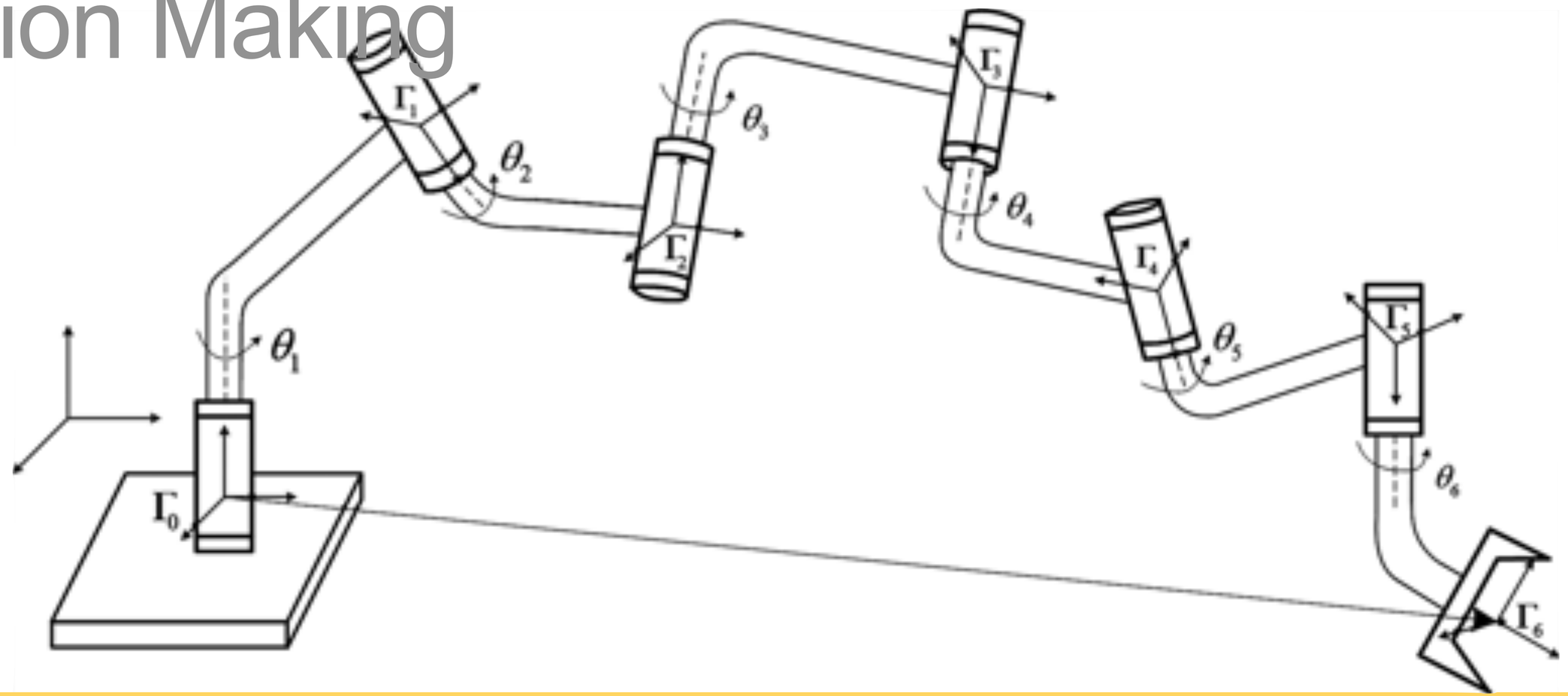
Lecture 05
Manipulation - I
Forward Kinematics
& Decision Making



Course Logistics

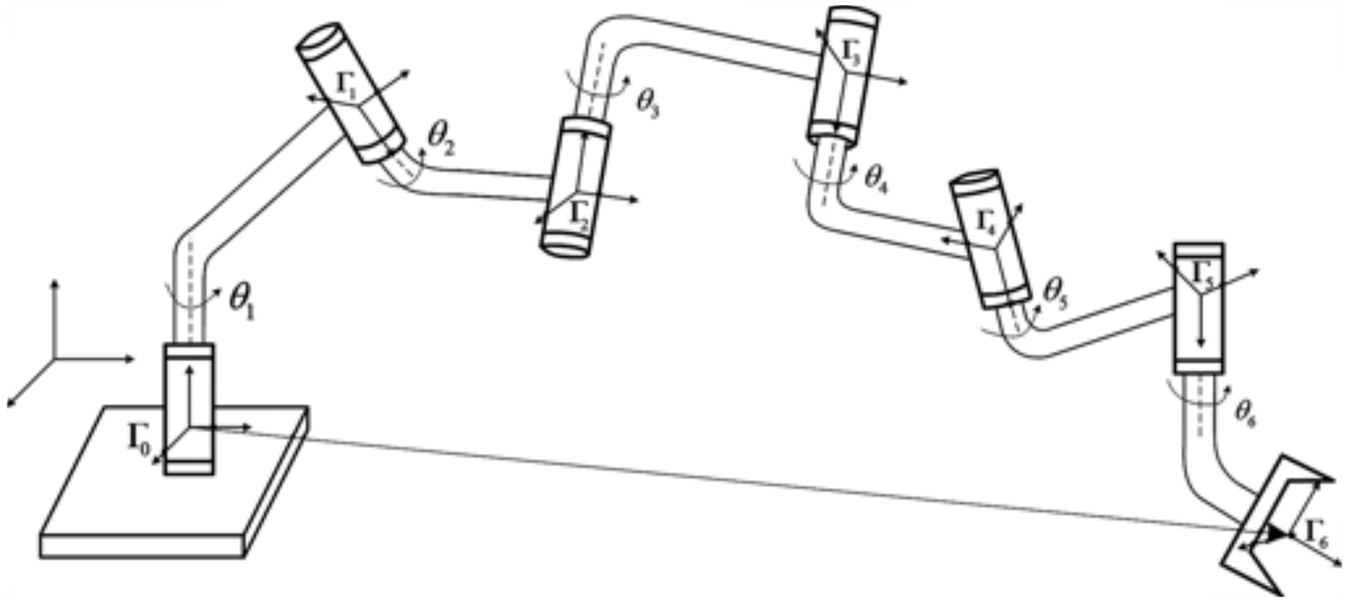
- **Quiz 3 was posted today and was due before the lecture.**
- Project 0 was posted on 09/13 and will be due **09/25 (extended)**.
- Autograder will be made available for you to test and submit your code.
  - **CSE-IT is still working on it. Will be made available soon.**
- Project 1 will be posted on 09/20 **(today)** and will be due 10/02.

Robot Kinematics

Goal: Given the structure of a robot arm, compute

– **Forward kinematics**: infer the pose of the end-effector, given the state of each joint.

– **Inverse kinematics**: infer the joint states to reach a desired end-effector pose.



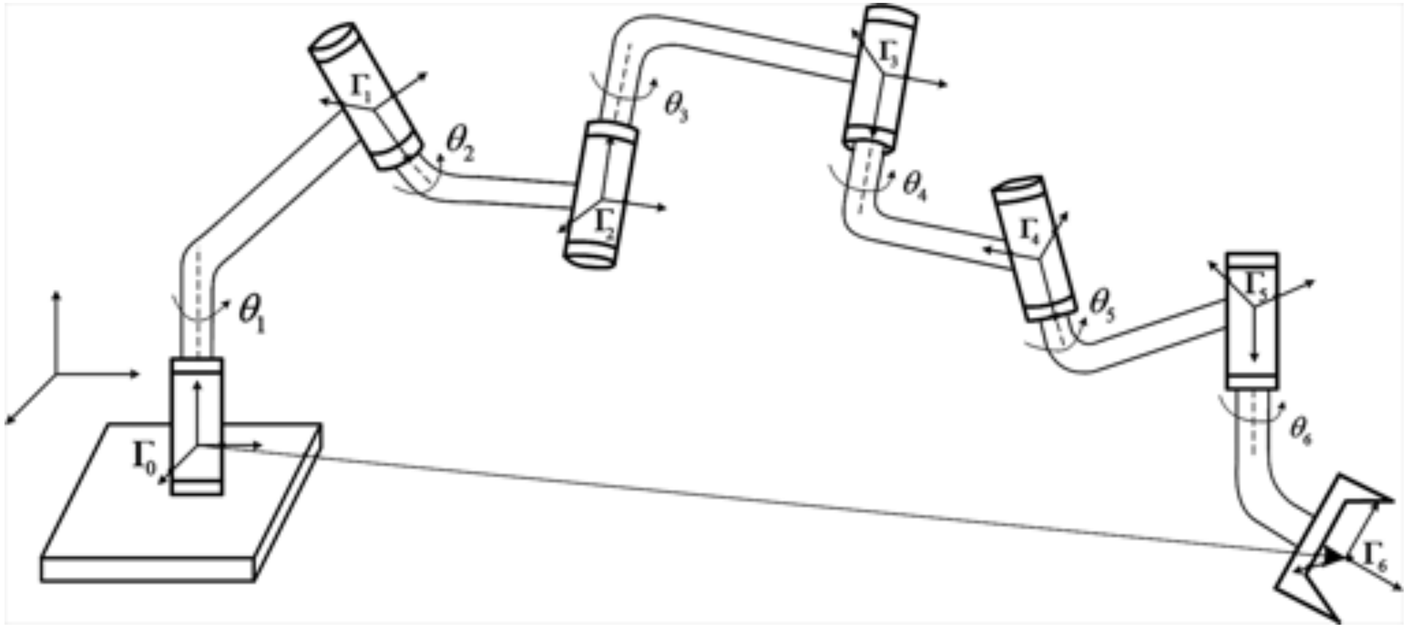
Robot Kinematics

– **Forward kinematics**: infer the pose of the end-effector, given the state of each joint.

Infer: pose of each joint and link in a common world workspace

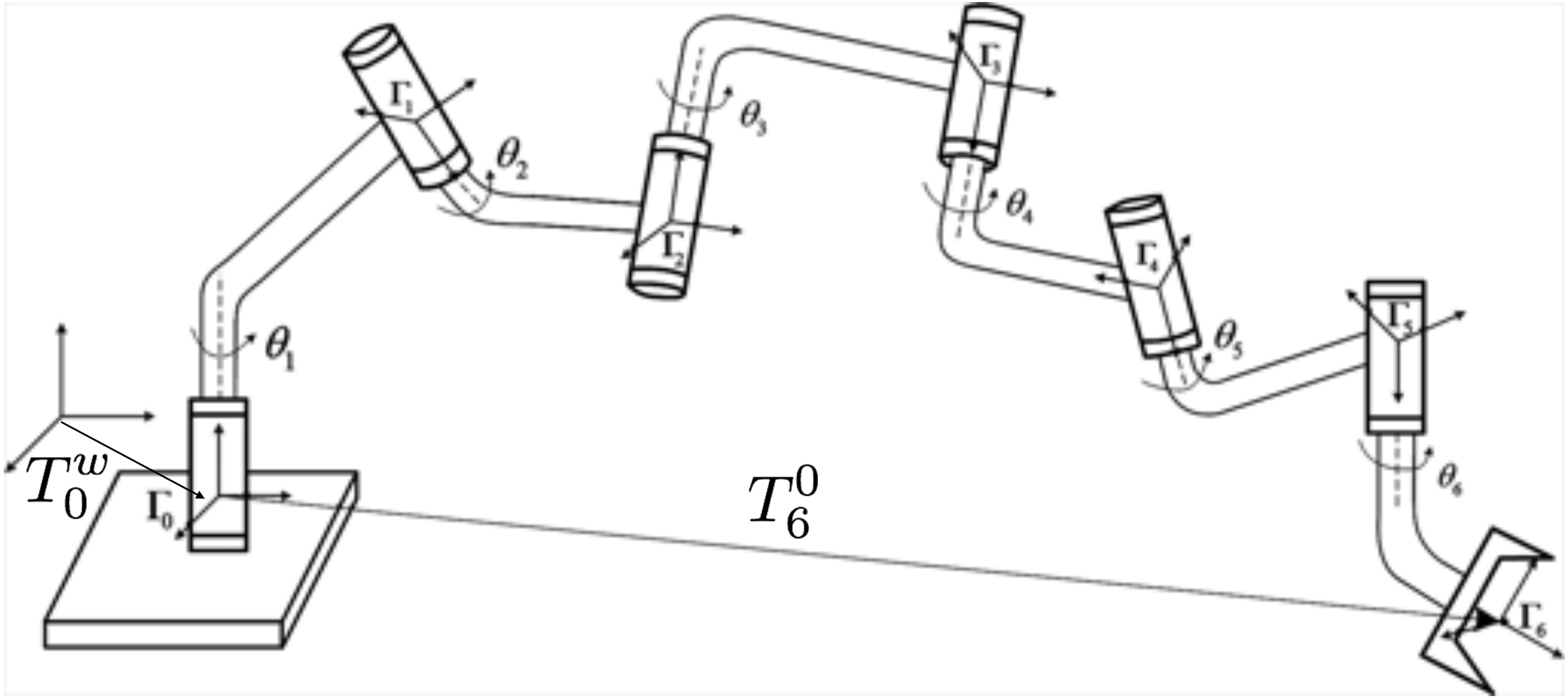
Assuming as given the:

- robot's kinematic definition
- geometry of each link
- current state of all joints
  - zero configuration
  - add motor motion



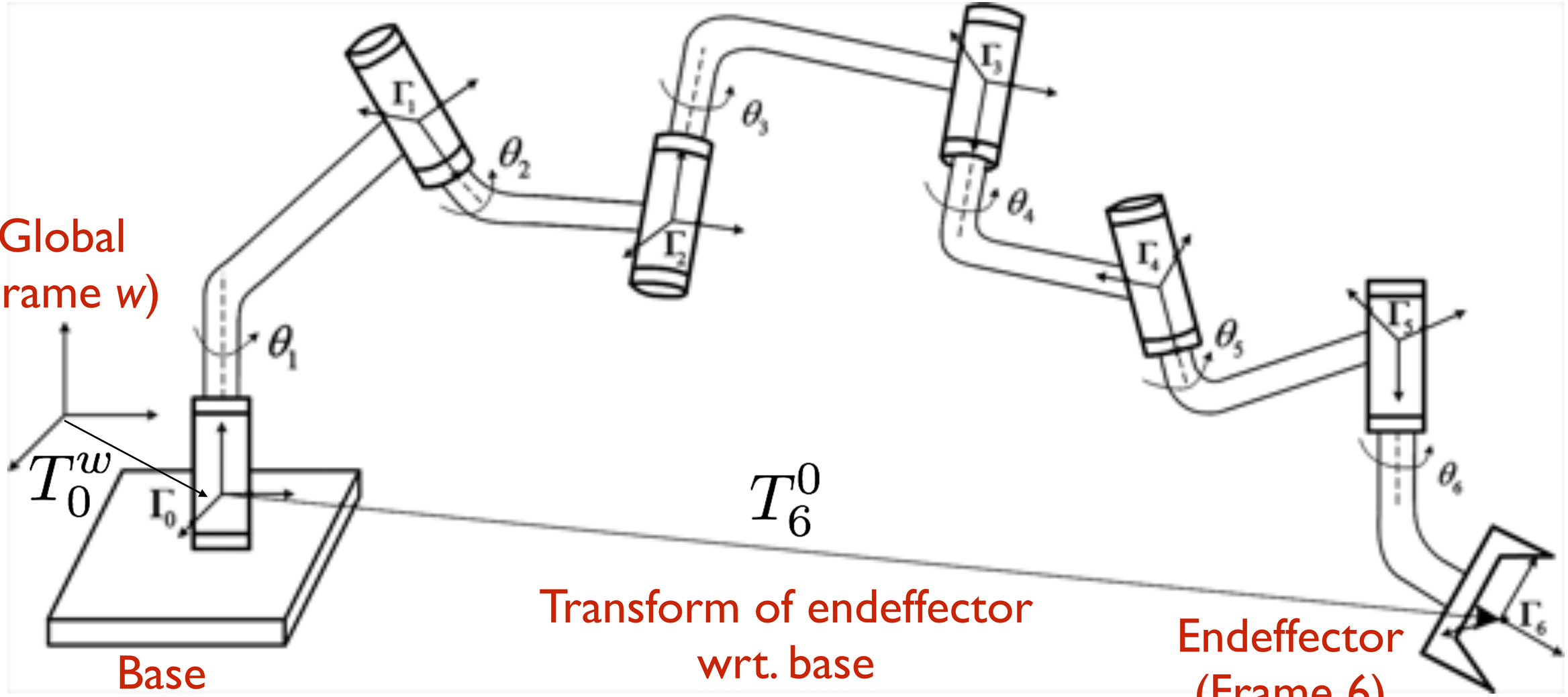
Slide borrowed from Michigan Robotics autorob.org

Forward kinematics: many-to-one mapping of robot configuration to reachable workspace endeffector poses



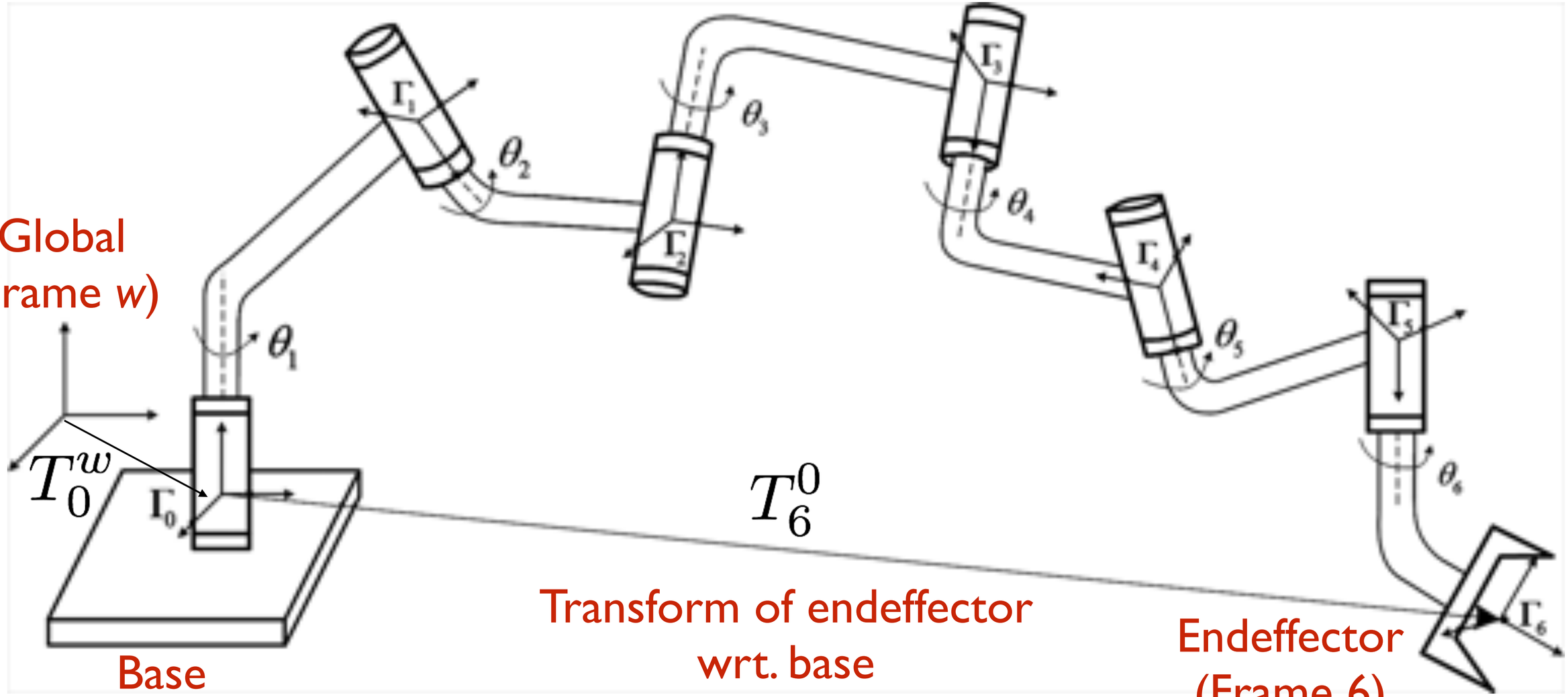
Slide borrowed from Michigan Robotics autorob.org

Forward kinematics: many-to-one mapping of robot configuration to reachable workspace endeffector poses



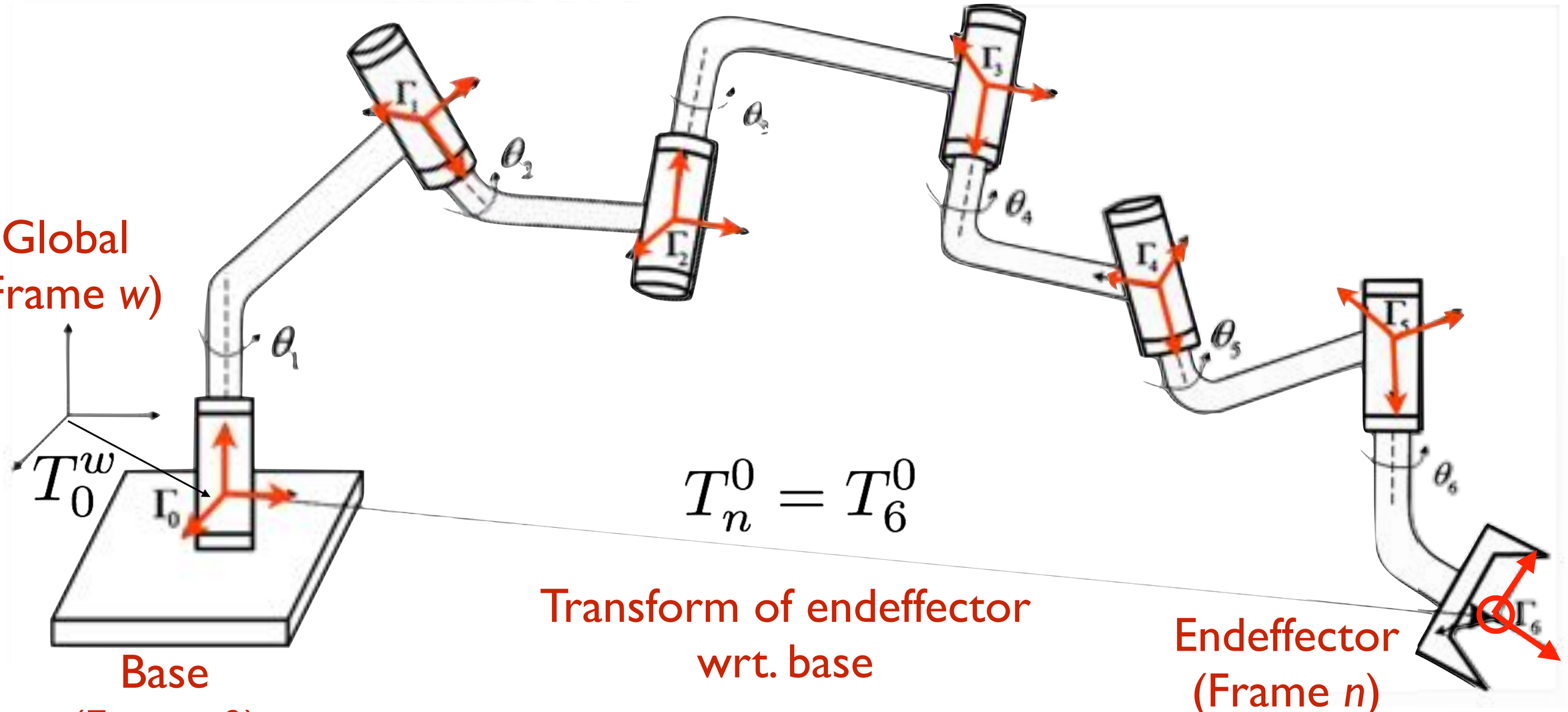
Slide borrowed from Michigan Robotics autorob.org

Workspace: 3D space defined in the global frame



Slide borrowed from Michigan Robotics autorob.org

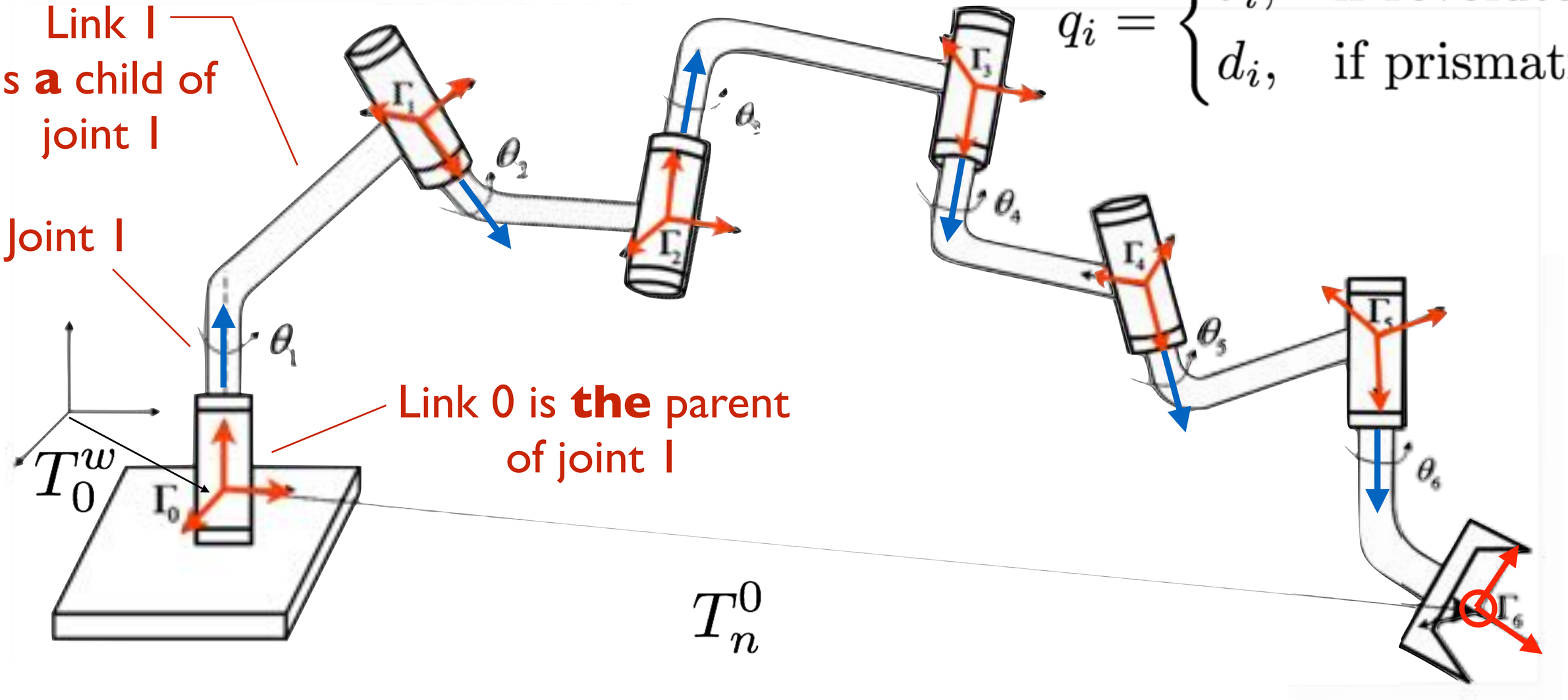
Kinematic chain: connects N+1 links together by N joints;
with a coordinate frame on each link



Slide borrowed from Michigan Robotics autorob.org

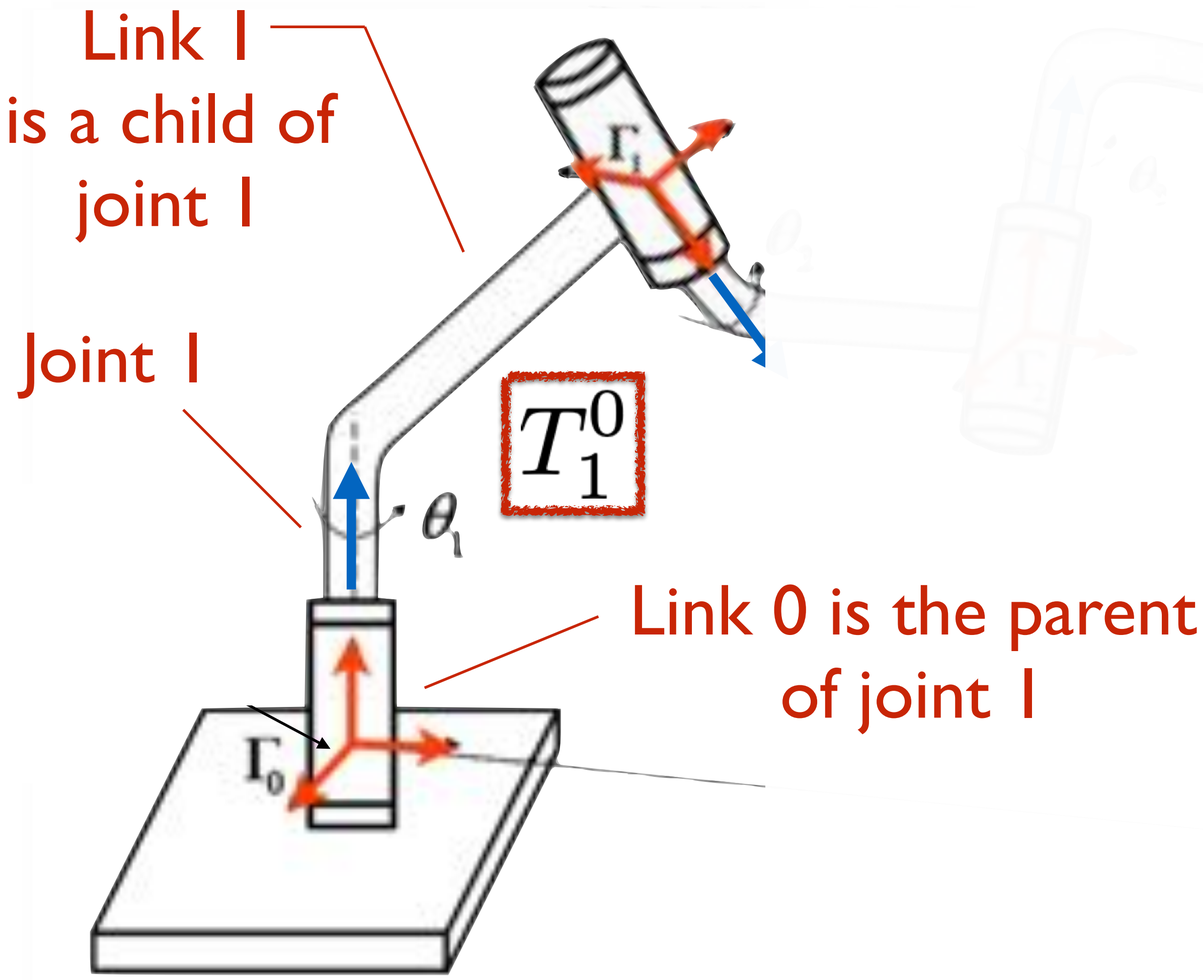
Joint (q_i): relates the motion of one link (the child link) wrt. another link (the parent)
joint motion only affects the child link

$$q_i = \begin{cases} \theta_i, & \text{if revolute} \\ d_i, & \text{if prismatic} \end{cases}$$



Slide borrowed from Michigan Robotics autorob.org

Joint (q_i): relates the motion of one link (the child link) wrt. another link (the parent)
joint motion only affects the child link, where its state



$$q_i = \begin{cases} \theta_i, & \text{if revolute} \\ d_i, & \text{if prismatic} \end{cases}$$

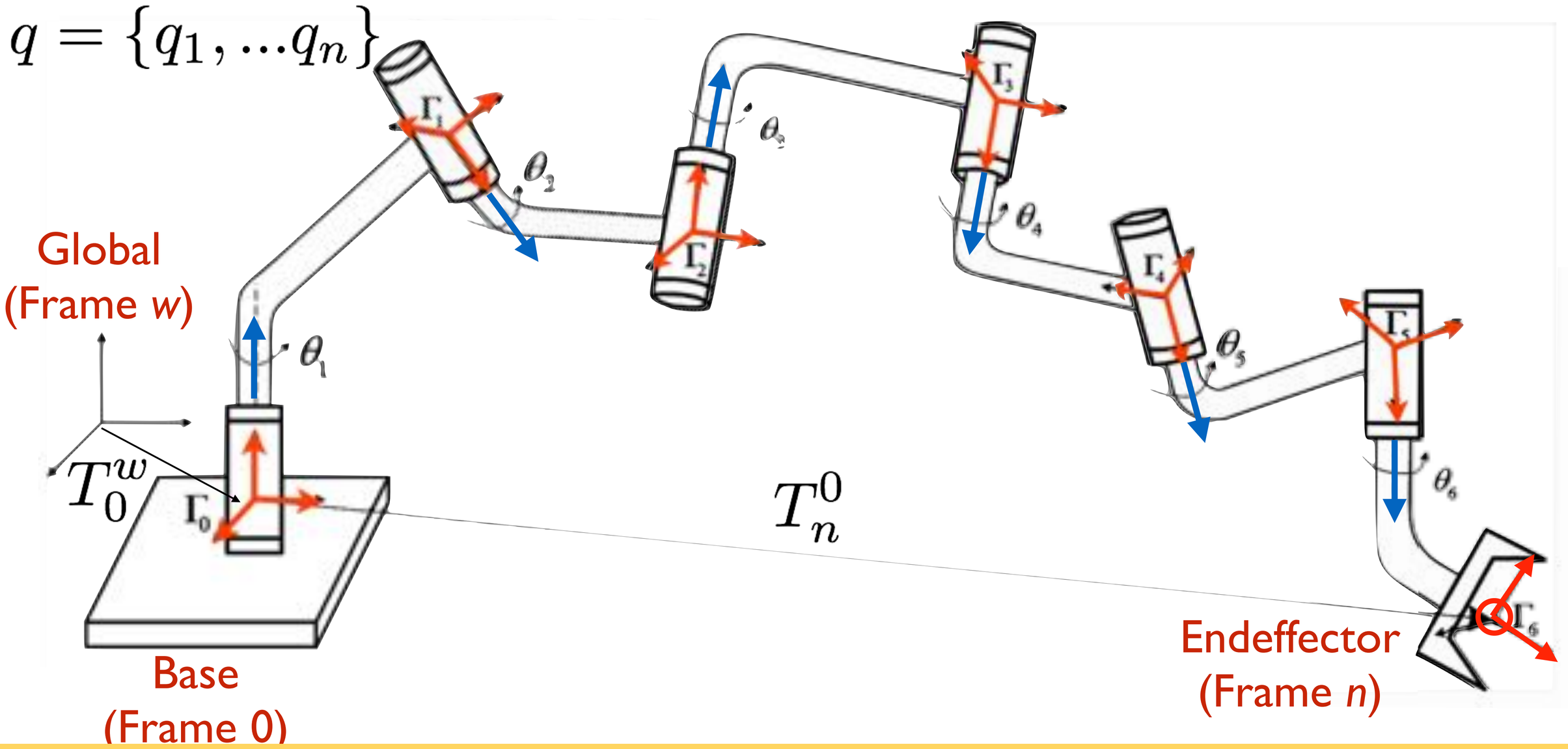
is used to express a *4*-by-*4* homogeneous transform $\boldsymbol{A_i(q_i)}$:

$$A_i = \begin{bmatrix} R_i^{i-1} & o^{i-1} \\ 0 & 1 \end{bmatrix}$$

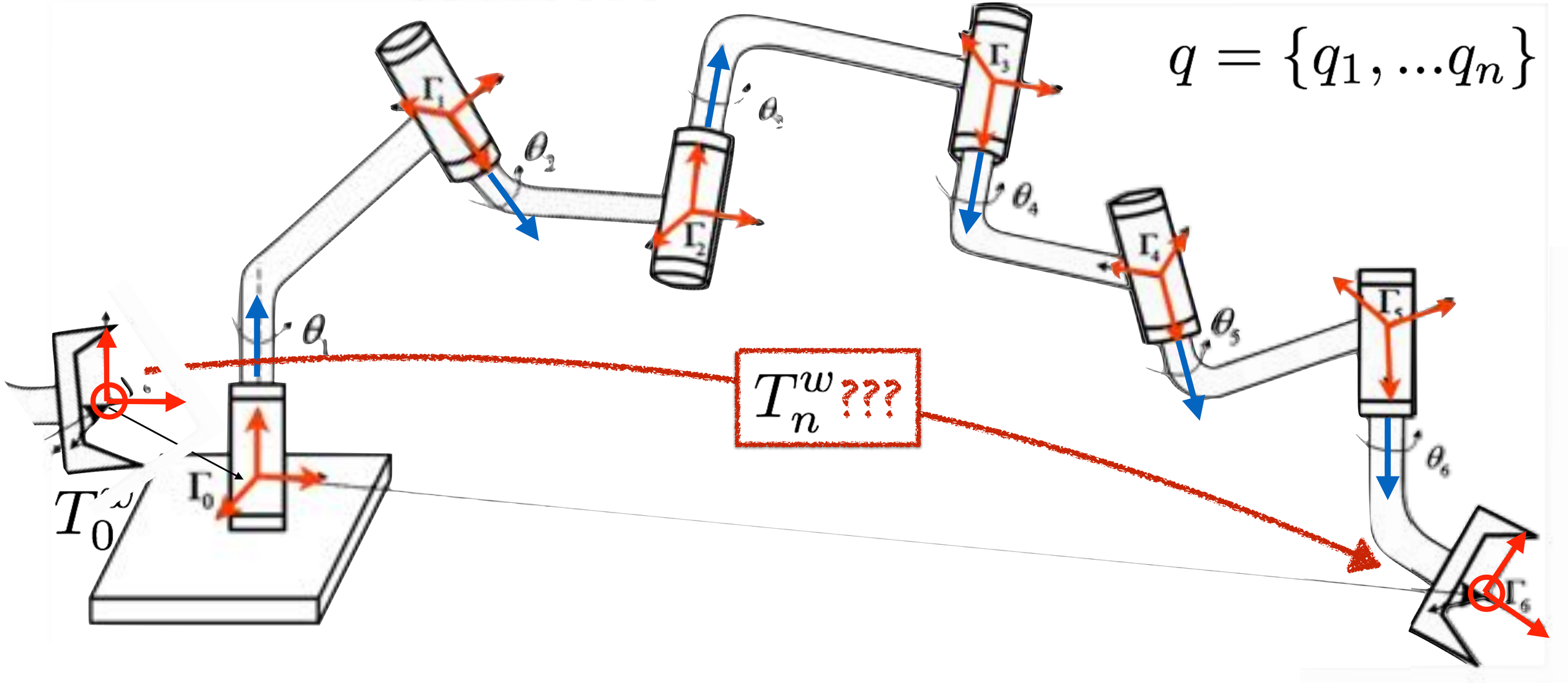
such that frames in a kinematic chain are related as by $\boldsymbol{T^i_j}$:

$$T_j^i = \begin{cases} A_{i+1} A_{i+2} \dots A_{j-1} A_j & \text{if } i < j \\ I & \text{if } i = j \\ (T_j^i)^{-1} & \text{if } j > i \end{cases}$$

Configuration (*q*): is the state of all joints in the kinematic chain
Configuration space: the space of all possible configurations

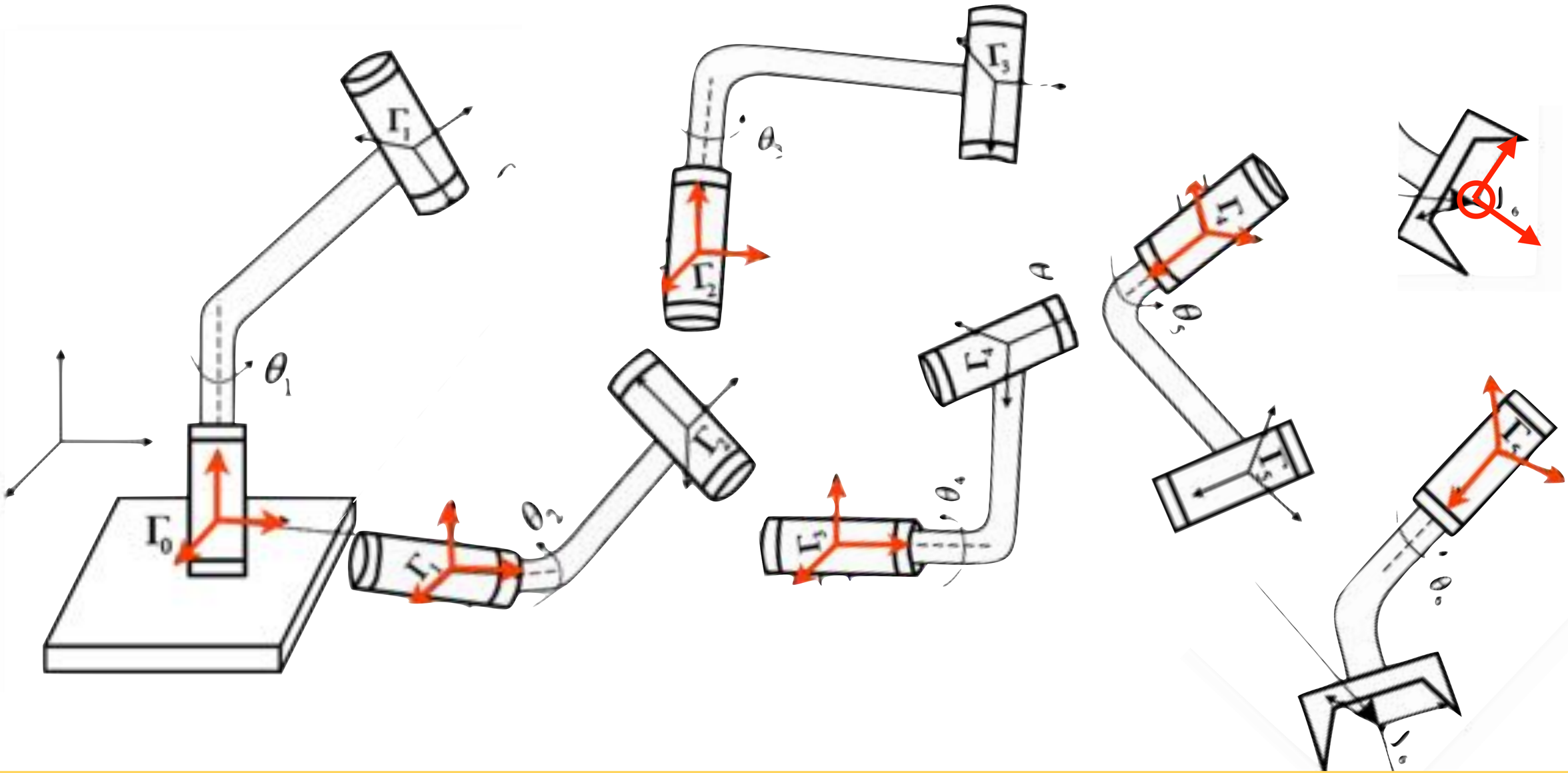


Forward kinematics restated: Given ***q***, *find* T^w_n;
T^w_n *transforms endeffector into workspace*



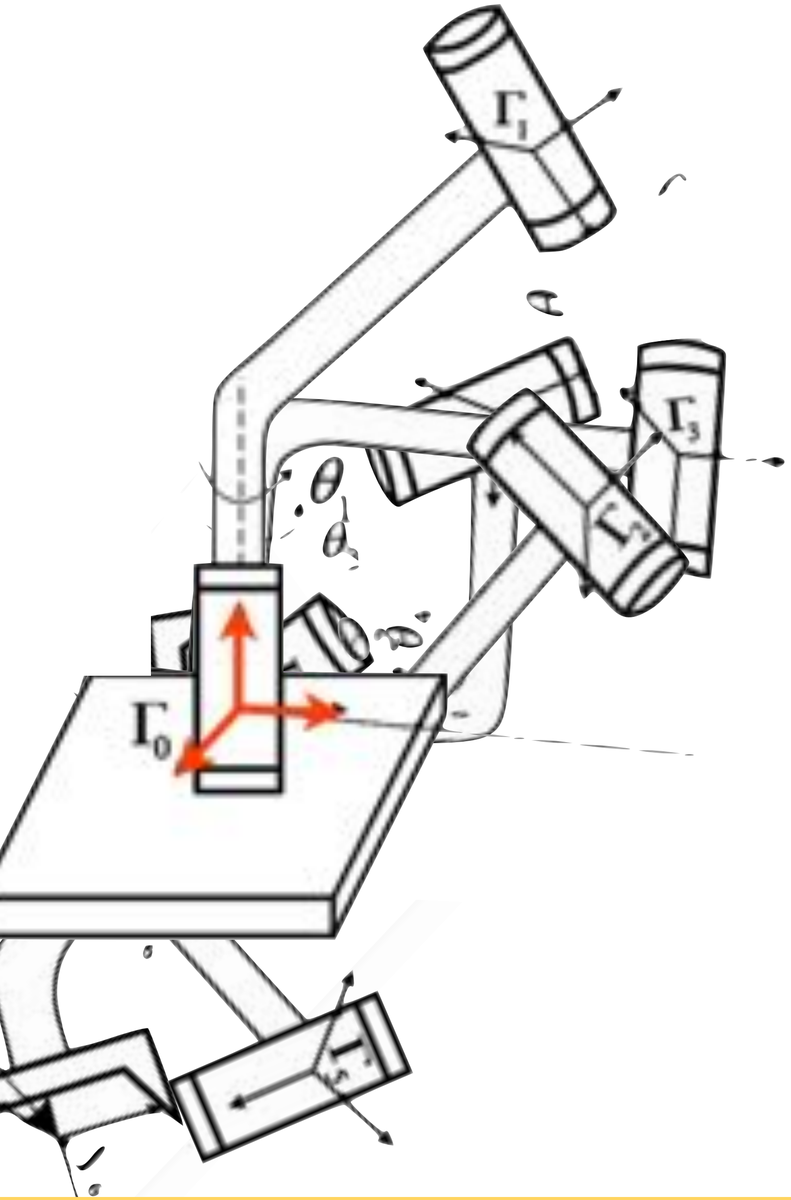
Slide borrowed from Michigan Robotics autorob.org

Problem: Every link considers itself to the center of the universe.
How do we properly pose link with respect to each other?



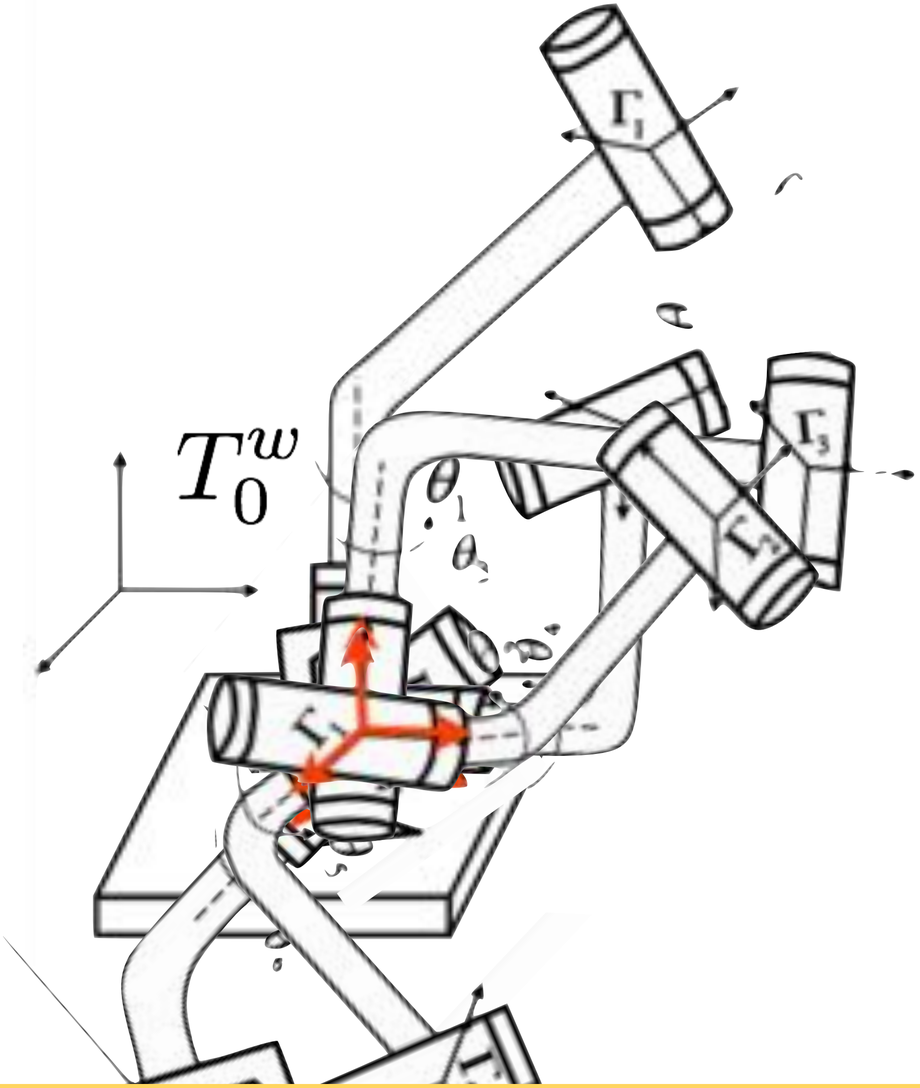
Slide borrowed from Michigan Robotics autorob.org

Approach: Consider all links to be aligned with the global origin …

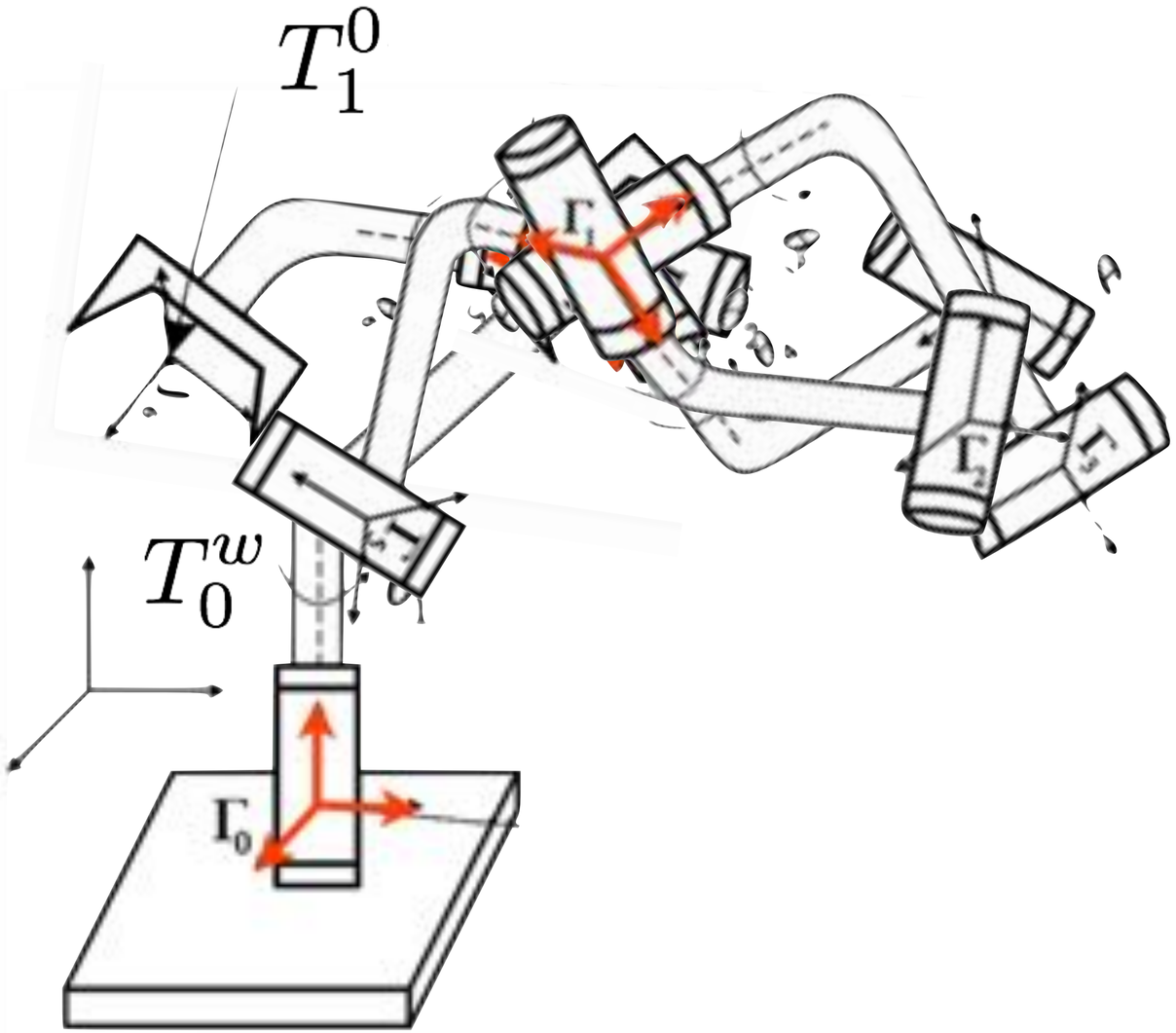


Slide borrowed from Michigan Robotics autorob.org

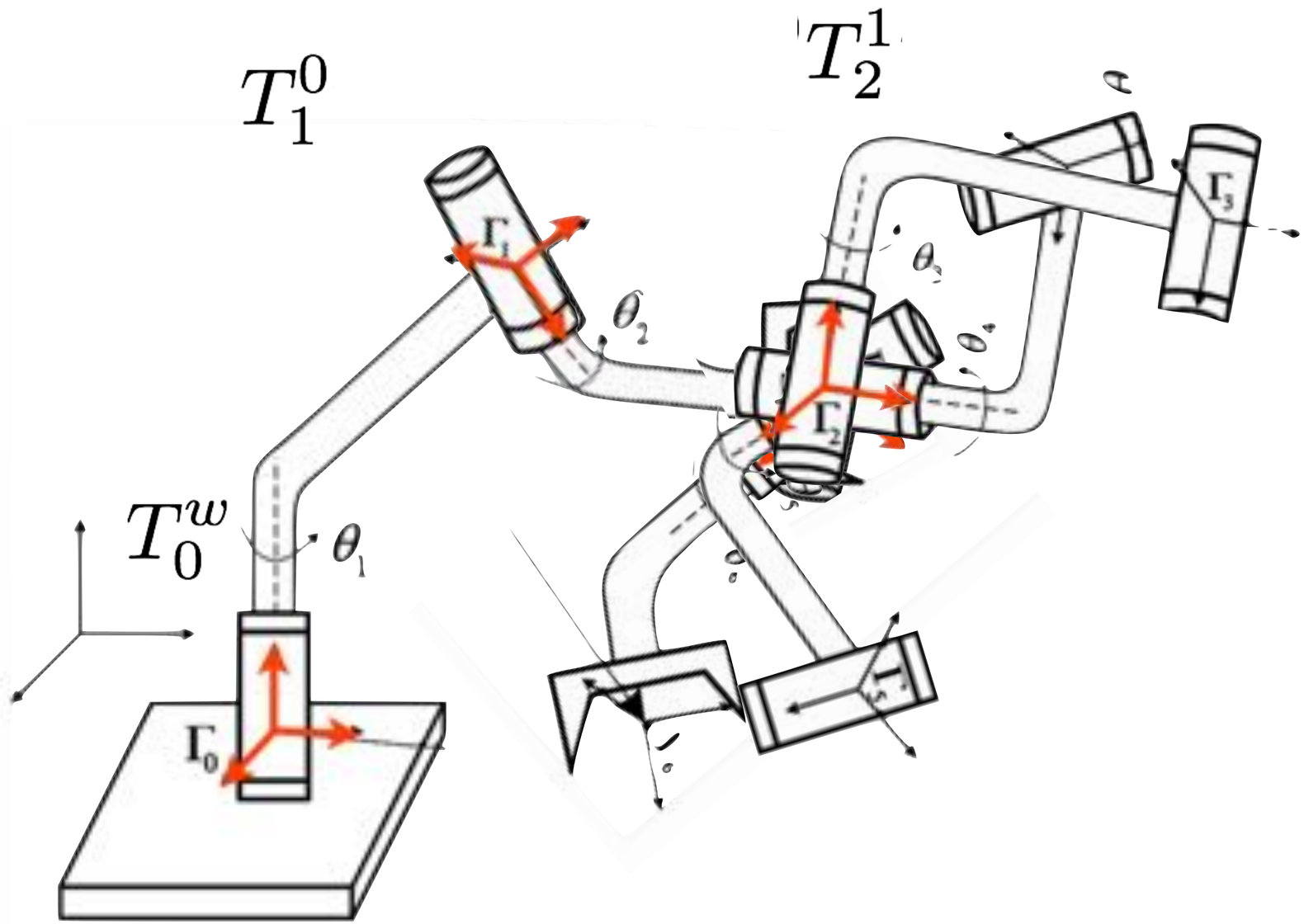
Approach: transform along kinematic chain bringing descendants along; each transform will consist of a rotation and a translation



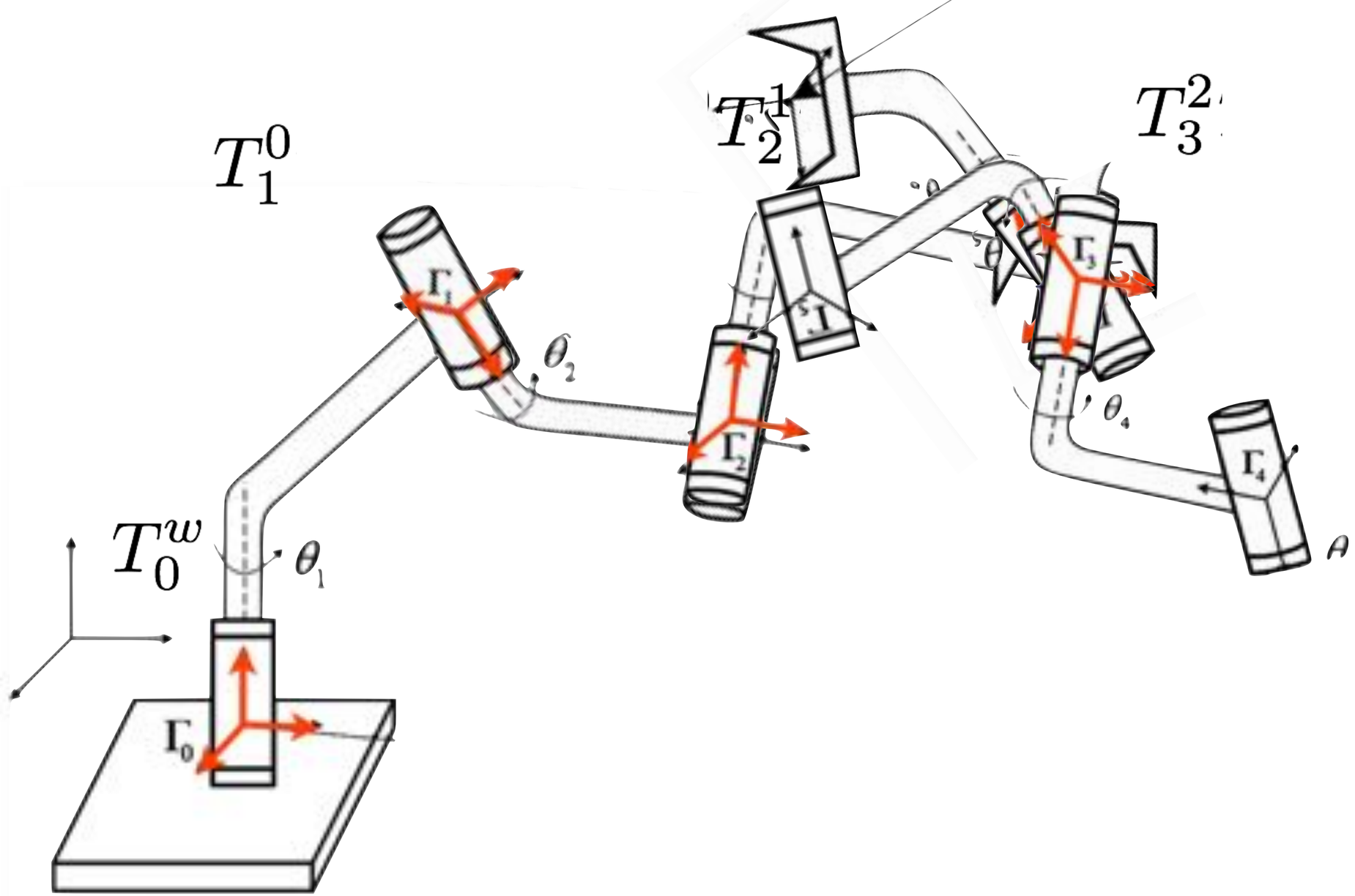
Slide borrowed from Michigan Robotics autorob.org



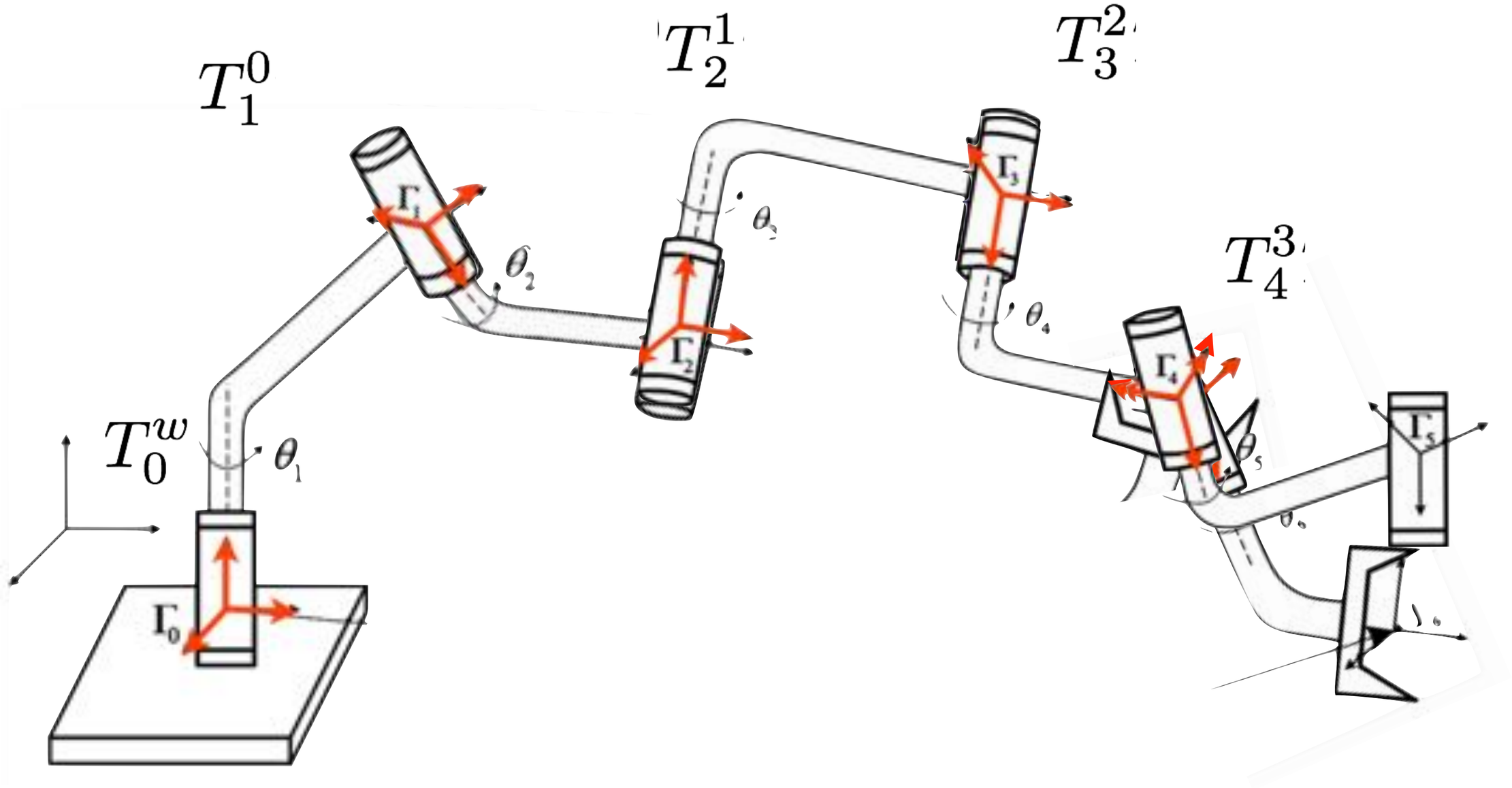
Slide borrowed from Michigan Robotics autorob.org



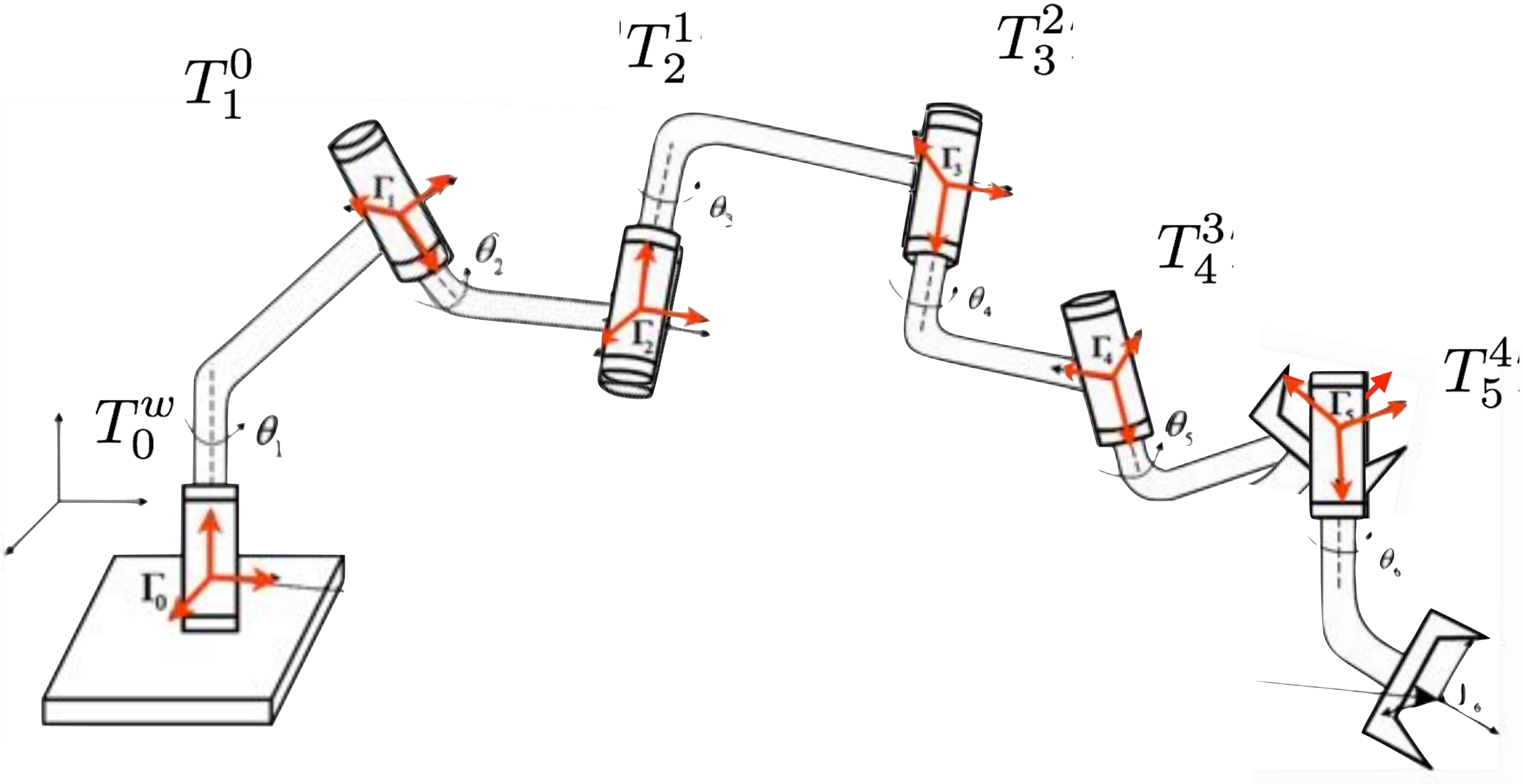
Slide borrowed from Michigan Robotics autorob.org



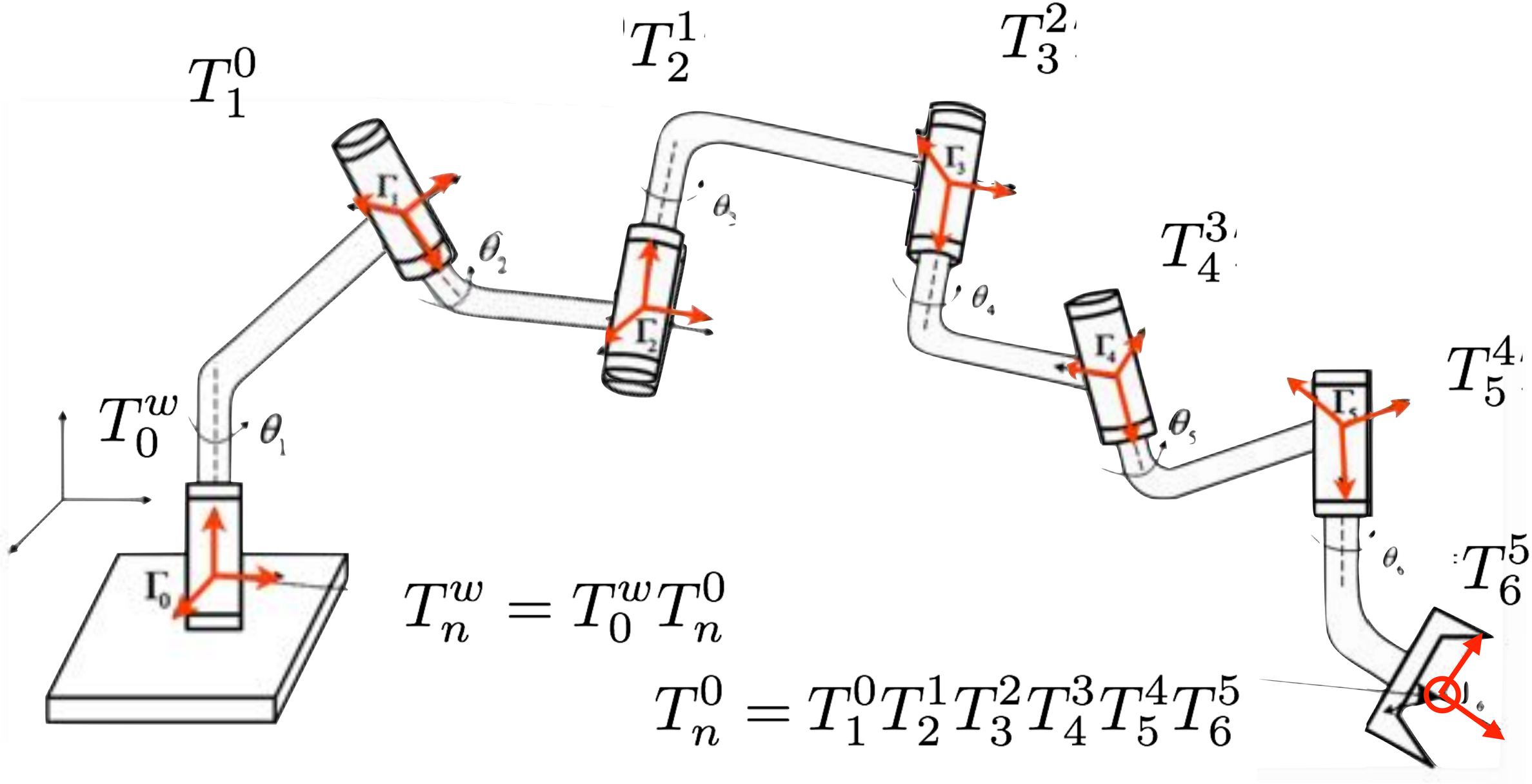
Slide borrowed from Michigan Robotics autorob.org



Slide borrowed from Michigan Robotics autorob.org



Slide borrowed from Michigan Robotics autorob.org

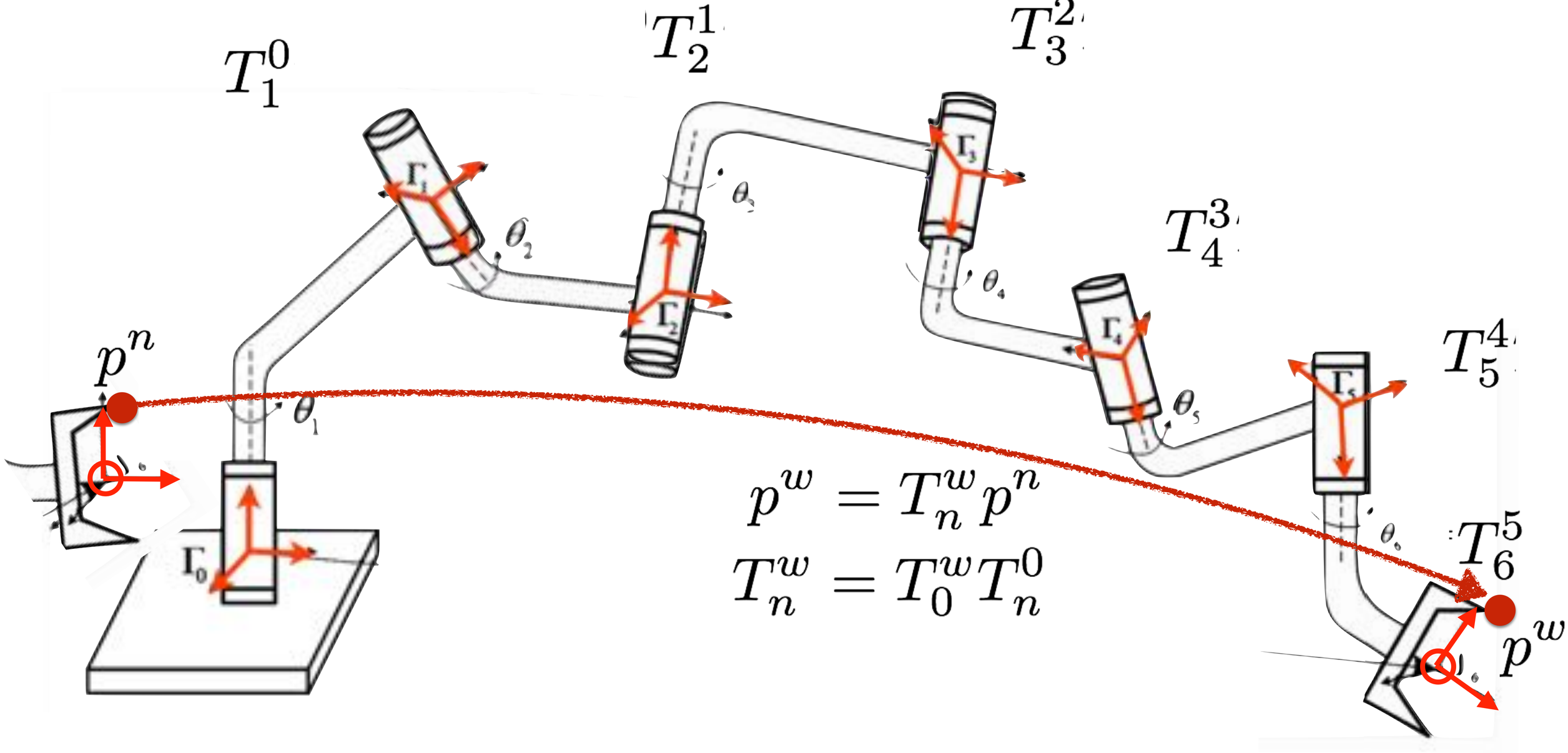


$$T_n^w = T_0^w T_n^0$$

$$T_n^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5$$

Slide borrowed from Michigan Robotics autorob.org

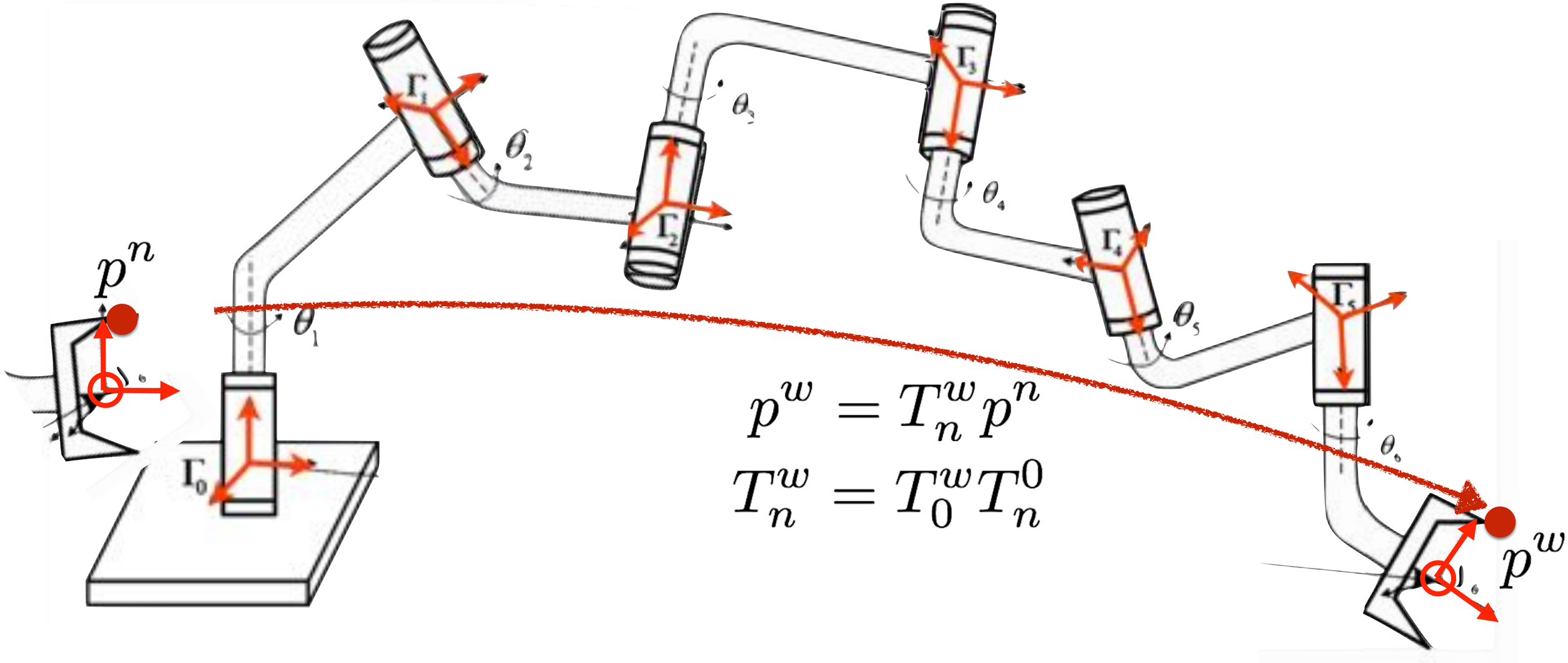
Any point on the endeffector can be transformed to its location in the world



Slide borrowed from Michigan Robotics autorob.org

1) How to represent homogeneous transforms?

2) How to compute transform to endeffector?



$$p^w = T_n^w p^n$$

$$T_n^w = T_0^w T_n^0$$

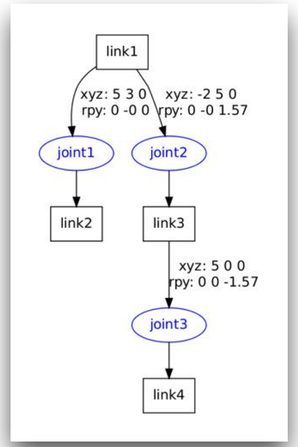
Slide borrowed from Michigan Robotics autorob.org

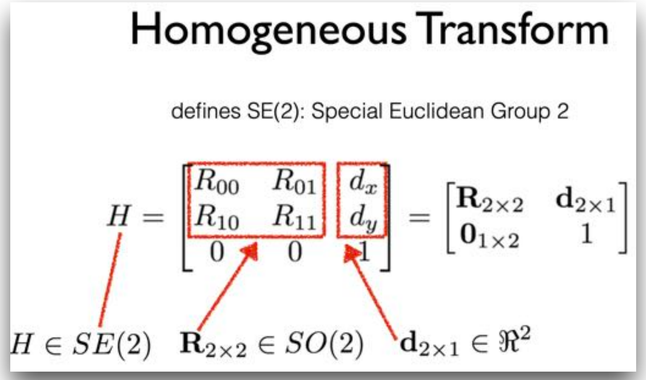
Previously

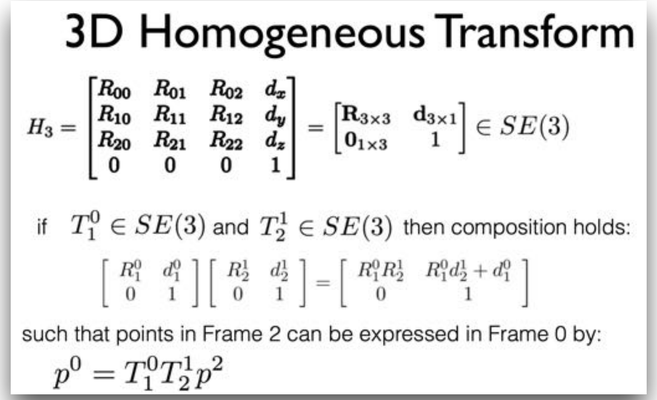
1) How to represent homogeneous transforms?

Assuming as given the:

- geometry of each link
- robot's kinematic definition

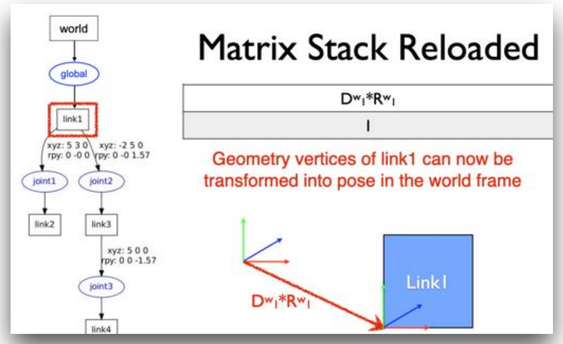






2) How to compute transform to endeffector?

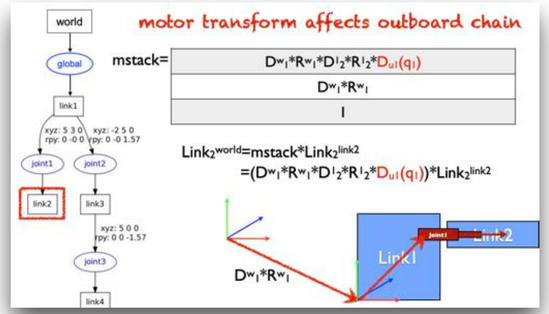
Zero configuration

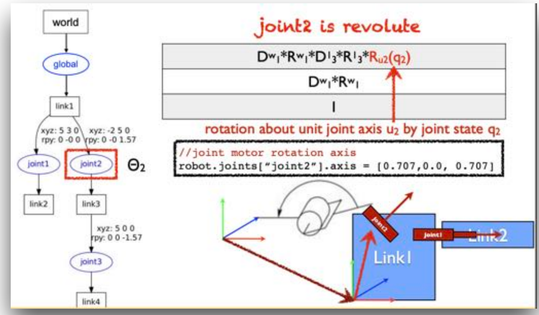


Assuming as given the:

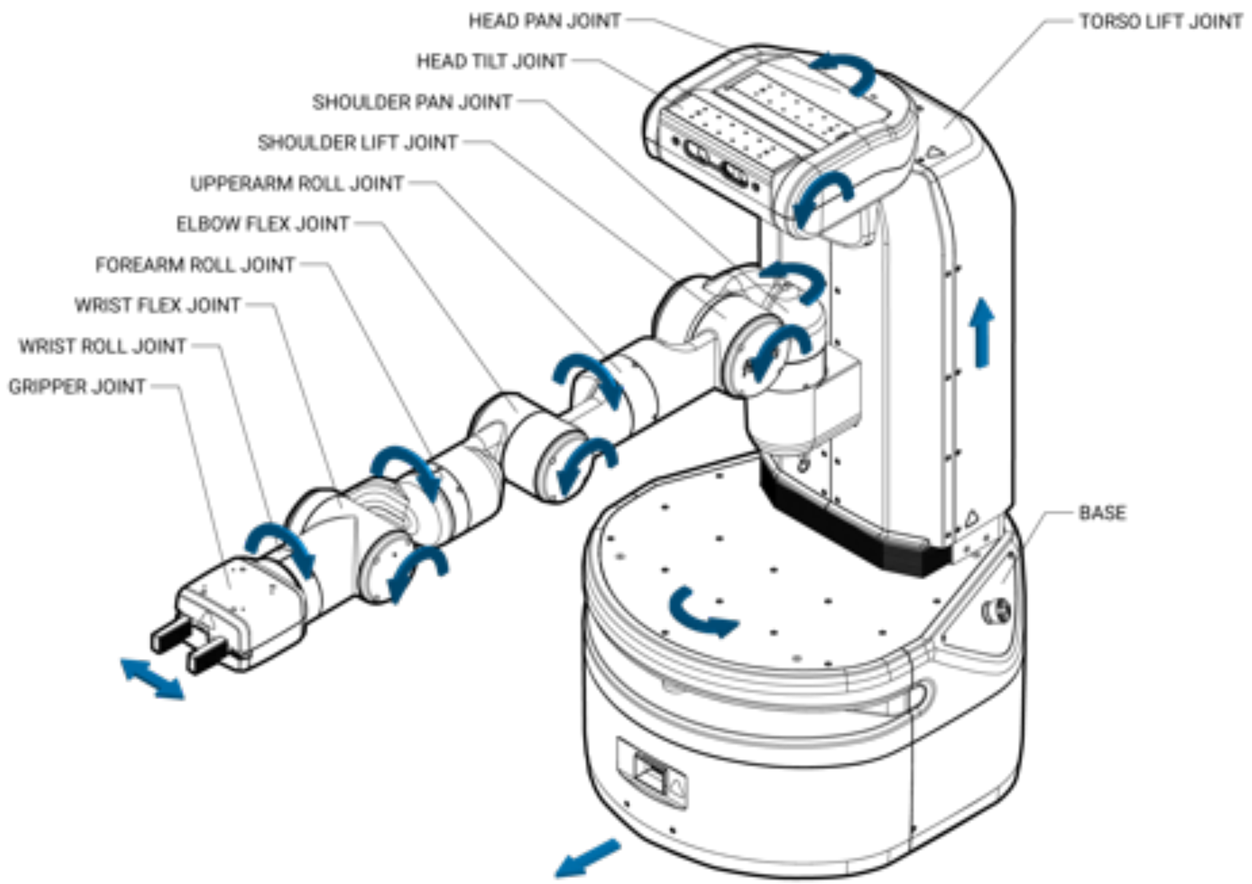
- geometry of each link
- robot's kinematic definition
- **current state of all joints**

Add motor motion





Can a joint move infinitely far?

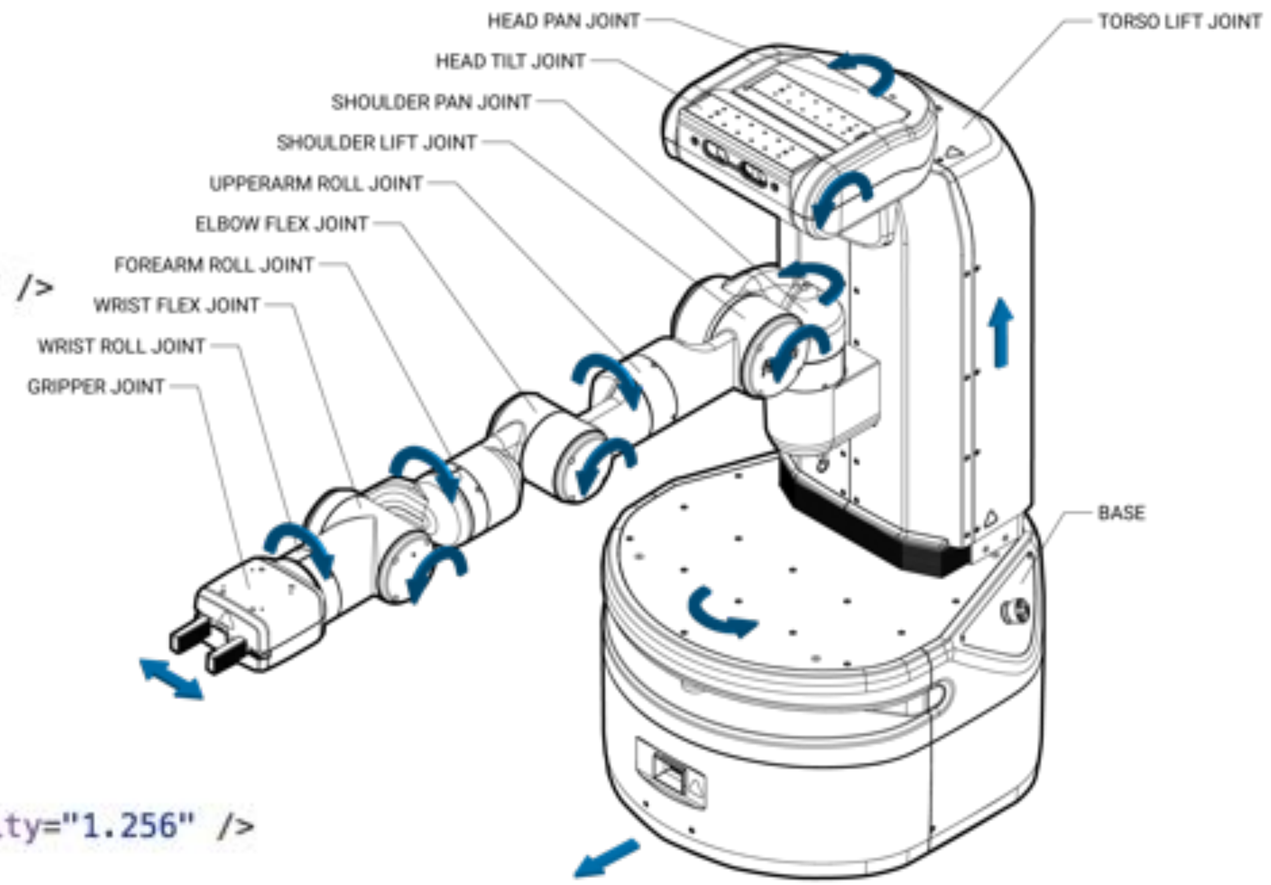


Slide borrowed from Michigan Robotics autorob.org

Joint Limits

Prismatic joint description

```
<joint name="torso_lift_joint" type="prismatic">
  <origin rpy="-6.123E-17 0 0" xyz="-0.086875 0 0.37743" />
  <parent link="base_link" />
  <child link="torso_lift_link" />
  <axis xyz="0 0 1" />
  <limit effort="450.0" lower="0" upper="0.4" velocity="0.1" />
<dynamics damping="100.0" /></joint>
```



Revolute joint description

```
<joint name="shoulder_pan_joint" type="revolute">
  <origin rpy="0 0 0" xyz="0.119525 0 0.34858" />
  <parent link="torso_lift_link" />
  <child link="shoulder_pan_link" />
  <axis xyz="0 0 1" />
  <dynamics damping="1.0" />
  <limit effort="33.82" lower="-1.6056" upper="1.6056" velocity="1.256" />
</joint>
```

Continuous joints have no limits

Slide borrowed from Michigan Robotics autorob.org

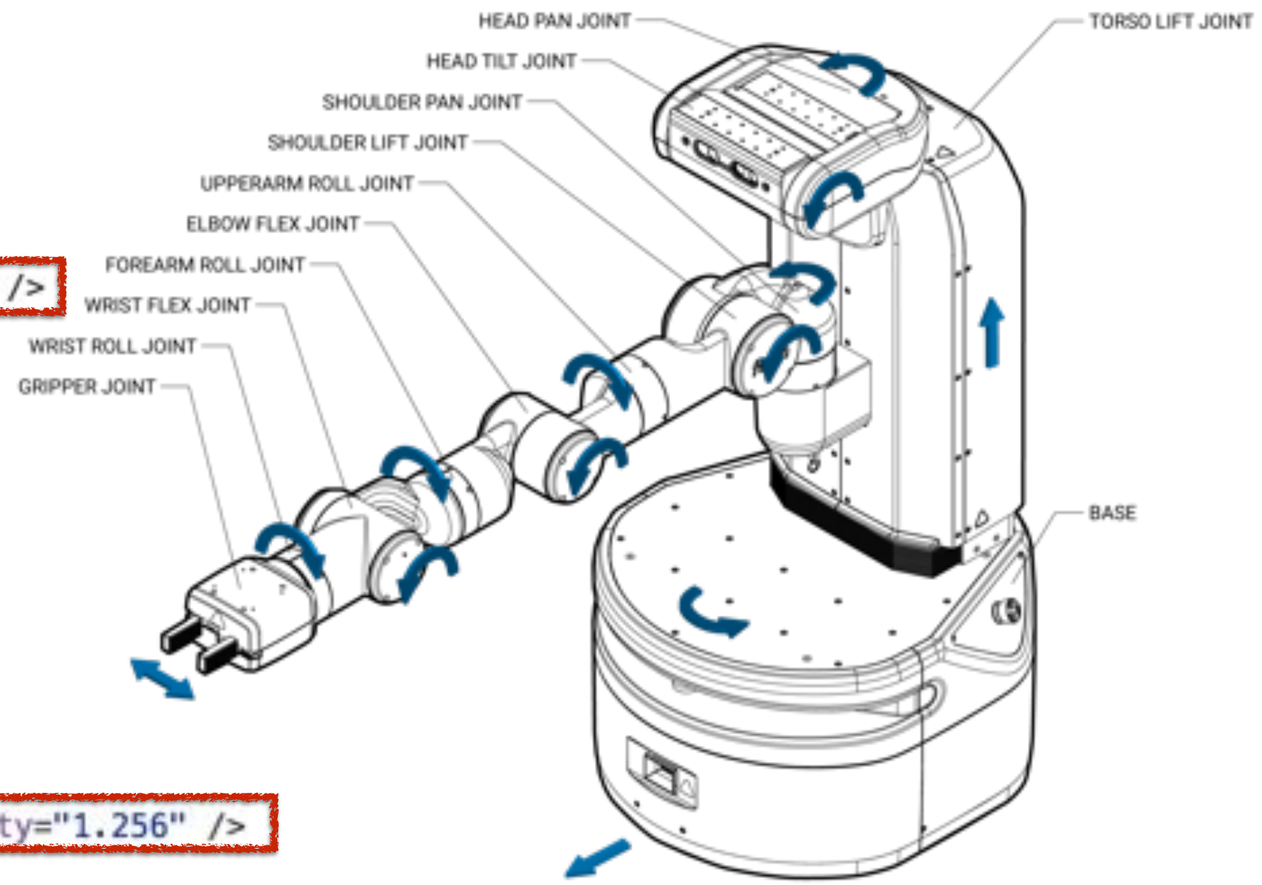
Joint Limits

Prismatic joint description

```
<joint name="torso_lift_joint" type="prismatic">
  <origin rpy="-6.123E-17 0 0" xyz="-0.086875 0 0.37743" />
  <parent link="base_link" />
  <child link="torso_lift_link" />
  <axis xyz="0 0 1" />
  <limit effort="450.0" lower="0" upper="0.4" velocity="0.1" />
<dynamics damping="100.0" /></joint>
```

Revolute joint description

```
<joint name="shoulder_pan_joint" type="revolute">
  <origin rpy="0 0 0" xyz="0.119525 0 0.34858" />
  <parent link="torso_lift_link" />
  <child link="shoulder_pan_link" />
  <axis xyz="0 0 1" />
  <dynamics damping="1.0" />
  <limit effort="33.82" lower="-1.6056" upper="1.6056" velocity="1.256" />
</joint>
```



Slide borrowed from Michigan Robotics autorob.org

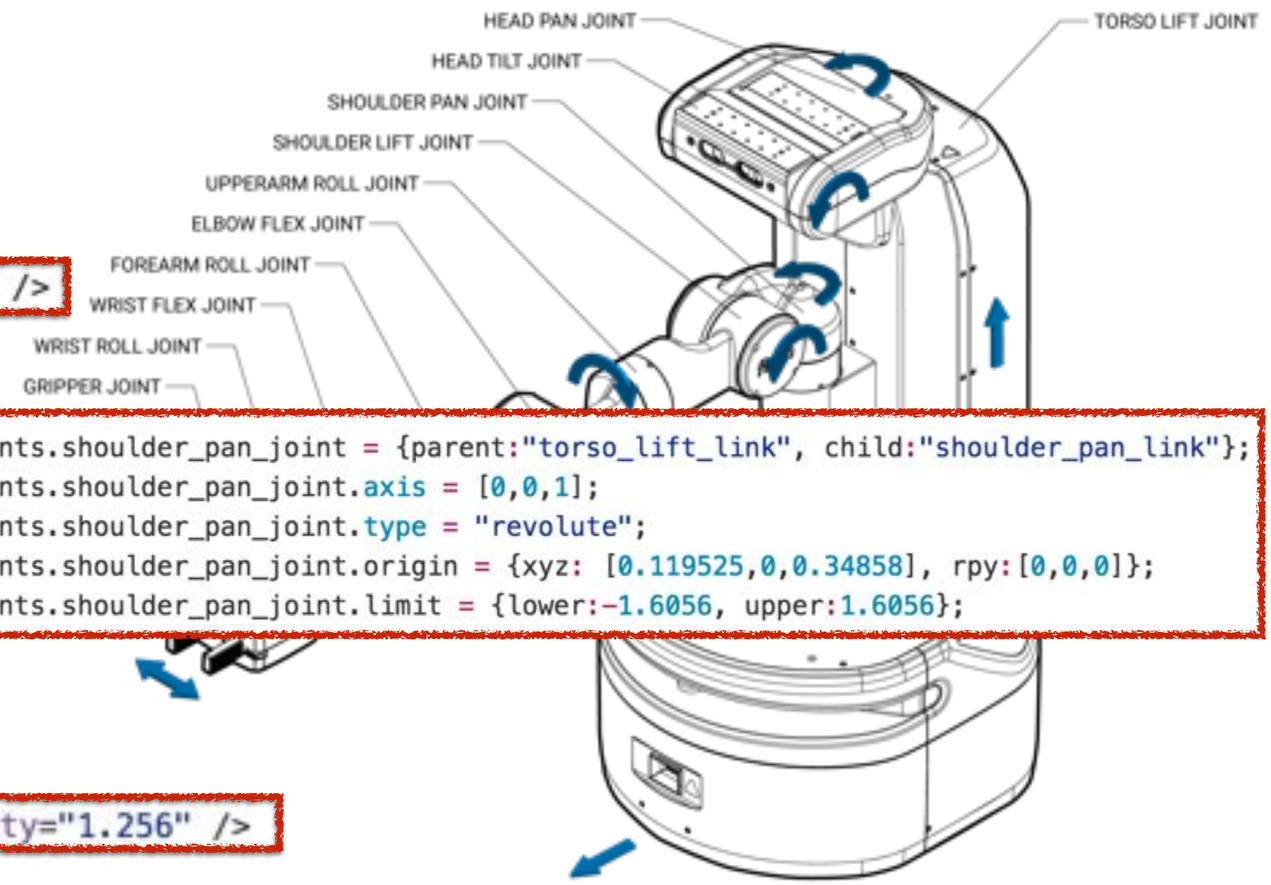
Prismatic joint description

```
<joint name="torso_lift_joint" type="prismatic">
  <origin rpy="-6.123E-17 0 0" xyz="-0.086875 0 0.37743" />
  <parent link="base_link" />
  <child link="torso_lift_link" />
  <axis xyz="0 0 1" />
  <limit effort="450.0" lower="0" upper="0.4" velocity="0.1" />
<dynamics damping="100.0" /></joint>
```

```
robot.joints.torso_lift_joint = {parent:"base_link", child:"torso_lift_link"};
robot.joints.torso_lift_joint.axis = [0,0,1];
robot.joints.torso_lift_joint.type = "prismatic";
robot.joints.torso_lift_joint.origin = {xyz: [-0.086875,0,0.37743], rpy:[-6.123E-17,0,0]};
robot.joints.torso_lift_joint.limit = {lower:0, upper:0.4};
```

Revolute joint description

```
<joint name="shoulder_pan_joint" type="revolute">
  <origin rpy="0 0 0" xyz="0.119525 0 0.34858" />
  <parent link="torso_lift_link" />
  <child link="shoulder_pan_link" />
  <axis xyz="0 0 1" />
  <dynamics damping="1.0" />
  <limit effort="33.82" lower="-1.6056" upper="1.6056" velocity="1.256" />
</joint>
```




```
robot.joints.shoulder_pan_joint = {parent:"torso_lift_link", child:"shoulder_pan_link"};
robot.joints.shoulder_pan_joint.axis = [0,0,1];
robot.joints.shoulder_pan_joint.type = "revolute";
robot.joints.shoulder_pan_joint.origin = {xyz: [0.119525,0,0.34858], rpy:[0,0,0]};
robot.joints.shoulder_pan_joint.limit = {lower:-1.6056, upper:1.6056};
```


Slide borrowed from Michigan Robotics autorob.org

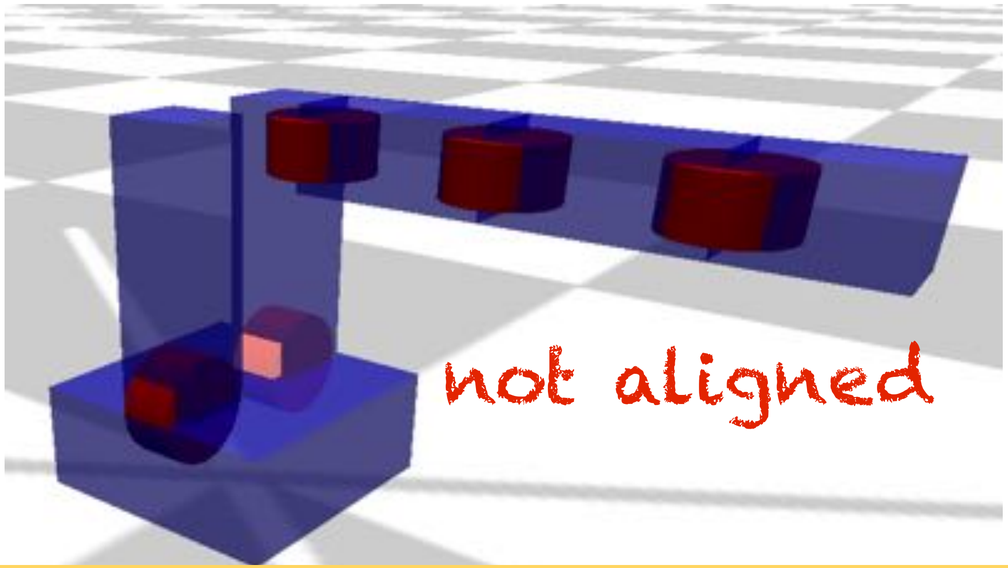
Important notes

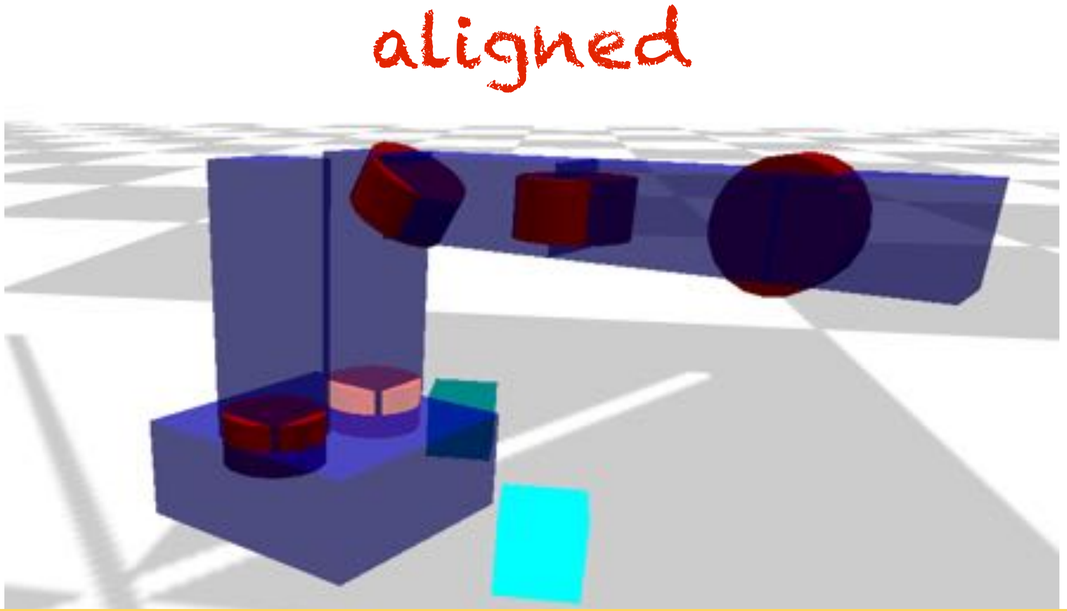
Important notes

- Rotation order I use: **XYZ** ($R_z R_y R_x$)
- vector_cross(): code stencil tests for and uses this function
- Base controls: must be implemented for interactive control
- The ".origin" field of links and joints are used to store transforms without consideration of joint motion (provided only for debugging)
- A joint and its child link will share the same coordinate frame

KinEval joint cylinder rendering

- threejs creates cylinders with axes aligned along y-axis
- you need to implement vector_cross() for KinEval to render joint cylinders properly along joint axis





Global controls for base

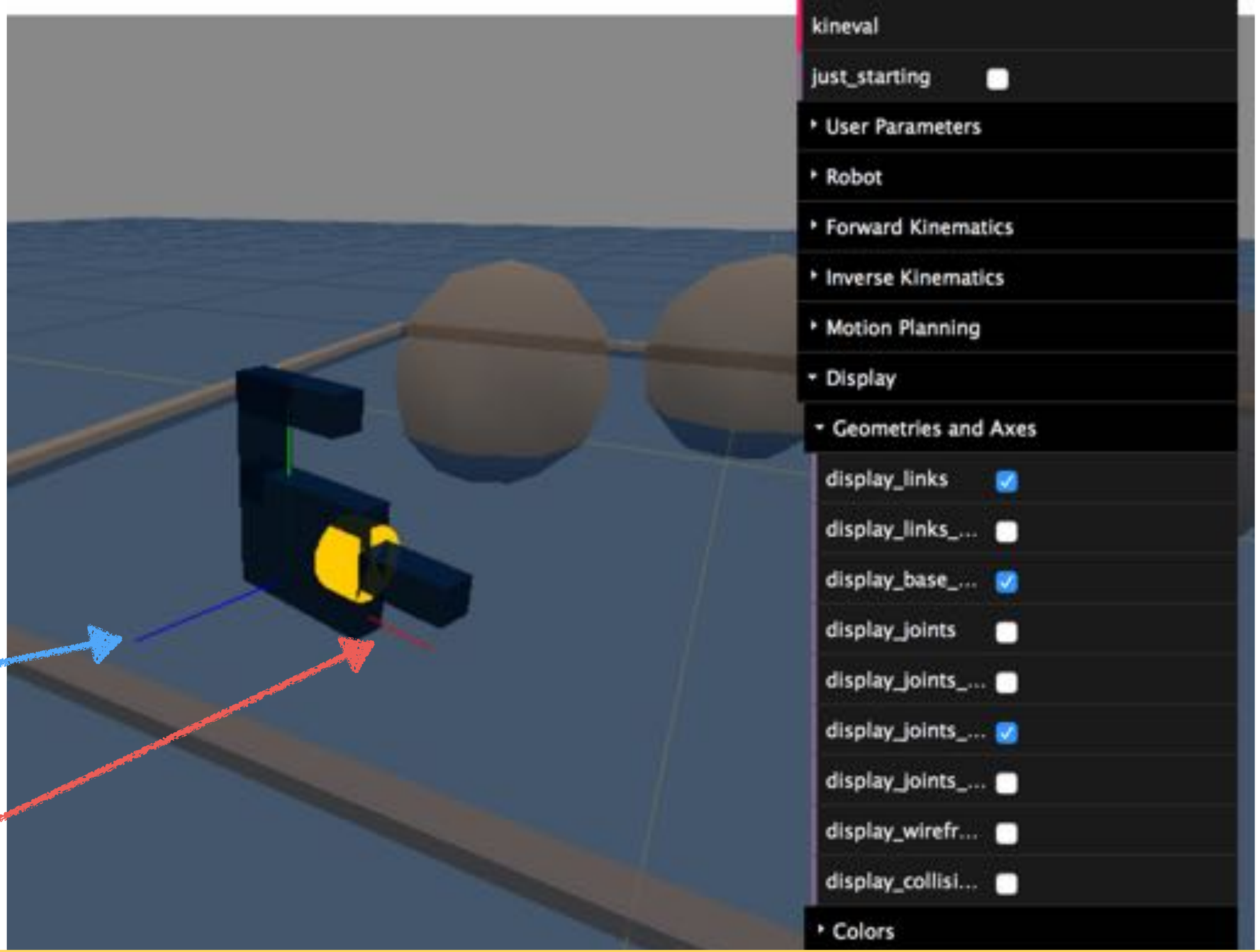
- Assume we have a base that is holonomic wrt. ground plane
  - holonomic: can move in any direction
- kineval_userinput.js assumes:

How to perform this base movement?

Slide borrowed from Michigan Robotics autorob.org

Transform vectors for heading (local z-axis) and lateral (local x-axis) of robot base into world coordinates

Store transformed vectors in variables “robot_heading” and “robot lateral”



Slide borrowed from Michigan Robotics autorob.org

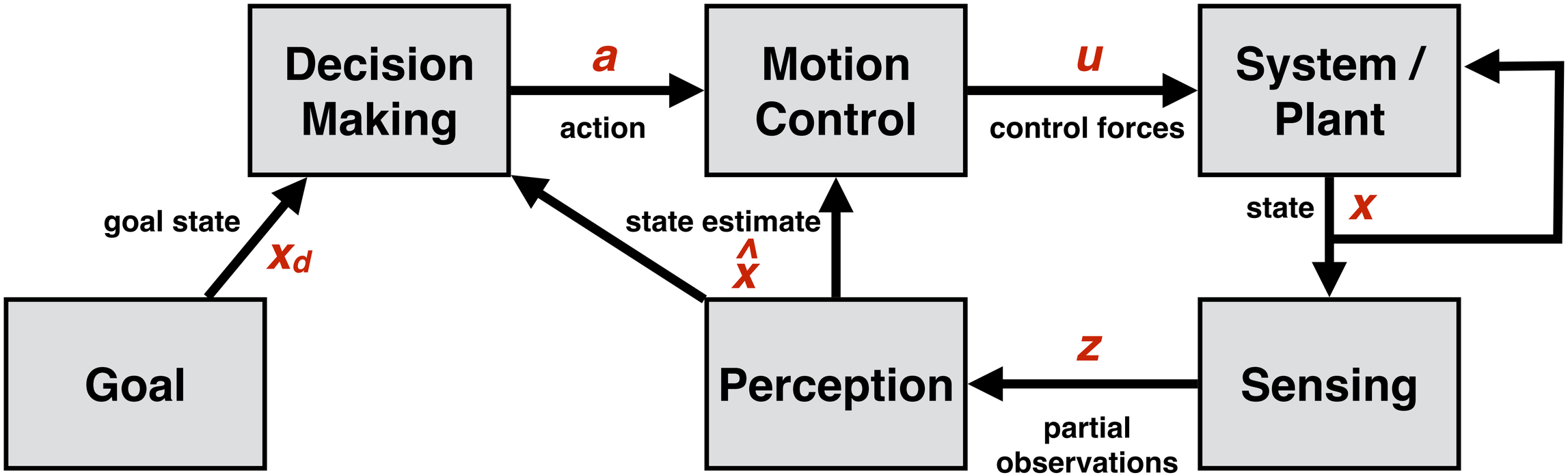
Alternatives for FK

- Denavit-Hartenberg Convention
- Product of Exponentials with Matrix Stack AutoRob & 5551
- Screw vectors as Dual Quaternions

[Kenwright 2012, Daniilidis 1999]

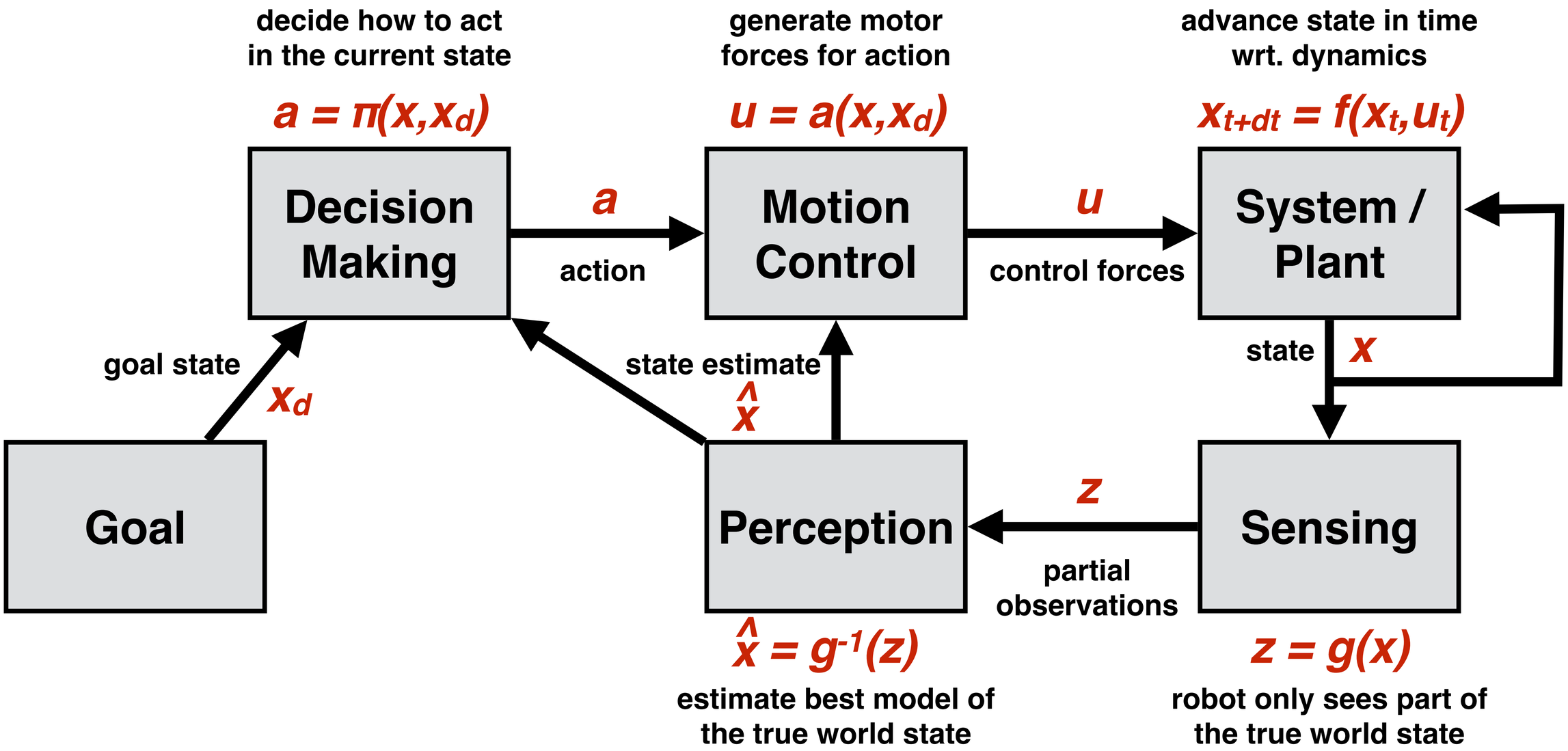
Decision Making

Robot Control Loop



Slide borrowed from Michigan Robotics autorob.org

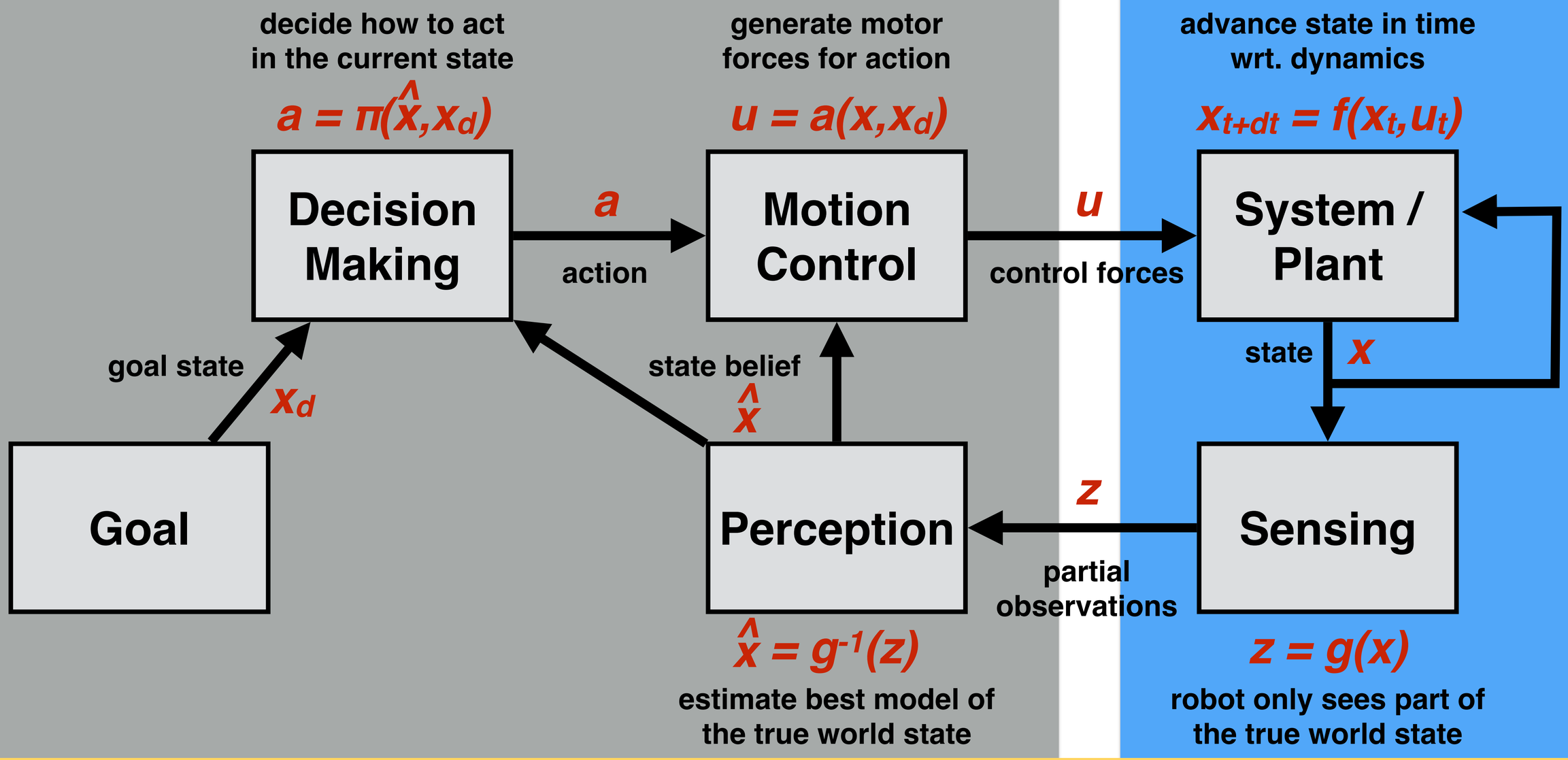
Robot Control Loop



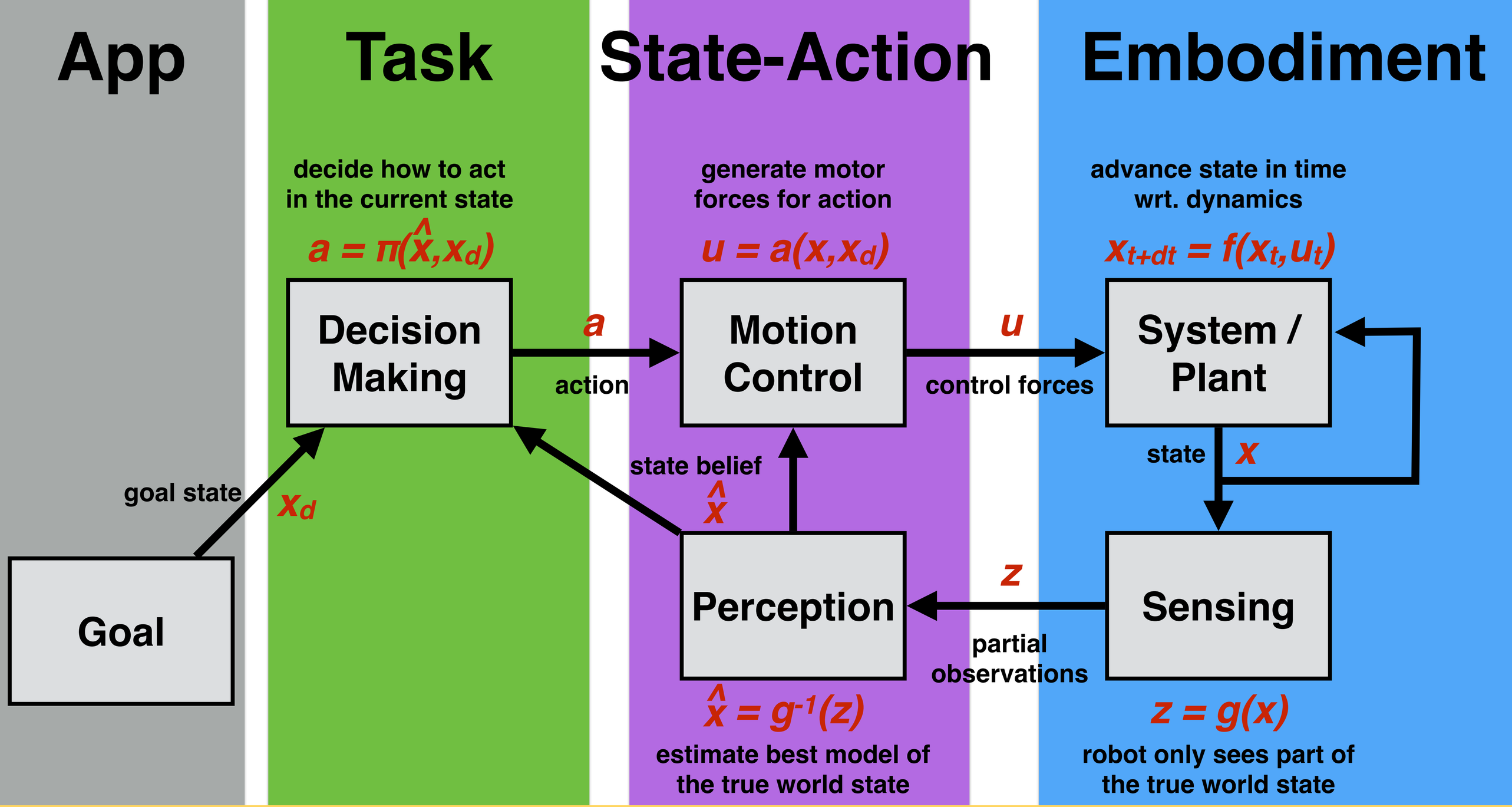
Slide borrowed from Michigan Robotics autorob.org

Autonomy

Embodiment

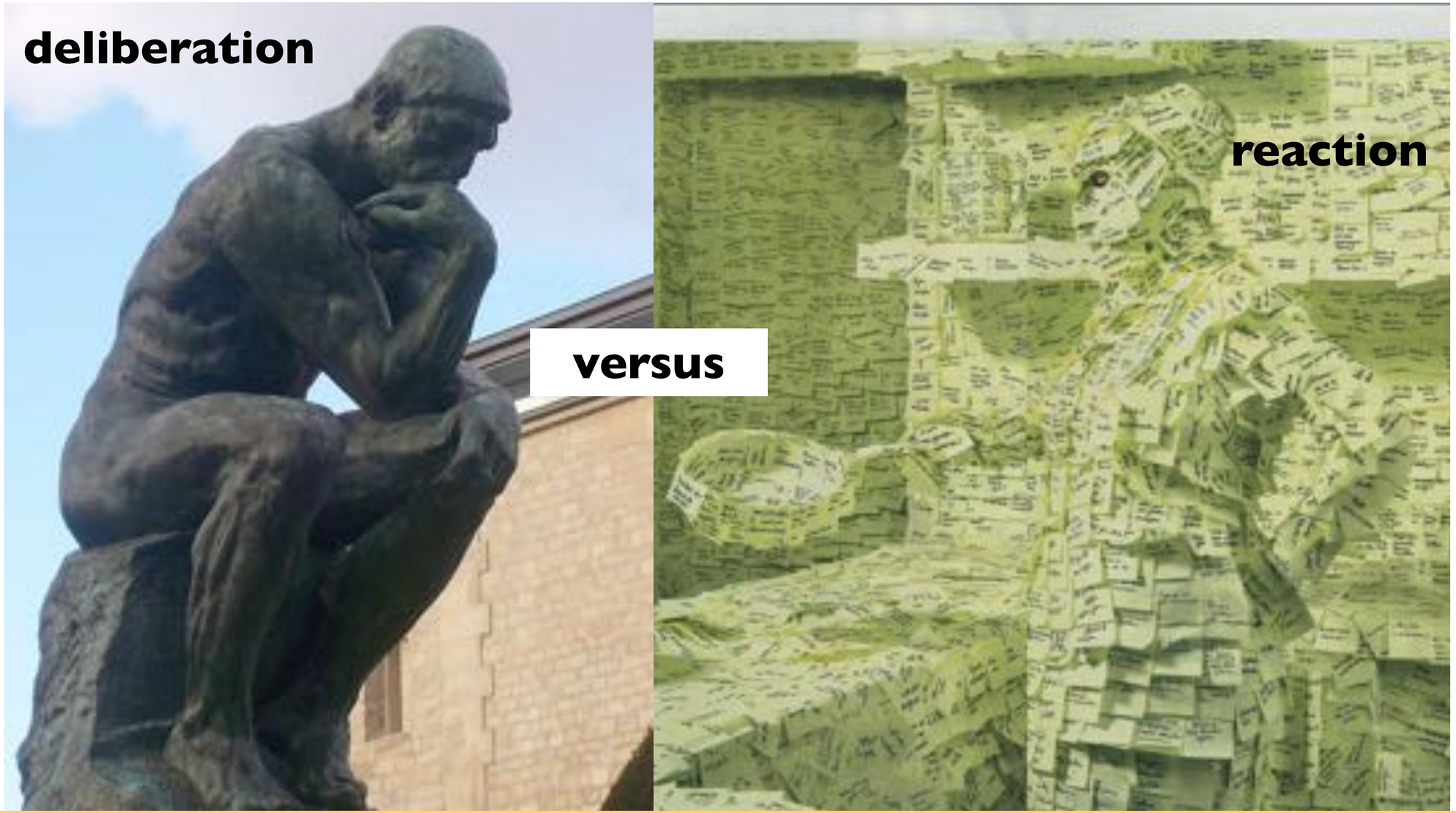


Slide borrowed from Michigan Robotics autorob.org



Slide borrowed from Michigan Robotics autorob.org

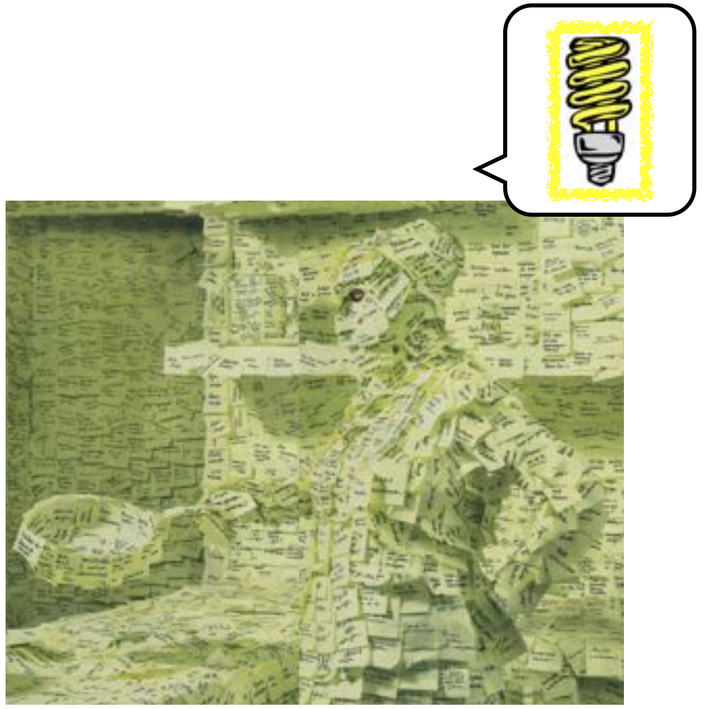
Robot Decision Making



Slide borrowed from Michigan Robotics autorob.org

Should your robot's decision making

OR

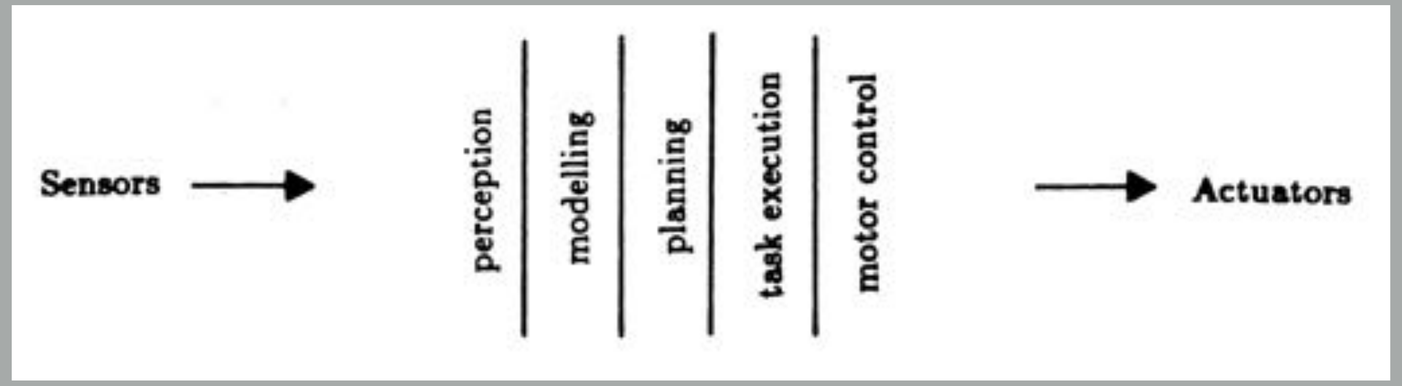


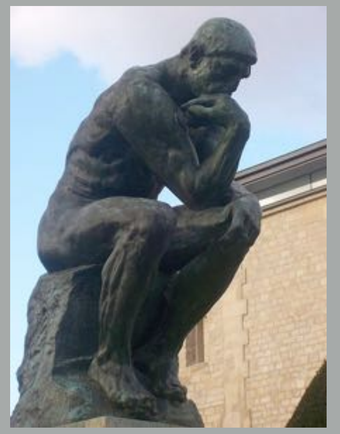
fully think through solving a problem?

react quickly to changes in its world?

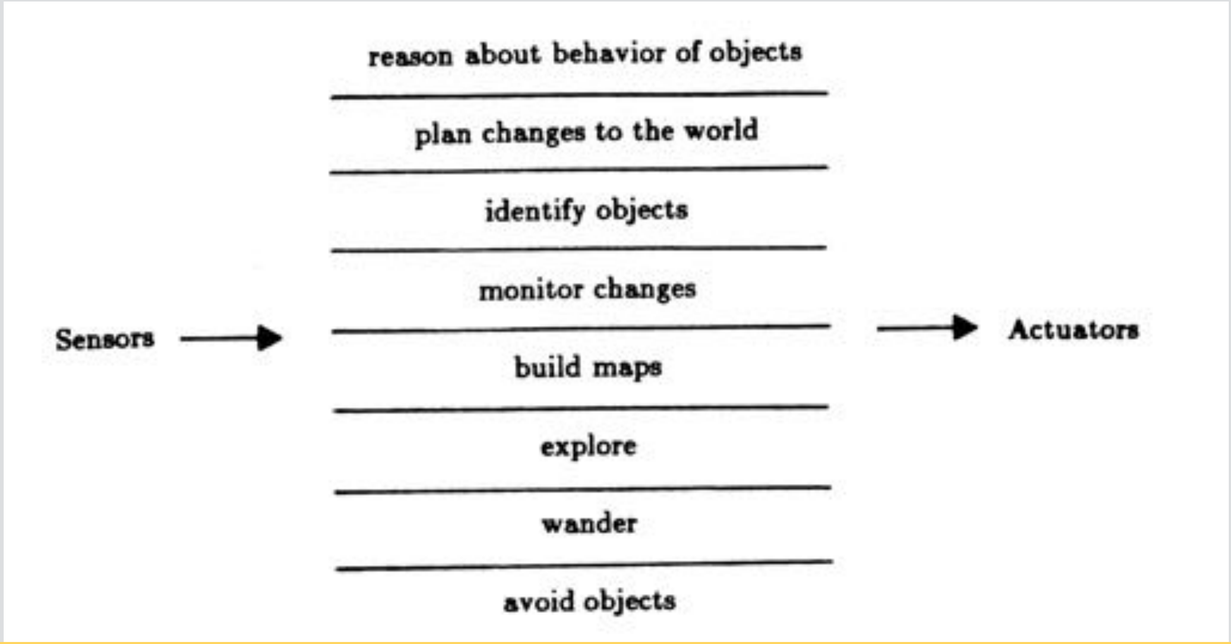
Deliberation v. Reaction

deliberative:
sense-plan-act,
path planning
motion planning





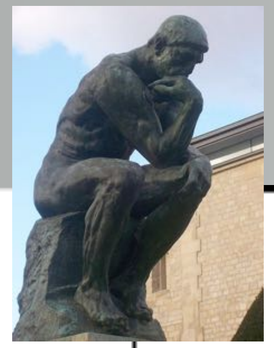
reaction:
controllers acting in parallel
subsumption,
Finite State Machine



Slide borrowed from Michigan Robotics autorob.org

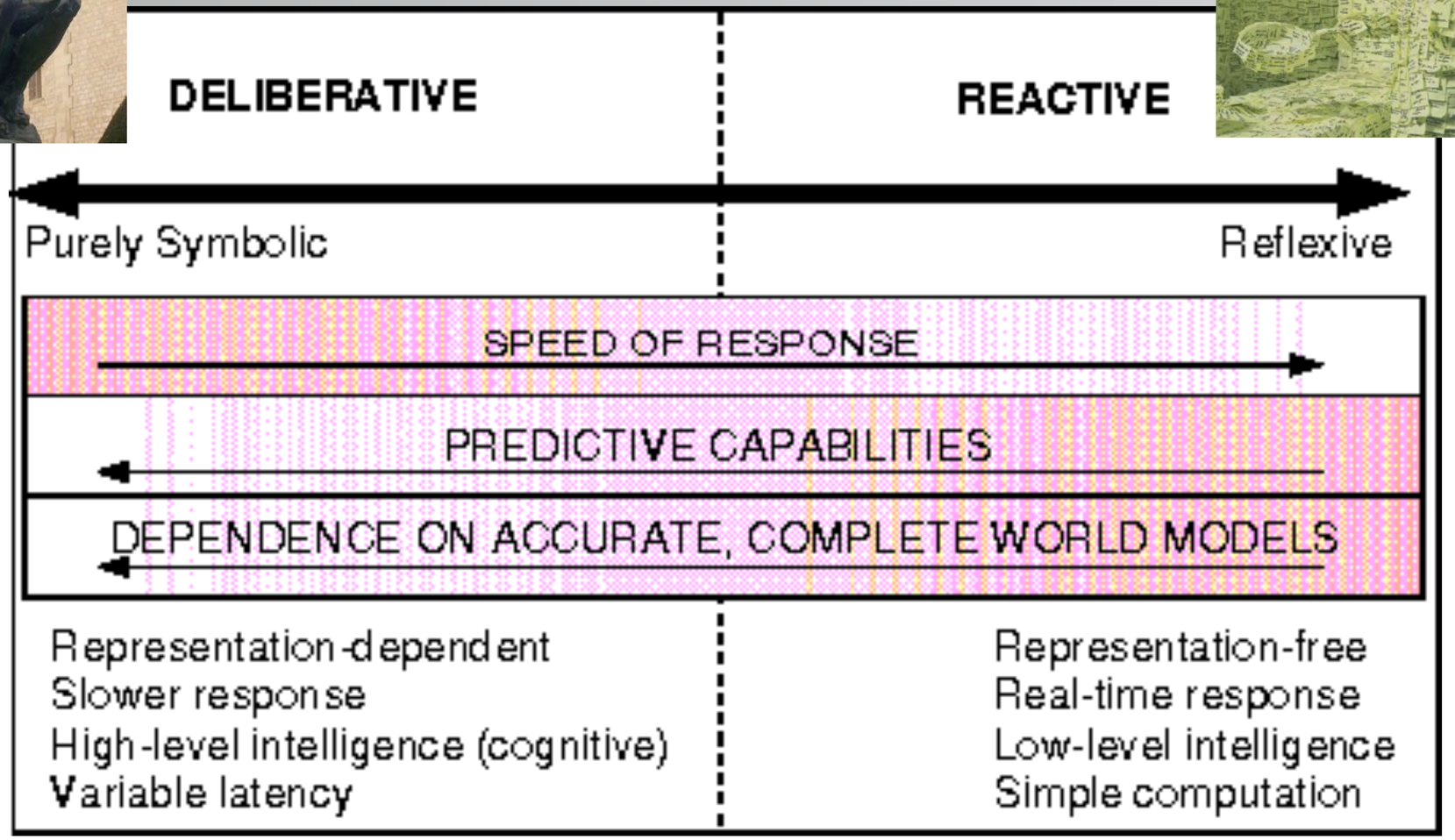
[Arkin 1998]

Deliberation-Reaction spectrum



Complete
Adaptive
Optimal
Slower

Requires complete model of the world

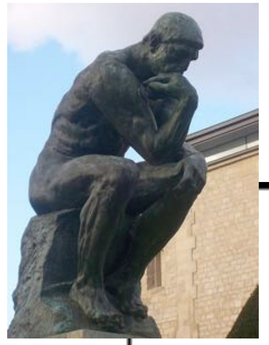


Faster
Cheaper
More robust
Forgetful

Requires a complete design of the problem

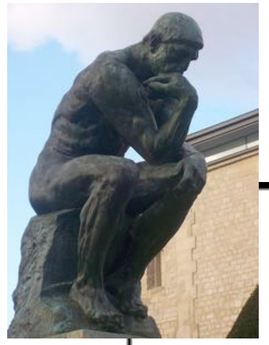
Slide borrowed from Michigan Robotics autorob.org

Examples?



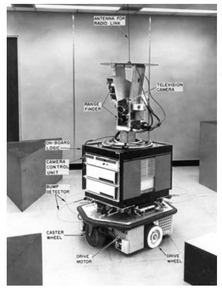
example???

Examples?



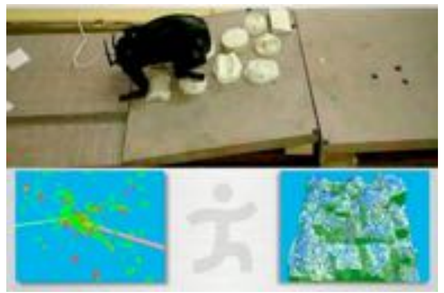
https://www.youtube.com/watch?v=jCB3pd-wBw0

Deliberation



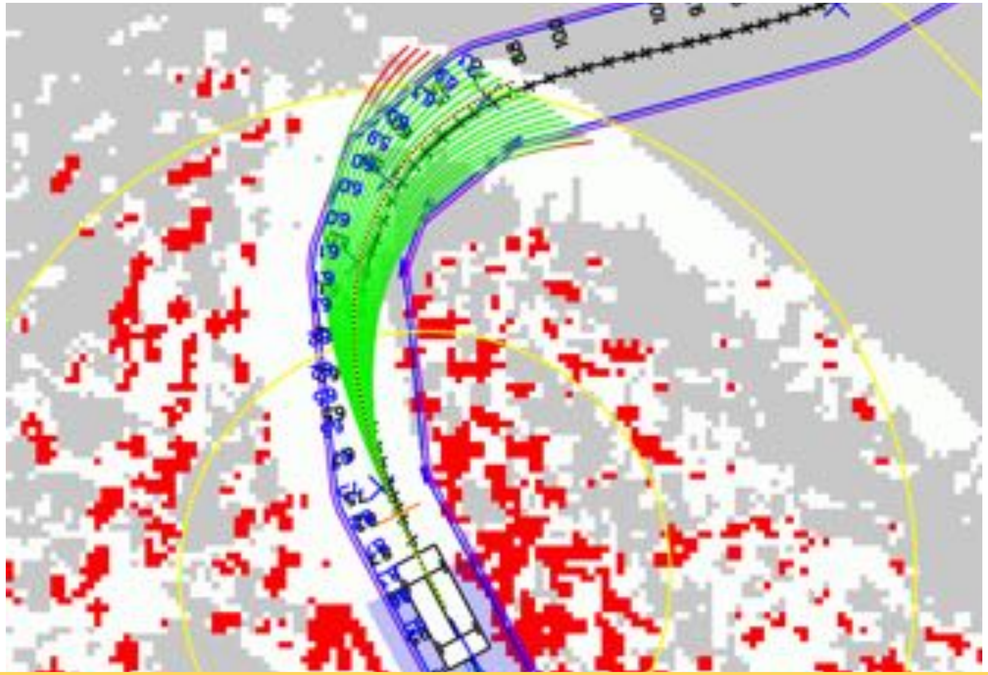
"Sense-Plan-Act" paradigm

- sense: build most complete model of world
  - GPS, SLAM, 3D reconstruction, affordances
- plan: search over all possible outcomes
  - Graph search, Roadmap planning
- act: execute plan through motor forces
  - PID control, Model predictive control



Stanley (Grand Challenge)

Road detection



Navigation

2005

MIT Talos (Urban Challenge)

2007

2013

Slide borrowed from Michigan Robotics autorob.org

Deliberation
requires a model of the world

Slide borrowed from Michigan Robotics autorob.org

Color+Depth Camera

Laser Rangefinder

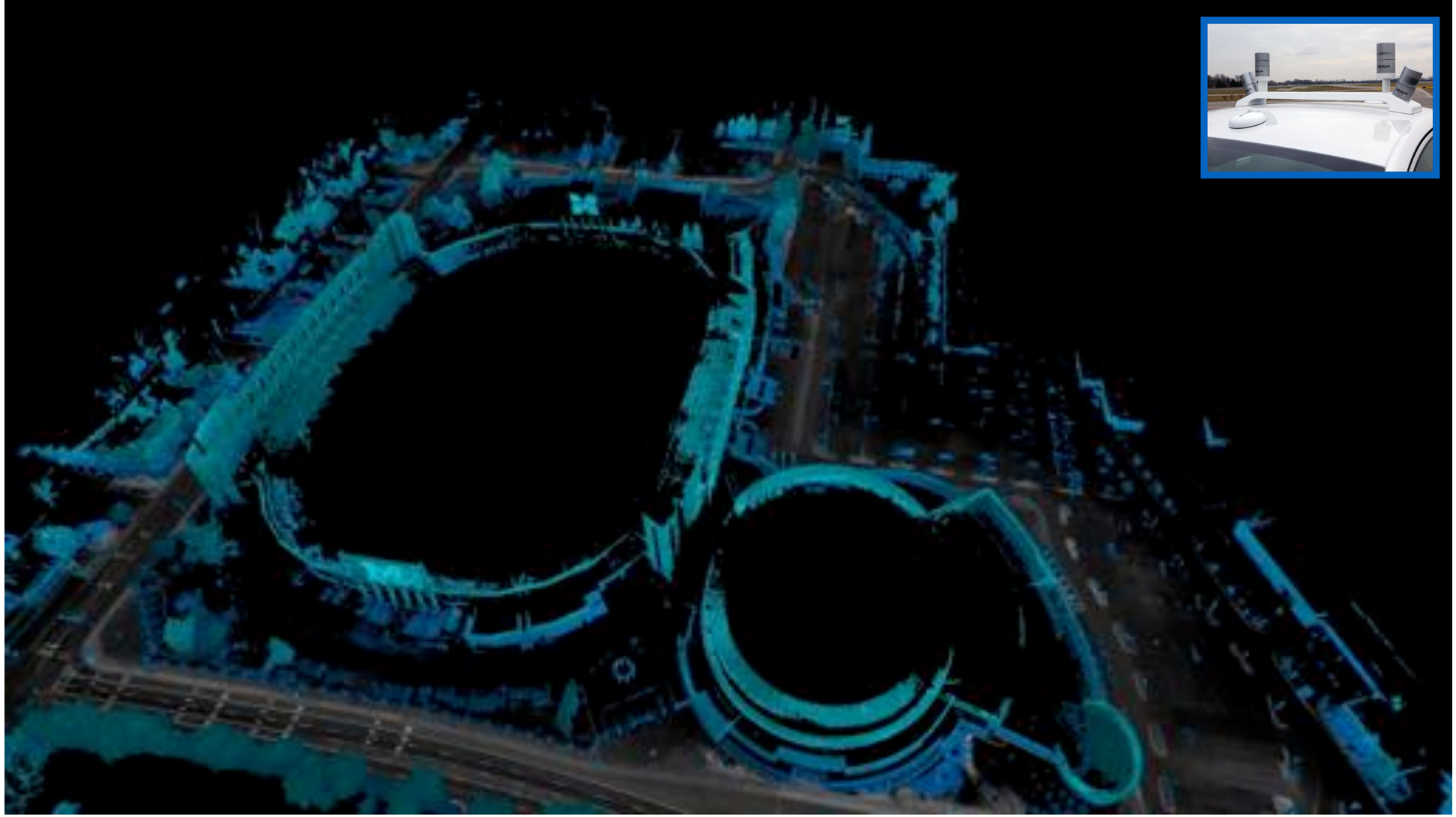
Slide borrowed from Michigan Robotics autorob.org

Simultaneous Localization and Mapping

Slide borrowed from Michigan Robotics autorob.org

Autonomous robot navigation from previously built map

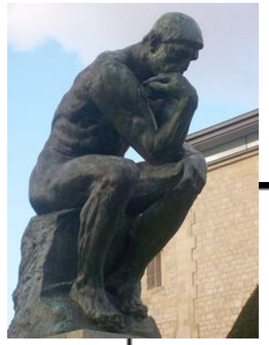
Michigan Next Generation Vehicle (Eustice, Olson et al.)



Autonomous Transportation

Michigan Next Generation Vehicle (Eustice, Olson et al.)

Examples?



DELIBERATIVE

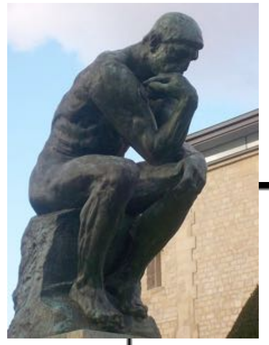
REACTIVE

Purely Symbolic

Reflexive

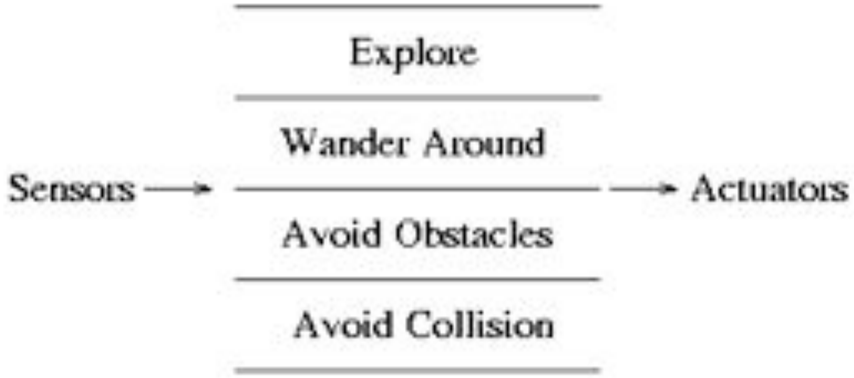
more common example???

Examples?



Slide borrowed from Michigan Robotics autorob.org

Reaction

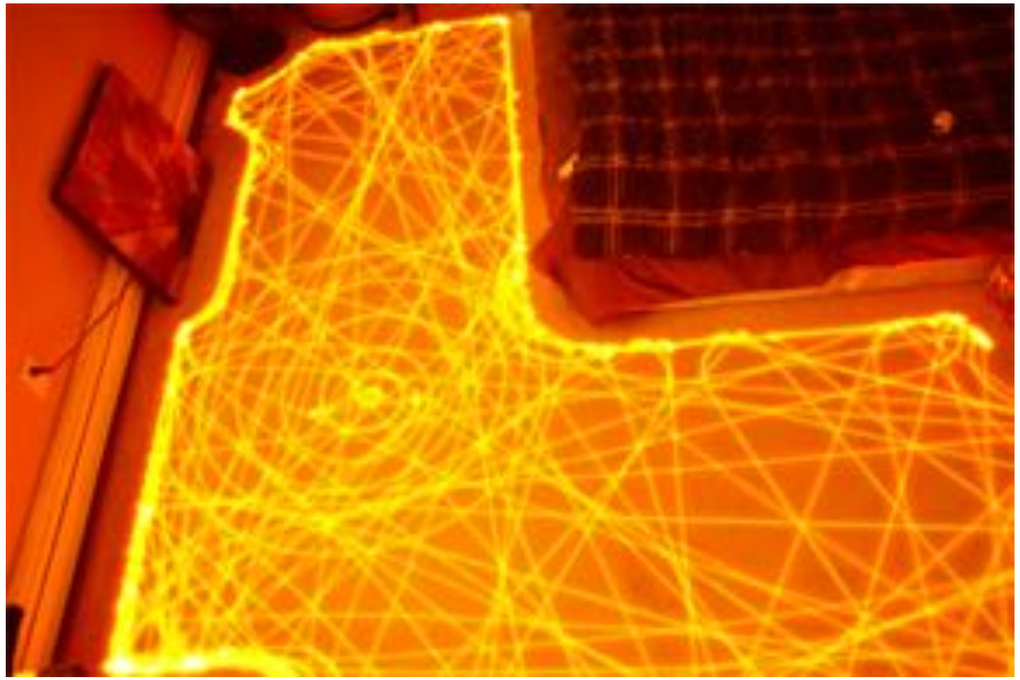


- No representation of state
  - Typically, fast hardcoded rules
- Embodied intelligence
  - behavior ← control + embodiment
  - Stigmergy (e.g, ant scouts using pheromones)

- Finite State Machines
  - most common
- Subsumption architecture
  - prioritized reactive policies

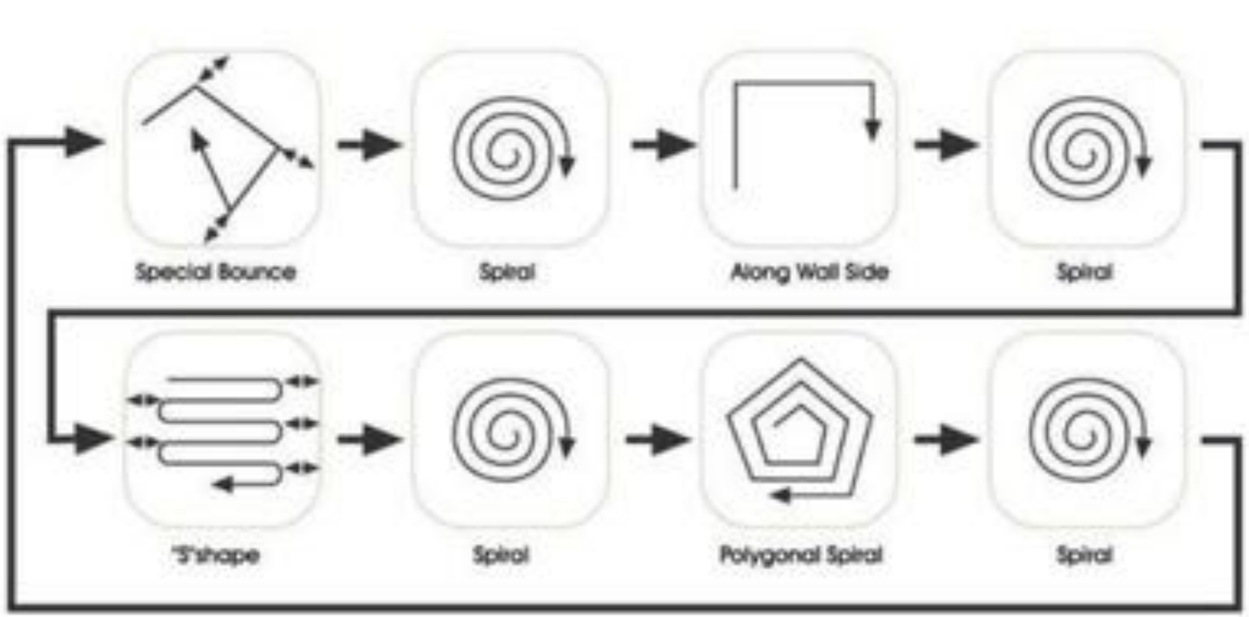
Slide borrowed from Michigan Robotics autorob.org

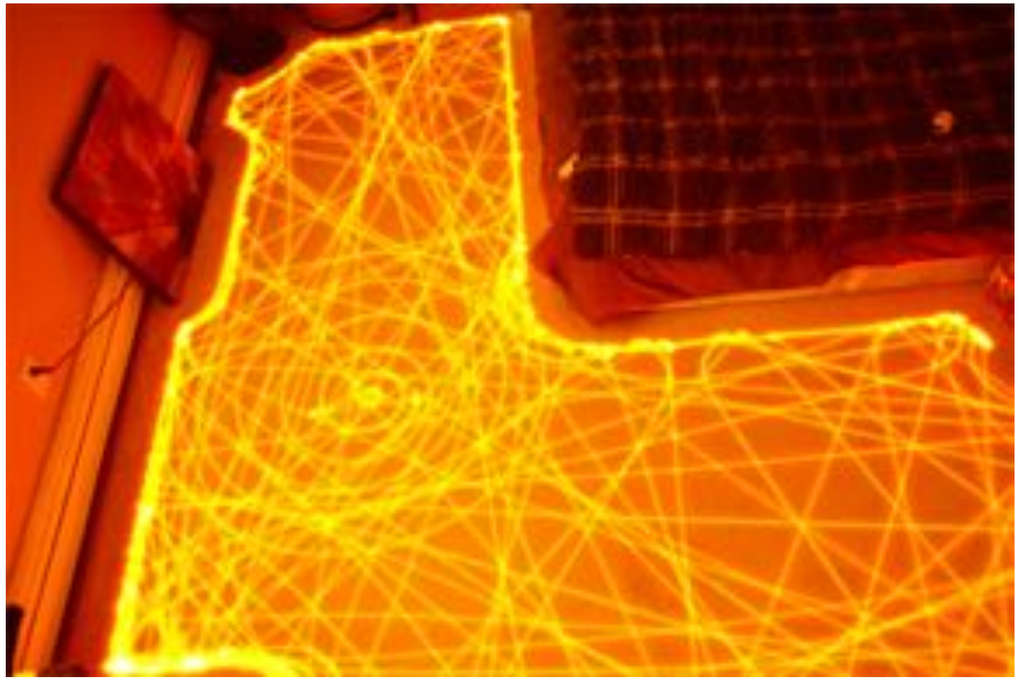
Roomba cleaning pattern



https://www.youtube.com/watch?v=G4ocrevf4ng

Vacuuming Finite State Machine





Slide borrowed from Michigan Robotics autorob.org

Manipulation Gaits

Collections of robust manipulation controllers

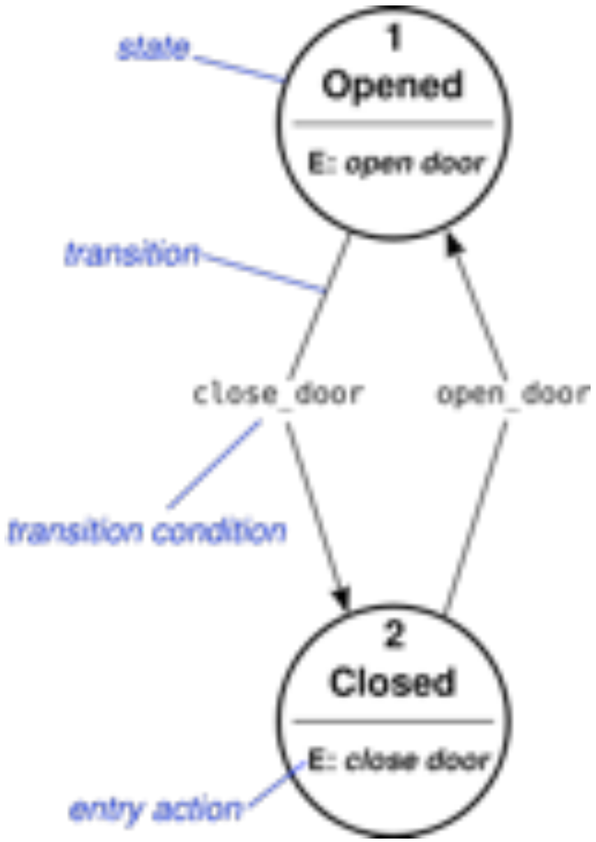
Slide borrowed from Michigan Robotics autorob.org

How do we computationally represent reactive control?

Slide borrowed from Michigan Robotics autorob.org

Finite State Machines

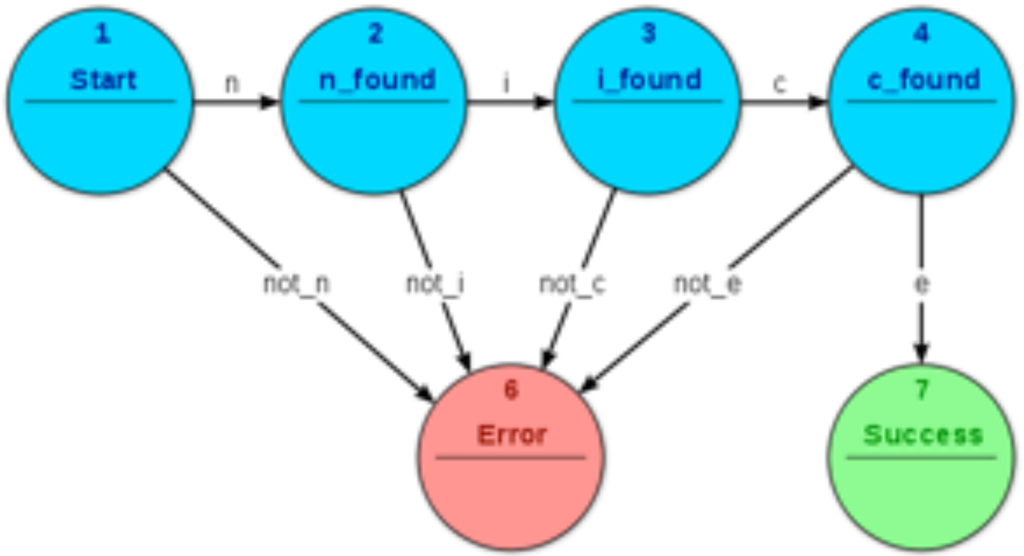
- Components
  - alphabet (or inputs)
    - “observations” in robotics
  - states (some robot action)
  - transitions (between states)
  - stopping condition
- Commonly, implemented as switch-case or if-else within a while loop



http://en.wikipedia.org/wiki/Switch_statement

“nice” recognizer

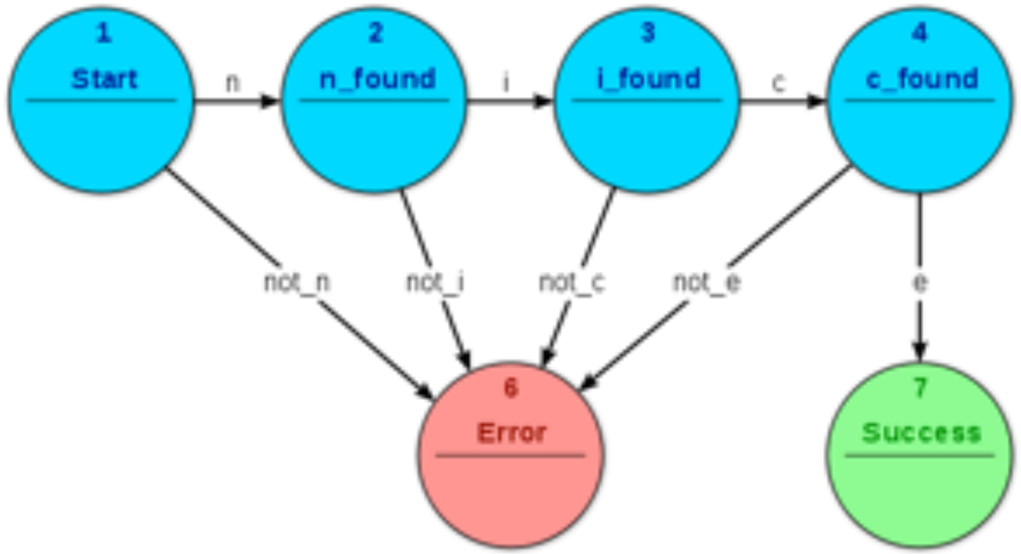
- recognize the string “nice” from input
  - if input is “nice”
    - output **success**
  - if input not “nice”
    - output **error**



- robotics uses
  - preconditions (enter state)
  - postconditions (exit state)

```
state ← start
while state != success and state != error
  token ← <next string character from input>
  switch (state):
    case start:
      if token = “n” then state ← n_found
      else state ← error
      break
    case n_found:
      if token = “i” then state ← i_found
      else state ← error
      break
    case i_found:
      if token = “c” then state ← c_found
      else state ← error
      break
    case c_found:
      if token = “e” then state ← success
      else state ← error
      break
end while loop
output ← state
```

“nice” recognizer



Consider input: “nice”

Consider input: “robotics”

Consider input: “niece”

Slide borrowed from Michigan Robotics autorob.org

Move to objects in sequence?

- How to move a mobile robot to a given sequence of objects?
  - yellow ball
  - green/orange landmark
  - pink landmark
  - orange/green landmark

Object Seeking

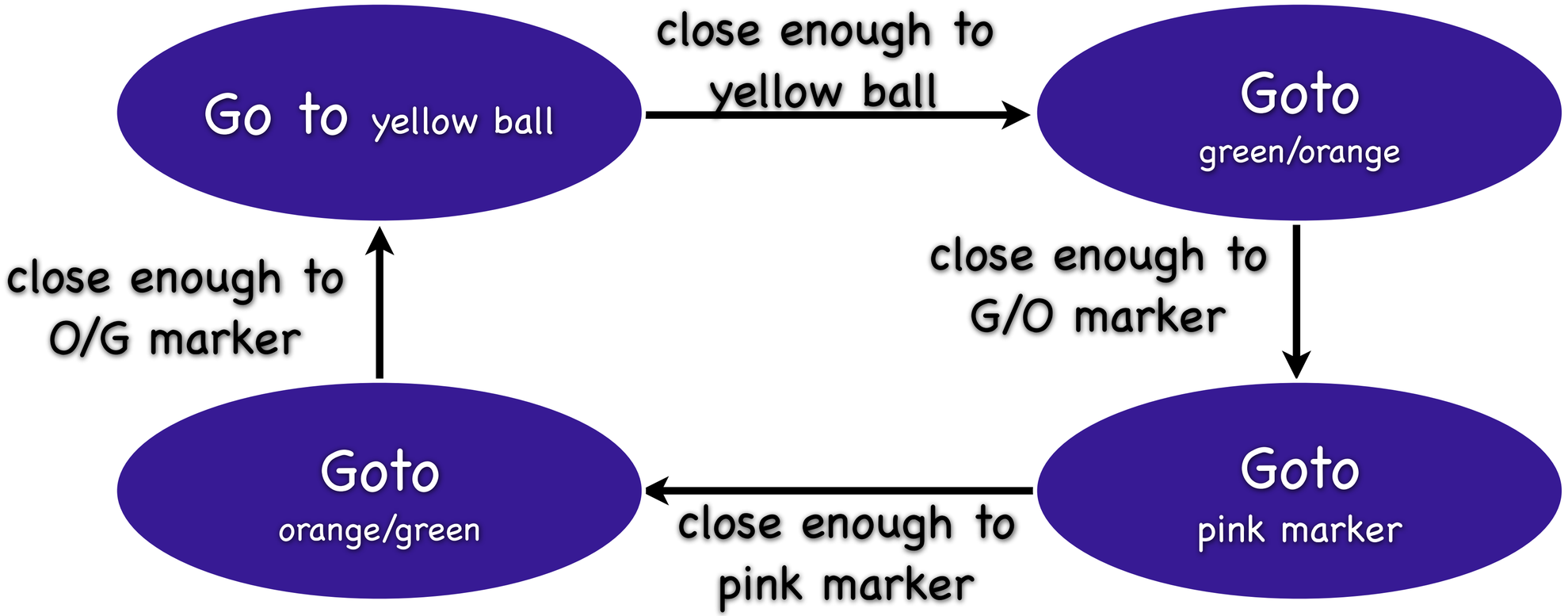
http://www.youtube.com/watch?v=-hOA0jMUggg

Slide borrowed from Michigan Robotics autorob.org

Move to objects in sequence?

- What are the states?
- What are the transitions?
- Preconditions for states?
- Postconditions for states?

Object seeking FSM

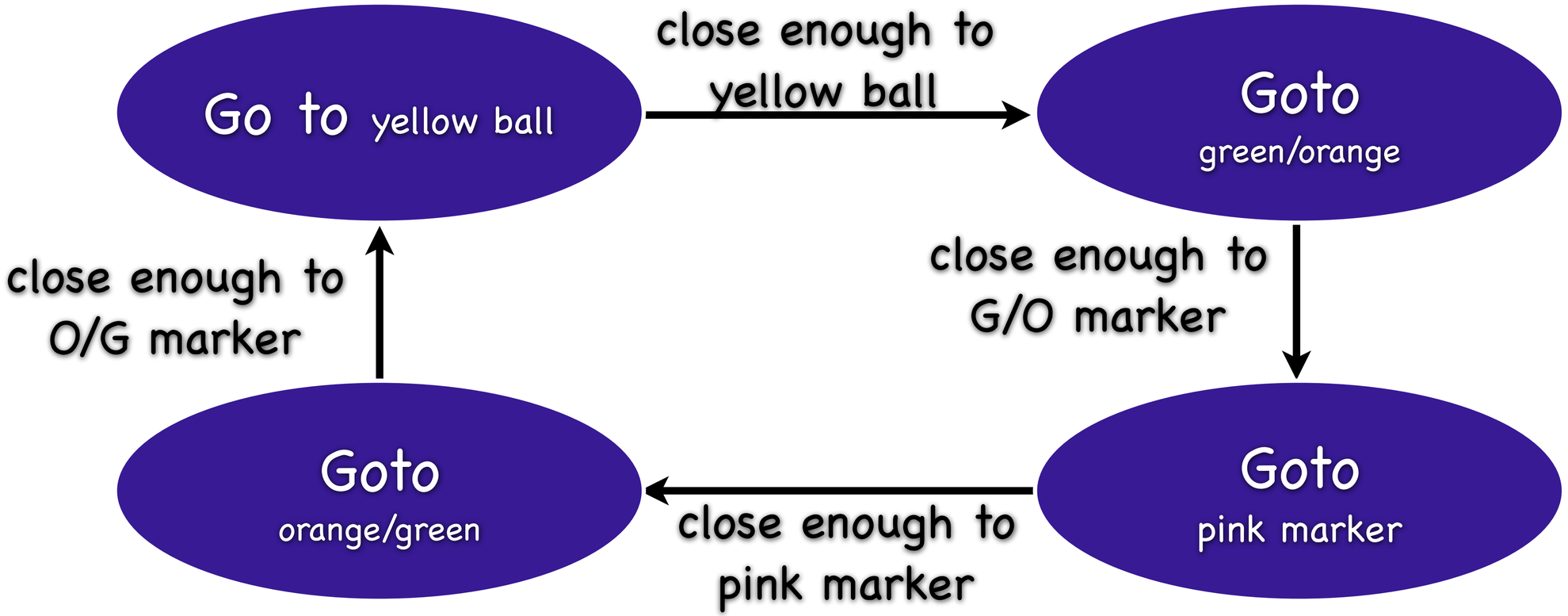


Slide borrowed from Michigan Robotics autorob.org

Object seeking FSM

How to implement state?

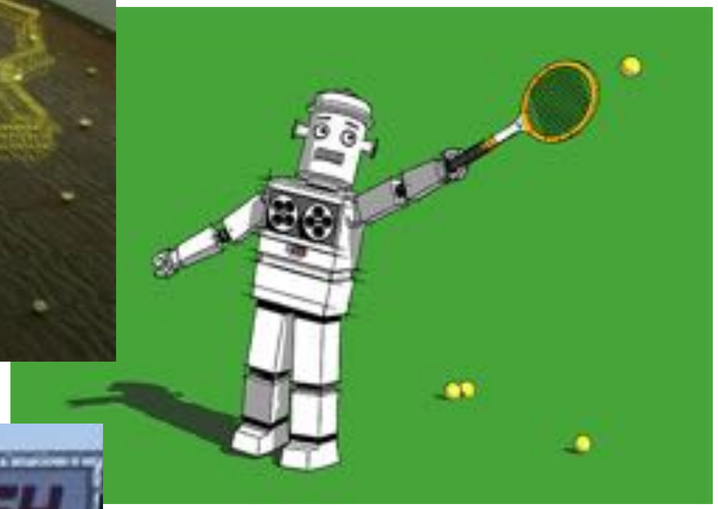
How to detect "close enough"?



Slide borrowed from Michigan Robotics autorob.org

FSMs for Other Tasks

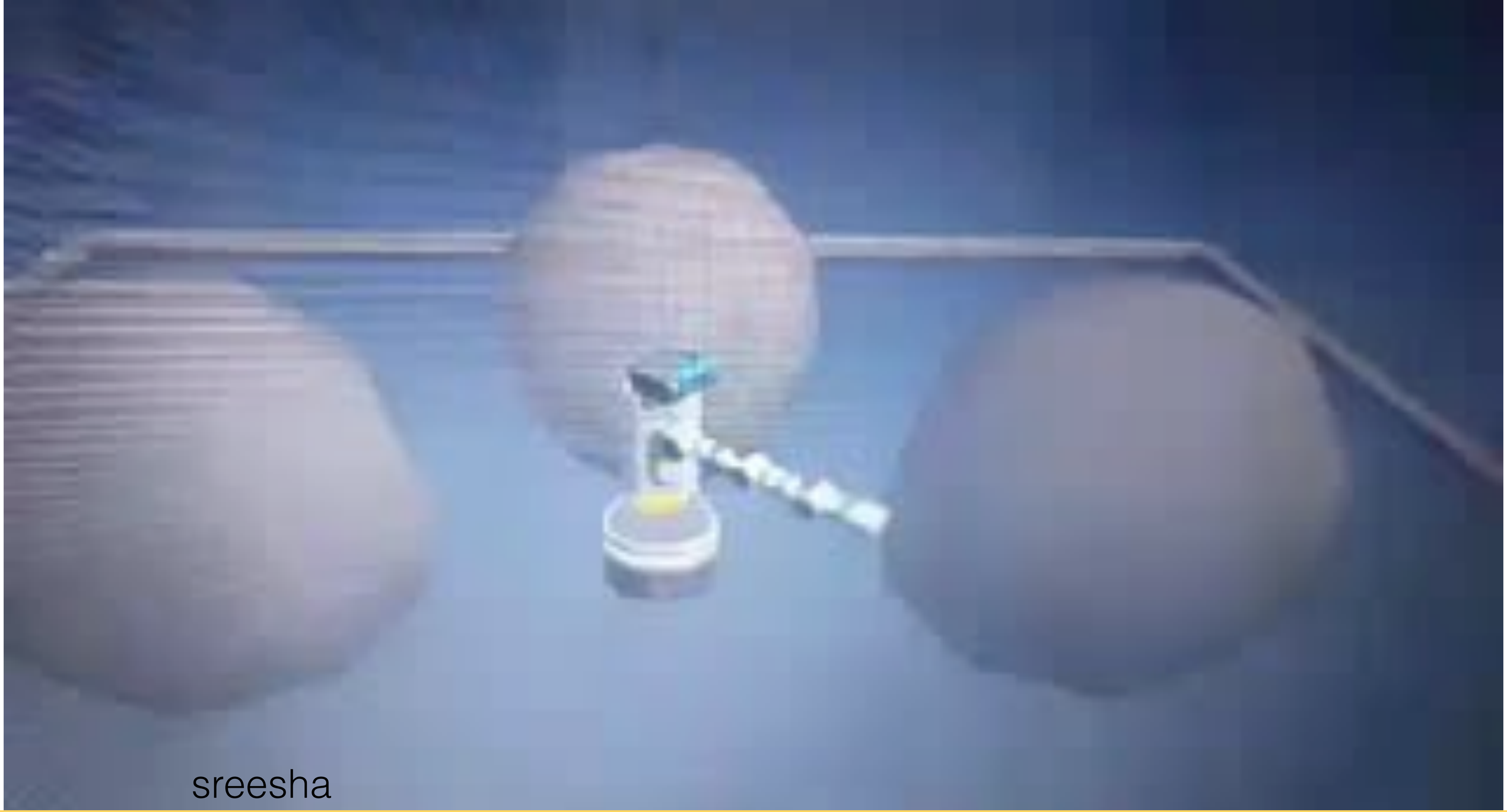
- Robot foraging?
- Robot tennis/pong?
- Pushing a ball into a goal?
- Vacuuming a room
- Driving a car?
- Robot dancing!

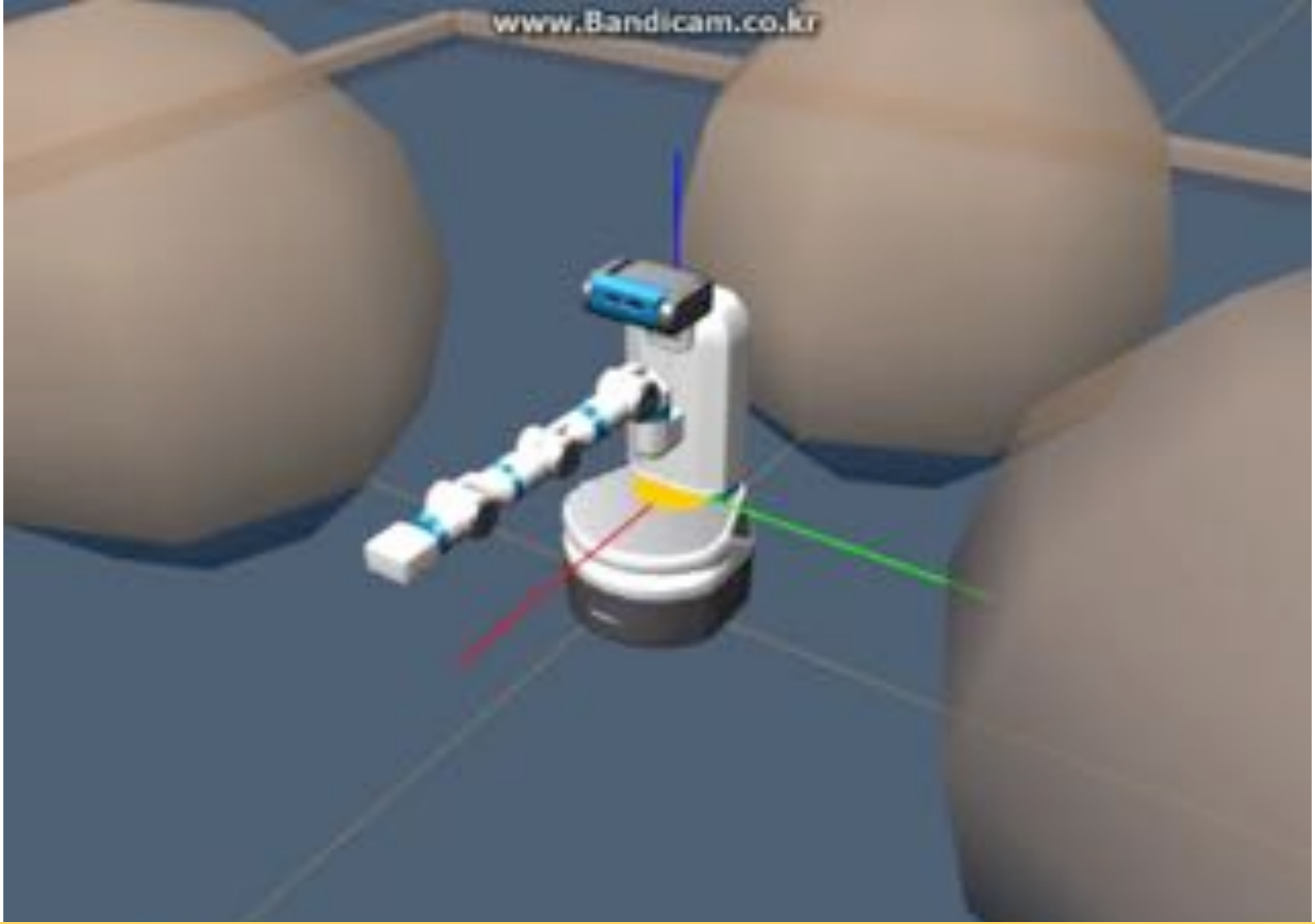


Slide borrowed from Michigan Robotics autorob.org

Assignment 4 robot dance (ohseejay)

https://www.youtube.com/watch?v=WyQ9aoB3bpI

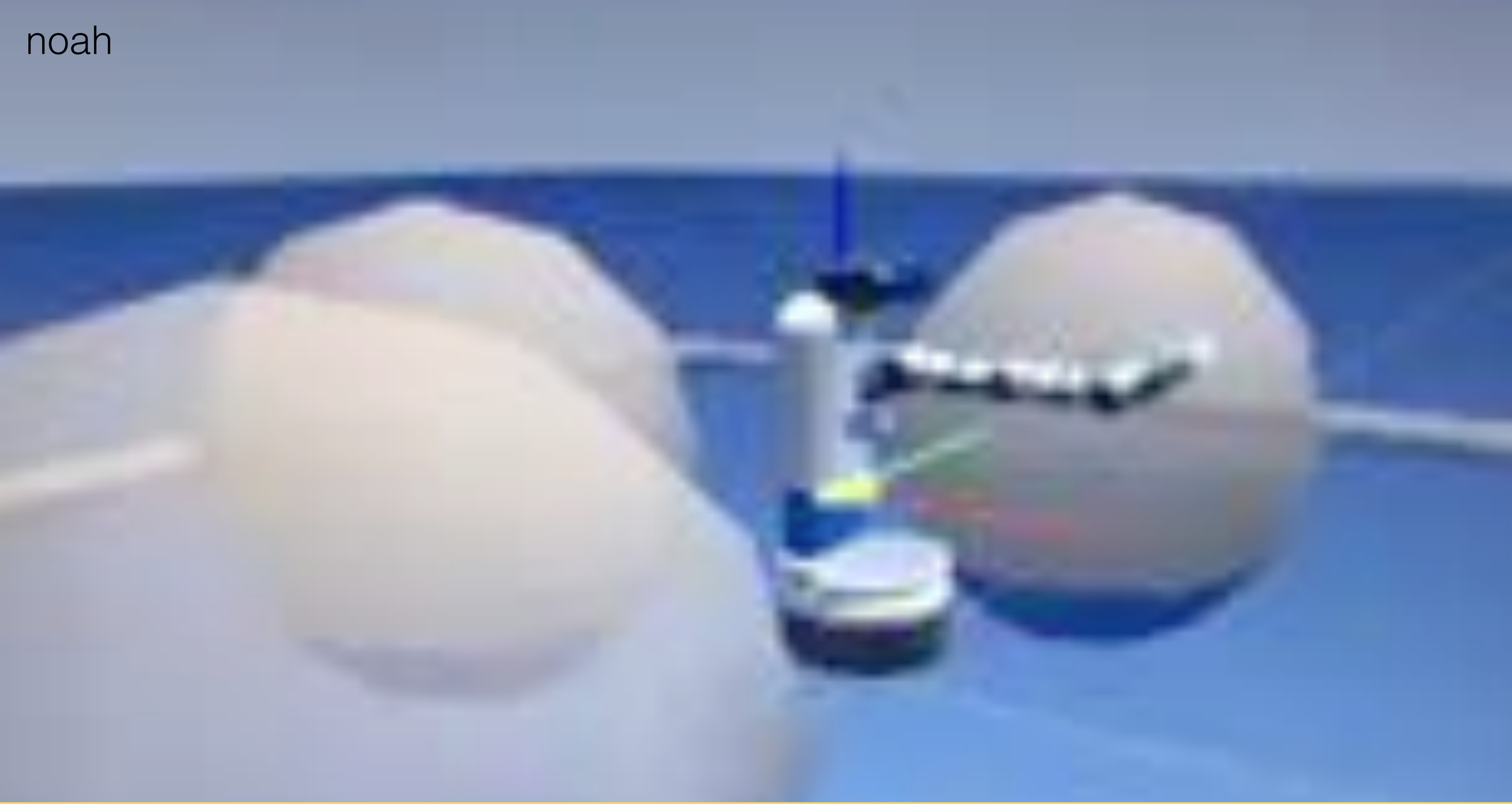

sreesha



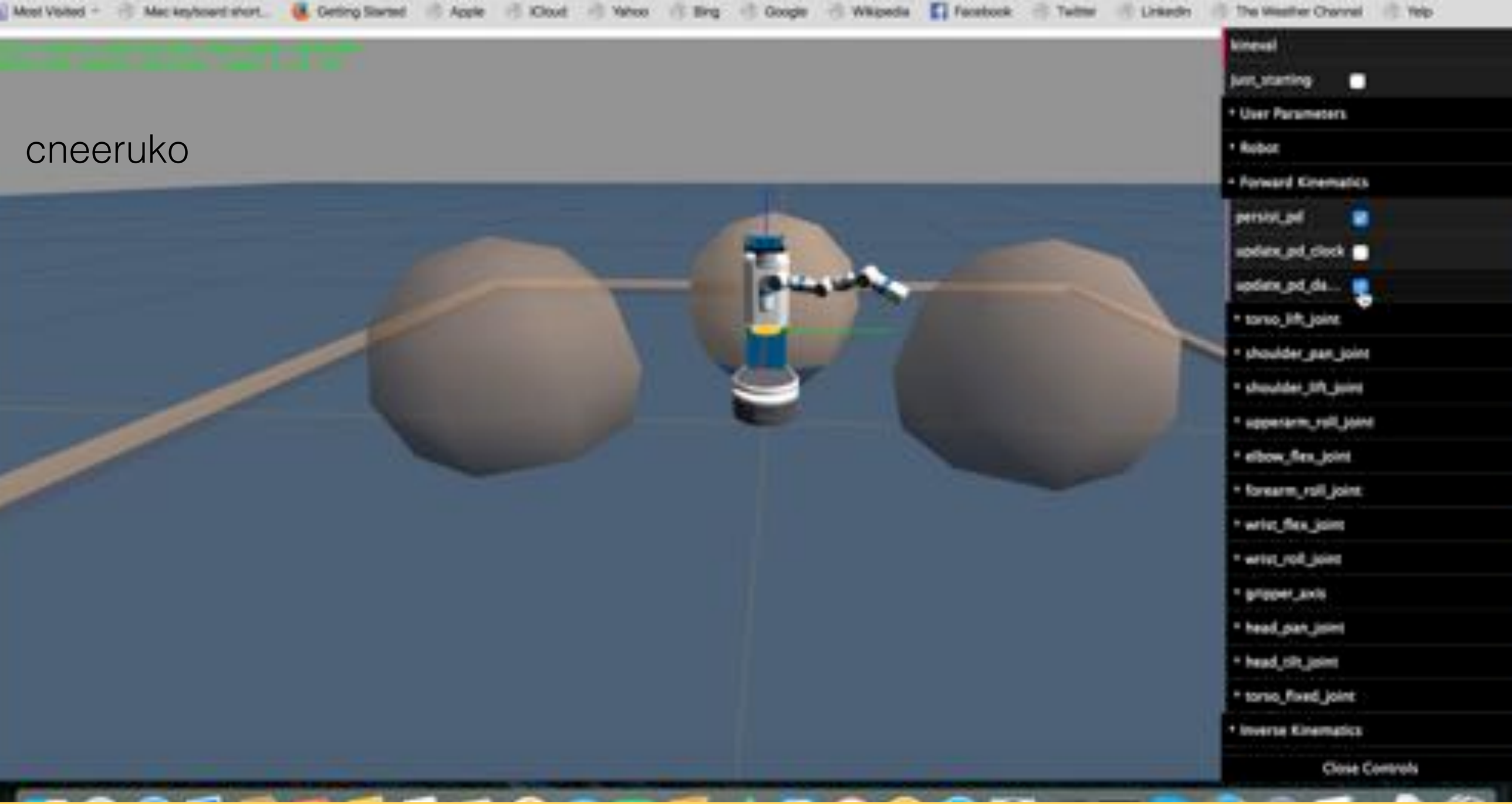
heostar

Slide borrowed from Michigan Robotics autorob.org

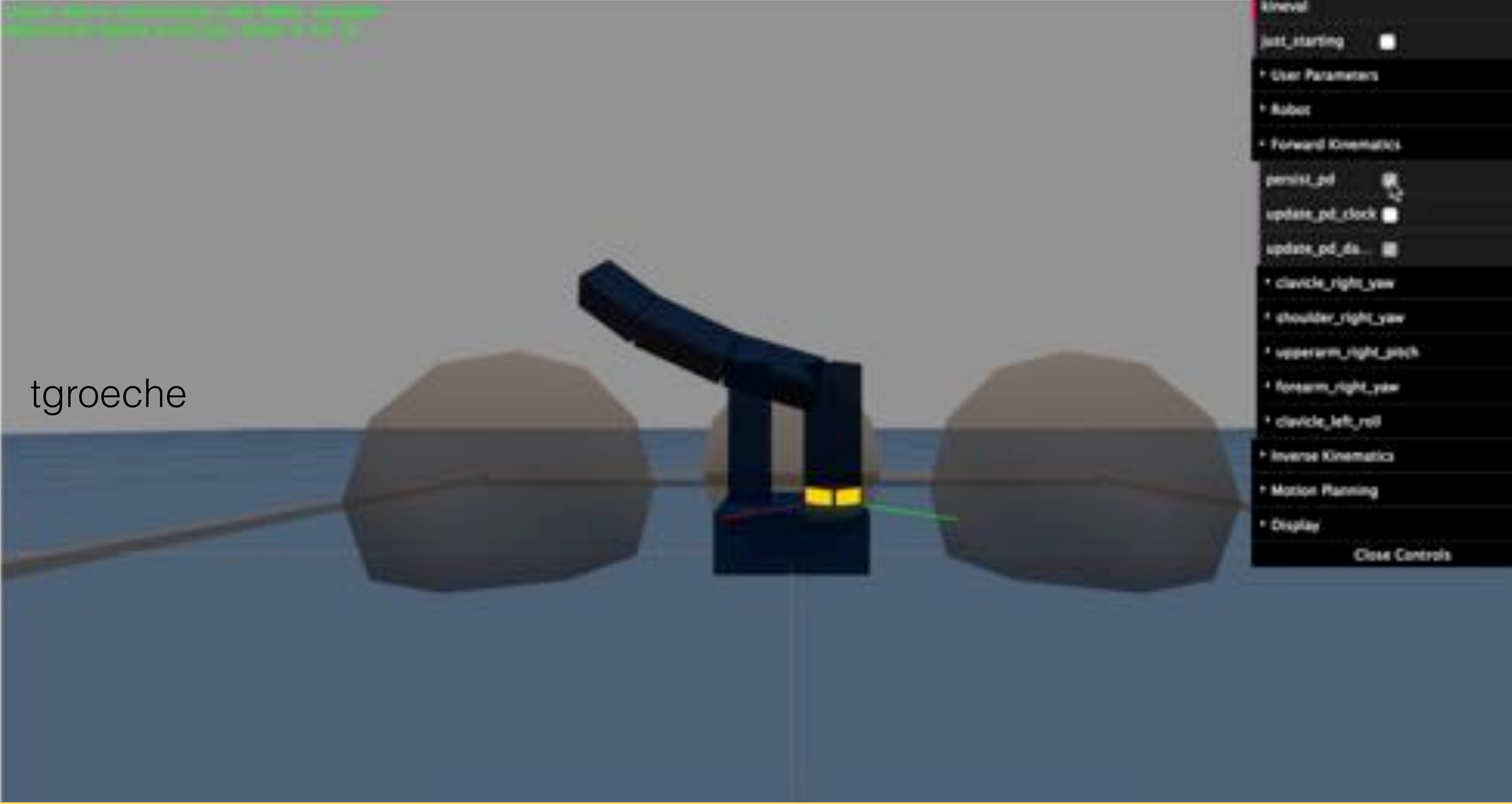
noah



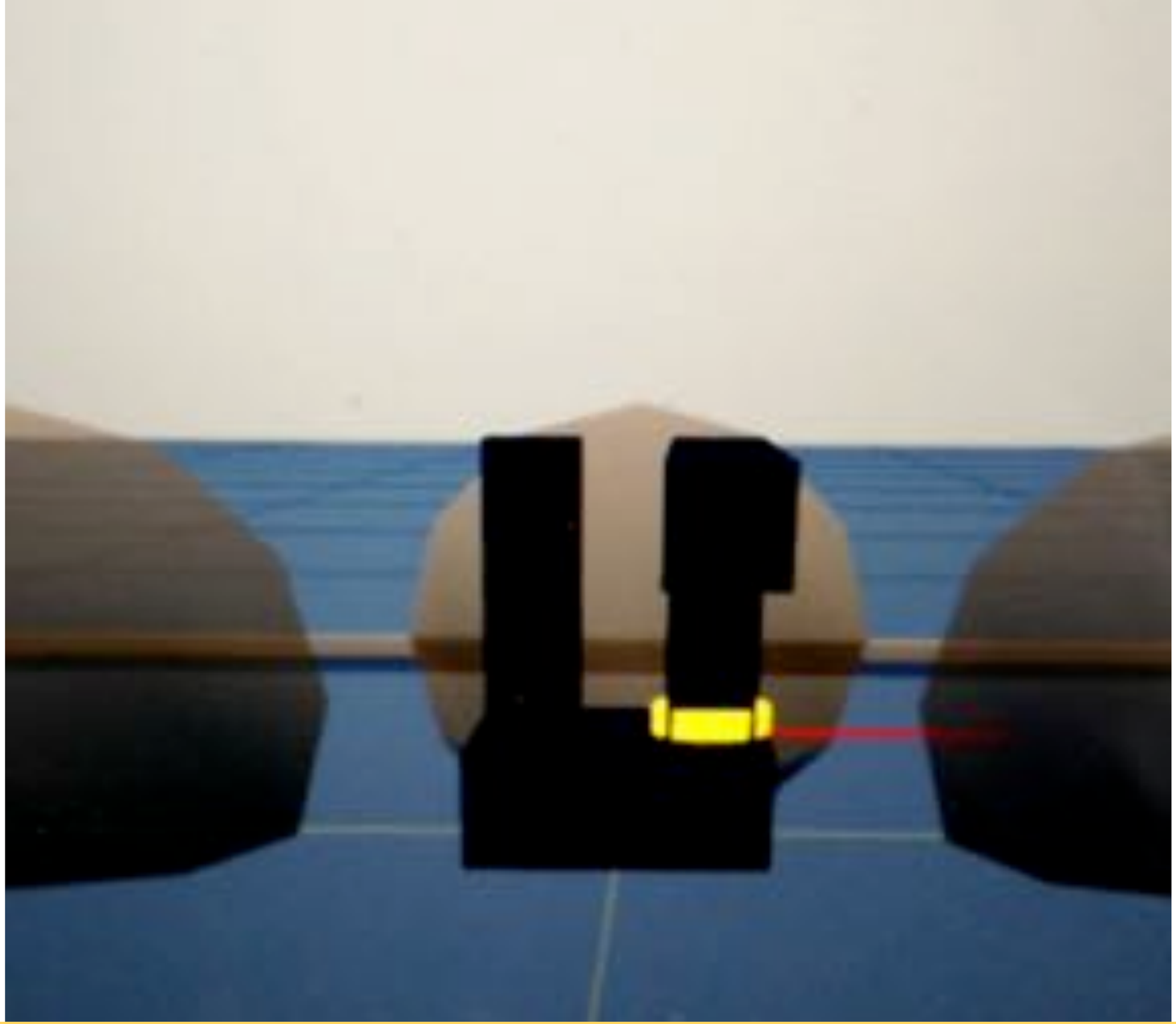
Slide borrowed from Michigan Robotics autorob.org



Slide borrowed from Michigan Robotics autorob.org

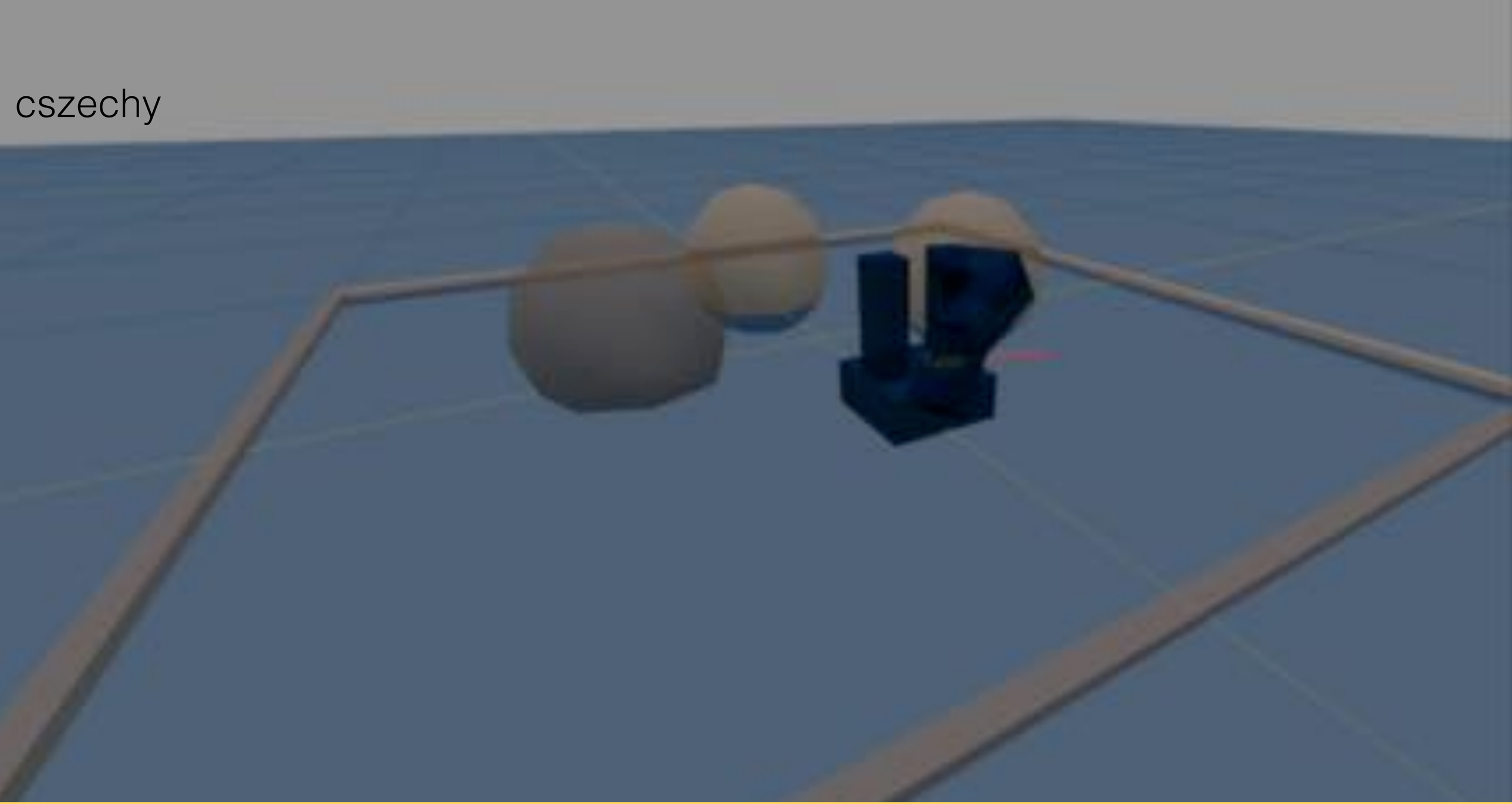


Slide borrowed from Michigan Robotics autorob.org



ankit

cszechy



Slide borrowed from Michigan Robotics autorob.org

cszechy
ohseejay

Let’s generalize FSMs for robot control

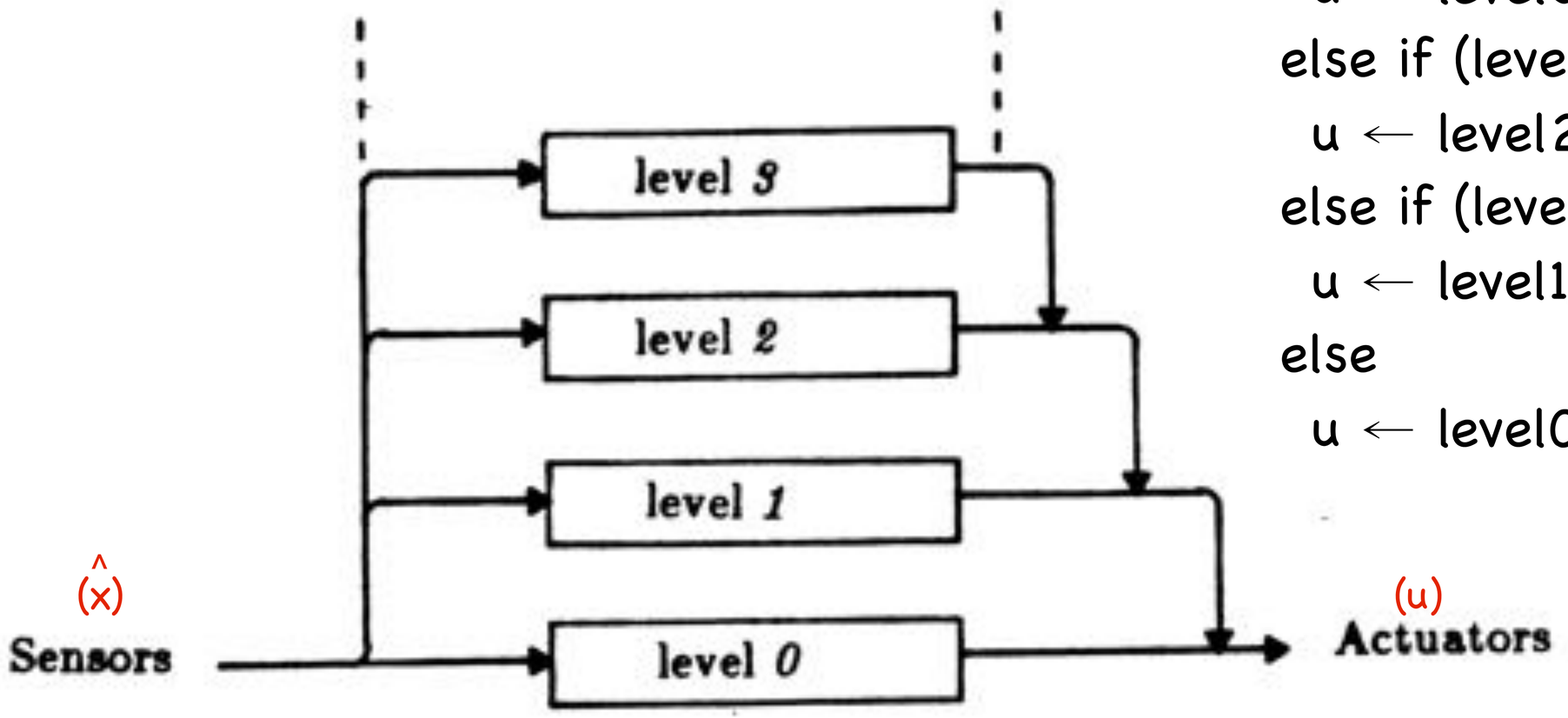
Slide borrowed from Michigan Robotics autorob.org

Subsumption Architecture

[Brooks 1986]

- Generalization of FSM-based control
- Collection of modular reactive controllers in a priority hierarchy
- Controllers can be FSMs
- Large nested if-else statement
- Most robots are controlled by some form of subsumption

Subsumption Architecture




```
if (level3_condition)
  u ← level3_control
else if (level2_condition)
  u ← level2_control
else if (level1_condition)
  u ← level1_control
else
  u ← level0_control
```


Slide borrowed from Michigan Robotics autorob.org

Subsumption Design Process

1. Divide your problem into basic competencies ordered simple to more complex. Designate a level for each basic competency.

2. Subdivide each level into multiple simple components that interact through shared variables. Limit the sharing of variables among levels to avoid incomprehensible code.

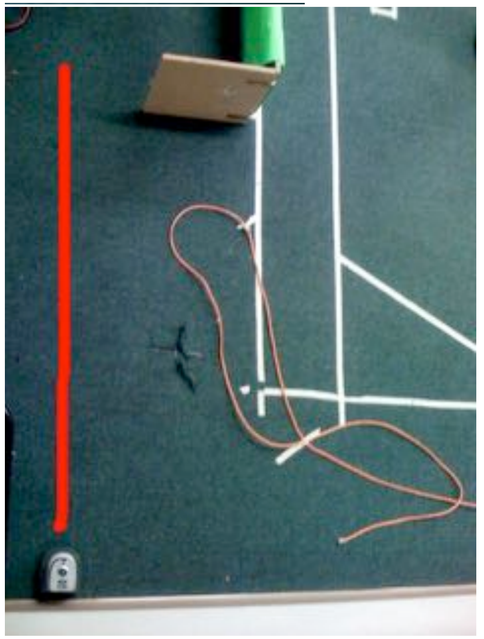
3. Implement each module as a separate light-weight thread. You might think of setting the priorities for these threads s.t. modules in a given level have the same priority.

4. Implement "arbitration" processes for suppression and inhibition as one or more separate that serve to control access to shared variables. You might want to control access using semaphores.

Slide borrowed from Michigan Robotics autorob.org

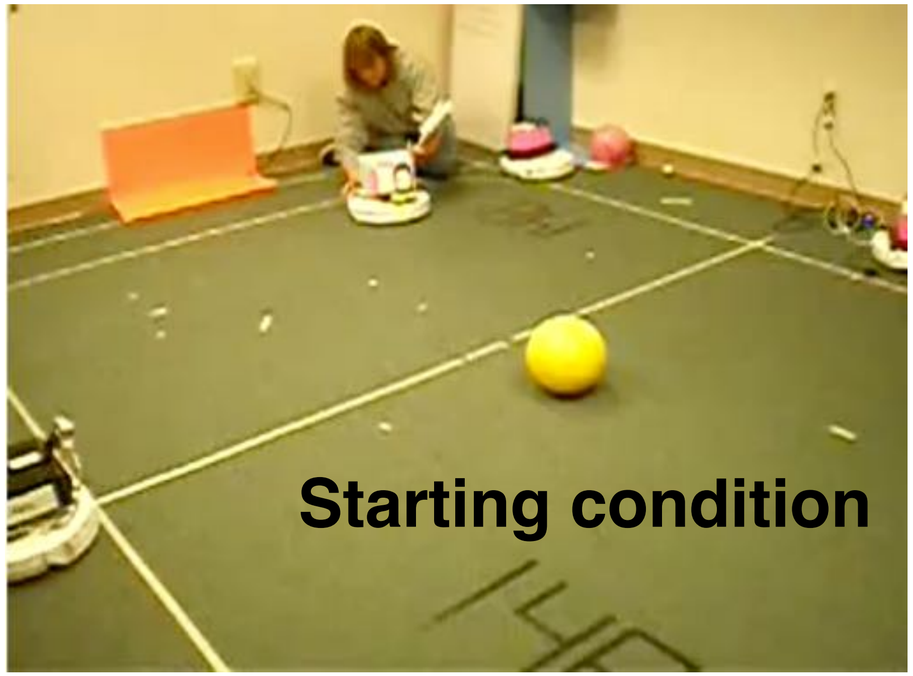
Subsumption for robot soccer

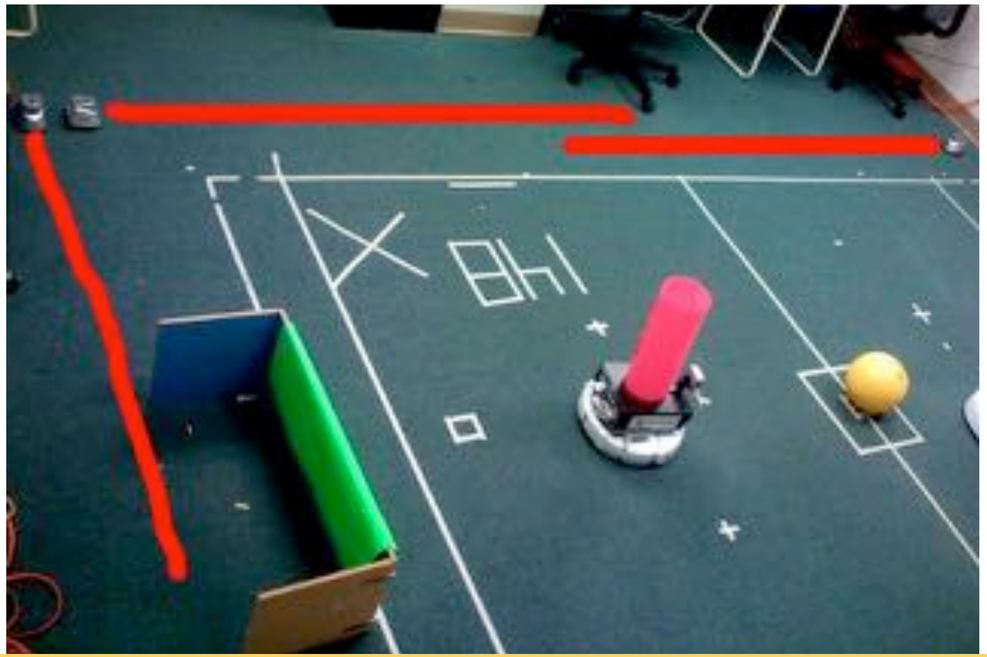
- Propose modules and priority?



What behavior will result?

1. Avoid IR Wall
2. Avoid Robot
3. Avoid Fiducial
4. Bumper Hit
5. Go To Opposite Goal
6. Go To Any Goal
7. Line Up On Ball
8. Go To Ball
9. Score Goal
10. At Ball
11. Look For Ball





Slide borrowed from Michigan Robotics autorob.org

Snappy's Subsumption: Goal Scoring

1. Avoid IR Wall
2. Avoid Robot
3. Avoid Fiducial
4. Bumper Hit
5. Go To Opposite Goal
6. Go To Any Goal
7. Line Up On Ball
8. Go To Ball
9. Score Goal
10. At Ball
11. Look For Ball

Goal Scoring Challenge - Put ball on orange

Slide borrowed from Michigan Robotics autorob.org

Snappy's Subsumption: Navigate to Ball

1. Avoid IR Wall
2. Avoid Robot
3. Avoid Fiducial
4. Bumper Hit
5. Go To Opposite Goal
6. Go To Any Goal
7. Line Up On Ball
8. Go To Ball
9. Score Goal
10. At Ball
11. Look For Ball

Slide borrowed from Michigan Robotics autorob.org

Snappy in competition

Slide borrowed from Michigan Robotics autorob.org

Are there other methods of decision making?

Types of Decision Making

- Deliberative (Planner-based) Control
  - "Think hard, act later."
- Reactive Control
  - "Don't think, (re)act."
- Hybrid Control
  - "Think and act separately & concurrently."
- Behavior-Based Control
  - "Think the way you act."

Next lecture:
Inverse Kinematics