Lecture 15
Planning - New Frontiers
& Motion Control

Course Logistics

- Quiz 7 was posted yesterday and was due today at noon.
  - We will start posting quiz at 3pm starting next week.
- Project 5 was posted on 03/05 and is due on 03/24 (coming Monday).
- Project 6 will be posted on 03/24 and will be due on 04/02.
- Forming groups for P7 and Final Project
  - Google-form was sent on Ed discussion board.
  - Please form groups of 4 by 03/24 (coming Monday).
  - UNITE students will likely have different group formations. Karthik has reached out via email.

Course Logistics

- Quiz 7 was posted yesterday and
  - We will start posting quiz at 3p
- Project 5 was posted on 03/05 ar
- Project 6 will be posted on 03/24
- Forming groups for P7 and Final
  - Google-form was sent on Ed d
  - Please form groups of 4 by 03/
  - UNITE students will likely have

ec #	Date	Topic	Project Announcement	Project Due	Pre-class Quiz
1	01/22	Introduction			
2	01/27	Planning I - Path Planning			
3	01/29	Linear Algebra Refresher	P1: JS, BFS, DFS		Q1
4	02/03	Representations I - Transformations			
5	02/05	Representations II - Rotations - Quaternions	P2: Forward Kinematics	P1: Due	Q2
6	02/10	Manipulation I - Forward Kinematics			
7	02/12	Manipulation II - Inverse Kinematics	P3: Robot Dance	P2: Due	Q3
8	02/17	Manipulation III - Inverse Kinematics			
9	02/19	Manipulation - New Frontiers	P4: Inverse Kinematics	P3: Due	Q4
10	02/24	Planning II - Bug Algorithms			
11	02/26	Planning III - Configuration Space			Q5
12	03/03	Planning IV - Sampling-based Planning			
13	03/05	Planning V - Collision Detection	P5: Planning	P4: Due	Q6
14	03/10	Spring Break			
15	03/12	Spring Break			
16	03/17	Planning VI - Potential Fields	*Forming groups for P7 and FP*		
17	03/19	Planning - New Frontiers - Motion Control			Q7
18	03/24	Mobile Robotics I - Probability	P6: Mobile Manipulation	P5: Due & Groups	
19	03/26	Mobile Robotics II - Sensor and Motion Models			Q8
20	03/31	Mobile Robotics III - Kalman	FP: Proposals Request		
21	04/02	Mobile Robotics IV - Localization	P7: Real Robot Challenge	P6: Due	Q9
22	04/07	Mobile Robotics V - Localization			
23	04/09	Mobile Robotics VI - Mapping			Q10
24	04/14	Mobile Robotics VII - SLAM		FP: Proposals Due	
25	04/16	Open Ended Final Project Pitches			Q11
26	04/21	Open Ended Final Project Pitches			
27	04/23	Open Ended Final Project Pitches		P7: Due	Q12
28	04/28	Guest Lecture - Adam Imdieke (PhD student)			
29	04/30	Guest Lecture - Xun Tu (PhD student)		FP Posters Due	*Extra Q13*
30	05/05	**Poster Day**		**FP Videos Due**	

Where are we in the course?

Representations

1. Transformations
2. Rotations & Quaternions

Manipulation

1. Forward Kinematics
2. Inverse Kinematics

Planning

1. Path Planning
2. Bugs
3. Configuration space
4. Sampling based planners
5. Potential Fields
6. Collision Detection

Motion Control

Mobile Robotics

Motion Planning

Sampling-based Planning

PRM, RRT, …

Heuristic Discrete Search Methods

Graph, A, …*

Optimization-based Planning

TrajOPT, CHOMP, …

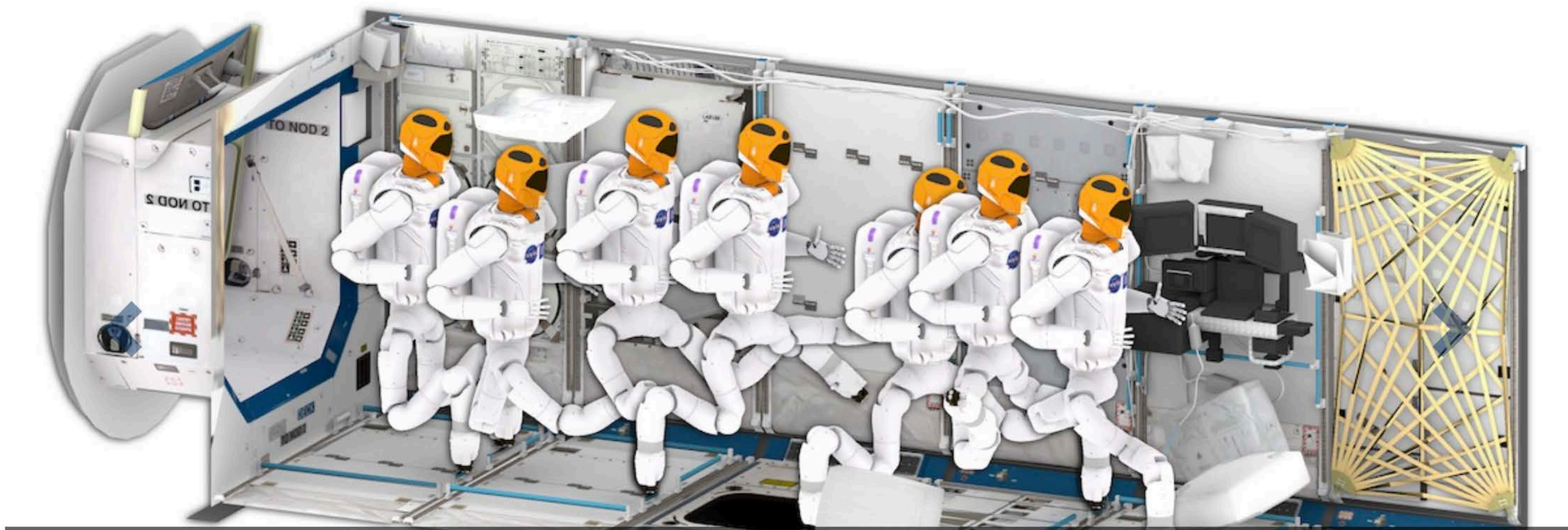
Demo

https://demonstrations.wolfram.com/ProbabilisticRoadmapMethodForRobotArm/

Motion Planning

OMPL Download Documentation Gallery OMPL Integrations Code Issues Community About Blog Search

The Open Motion Planning Library



OMPL used in ROS/MoveIt

OMPL is the default planning library in MoveIt and has been used for many robots. Here, OMPL is used to plan footsteps for NASA's Robonaut2 aboard the International Space Station.

OMPL, the Open Motion Planning Library, consists of many state-of-the-art sampling-based motion planning algorithms. OMPL itself does not contain any code related to, e.g., collision checking or visualization. This is a deliberate design choice, so that OMPL is not tied to a particular collision checker or visualization front end. The library is designed so it can be easily integrated into systems that provide the additional needed components.

OMPL.app, the front-end for OMPL, contains a lightweight wrapper for the FCL and PQP collision checkers and a simple GUI based on PyQt / PySide. The graphical front-end can be used for planning motions for rigid bodies and a few vehicle types (first-order and second-order cars, a blimp, and a quadrotor). It relies on the Assimp library to import a large variety of mesh formats that can be used to represent the robot and its environment.

Download version 1.6.0
Released: Jan 16, 2023

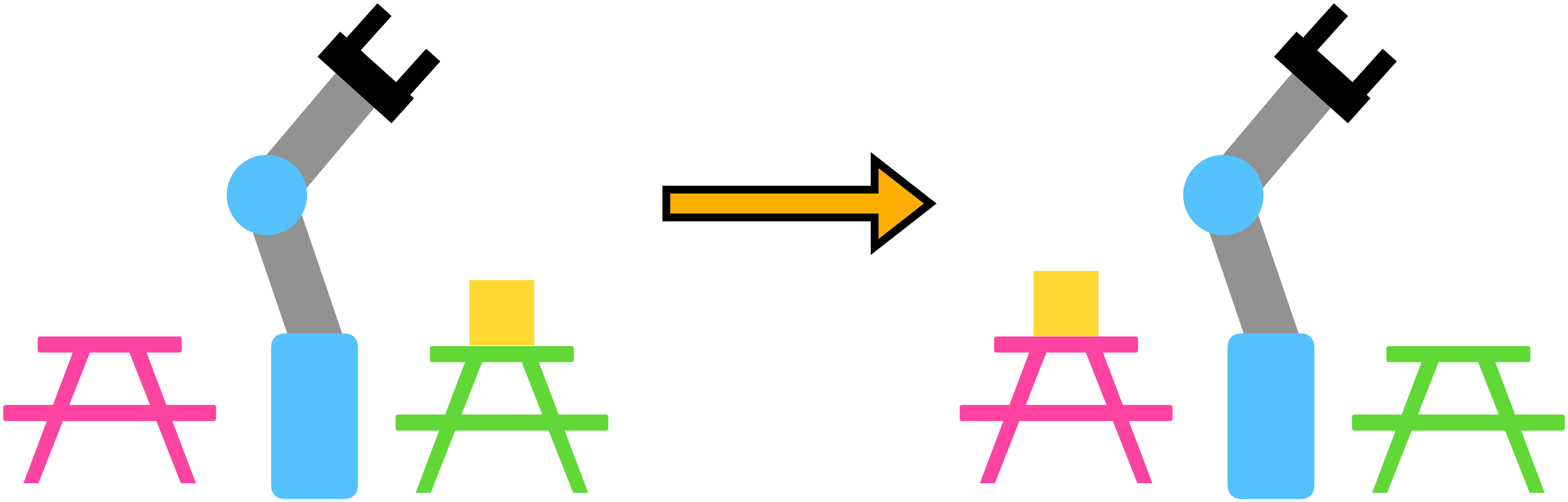
Click for citation,
if you use OMPL in your work

https://ompl.kavrakilab.org/

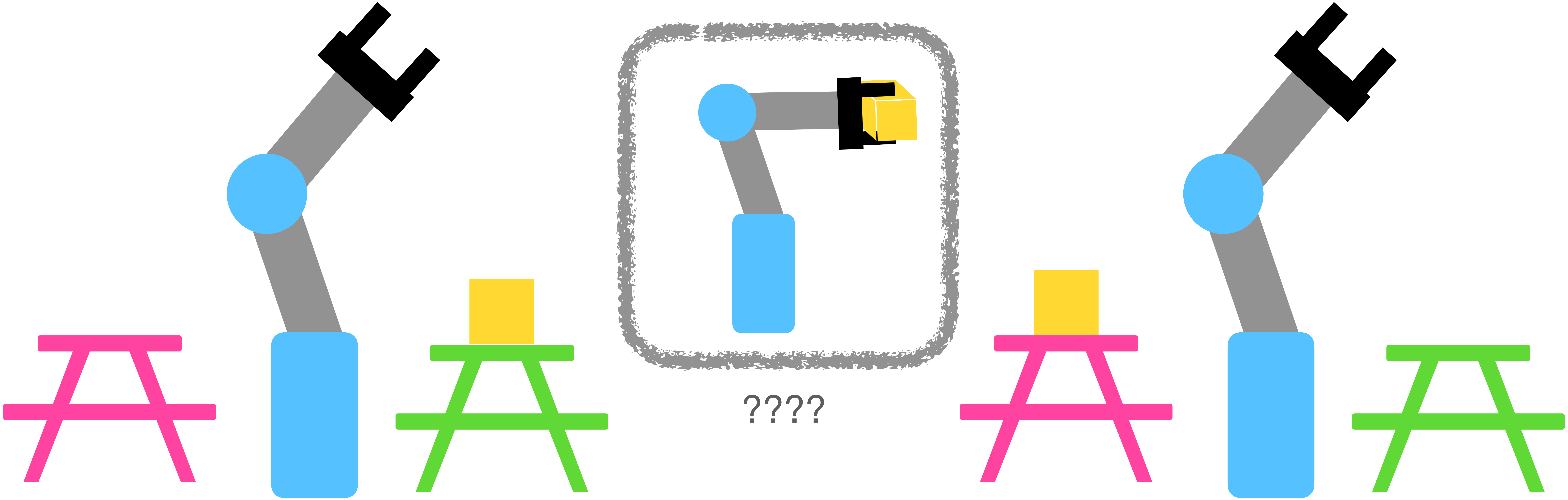
Demo

https://ompl.kavrakilab.org/

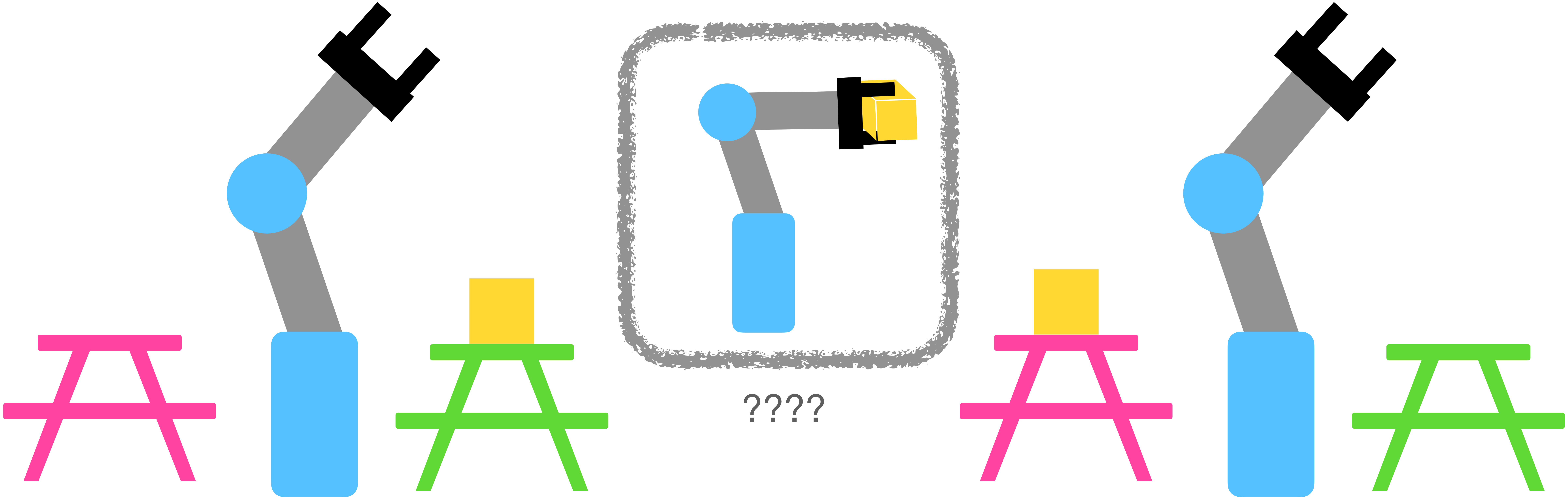
Is motion planning enough to solve all our problems?



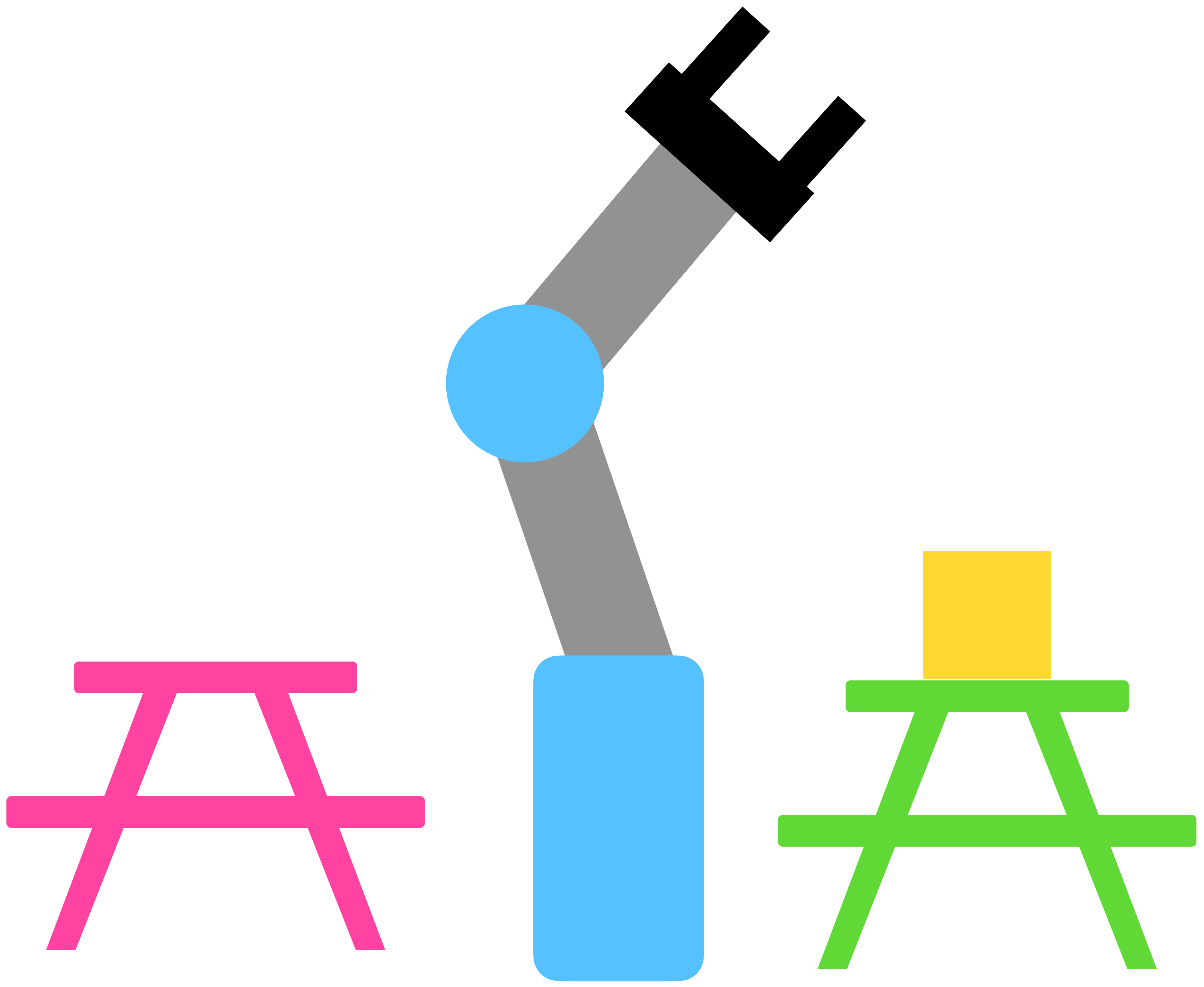
Is motion planning enough to solve all our problems?



Think about C-Space during the entire task execution.



Think about C-Space during the entire task execution.



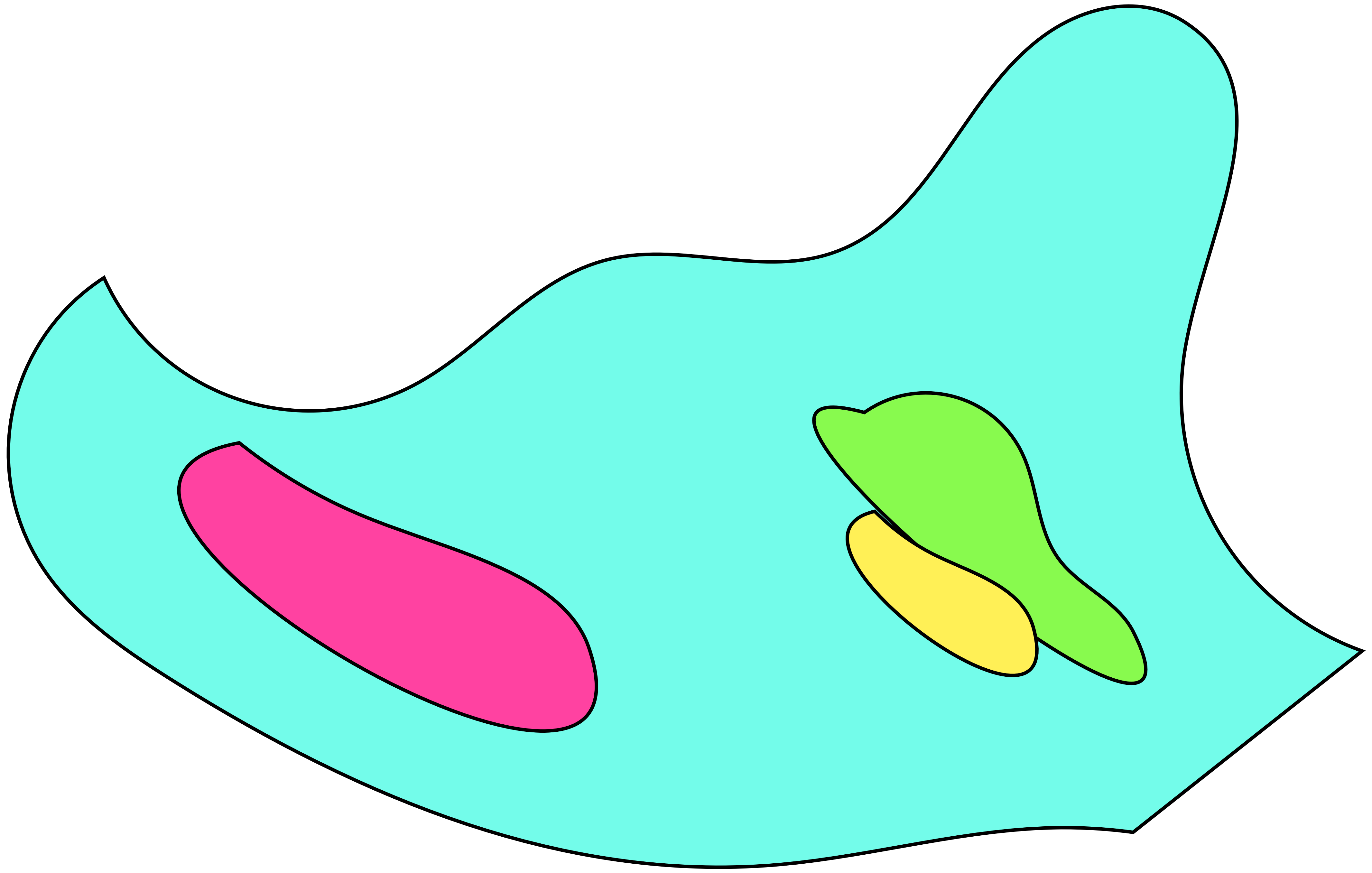


Illustration of a Complex C-space

Think about C-Space during the entire task execution.

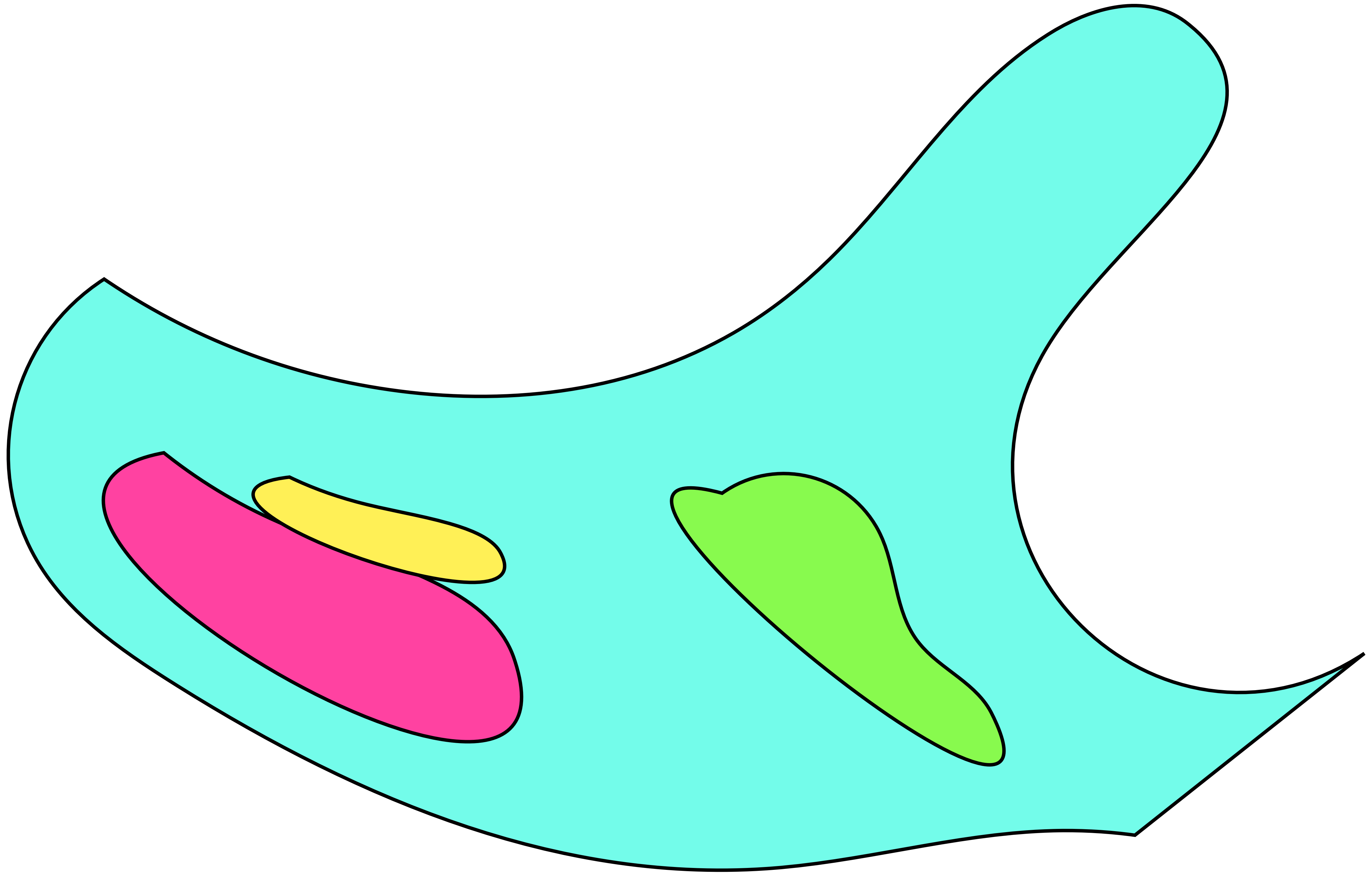
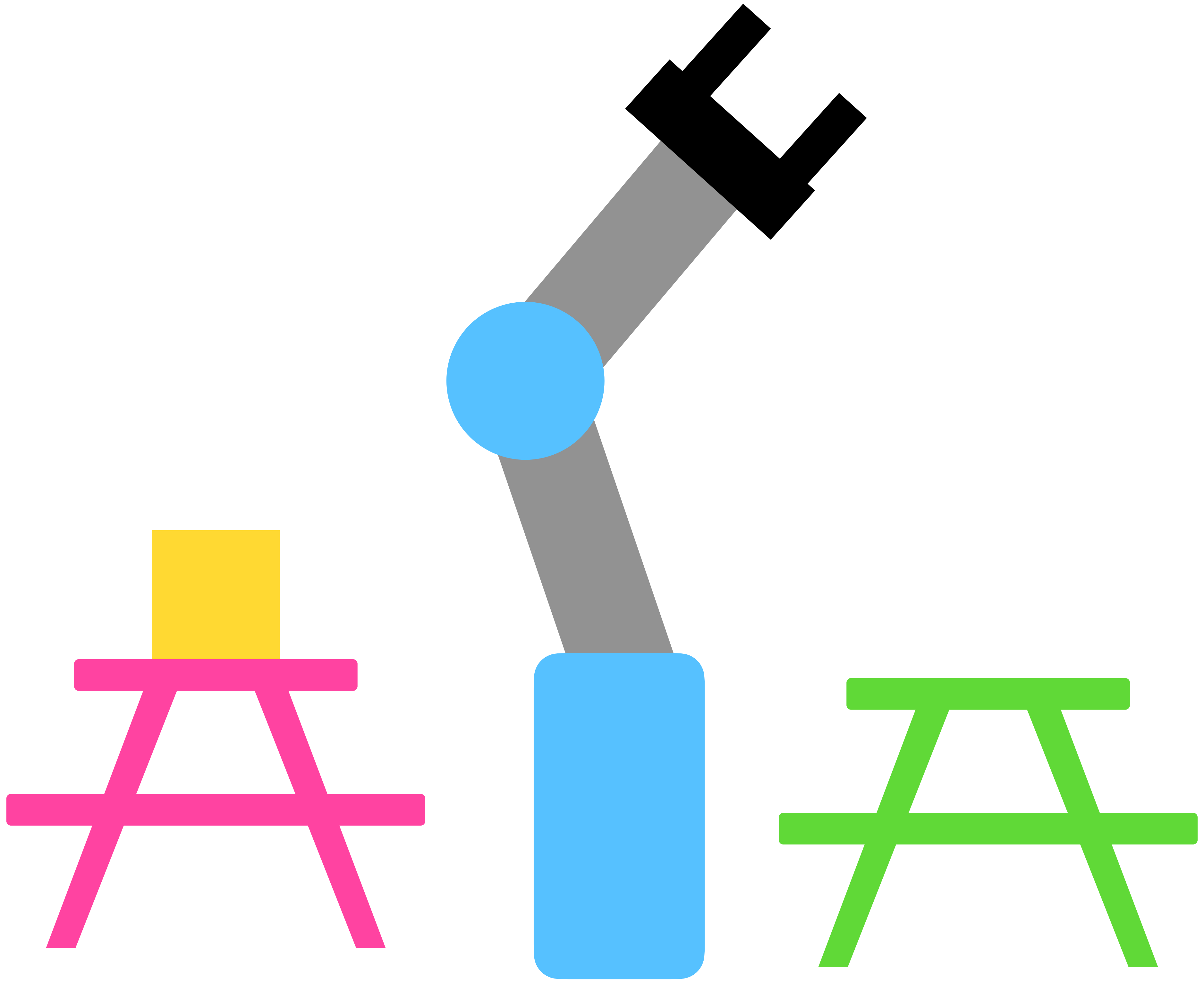
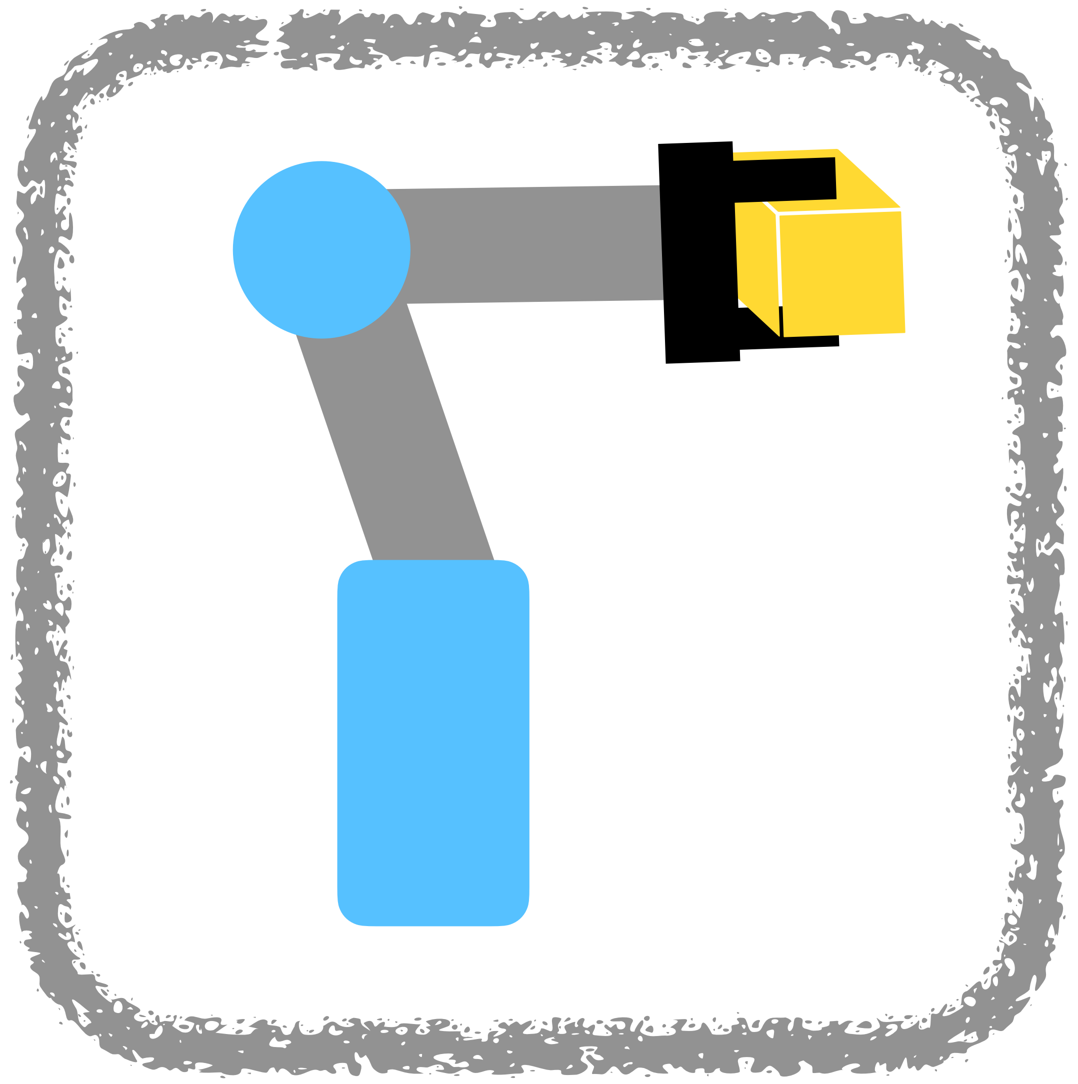


Illustration of a Complex C-space



Think about C-Space during the entire task execution.



????

What would be a C-space
after robot grasped the object?

How do we switch between these C-spaces?

Think about C-Space during the entire task execution.

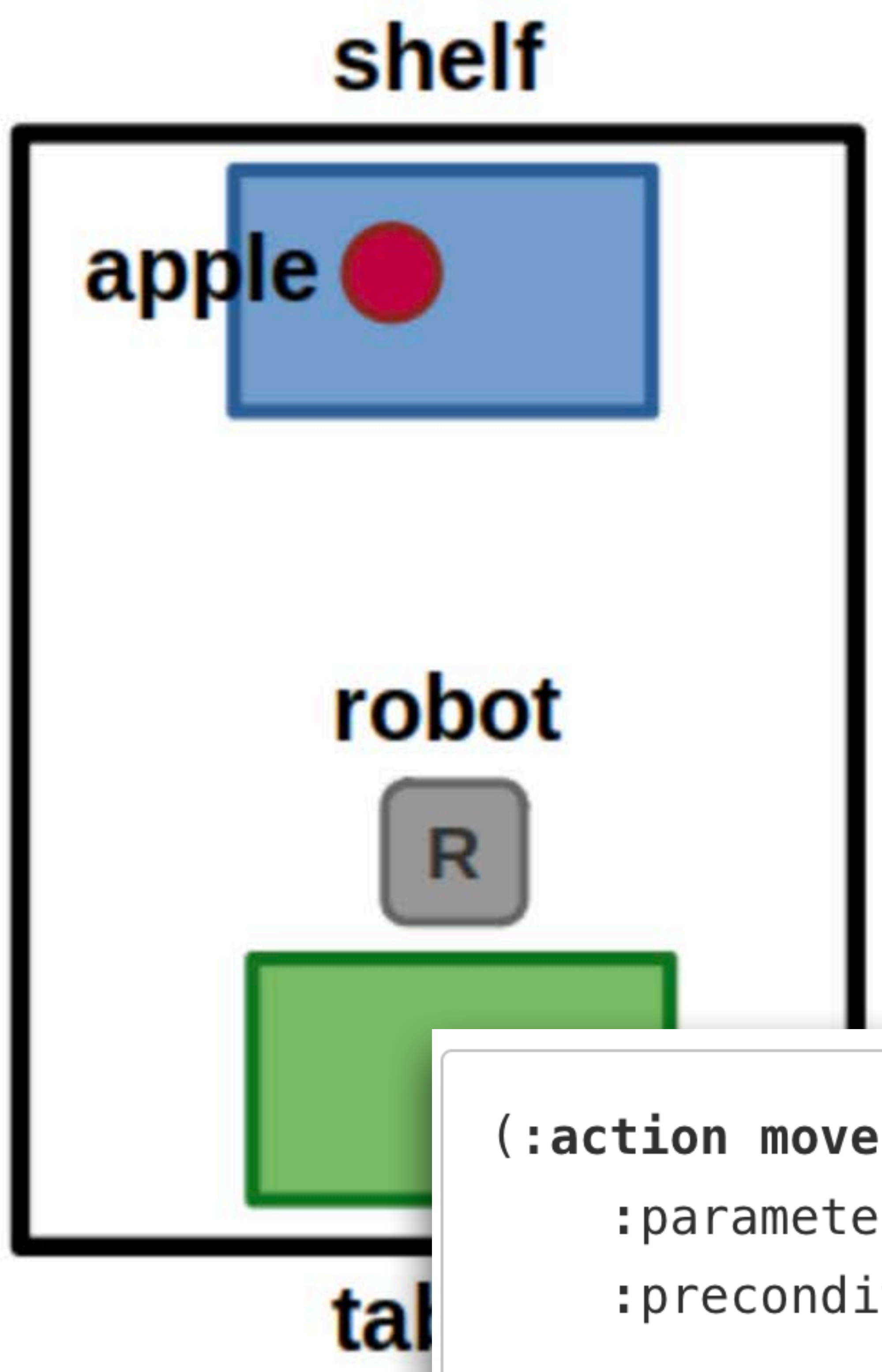
Robot 'PR2' flips a pancake in a laboratory kitchen of the Institute for Artificial Intelligence (IAI) at the Institute of Informatics and Automation (TZI) of Bremen University in Bremen, Germany. The robot can experimentally do household tasks and is part of a European project.

How do we switch between these C-spaces?

Considering we have so many *high-level* actions that robot can perform, how to plan to choose the sequence of *high-level* actions?

Task Planning

Task Planning



Simple task planning e
move between a finite
pick and place obje

Fetch the apple and put it on the table

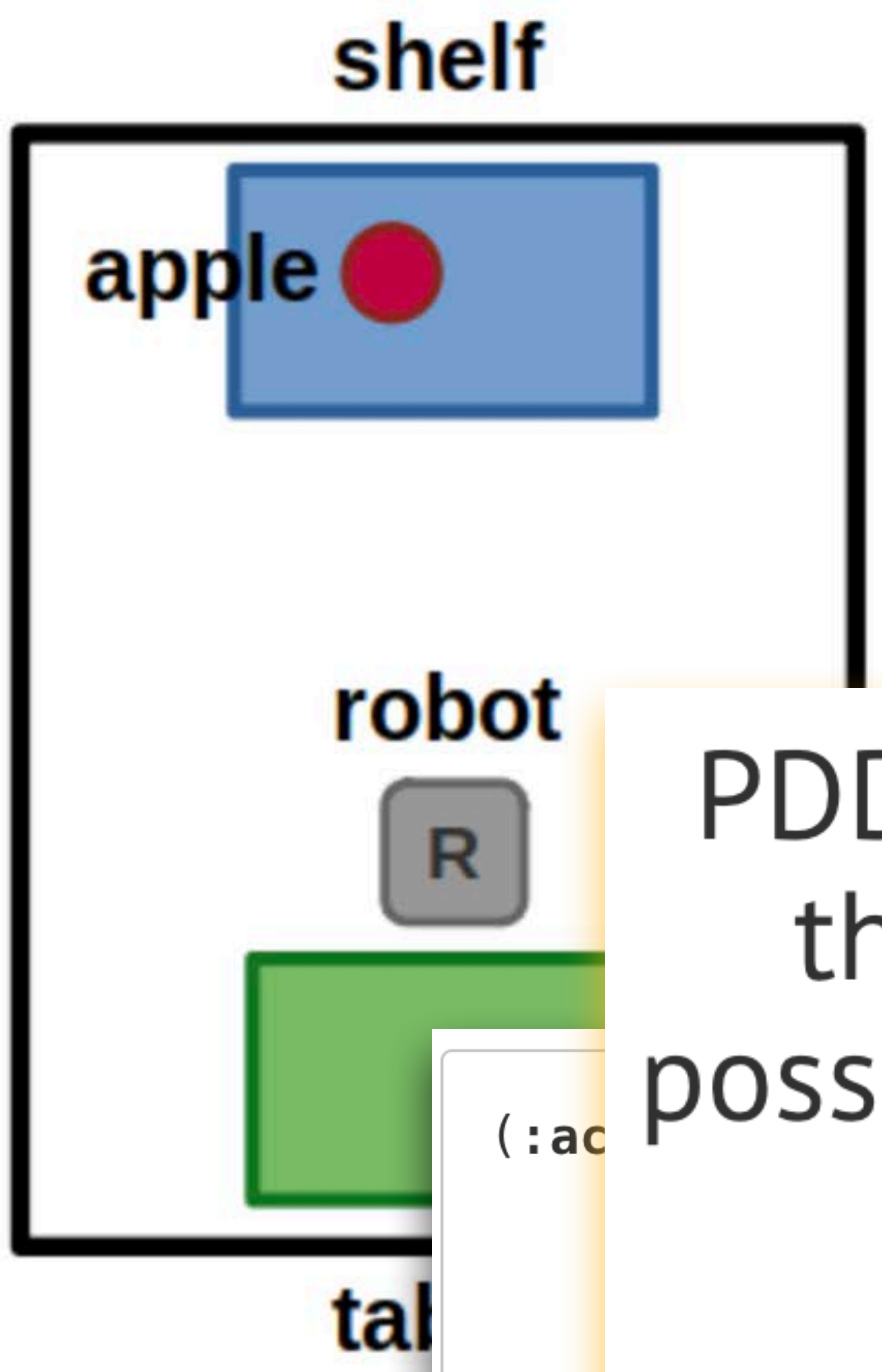
1. **Move** to the *shelf*
2. **Pick** up the *apple*
3. **Move** back to the *table*
4. **Place** the *apple*

- **Domain: The task-agnostic part**
  - **Predicates:** `(Robot ?r)`, `(Object ?o)`, `(Location ?loc)`, `(At ?r ?loc)`, `(Holding ?r ?o)`, etc.
  - **Actions:** `move(?r ?loc1 ?loc2)`, `pick(?r ?o ?loc)`, `place(?r ?o ?loc)`
- **Problem: The task-specific part**
  - **Objects:** `(Robot robot)`, `(Location shelf)`, `(Location table)`, `(Object apple)`
  - **Initial state:** `(HandEmpty robot)`, `(At robot table)`, `(At apple shelf)`
  - **Goal specification:** `(At apple table)`

```
(:action move
    :parameters (?r ?loc1 ?loc2)
    :precondition (and (Robot ?r)
                       (Location ?loc1)
                       (Location ?loc2)
                       (At ?r ?loc1))
    :effect (and (At ?r ?loc2)
                 (not (At ?r ?loc1)))
)
```

`on ?loc)` is a *unary* predicate, meaning it has one parameter. In our
`shelf` and `table` are specific location instances, so we say that
`on table)` and `(Location shelf)` are part of our initial state and
nge over time.
`?loc)` is a *binary* predicate, meaning it has two parameters. In our
he robot may begin at the table, so we say that `(At robot table)` is
r initial state, though it may be negated as the robot moves to another

https://roboticseabass.com/2022/07/19/task-planning-in-robotics/



Simple task planning e
move between a finite
pick and place obje

Fetch the apple and put it on the table

1. **Move** to the *shelf*
2. **Pick** up the *apple*

- **Domain: The task-agnostic part**
  - **Predicates:** `(Robot ?r)`, `(Object ?o)`, `(Location ?loc)`, `(At ?r ?loc)`, `(Holding ?r ?o)`, etc.
  - **Actions:** `move(?r ?loc1 ?loc2)`, `pick(?r ?o ?loc)`, `place(?r ?o ?loc)`
- **Problem: The task-specific part**
  - **Objects:** `(Robot robot)`, `(Location shelf)`, `(Location table)`, `(Object apple)`
  - **Initial state:** `(HandEmpty robot)`, `(At robot table)`, `(At apple`

PDDL is intended to express the "physics" of a domain, that is, what predicates there are, what actions are possible, what the structure of compound actions is, and what the effects of actions are.

GHALLAB ET AL. (1998), PDDL – THE PLANNING DOMAIN DEFINITION LANGUAGE

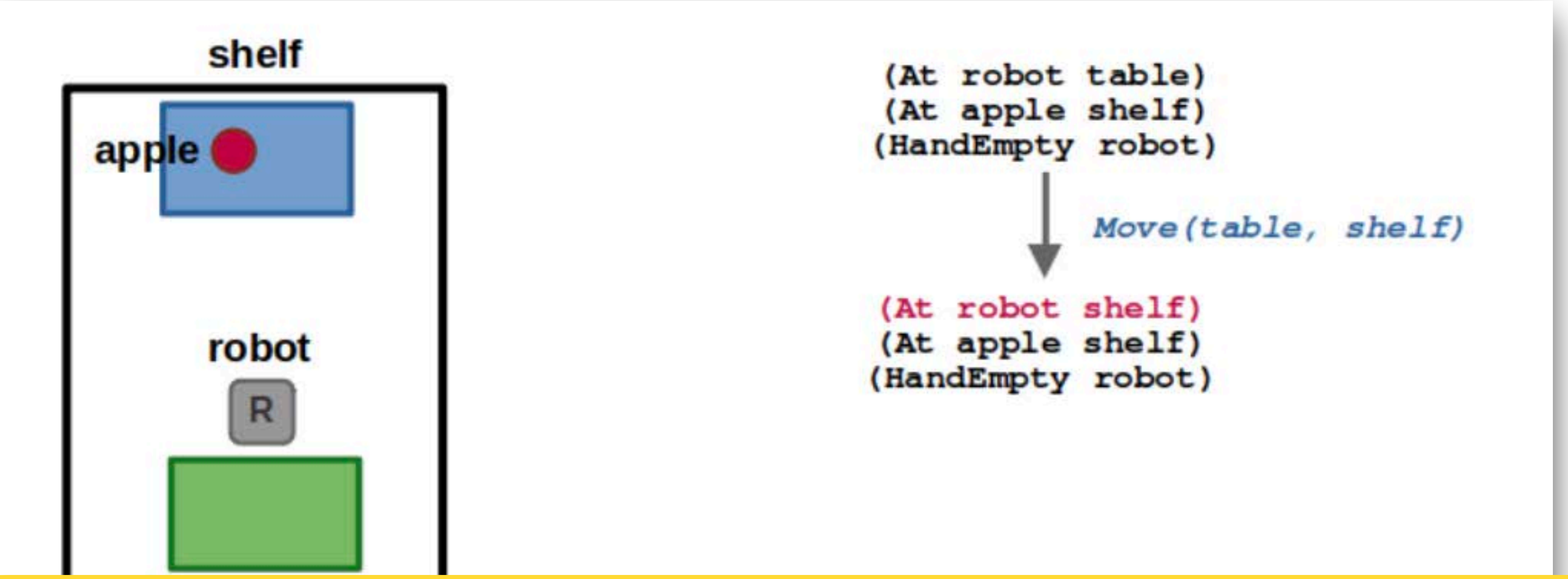
```
(:ac

  :effect (and (At ?r ?loc2)
               (not (At ?r ?loc1)))
)
```

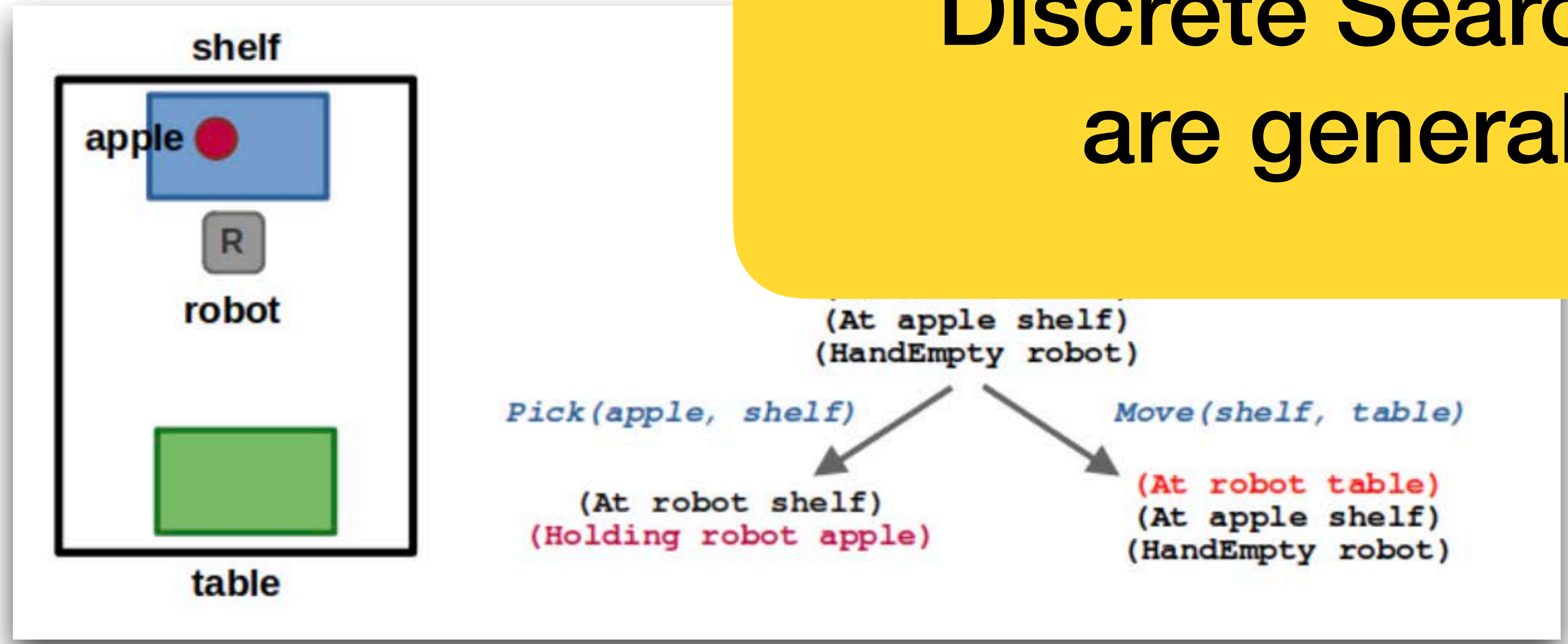
parameter. In our
we say that
our initial state and

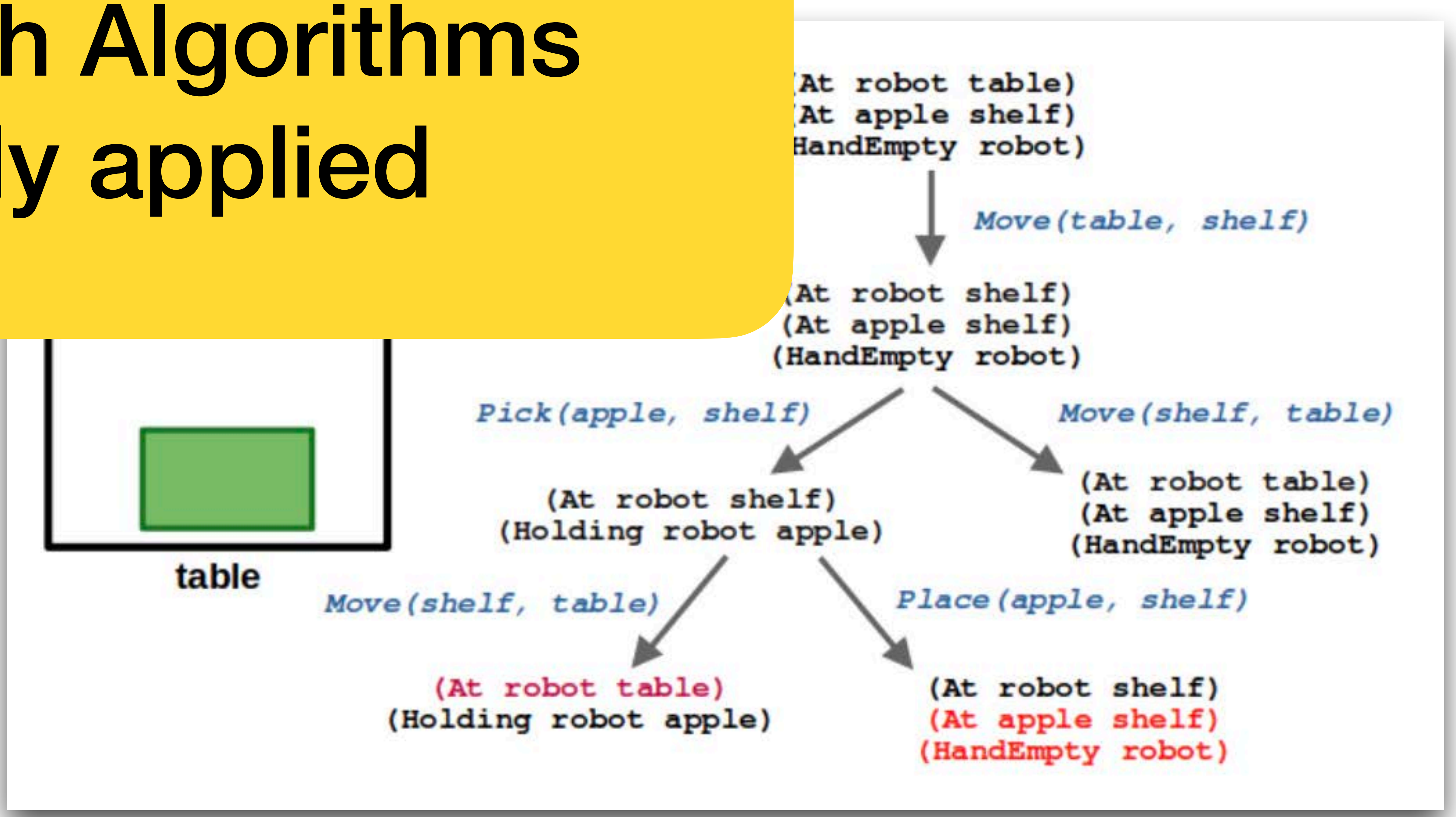
ameters. In our
he robot may begin at the table, so we say that `(At robot table)` is
r initial state, though it may be negated as the robot moves to another

https://roboticseabass.com/2022/07/19/task-planning-in-robotics/



Discrete Search Algorithms are generally applied





https://roboticseabass.com/2022/07/19/task-planning-in-robotics/

STRIPS planner

WIKIPEDIA
The Free Encyclopedia

Search Wikipedia | Search

Cre

Stanford Research Institute Problem Solver

10 languages

Article Talk Read Edit View history Tools

Contents [hide]

From Wikipedia, the free encyclopedia

The **Stanford Research Institute Problem Solver**, known by its acronym **STRIPS**, is an automated planner developed by Richard Fikes and Nils Nilsson in 1971 at SRI International.[1] The same name was later used to refer to the formal language of the inputs to this planner. This language is the base for most of the languages for expressing automated planning problem instances in use today; such languages are commonly known as action languages. This article only describes the language, not the planner.

Definition [edit]

A STRIPS instance is composed of:

- An initial state;
- The specification of the goal states – situations that the planner is trying to reach;
- A set of actions. For each action, the following are included:
  - preconditions (what must be established before the action is performed);
  - postconditions (what is established after the action is performed).

Mathematically, a STRIPS instance is a quadruple $\langle P, O, I, G \rangle$, in which each component has the following meaning:

1. P is a set of *conditions* (i.e., propositional variables);
2. O is a set of *operators* (i.e., actions); each operator is itself a quadruple $\langle \alpha, \beta, \gamma, \delta \rangle$, each element being a set of conditions. These four sets specify, in order, which conditions must be true for the action to be executable, which ones must be false, which ones are made true by the action and which ones are made false;
3. I is the initial state, given as the set of conditions that are initially true (all others are assumed false);
4. G is the specification of the goal state; this is given as a pair $\langle N, M \rangle$, which specify which conditions are true and false, respectively, in order for a state to be considered a goal state.

https://en.wikipedia.org/wiki/Stanford_Research_Institute_Problem_Solver

Task Planning with LLMs

Lets ask ChatGPT

I have the following objects on my kitchen counter.

1. Coffee mug.
2. Water Kettle.
3. Water dispenser.
4. Instant Coffee powder in a container.
5. Table spoon.

Additionally, I have the following in the fridge.

1. oat milk,
2. whole milk,
3. chocolate milk.

What steps should I take in order make a cup of coffee with chocolate milk.

ChatGPT response

To make a cup of coffee with chocolate milk, you can follow these steps:

Ingredients:

1. Coffee mug
2. Instant coffee powder
3. Chocolate milk (from the fridge)
4. Water (for the coffee)
5. Table spoon
6. Water kettle
7. Water dispenser (for room tem ratu wa r)

Instructions:

1. **Prepare the Coffee:**
   - Fill the water kettle with the desired amount of water for your coffee and turn it on to boil. You'll need this hot water to make coffee.
   - While waiting for the water to boil, place a table spoon of instant coffee powder into your coffee mug.
2. **Pour Hot Water:**
   - Once the water in the kettle has boiled, carefully pour the hot water into the coffee mug with the instant coffee powder. Use the desired amount of hot water to achieve your preferred coffee strength.
3. **Stir the Coffee:**
   - Use the table spoon to stir the hot water and instant coffee powder together until the coffee is well mixed and any lumps have dissolved.
4. **Add Chocolate Milk:**
   - Open your fridge and take the chocolate milk out.
   - Pour the desired amount of chocolate milk into the coffee mug with the coffee. The amount can vary depending on how strong you want your coffee to taste and how much chocolate flavor you prefer.
5. **Stir Again:**
   - Use t e ta le s o n o st e cho ol e m k nd offee t gether until they are well comb ned.
6. **Taste and Adjust:**
   - Taste the coffee with chocolate milk to ensure it's to your liking. If you prefer it sweeter, you can add more chocolate milk. If it's too strong or too sweet, you can add a bit of water to dilute it.
7. **Enjoy:**
   - Your cup of coffee with chocolate milk is now ready to enjoy. You can sip it while it's still warm.

Remember to be cautious when handling hot water, and be sure not to overheat your chocolate milk when adding it to the coffee. Adjust the ratios of coffee to chocolate milk to match your taste preferences.

What are we missing?

ChatGPT response

To make a cup of coffee with chocolate milk, you can follow these steps:

Ingredients:

1. Coffee mug
2. Instant coffee powder
3. Chocolate milk (from the fridge)
4. Water (for the coffee)
5. Table spoon
6. Water kettle
7. Water dispenser (for room temperature water)

Instructions:

1. **Prepare the Coffee:**
   - Fill the water kettle with the desired amount of water for your coffee and turn it on to boil. You'll need this hot water to make coffee.
   - While waiting for the water to boil, place a table spoon of instant coffee powder into your coffee mug.
2. **Pour Hot Water:**
   - Once the water in the kettle has boiled, carefully pour the hot water into the coffee mug with the instant coffee powder. Use the desired amount of hot water to achieve your preferred coffee strength.
3. **Stir the Coffee:**
   - Use the table spoon to stir the hot water and instant coffee powder together until the coffee is well mixed and any lumps have dissolved.
4. **Add Chocolate Milk:**
   - Open your fridge and take the chocolate milk out.
   - Pour the desired amount of chocolate milk into the coffee mug with the coffee. The amount can vary depending on how strong you want your coffee to taste and how much chocolate flavor you prefer.
5. **Stir Again:**
   - Use the table spoon to stir the chocolate milk and coffee together until they are well combined.
6. **Taste and Adjust:**
   - Taste the coffee with chocolate milk to ensure it's to your liking. If you prefer it sweeter, you can add more chocolate milk. If it's too strong or too sweet, you can add a bit of water to dilute it.
7. **Enjoy:**
   - Your cup of coffee with chocolate milk is now ready to enjoy. You can sip it while it's still warm.

Remember to be cautious when handling hot water, and be sure not to overheat your chocolate milk when adding it to the coffee. Adjust the ratios of coffee to chocolate milk to match your taste preferences.

How will a robot know what these are?

say-can.github.io

Do As I Can, Not As I Say:

Grounding Language in Robotic Affordances

Michael Ahn* Anthony Brohan* Noah Brown* Yevgen Chebotar* Omar Cortes* Byron David* Chelsea Finn*
Chuyuan Fu* Keerthana Gopalakrishnan* Karol Hausman* Alex Herzog* Daniel Ho* Jasmine Hsu* Julian Ibarz*
Brian Ichter* Alex Irpan* Eric Jang* Rosario Jauregui Ruano* Kyle Jeffrey* Sally Jesmonth* Nikhil Joshi*
Ryan Julian* Dmitry Kalashnikov* Yuheng Kuang* Kuang-Huei Lee* Sergey Levine* Yao Lu* Linda Luu* Carolina Parada*
Peter Pastor* Jornell Quiambao* Kanishka Rao* Jarek Rettinghouse* Diego Reyes* Pierre Sermanet* Nicolas Sievers*
Clayton Tan* Alexander Toshev* Vincent Vanhoucke* Fei Xia* Ted Xiao* Peng Xu* Sichun Xu* Mengyuan Yan* Andy Zeng*

Robotics at Google Everyday Robots

* Authors listed in alphabetical order (see paper appendix for contribution statement).

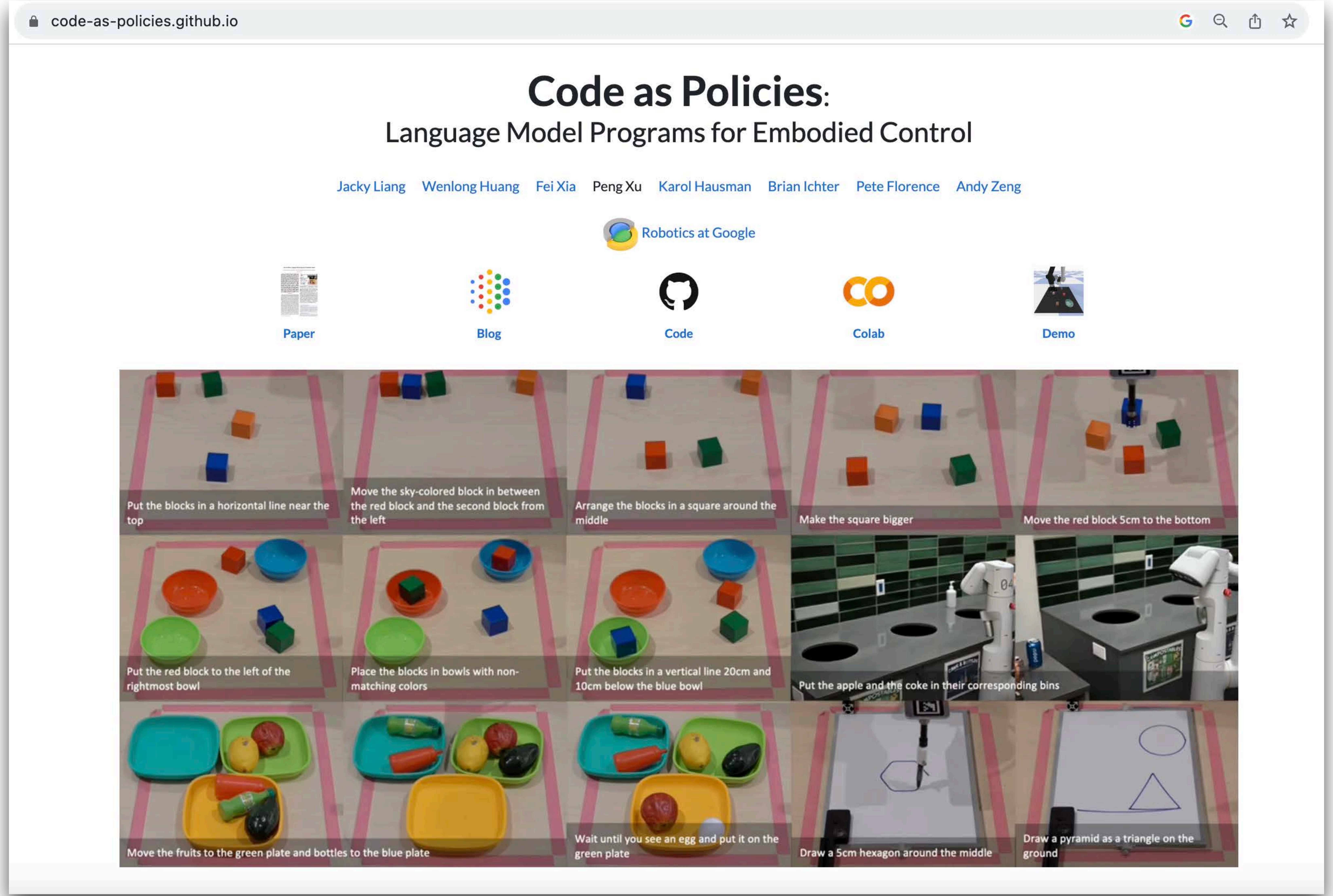
Paper Video Blogpost Code Demo

What's New

- [8/16/2022] We integrated SayCan with Pathways Language Model (PaLM), and updated the results. We also added new capabilities including drawer manipulation, chain of thought prompting and multilingual instructions. You can see all the new results in the updated paper.
- [8/16/2022] Our updated results show that SayCan combined with the improved language model (PaLM), which we refer to as PaLM-SayCan, improves the **robotics performance** of the entire system compared to a previous LLM (FLAN). PaLM-SayCan chooses the correct sequence of skills 84% of the time and executes them successfully 74% of the time, reducing errors by a half compared to FLAN. This is particularly exciting because it represents the first time we can see how an improvement in language models translates to a similar improvement in robotics.
- [8/16/2022] We open-sourced a version of SayCan on a simulated tabletop environment.
- [4/4/2022] Initial release of SayCan.

https://say-can.github.io/

4x speed
Language
× Affordance
Combined Score
put down the water bottle
1.00
done
0.02
find an apple
0.01
find a water bottle
0.00
find an orange can
0.00
User input: I just worked out, can you bring me a drink and a snack to recover?
Robot: I would 1. find a water bottle, 2. pick up the water bottle 3. bring it to you, 4.___



https://code-as-policies.github.io/

Code as Policies: Language Model Programs for Embodied Control

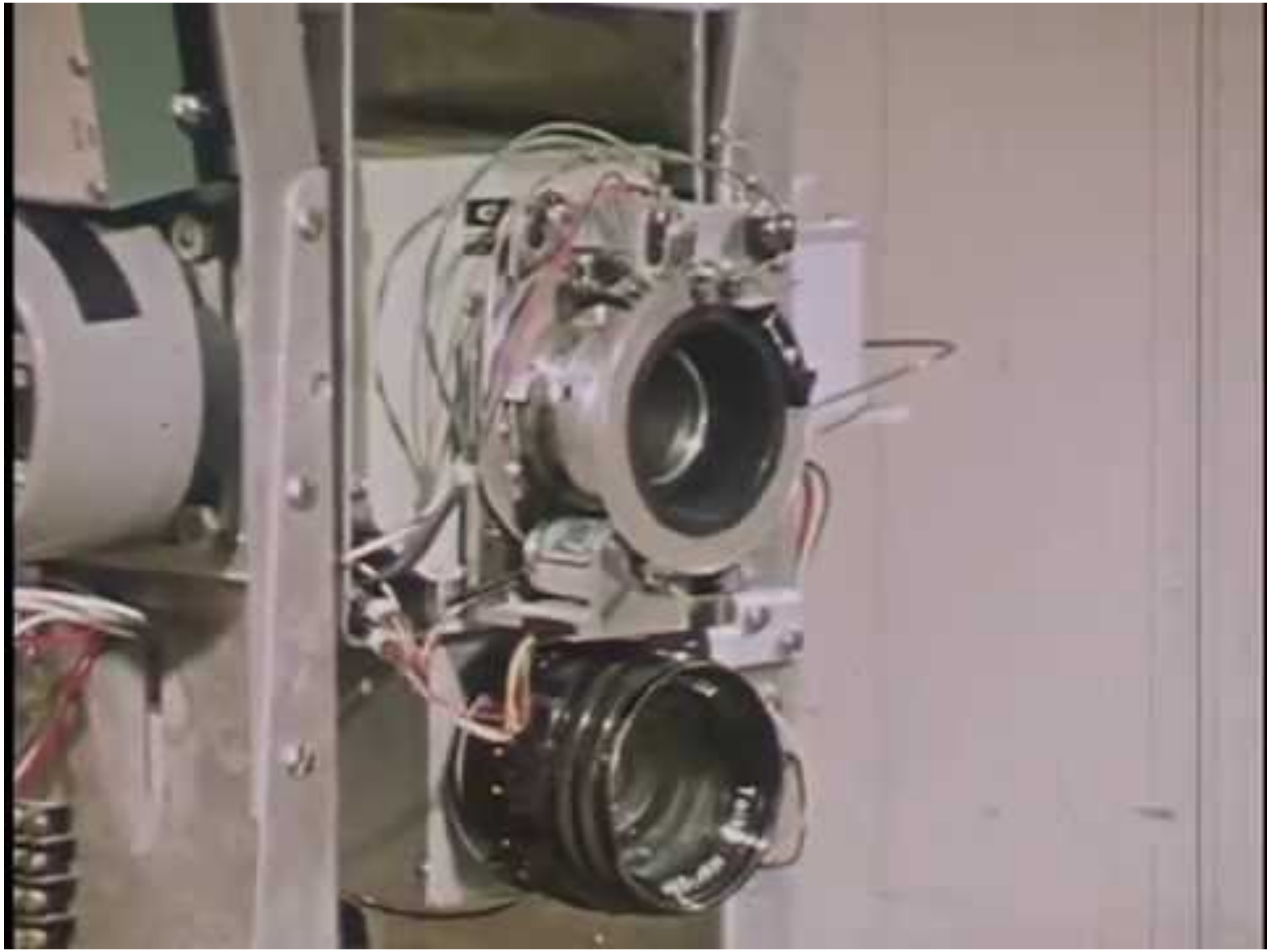
Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, Andy Zeng

This video is voice-narrated

code-as-policies.github.io

Robotics at Google

Task and Motion Planning



From 1966 through 1972, the Artificial Intelligence Center at SRI International (then Stanford Research Institute) conducted research on a mobile robot system nicknamed “Shakey.” Endowed with a limited ability to perceive and model its environment, Shakey could perform tasks that required planning, route-finding, and the rearranging of simple objects.

https://www.youtube.com/watch?v=GmU7SimFkpU

Task and Motion Planning

Integrated Task and Motion Planning

Caelan Reed Garrett [1], Rohan Chitnis [1], Rachel Holladay [1], Beomjoon Kim [1], Tom Silver [1], Leslie Pack Kaelbling [1] and Tomás Lozano-Pérez [1]

[1]CSAIL, MIT, Cambridge, USA, 02139; email: caelan@csail.mit.edu

Annu. Rev. Control Robot. Auton. Syst. 2021. 2021. 4:1–30

Keywords

task and motion planning, robotics, automated planning, motion planning, manipulation planning

Abstract

The problem of planning for a robot that operates in environments containing a large number of objects, taking actions to move itself through the world as well as to change the state of the objects, is known as *task and motion planning* (TAMP). TAMP problems contain elements of discrete task planning, discrete-continuous mathematical programming, and continuous motion planning, and thus cannot be effectively addressed by any of these fields directly. In this paper, we define a class of TAMP problems and survey algorithms for solving them, characterizing the solution methods in terms of their strategies for solving the continuous-space subproblems and their techniques for integrating the discrete and continuous components of the search.

	Pre-discretized	Sampling	Optimization
Satisfaction First	Ferrer-Mestres* (84, 85)	Siméon† (22) Hauser† (13, 29, 14) Garrett* (86, 21) Krontiris† (87, 88) Akbari* (89) Vega-Brown† (90)	
Interleaved	Dornhege* (62, 63, 91) Gaschler* (92, 93, 94) Colledanchise* (95)	Gravot* (96, 97) Stilman† (23, 98, 99) Plaku† (100) Kaelbling* (101, 102) Barry† (103, 30, 104) Garrett* (70, 71) Thomason* (105) Kim* (106, 107) Kingston† (108)	Fernandez-Gonzalez* (109)
Sequence First	Nilsson* (2) Erdem* (74, 75) Lagriffoul* (65, 66, 67) Pandey* (110, 111) Lozano-Pérez* (112) Dantam* (77, 78, 79) Lo* (113)	Wolfe* (114) Srivastava* (76, 60) Garrett* (55, 73)	Toussaint* (61, 68, 69) Shoukry* (81, 82, 83) Hadfield-Menell* (115)

Table 1: A table that categorizes MMMP and TAMP approaches, based on how they solve HC-SPS and how they integrate with constraint satisfaction with action sequencing. Approaches for MMMP are designated with †, and approaches for TAMP are designated with *. Each table cell is listed chronologically.

Garrett, Caelan Reed, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. "Integrated task and motion planning." *Annua review of control, robotics, and autonomous systems* 4 (2021): 265-293.

Motion Control


DIRECTORY


Michael Jackson - “Smooth Criminal” - https://youtu.be/h_D3VFfhvs4

https://youtu.be/7osiCOjdjBQ

https://youtu.be/7osiCOjdjBQ

https://youtu.be/KiGGTmuBujs

https://youtu.be/lXkYz0AfcAk

Biped Robotics Lab @ Michigan - https://youtu.be/-35xJfFMDkE

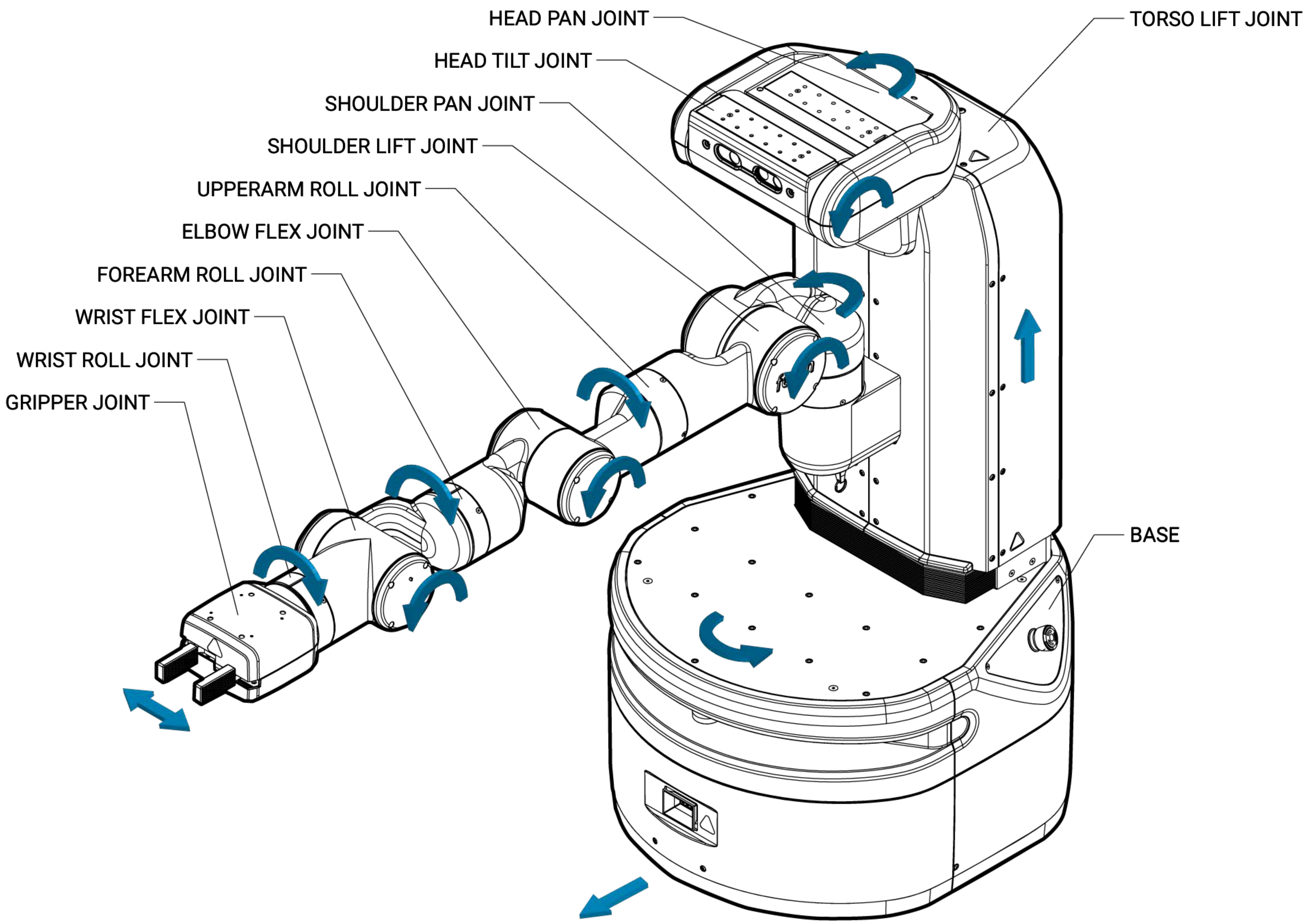
Biped Robotics Lab @ Michigan - https://youtu.be/0gauVSUJzd0

Control must be fast and often

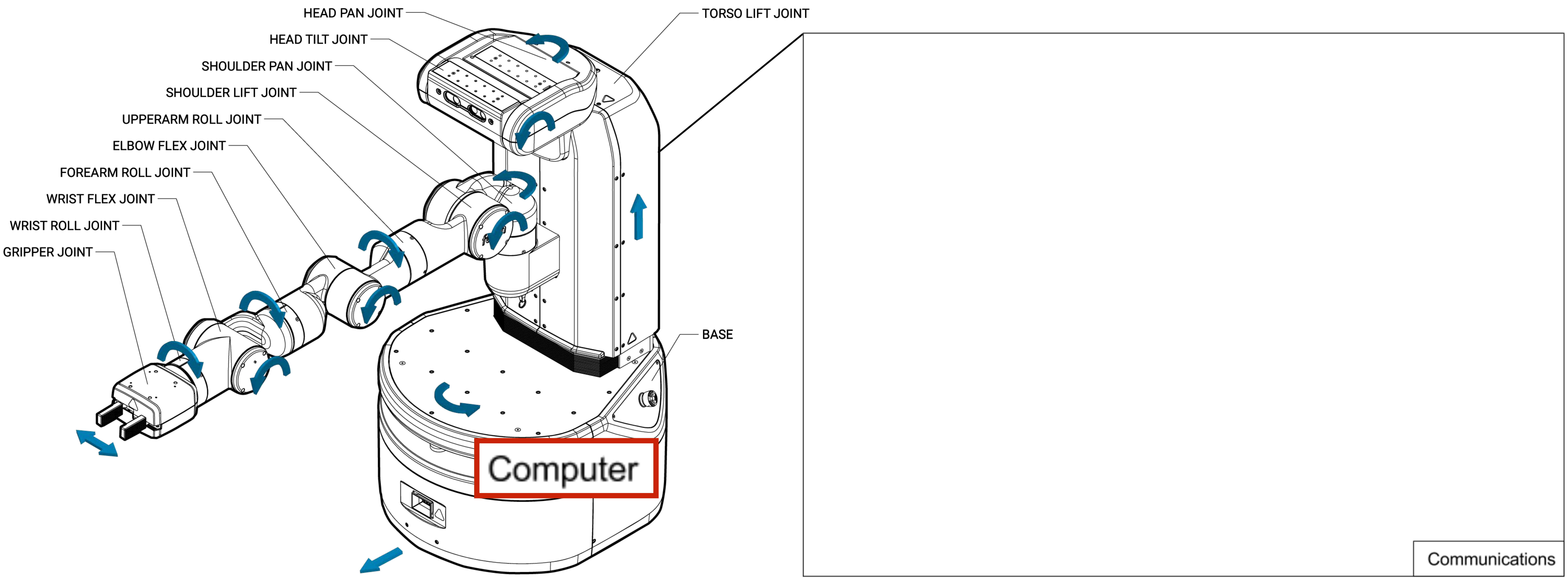
https://youtu.be/8o6FroUWkpE

Communication inside a robot is like its central nervous system

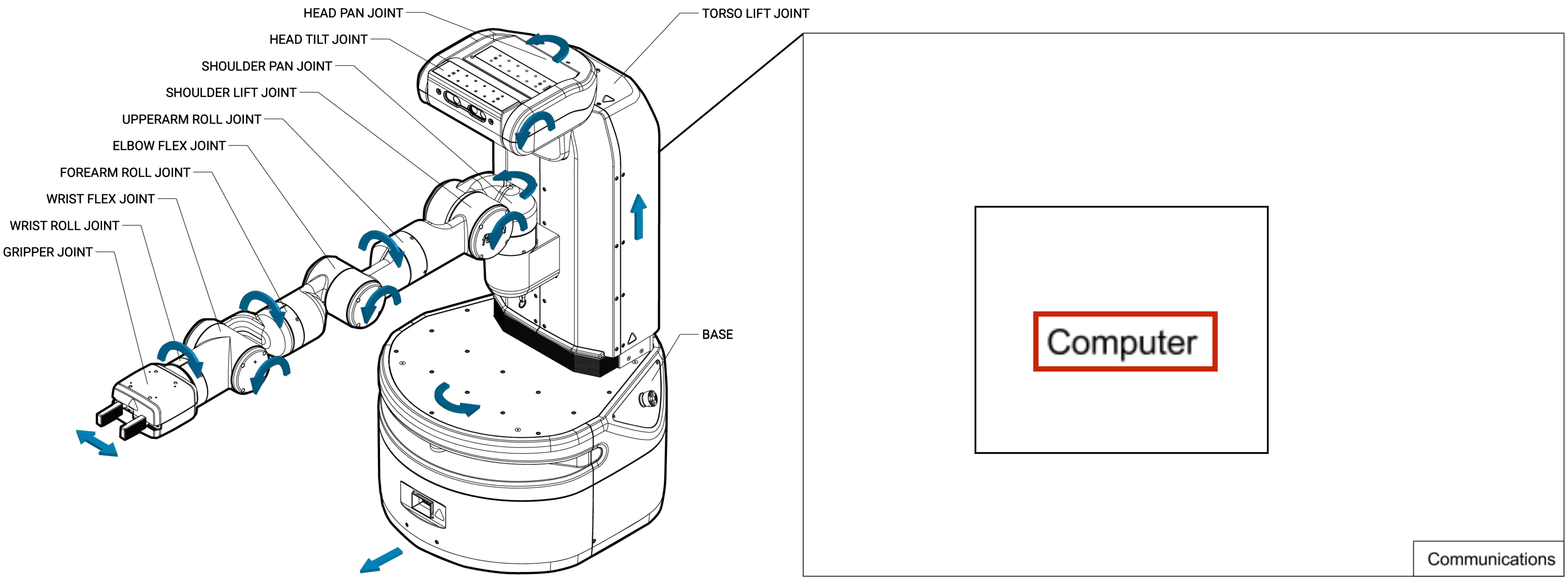
Communication inside a robot is like its central nervous system

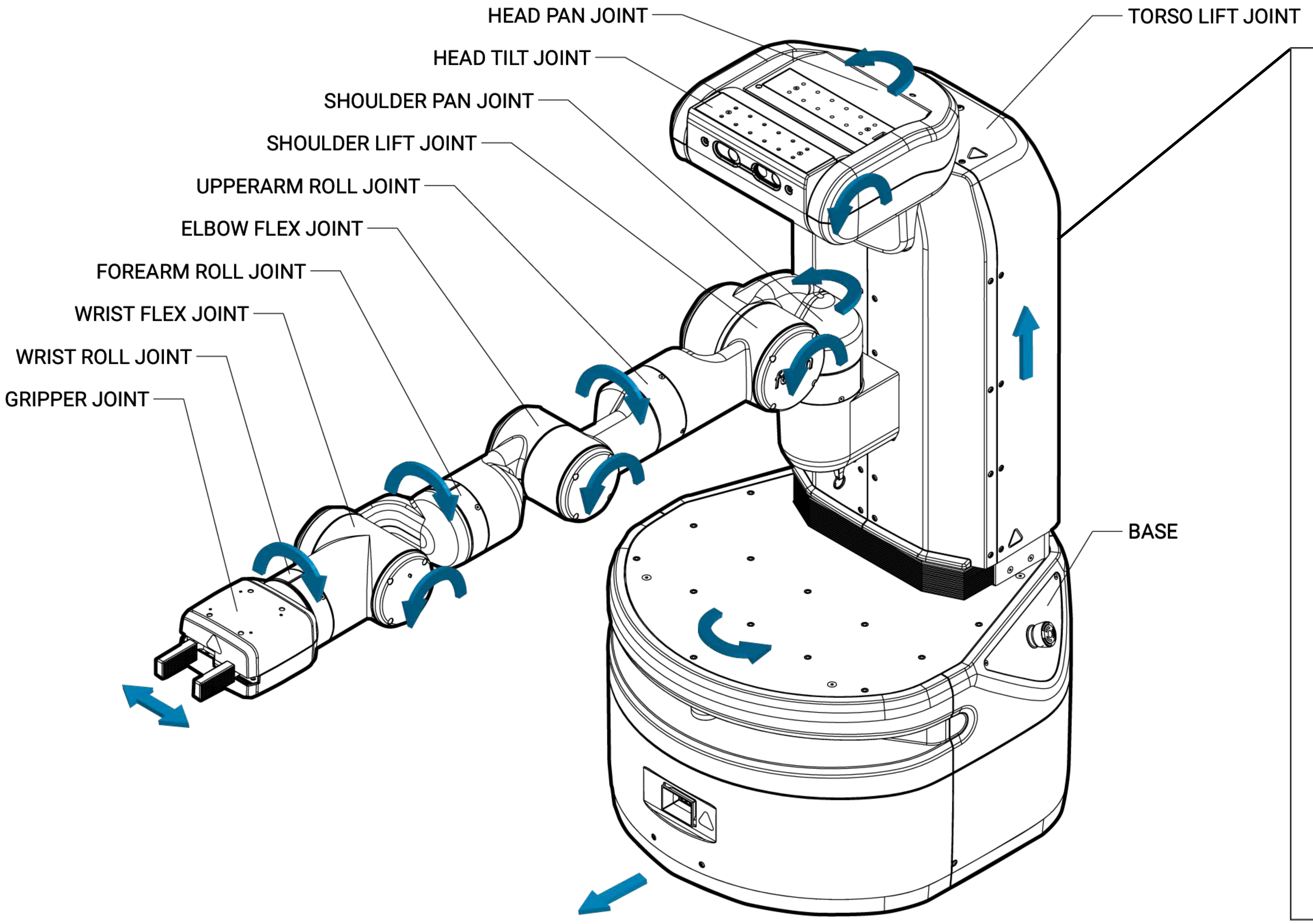


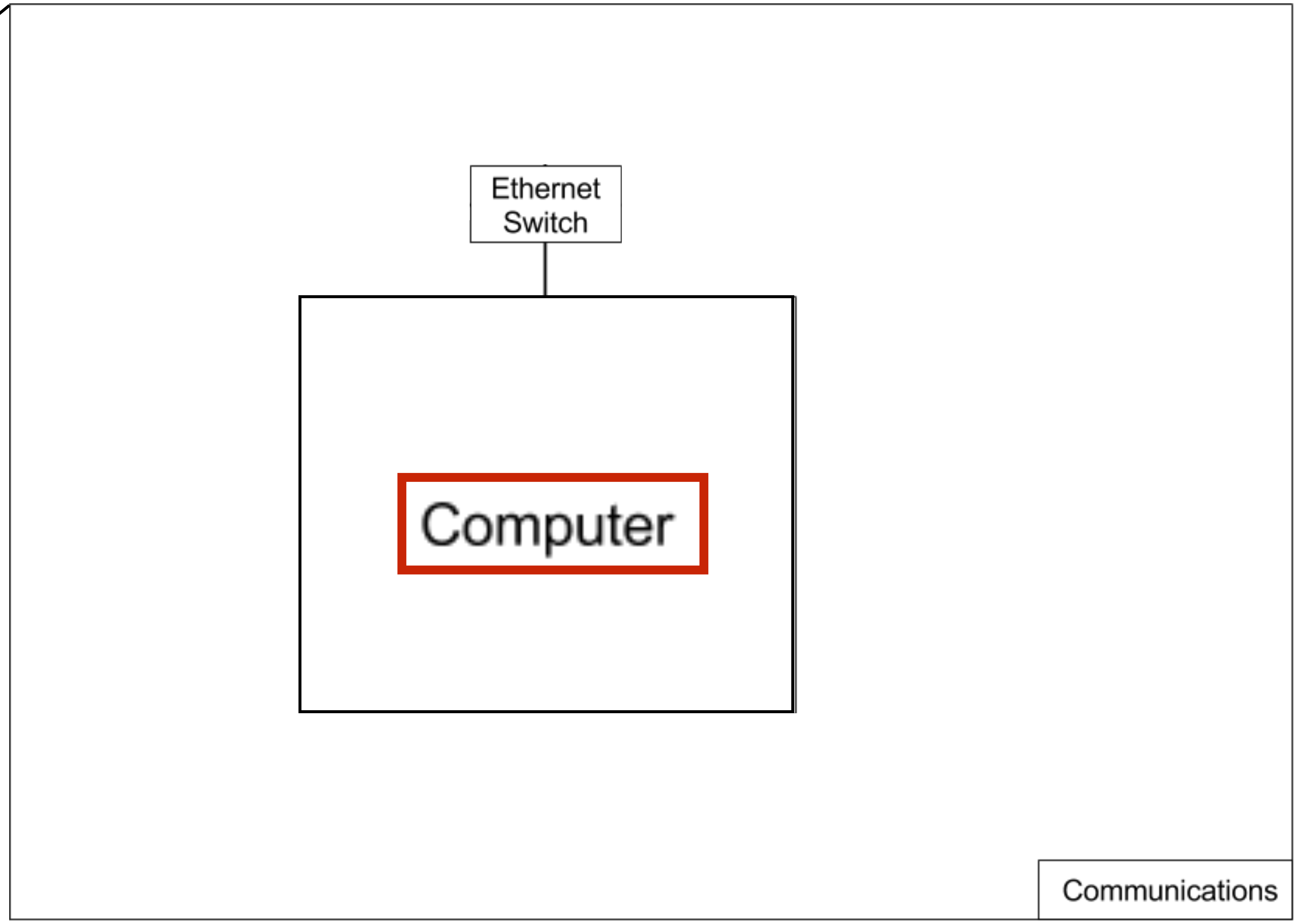
Slide borrowed from Michigan Robotics autorob.org



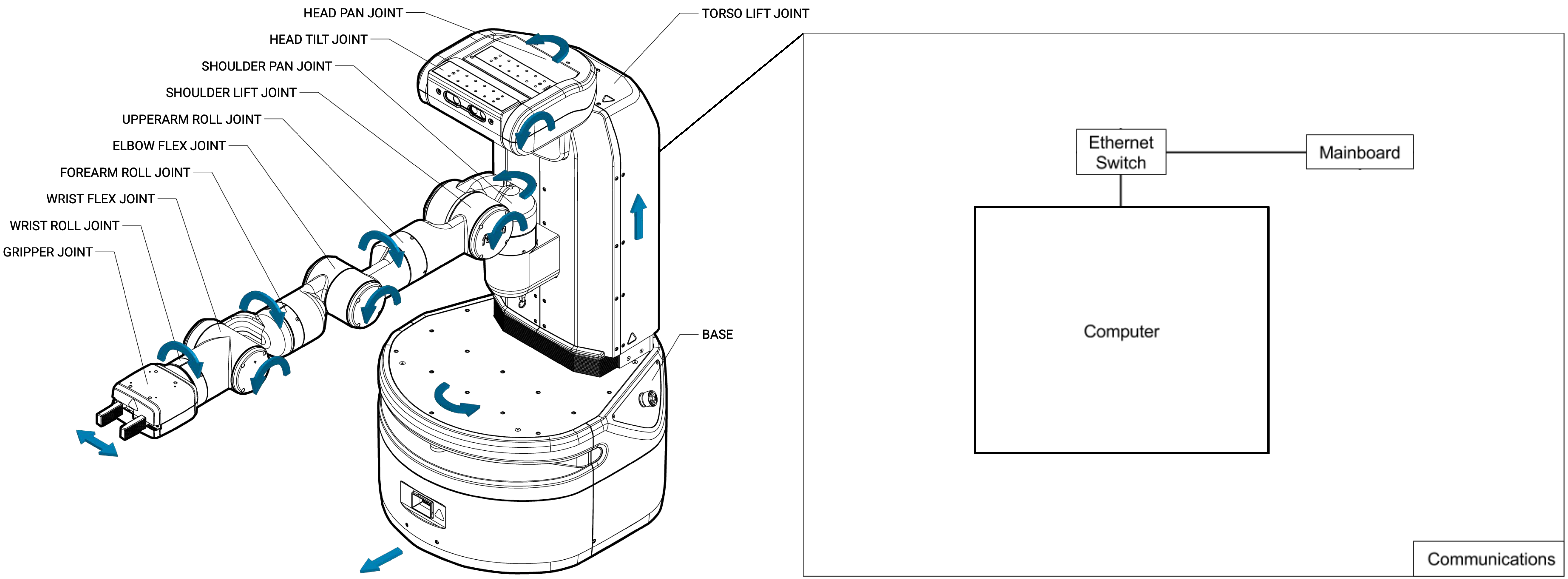
Slide borrowed from Michigan Robotics autorob.org


HEAD PAN JOINT
HEAD TILT JOINT
SHOULDER PAN JOINT
SHOULDER LIFT JOINT
UPPERARM ROLL JOINT
ELBOW FLEX JOINT
FOREARM ROLL JOINT
WRIST FLEX JOINT
WRIST ROLL JOINT
GRIPPER JOINT
TORSO LIFT JOINT
BASE
Computer
Communications

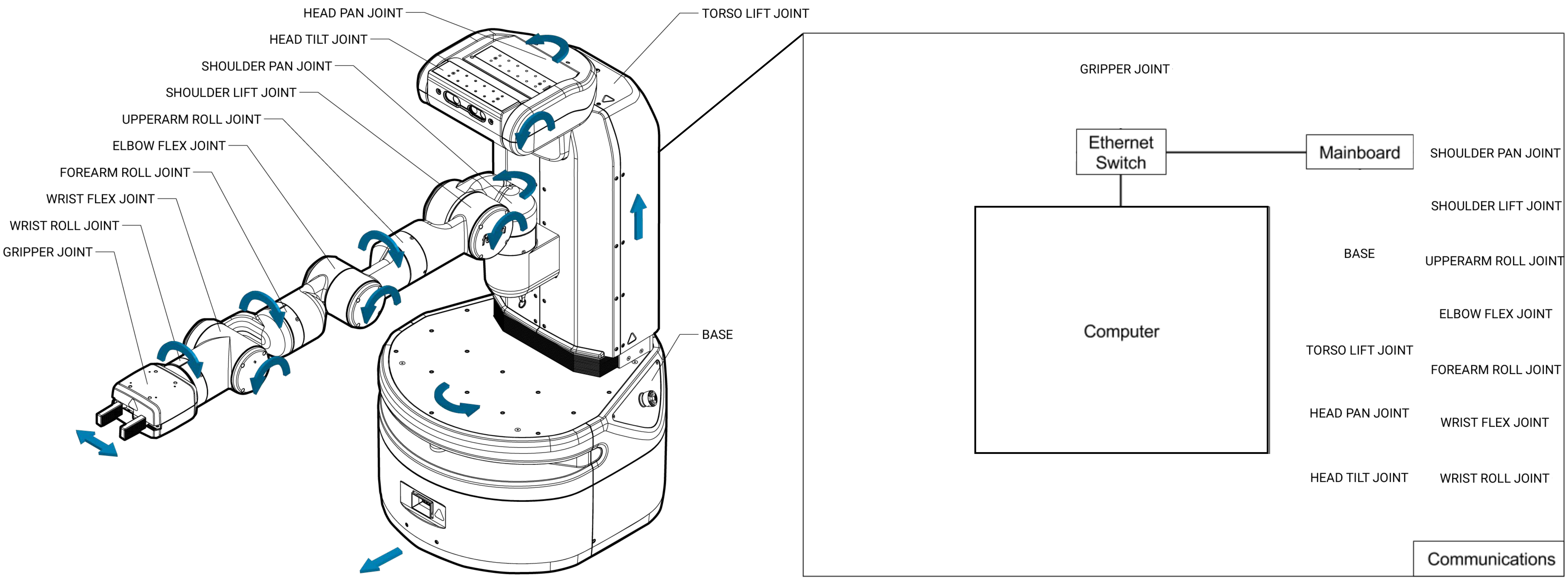




Slide borrowed from Michigan Robotics autorob.org

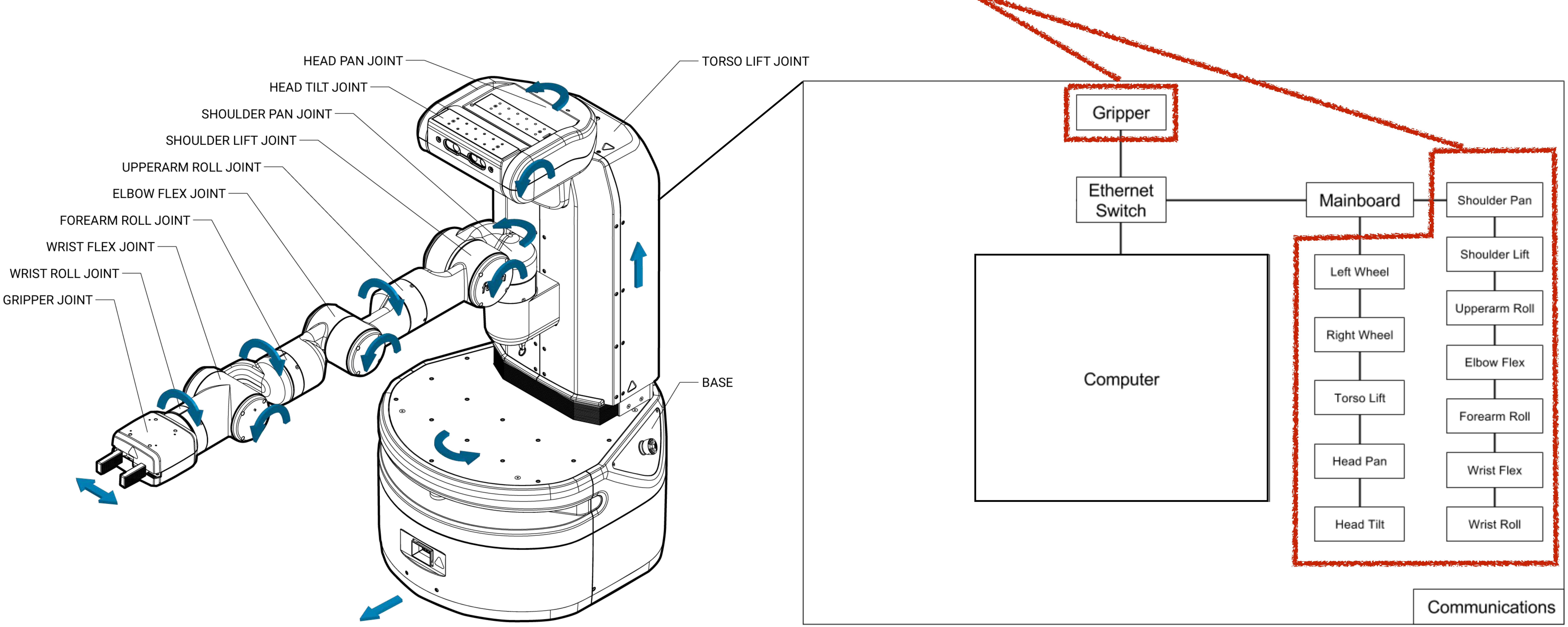


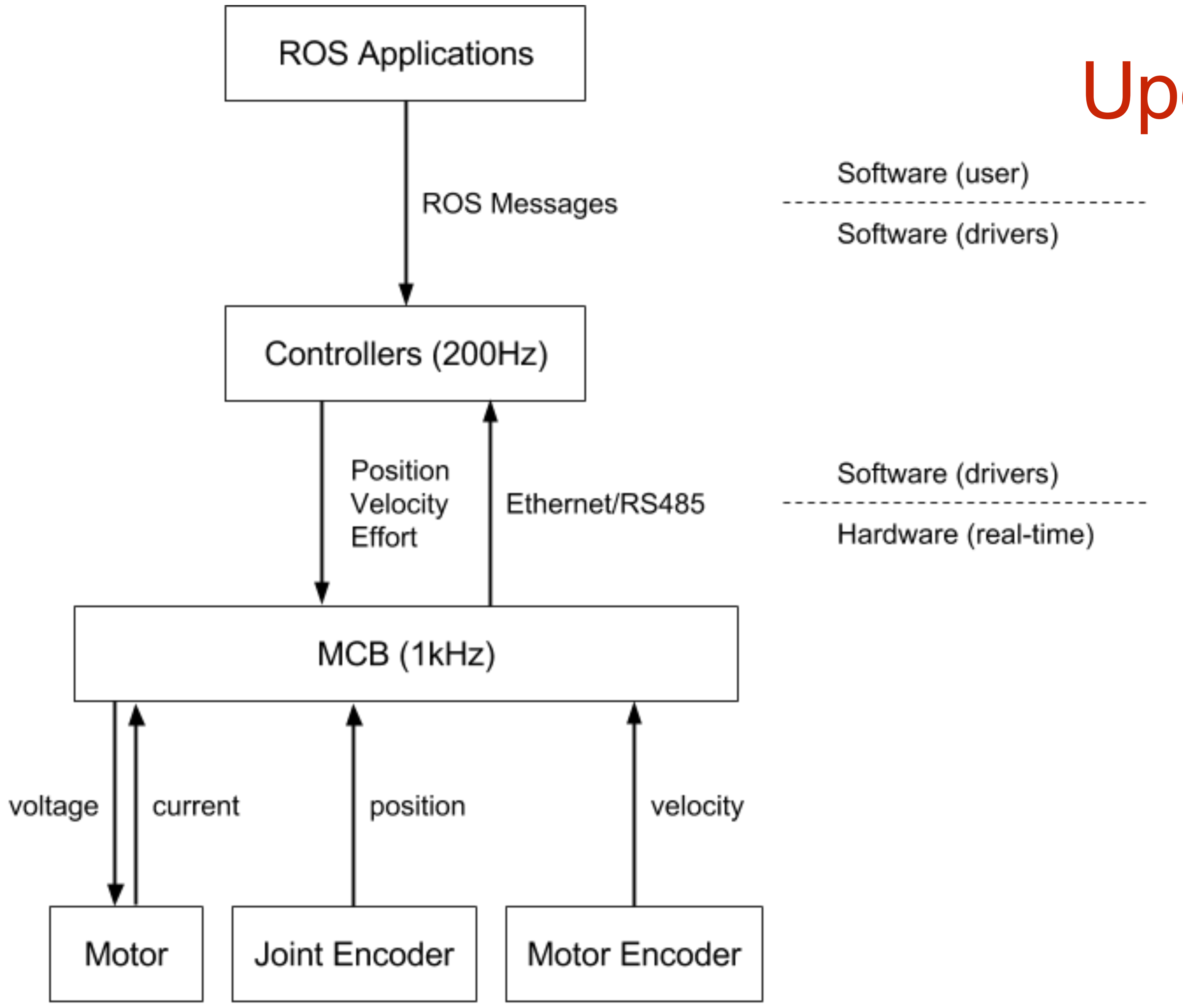
Slide borrowed from Michigan Robotics autorob.org

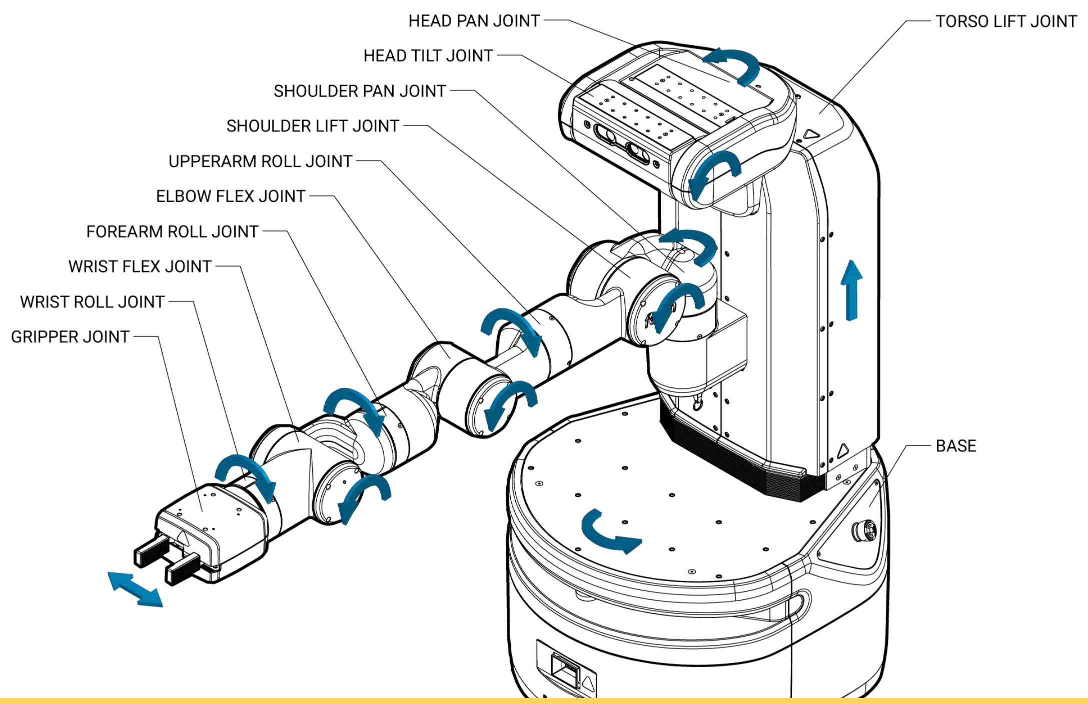


Slide borrowed from Michigan Robotics autorob.org

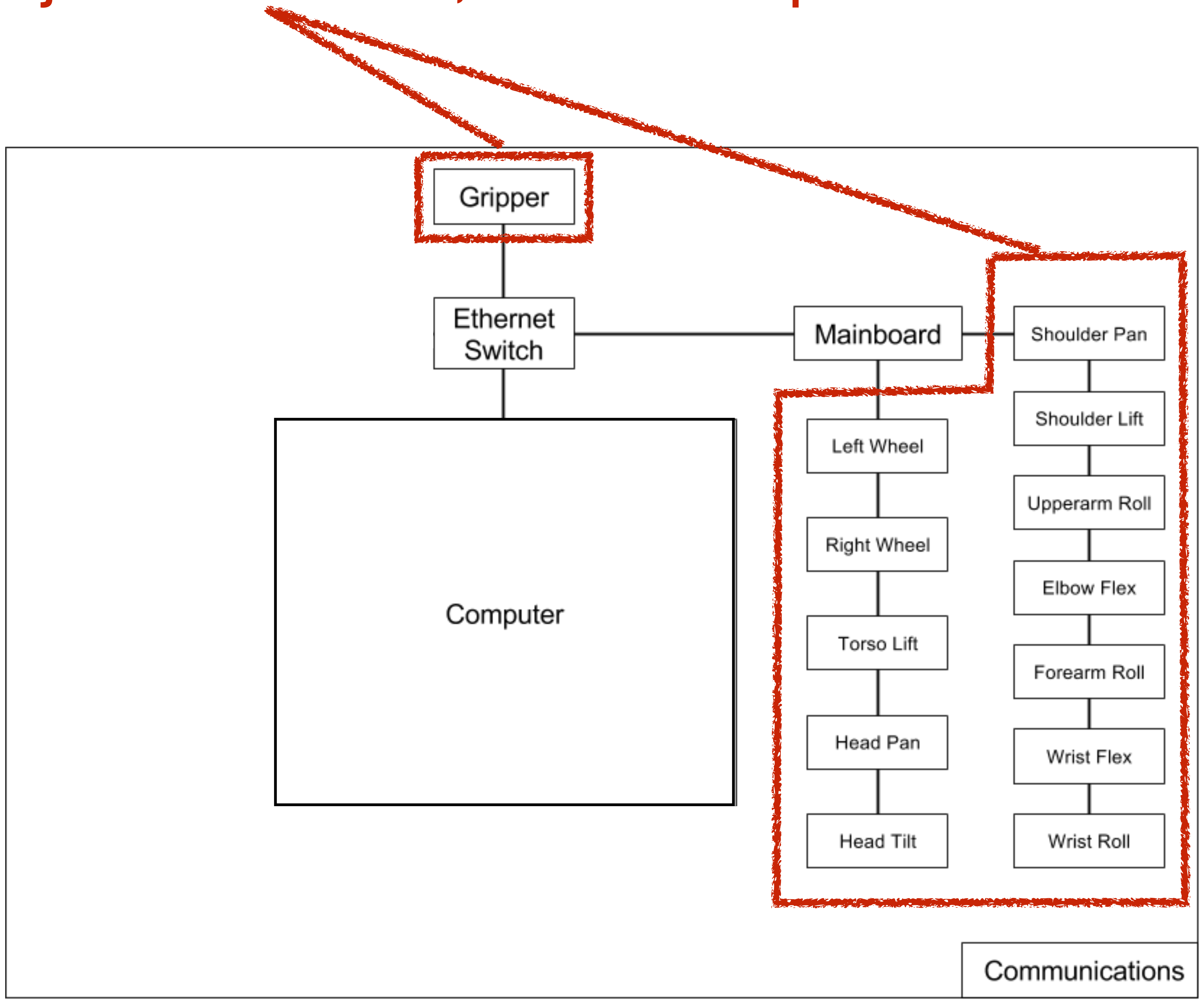
Update joint control 1,000 times per second

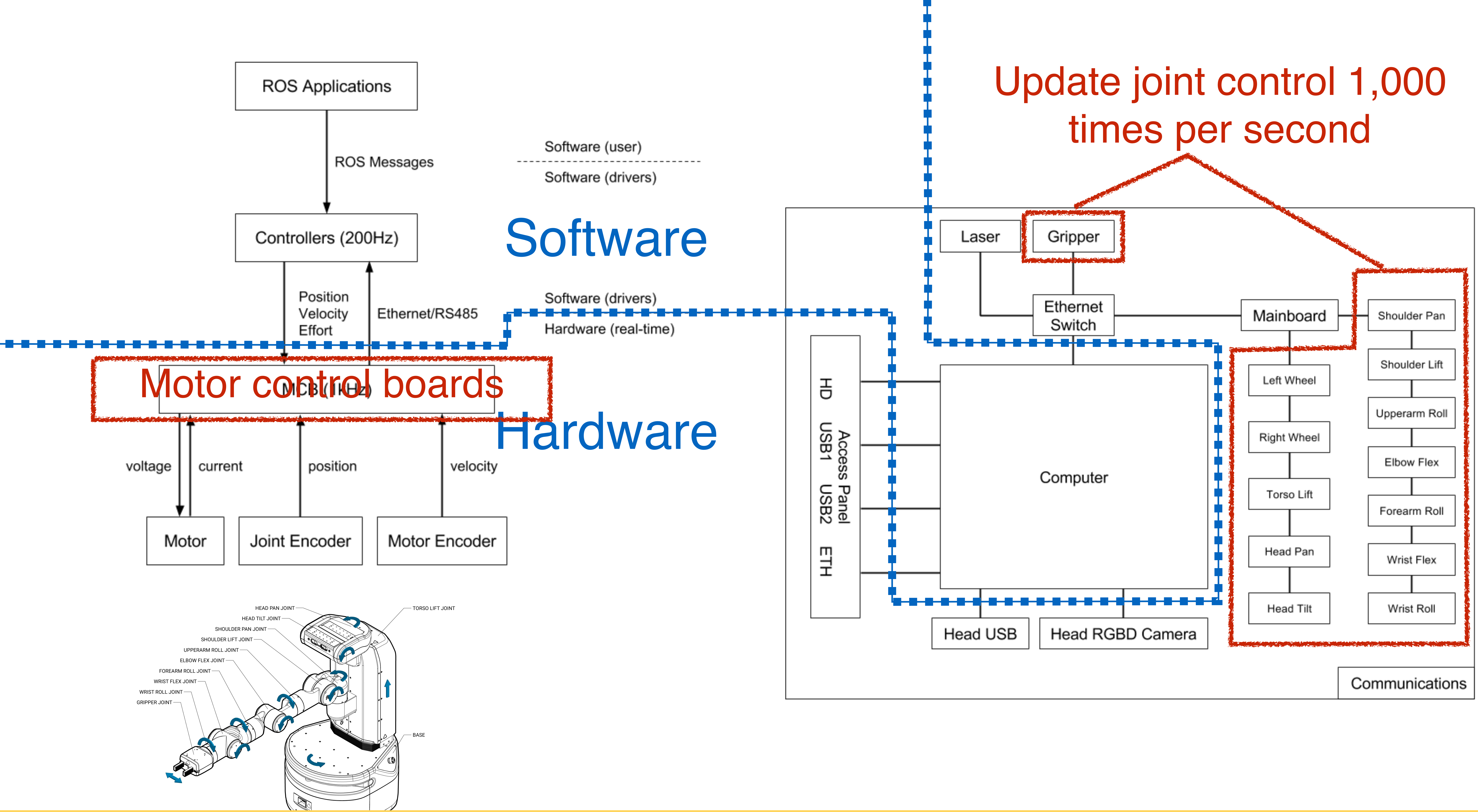






Update joint control 1,000 times per second

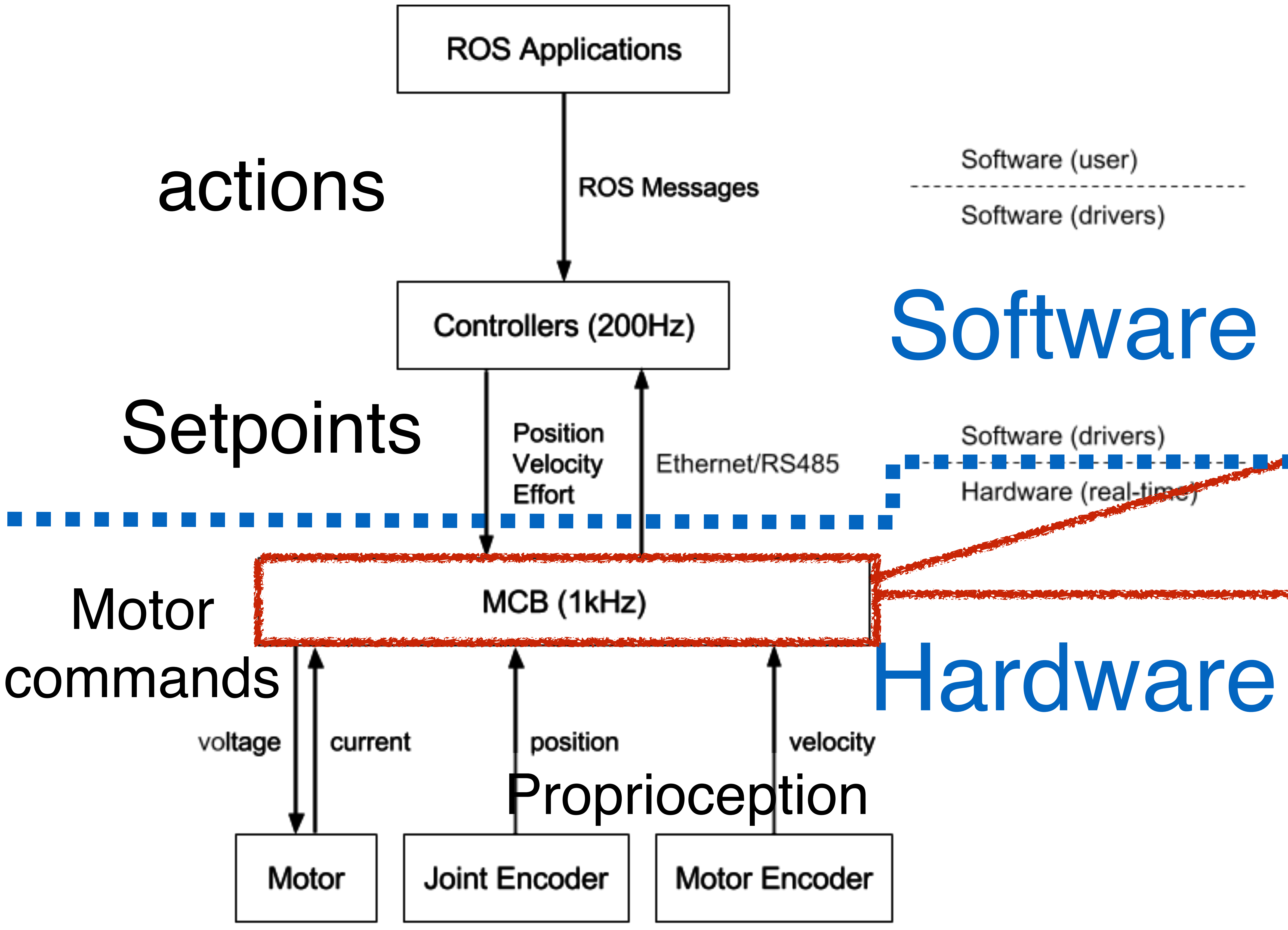


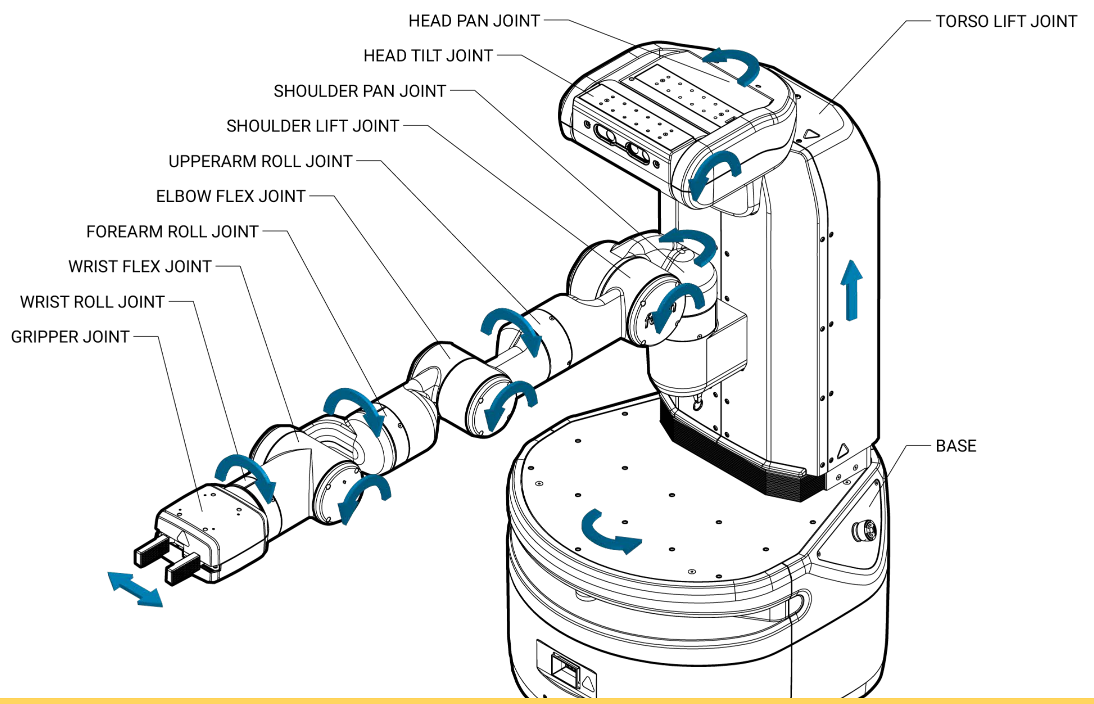


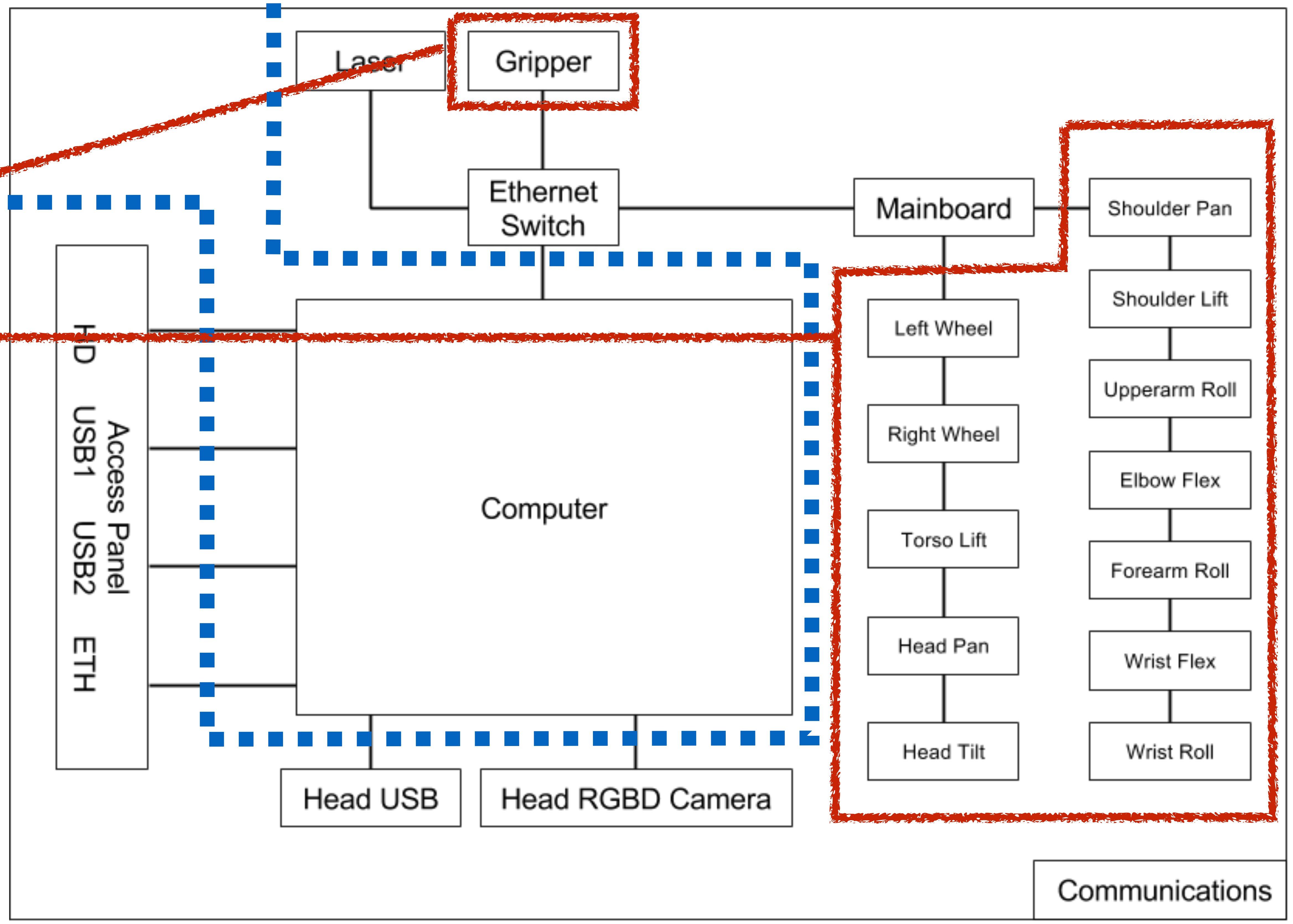
Slide borrowed from Michigan Robotics autorob.org

Motor control boards

Update joint control 1,000 times per second

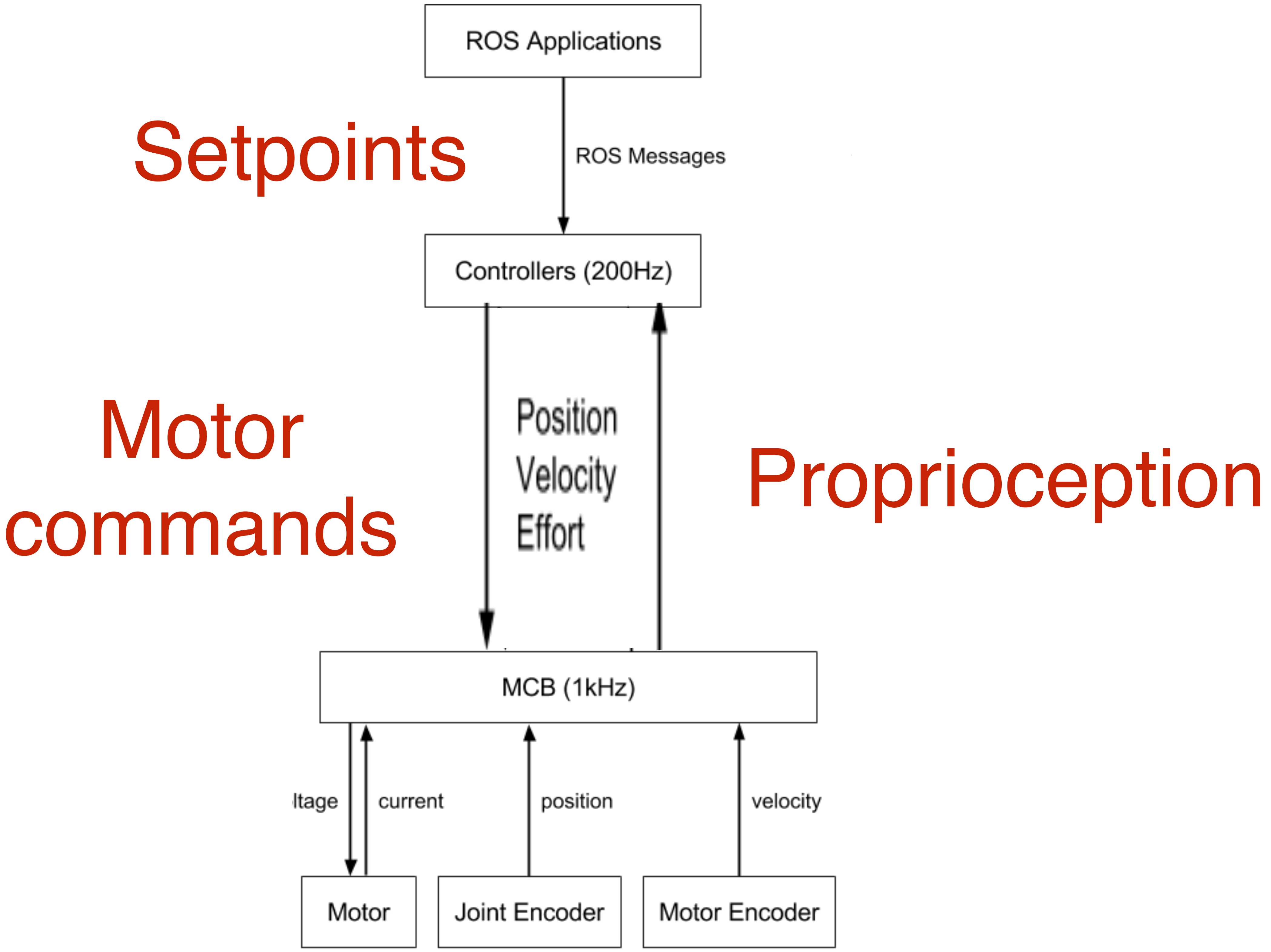


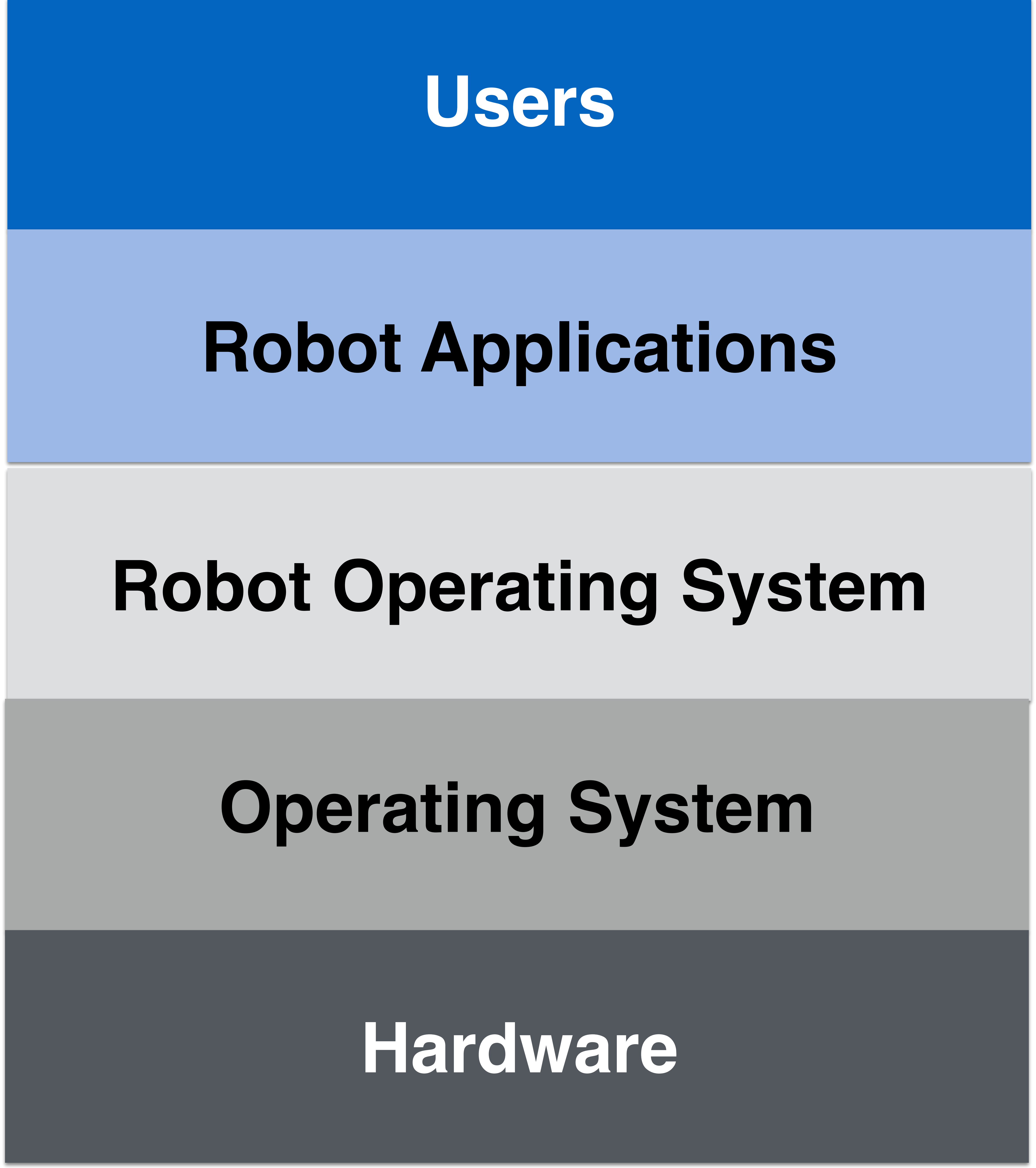




Motion control:

process to achieve and maintain a desired state (or setpoint) for a joint through applying motor commands based on current proprioceived state





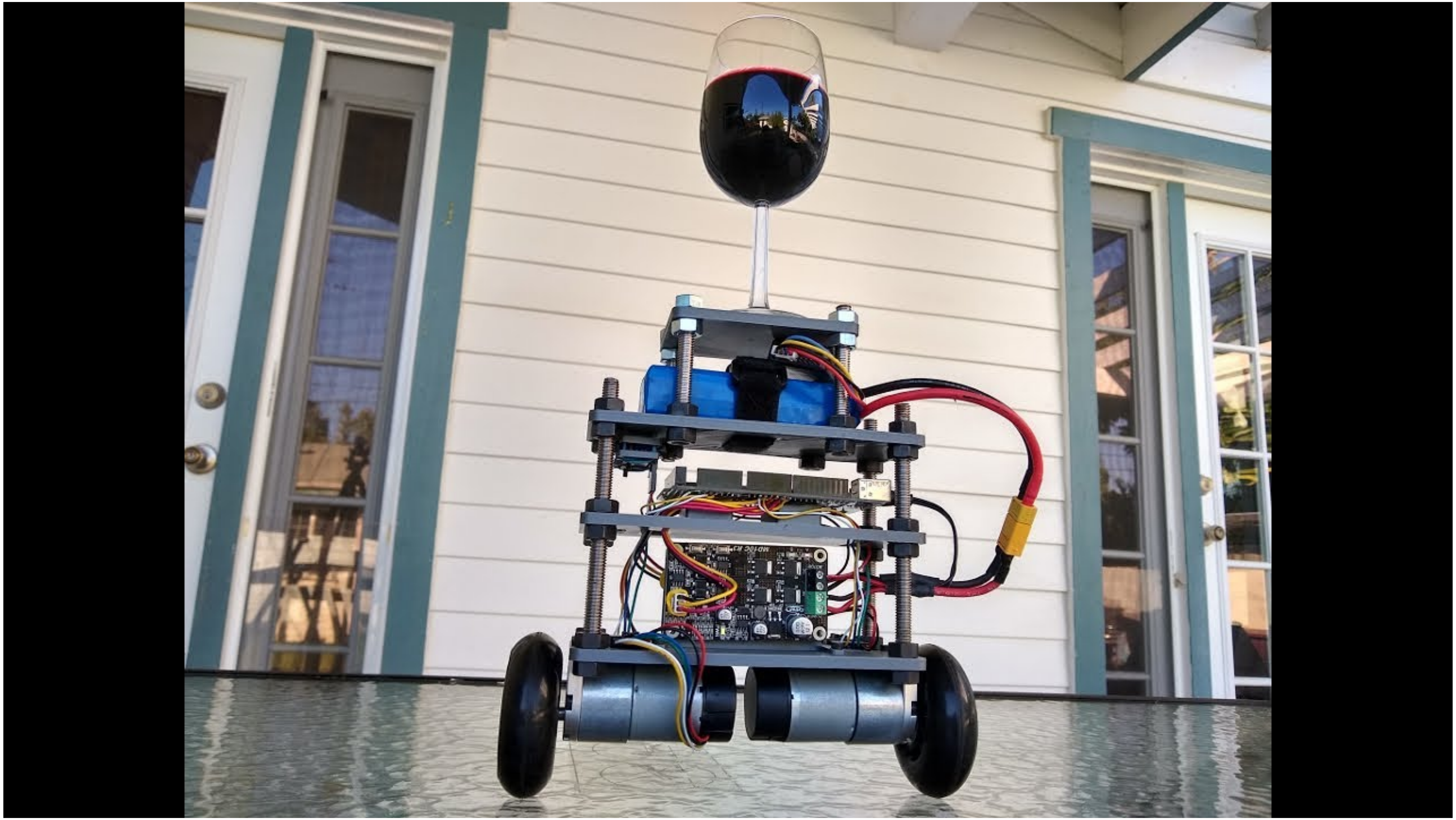
Pendularm

Inverted Pendulum by PID control - Chengcheng Zhu et al. @UMich

Inverted Pendulum by PID control - Chengcheng Zhu et al. @UMich

Monster Inverted Pendulum - K. French et al.

Building up to more…



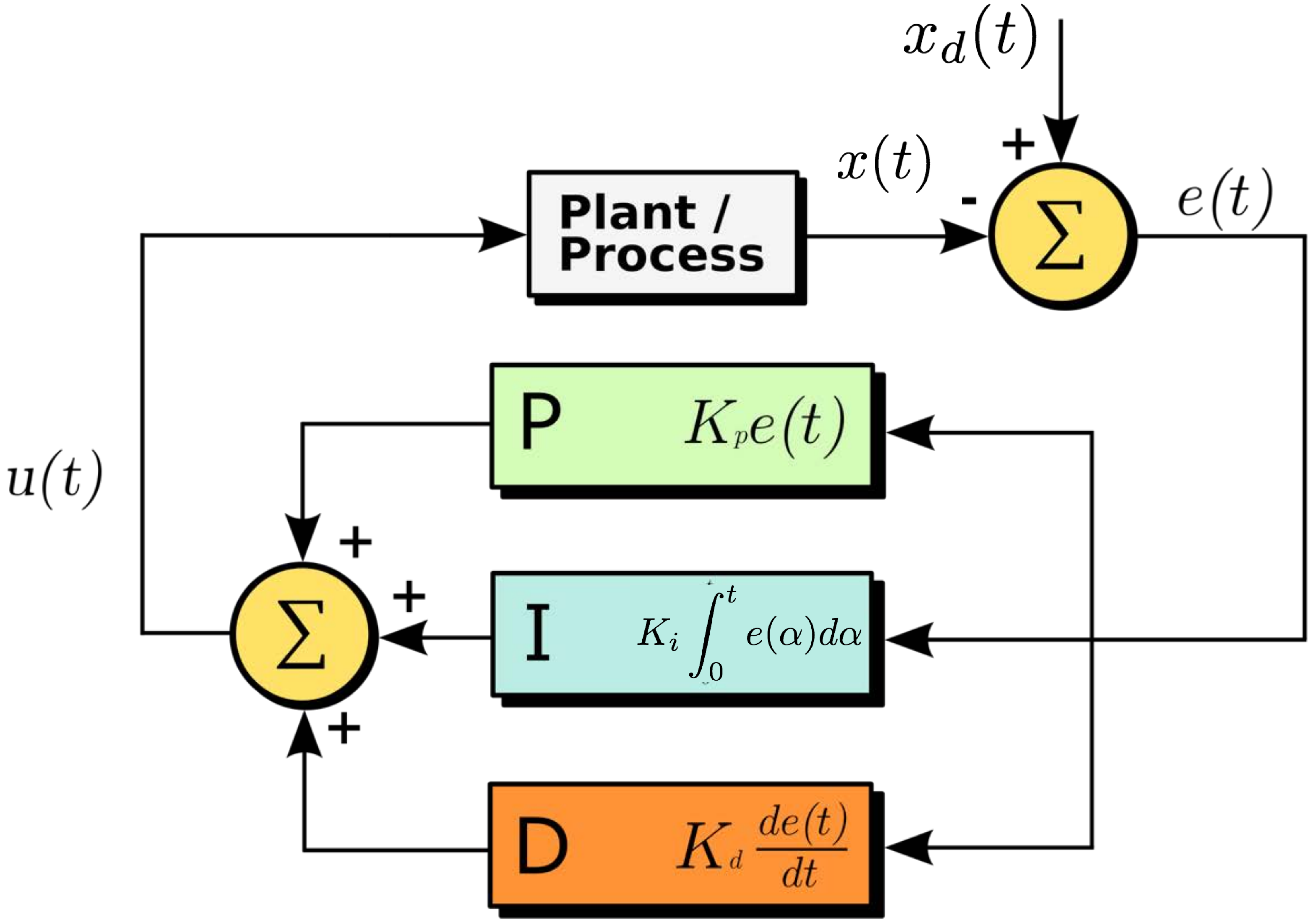
Shay Sackett - https://www.youtube.com/watch?v=t7skvD6v2TI

Remember PID?

PID Control

PID Control

- Proportional-Integral-Derivative Control
- Sum of different responses to error
- Based on the mass spring and damper system
- Feedback correction based on the current error, past error, and predicted future error

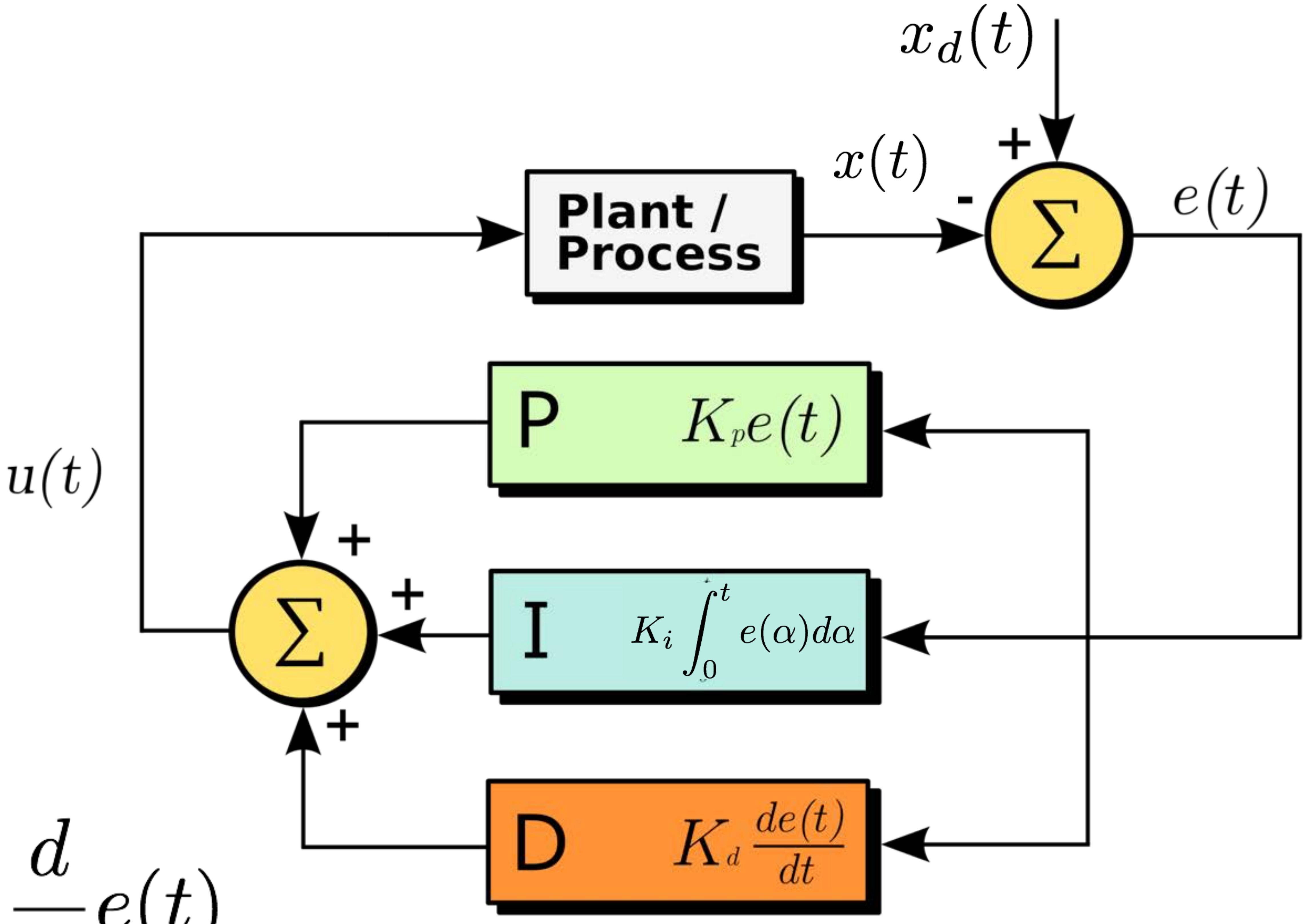


Slide borrowed from Michigan Robotics autorob.org

PID Control

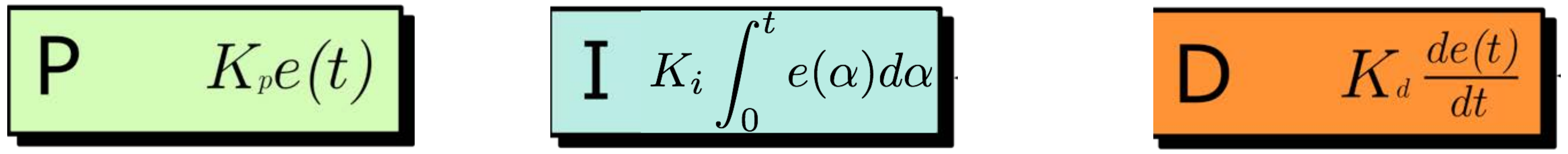
Error signal:

$$e(t) = x_{desired}(t) - x(t)$$



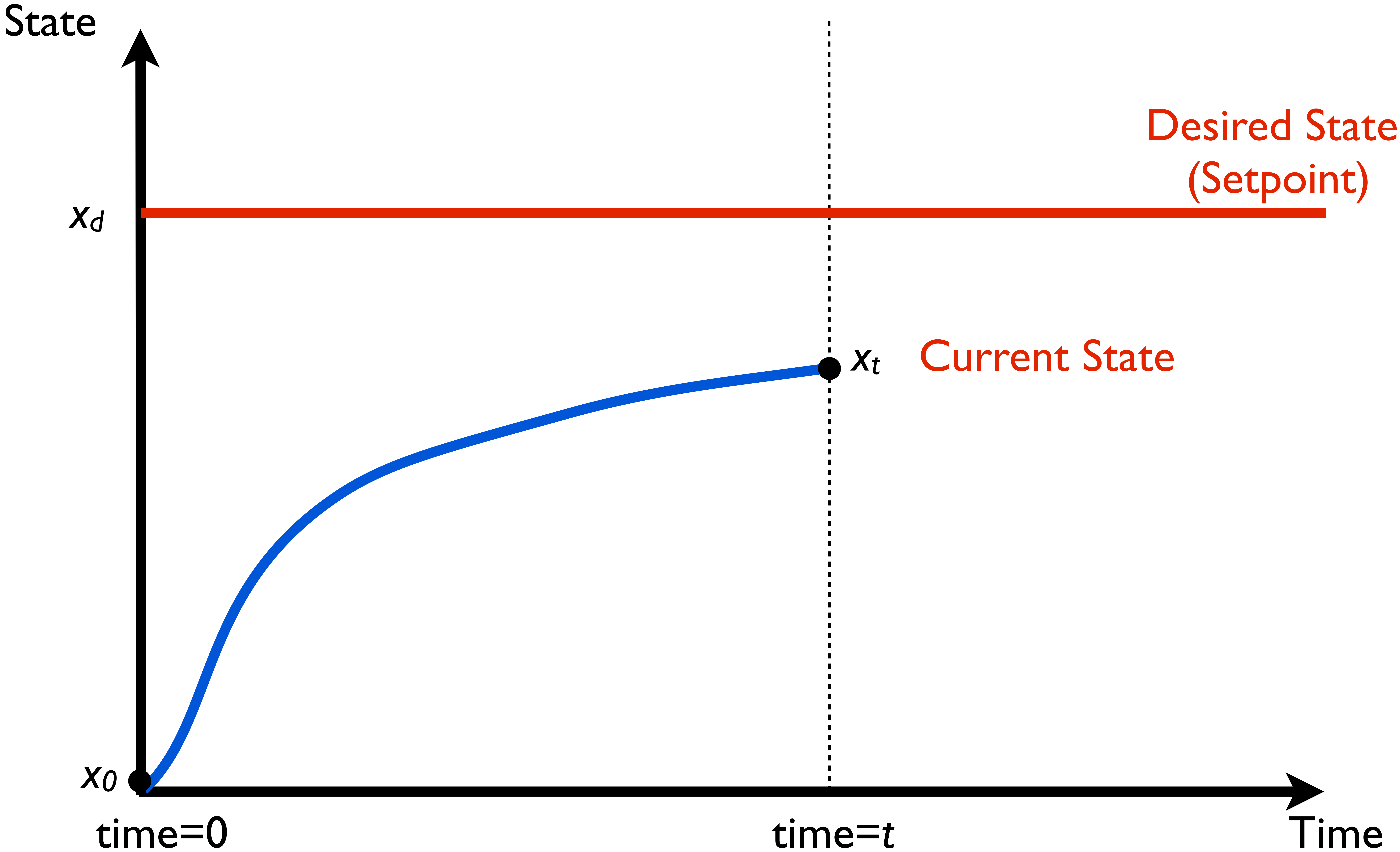
Control signal:

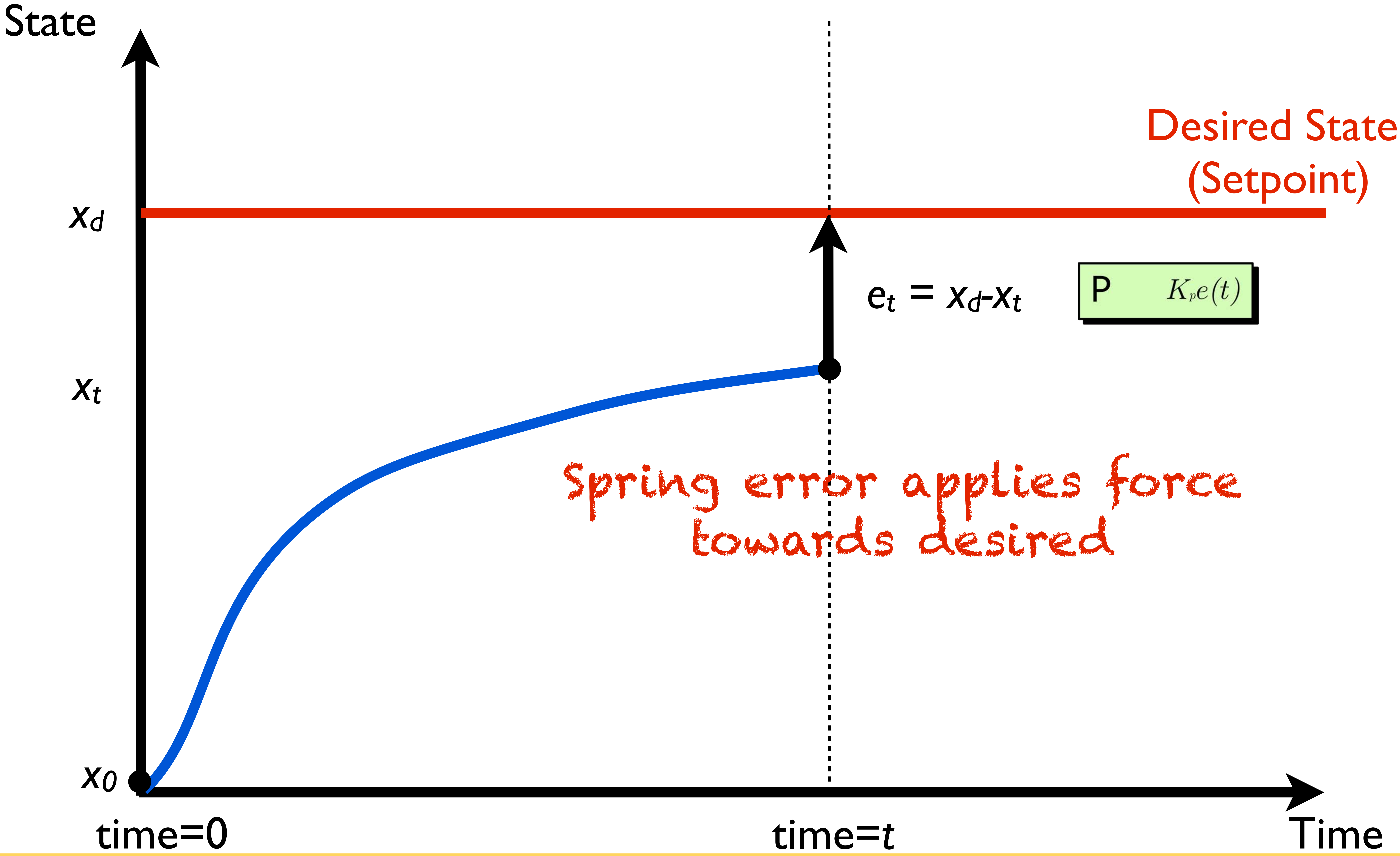
$$u(t) = K_p e(t) + K_i \int_0^t e(\alpha) d\alpha + K_d \frac{d}{dt} e(t)$$

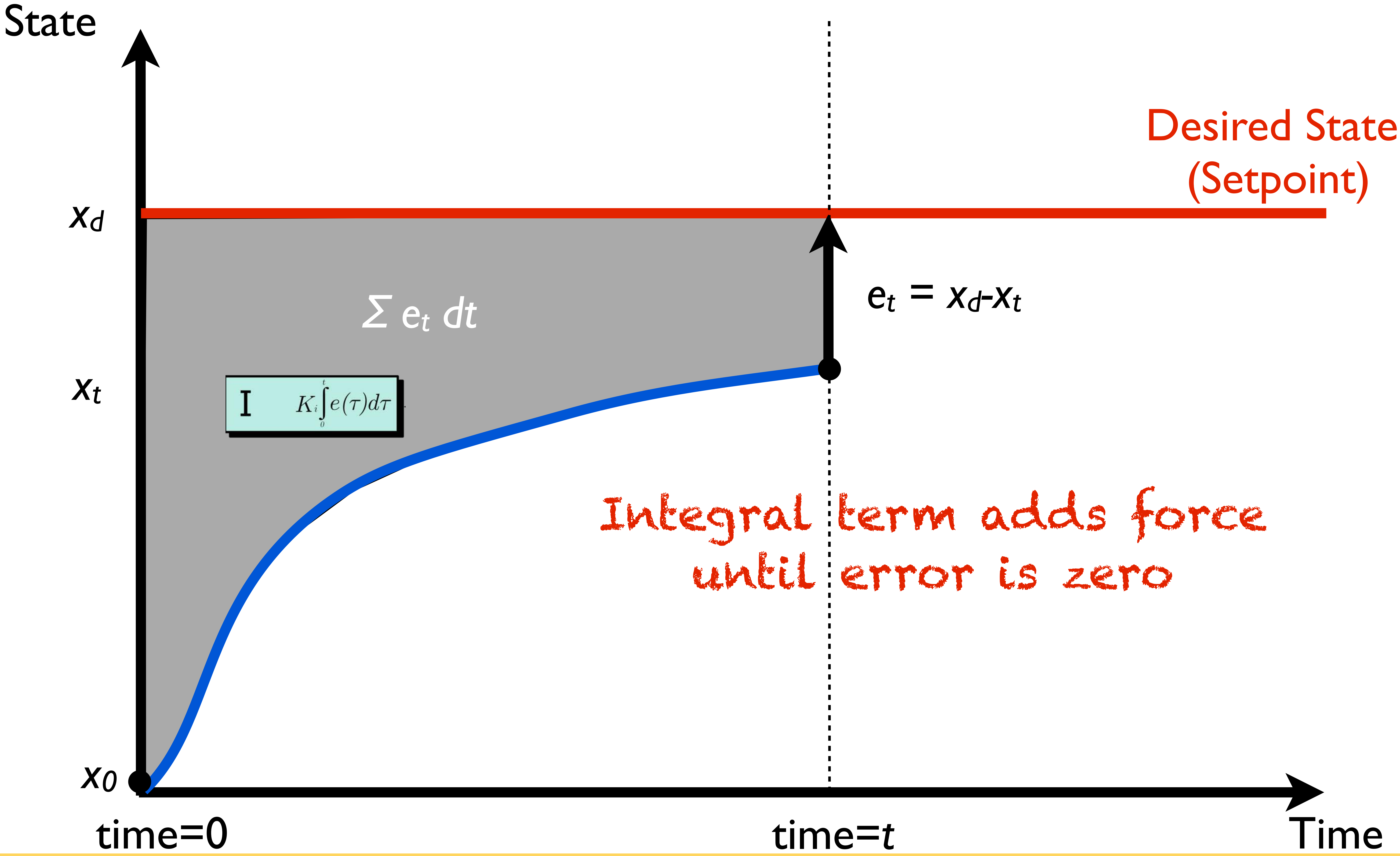


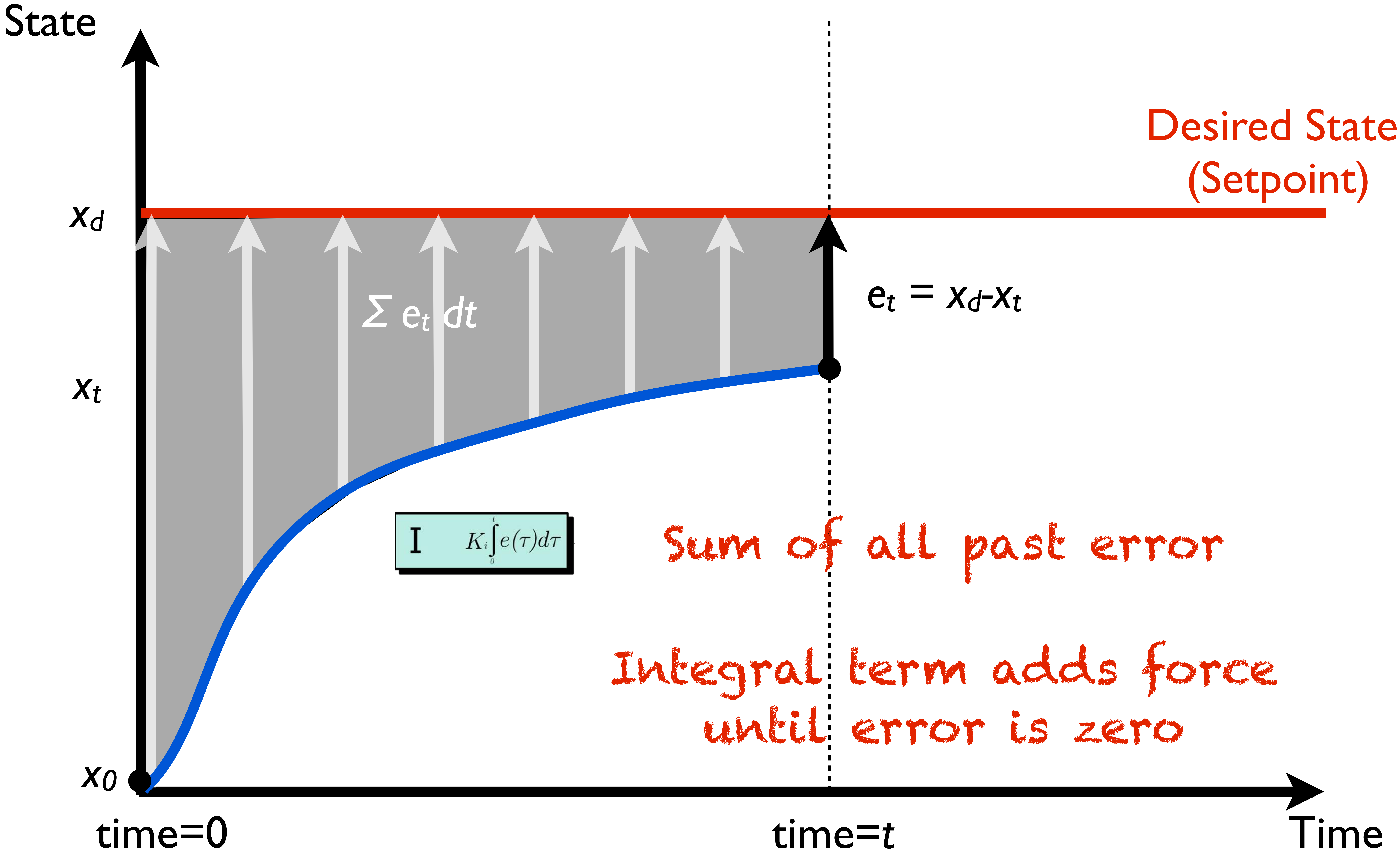
Current Past Future

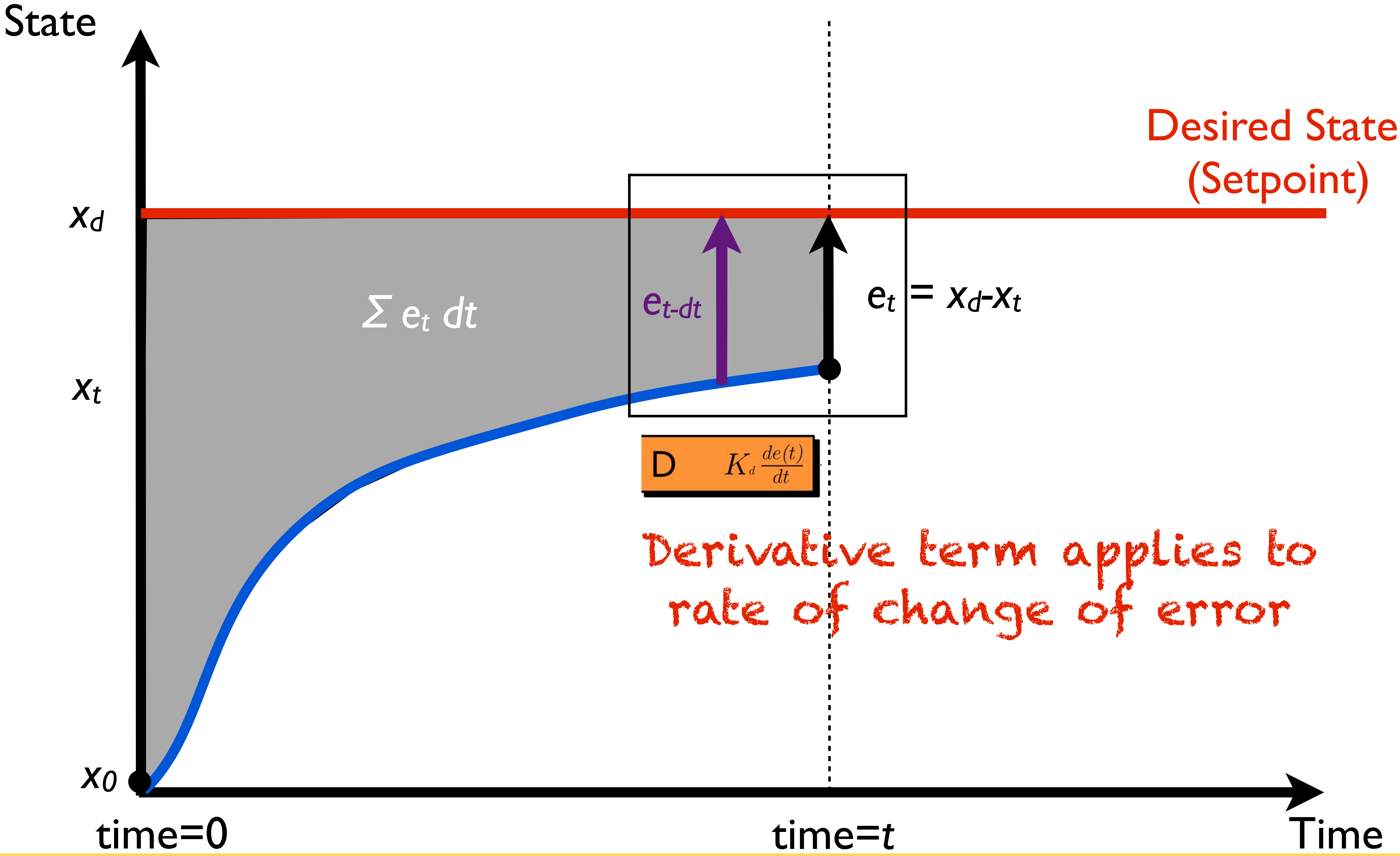
Consider PID wrt. state over time


State
Desired State
(Setpoint)
x_d
x_t
Current State
x_0
time=0
time=t
Time

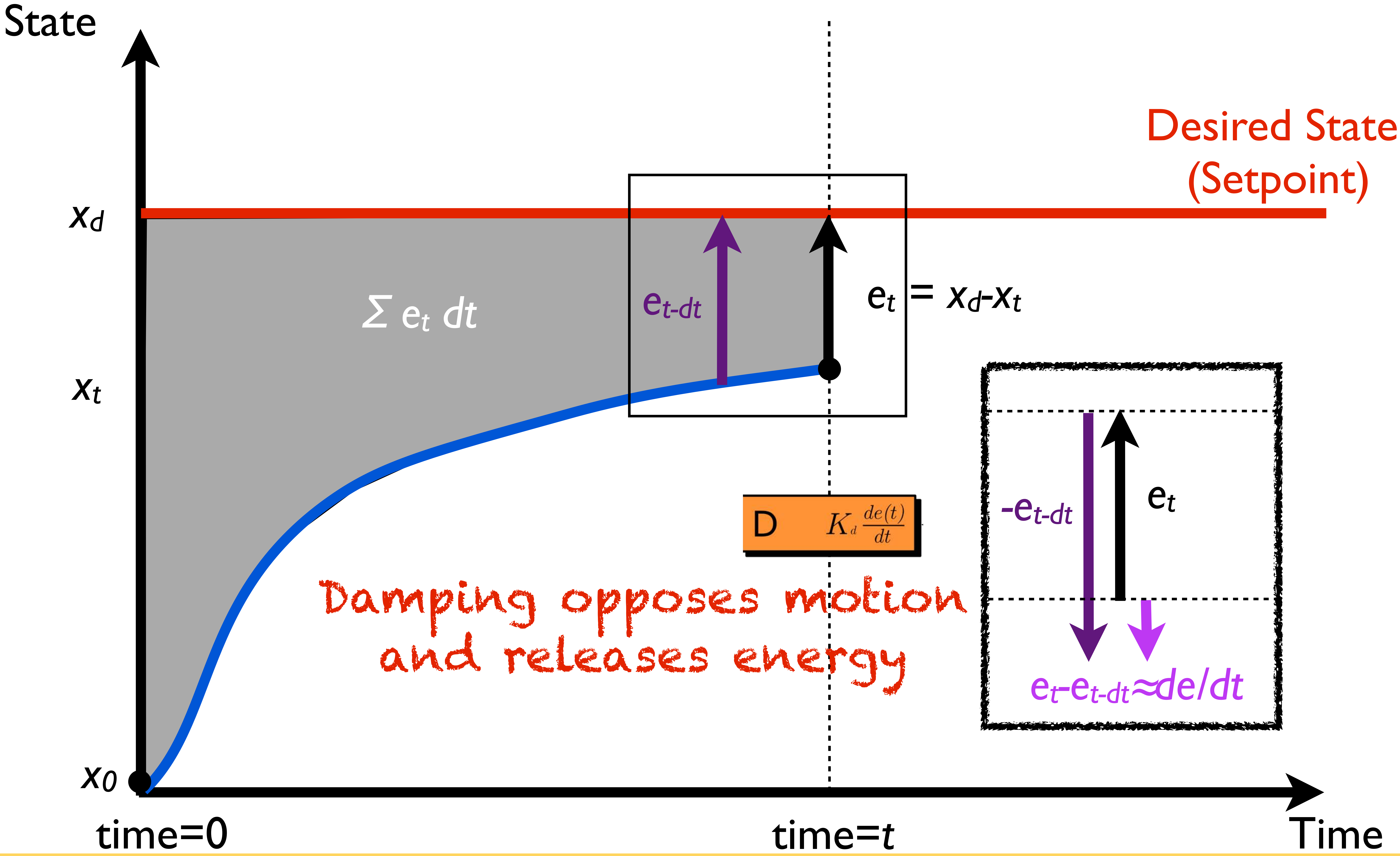

State
Desired State
(Setpoint)
x_d
$e_t = x_d - x_t$
P $K_p e(t)$
x_t
Spring error applies force
towards desired
x_0
time=0
time=t
Time

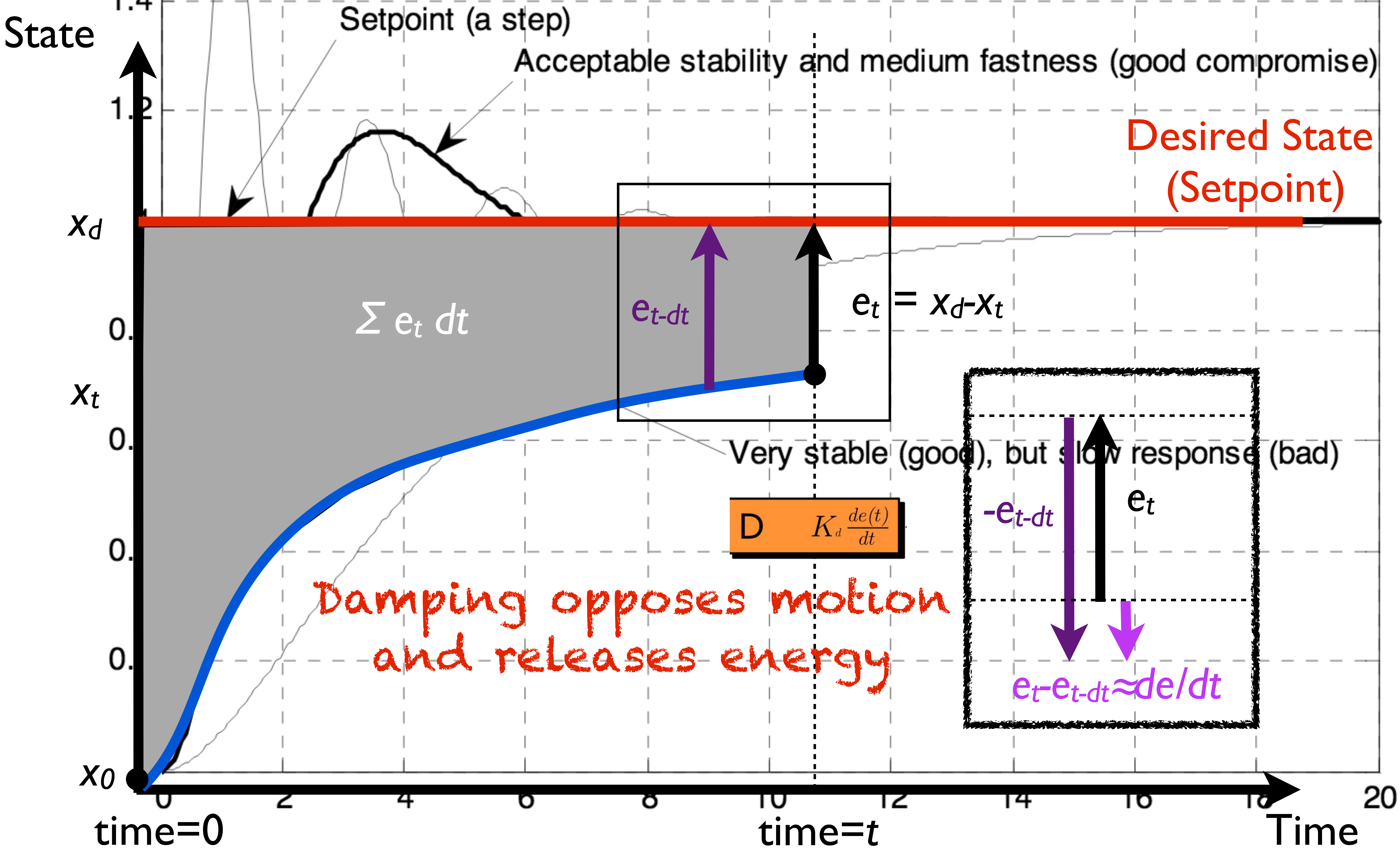

State
Desired State
(Setpoint)
x_d
x_t
x_0
$\Sigma\ e_t\ dt$
I $K_i \int_0^t e(\tau)d\tau$
$e_t = x_d - x_t$
Integral term adds force
until error is zero
time=0
time=t
Time


State
Desired State
(Setpoint)
x_d
x_t
x_0
$\Sigma e_t dt$
$e_t = x_d - x_t$
I $K_i \int_0^t e(\tau) d\tau$
Sum of all past error
Integral term adds force
until error is zero
time=0
time=t
Time



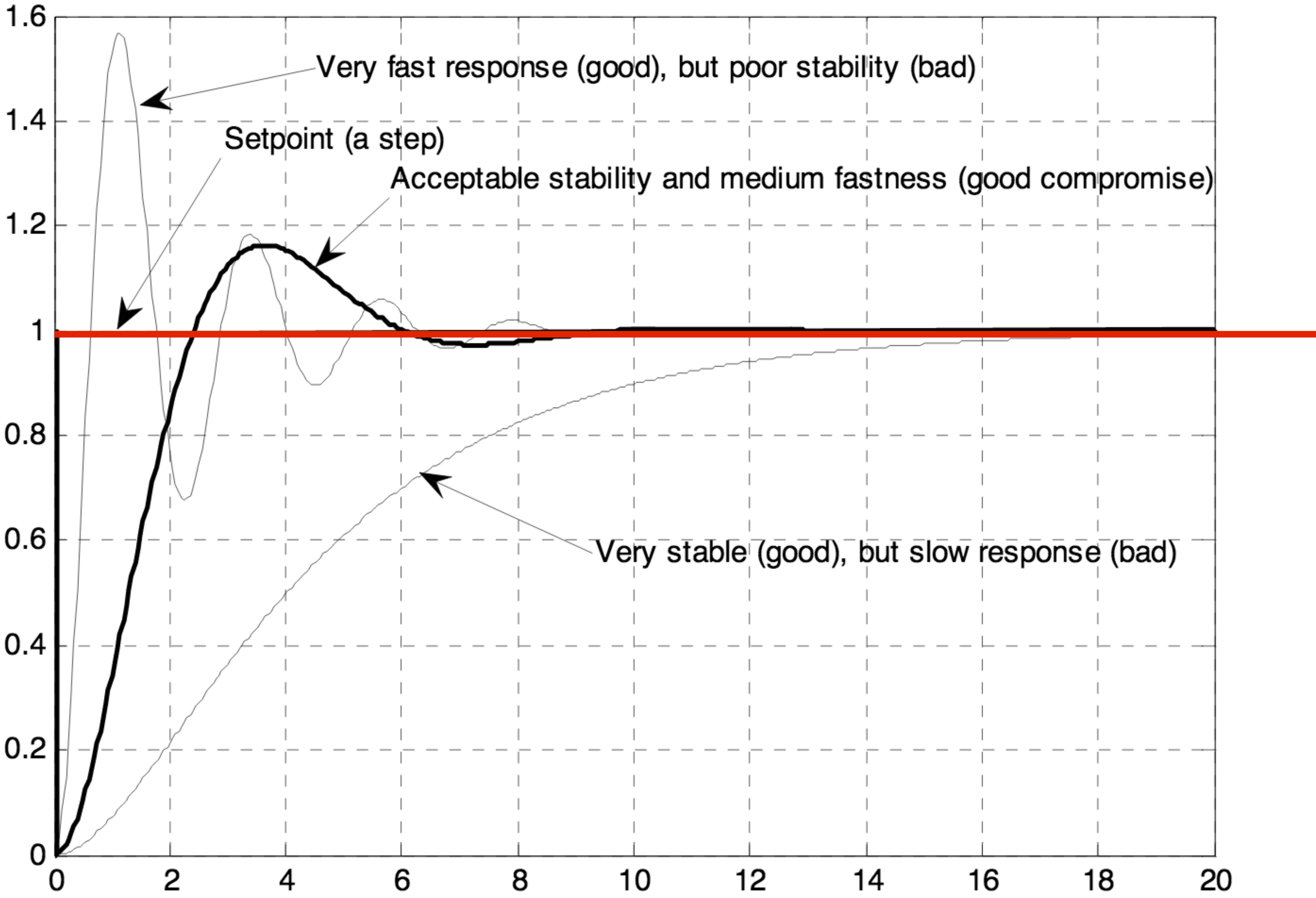
Derivative term applies to rate of change of error


State
Desired State
(Setpoint)
x_d
x_t
x_0
$\Sigma\, e_t\, dt$
e_{t-dt}
$e_t = x_d - x_t$
D $K_d \frac{de(t)}{dt}$
Damping opposes motion
and releases energy
$-e_{t-dt}$
e_t
$e_t - e_{t-dt} \approx de/dt$
time=0
time=t
Time



Slide borrowed from Michigan Robotics autorob.org

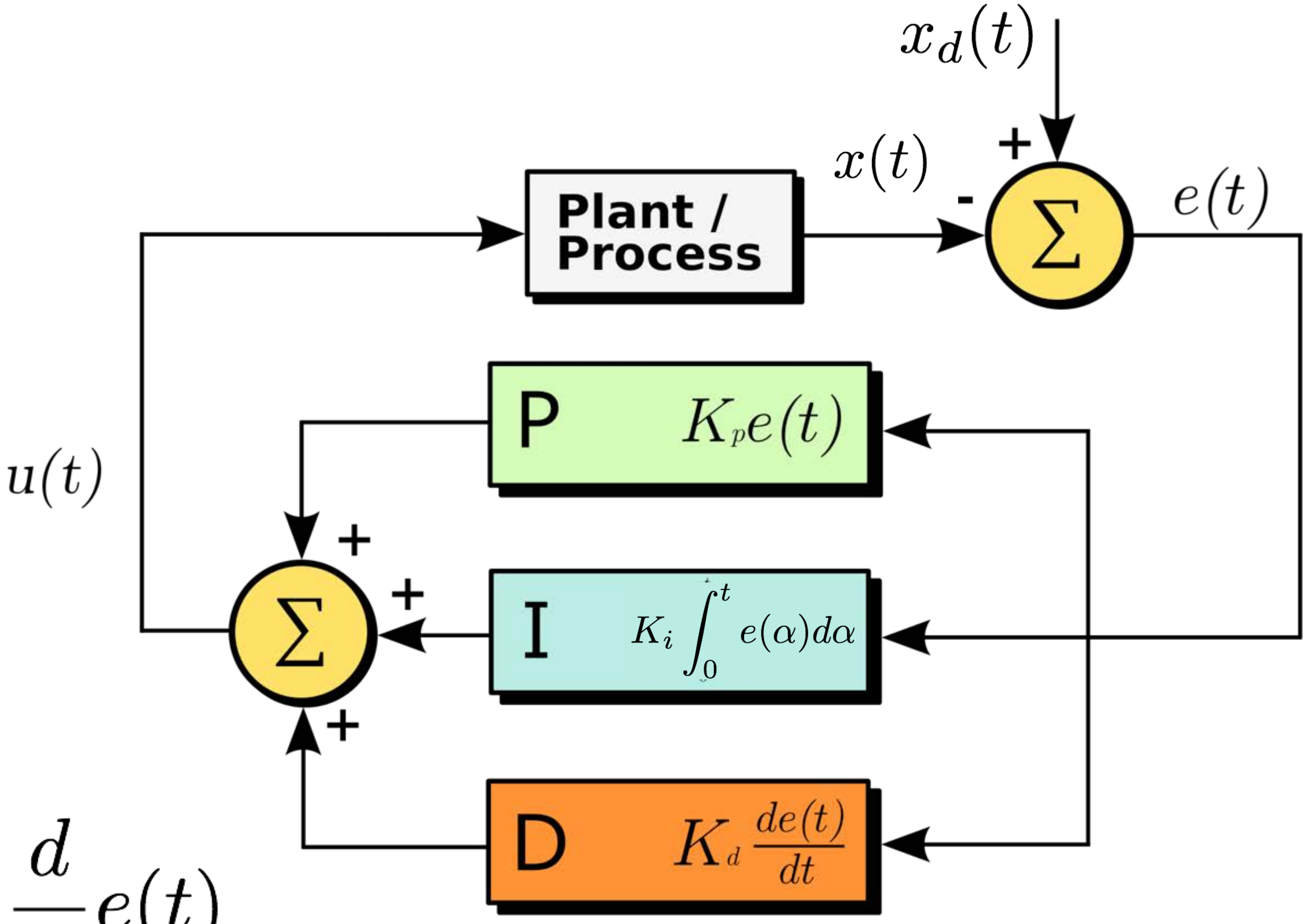
PID Converenge



Slide borrowed from Michigan Robotics autorob.org

PID as a spring and damper model

PID Control

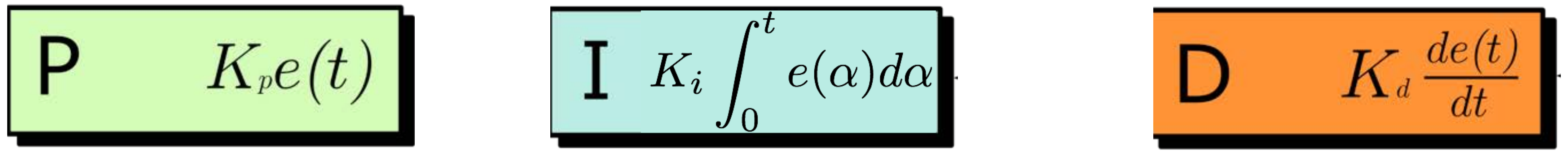


Error signal:

$$e(t) = x_{desired}(t) - x(t)$$

Control signal:

$$u(t) = K_p e(t) + K_i \int_0^t e(\alpha)d\alpha + K_d \frac{d}{dt}e(t)$$



Current Past Future

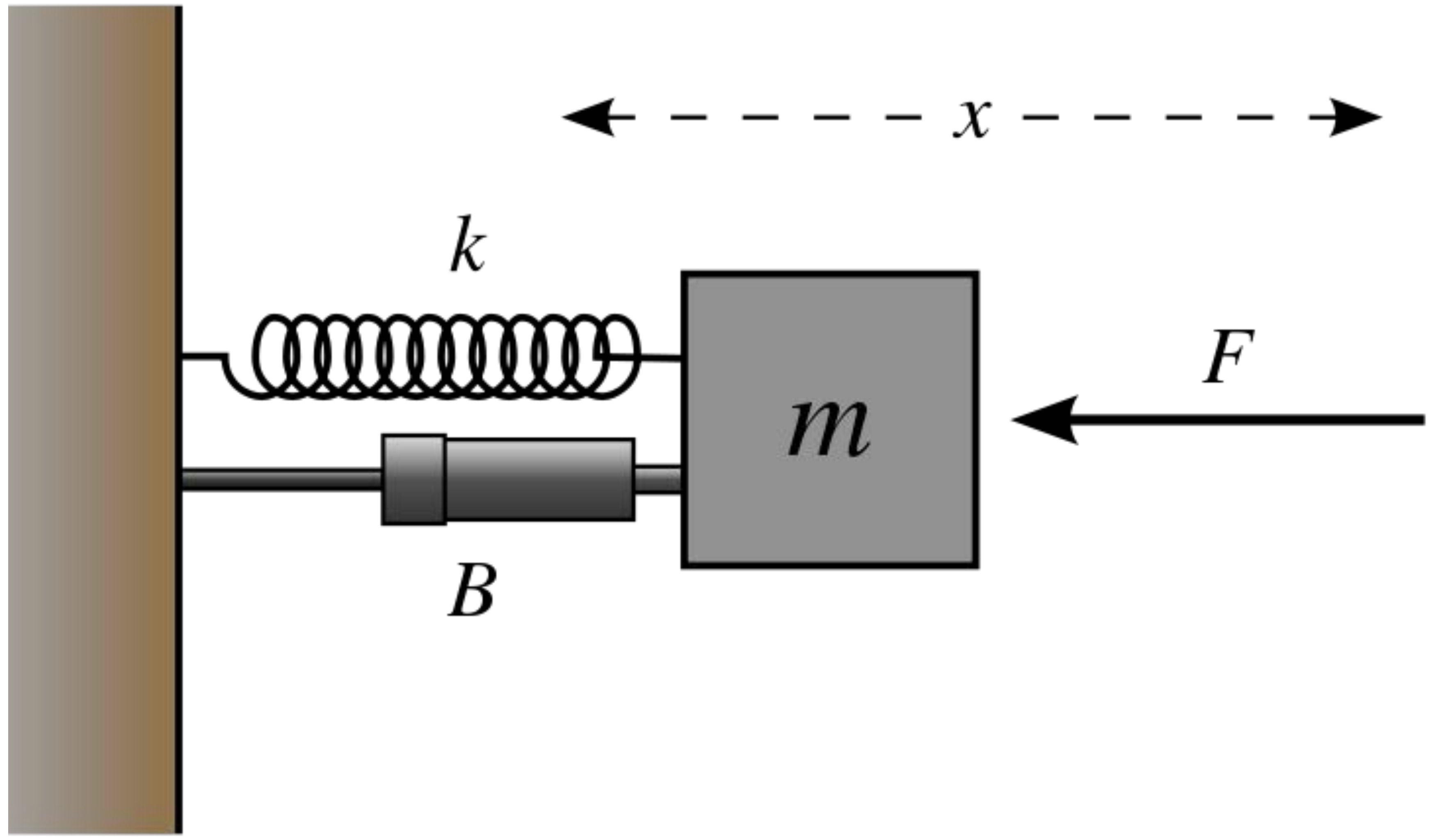
Hooke's Law

P $K_p e(t)$

- Describes motion of mass spring damper system as

$$F = -kx$$

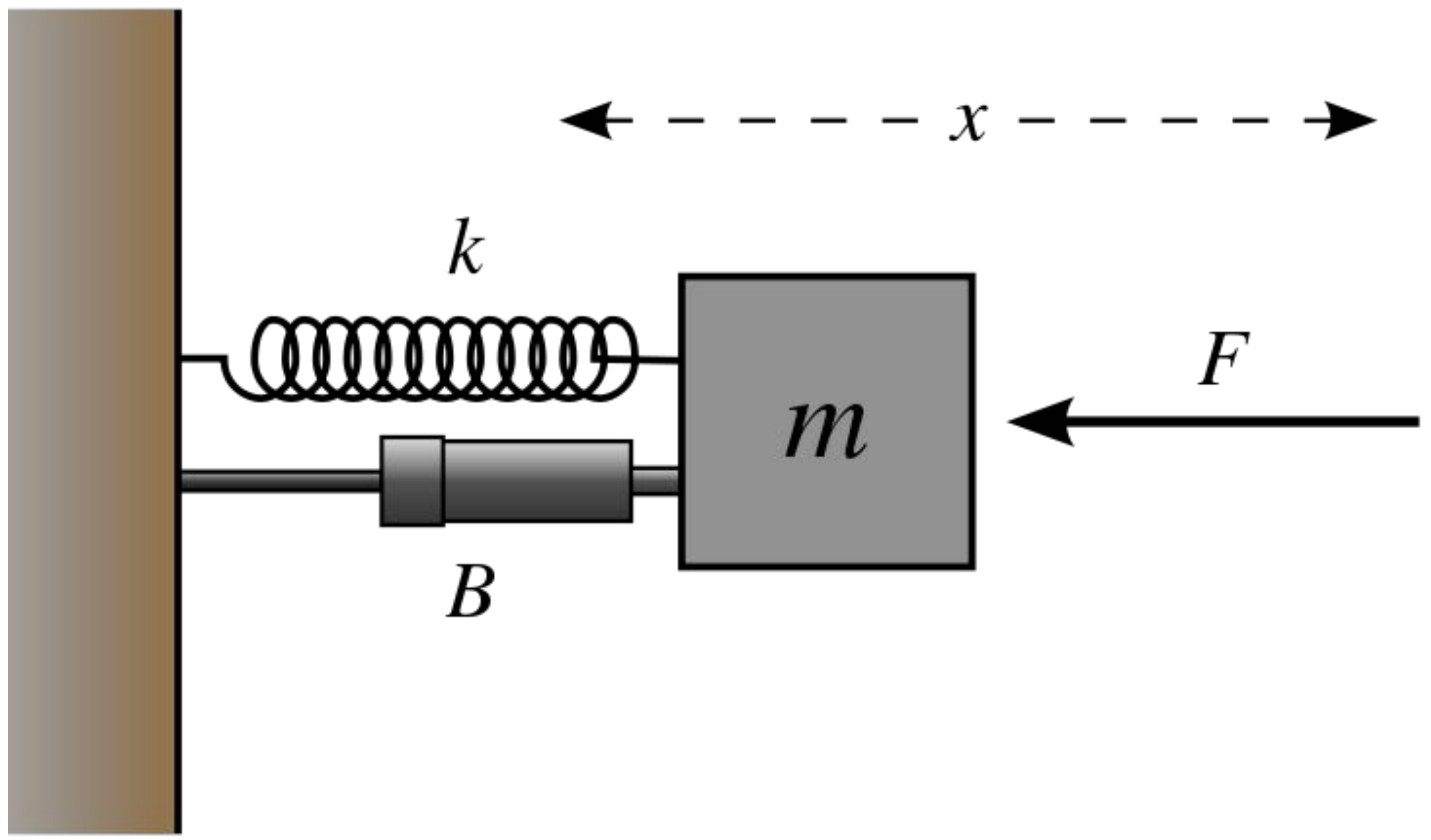
Robert Hooke
(1635-1703)



Hooke's Law

P $K_p e(t)$

- Describes motion of mass spring damper system as



$$F = -kx$$

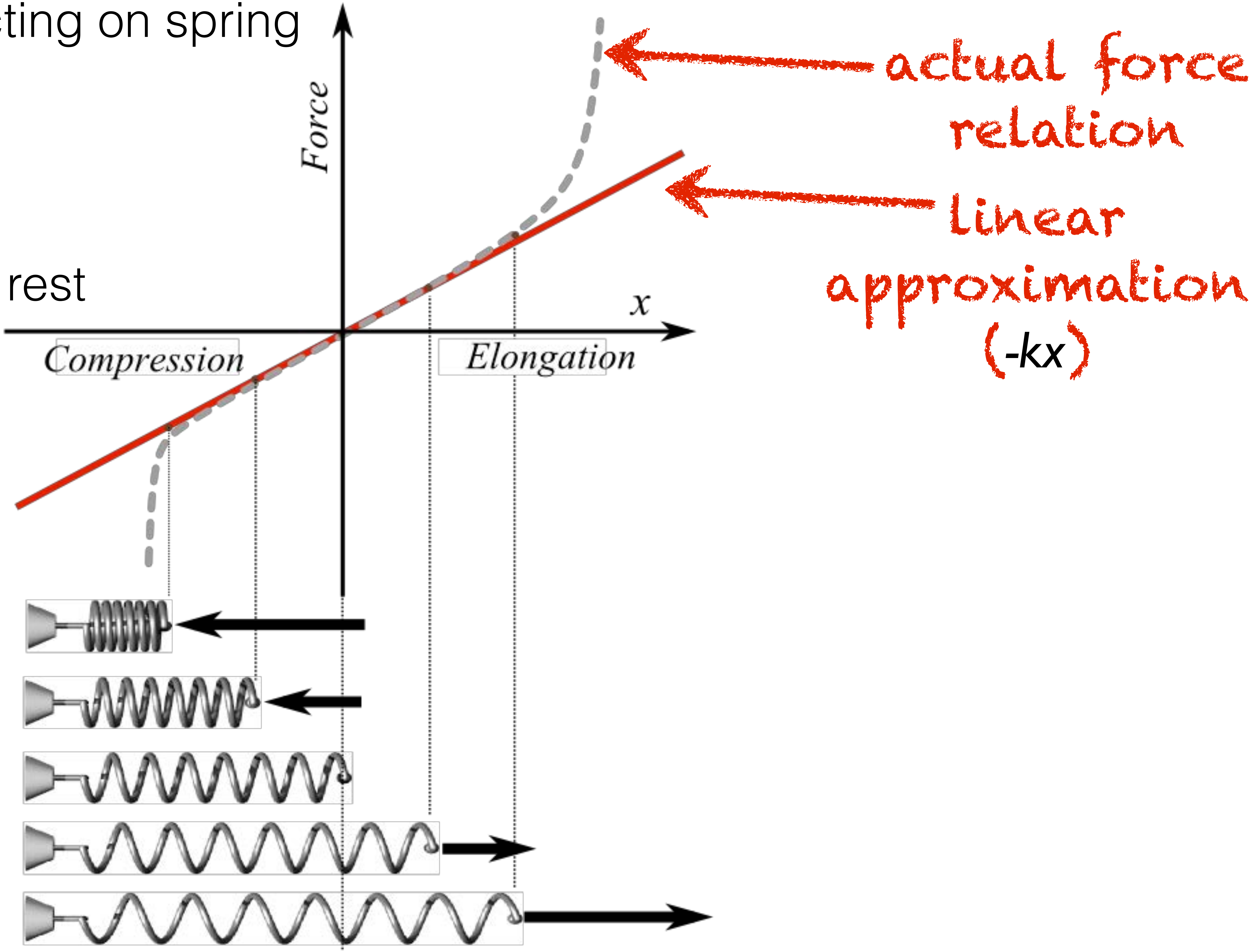
force moving spring towards rest

spring stiffness

distance from rest displacement

Vertical: Force acting on spring

Horizontal: Position from rest



Slide borrowed from Michigan Robotics autorob.org

$K_p e(t)$

Slide borrowed from Michigan Robotics autorob.org

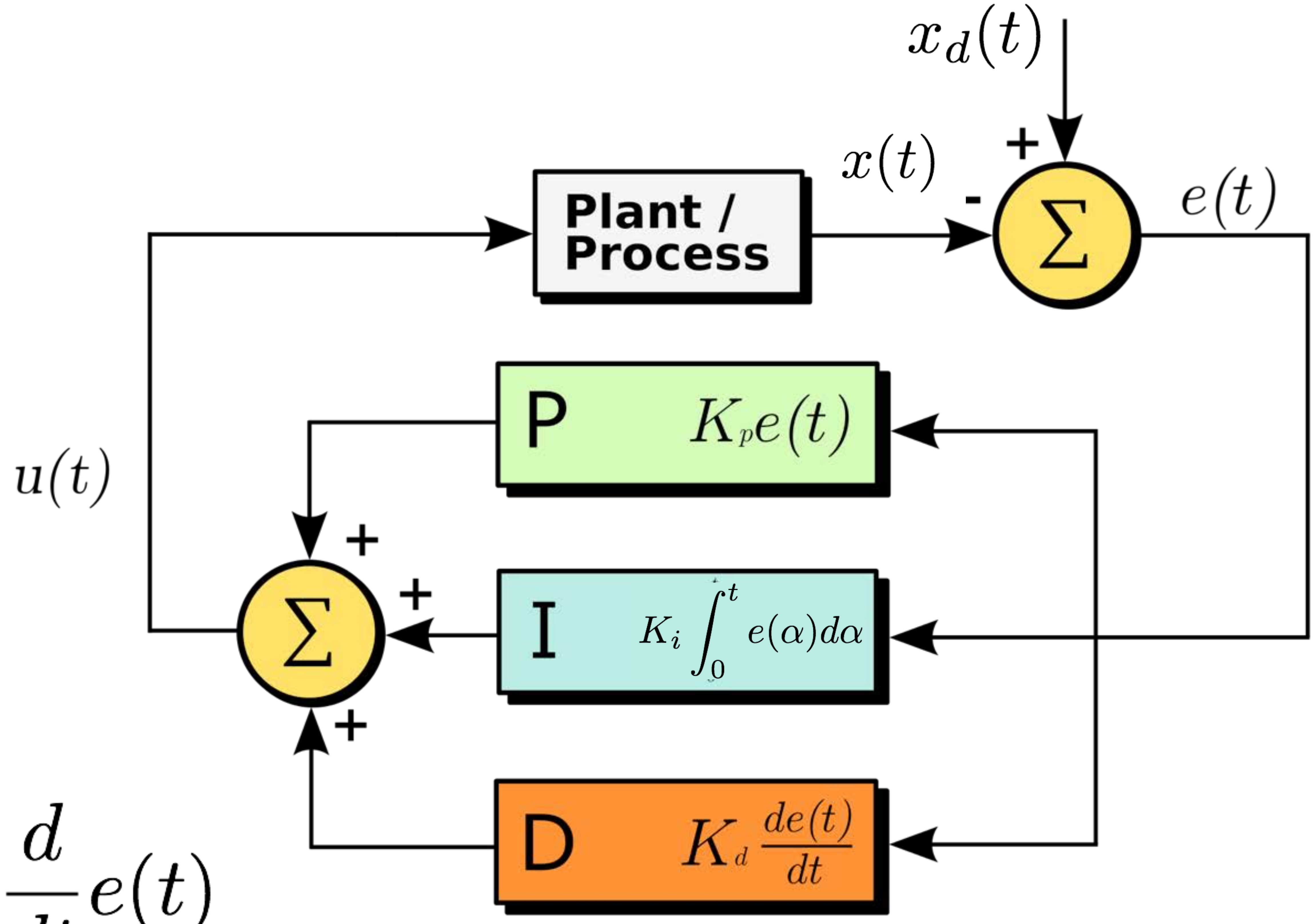
PID Control

Error signal:

$$e(t) = x_{desired}(t) - x(t)$$

Control signal:

$$u(t) = K_p e(t) + K_i \int_0^t e(\alpha) d\alpha + K_d \frac{d}{dt} e(t)$$



P $K_p e(t)$ — Current

I $K_i \int_0^t e(\alpha) d\alpha$ — Past

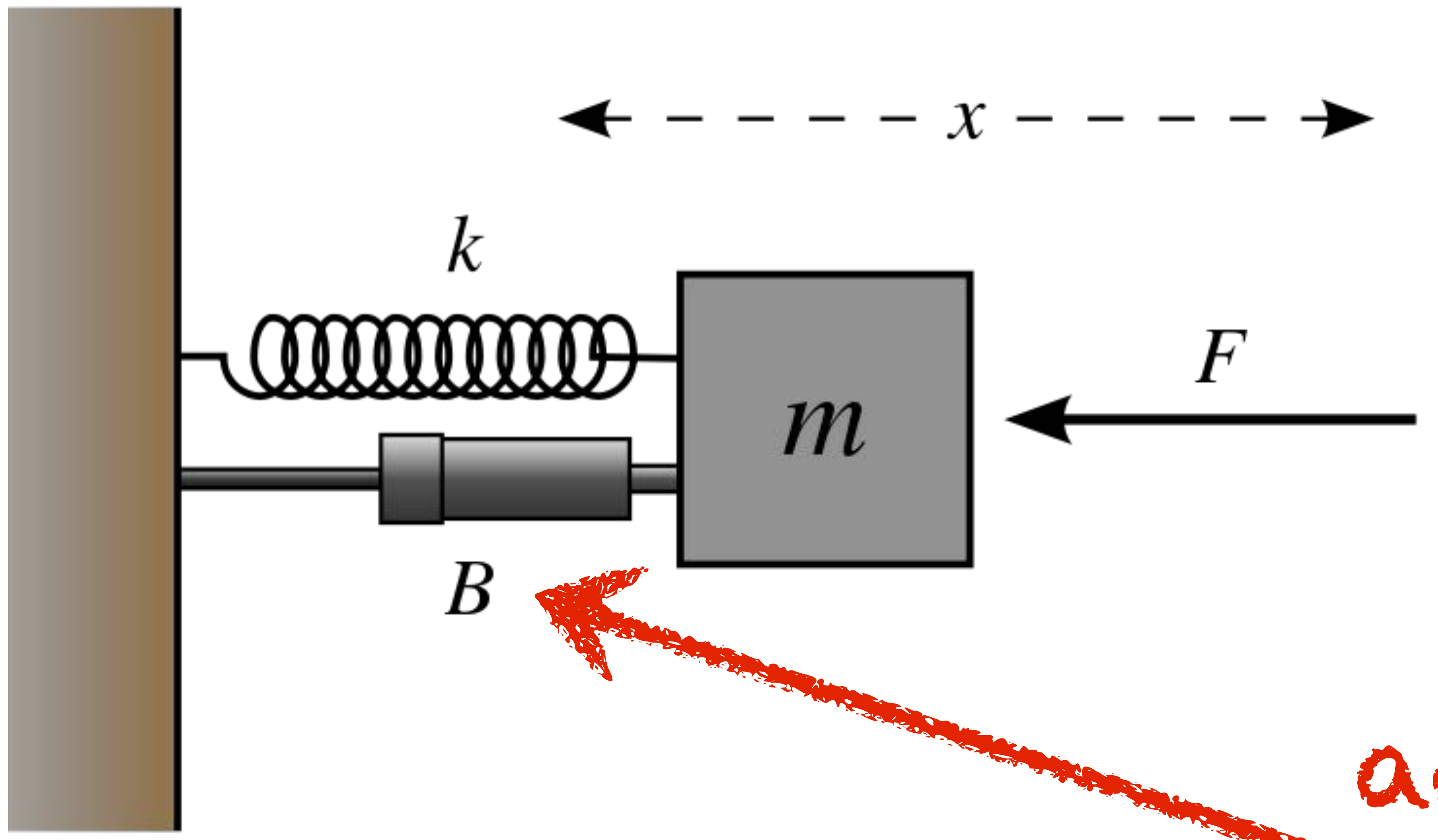
D $K_d \frac{de(t)}{dt}$ — Future

Spring and Damper

P $K_p e(t)$

D $K_d \frac{de(t)}{dt}$

$$F = -kx + -b\dot{x}$$

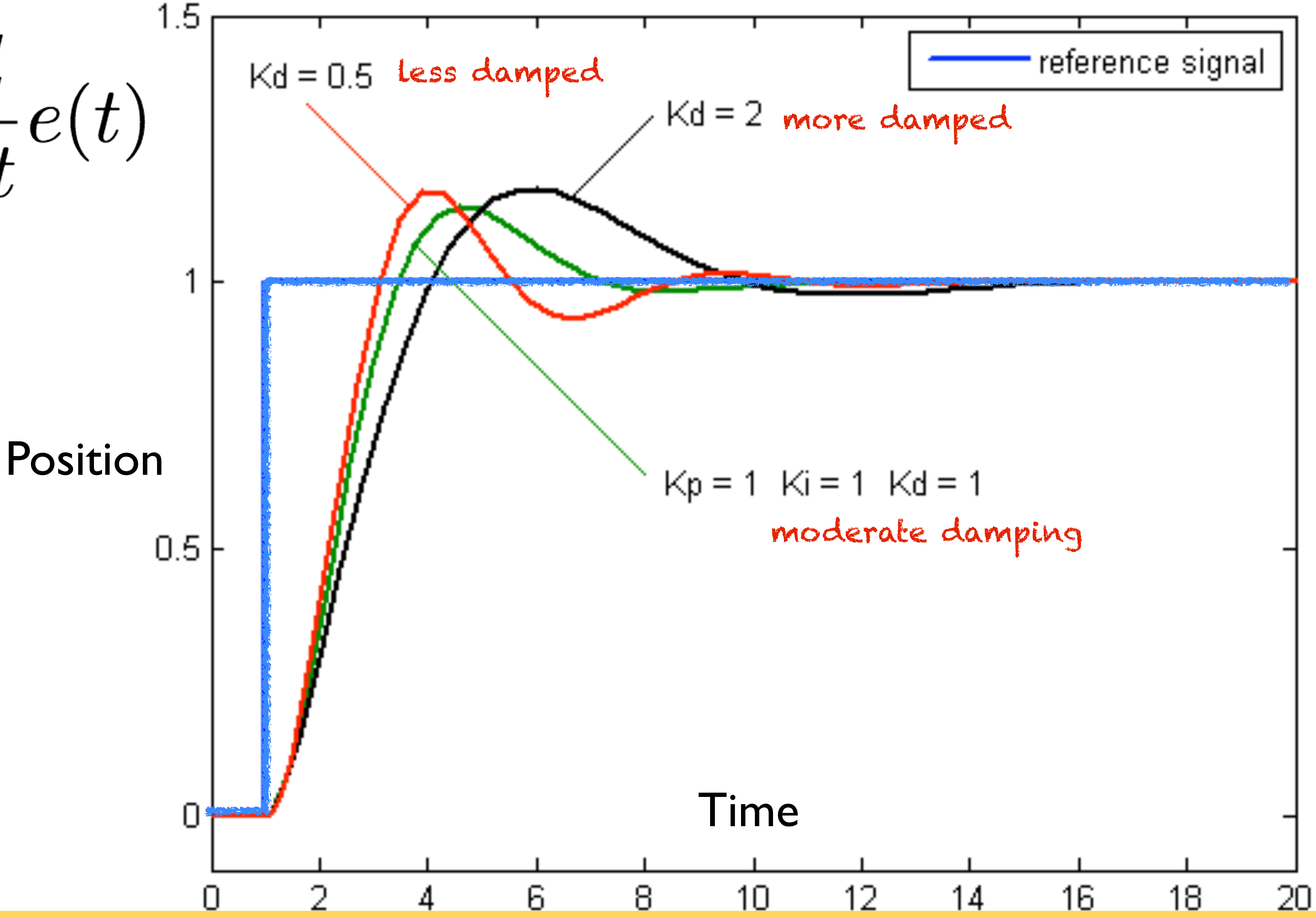


assuming constant set point, velocity is derivative of error

add damper to release energy

Slide borrowed from Michigan Robotics autorob.org

$$K_d \frac{d}{dt} e(t)$$



Slide borrowed from Michigan Robotics autorob.org

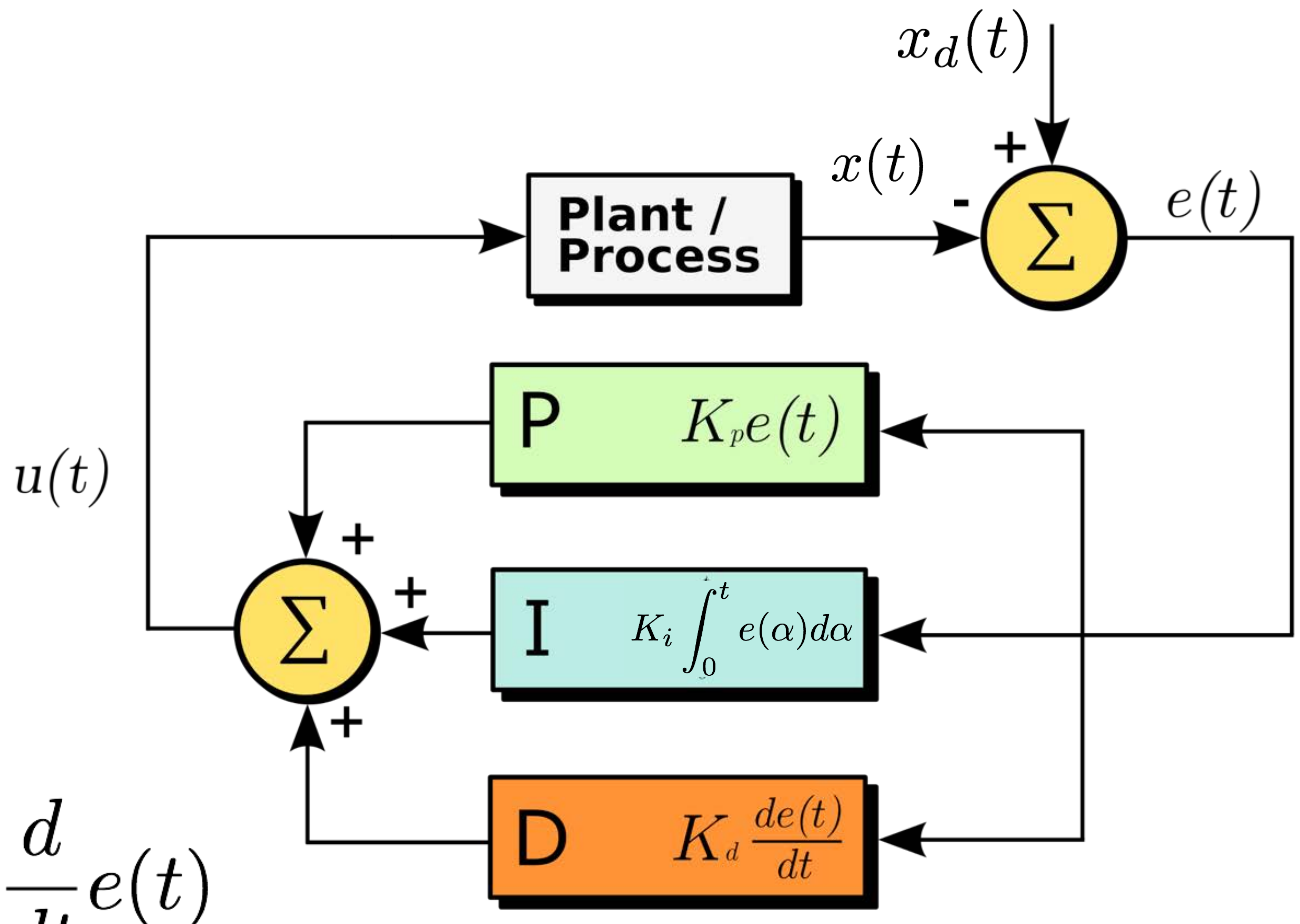
PID Control

Error signal:

$$e(t) = x_{desired}(t) - x(t)$$

Control signal:

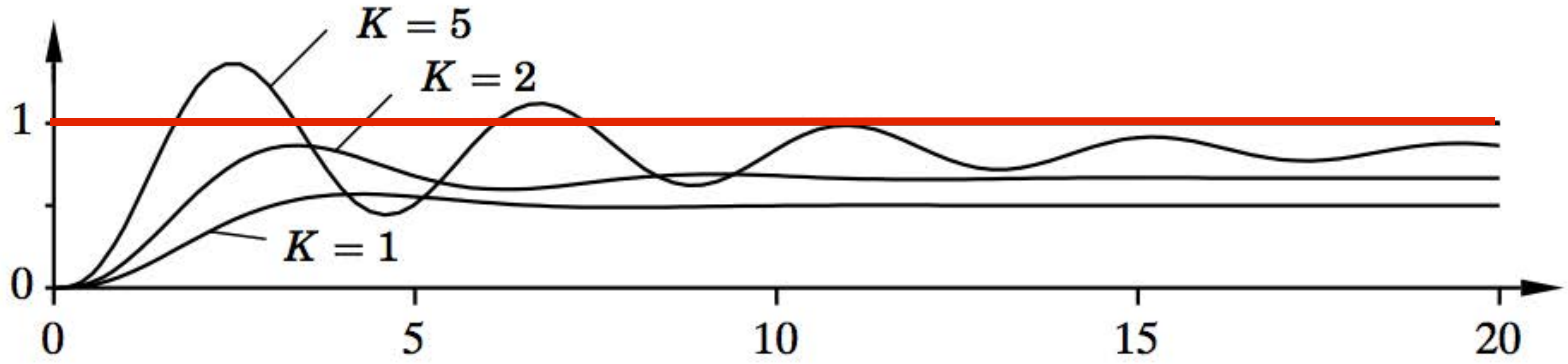
$$u(t) = K_p e(t) + K_i \int_0^t e(\alpha) d\alpha + K_d \frac{d}{dt} e(t)$$

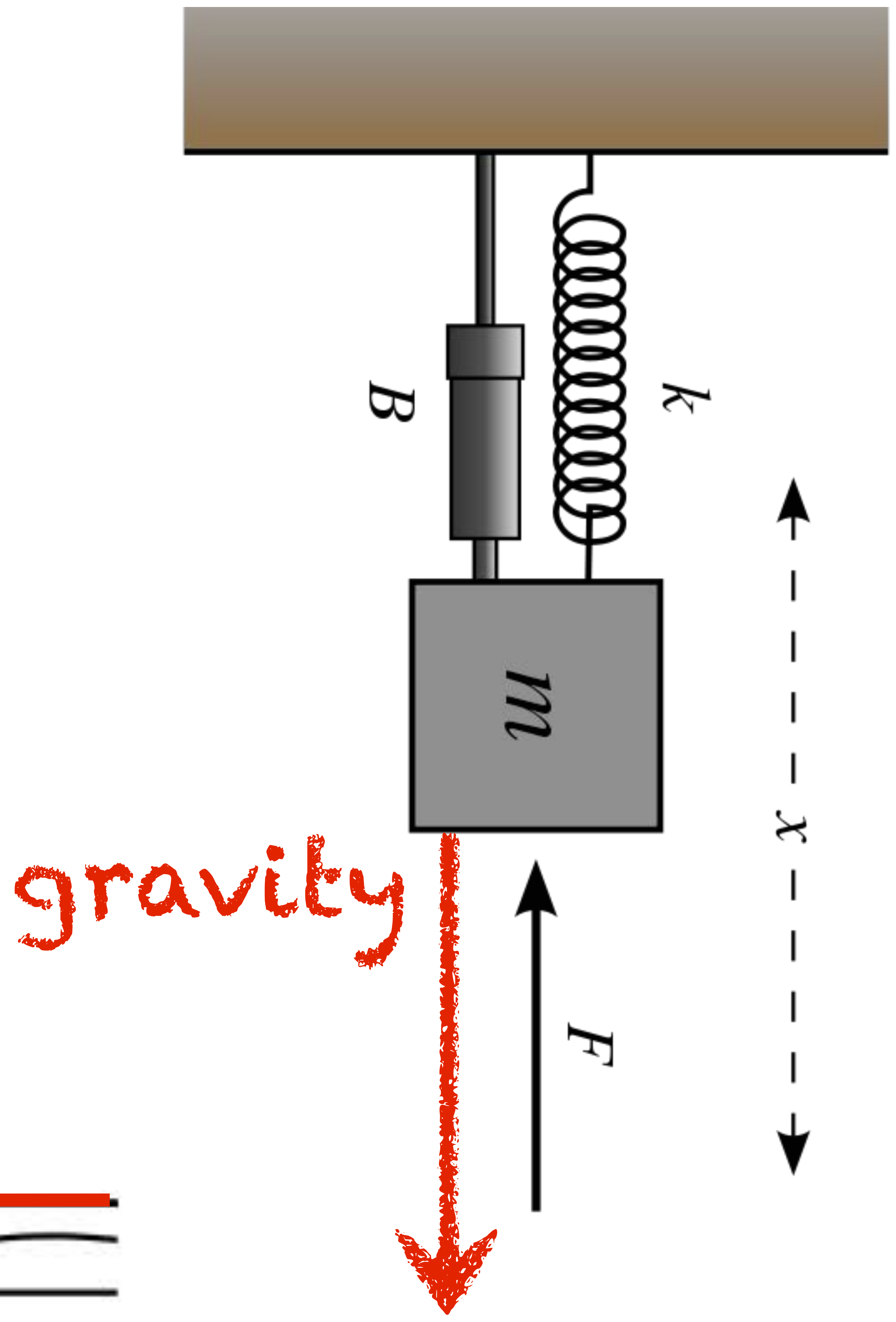


P $K_p e(t)$	I $K_i \int_0^t e(\alpha) d\alpha$	D $K_d \frac{de(t)}{dt}$
Current	Past	Future

Steady state error

- Steady state error occurs when the system rests at equilibrium before reaching desired state
- Cause could be an significant external force, weak motor, low proportional gain, etc.
- PID integral term compensates by accumulating and acting against error toward convergence

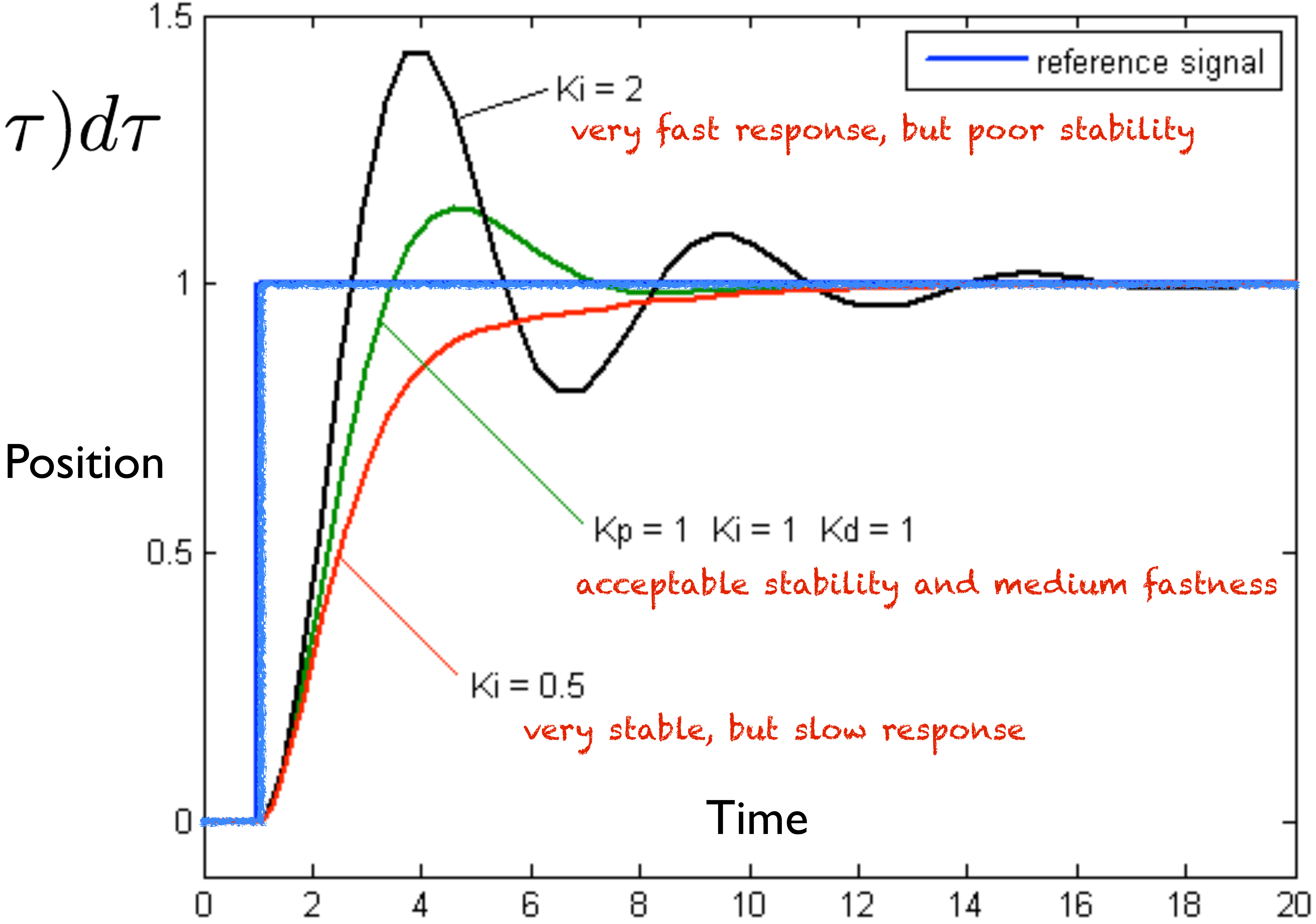




Slide borrowed from Michigan Robotics autorob.org

$$K_i \int_0^t e(\tau)d\tau$$

I $K_i \int_0^t e(\tau)d\tau$



Gain tuning

- Implementing PID algorithm will not necessarily produce a good controller
- Selection of the gains will greatly affect the performance of the controller
- PID gain tuning is more of an art than a science. Choose carefully.

$$u(t) = K_p e(t) + K_i \int_0^t e(\alpha)d\alpha + K_d \frac{d}{dt} e(t)$$

P $K_p e(t)$

I $K_i \int_0^t e(\alpha)d\alpha$

D $K_d \frac{de(t)}{dt}$

Some tips to PID tuning

(take it or leave it)

- Start all gains at zero : $K_i = K_d = K_p = 0$
- Increase spring gain K_p until system roughly meets desired state
  - overshooting and oscillation about the desired state can be expected
- Increase damping gain K_d until the system is consistently stable
  - damping stabilizes motion, but system will have steady state error
- Increase integral gain K_i until the system consistently reaches desired
- Refine gains as needed to improve performance; Test from different states

Atlas Gets a Grip | Boston Dynamics - https://www.youtube.com/watch?v=-e1_QhJ1EhQ

For inspiration on final projects

Look at projects done in the previous iteration of this class.

- Fall 23: https://rpm-lab.github.io/CSCI5551-Fall23-S2/final_presentations/
- Spring 24: https://rpm-lab.github.io/CSCI5551-Spr24/final_presentations/

Next Lecture
Mobile Robotics - I