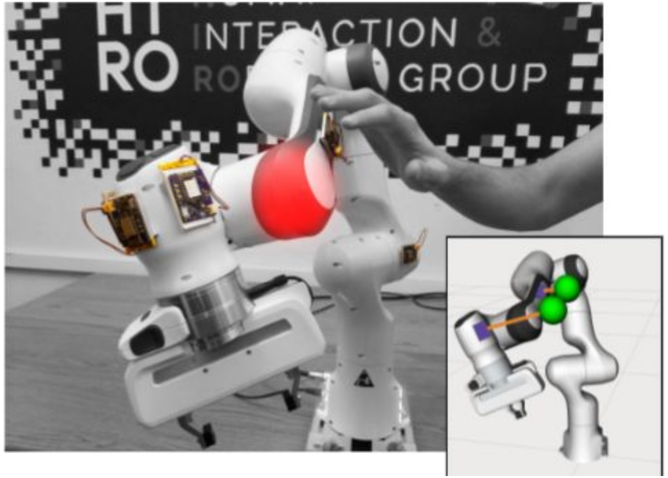
- **Problem 2**: Lack of a smooth transition between avoidance and (desirable) contact
- **Solution 2**: Implicit contact anticipation via those same onboard sensor units

Under our framework, a robot can anticipate contact with the SUs.

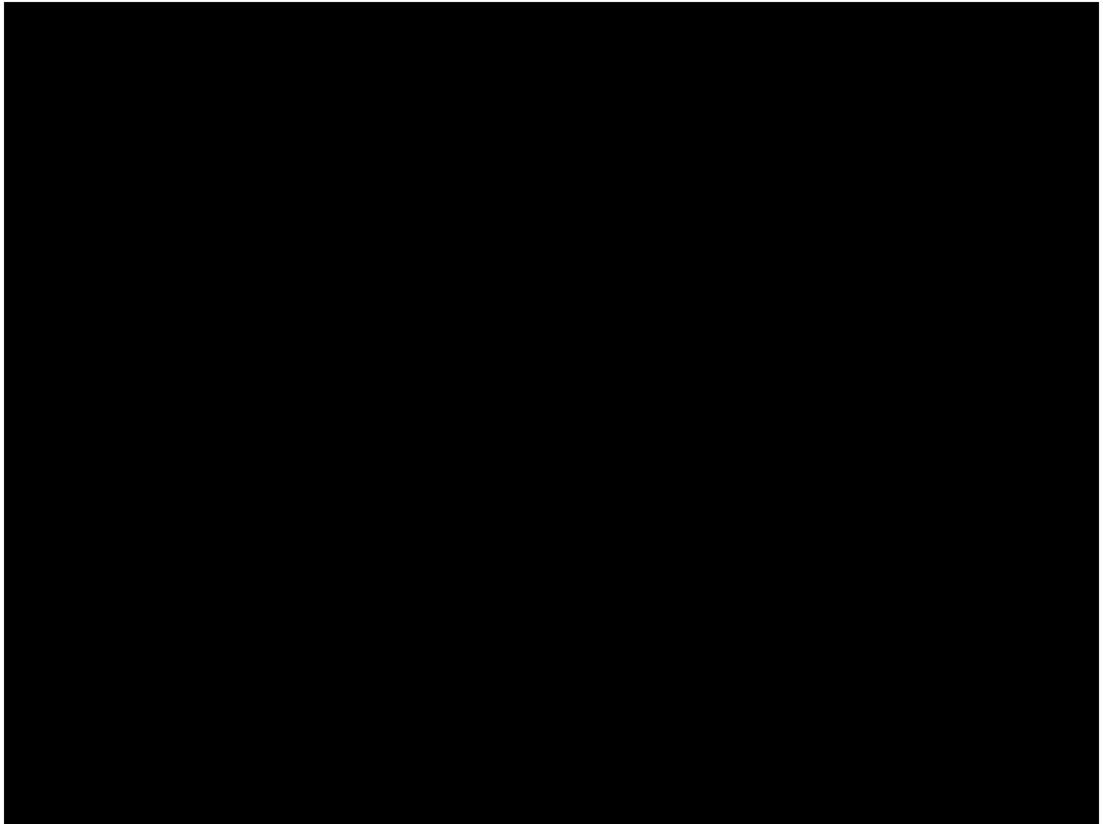


Implicit Contact Anticipation via Distributed Whole-Body Sensing

- Goal:
  - Enable the transition from avoidance to contact using whole body, nearby space perception

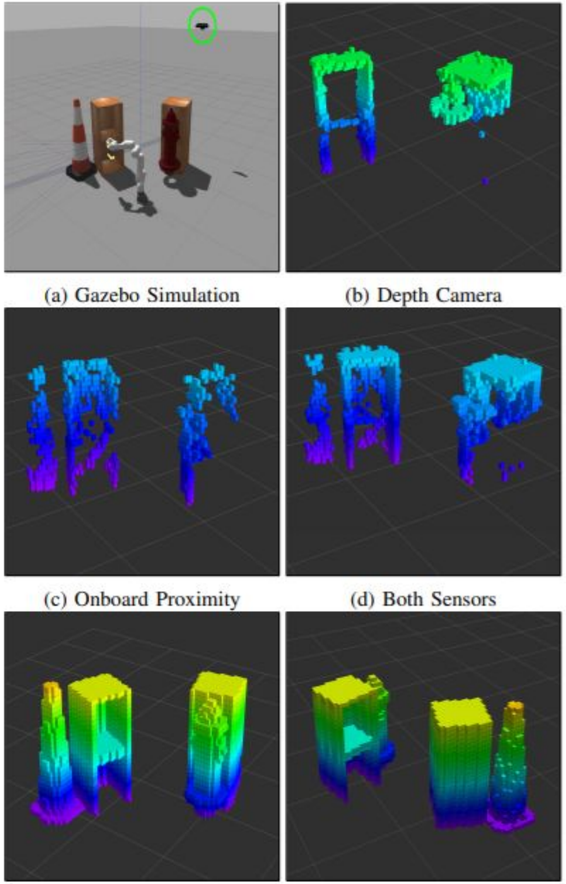


Framework: The robot slows down before contact and is able to make contact, or avoid



Mapping

- We can also do mapping with all of these sensors!
- Safety = you need to know what's around you = you need a precise and accurate map of your nearby space



(a) Gazebo Simulation (b) Depth Camera (c) Onboard Proximity (d) Both Sensors (e) Ground Truth Front (f) Ground Truth Back

How Did I Get There?

Me

Recommended Study Plan

- Go through https://github.com/Introduction-to-Autonomous-Robots/Introduction-to-Autonomous-Robots
- You can compile it or check out a PDF version under “Releases”

Recommended Study Plan

- Go through https://github.com/Introduction-to-Autonomous-Robots/Introduction-to-Autonomous-Robots
- You can compile it or check out a PDF version under “Releases”

The Foundations of Robotics: Coordinate Systems

Coordinate Systems

Some object in the robot's frame

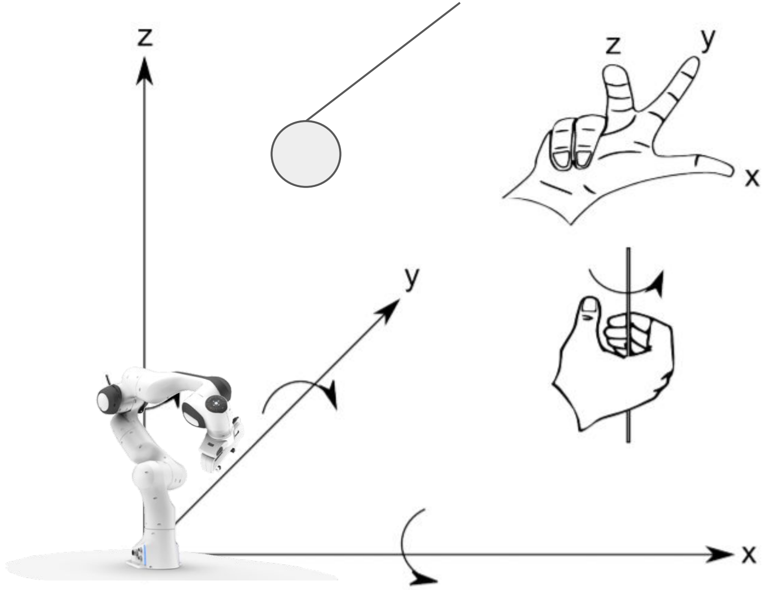
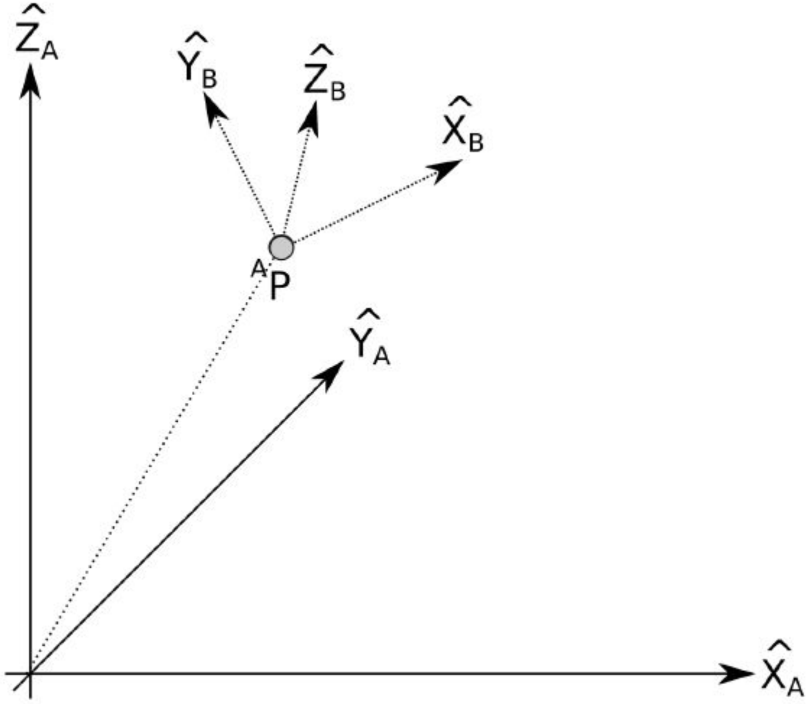


Figure 3.1.: *A coordinate system indicating the direction of the coordinate axes and rotation around them. These directions have been derived using the right-hand rules.*

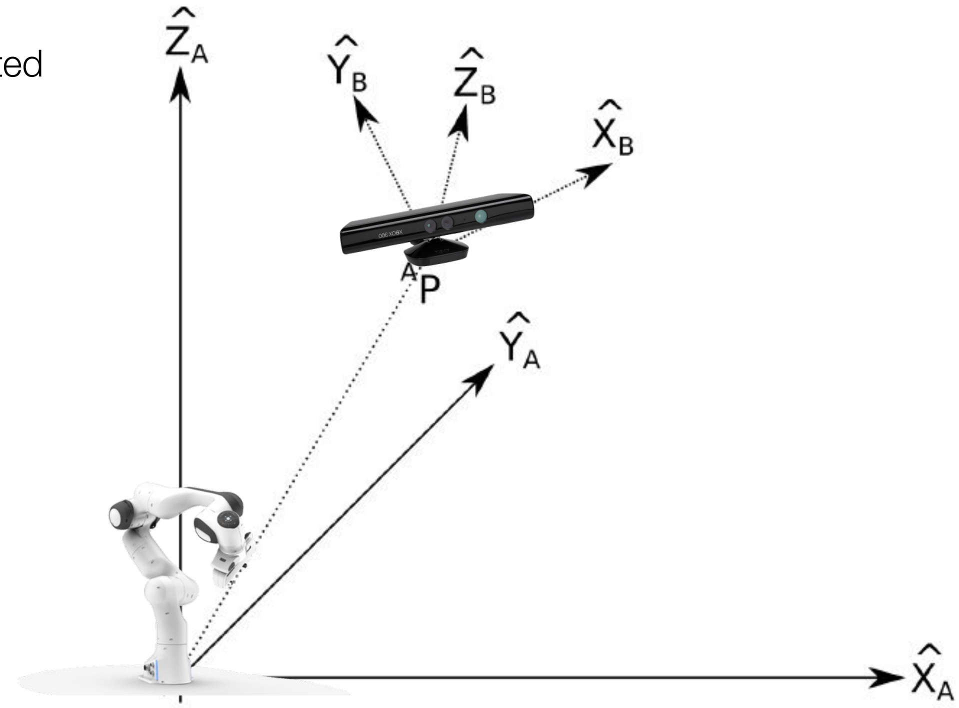
Coordinate Frames

Objects, robots, etc are associated with a coordinate frame



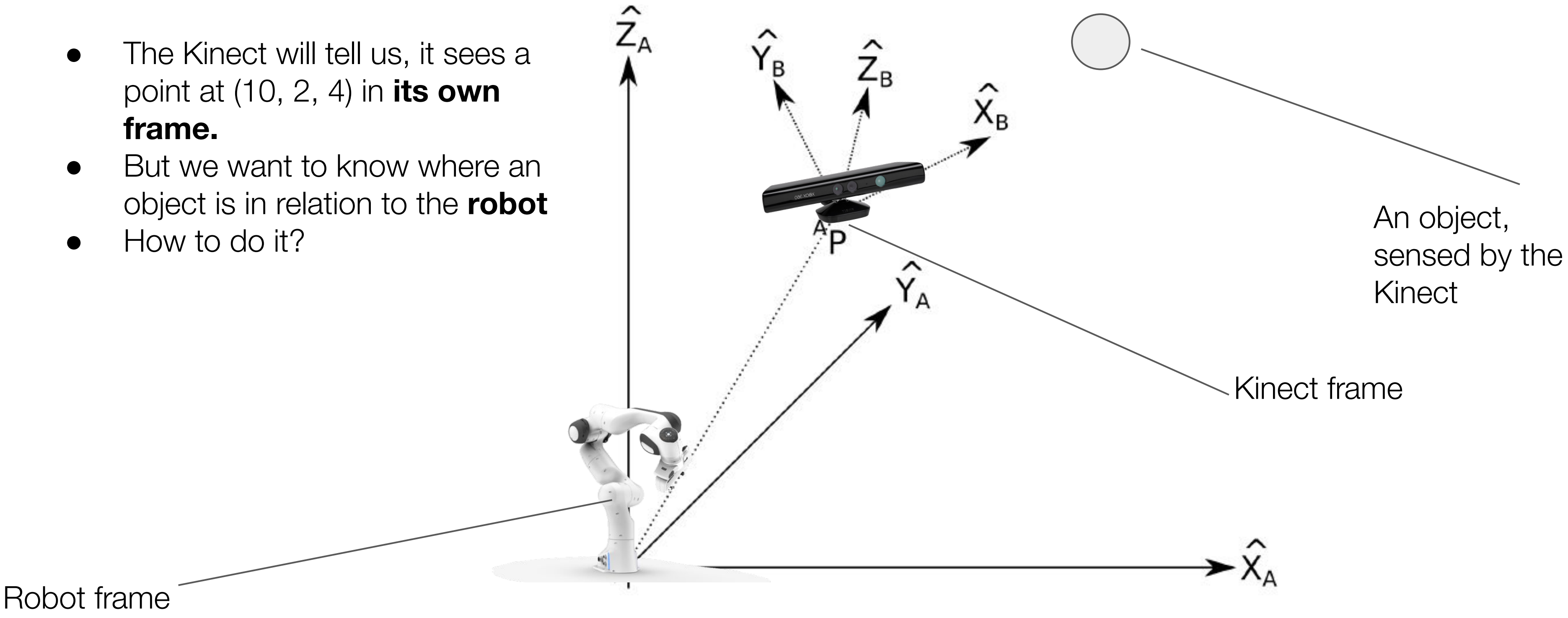
Coordinate Frames

Objects, robots, etc are associated with a coordinate frame



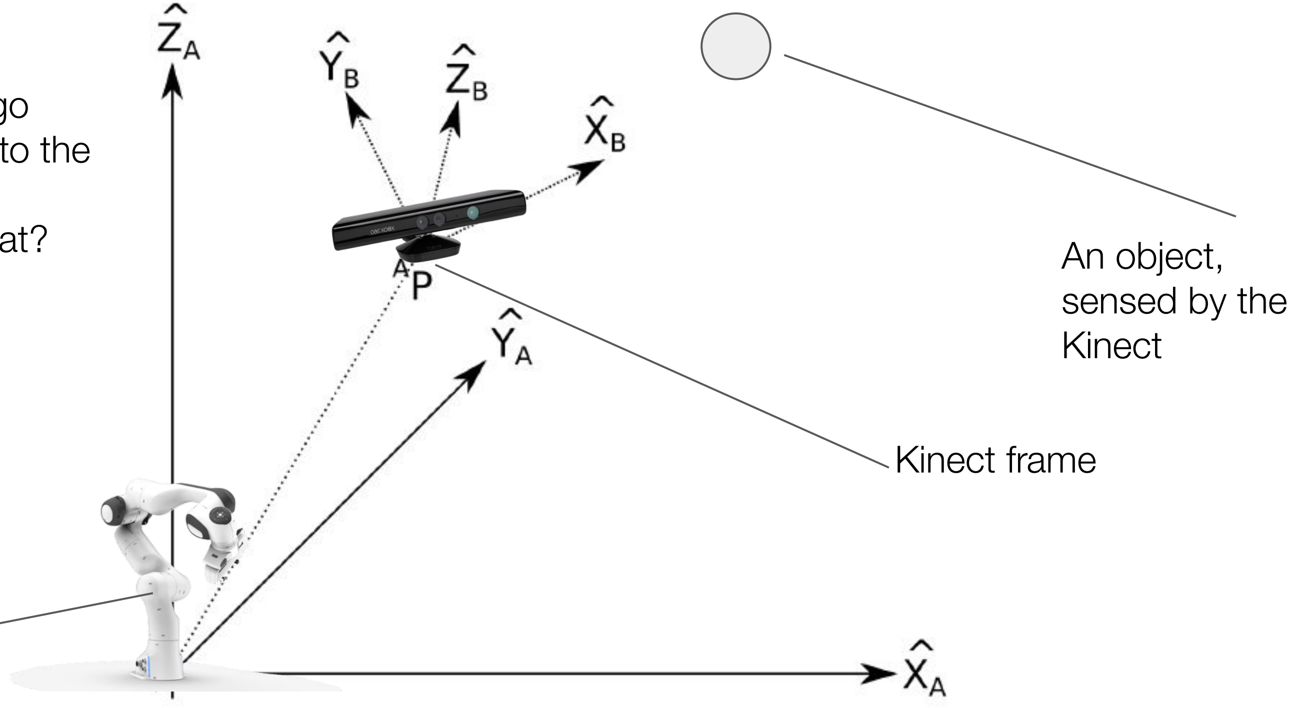
Coordinate Frames

- The Kinect will tell us, it sees a point at (10, 2, 4) in **its own frame.**
- But we want to know where an object is in relation to the **robot**
- How to do it?

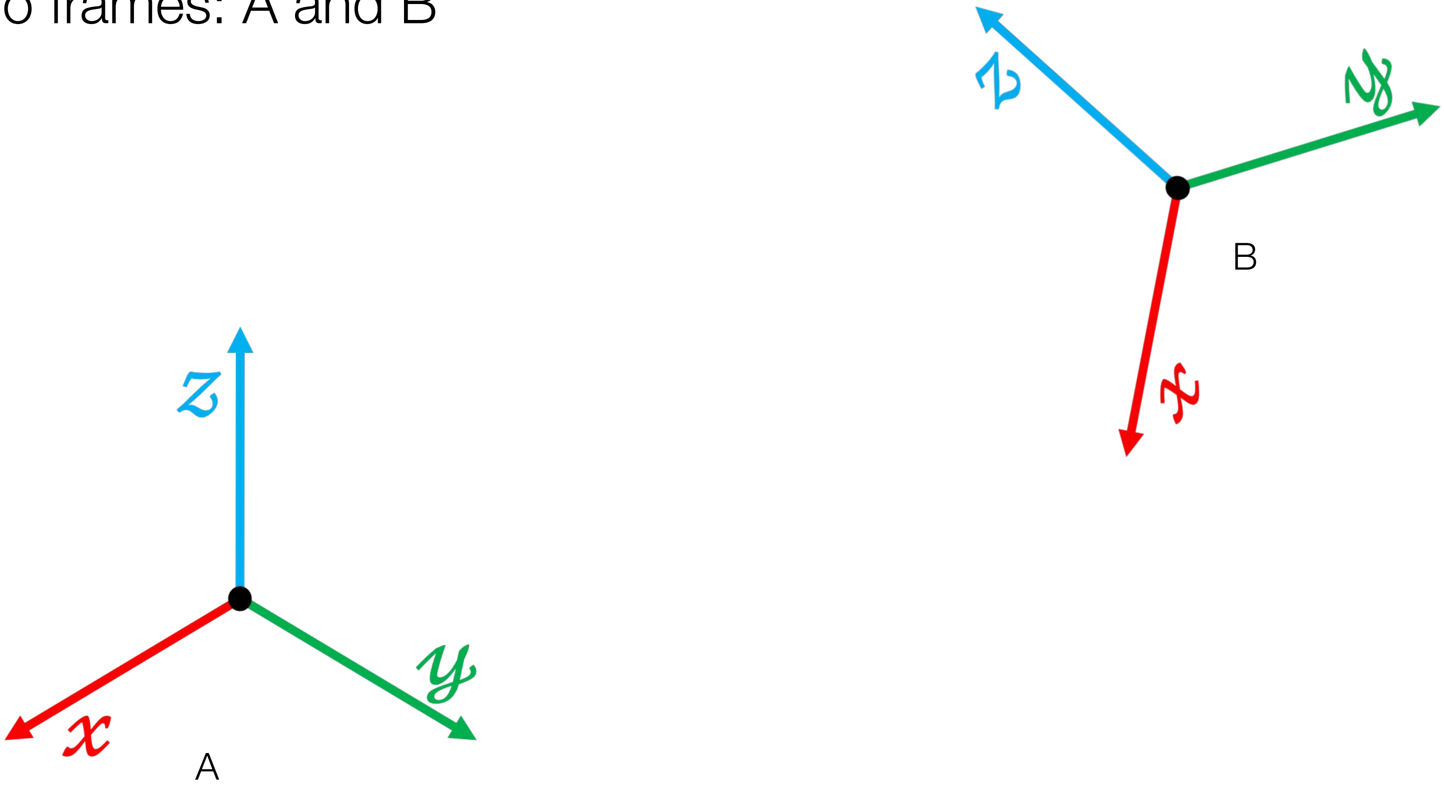


Coordinate Frames

- Recipe:
- Step 1: Figure out how to go from the robot base frame to the kinect base frame
    - Wait … how to do that?

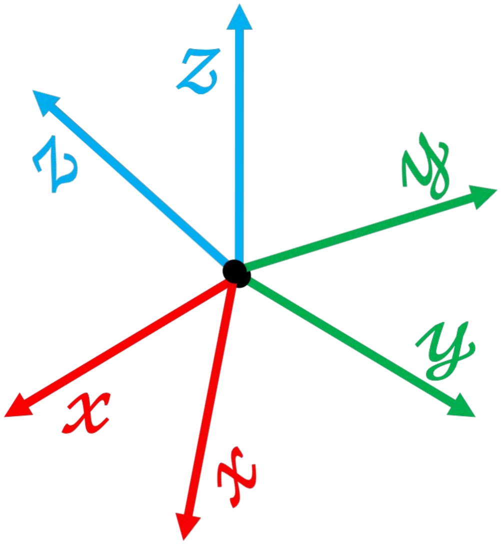


Two frames: A and B



Move A up and to the right to have the same **position** as B

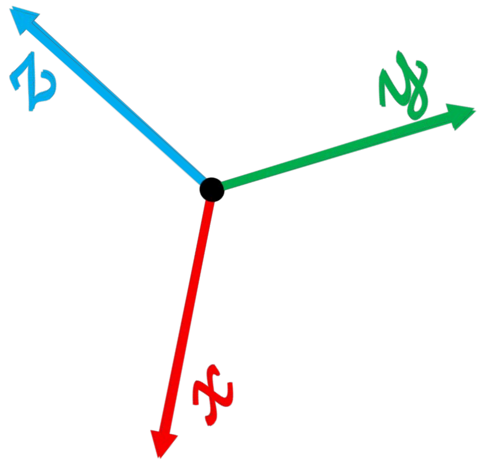
The same thing as adding a 3-d vector to A's position.



Rotate A to align with B

This requires a 3x3 matrix, called a **rotation matrix**

Doesn't change the position, but changes the orientation



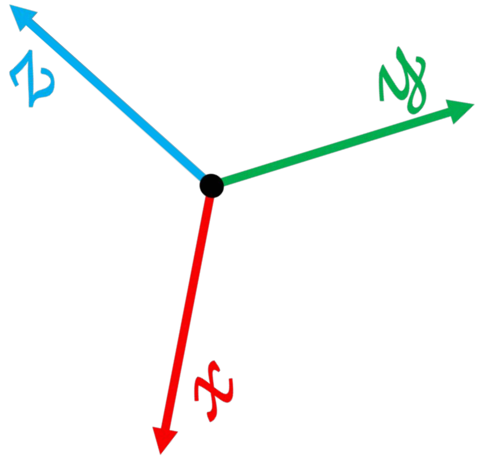
A point in frame B

$$\left[{}^{A}Q\right] = \left[\begin{array}{ccc|c} & {}^{A}_{B}R & & {}^{A}P \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} {}^{B}Q \\ 1 \end{array}\right]$$

A point in frame A

Rotating to frame A

How to go from frame A's origin to frame B's origin

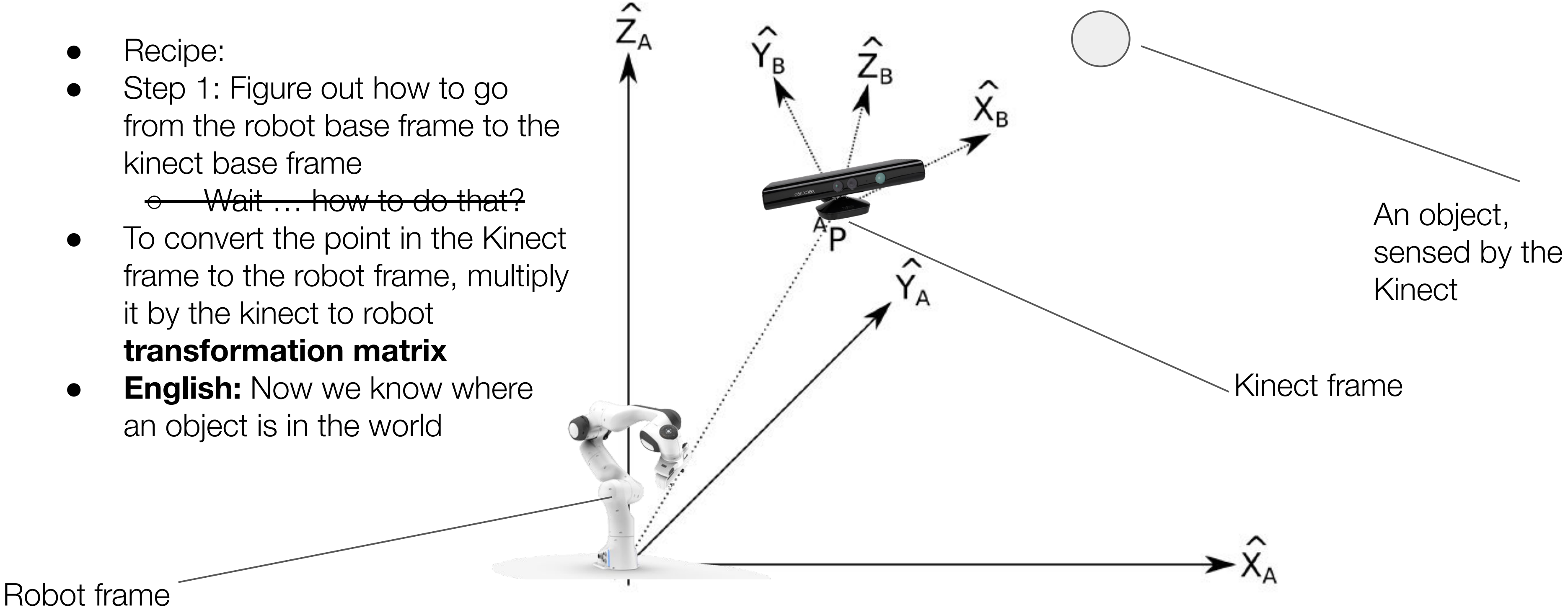


$${}^{A}Q = {}^{A}_{B}T\,{}^{B}Q$$

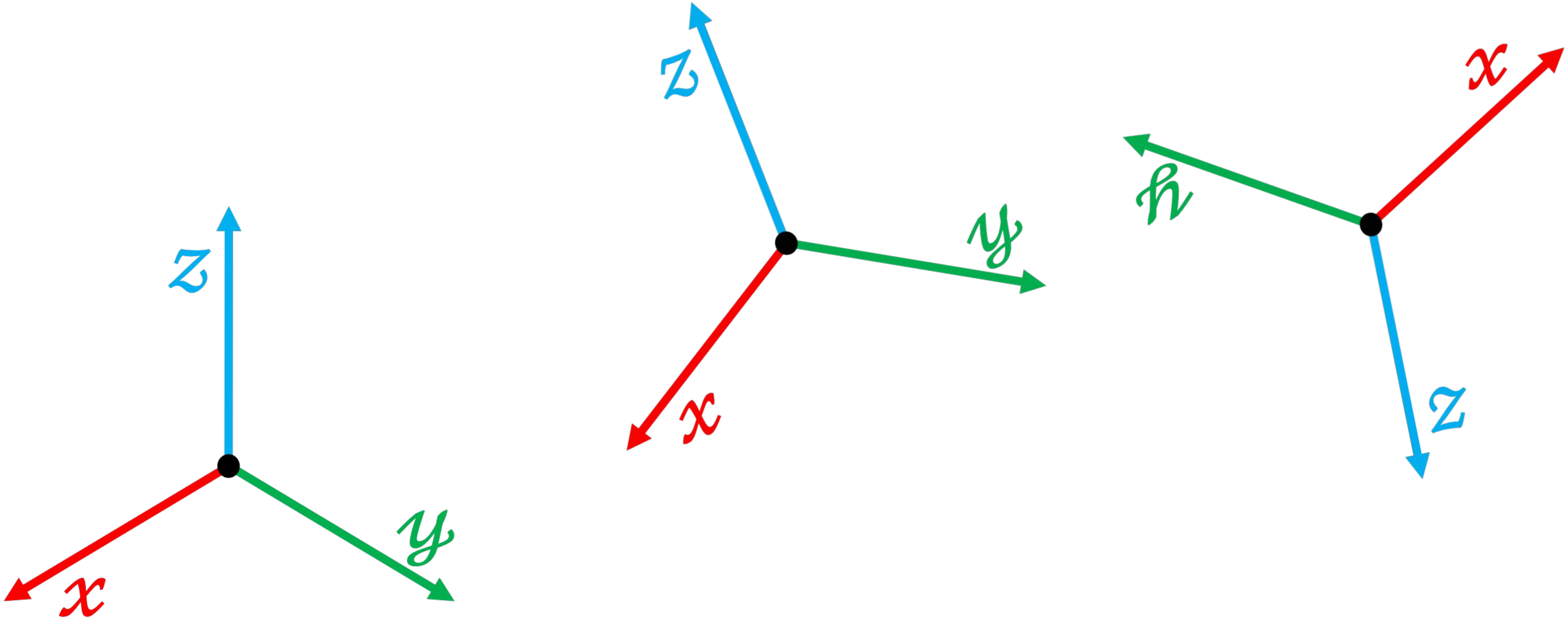
Transformation matrix from B to A

Coordinate Frames

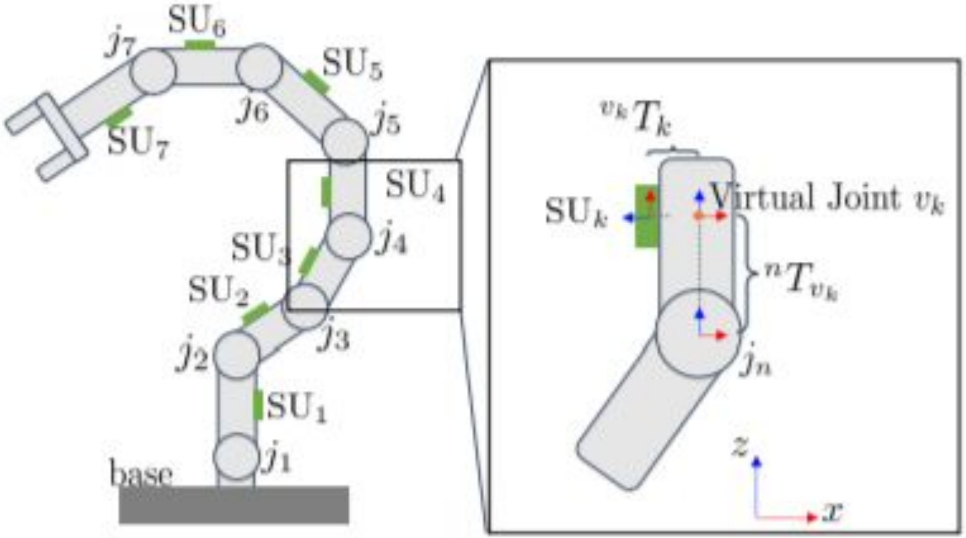
- Recipe:
- Step 1: Figure out how to go from the robot base frame to the kinect base frame
    - ~~Wait … how to do that?~~
- To convert the point in the Kinect frame to the robot frame, multiply it by the kinect to robot **transformation matrix**
- **English:** Now we know where an object is in the world



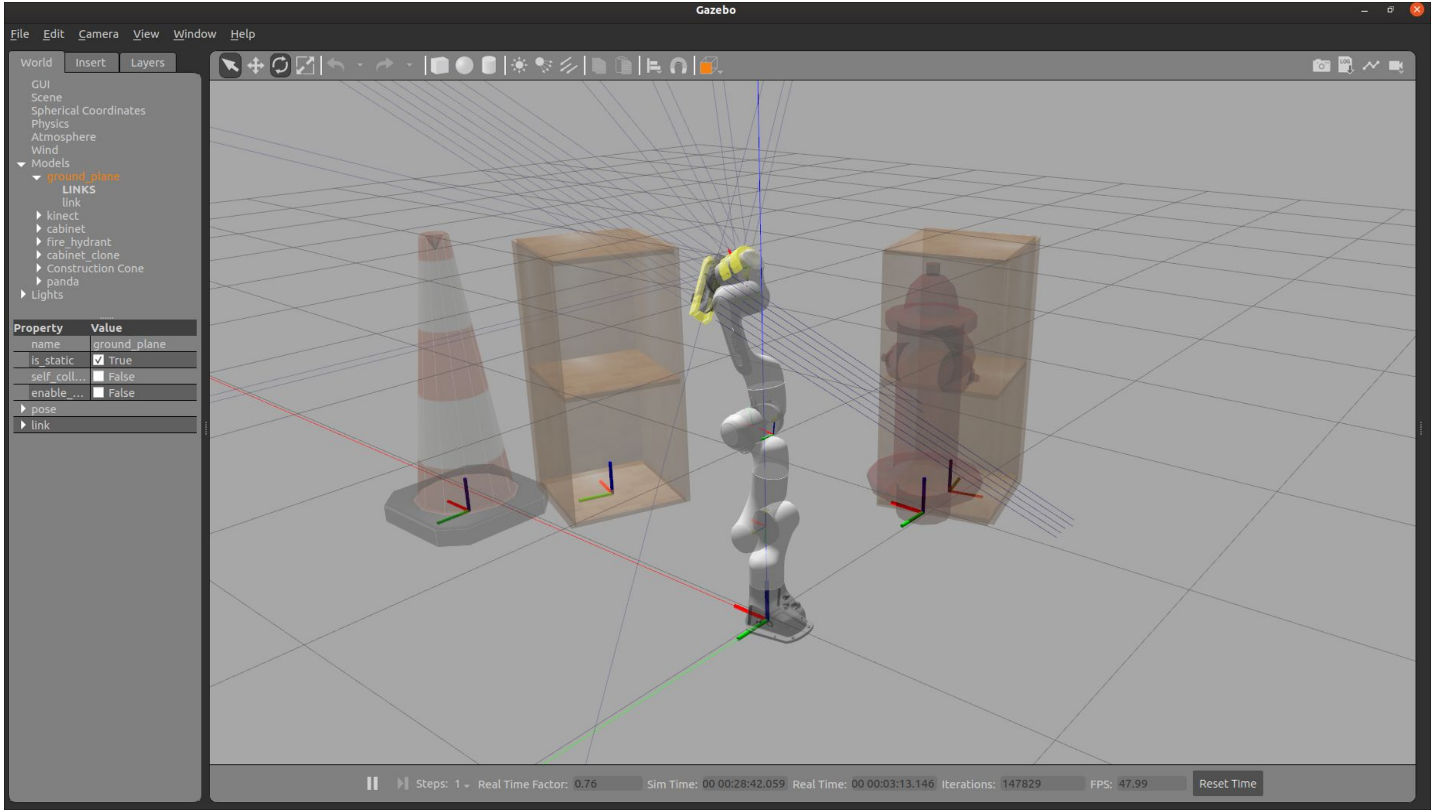
You can chain transformations together: to go from C to A, you just multiple transformations from C to B, and then B to A!



I made use of this in one of the papers I helped write!



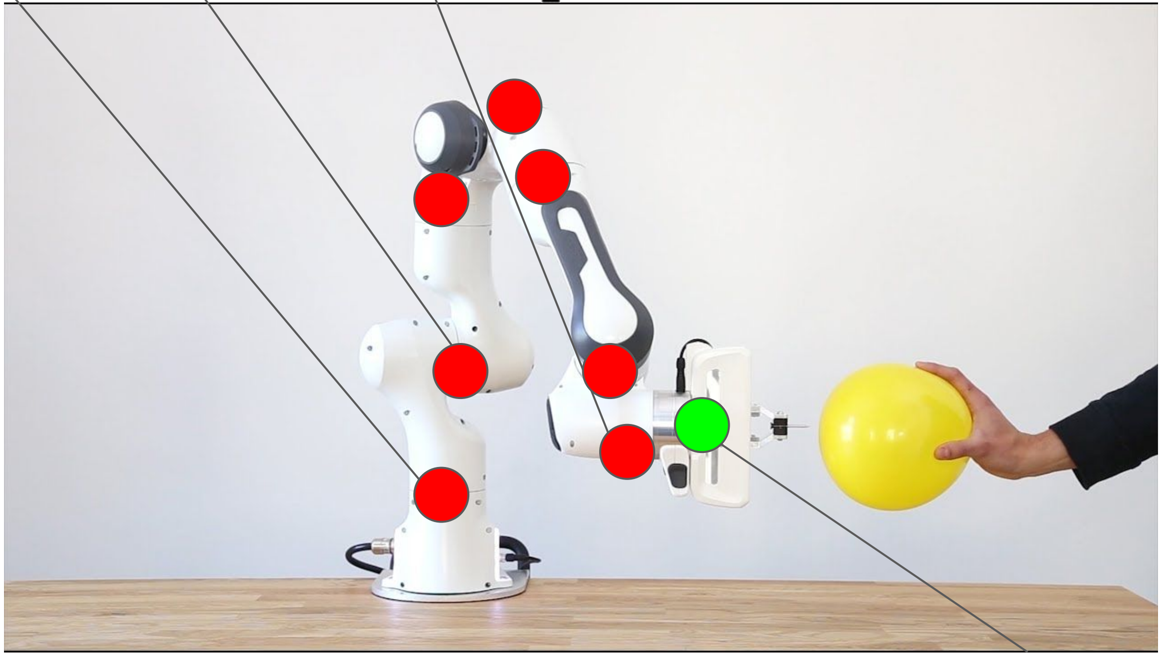
From My Research:



Forward Kinematics and Inverse Kinematics

- Forward Kinematics
  - Given where the robot's joint positions are, where are we with respect to the world frame?
  - You can use transformation matrices here! To figure out where the end-effector is in the base/world frame, multiply transformations as so:
    - Base frame to joint 1 frame
    - Joint 1 frame to joint 2 frame
    - Joint 3 frame to joint 4 frame
    - …..joint n frame to end-effector frame!
  - You can apply this to **any** robot

$$[j_1, j_2, \dots j_n]^T$$

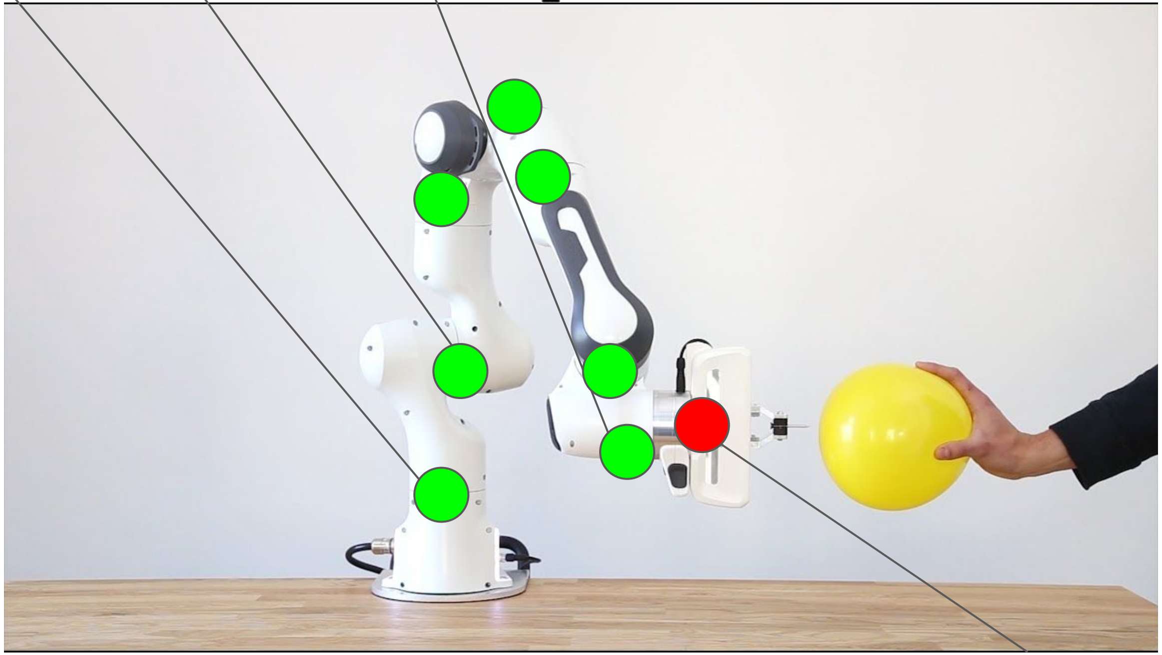


$$[x, y, z, r, p, y]^T$$

Forward Kinematics and **Inverse Kinematics**

- Inverse Kinematics
  - Given where we want to go into the world, what joint configurations will get us there?
  - Analytical approaches
    - Closed form solution
    - Get very hard with increased complexity
  - Numerical approaches
    - Iterative, optimization based
    - Work for more complex kinematics (ML also uses optimization for complex problems!)

$$[j_1, j_2, \ldots j_n]^T$$



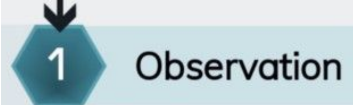
$$[x, y, z, r, p, y]^T$$

Degrees of Freedom

The concept of degrees-of-freedom, often abbreviated as DOF, is important for defining the possible positions and orientations a robot can reach.

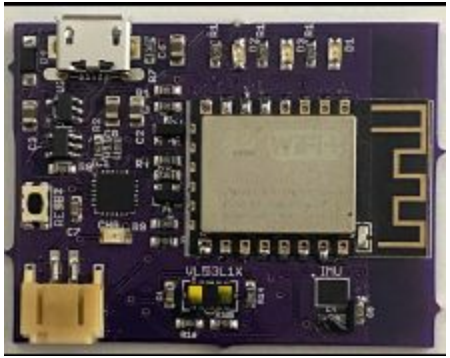
The Franka Panda has 7 joints and 7 degrees of freedom = **redundancy**

The Robotic Pipeline



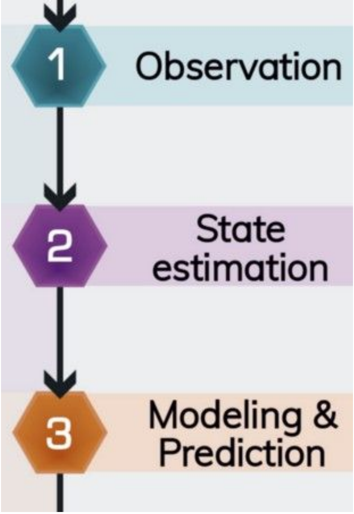
Sensors

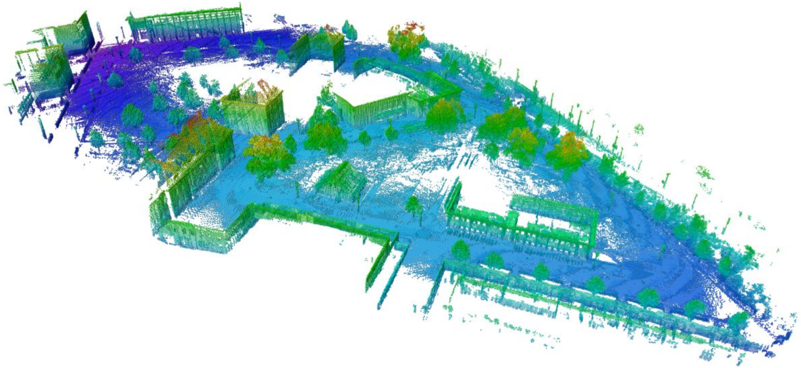
- Get data from real world (or simulated observations)
- Lidar: 3-d point cloud
- Proximity sensors: single points
- IMU: acceleration and gyroscope data
- Robots have their own sensors that can detect joint positions, velocities, accelerations, and more



Perception

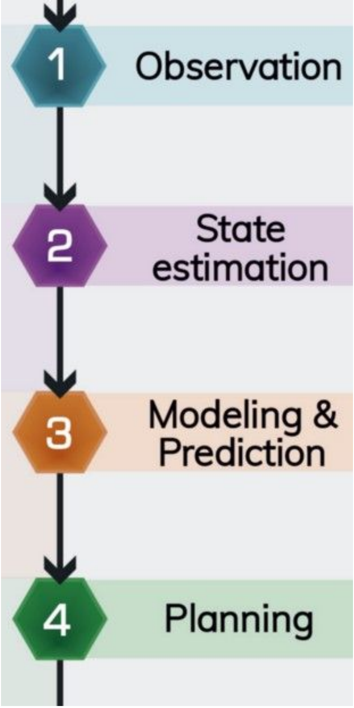
- Where are you in the world, and where are other things?
- Knowing the robot's joint information = you can compute where the robot is
- Noisy sensor data though….
  - Probabilistic formulations
  - If you have **two** noisy sensors, if they agree on similar points, it's better than **one** noisy sensor
  - Under the hood: lots of transformations to get everyone on the same page (frame)!
- Prediction: predicting future states

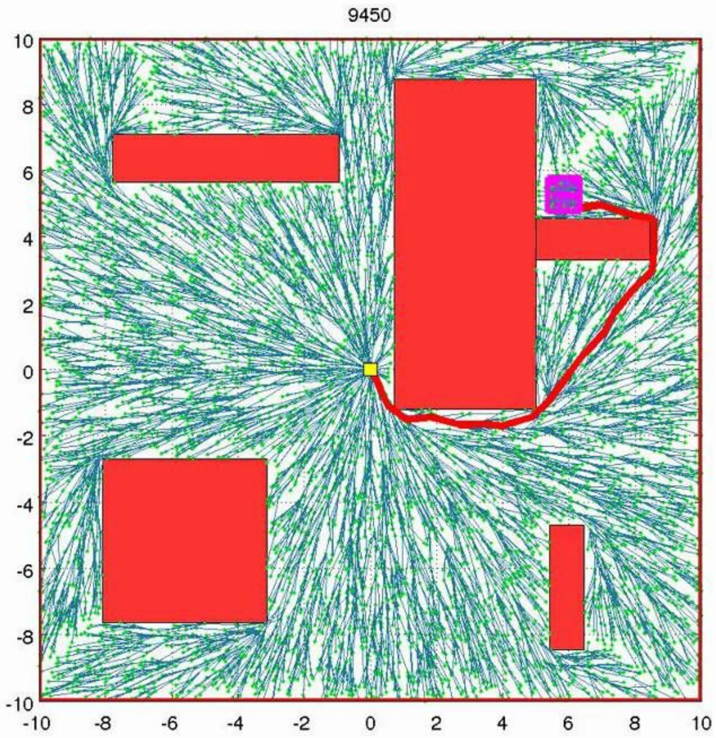


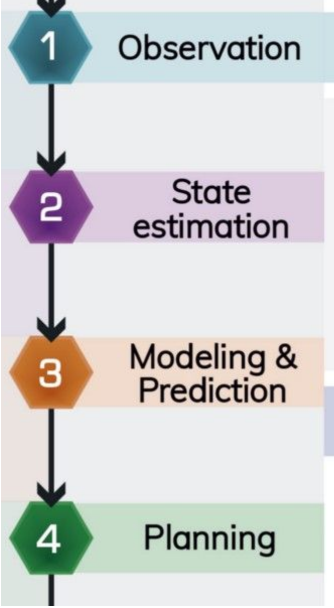


Planning

- We know the environment and where we want to go!
- Construct a path!
- MANY ways of achieving this
- Creates a trajectory of achievable points for the robot to follow that will get it to a desired goal and not crash

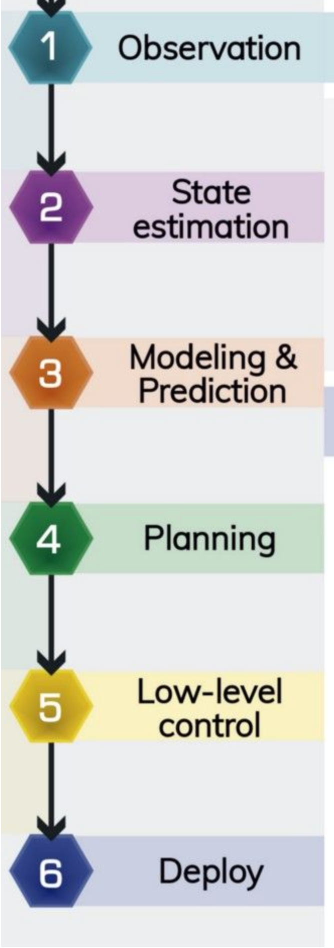



9450
10
8
6
4
2
0
-2
-4
-6
-8
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10


1
Observation
2
State estimation
3
Modeling & Prediction
4
Planning

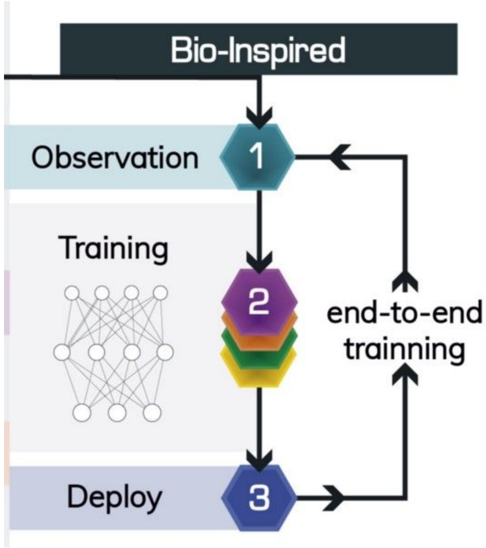
Robotic Control

- **Joint** Space Control:
  - How much to move joints, how fast, how much torque
  - Drones: how much thrust to send to each propellor
  - Wheeled robots: amount of acceleration to apply
- **Operational** (Task) Space Control
  - I want to move the robot to point (x,y, z)
  - The robot moves in a way that makes sense to a human
  - This approach is “**task**” oriented
  - We use IK to go from task space to joint space, which the robot can understand



ML+Robotics

- Approaches in ML and Robotics simply force the robot to learn a variety of behaviors from data!



But Here's What Can Happens With a Combination of Both:

Questions?

- Matt Strong
- matthew.h.strong@gmail.com
- Github: **peasant98** (I'll follow you if you follow me)
- If you're interested in opportunities at **Microsoft**, send me your resume to mattstrong@microsoft.com and I will refer you if you're a good fit.