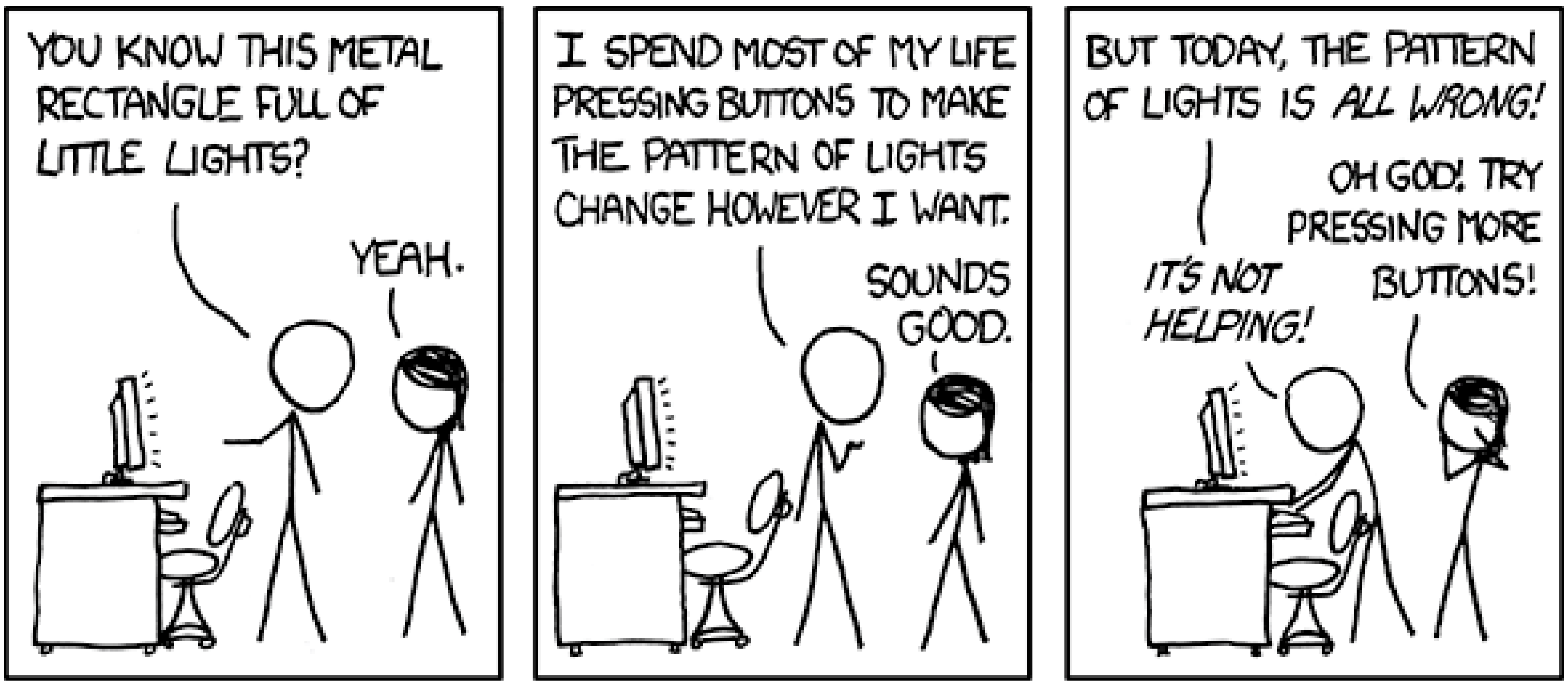
CS249r – 2019 Nuts and Bolts



What are the prerequisites for CS 249r?

1. CS 141 and/or basic computer architecture and digital design
2. CS 61/161 and/or a basic systems programming experience
3. CS 124 and/or a basic algorithms experience

We hope to have a diverse class and assume few students will have full exposure to the full breadth of topics we will cover. As such, we intend to provide some background on all of the topics. That said, students may find it helpful if they also have some background in some of the algorithms employed in autonomous systems from classes such as CS 181/182 or AM 121. Please contact the instructor or teaching fellow if you are interested in taking the course but are unsure about whether the background you have is suitable.

So how is CS249r actually going to run?

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

So how is CS249r actually going to run?

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

We will provide high level background lectures to get everyone up to speed on the relevant topics from both Autonomous Systems / Robotics and Computer Systems / Architecture

So how is CS249r actually going to run?

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

We will provide high level background lectures to get everyone up to speed on the relevant topics from both Autonomous Systems / Robotics and Computer Systems / Architecture

Class on 9/11 will be video taped (but not posted anywhere) as I am doing a Bok Center teaching review. We will have a "no camera" section as well.

So how is CS249r actually going to run?

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

We are also going to have a day of sample presentations to provide a guide for the types of presentations we hope you will give on your research papers throughout the semester and on your final projects

So how is CS249r actually going to run?

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

2 students per class will present on selected papers organized by topic

So how is CS249r actually going to run?

We have posted a tentative paper list to Canvas (along with PDFs and links)

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

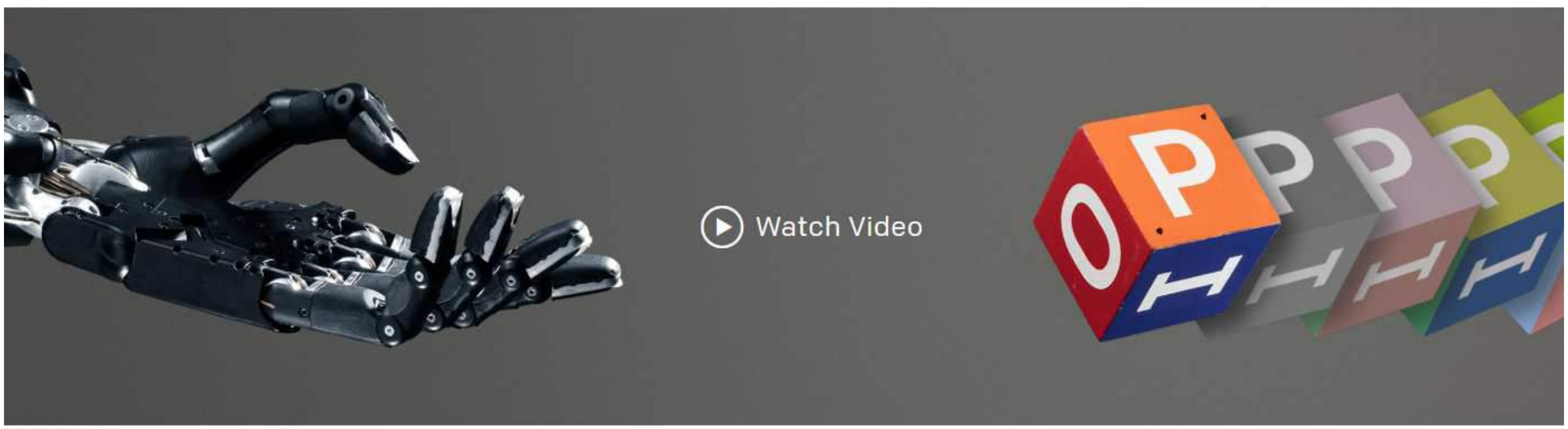
2 students per class will present on selected papers organized by topic

So how is CS249r actually going to run?

JULY 30, 2018 • 9 MINUTE READ

Learning Dexterity

We've trained a human-like robot hand to manipulate physical objects with unprecedented dexterity.

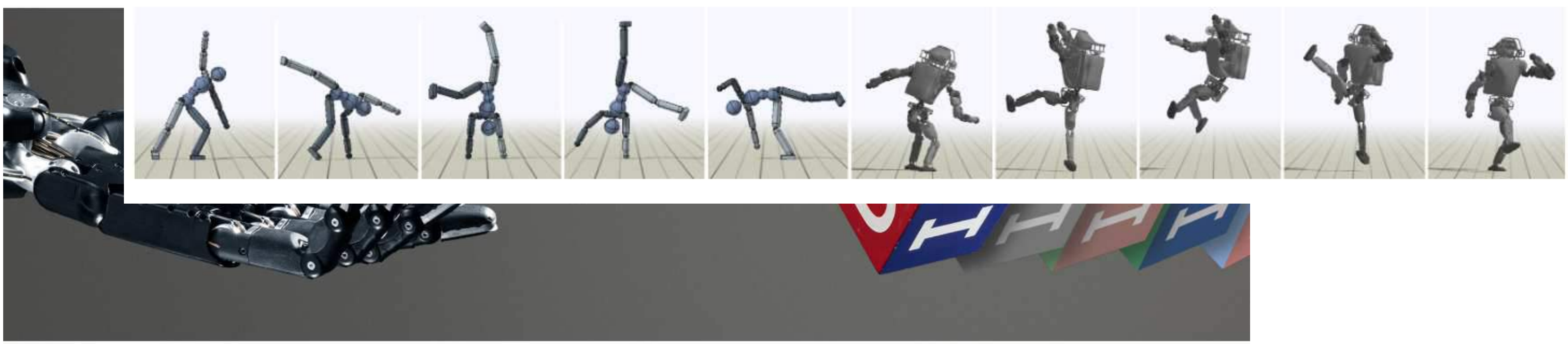


So how is CS249r actually going to run?

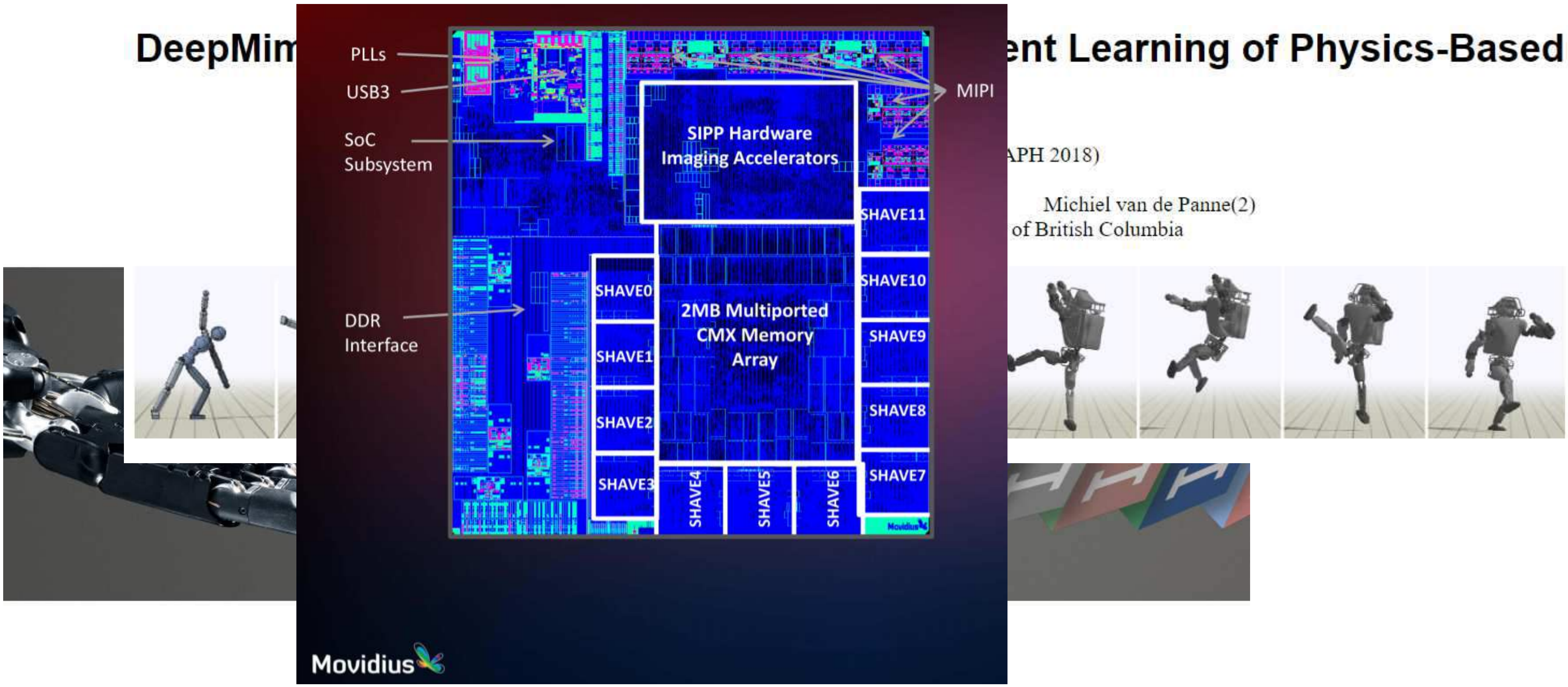
DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills

Transactions on Graphics (Proc. ACM SIGGRAPH 2018)

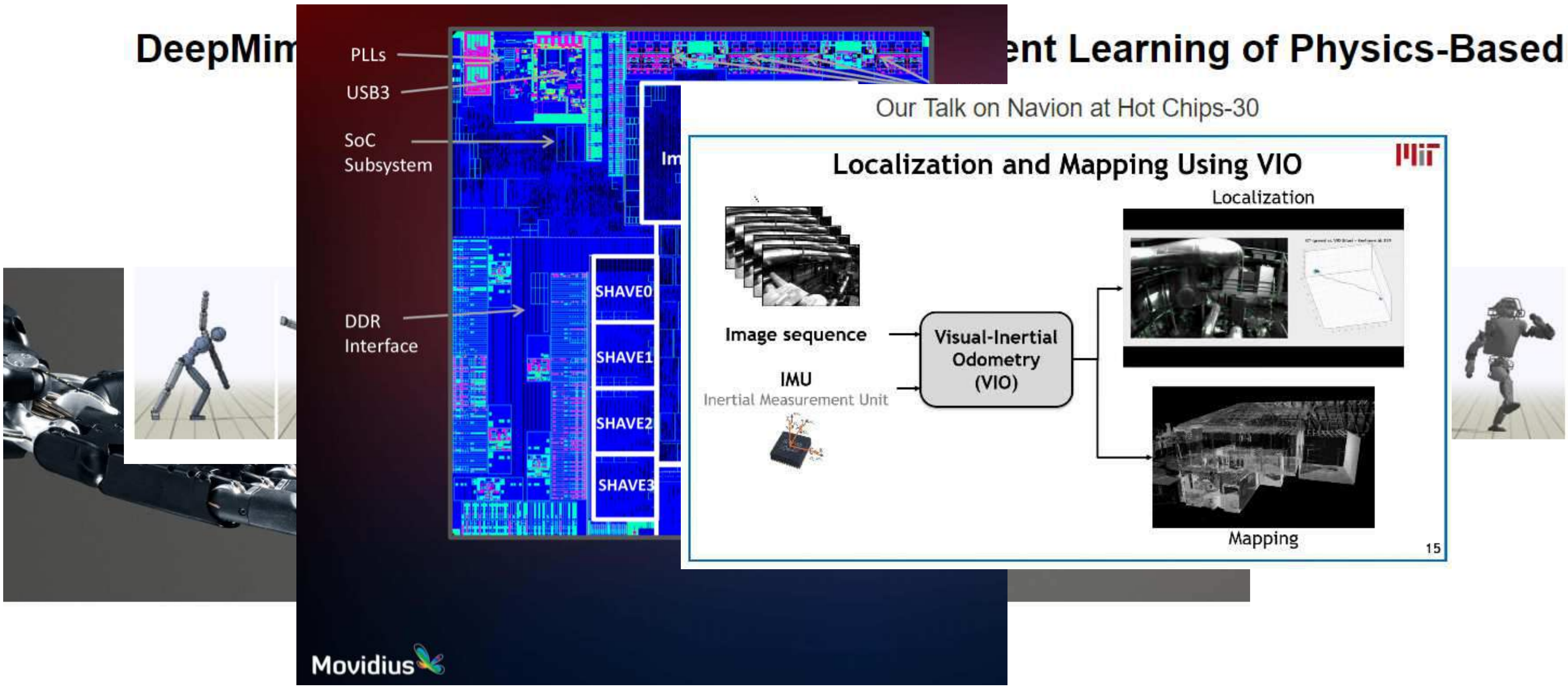
Xue Bin Peng(1) Pieter Abbeel(1) Sergey Levine(1) Michiel van de Panne(2)
(1)University of California, Berkeley (2)University of British Columbia



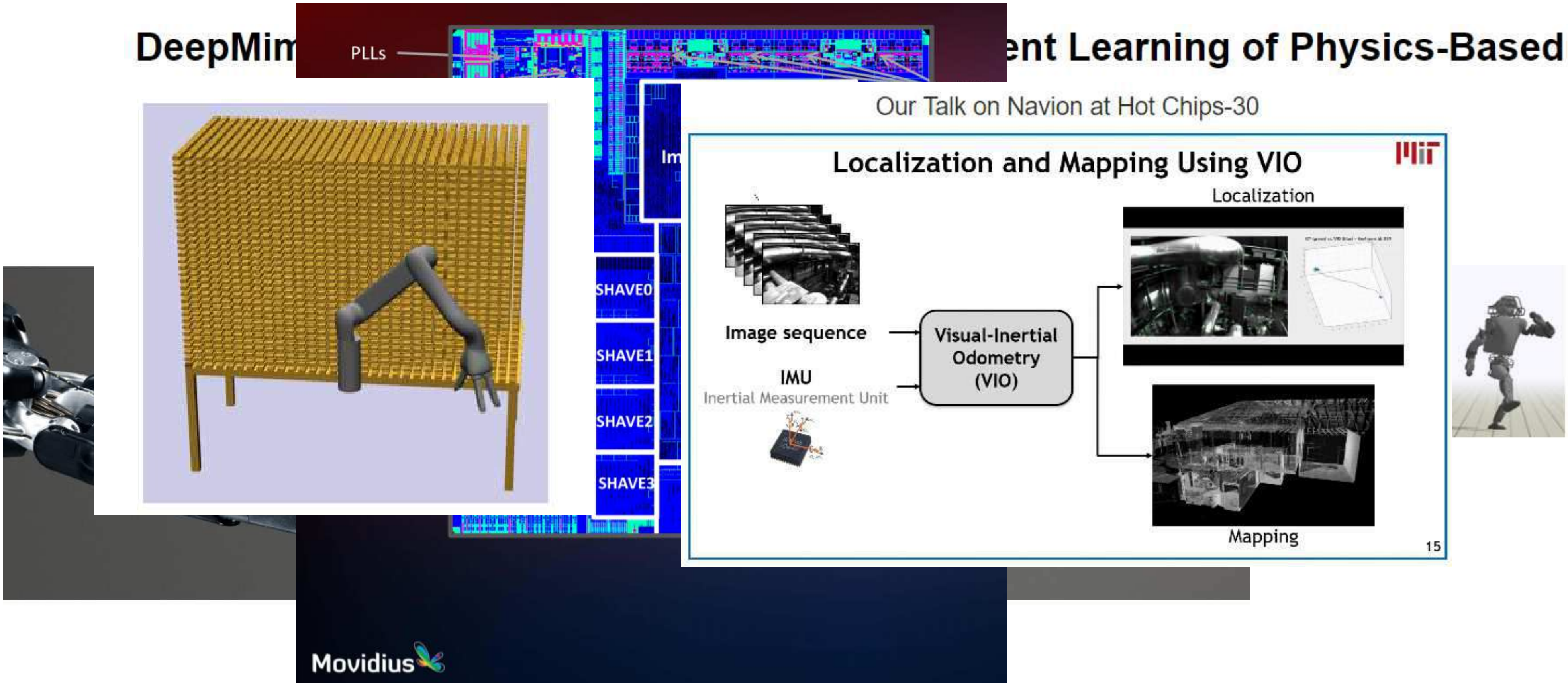
So how is CS249r actually going to run?



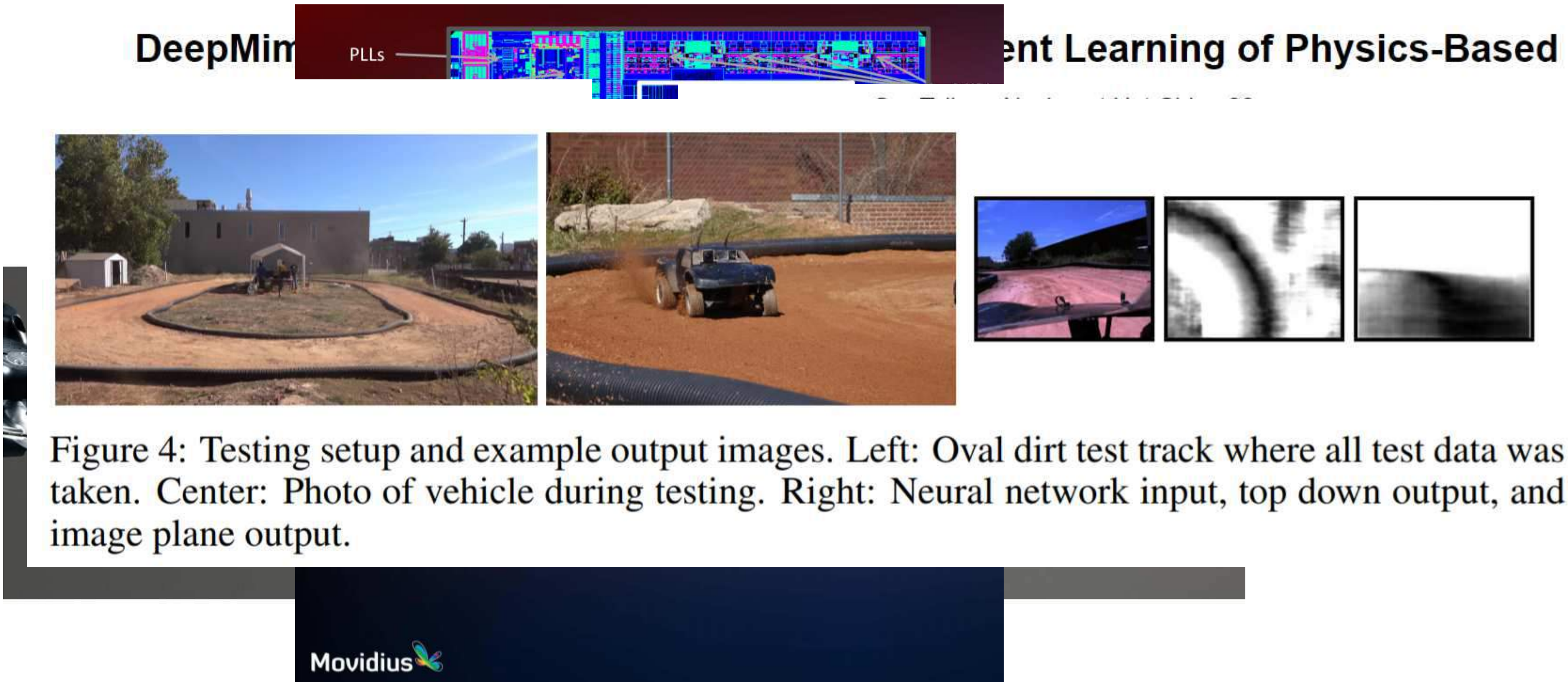
So how is CS249r actually going to run?



So how is CS249r actually going to run?



So how is CS249r actually going to run?



So how is CS249r actually going to run?

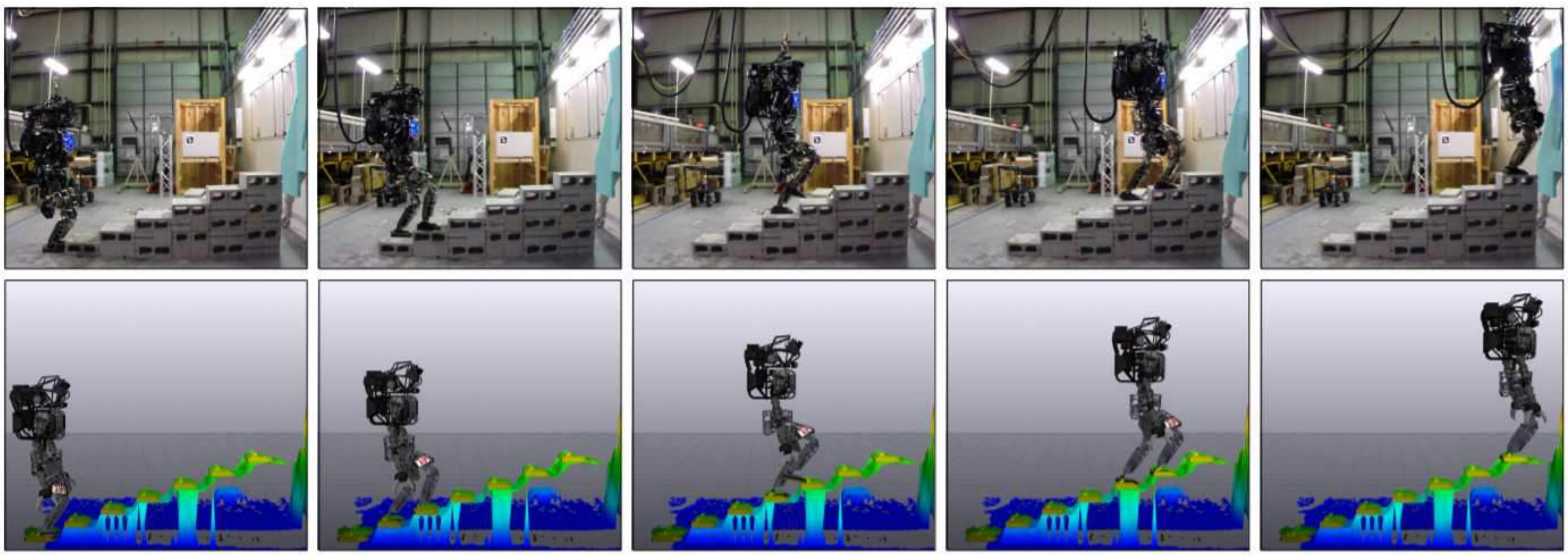


Fig. 12 Atlas walking continuously up six cinder block steps using LIDAR-based state estimation in a closed loop with the walking controller. Top: images of the robot climbing the stack of cinder blocks in our laboratory. Bottom: the state estimate rendering in our user interface.

So how is CS249r actually going to run?

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

We have posted a tentative paper list to Canvas (along with PDFs and links)

Start to think about which papers you want as we will be allocating them in a week or two!

2 students per class will present on selected papers organized by topic

So how is CS249r actually going to run?

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

We have posted a tentative paper list to Canvas (along with PDFs and links)

Start to think about which papers you want as we will be allocating them in a week or two!

If you have an idea for a paper not on the list please run it by us and we may be willing to swap it in!

2 students per class will presentations on selected papers organized by topic

So how is CS249r actually going to run?

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

We will simulate the conference review process in the middle of the term to give students insight into how papers are judged and thus accepted or rejected

We will discuss the reviews of an accepted paper during the example paper presentations

So how is CS249r actually going to run?

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

We will simulate the conference review process in the middle of the term to give students insight into how papers are judged and thus accepted or rejected

We will discuss the reviews of an accepted paper during the example paper presentations

You'll actually get to see the submitted version and final version of one of my papers with the actual reviews

So how is CS249r actually going to run?

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

Finally we wrap up the semester with a lot of time to work on and then present final projects.

Note the mid semester project proposal due date!

How do you get an A in CS 249r?

1. Paper Reviews – 20%
2. Paper Presentation – 20%
3. Class Participation – 10%
4. Final Project – 50%

Paper Reviews – 20%

Goals:

1. To develop the skill of reading papers and quickly taking away the big picture ideas.

Assignments:

1. Submit a short “review” on each research paper read during the course (and submit the review 36 hours BEFORE the class in which it is presented)

Paper Reviews – 20%

We will use HOTCRP (the standard submission system from Computer Architecture Conferences)

Goals:

1. To develop the skill of reading papers and quickly taking away the big picture ideas.

Assignments:

1. Submit a short "review" on each research paper read during the course (and submit the review 36 hours BEFORE the class in which it is presented)

Paper Reviews – 20%

Goals:

1. To develop the skill of reading papers and quickly taking away the big picture ideas.
2. **Crowdsource a best practice guide on writing papers**

Assignments:

1. Submit a short "review" on each research paper read during the course (and submit the review 36 hours BEFORE the class in which it is presented)

Paper Presentation(s) – 20%

Goals:

1. To develop the skill of understanding a paper in detail
2. Practice presenting a (conference) paper to audience and teaching a concept to a class

Assignments:

1. Give at least one 18 minute presentation on a research paper followed by 10 minutes of Q&A (and meet with the course staff a week prior to your presentation)

Paper Presentation(s) – 20%

Goals:

1. To develop the skill of understanding a paper in detail
2. Practice presenting a (conference) paper to audience and teaching a concept to a class

Assignments:

1. Give at least one 18 minute presentation on a research paper followed by 10 minutes of Q&A (and meet with the course staff a week prior to your presentation)

- **~5 minutes of setup** (What is the problem? Why is it important? What are the key challenges?)
- **~5 minutes of contribution** (What did the author(s) do? Why was it novel?)
- **~8 minutes of context** (What work did it build on /how does it compare?)

Class Participation – 10%

Goals:

1. Practice absorbing a (conference) paper presentation
2. To give feedback to presenters

Assignments:

1. Be an active participant in class
2. Submit anonymous feedback on each presentation

Final Project – 50%

Goals:

1. Practice being a graduate student:
   a) Coming up with a research idea
   b) Workshopping the idea with others / advisors
   c) Collaboratively conducting the research
   d) Writing up a (conference) paper in Latex
   e) Giving a presentation on the paper

Assignments:

1. Work in teams of 2-3 students to submit a project proposal midway through the semester and a final project report at the end of the semester as well as presenting that research to the class

Final Project – 50%

We would love to find a way to incorporate your research into your final project

Goals:

1. Practice being a graduate student:
   a) Coming up with a research idea
   b) Workshopping the idea with others / advisors
   c) Collaboratively conducting the research
   d) Writing up a (conference) paper in Latex
   e) Giving a presentation on the paper

Assignments:

1. Work in teams of 2-3 students to submit a project proposal midway through the semester and a final project report at the end of the semester as well as presenting that research to the class

Any questions?

Quick survey of all of you

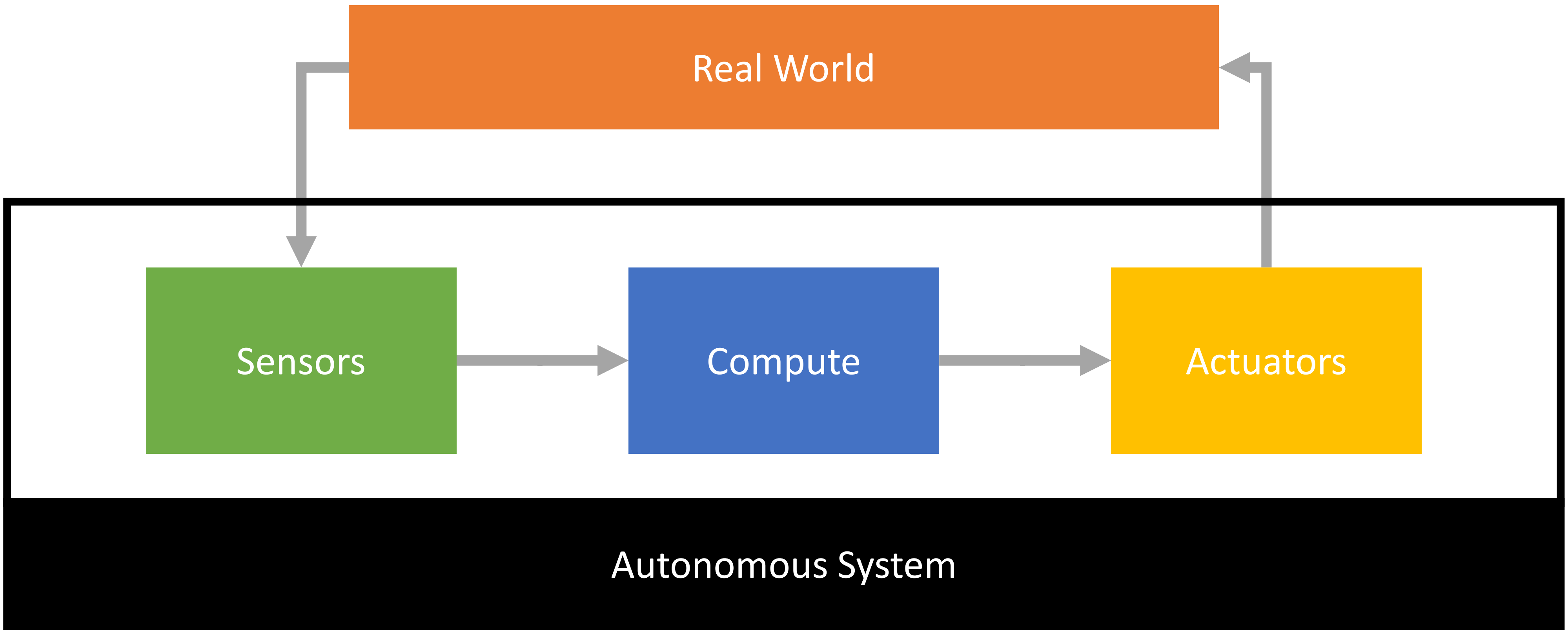
Undergrads vs Grads

Definitely vs Maybe Enrolling

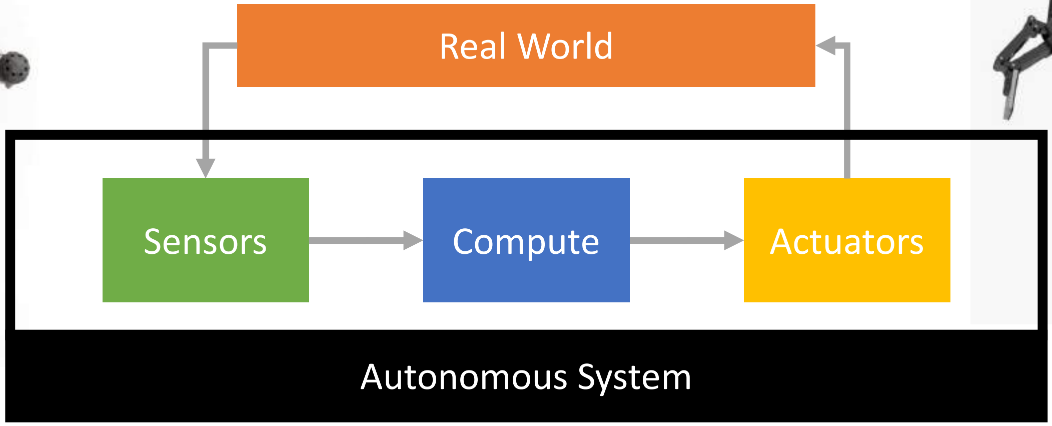
Architecture vs. Robotics / Autonomous Systems vs. Neither

Ok so lets dive into a little material for next week!

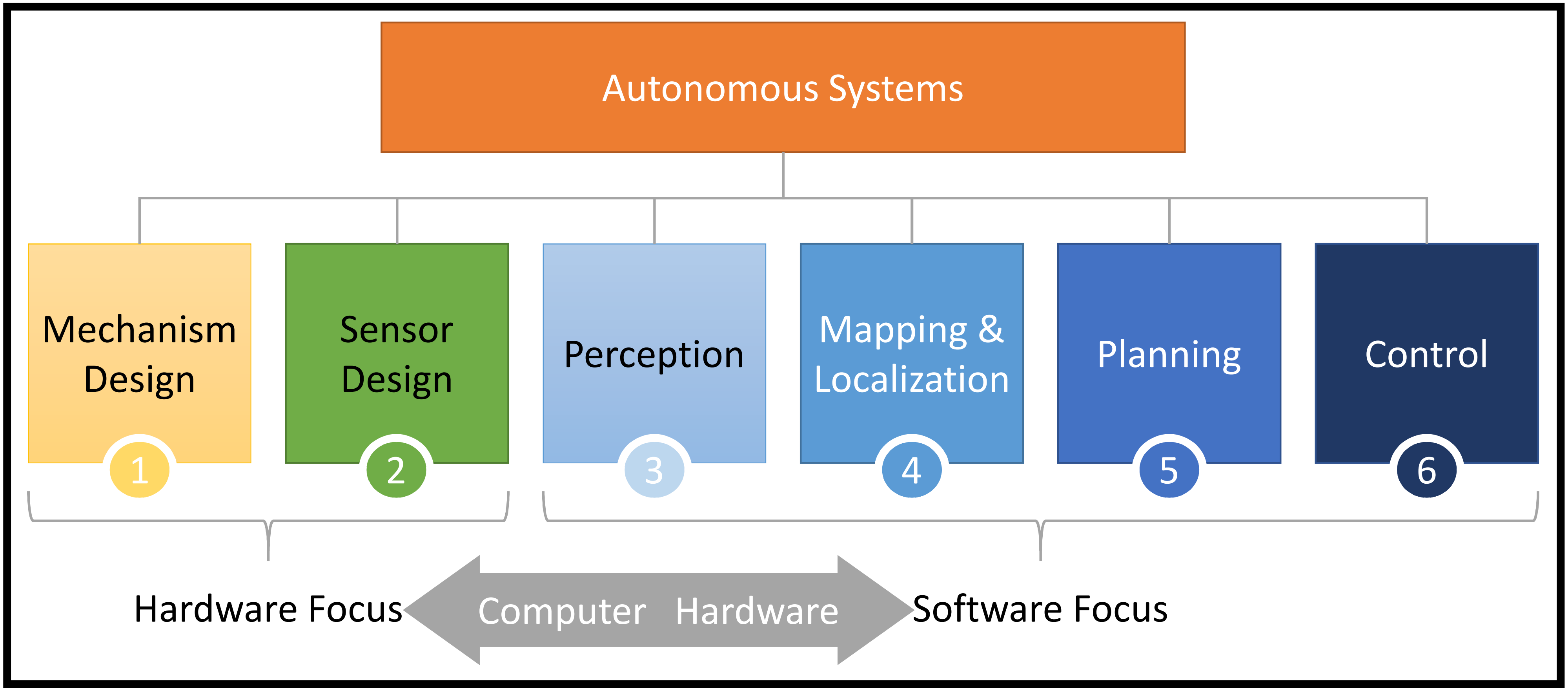
What do we mean by an Autonomous System?



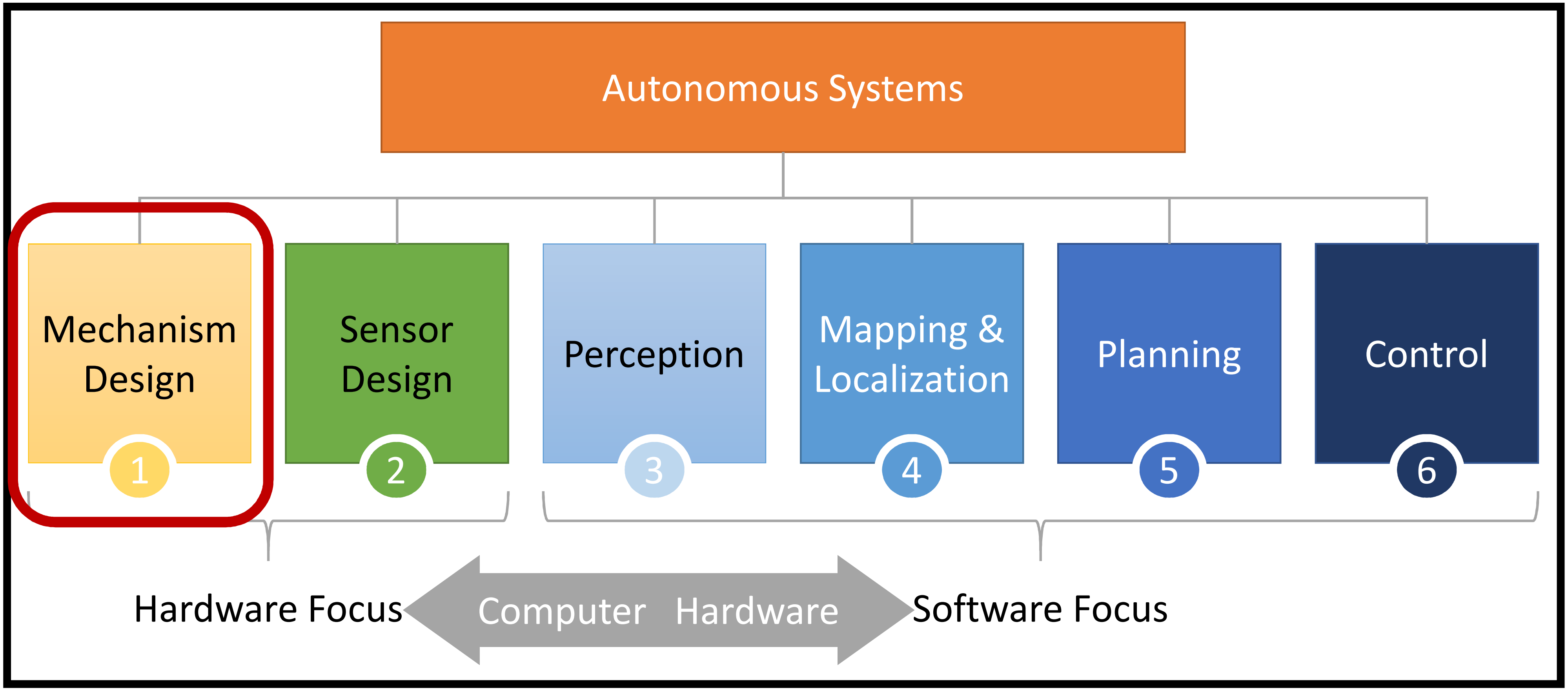
What do we mean by an Autonomous System?



Autonomous Systems / Robotics is a BIG space



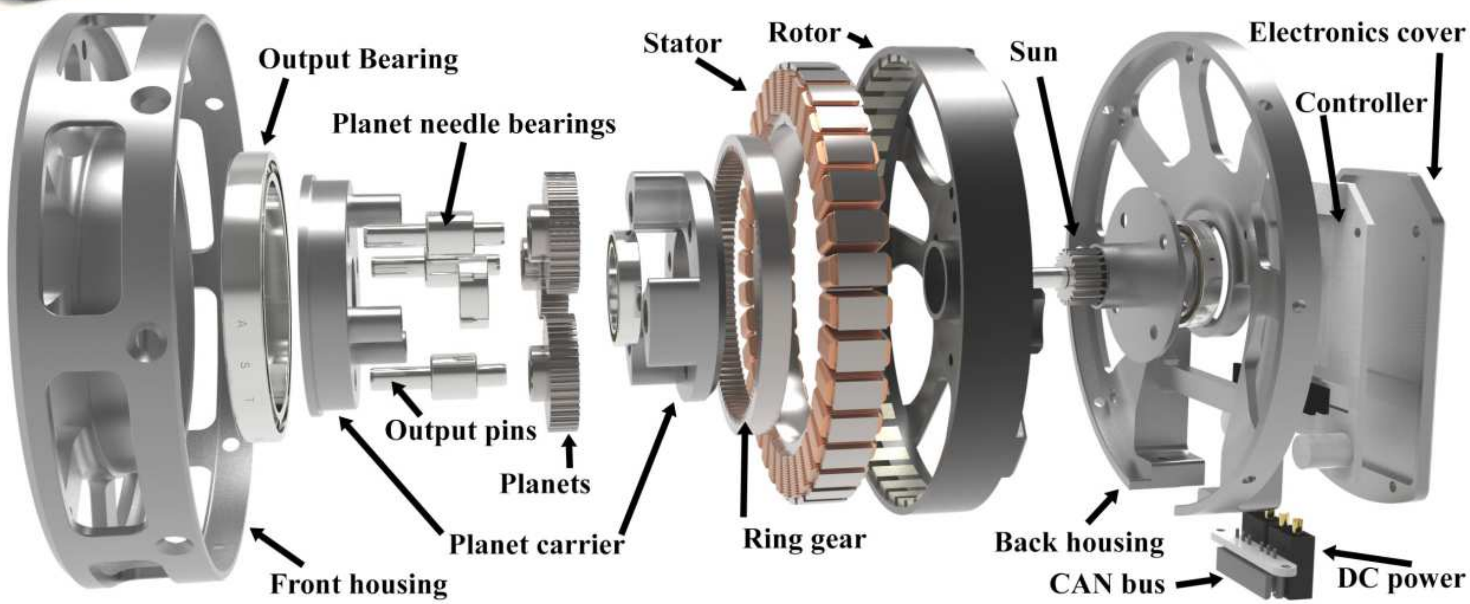
Autonomous Systems / Robotics is a BIG space



1 Mechanism designers create new robots and actuators

Fig. 4: The modular actuator used in the Mini Cheetah. Motor, planetary gear set, and control electronics are all built-in.

Fig. 5: Exploded view of the actuator.



Katz, Di Carlo and Kim ICRA 2019

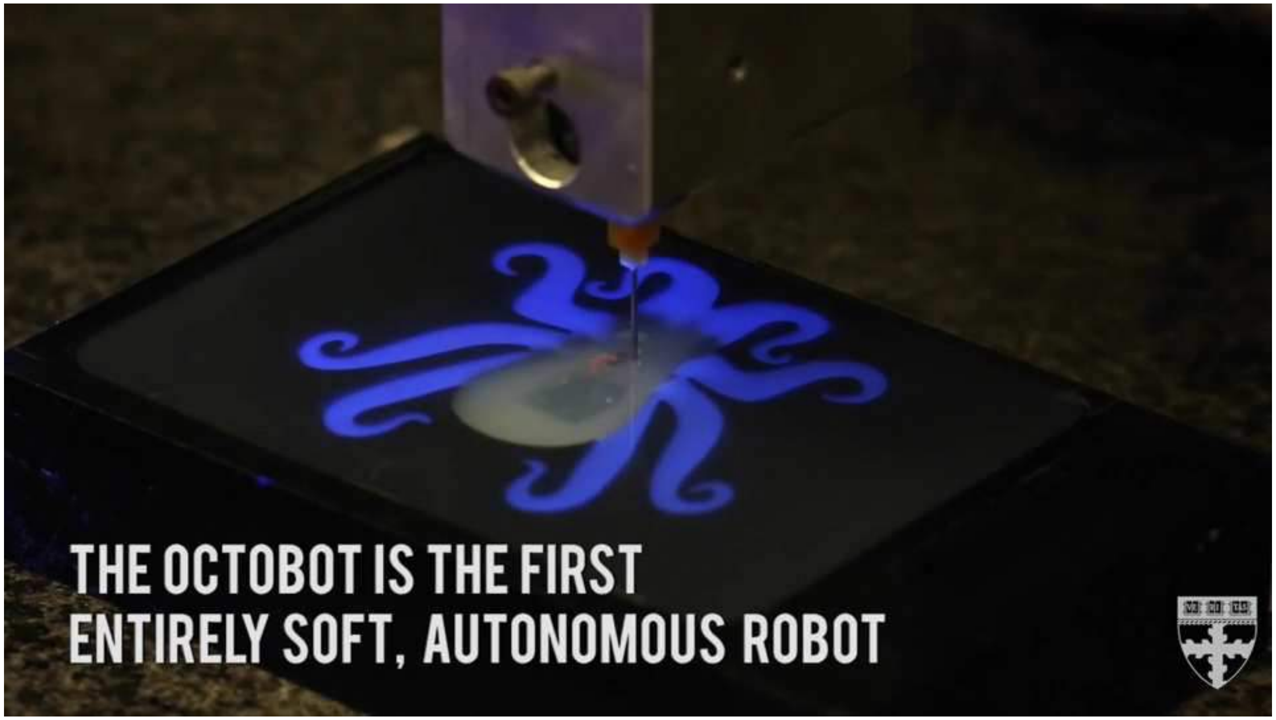
1 Mechanism designers create new robots and actuators

MIT 2.74

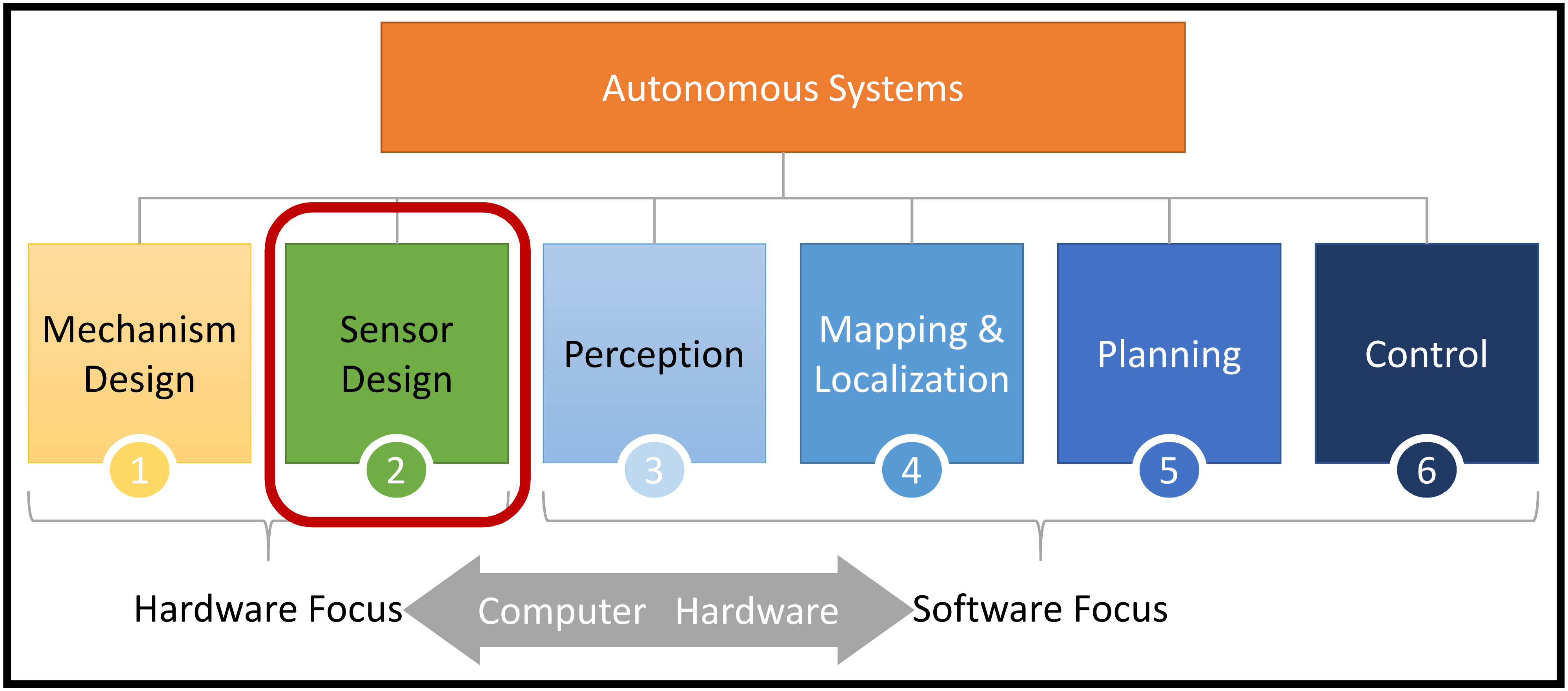
1 Mechanism designers create new robots and actuators

1 Mechanism designers create new robots and actuators

WYSS INSTITUTE



Autonomous Systems / Robotics is a BIG space



2

Sensor designers try to find new ways to collect data about the world around the autonomous system

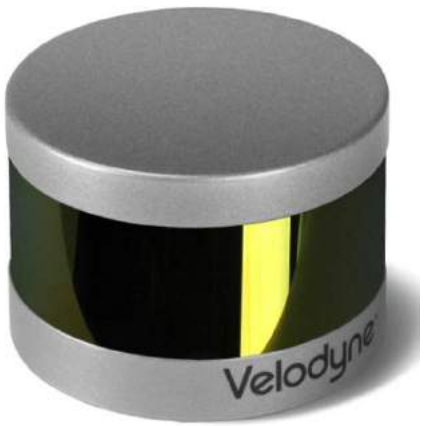
MEMs IMUs / Gyroscopes

Motor Encoders

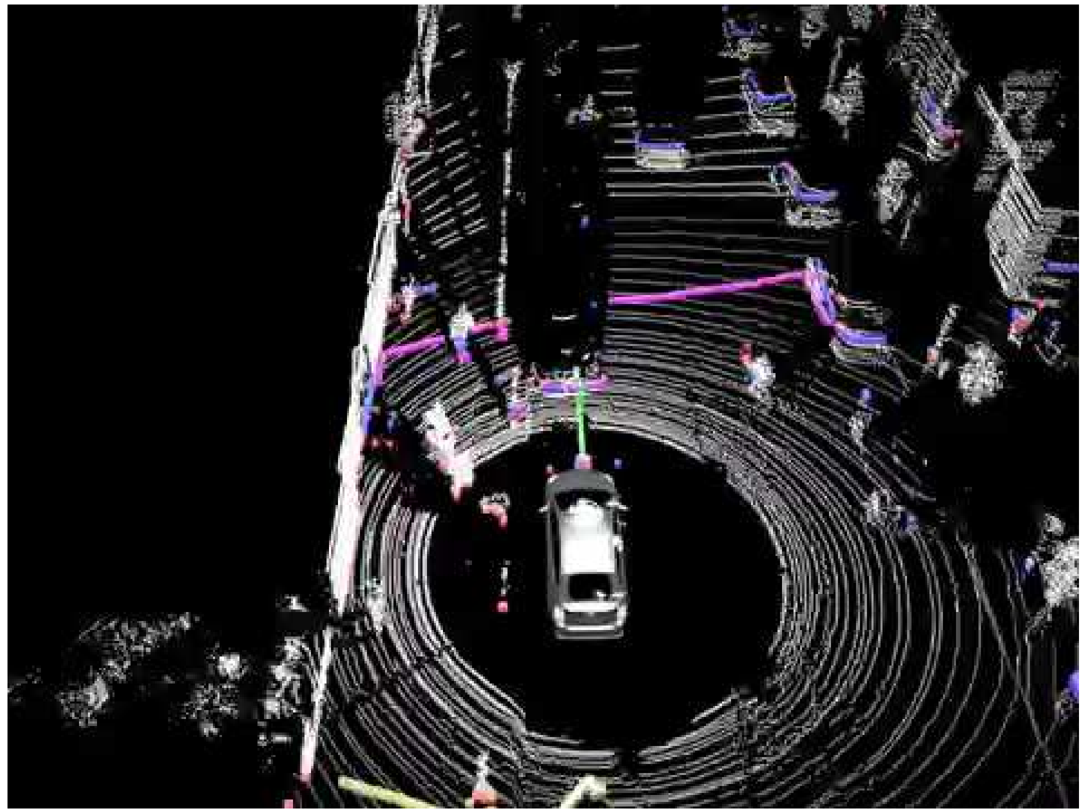
2

Sensor designers try to find new ways to collect data about the world around the autonomous system

Structured Light (e.g., LIDAR)



(and other Structured Waves e.g., Sonar, RADAR, etc.)



❷ Sensor designers try to find new ways to collect data about the world around the autonomous system

Unstructured Light (aka Cameras)

2

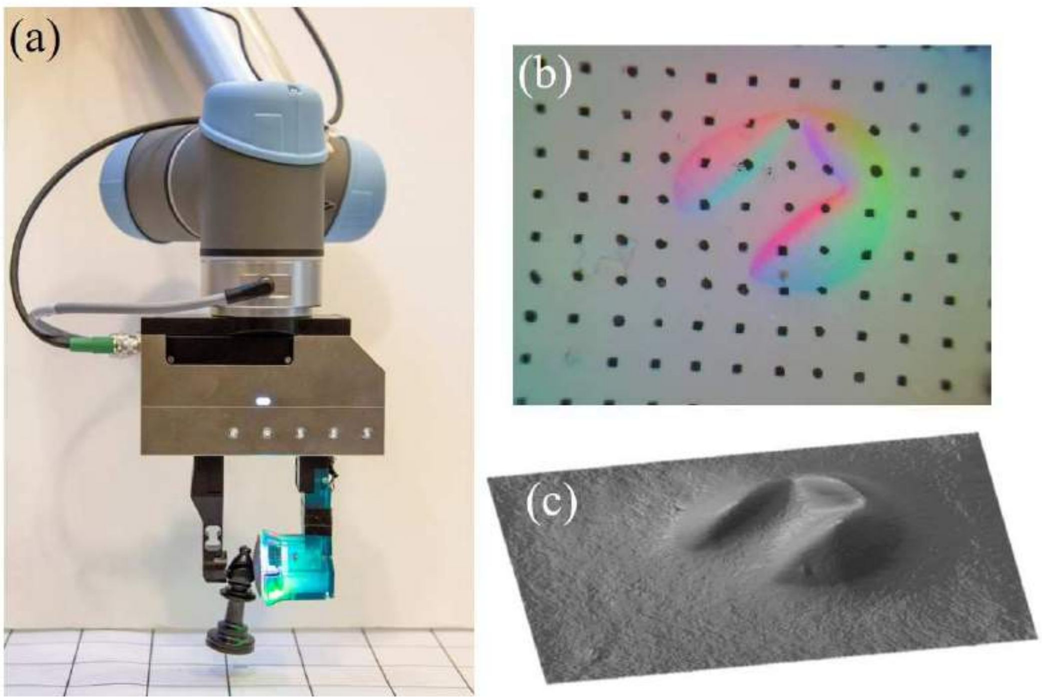
Sensor designers try to find new ways to collect data about the world around the autonomous system

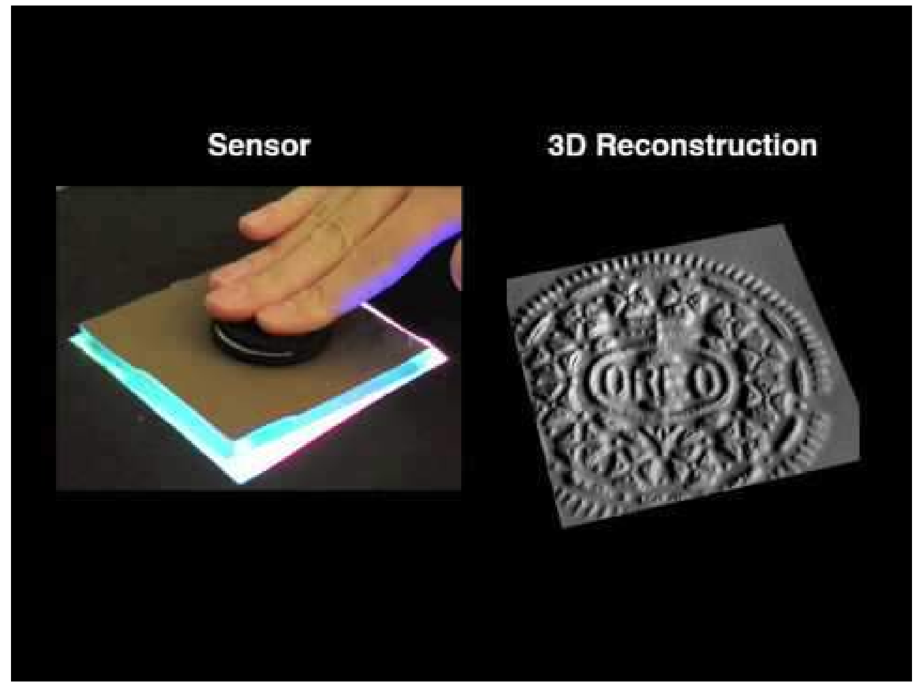
Unstructured Light (aka Cameras)

Computer Vision (we'll talk about this later)

Usable Data

2 Sensor designers try to find new ways to collect data about the world around the autonomous system





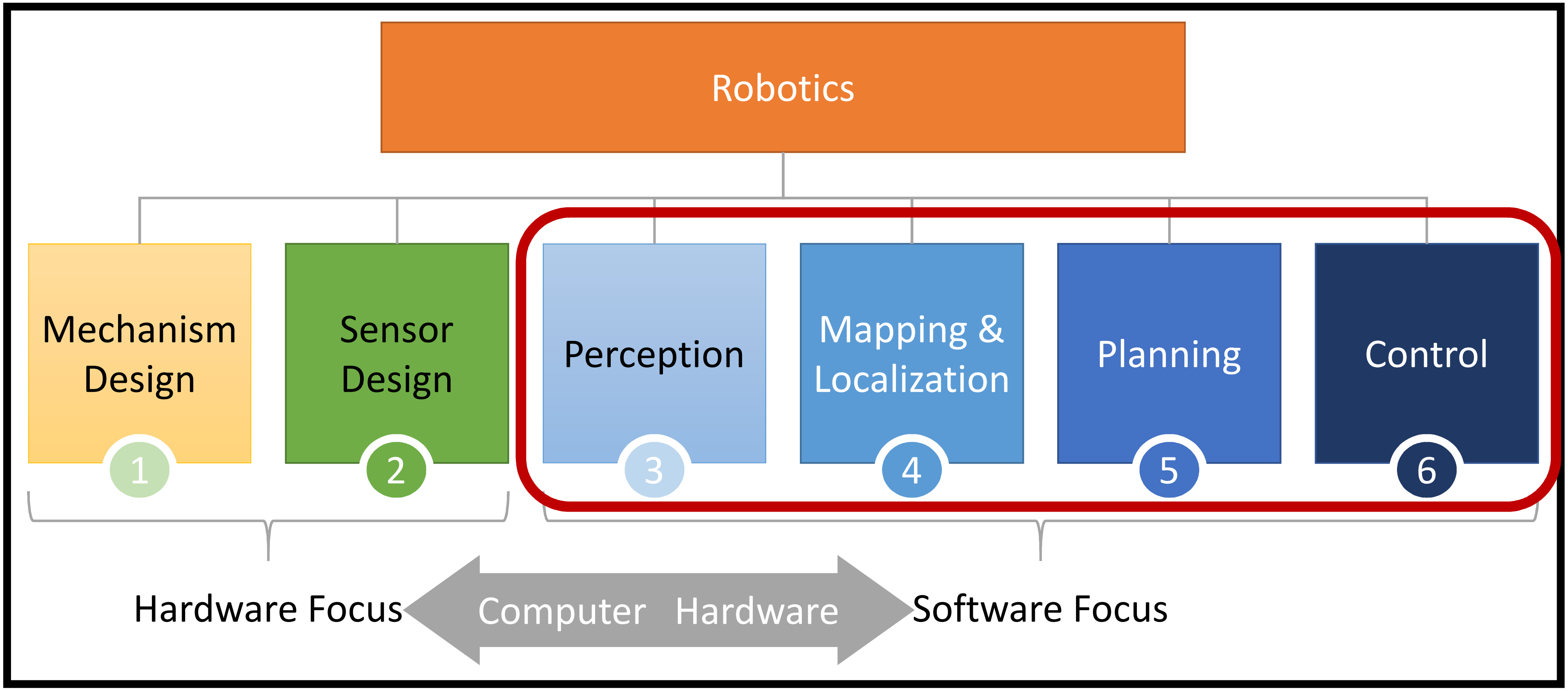
http://www.gelsight.com/

1 2 Key Takeaways:

1. Different kinds of systems will have different power, weight, and performance budgets for computer hardware and requirements for control algorithms
2. Understanding the sensors on your system will help you understand at what rate you can get information and the bandwidth of the information you will need to process
3. Different kinds of sensors will require different amounts of onboard compute to process the information

Our topic for next week – Compute!

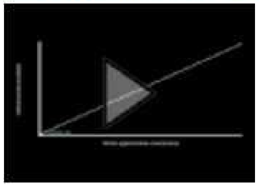
Autonomous Systems / Robotics is a BIG space



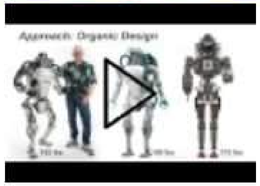
Your homework for next week 1/2

Pre-Reads for Intro to Robotics (Perception and Mapping)

[Chris Urmson: How a driverless car sees the road]



[Meet Spot, the robot dog that can run, hop and open doors | Marc Raibert]



Your homework for next week 2/2

Pre-Reads for Intro to Robotics (Planning and Control)

Computer Architeecture to Close the Loop in Real-time Optimization: https://ieeexplore.ieee.org/document/7402937

The Architectural Implications of Autonomous Driving: Constraints and Acceleration: https://web.eecs.umich.edu/~shihclin/papers/AutonomousCar-ASPLOS18.pdf

A Summary of Team MIT's Approach to the Virtual Robotics Challenge: https://agile.seas.harvard.edu/files/agile/files/vrc_entry.pdf

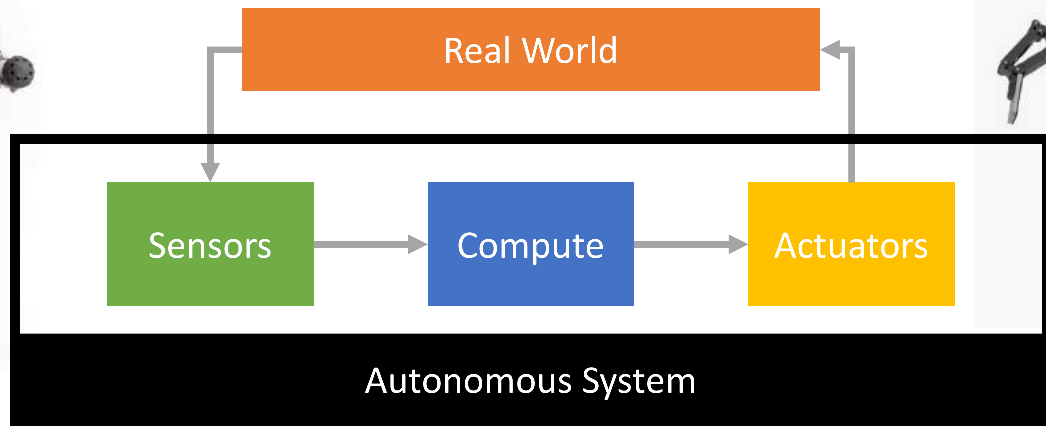
And finally some fun robot videos

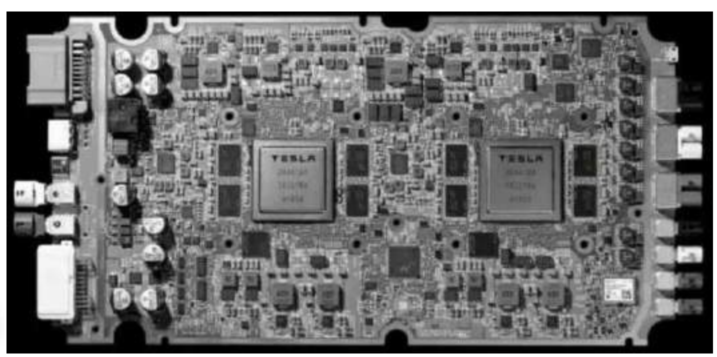
CS 249r: Special Topics in Edge Computing

Intro to Autonomous Systems / Robotics Part 1

Brian Plancher
Fall 2019

What do we mean by an Autonomous System?





So how is CS249r actually going to run?

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

Background Lectures

Reading / Presenting Papers

Final Project

So how is CS249r actually going to run?

FYI the exact dates of the first couple weeks are moving around a little bit

Date	Module	Class Type	Topic	Notes
Wed, Sep 4	Introduction	Lecture	Course Introduction, Overview, and Nuts and Bolts	
Mon, Sep 9	Motivation	Lecture	Intro to Robotics (Perception and Mapping)	
Wed, Sep 11		Lecture	Intro to Robotics (Planning and Control)	
Mon, Sep 16		Lecture	Intro to Domain Specific Architectures	
Wed, Sep 18	Sample Presentations	Research Paper(s)	Example Research Paper Presentations	
Mon, Sep 23	Domain Specific Accelerators	Research Paper(s)	Domain Specific Accelerators	
Wed, Sep 25		Research Paper(s)	Domain Specific Accelerators	
Mon, Sep 30	ML Motivation	Guest Lecture	Reinforcement Learning 101	Tentative
Wed, Oct 2		Guest Lecture	Deep Reinforcement Learning 101	Tentative
Mon, Oct 7		No Class	Columbus Day	
Wed, Oct 9	E2E Control	Research Paper(s)	E2E Control	
Mon, Oct 14		Research Paper(s)	E2E Control	
Wed, Oct 16		Research Paper(s)	E2E Control	
Mon, Oct 21		Research Paper(s)	E2E Control	
Wed, Oct 23	Conference Paper Review	Conference Paper Review	Simulated Conference Paper Review Meeting	
Mon, Oct 28	Perception / Mapping	Research Paper(s)	Perception / Mapping	Project Proposals Due
Wed, Oct 30		Research Paper(s)	Perception / Mapping	
Mon, Nov 4		Research Paper(s)	Perception / Mapping	
Wed, Nov 6		Research Paper(s)	Perception / Mapping	
Mon, Nov 11	Planning / Control	Research Paper(s)	Planning / Control	
Wed, Nov 13		Research Paper(s)	Planning / Control	
Mon, Nov 18		Research Paper(s)	Planning / Control	
Wed, Nov 20		Research Paper(s)	Planning / Control	
Mon, Nov 25	Final Project	No Class	Thanksgiving	
Wed, Nov 27		No Class	Thanksgiving	
Mon, Dec 2		Final Class	Wrap Up / Project Check-Ins / Office Hours in Class	
Wed, Dec 4		No Class	Reading period	
Mon, Dec 9		Project Presentations	Project presentations	Project Reports Due

Background Lectures

Reading / Presenting Papers

Final Project

How do you get an A in CS 249r?

1. Paper Reviews – 20%
2. Paper Presentation – 20%
3. Class Participation – 10%
4. Final Project – 50%

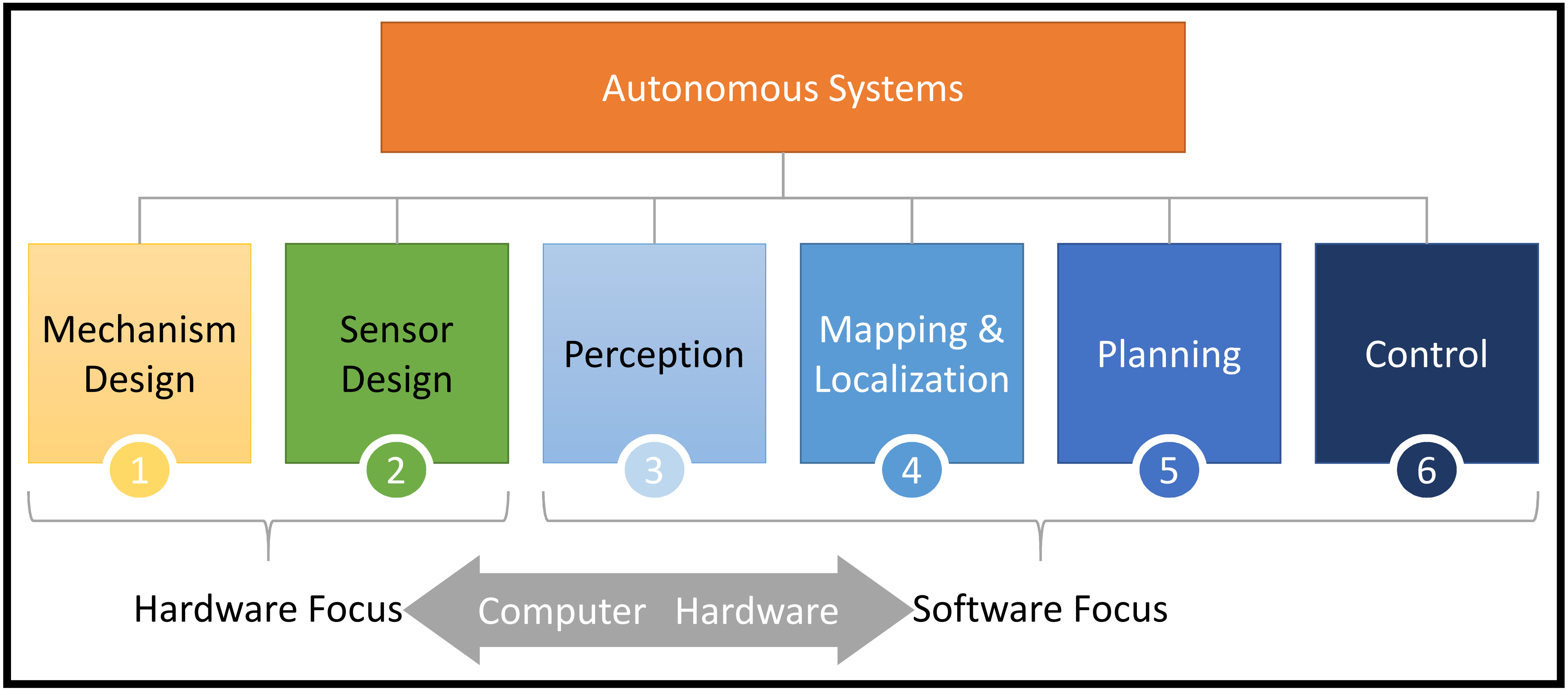
What are the prerequisites for CS 249r?

1. CS 141 and/or basic computer architecture and digital design
2. CS 61/161 and/or a basic systems programming experience
3. CS 124 and/or a basic algorithms experience

We hope to have a diverse class and assume few students will have full exposure to the full breadth of topics we will cover. As such, we intend to provide some background on all of the topics. That said, students may find it helpful if they also have some background in some of the algorithms employed in autonomous systems from classes such as CS 181/182 or AM 121. Please contact the instructor or teaching fellow if you are interested in taking the course but are unsure about whether the background you have is suitable.

Any quick nuts and bolts questions?

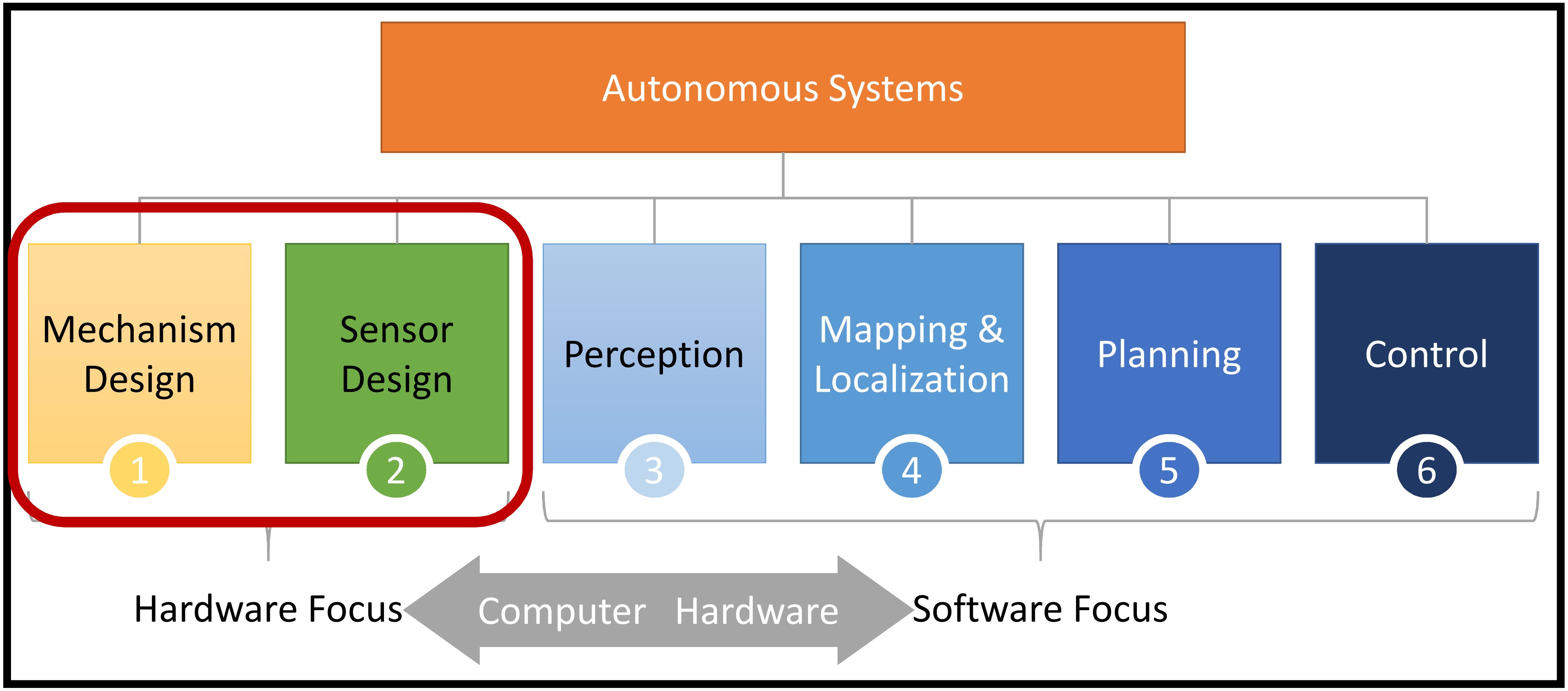
Autonomous Systems / Robotics is a BIG space



The goal for the next couple of lectures is to develop a high level understanding of:

1. What is an autonomous system
2. Key problems for autonomous systems
3. Some of the most important (classes of) algorithms in robotics
4. The model based vs. model free tradeoff
5. The online vs offline tradeoff
6. The no free lunch theorem and the need for approximations
7. How computer systems / architecture design has and can play a role in improving autonomous systems

Autonomous Systems / Robotics is a BIG space

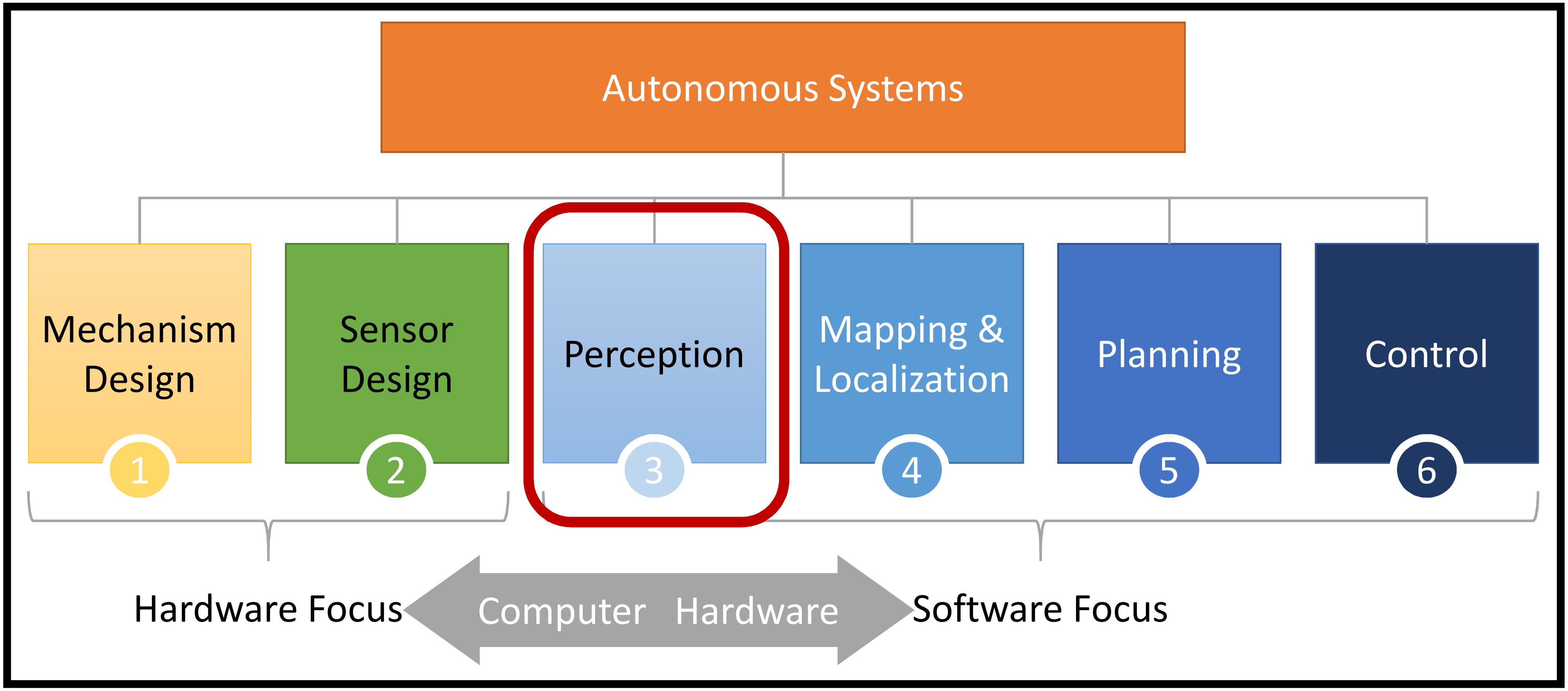


1 2 Key Takeaways:



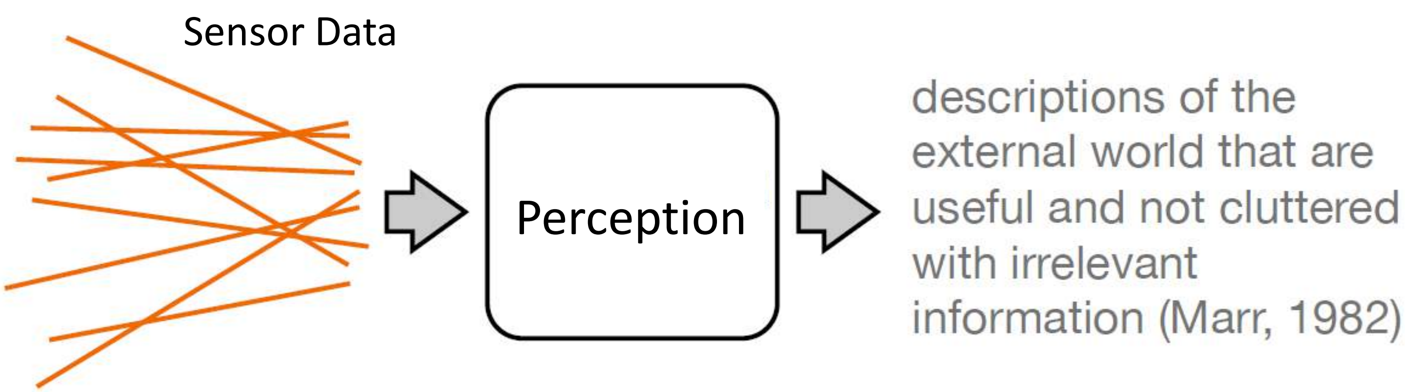

1. When designing algorithms for robots you need to understand the physical capabilities of the robot and you (potentially) need to understand how to model its physical behaviors
2. Different kinds of systems will have different power, weight, and performance budgets for computer hardware

Autonomous Systems / Robotics is a BIG space



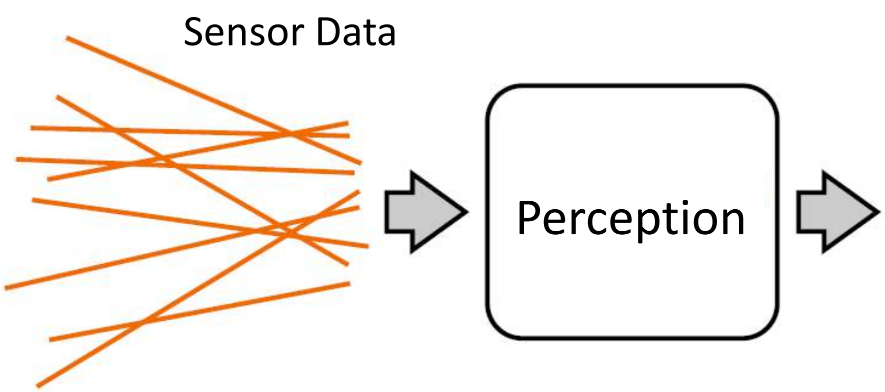
3

Perception is the processing of sensor data to understand the world around the robot



Slide Credit: Todd Zickler CS 283

Perception is the processing of sensor data to understand the world around the robot



descriptions of the external world that are useful and not cluttered with irrelevant information (Marr, 1982)

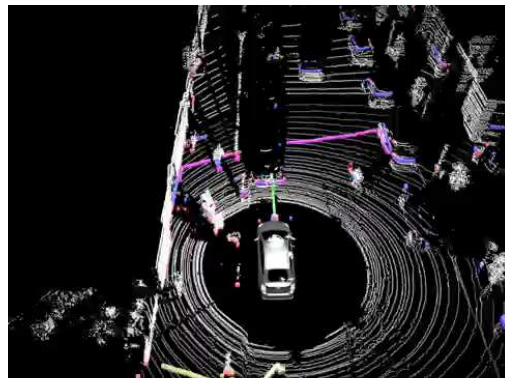
- Geometric information (3D shape, position)

Slide Credit: Todd Zickler CS 283

3

We can compute the depth to objects by using geometry and physics

Structured Light (e.g., LIDAR)



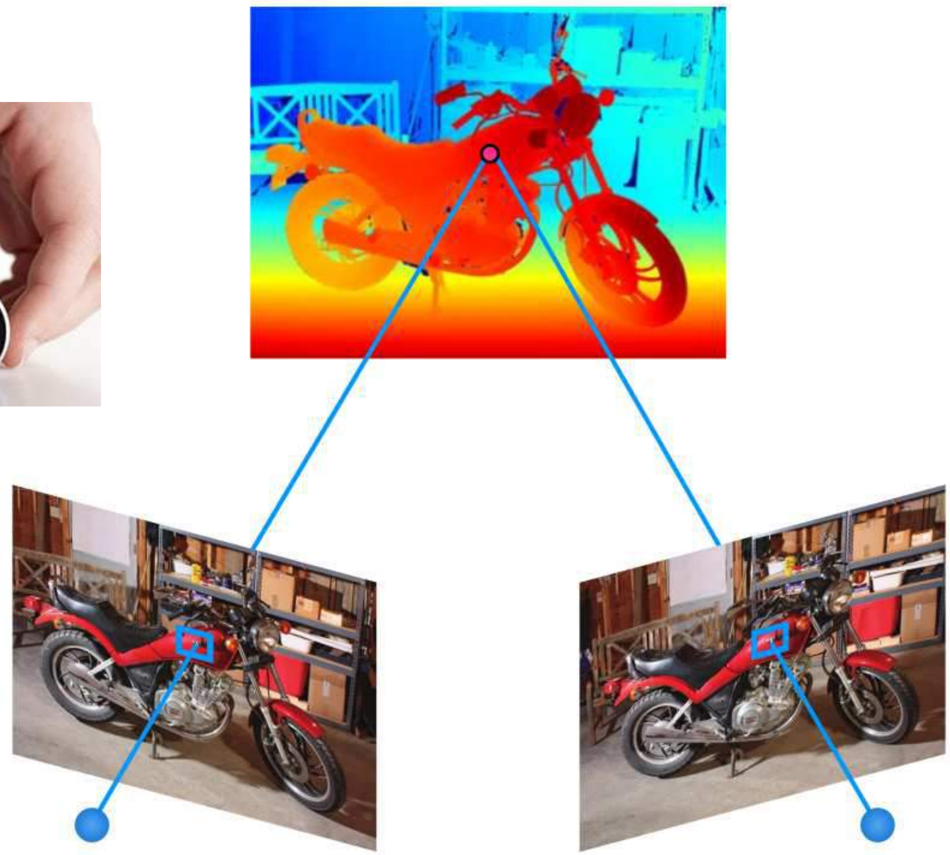
(and other Structured Waves e.g., Sonar, RADAR, etc.)

Unstructured Light (aka Cameras)

Computer Vision → Usable Data

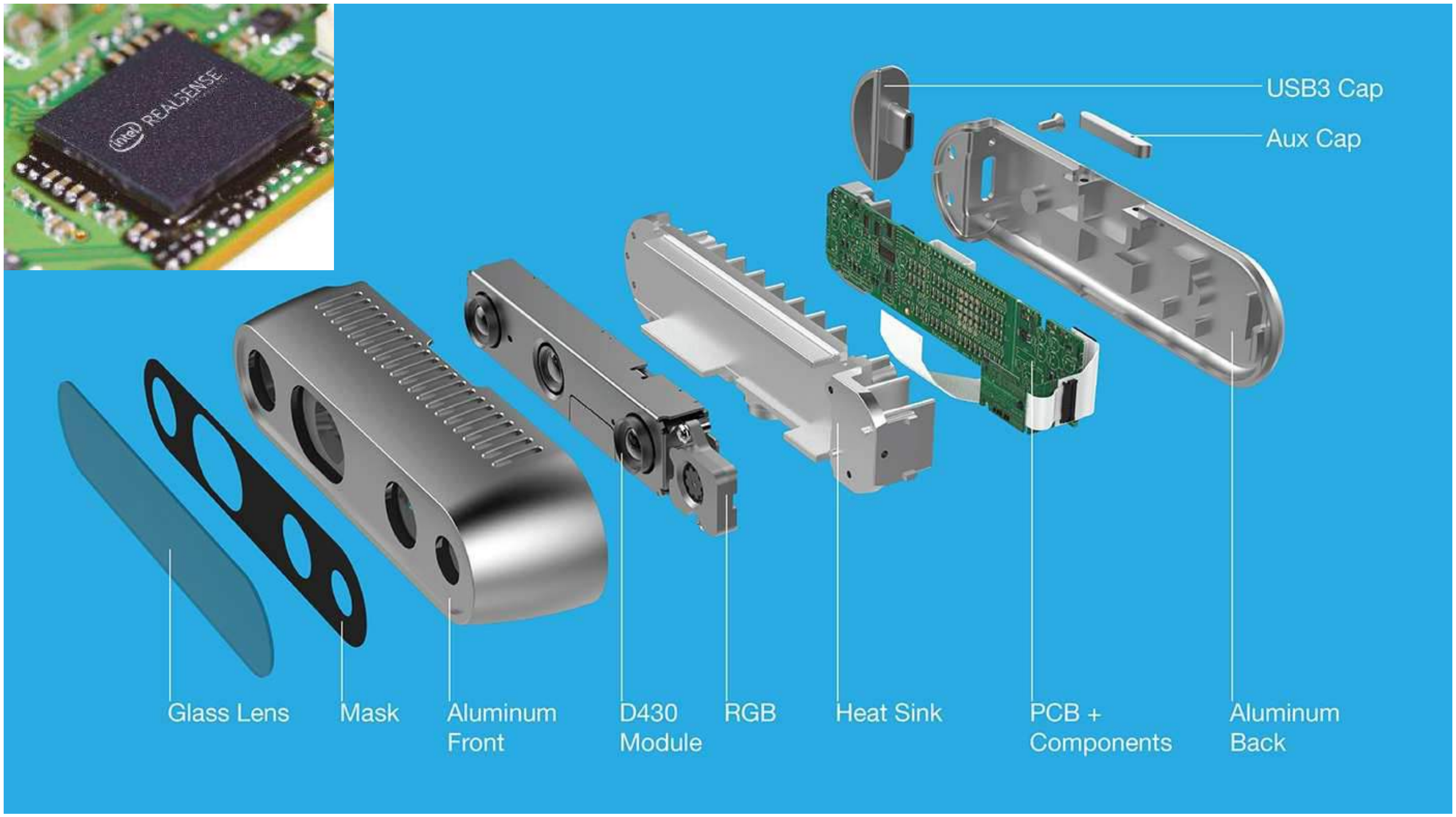
3

We can compute the depth to objects by using geometry and physics



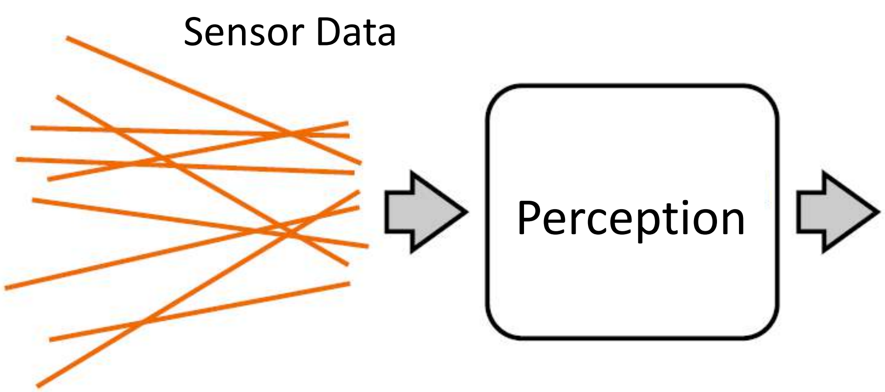
3

Stereo depth is such an important problem that Intel has designed a custom chip!



Perception is the processing of sensor data to understand the world around the robot

3



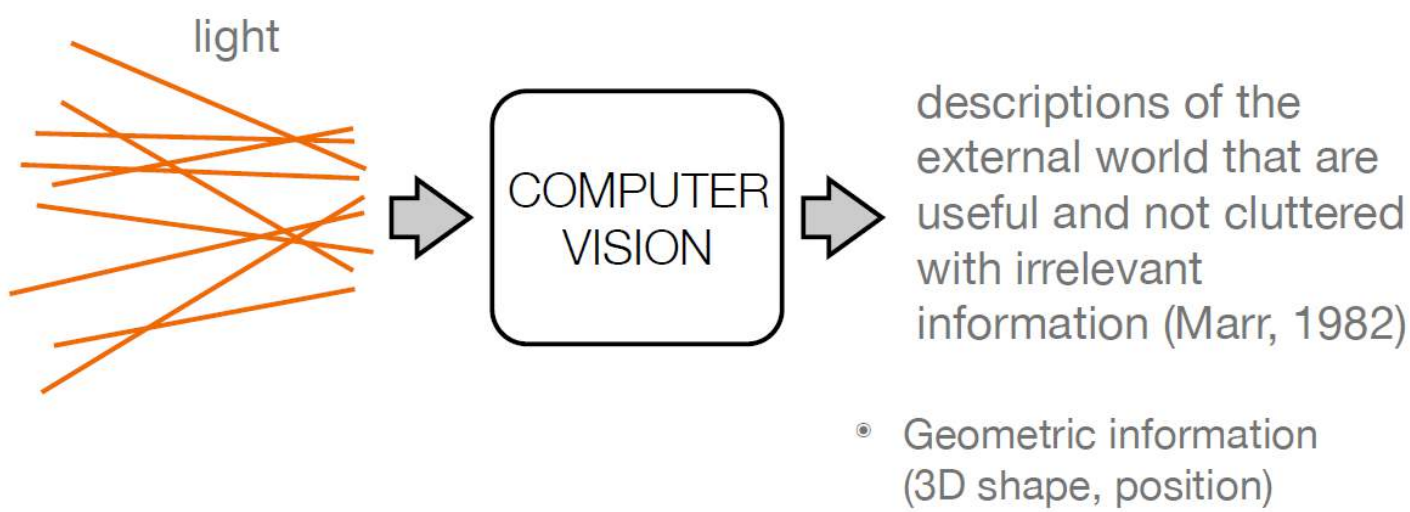
descriptions of the external world that are useful and not cluttered with irrelevant information (Marr, 1982)

- Geometric information (3D shape, position)

Slide Credit: Todd Zickler CS 283

3

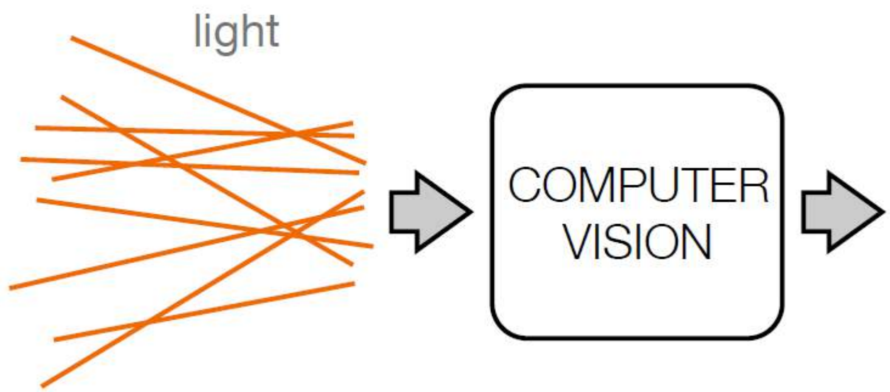
Perception is the processing of sensor data to understand the world around the robot



Slide Credit: Todd Zickler CS 283

3

Perception is the processing of sensor data to understand the world around the robot



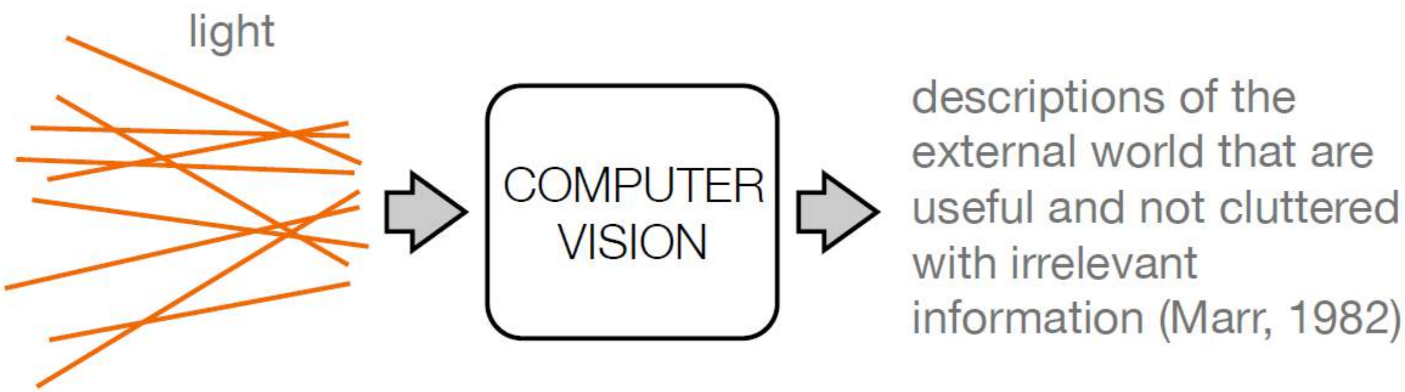
descriptions of the external world that are useful and not cluttered with irrelevant information (Marr, 1982)

- Geometric information (3D shape, position)
- Dynamic information (velocities)

Slide Credit: Todd Zickler CS 283

3

Perception is the processing of sensor data to understand the world around the robot



- Geometric information (3D shape, position)
- Dynamic information (velocities)
- Semantic information (object, scene categories)

Slide Credit: Todd Zickler CS 283

3 Computer Vision is a hard problem

3 Computer Vision is a hard problem

What color(s) are this shirt and these pants?

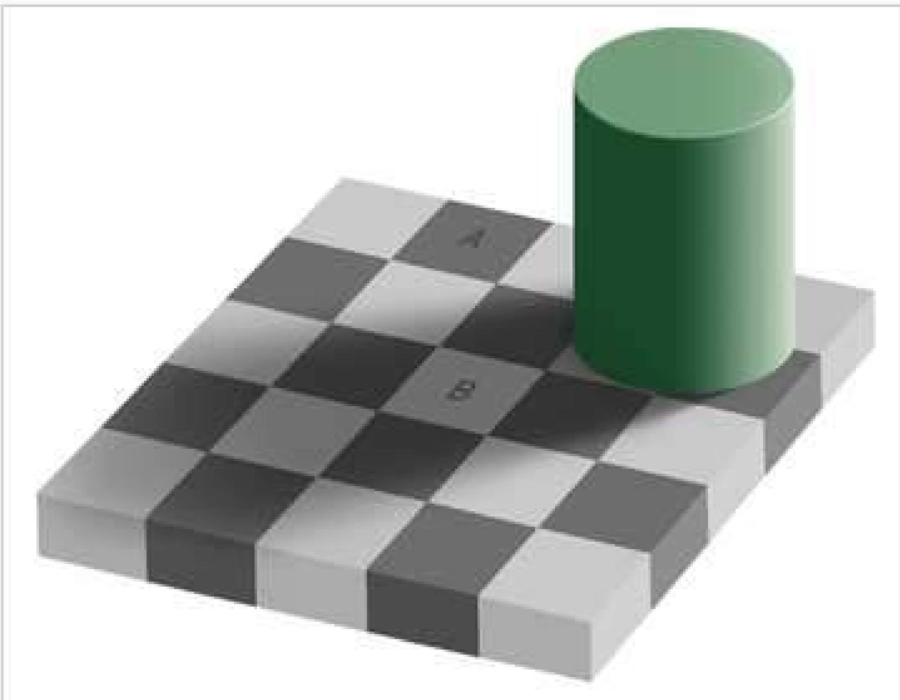
Slide Credit: Hamilton Chong

3

Computer Vision is a hard problem

Slide Credit: Hamilton Chong

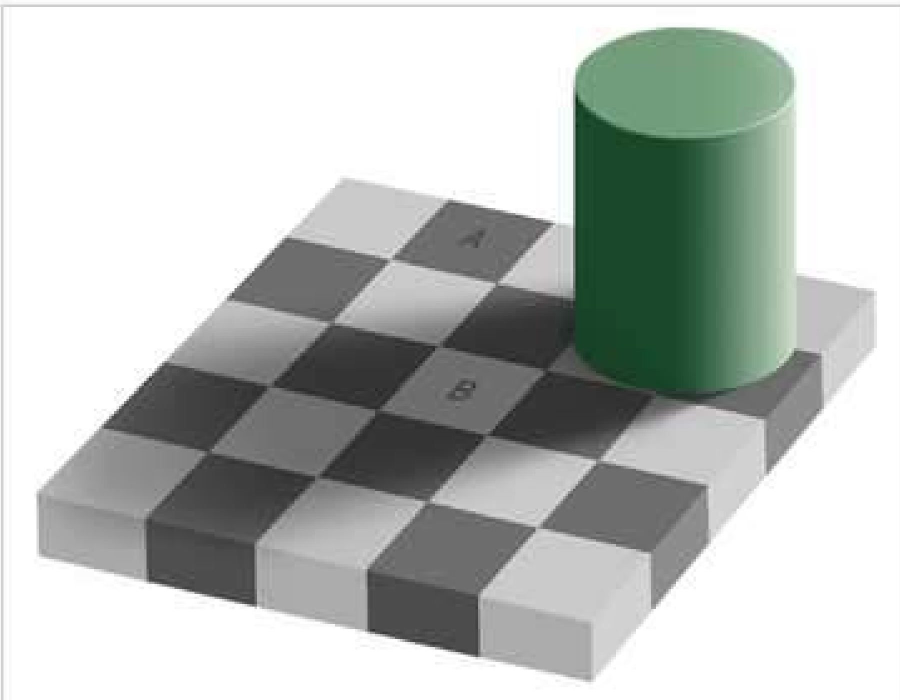
3 Computer Vision is a hard problem



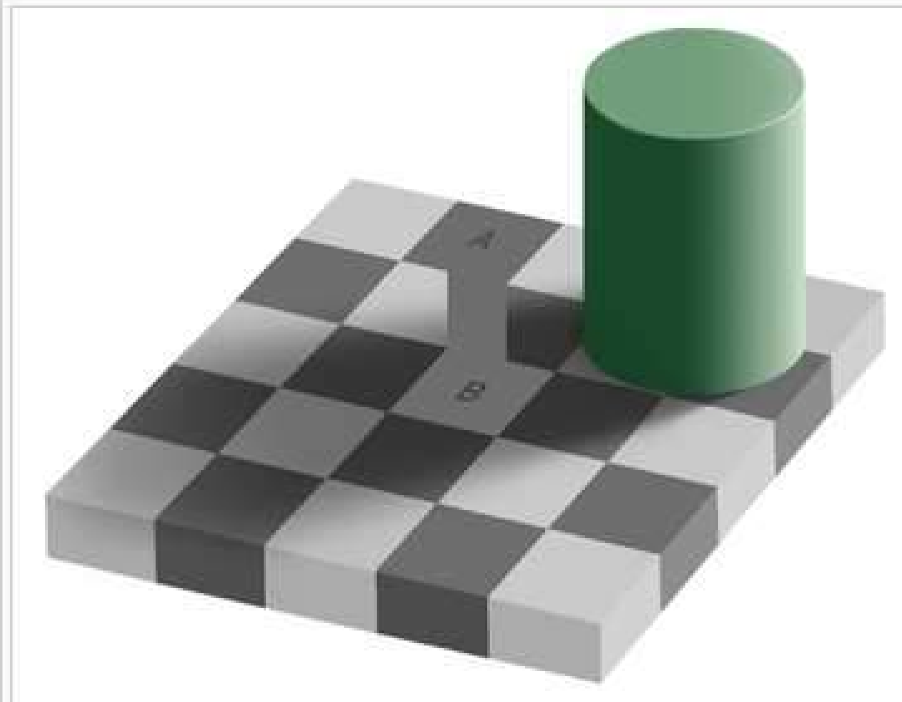
Areas of the image A and B are the same color

Adelson 1995

3 Computer Vision is a hard problem



Areas of the image A and B are the same color



A rectangle of the same color has been drawn connecting the two areas of the image

Adelson 1995

3

Computer Vision is a hard problem

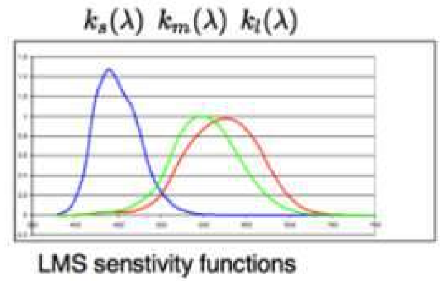
Sinha *et al*.: Face Recognition by Humans: Nineteen Results Researchers Should Know About

Computer Vision is a hard problem

Retinal color

$$\mathbf{c}(\ell(\lambda)) = (c_s, c_m, c_l)$$

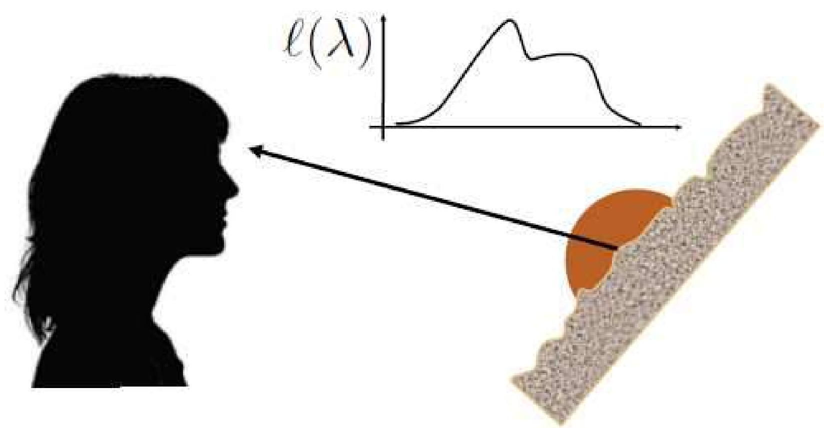
$$c_s = \int k_s(\lambda)\ell(\lambda)d\lambda$$



Perceived color

Object color

Color names



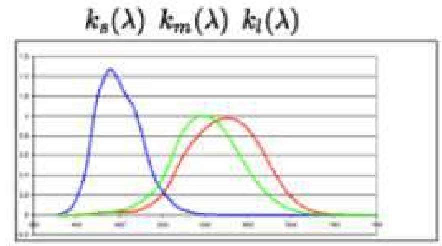
Slide Credit: Todd Zickler CS 283

③ Computer Vision is a hard problem

Retinal color

$$\mathbf{c}(\ell(\lambda)) = (c_s, c_m, c_l)$$

$$c_s = \int k_s(\lambda)\ell(\lambda)d\lambda$$



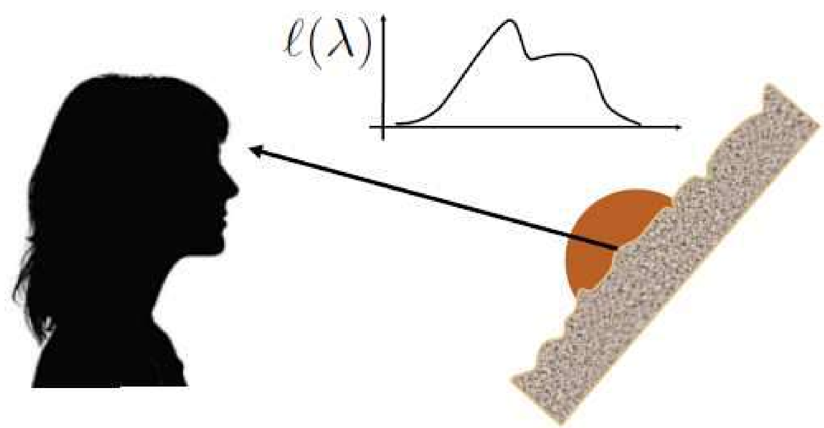
LMS senstivity functions

So how can we represent this with an algorithm?

Perceived color

Object color

Color names



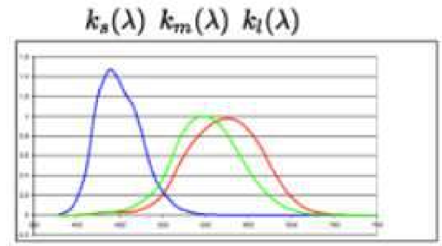
Slide Credit: Todd Zickler CS 283

3 Computer Vision is a hard problem

Retinal color

$$\mathbf{c}(\ell(\lambda)) = (c_s, c_m, c_l)$$

$$c_s = \int k_s(\lambda)\ell(\lambda)d\lambda$$



LMS senstivity functions

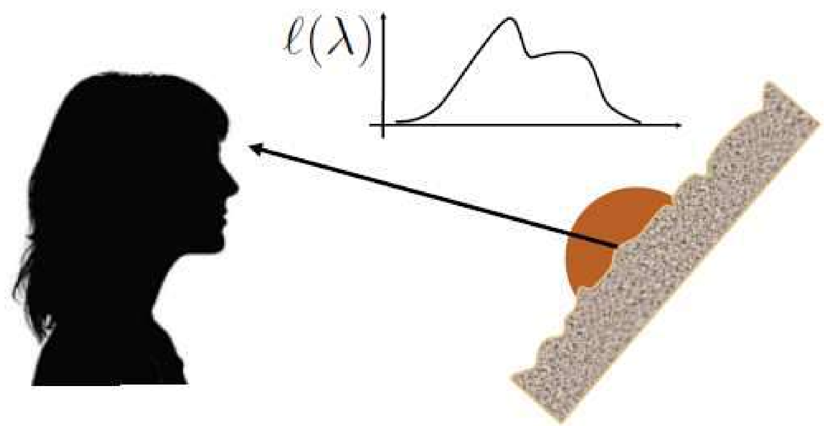
So how can we represent this with an algorithm?

Perceived color

Object color

Color names

Well lets start by building up some intuition for how to find an edge!

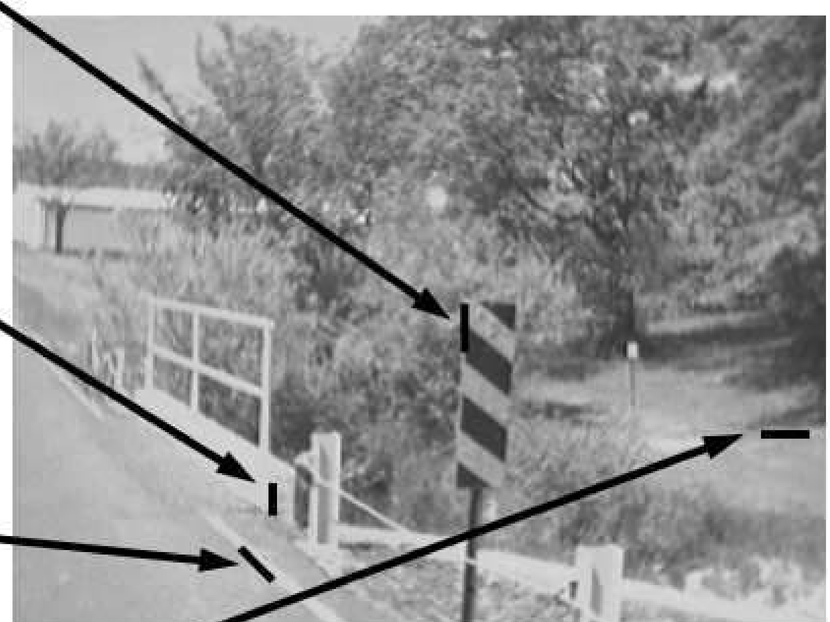


Slide Credit: Todd Zickler CS 283

3

Edges are where discontinuities occur in images

- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)
- Illumination discontinuity (e.g., shadow)

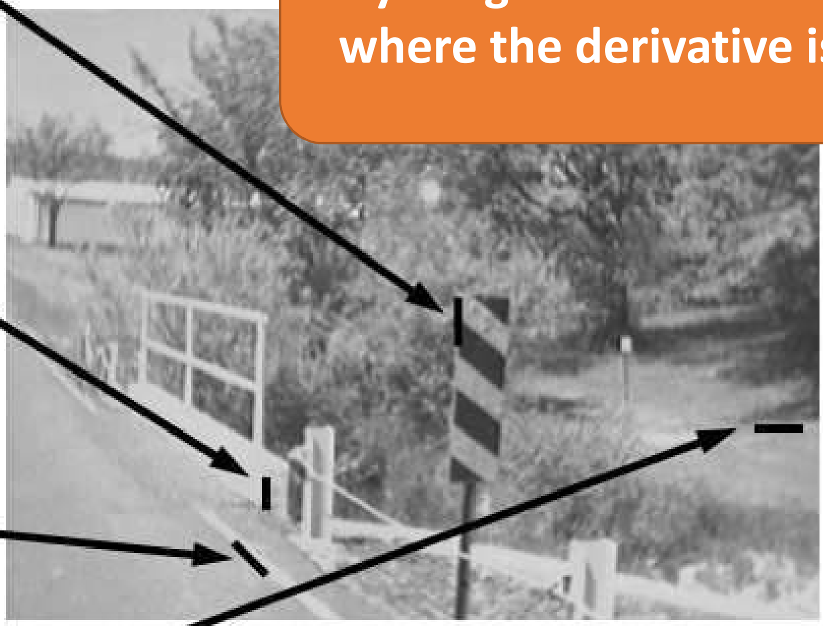


Slide credit: Christopher Rasmussen

3

Edges are where discontinuities occur in images

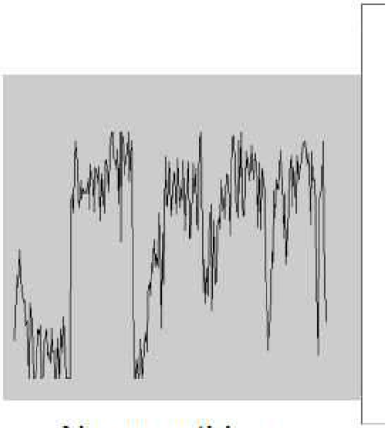
- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)
- Illumination discontinuity (e.g., shadow)



Key insight: discontinuities are where the derivative is high!

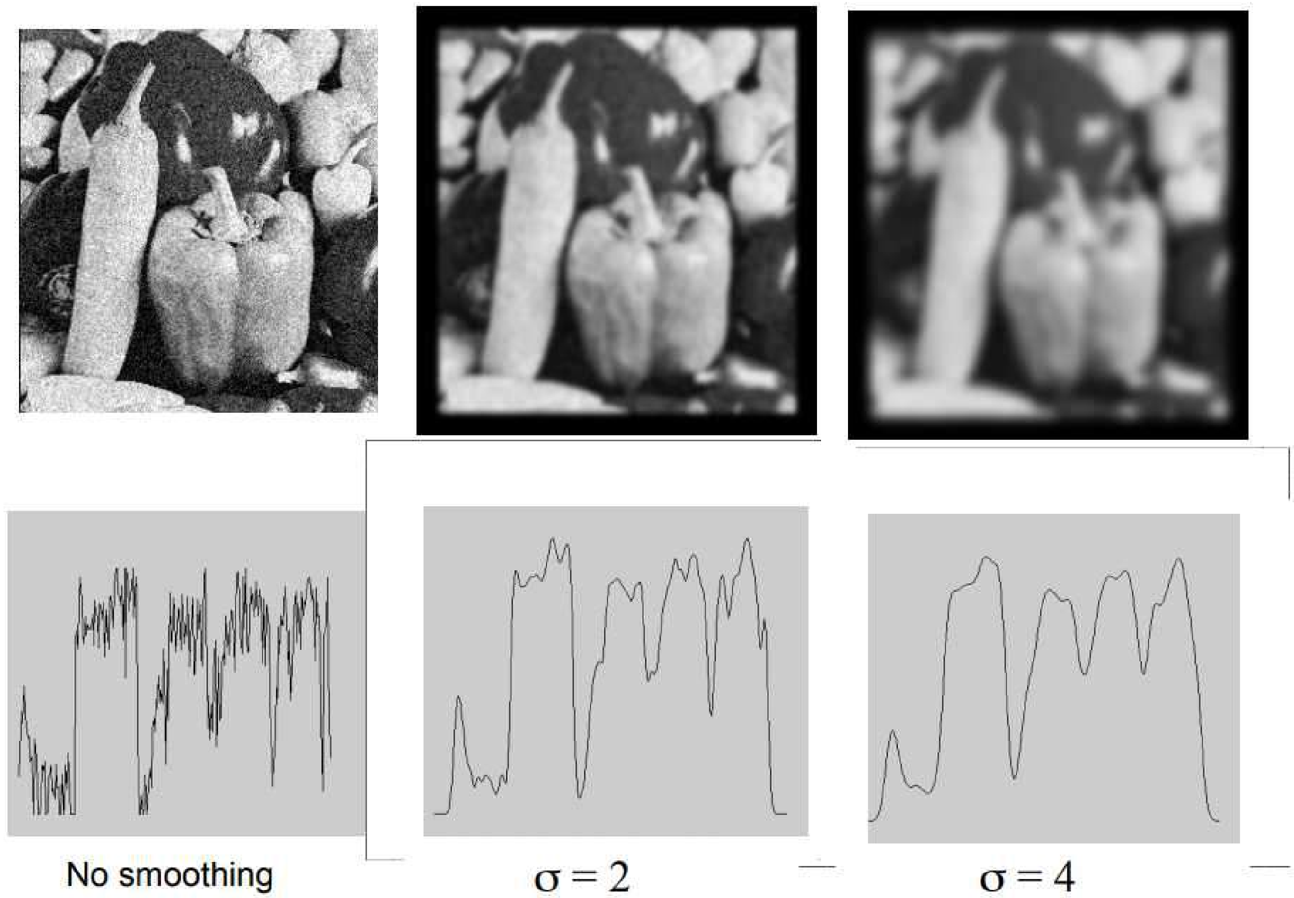
Slide credit: Christopher Rasmussen

3 Noise will corrupt our derivative computation



No smoothing

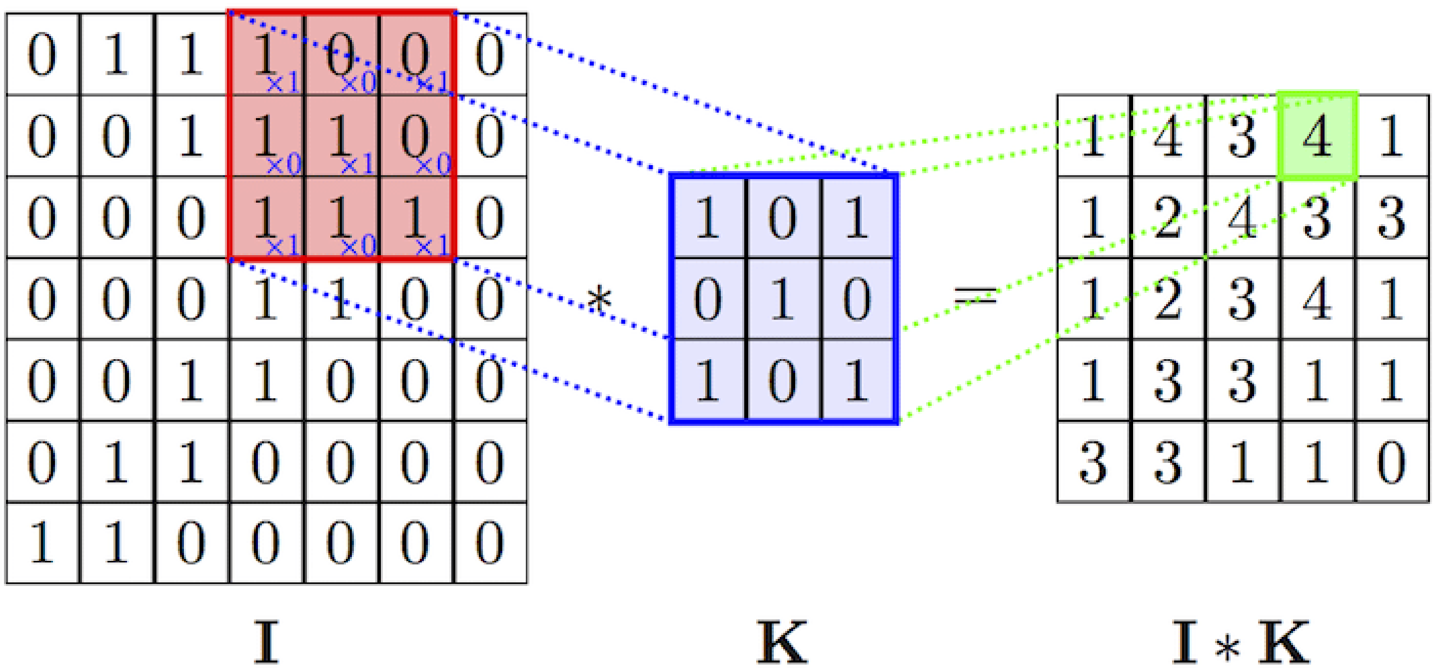
3 "Spatially local averaging" reduces noise



3

The traditional Computer Vision approach is through convolution of linear filters

So what is a convolution of a linear filter?

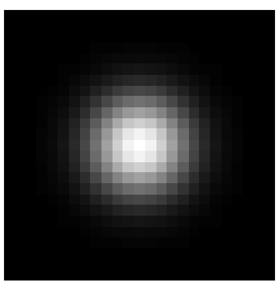


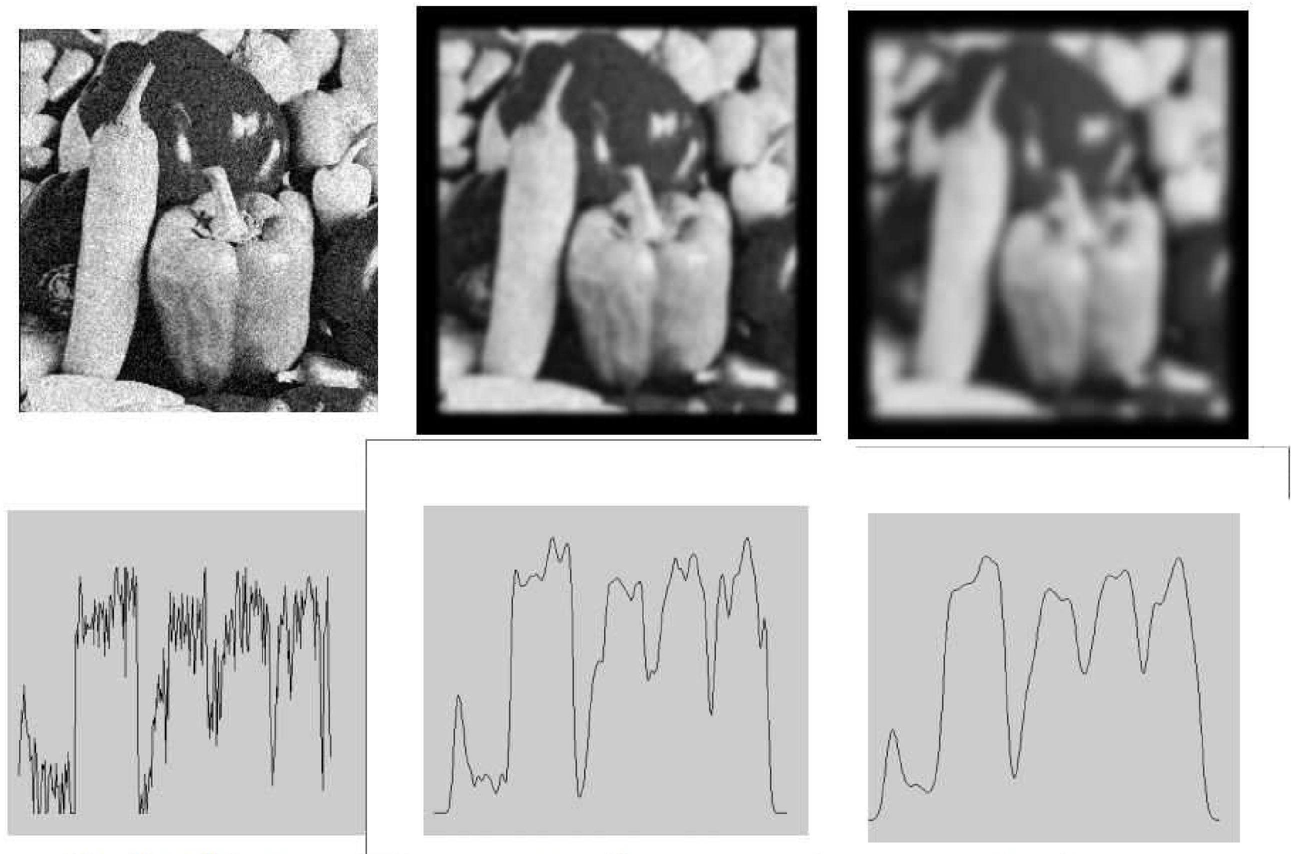
3 “Spatially local averaging” reduces noise

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$





No smoothing

$\sigma = 2$

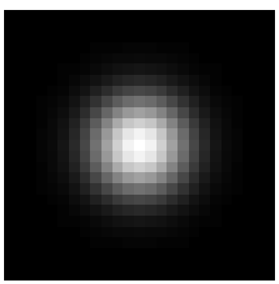
$\sigma = 4$

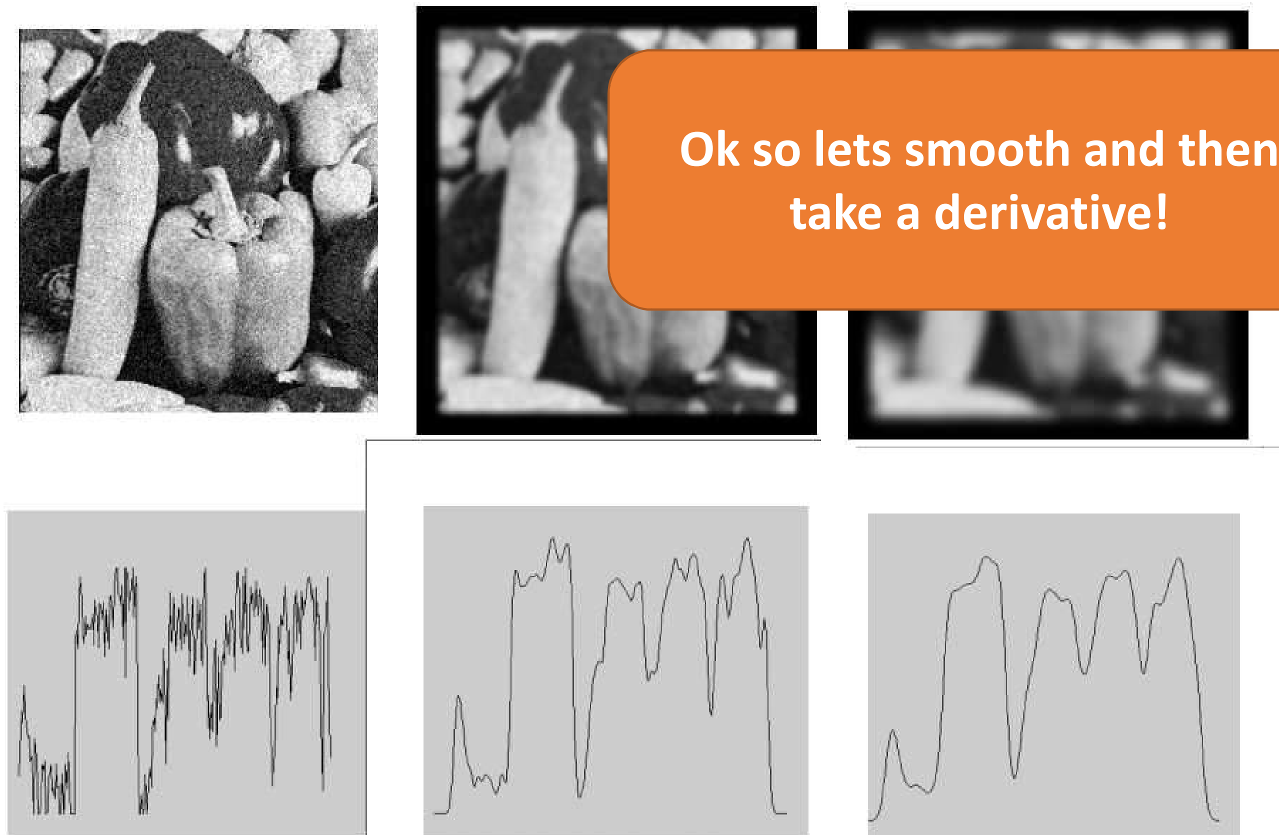
3 "Spatially local averaging" reduces noise

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$





No smoothing $\quad$ $\sigma = 2$ $\quad$ $\sigma = 4$

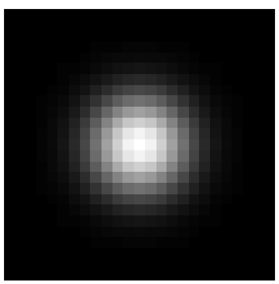
Ok so lets smooth and then take a derivative!

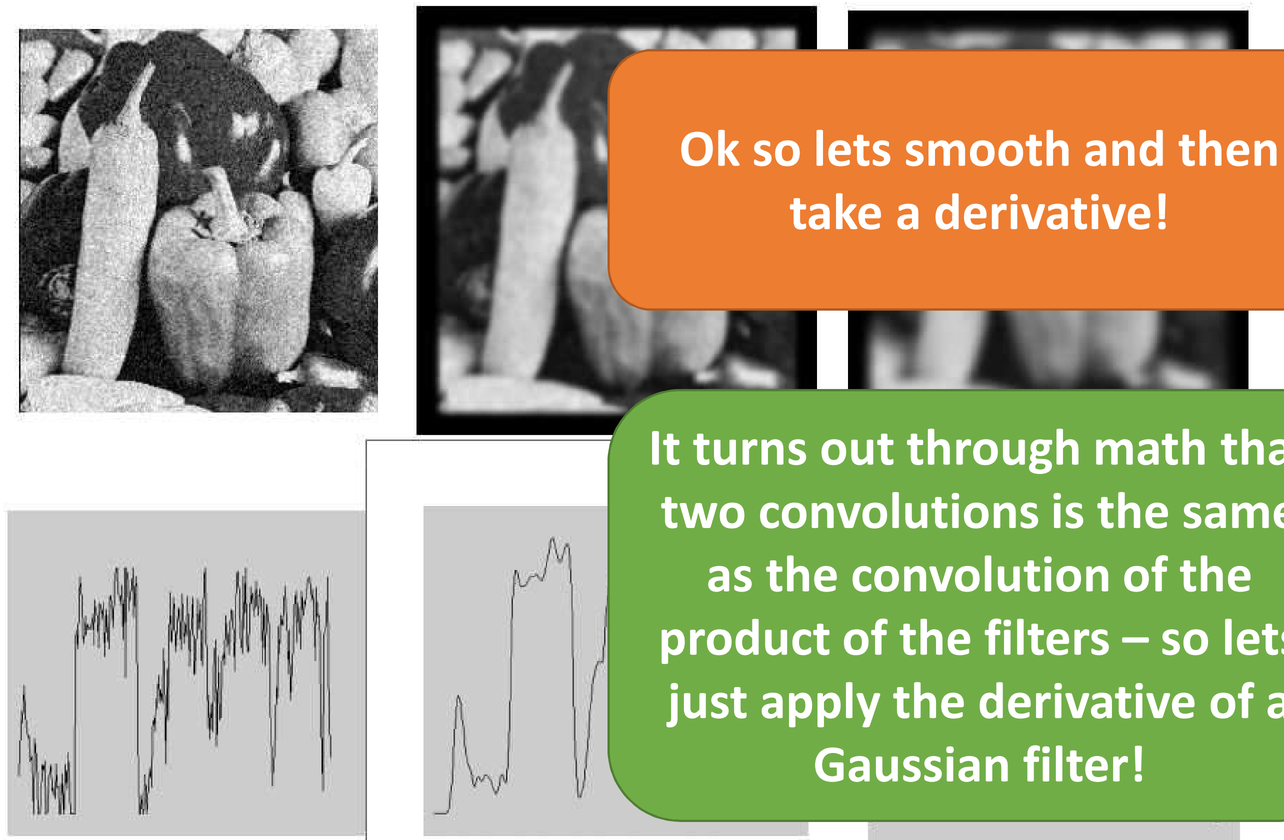
3 “Spatially local averaging” reduces noise

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, σ = 1





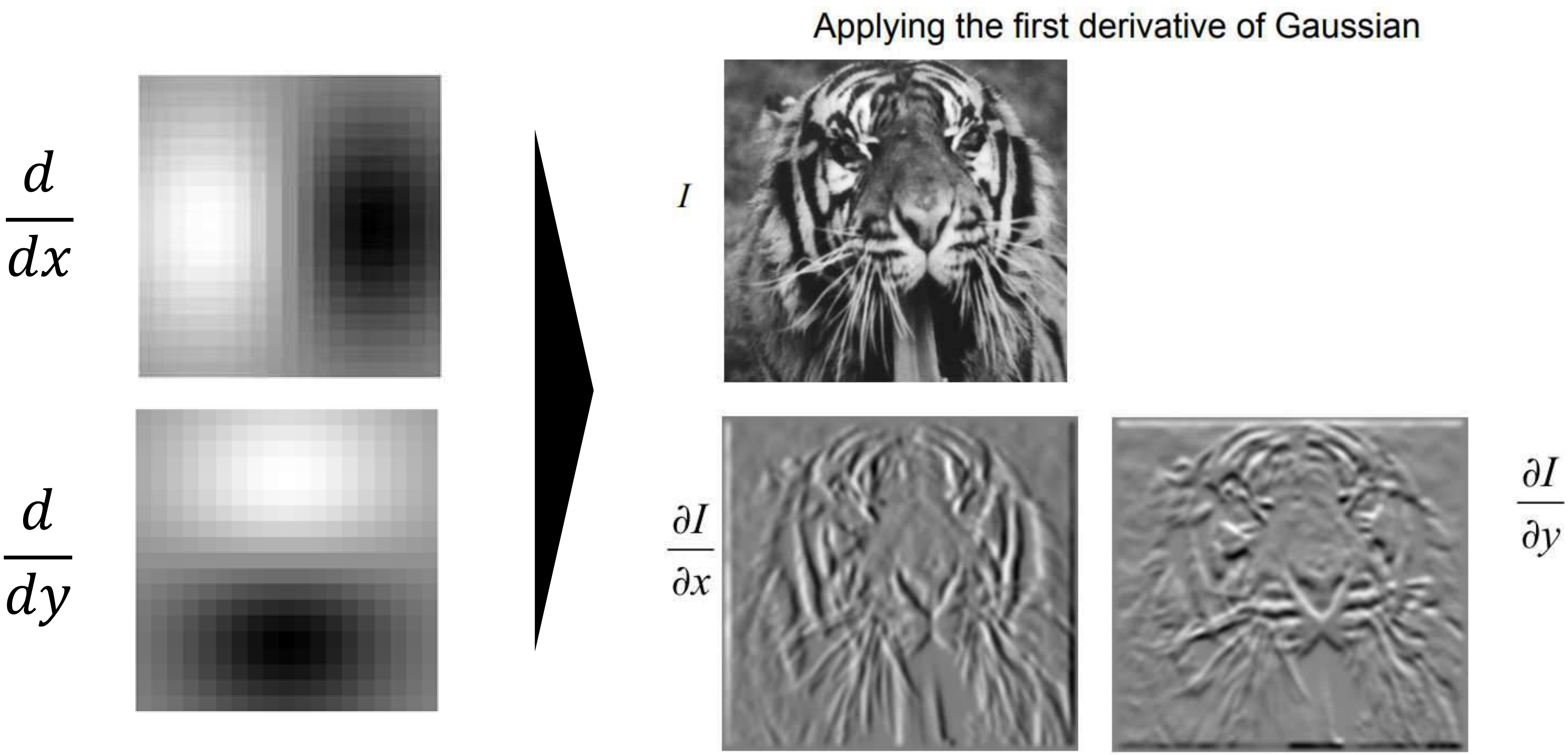
No smoothing $\sigma = 2$ $\sigma = 4$

Ok so lets smooth and then take a derivative!

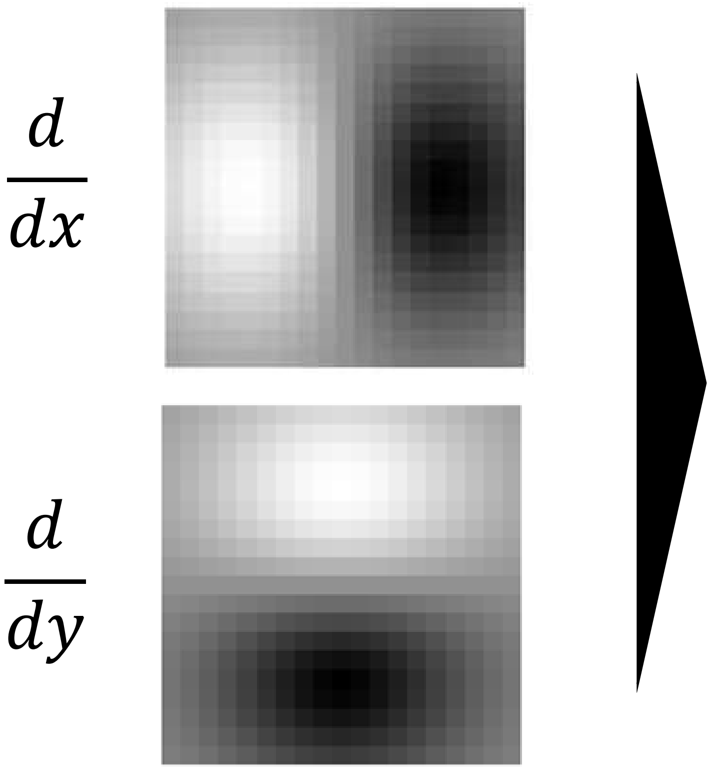
It turns out through math that two convolutions is the same as the convolution of the product of the filters – so lets just apply the derivative of a Gaussian filter!

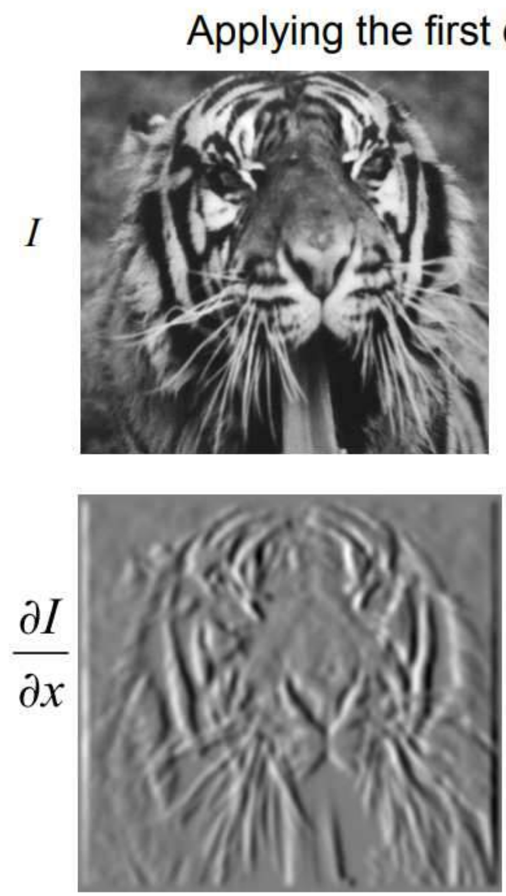
3

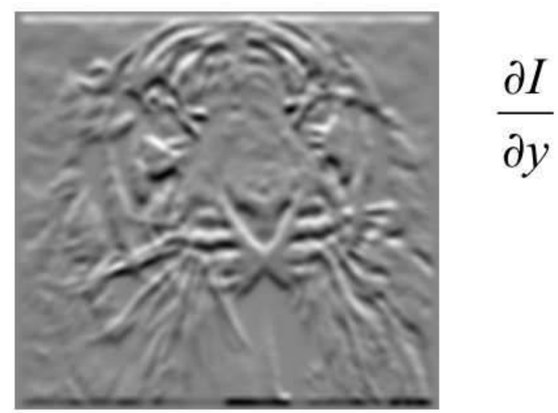
Derivatives increase noise so we can find edges using a derivative of Gaussian Filter



③ Derivatives increase noise so we can find edges using a derivative of Gaussian Filter



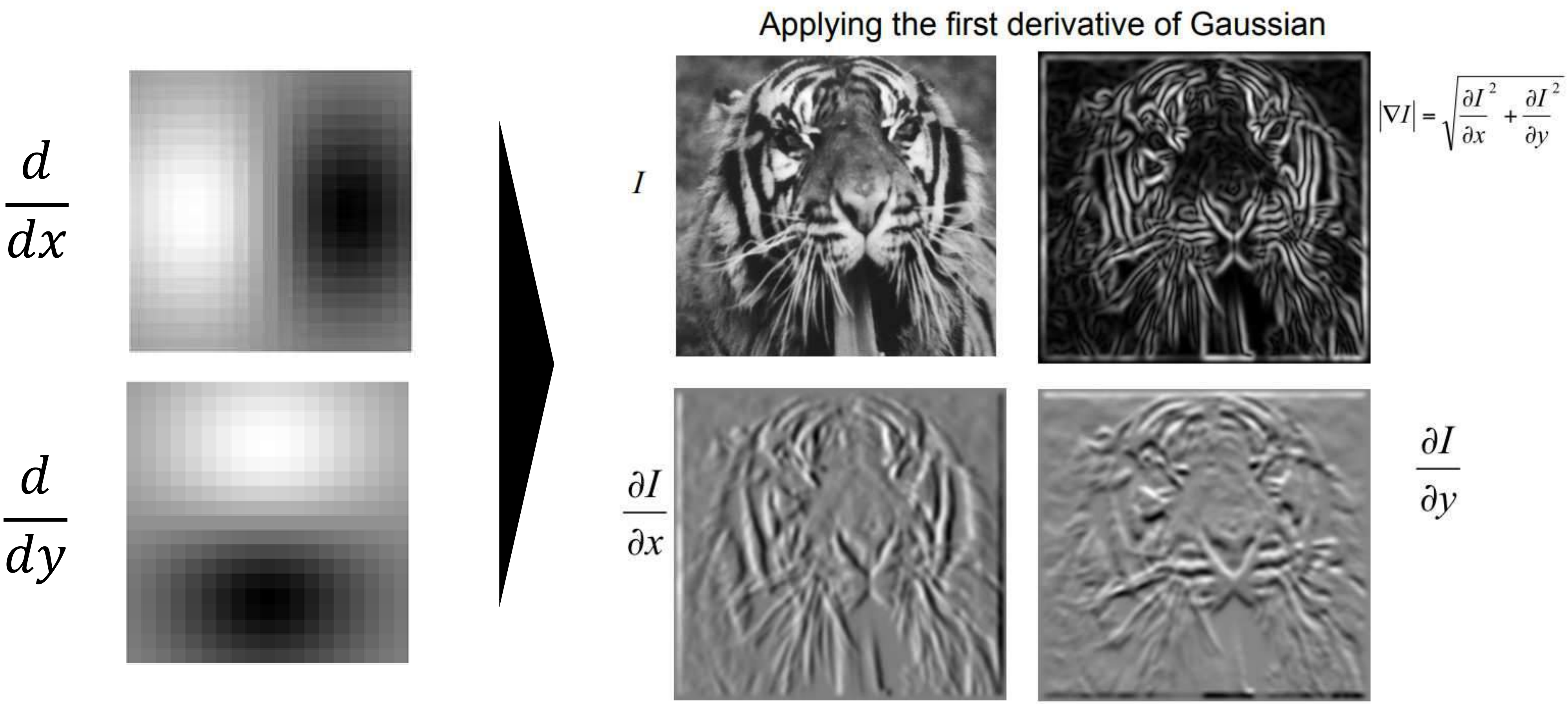




But not all edges are vertical or horizontal what can we do?

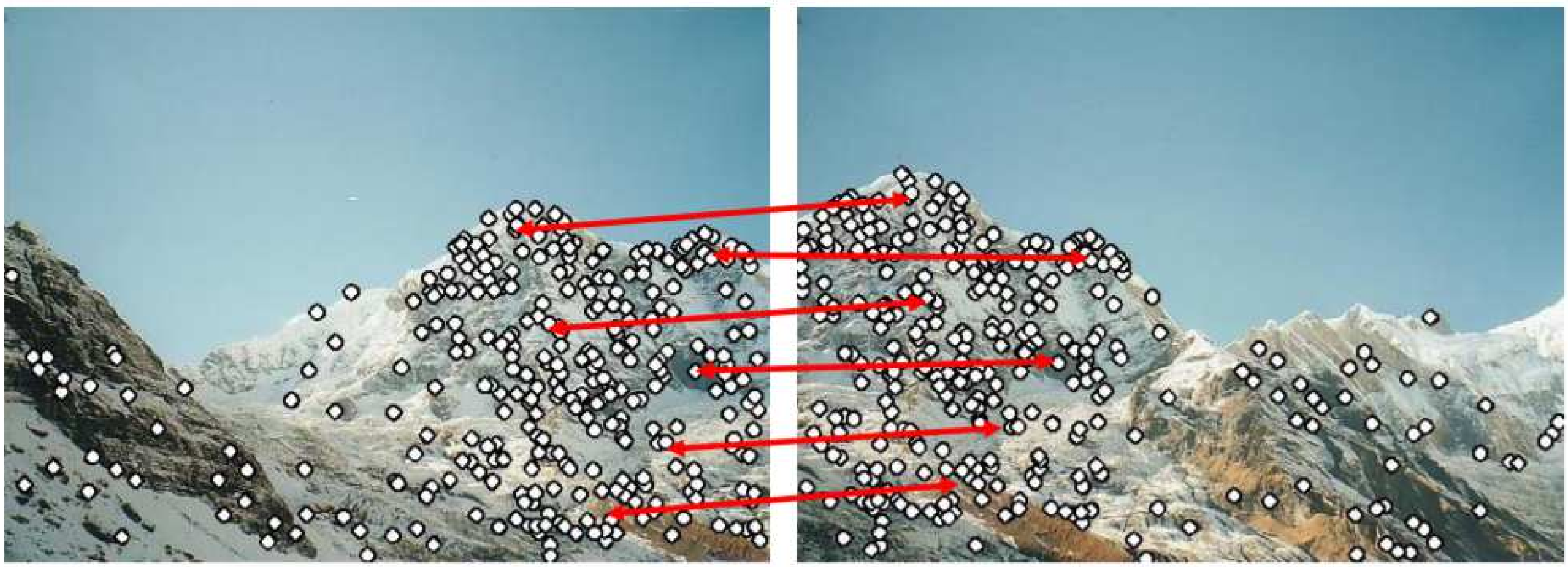
3

Derivatives increase noise so we can find edges using a derivative of Gaussian Filter



3

Various filters can be used to extract features to e.g., stich panoramas



Step 1: extract features

Step 2: match features

3

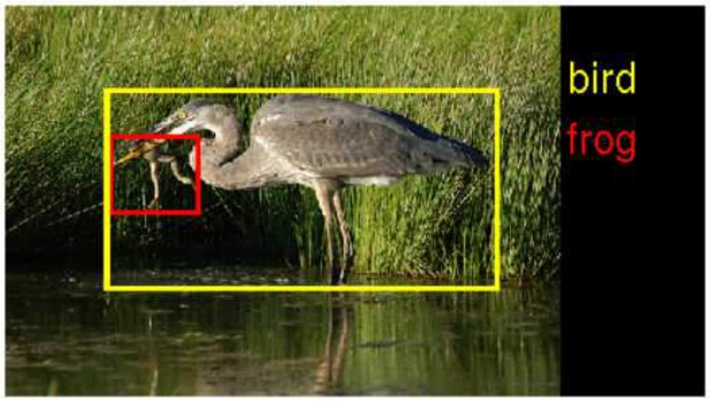
Various filters can be used to extract features to e.g., stich panoramas

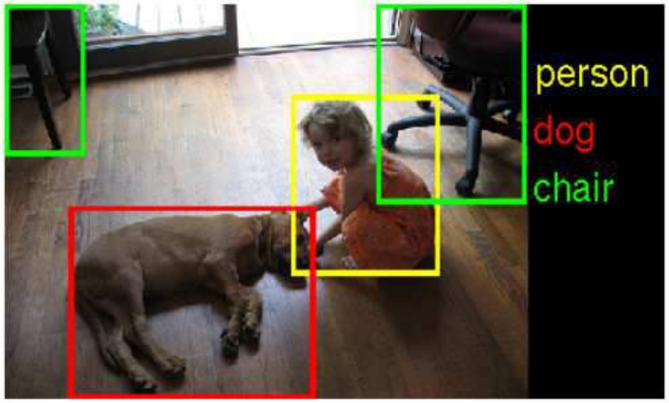
Step 1: extract features

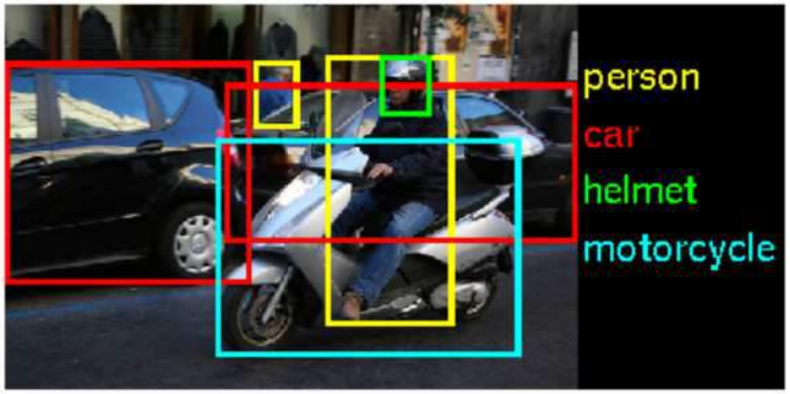
Step 2: match features

Step 3: align images

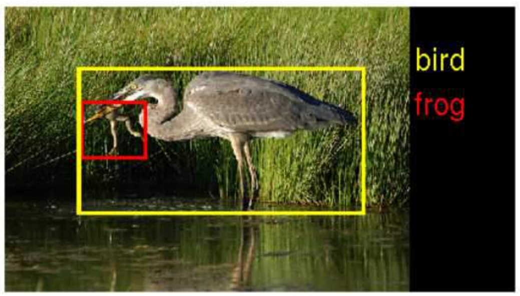
But what features should we use for object recognition?

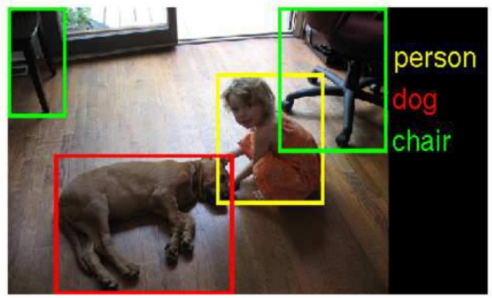


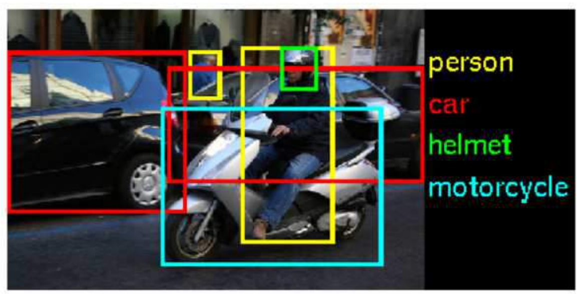




3 The ImageNet Challenge

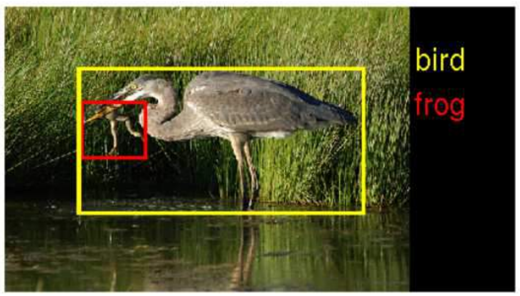


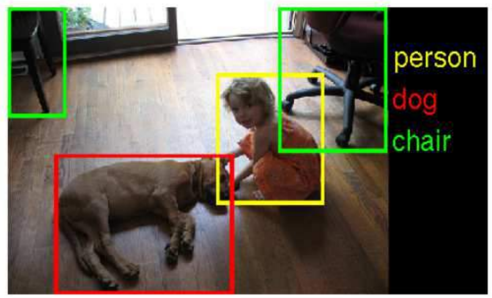


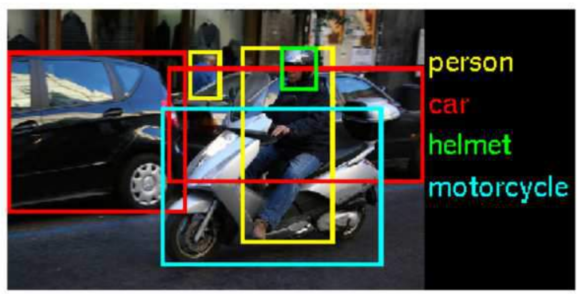


The ImageNet Challenge provided 1.2 million examples of 1,000 **labeled** items and challenged algorithms to learn from the data and then was tested on another 100,000 images

3 The ImageNet Challenge



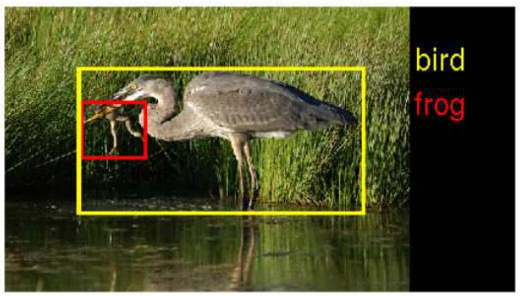


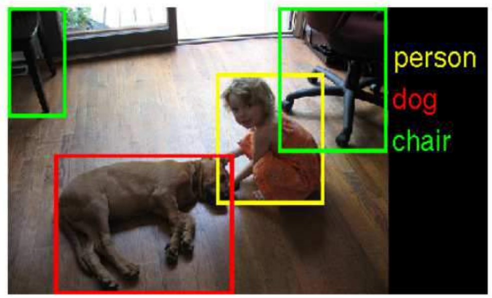


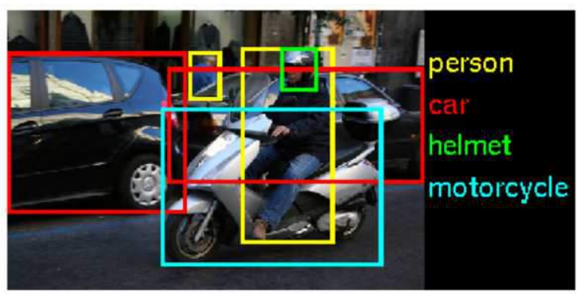
In 2010 teams had
75-50% error

In 2011 teams had
75-25% error

③ The ImageNet Challenge





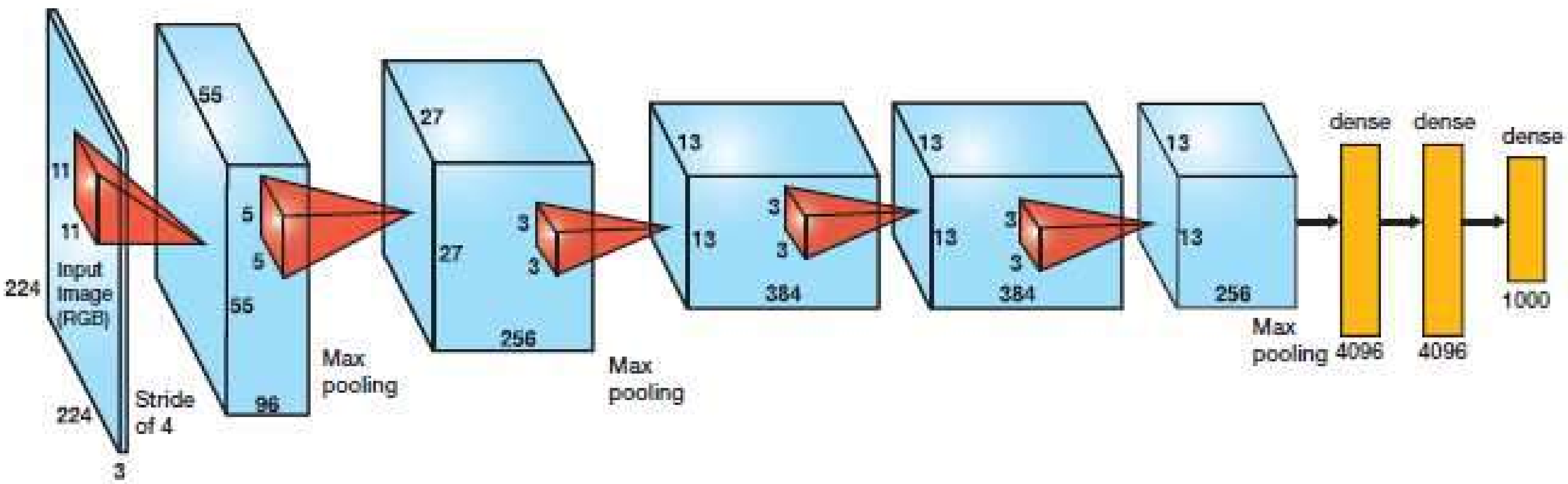


In 2012 still no team had less than 25% error barrier except **AlexNet at 15%**

Deep learning automates the design, selection and extraction of features, with amazing results

3

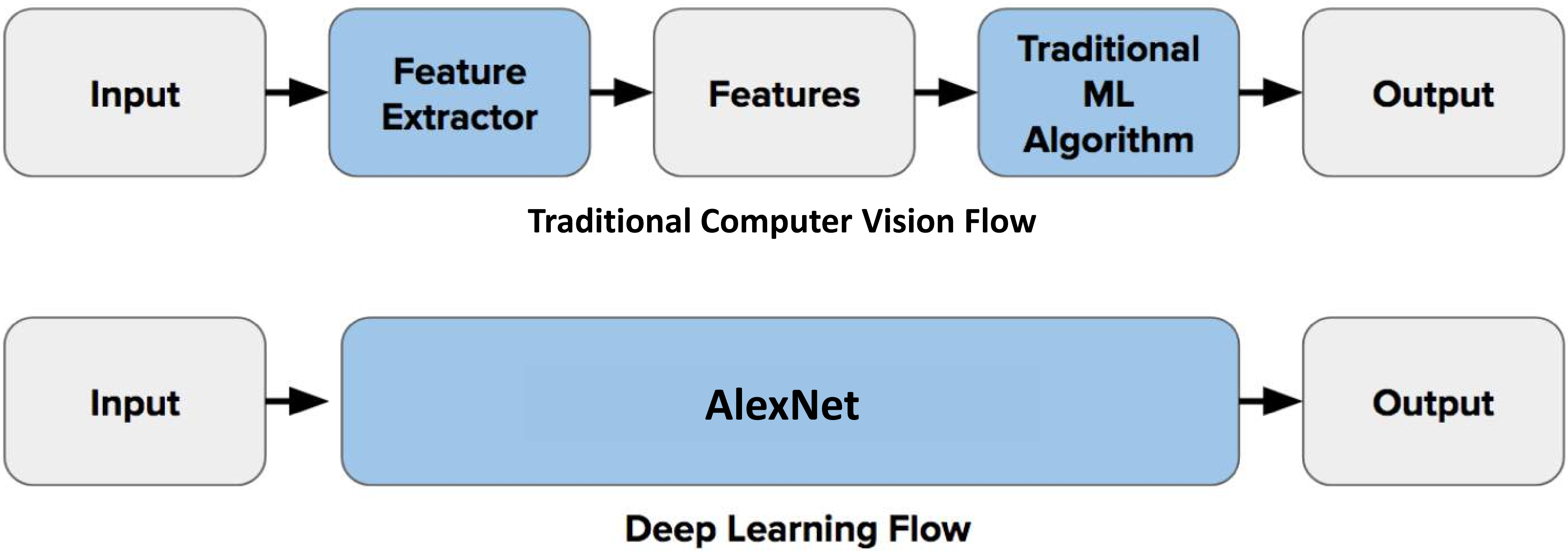
AlexNet: the first widely successful application of deep learning



https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

3

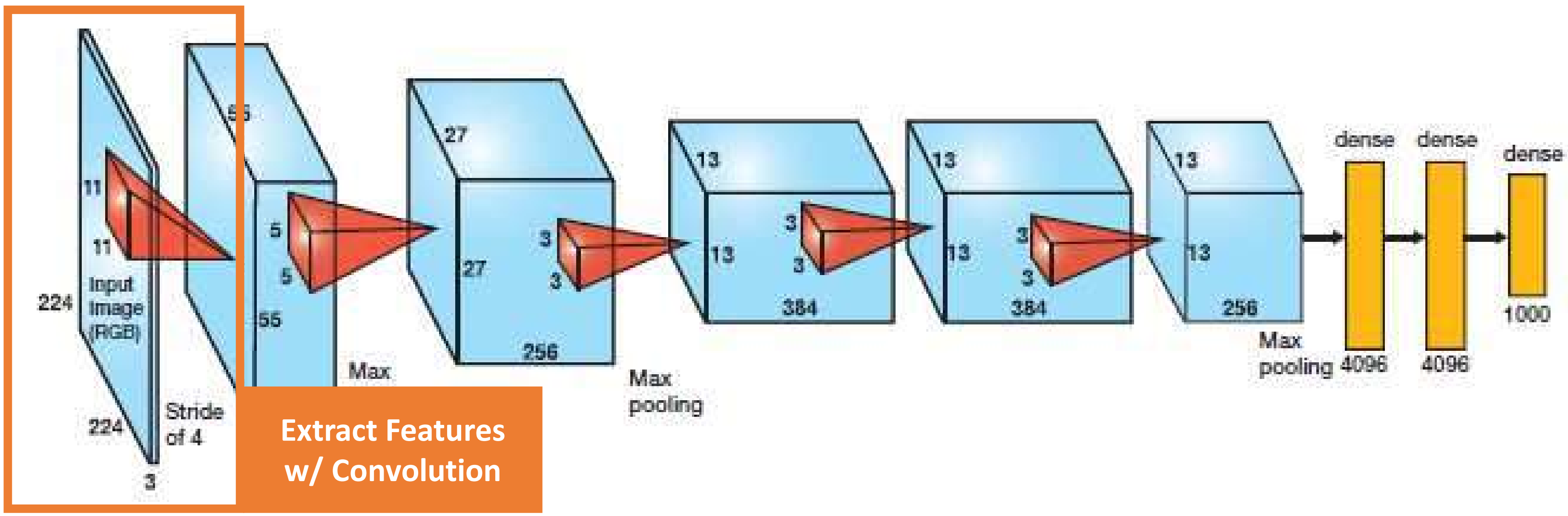
Deep learning automates the design, selection and extraction of features, with amazing results



3

Deep learning automates the design, selection and extraction of features, with amazing results

AlexNet: the first widely successful application of deep learning

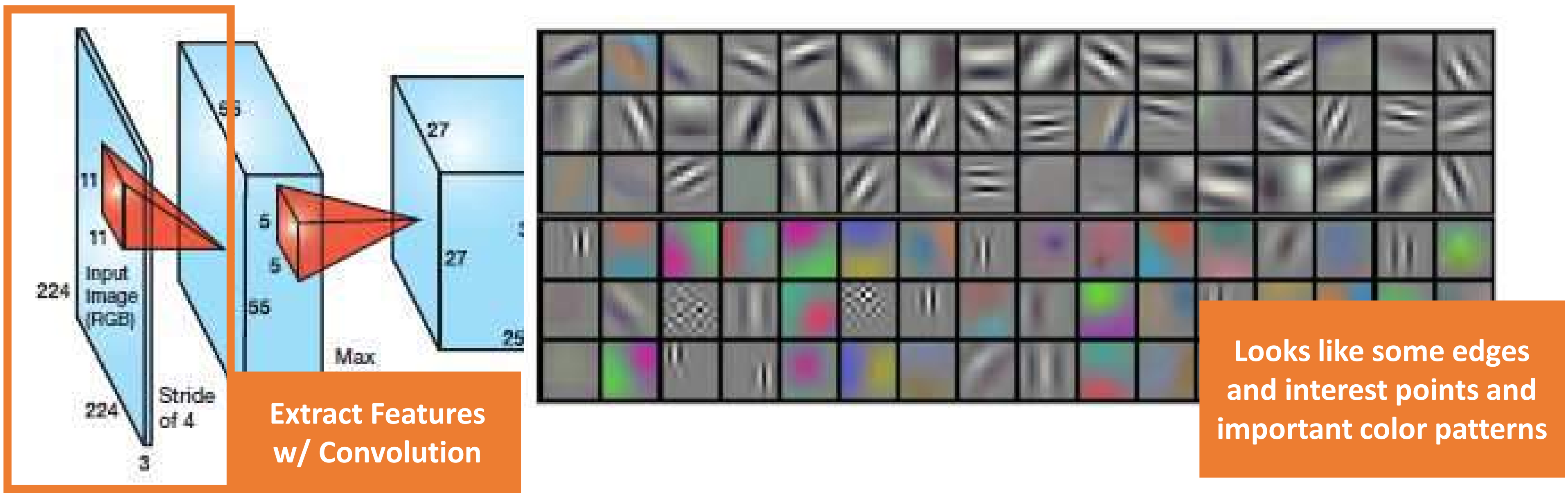


https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

3

Deep learning automates the design, selection and extraction of features , with amazing results

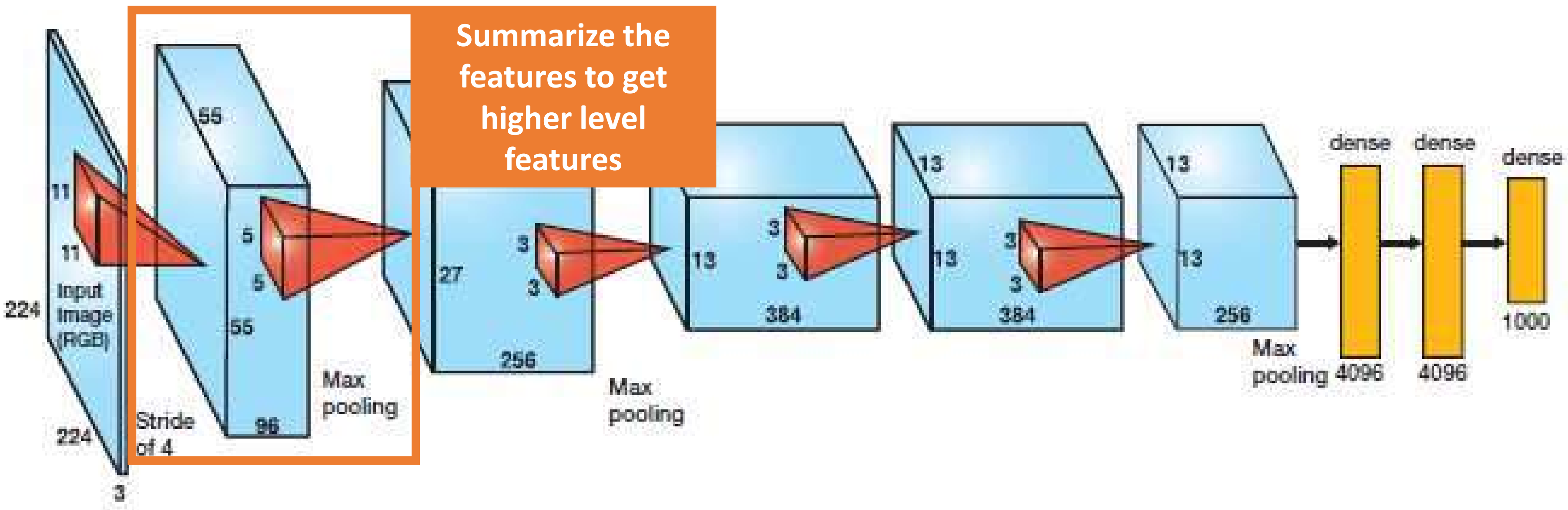
AlexNet: the first widely successful application of deep learning



https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

Deep learning automates the design, selection and extraction of features, with amazing results

AlexNet: the first widely successful application of deep learning

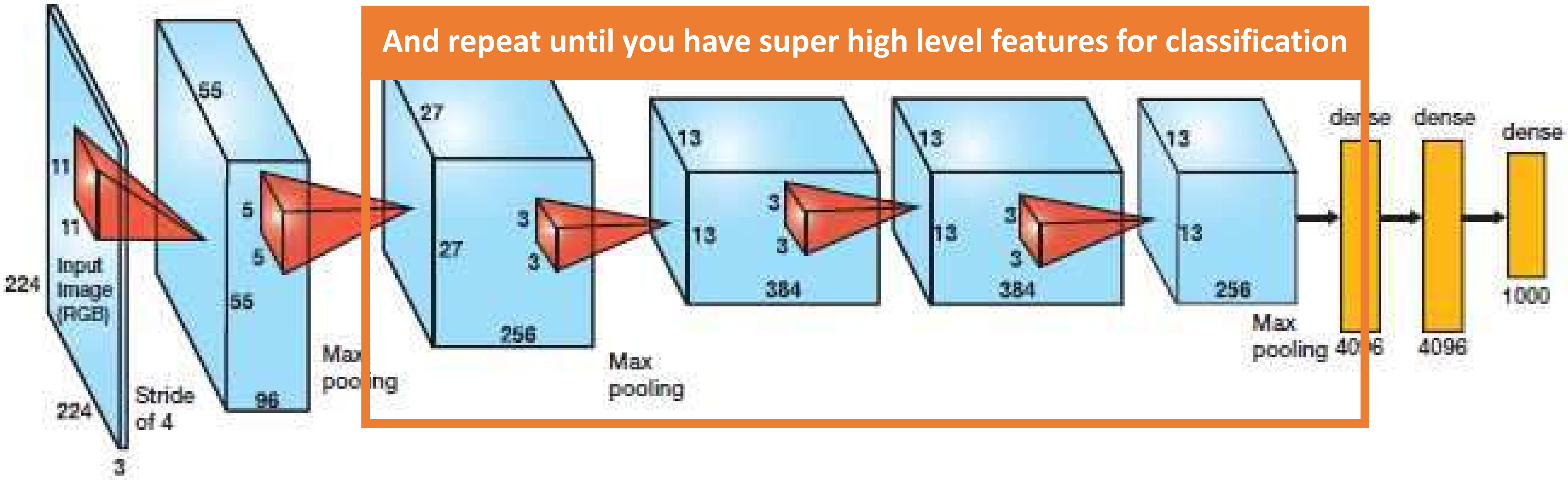


https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

3

Deep learning automates the design, selection and extraction of features, with amazing results

AlexNet: the first widely successful application of deep learning

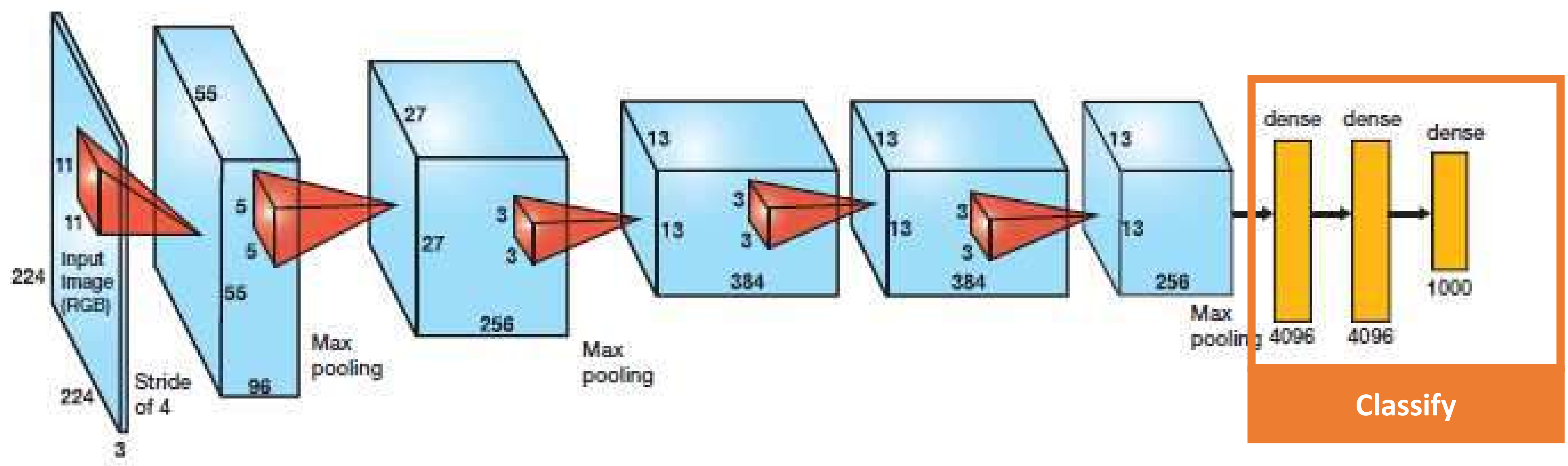


https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

3

Deep learning automates the design, selection and extraction of features, with amazing results

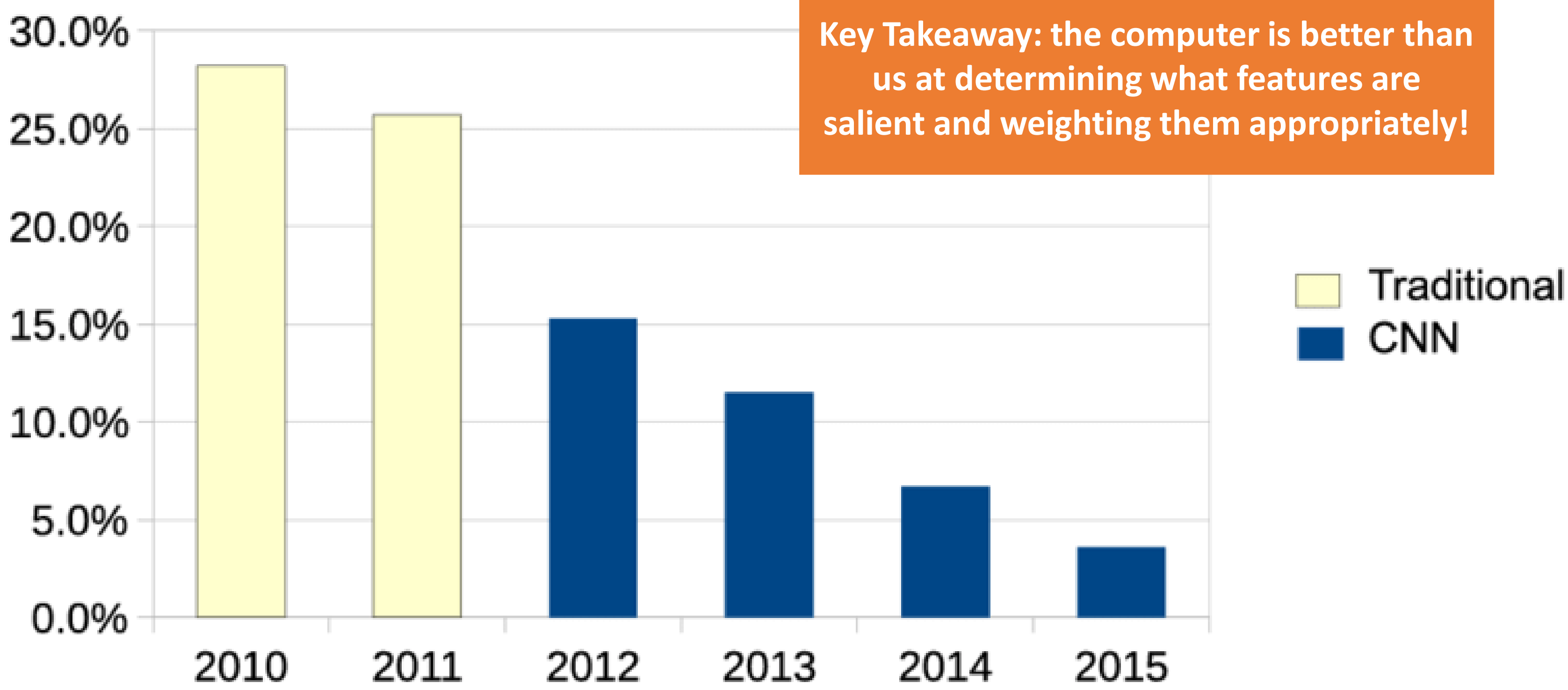
AlexNet: the first widely successful application of deep learning



https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

3

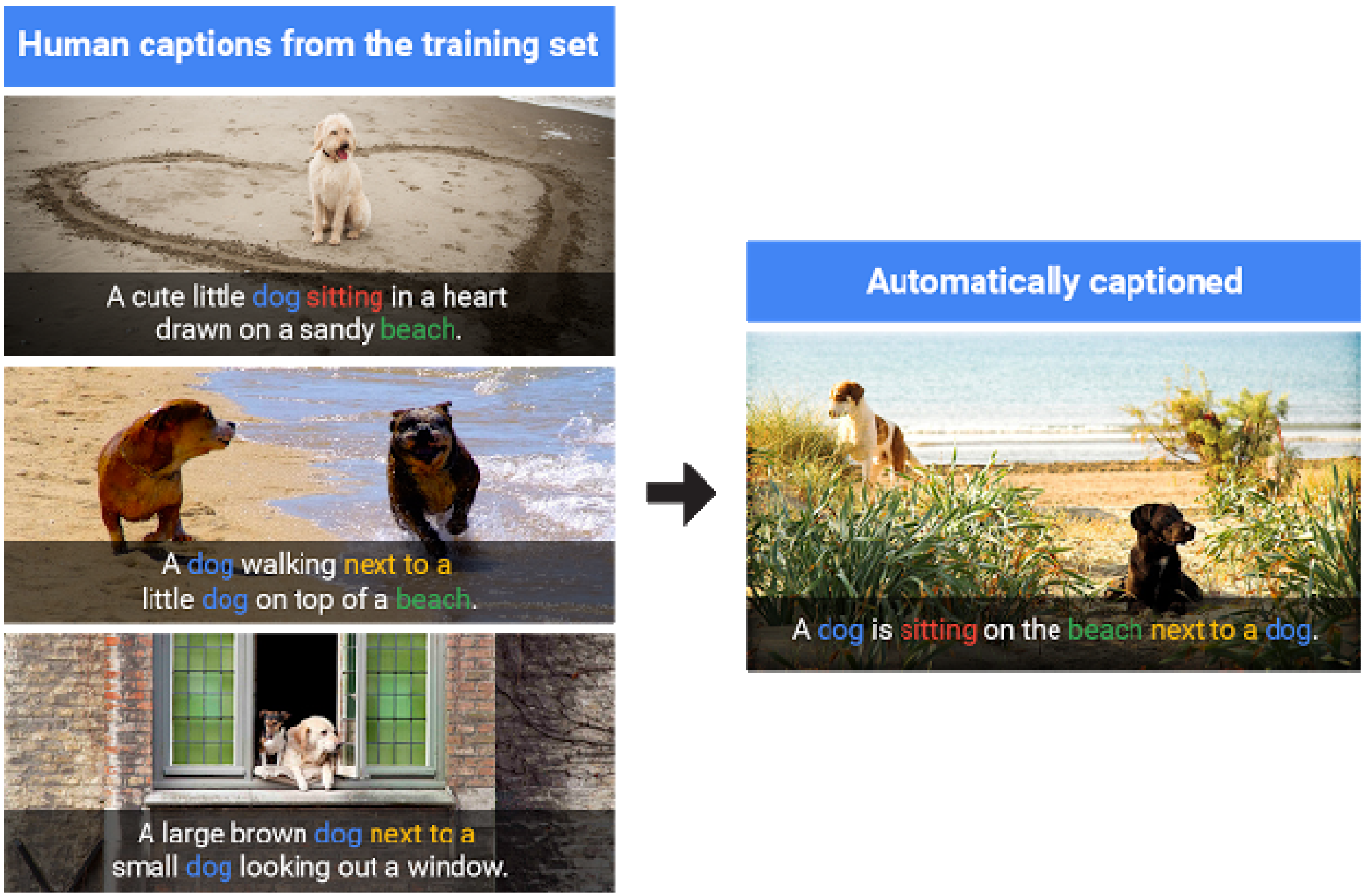
Deep learning automates the design, selection and extraction of features, with amazing results



Key Takeaway: the computer is better than us at determining what features are salient and weighting them appropriately!

https://www.researchgate.net/figure/Historical-top5-error-rate-of-the-annual-winner-of-the-ImageNet-image-classification_fig7_303992986

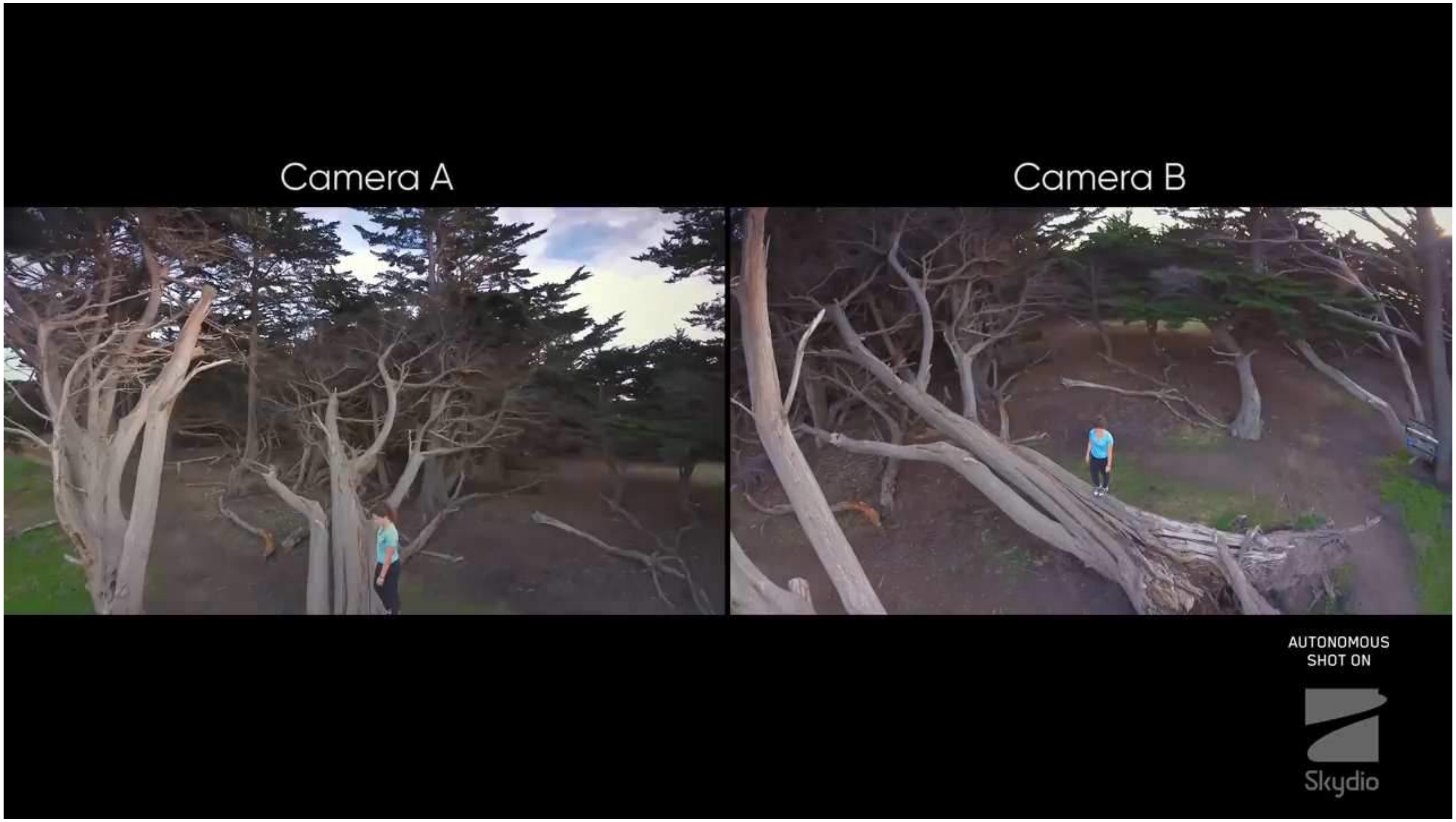
3 Google can now even automatically caption images!



3

The latest and greatest detectors can now find thousands of images in real-time

3 And can be used to track objects in real time

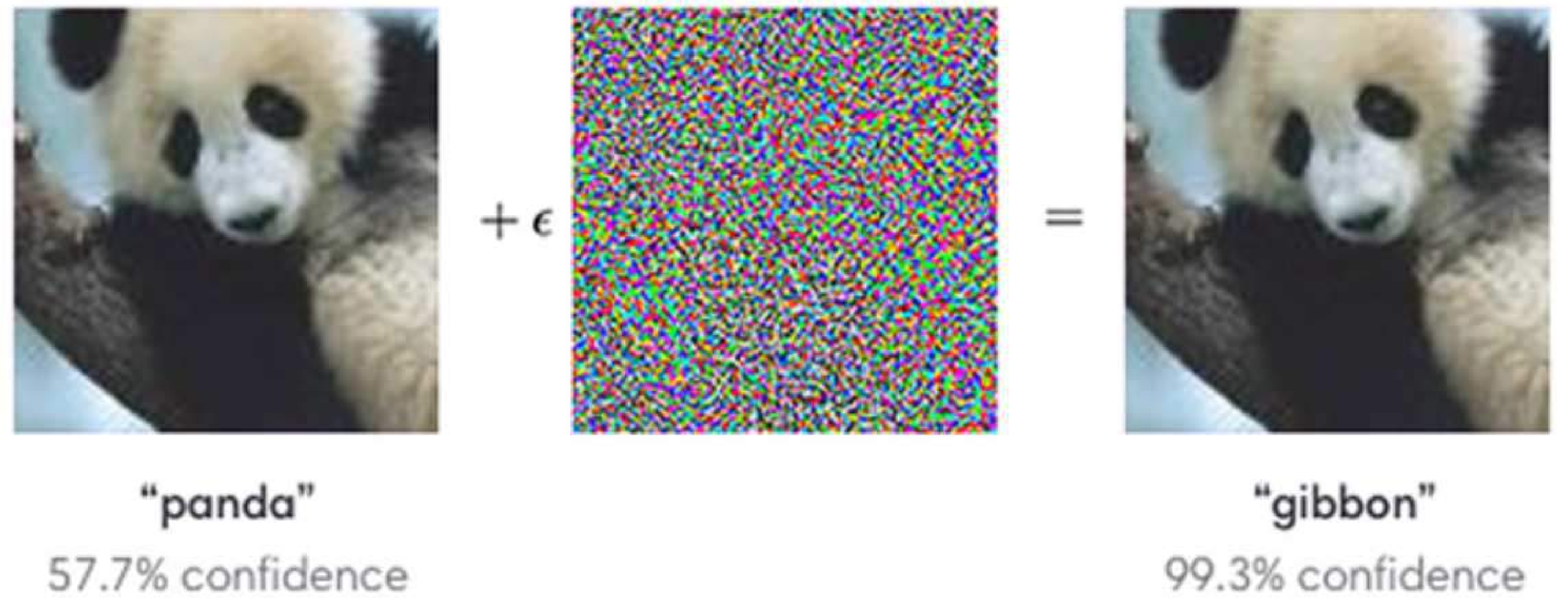


3 What might be the downside to using NNs?

3 For one, NNs can be tricked by adversarial markings

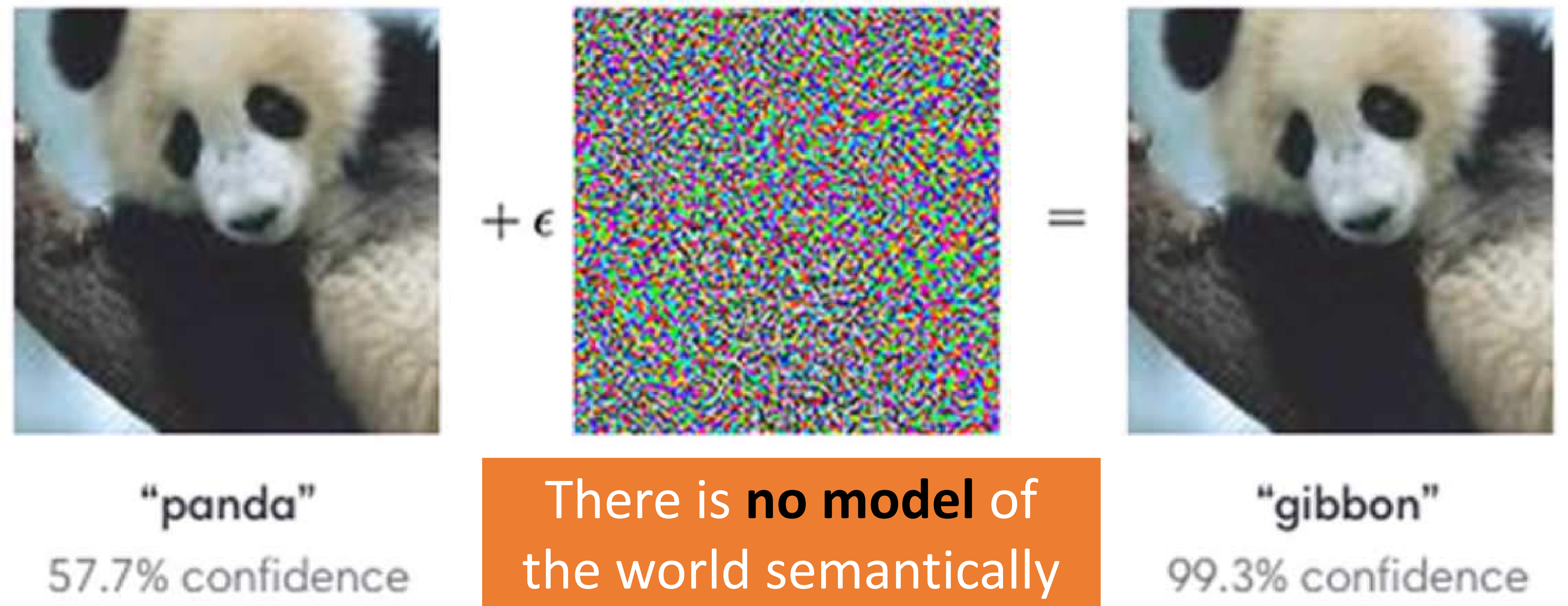
3 For one, NNs can be tricked by adversarial markings

Ackerman "Hacking the Brain With Adversarial Images"



3 For one, NNs can be tricked by adversarial markings

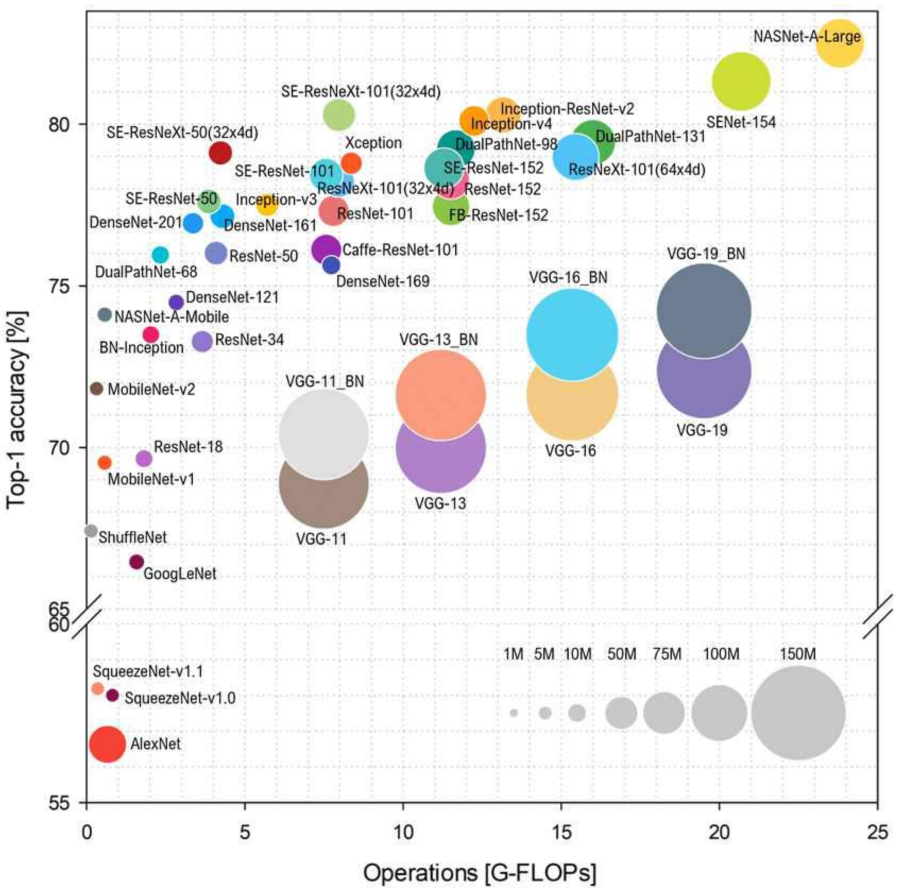
Ackerman “Hacking the Brain With Adversarial Images”

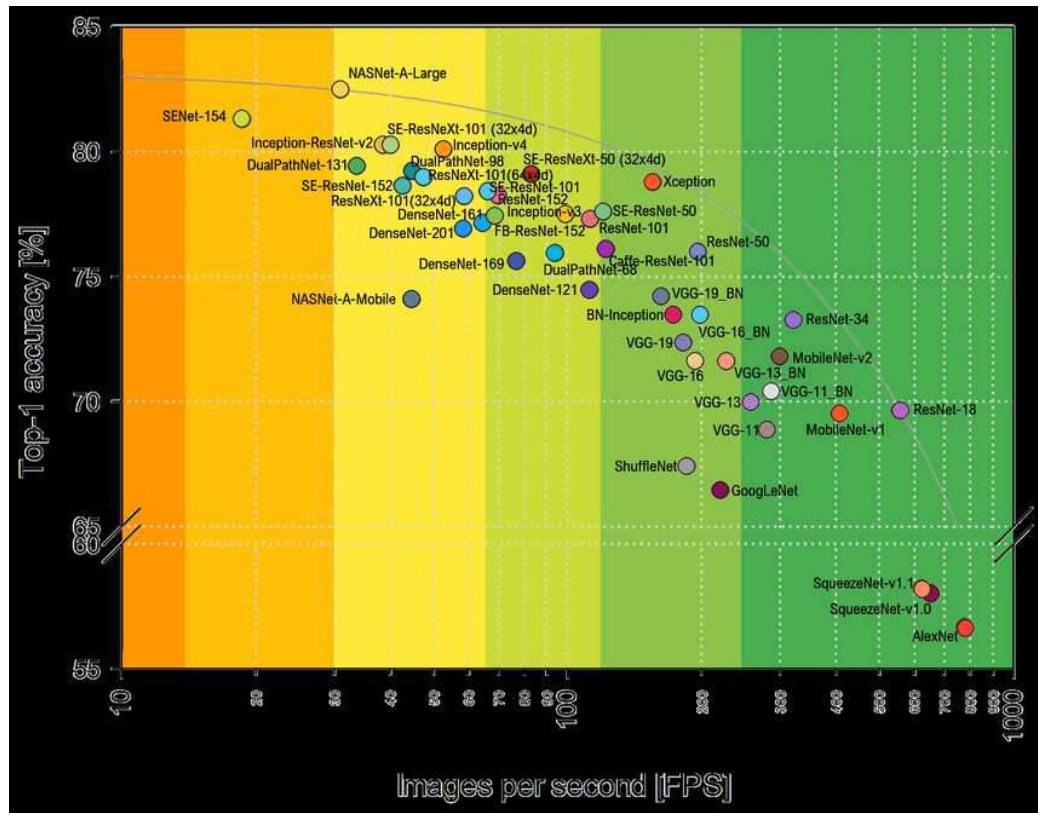


There is **no model** of the world semantically just mathematically

Second, (good) NN models are (often) large and expensive to train and compute

3

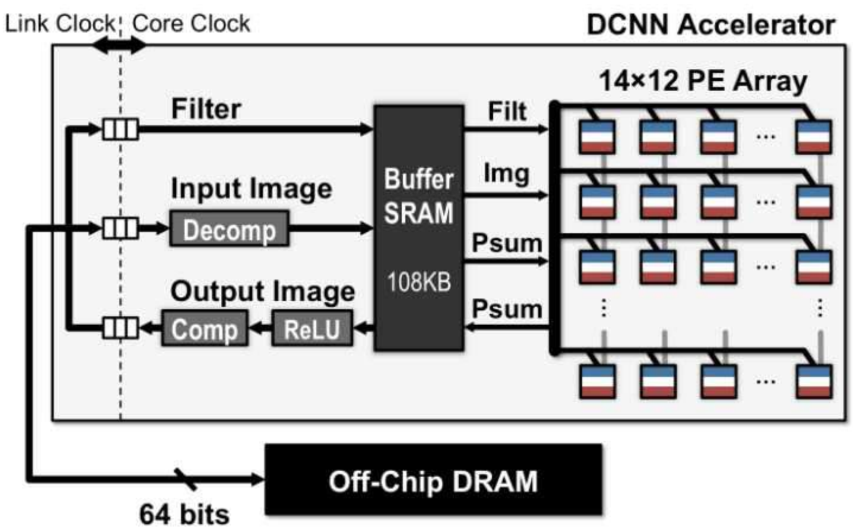




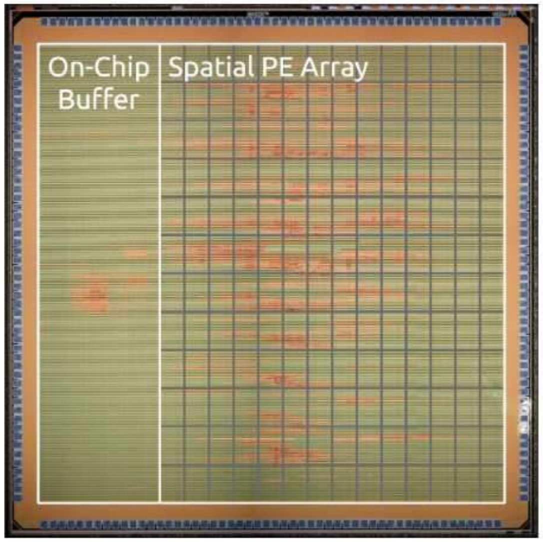
[Bianco et. al. Benchmark Analysis of Representative Deep Neural Network Architectures]

3

For this reason NNs (often) need accelerators to run online (and this is a very active area of research)



Eyeriss Architecture



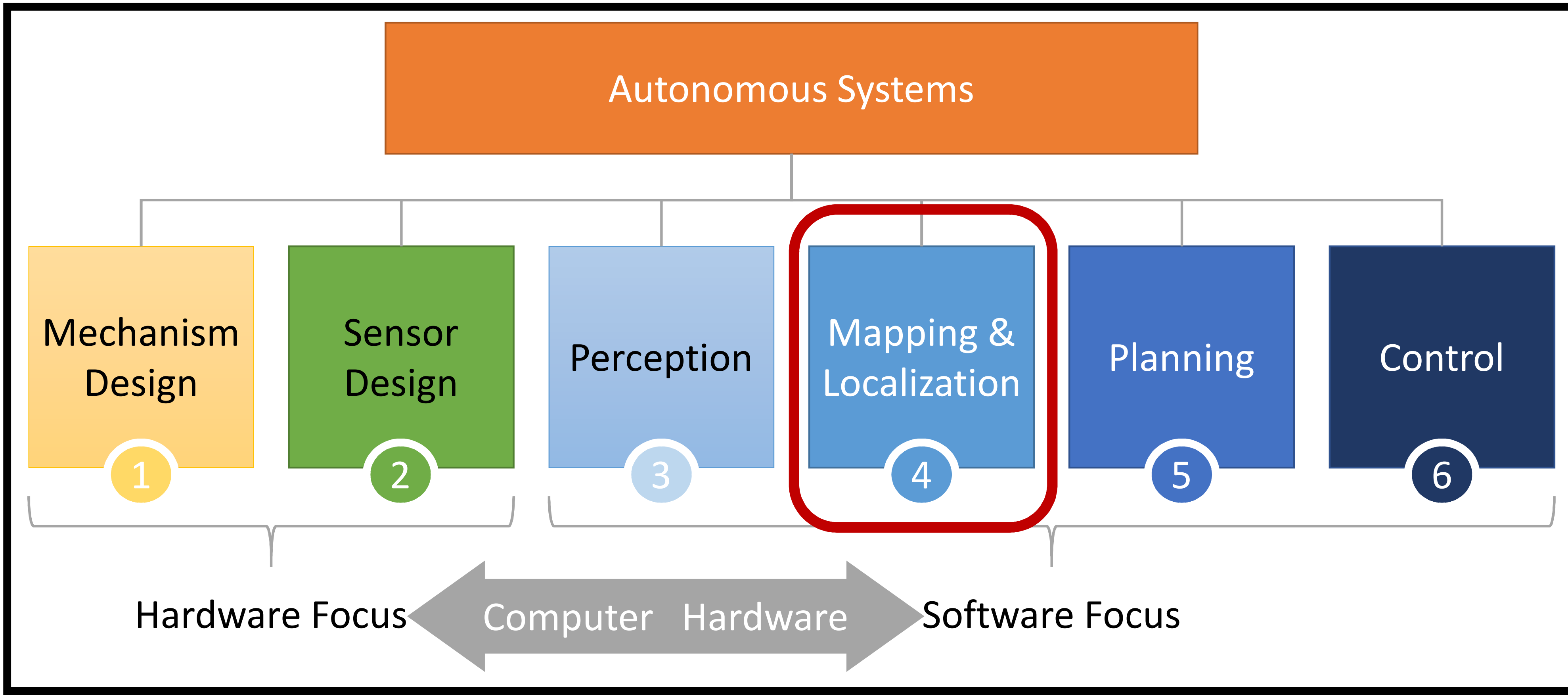
Die Photo

Can run in real time (35fps) at a 10X energy reduction over a mobile GPU (TX2?)!

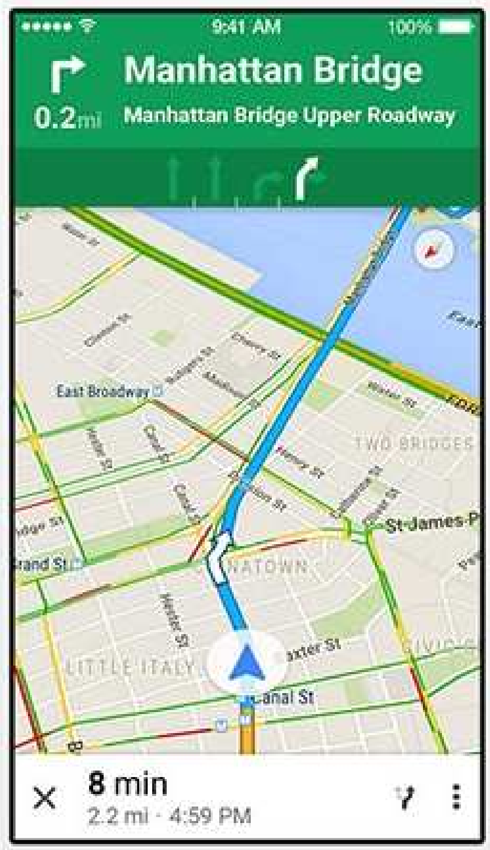
3 Key Takeaways:

1. As of today it seems like CNNs that automate the design and summary of salient features via convolution are the way to go
   - But/and will need specialized NN running on specialized accelerator chips to get them small enough to fit on small power constrained autonomous systems (e.g., small drones)
   - And we will need to find ways to secure them against attacks!
2. Also, other more targeted problems such as Stereo Depth seem to need accelerators!

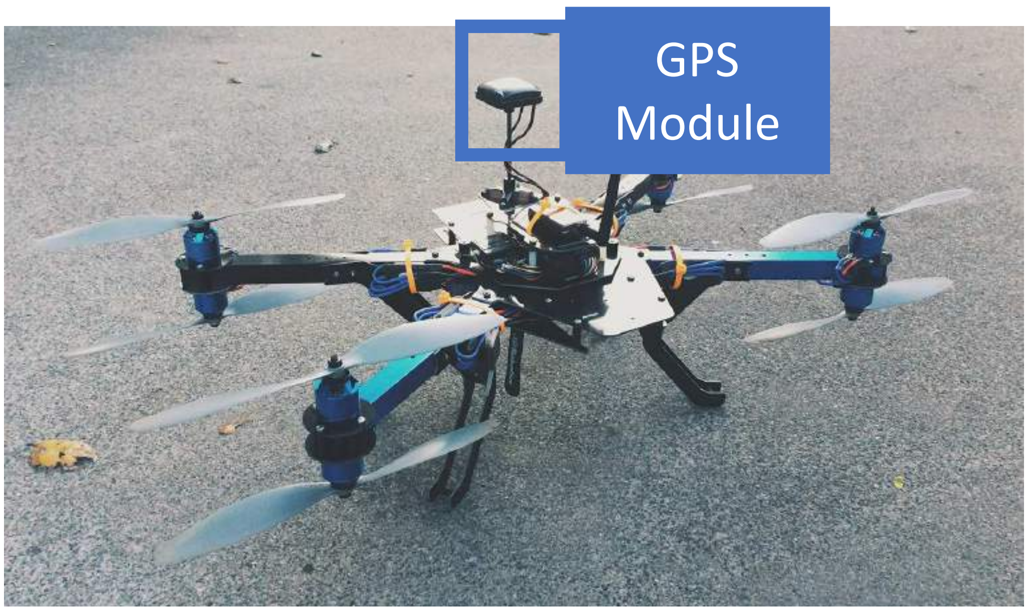
Autonomous Systems / Robotics is a BIG space

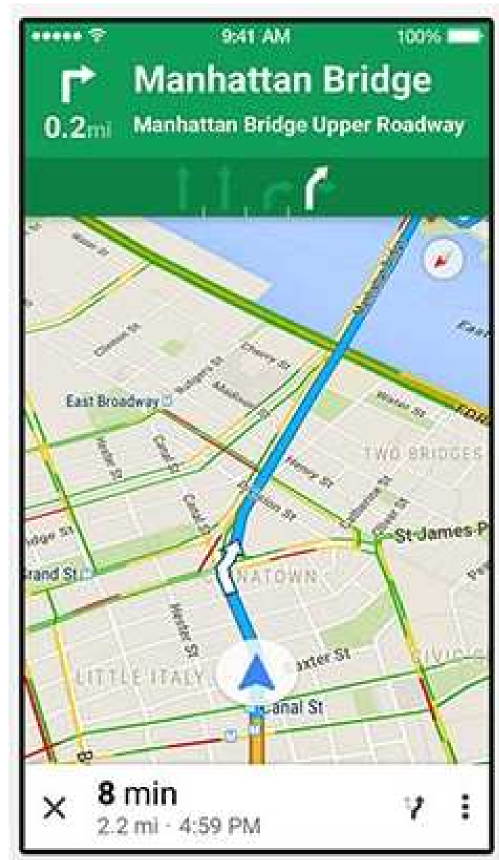


4 Mapping & Localization is the process of using perception information to understand where a robot is in the world

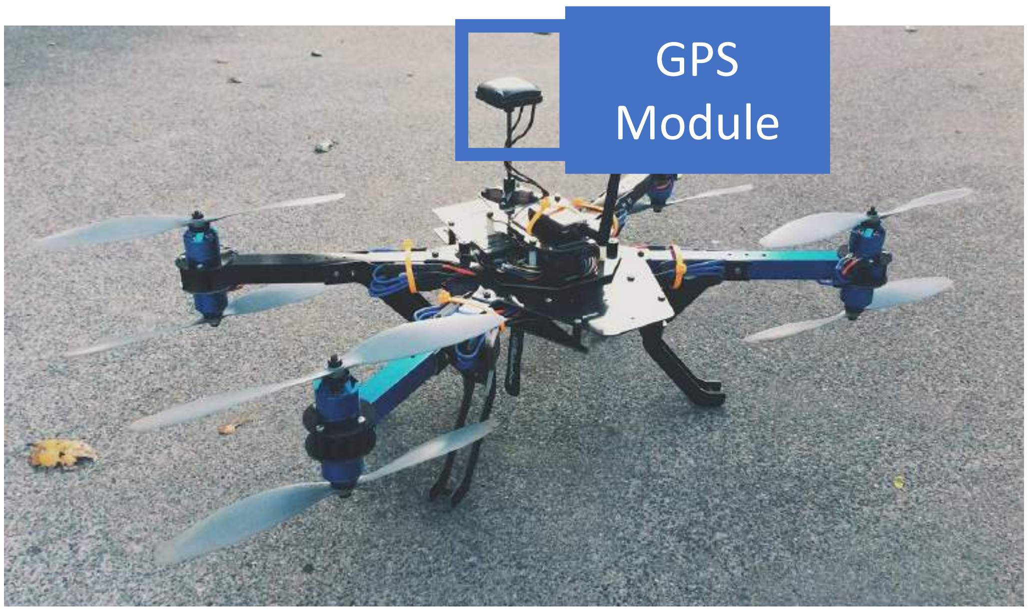


4 GPS provides a good idea of where a robot is globally…





4 … but isn't very accurate locally and requires a map



Two Problems

1. GPS is only accurate to O(10m)
2. GPS relies on already having a perfect map of the environment (unrealistic often)

4 … but isn't very accurate locally and requires a map

Localization Problem

Mapping Problem

Two Problems

1. GPS is only accurate to O(10m)
2. GPS relies on already having a perfect map of the environment (unrealistic often)

4 … but isn't very accurate locally and requires a map

Localization Problem

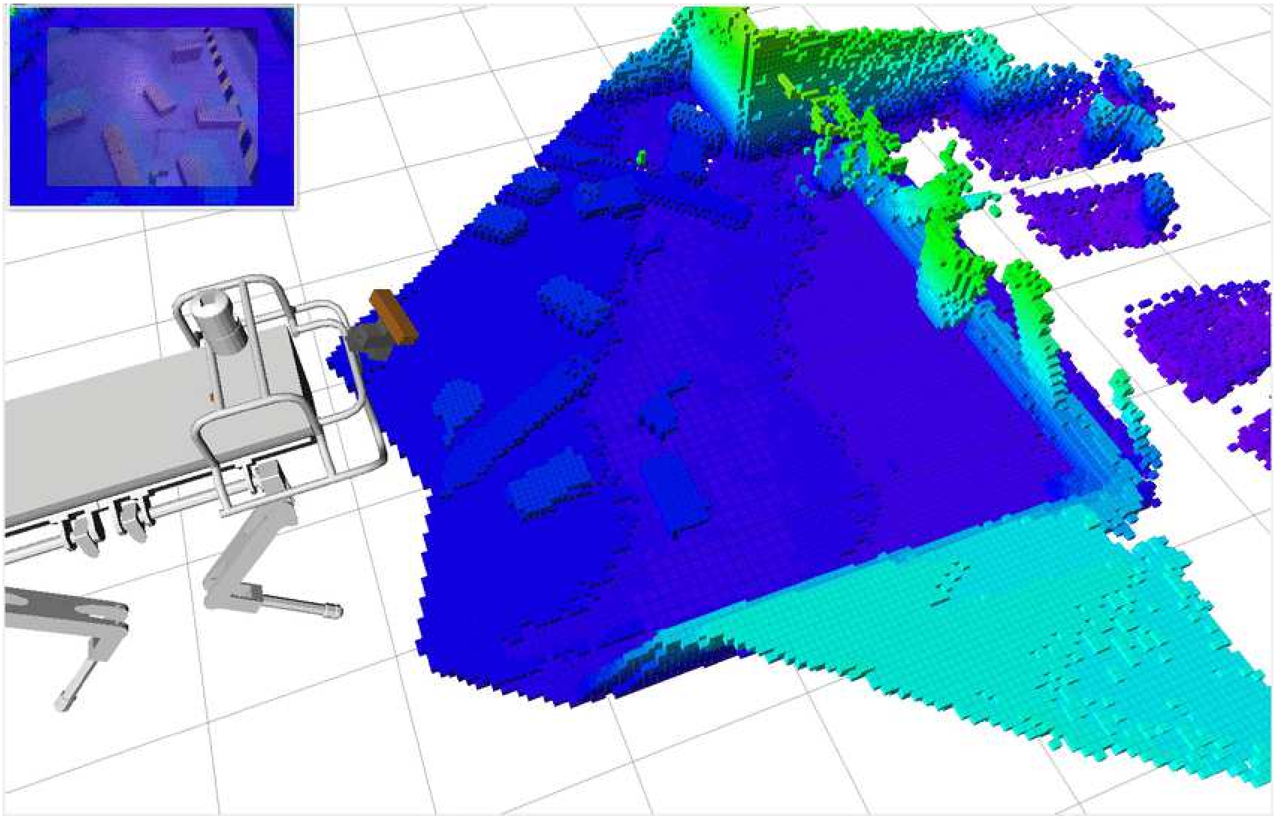
Mapping Problem

Two Problems

1. GPS is only accurate to O(10m)
2. GPS relies on already having a perfect map of the environment (unrealistic often)

4

We can use cameras and other sensors to measure the local environment but these sensors are also noisy

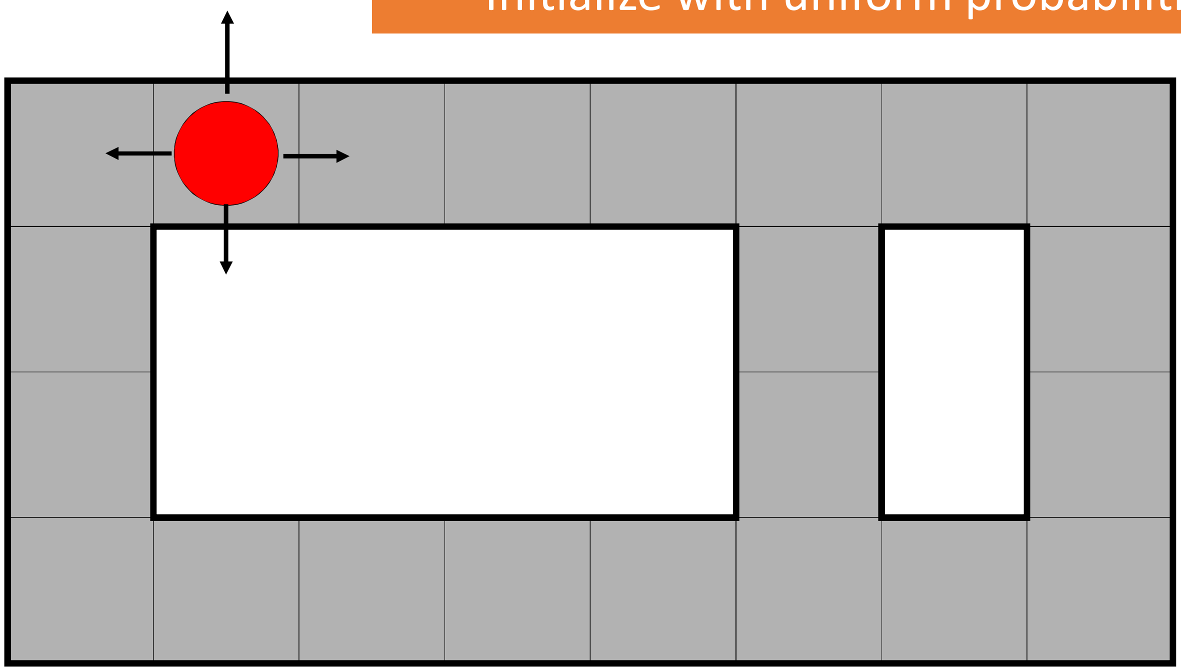


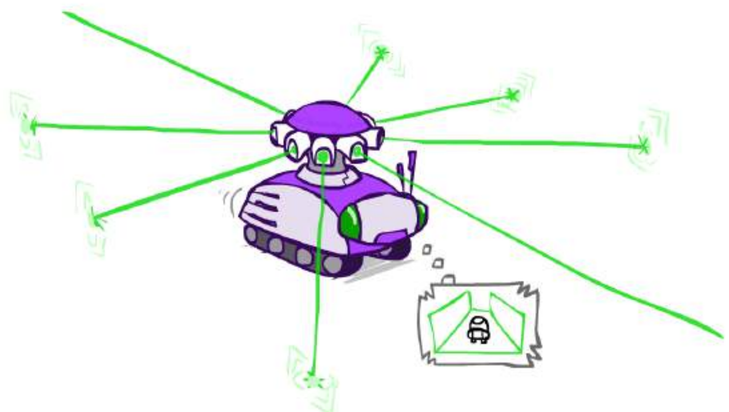
4 So what can we do?

Track the **Belief State** B_t	$\boldsymbol{B_t = p(x_t = X}\vert\text{Past States and Sensor Info})$

4 Let’s look at a concrete example of this in action

Initialize with uniform probabilities everywhere





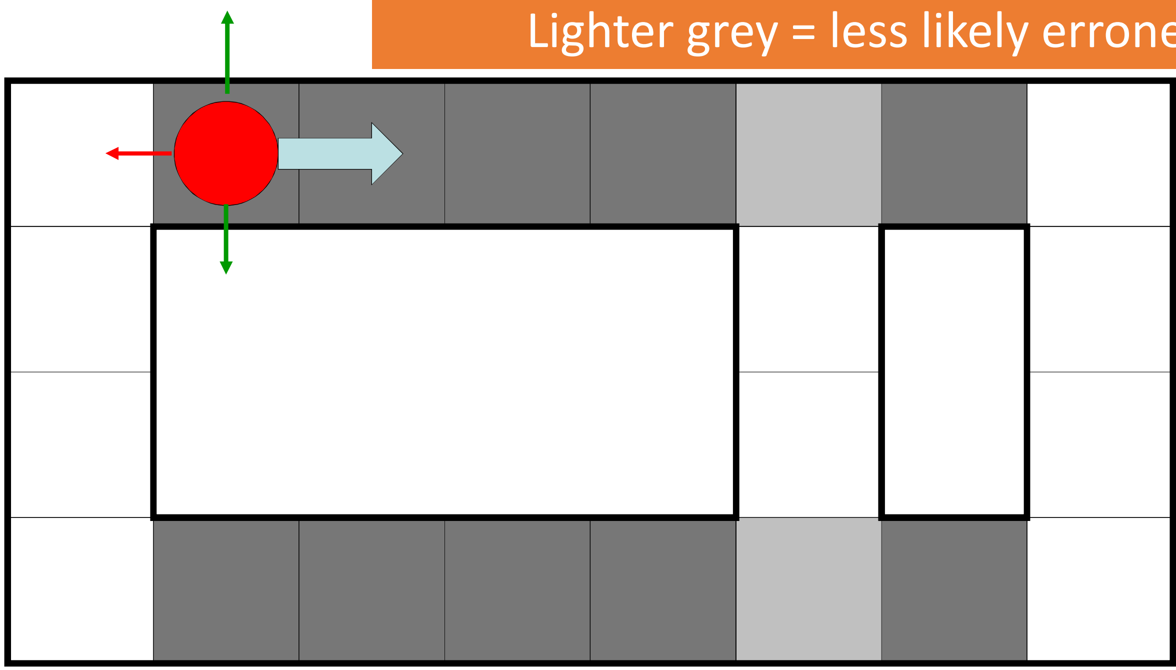
t=0

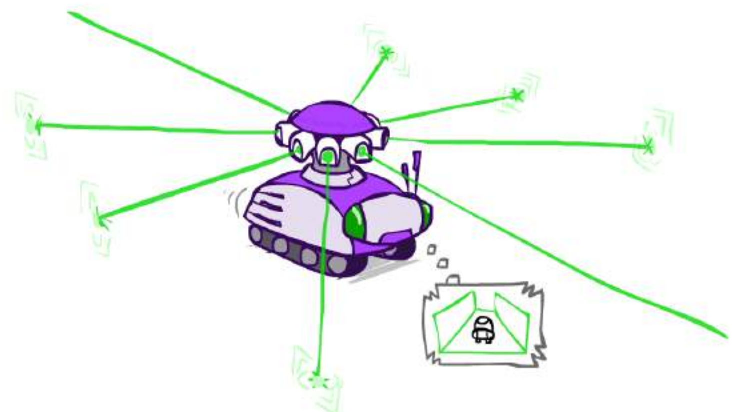
Probabilities 0 — 1

Example from Michael Pfeiffer

4 Let’s look at a concrete example of this in action

Take initial measurement and update
Lighter grey = less likely erroneous readings





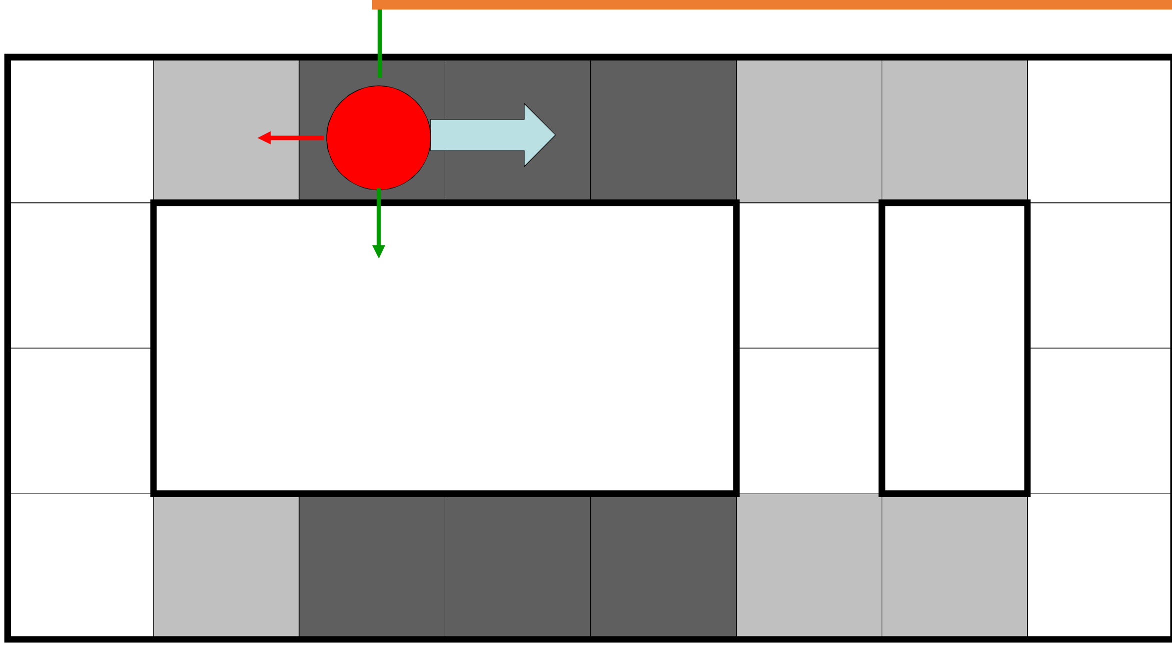
Probabilities
0
1

Example from Michael Pfeiffer

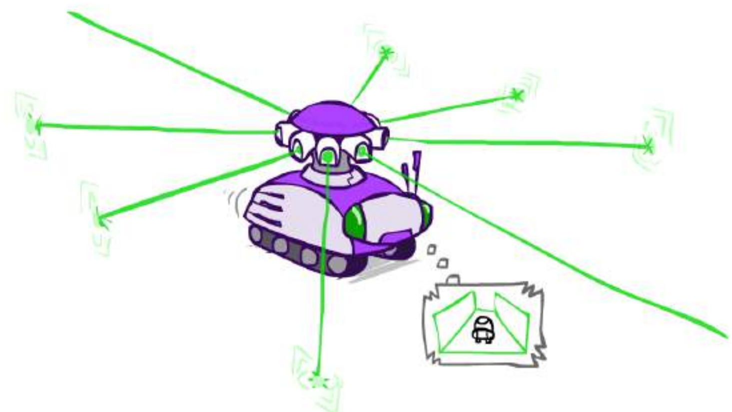
4

Let's look at a concrete example of this in action

Update for motion and next sensor reading



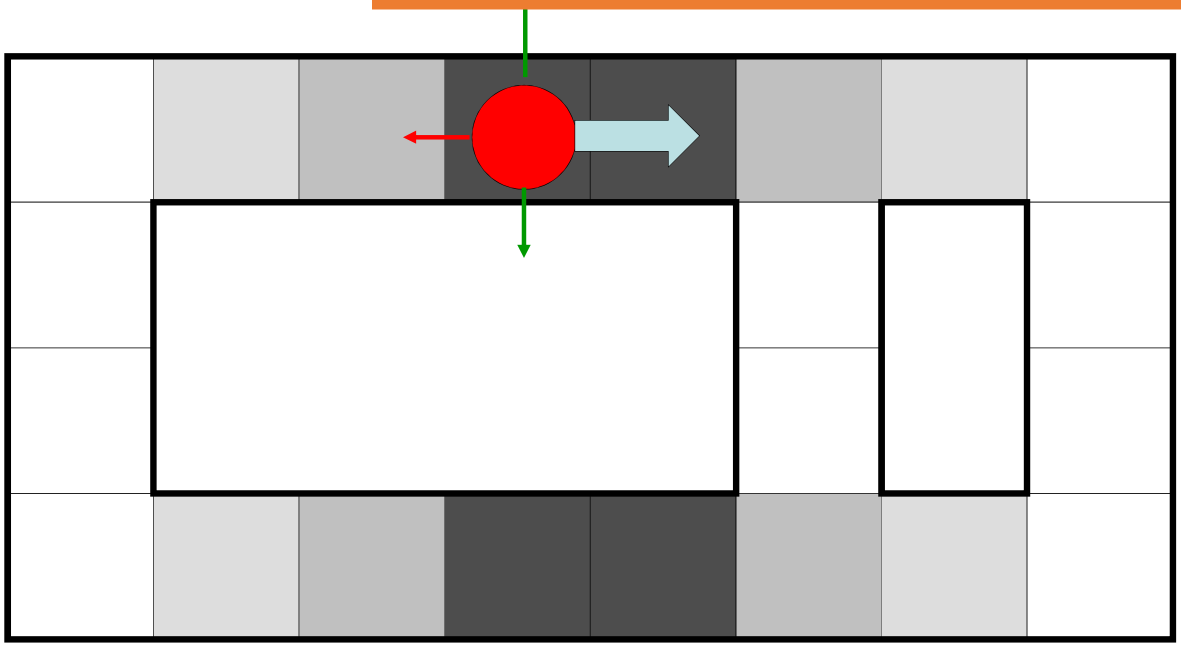
t=2



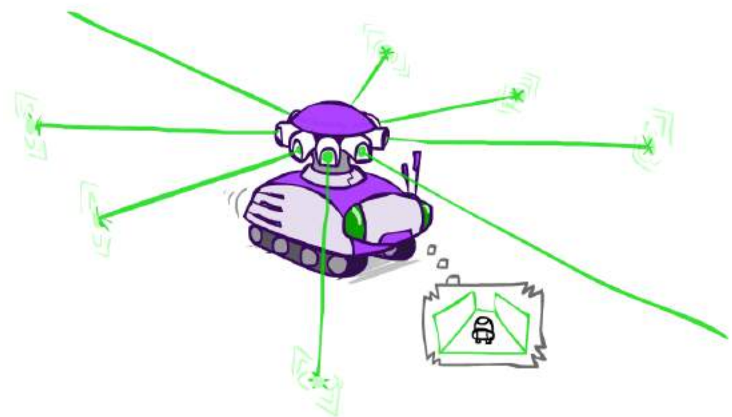
Example from Michael Pfeiffer

4 Let’s look at a concrete example of this in action

Update for motion and next sensor reading



t=3

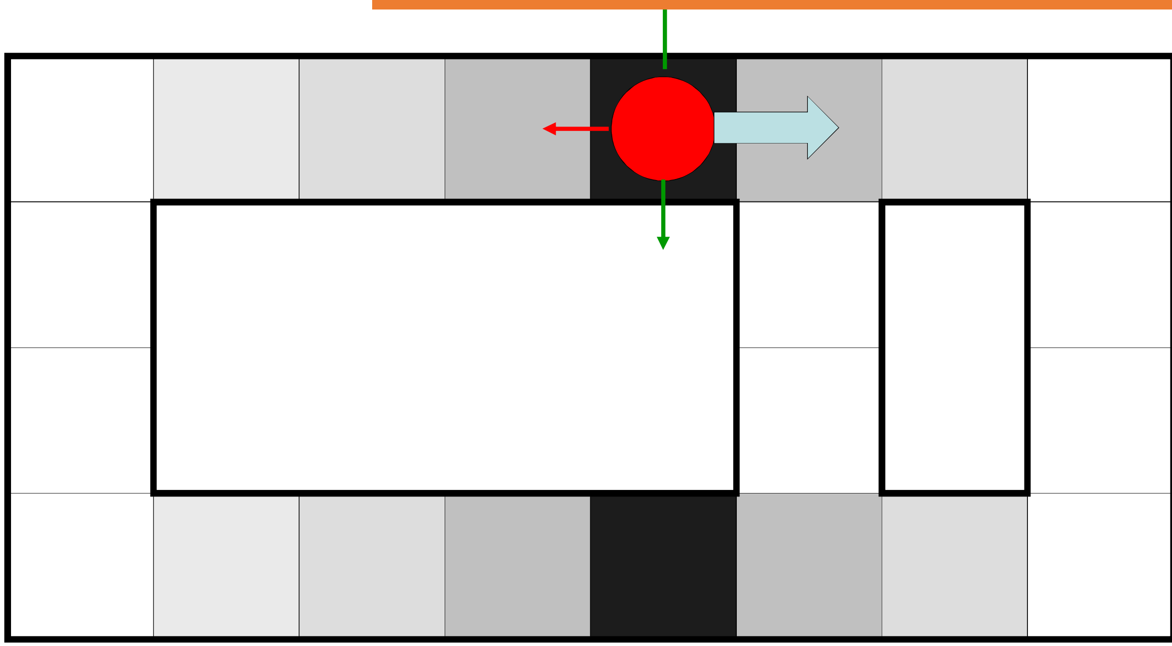


Probabilities 0 1

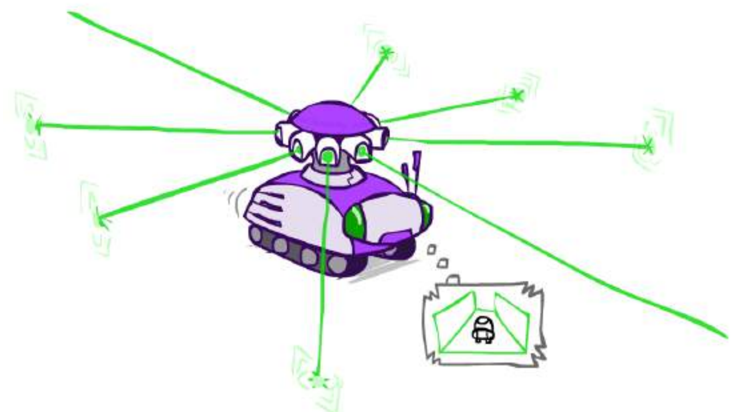
Example from Michael Pfeiffer

4 Let's look at a concrete example of this in action

Update for motion and next sensor reading



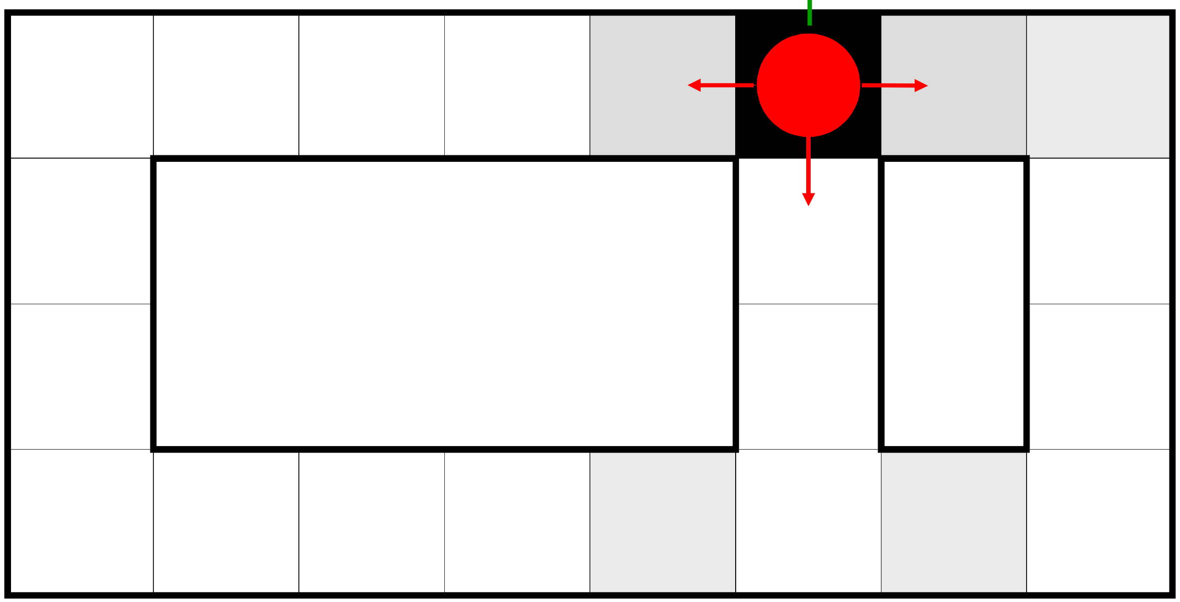
t=4



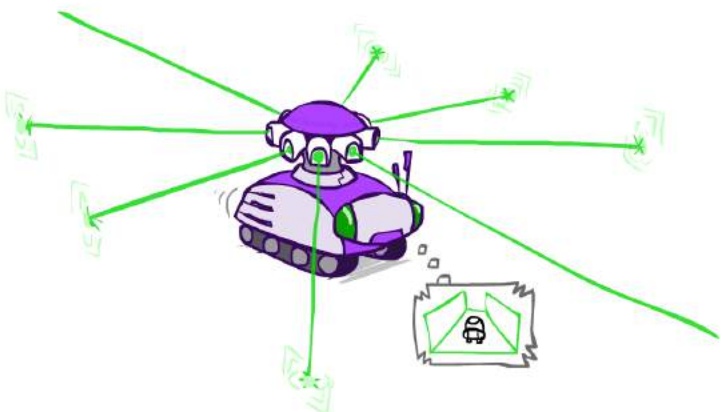
Example from Michael Pfeiffer

4 Let's look at a concrete example of this in action

Converge after motion and next sensor reading



t=5



Probabilities

Example from Michael Pfeiffer

4

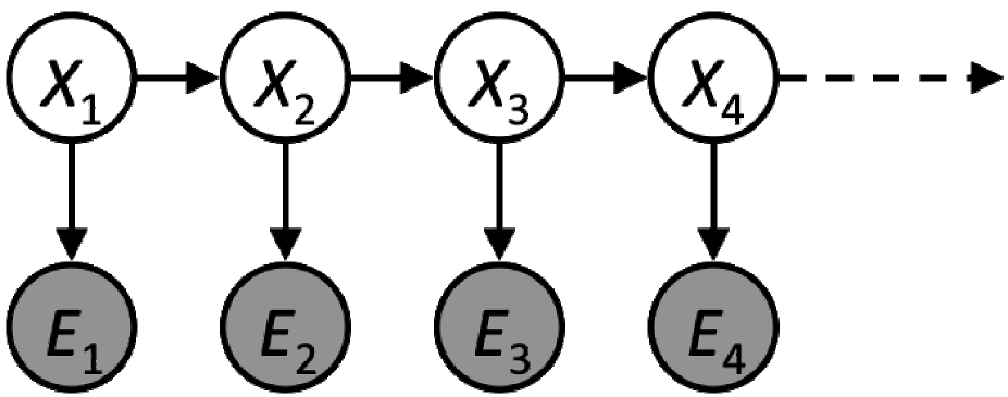
One approach is to model the probability of being in a given state with a Hidden Markov Model

Track the **Belief State** B_t

$$B_t = p(X_t | X_o, E_o \cdots E_{t-1})$$

Hidden Markov Model (HMM)

States X update in time but we only observe the effects E

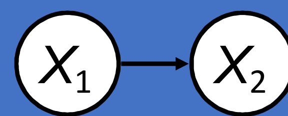


Report the **mean** of the **Belief State** (which is a probability distribution) as our current best estimate of the state

4

The Kalman Filter updates the belief state (a probability distribution) for the passage of time and for evidence

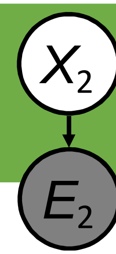
Time Update



$$P(x_{t+1}|x_0, e_1 \dots e_t) = \int P(x_t|x_0, e_1 \dots e_t) * P(x_{t+1}|x_t)$$

Based on a model of physics usually

Evidence Update

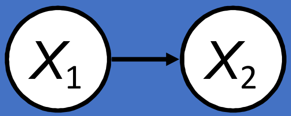


$$P(x_{t+1}|x_0, e_1 \dots e_{t+1}) \propto \int P(x_{t+1}|x_0, e_1 \dots e_t) * P(e_{t+1}|x_{t+1})$$

Based on a model of the sensor data usually

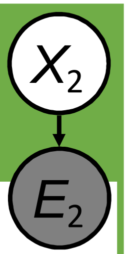
4 The Kalman Filter updates the belief state (a probability distribution) for the passage of time and for evidence

Time Update



$$P(x_{t+1}|x_0, e_1 \dots e_t) = \int P(x_t|x_0, e_1 \dots e_t) * P(x_{t+1}|x_t)$$

Evidence Update



$$P(x_{t+1}|x_0, e_1 \dots e_{t+1}) \propto \int P(x_{t+1}|x_0, e_1 \dots e_t) * P(e_{t+1}|x_{t+1})$$

Based on a model of

There are a variety of ways to compute this (and we'll highlight 4)

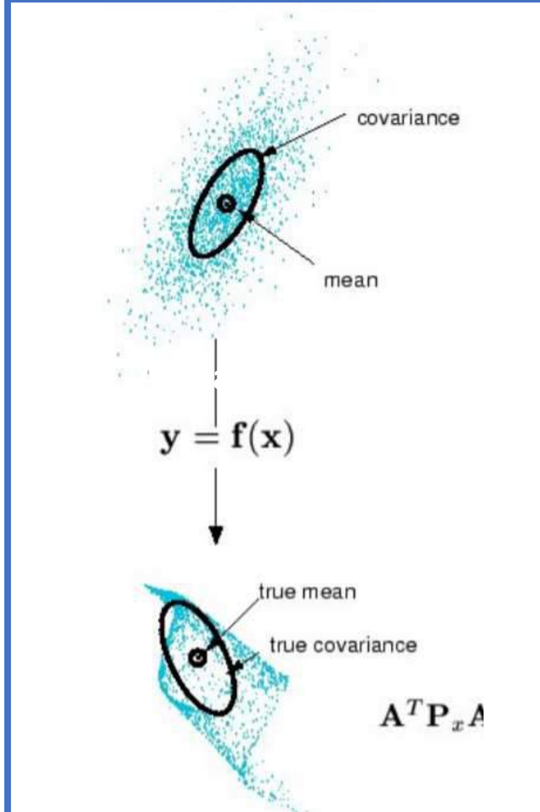
usually

4 There are four popular ways to compute this in practice

van der Merwe and Wan (2001)

1. Pass the full belief PDF through **nonlinear** equations for the motion update (physics) and the sensor update

Actual (sampling for clarity)

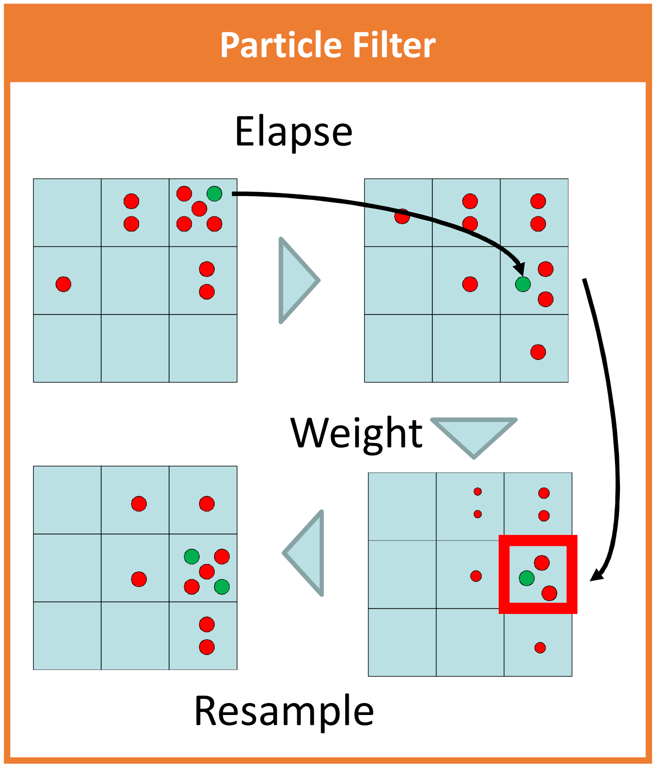


Most accurate but computationally very expensive (often intractable)

There are four popular ways to compute this in practice

Berkely AI Material and Scott Kuindersma

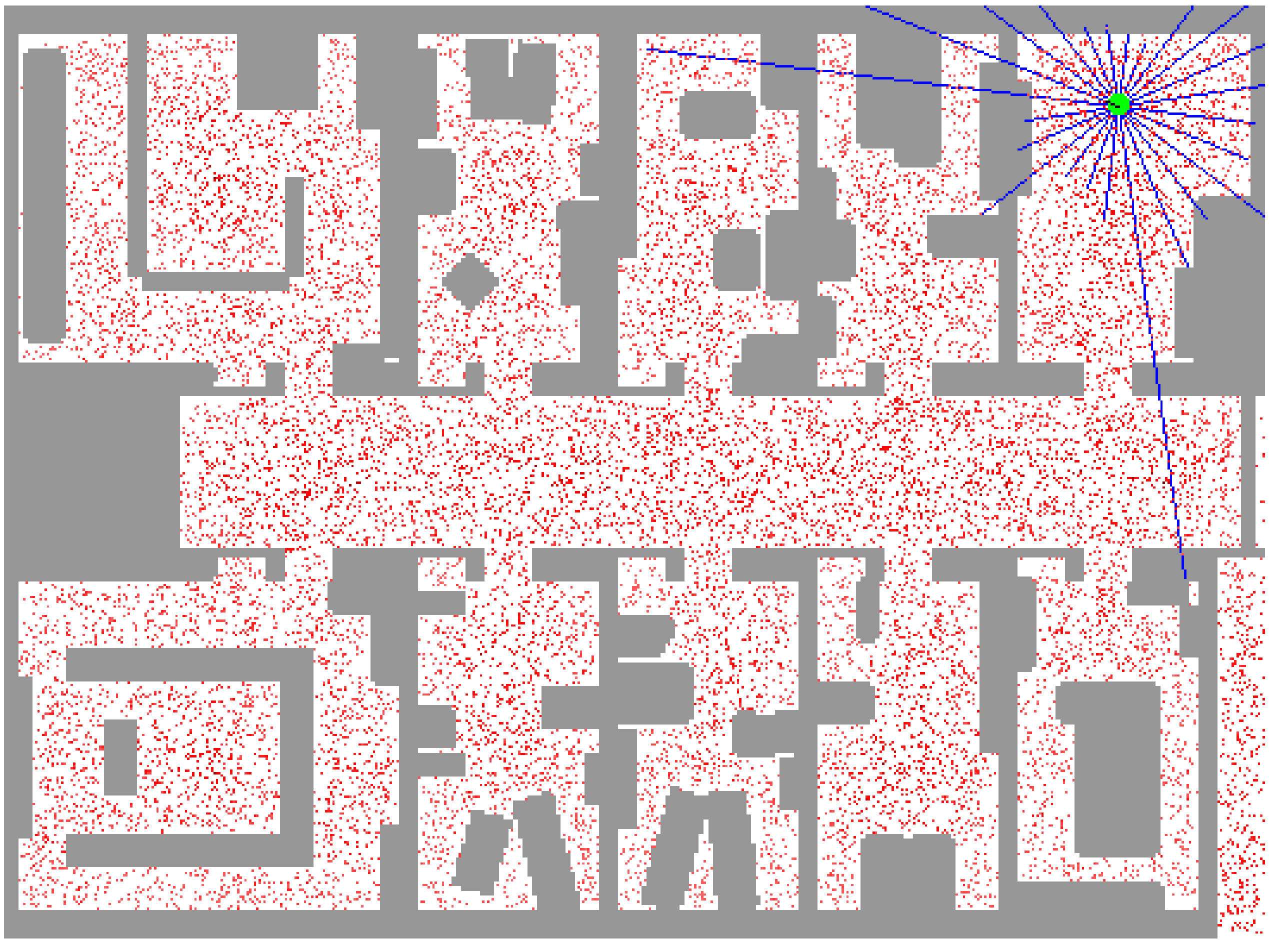
2. **Pass many samples** through the **nonlinear** equations for the motion update (physics) and the sensor update and use the samples as a discrete approximation of the probability distribution



Can be very accurate but also computation-ally expensive (lots of particles)

4 There are four popular ways to compute this in practice

Dieter Fox



4 There are four popular ways to compute this in practice

What if we don’t want to sample?

$X_1 \rightarrow X_2$

$$P(x_{t+1}|x_0, e_1 \dots e_t) = \int P(x_t|x_0, e_1 \dots e_t) * P(x_{t+1}|x_t)$$

4 There are four popular ways to compute this in practice

Lets do some math for a minute

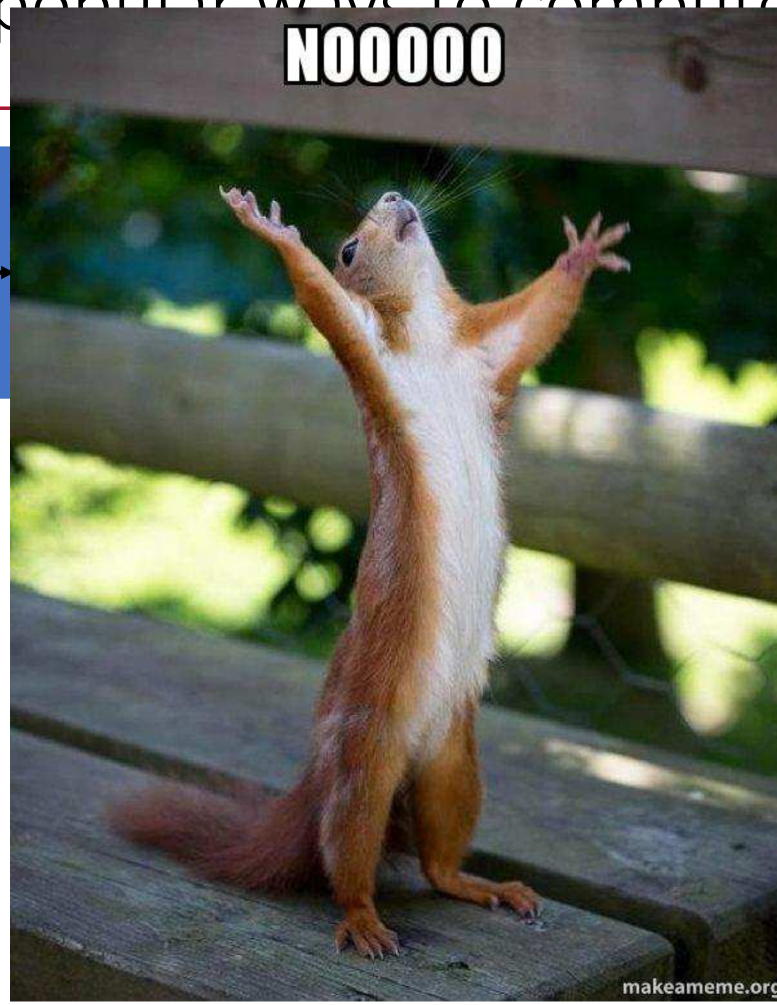
$X_1 \rightarrow X_2$

$$P(x_{t+1}|x_0, e_1 \dots e_t) = \int P(x_t|x_0, e_1 \dots e_t) * P(x_{t+1}|x_t)$$

4 There are four popular ways to compute this in practice

Lets do some math for a minute

$X_1 \rightarrow$



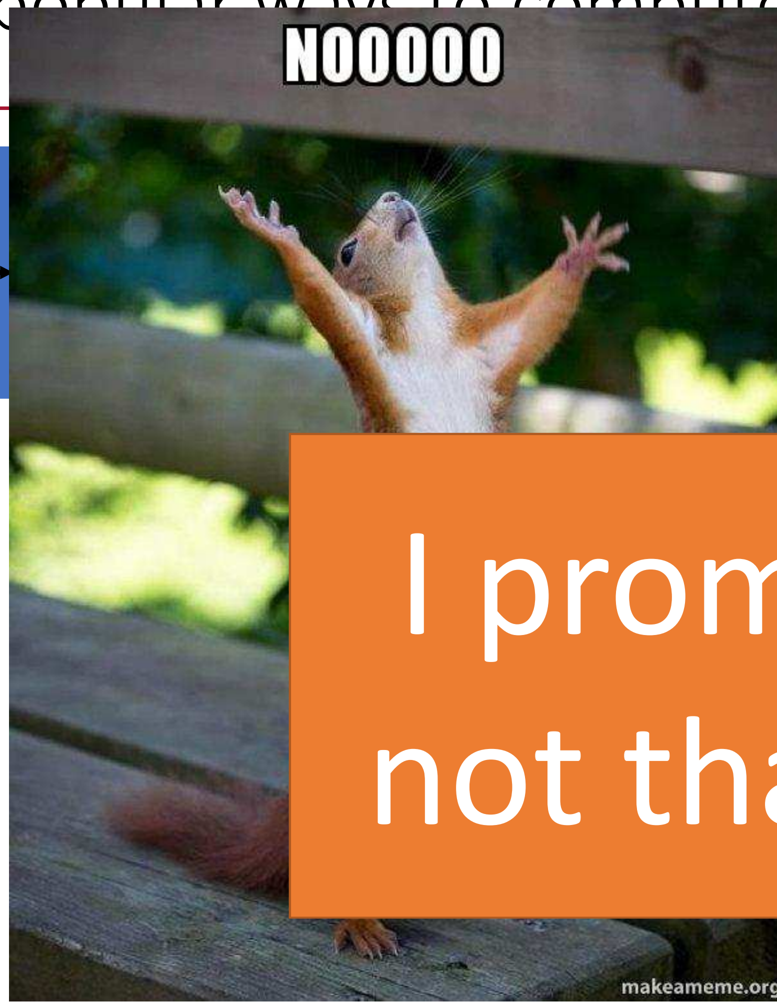
$\ldots_t | x_0, e_1 \ldots e_t) * P(x_{t+1} | x_t)$

4 There are four popular ways to compute this in practice

Lets do some math for a minute

$X_1 \rightarrow$

$\ldots_t | x_0, e_1 \ldots e_t) * P(x_{t+1} | x_t)$



I promise its not that bad!

4 There are four popular ways to compute this in practice

Lets do some math for a minute

X_1

$x_0, e_1 \ldots e_t) * P(x_{t+1}|x_t)$

ise its
t bad!

4 There are four popular ways to compute this in practice

http://www.cs.columbia.edu/~liulp/pdf/linear_normal_dist.pdf

Lets do some math for a minute

$X_1 \rightarrow X_2$

$$P(x_{t+1}|x_0, e_1 \dots e_t) = \int P(x_t|x_0, e_1 \dots e_t) * P(x_{t+1}|x_t)$$

Suppose $\mathbf{x} \sim \mathcal{N}(\mu_x, \Sigma_x)$ and $\mathbf{y} = A\mathbf{x} + \mathbf{b}$, where $\mathbf{b} \sim \mathcal{N}(0, \Sigma_b)$.

$$\begin{aligned} \mu_y &= E[\mathbf{y}] = E[A\mathbf{x} + \mathbf{b}] = AE[\mathbf{x}] + E[\mathbf{b}] = A\mu_x, \\ \Sigma_y &= \mathrm{Var}(A\mathbf{x} + \mathbf{b}) = \mathrm{Var}(A\mathbf{x}) + \mathrm{Var}(\mathbf{b}) = A\Sigma_x A^T + \Sigma_b, \end{aligned}$$

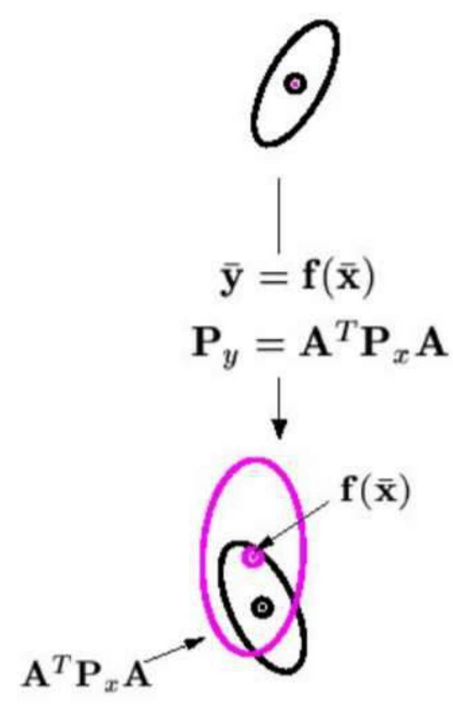
Well if we represent the transition from one state to the next by a linear equation (just linearize physics) and represent P(x) as Gaussian then we can just use this simple linear transformation to do all of the math super fast!

4 There are four popular ways to compute this in practice

van der Merwe and Wan (2001)

3. Assume the belief PDF is **Gaussian** and pass it through **linearized** equations for the motion update (physics) and the sensor update

Extended Kalman Filter - EKF



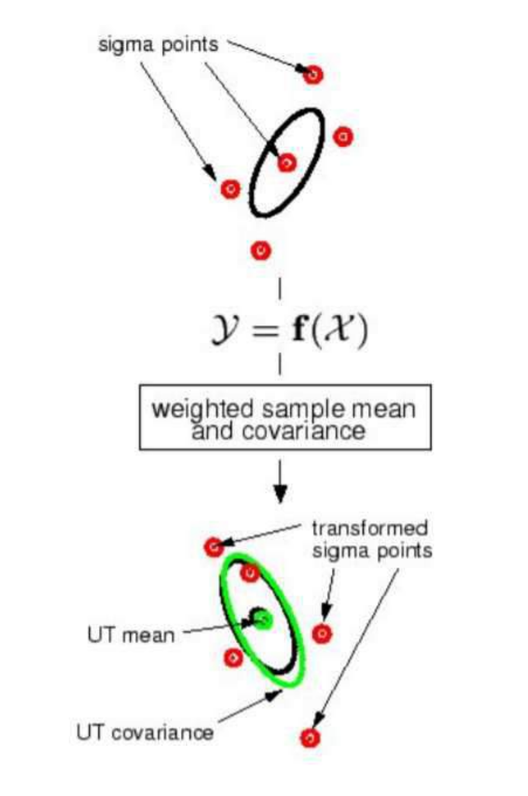
Simplest and least accurate as assumes Gaussian + linear

4 There are four popular ways to compute this in practice

van der Merwe and Wan (2001)

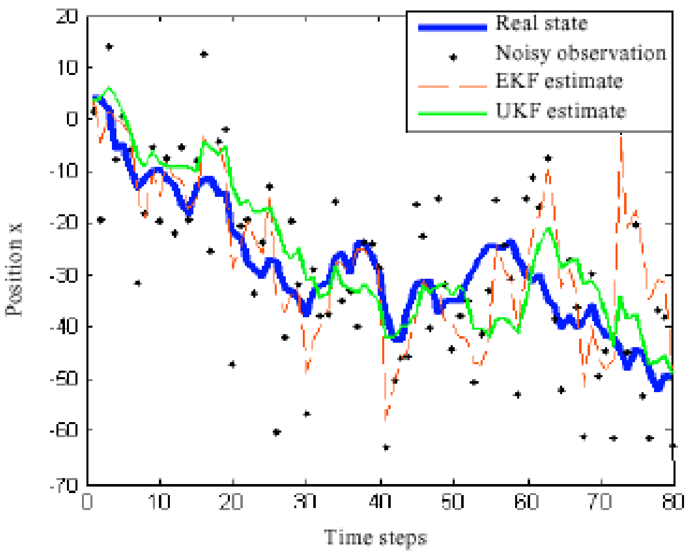
4. Assume the belief PDF is **Gaussian** and **pass limited samples** through the **nonlinear** equations for the motion update (physics) and the sensor update and reconstruct the Gaussian on the other side

Unscented Kalman Filter - UKF



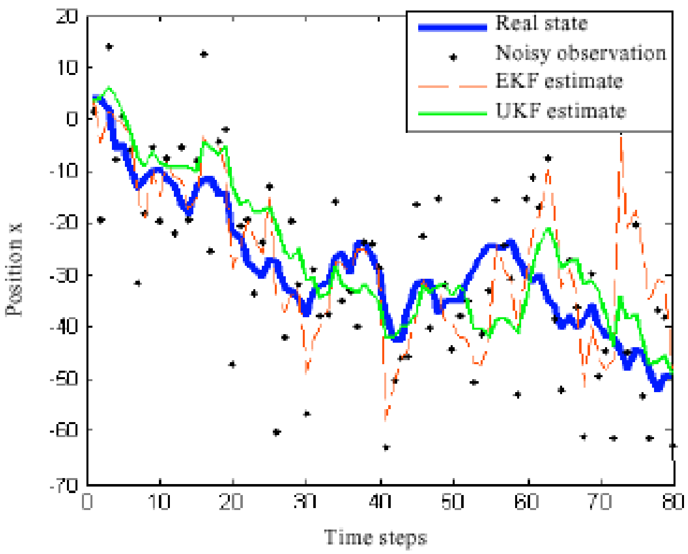
Moderately accurate but assumes Gaussian

4 There are four popular ways to compute this in practice



Hassanzadeh and Fallah 2008

4 There are four popular ways to compute this in practice



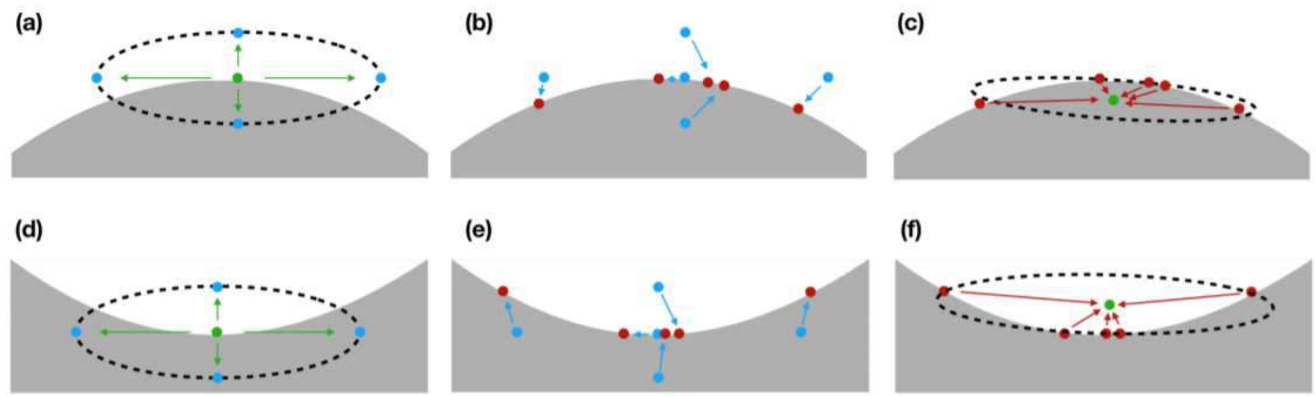


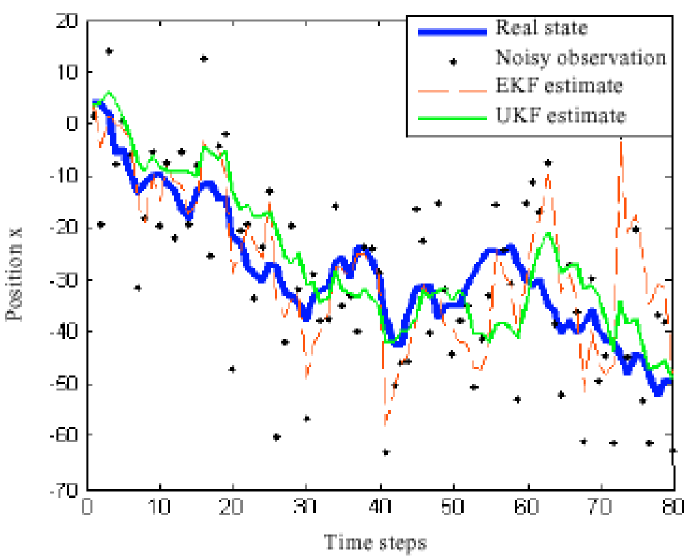
Fig. 1: Two illustrations of fundamental problems associated with the UKF in the presence of the inequalities associated with contact. When sample points are generated (a and d) samples are either infeasible or have different contact modes than the mean estimate. In the first sequence (a-c) the resulting estimate (c) is infeasible even though all of the samples are feasible. In the second sequence (d-f) the resulting estimate (f) is feasible, but has a contact mode that is different from any of the individual sample points. In our experience this is the more common behavior, biasing the estimate away from the contact manifold.

Hassanzadeh and Fallah 2008

Varin and Kuindersma 2018

4 There are four popular ways to compute this in practice

Modeling is helpful to reduce computation but **No Free Lunch**!



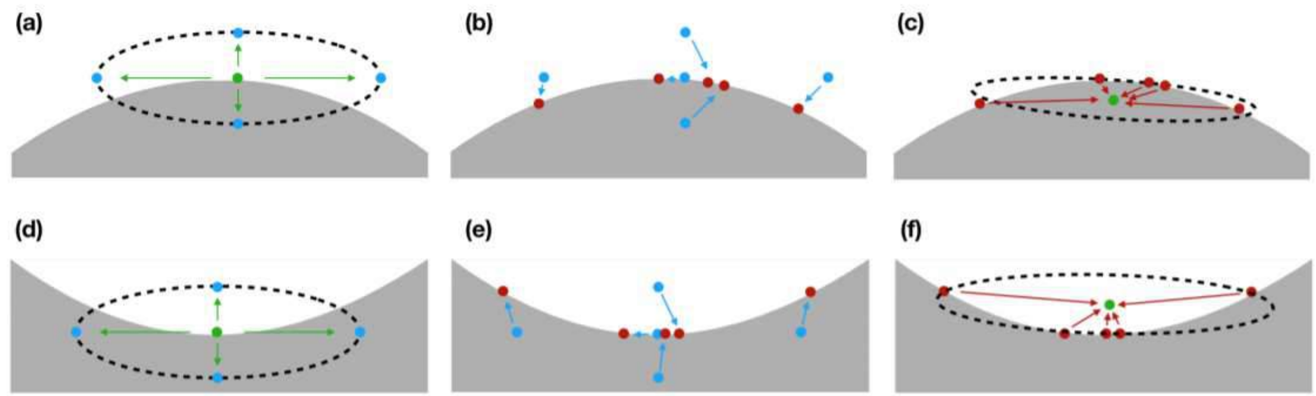


Fig. 1: Two illustrations of fundamental problems associated with the UKF in the presence of the inequalities associated with contact. When sample points are generated (a and d) samples are either infeasible or have different contact modes than the mean estimate. In the first sequence (a-c) the resulting estimate (c) is infeasible even though all of the samples are feasible. In the second sequence (d-f) the resulting estimate (f) is feasible, but has a contact mode that is different from any of the individual sample points. In our experience this is the more common behavior, biasing the estimate away from the contact manifold.

Hassanzadeh and Fallah 2008

Varin and Kuindersma 2018

4 There are four popular ways to compute this in practice

1. Pass the full belief PDF through **nonlinear** equations for the motion update (physics) and the sensor update	Most accurate but computationally very expensive (often intractable)
2. Pass **many samples** through the **nonlinear** equations for the motion update (physics) and the sensor update and use the samples as a discrete approximation of the probability distribution	Can be very accurate but can also be computationally expensive (**Particle Filter**)

4 There are four popular ways to compute this in practice

1. Pass the full belief PDF through **nonlinear** equations for the motion update (physics) and the sensor update	Most accurate but computationally very expensive (often intractable)
2. Pass **many samples** through the **nonlinear** equations for the motion update (physics) and the sensor update and use the samples as a discrete approximation of the probability distribution	Can be very accurate but can also be computationally expensive (**Particle Filter**)
3. Assume the belief PDF is **Gaussian** and pass it through **linearized** equations for the motion update (physics) and the sensor update	Simplest and least accurate as assumes linear (**Extended Kalman Filter - EKF**)
4. Assume the belief PDF is **Gaussian** and **pass limited samples** through the **nonlinear** equations for the motion update (physics) and the sensor update and reconstruct the Gaussian on the other side	Moderately accurate but assumes Gaussian (**Unscented Kalman Filter - UKF**)

4 But what if we don't have a map of the environment?

Localization Problem

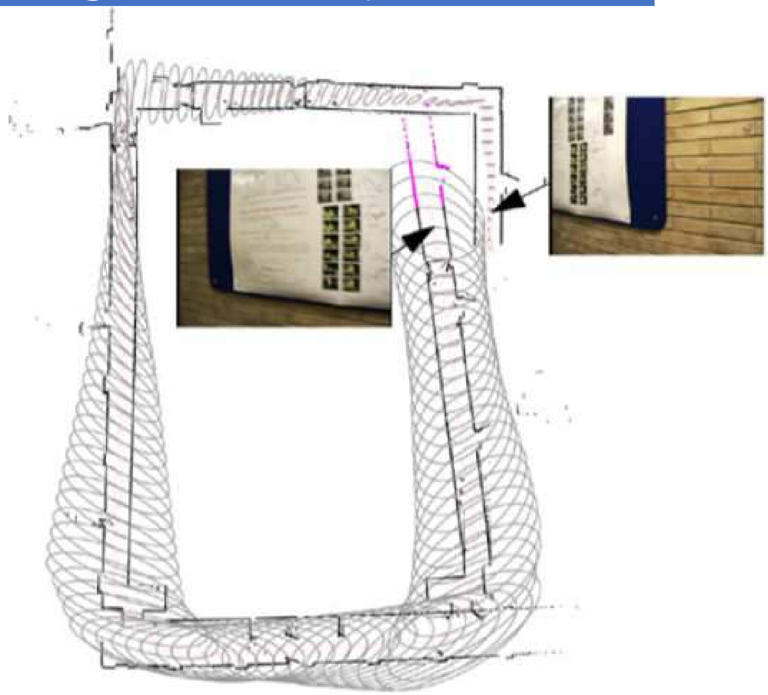
Mapping Problem

Two Problems

1. GPS is only accurate to O(10m)
2. GPS relies on already having a perfect map of the environment (unrealistic often)

4 But what if we don't have a map of the environment? Enter Simultaneous Localization and Mapping (SLAM)

Essentially just additionally tracking the belief of **landmarks** in the environment (walls, buildings, trees, etc.)



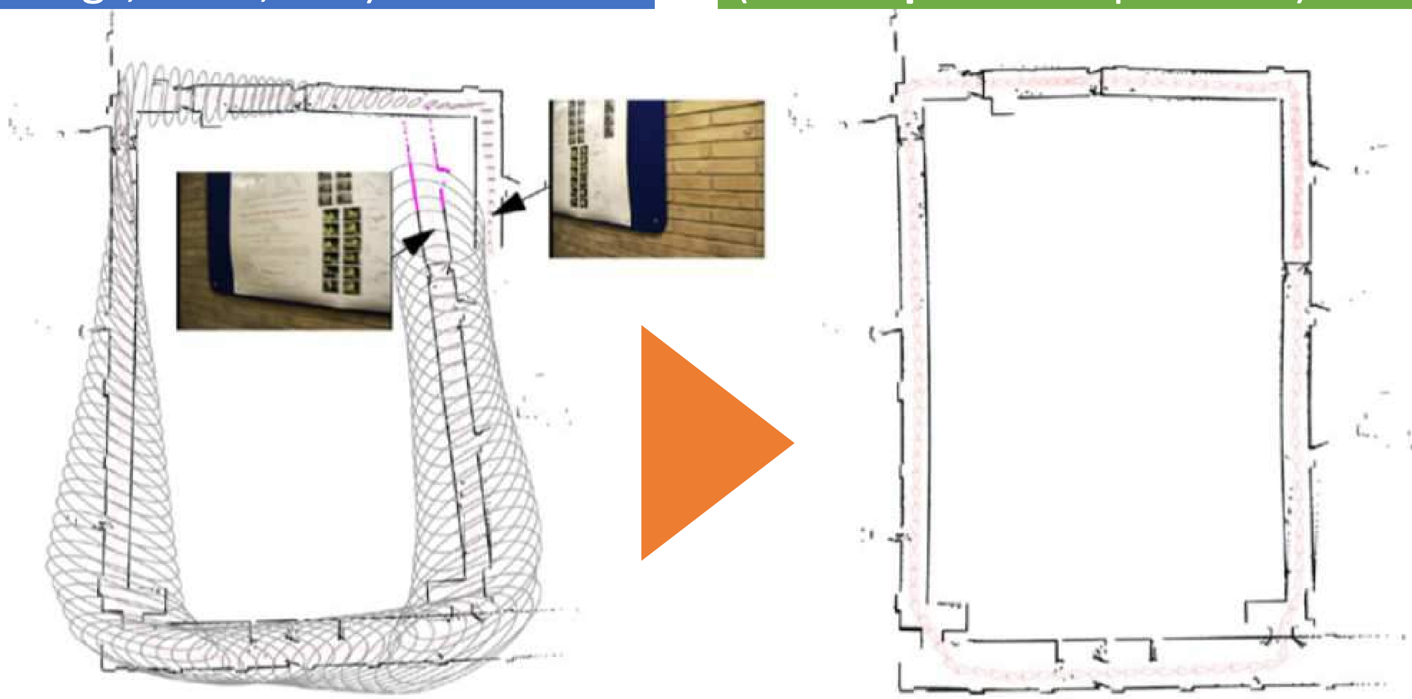
Ho and Newman 2006

4

But what if we don't have a map of the environment? Enter Simultaneous Localization and Mapping (SLAM)

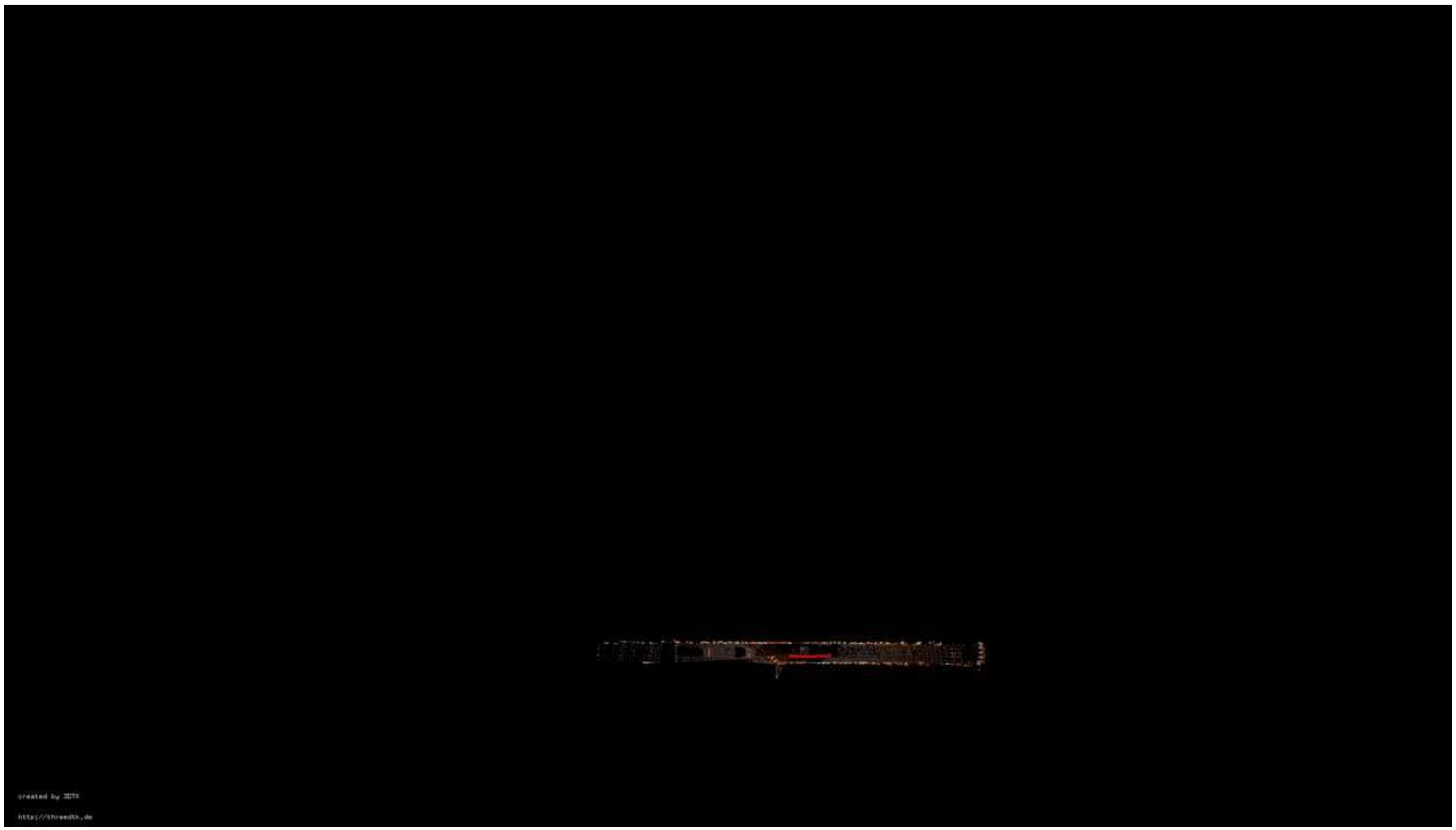
Essentially just additionally tracking the belief of **landmarks** in the environment (walls, buildings, trees, etc.)

The real hard part is figuring out when you have been somewhere before as measurements drift (the **loop closure** problem)

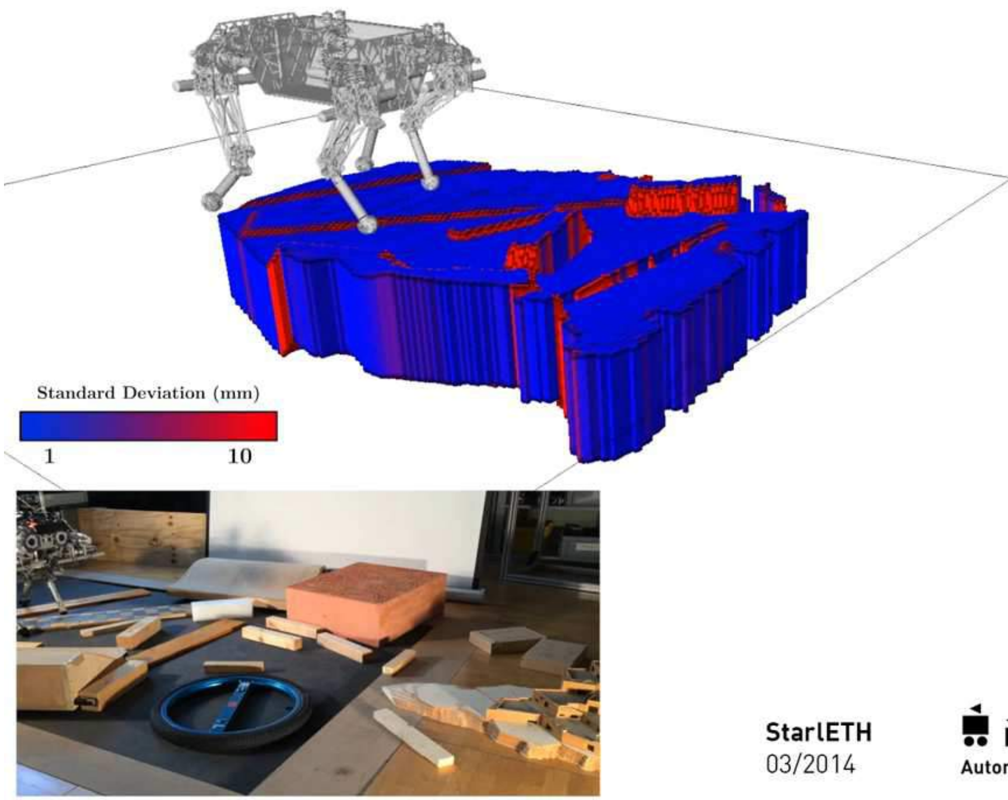


Ho and Newman 2006

4 SLAM with Loop Closure



4 Mapping can even be done in 3D!

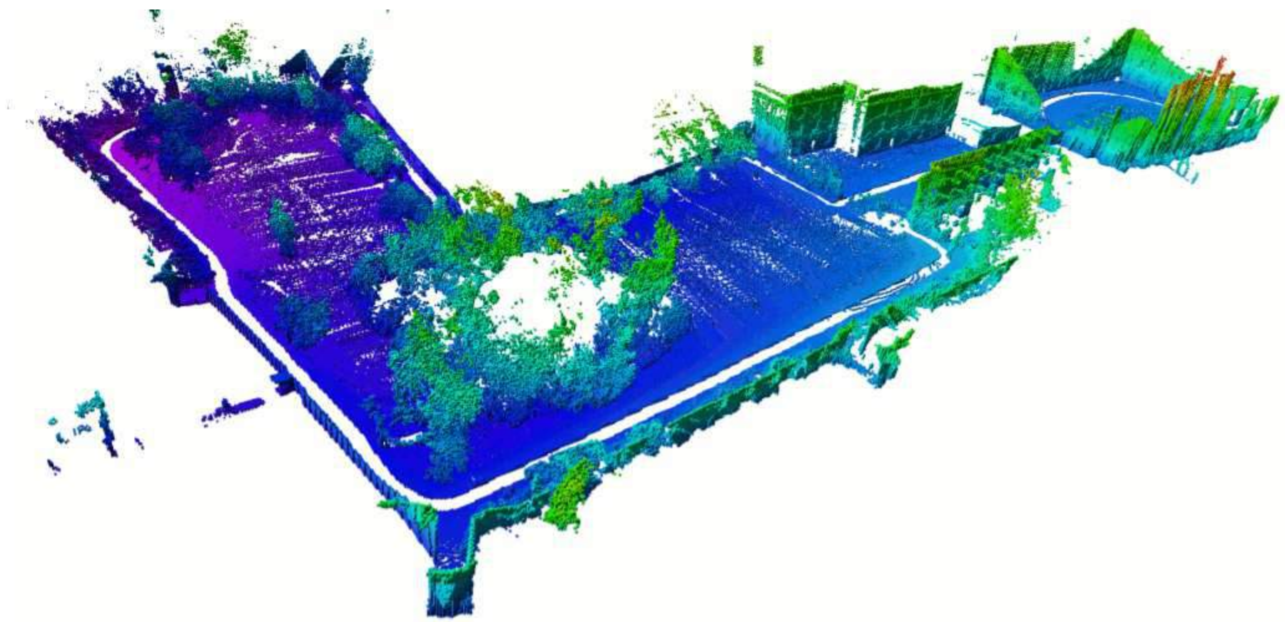


StarlETH
03/2014

Autonomous Systems Lab

ETH zürich

4 However building (and even storing) maps leads to a huge memory problem especially on small mobile systems

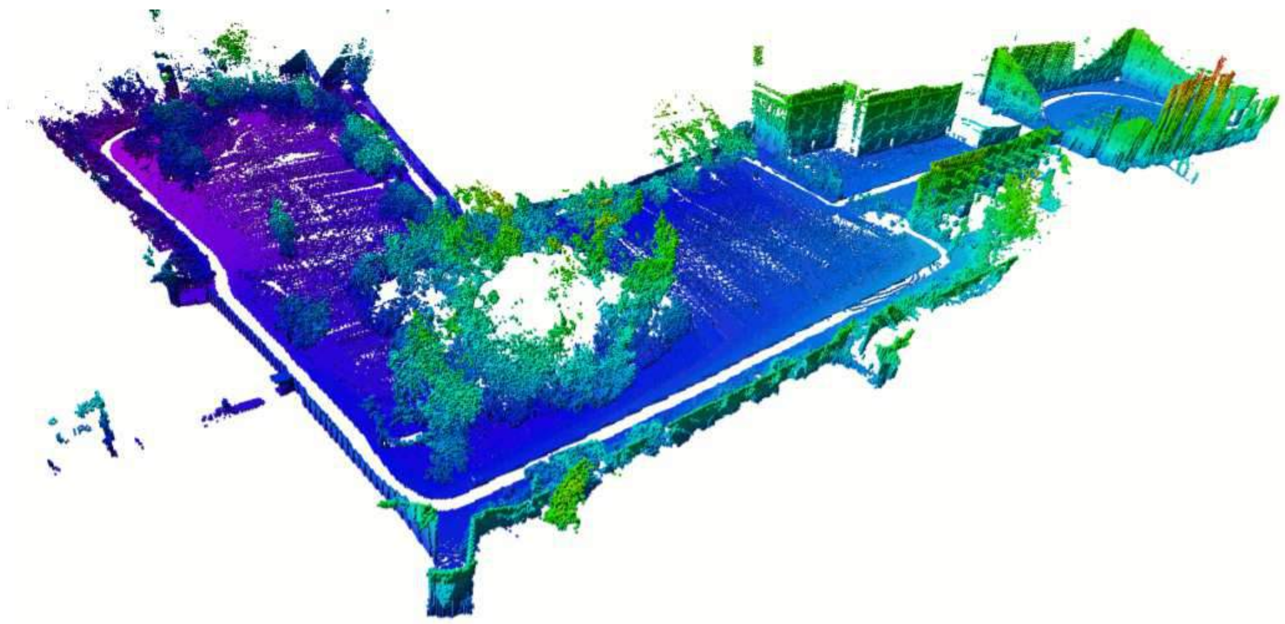


3D grid at 10cm resolution was 5058.76 MB (over 5 GB)

“Octomap” Hornung et. al. 2012

4

However building (and even storing) maps leads to a huge memory problem especially on small mobile systems



3D grid at 10cm resolution was 5058.76 MB (over 5 GB)

Oct-tree w/ Maximum Likelihood metric was able to compress that to 230.33 MB

"Octomap" Hornung et. al. 2012

4 However building (and even storing) maps leads to a huge memory problem especially on small mobile systems

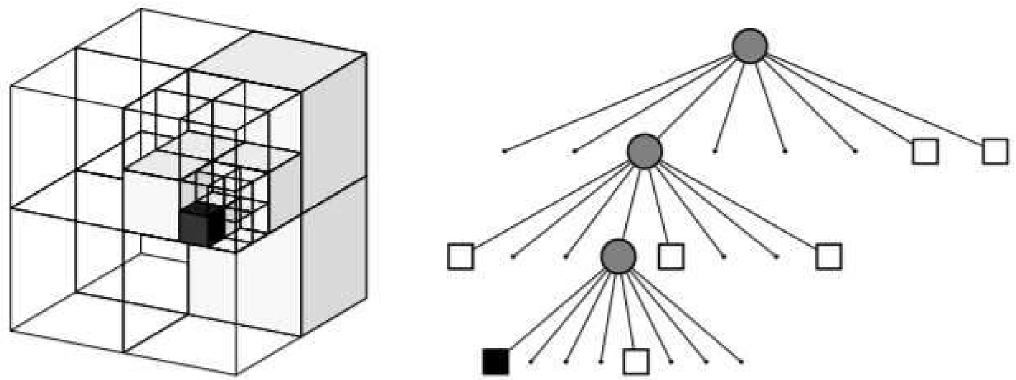


Fig. 2 Example of an octree storing free (shaded white) and occupied (black) cells. The volumetric model is shown on the left and the corresponding tree representation on the right.

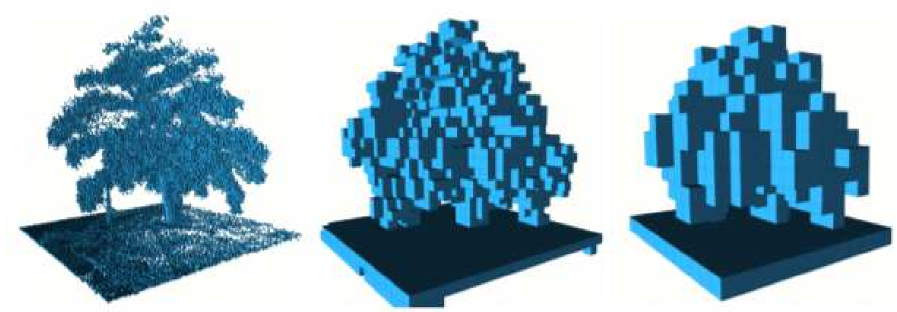


Fig. 3 By limiting the depth of a query, multiple resolutions of the same map can be obtained at any time. Occupied voxels are displayed in resolutions 0.08 m, 0.64 , and 1.28 m.

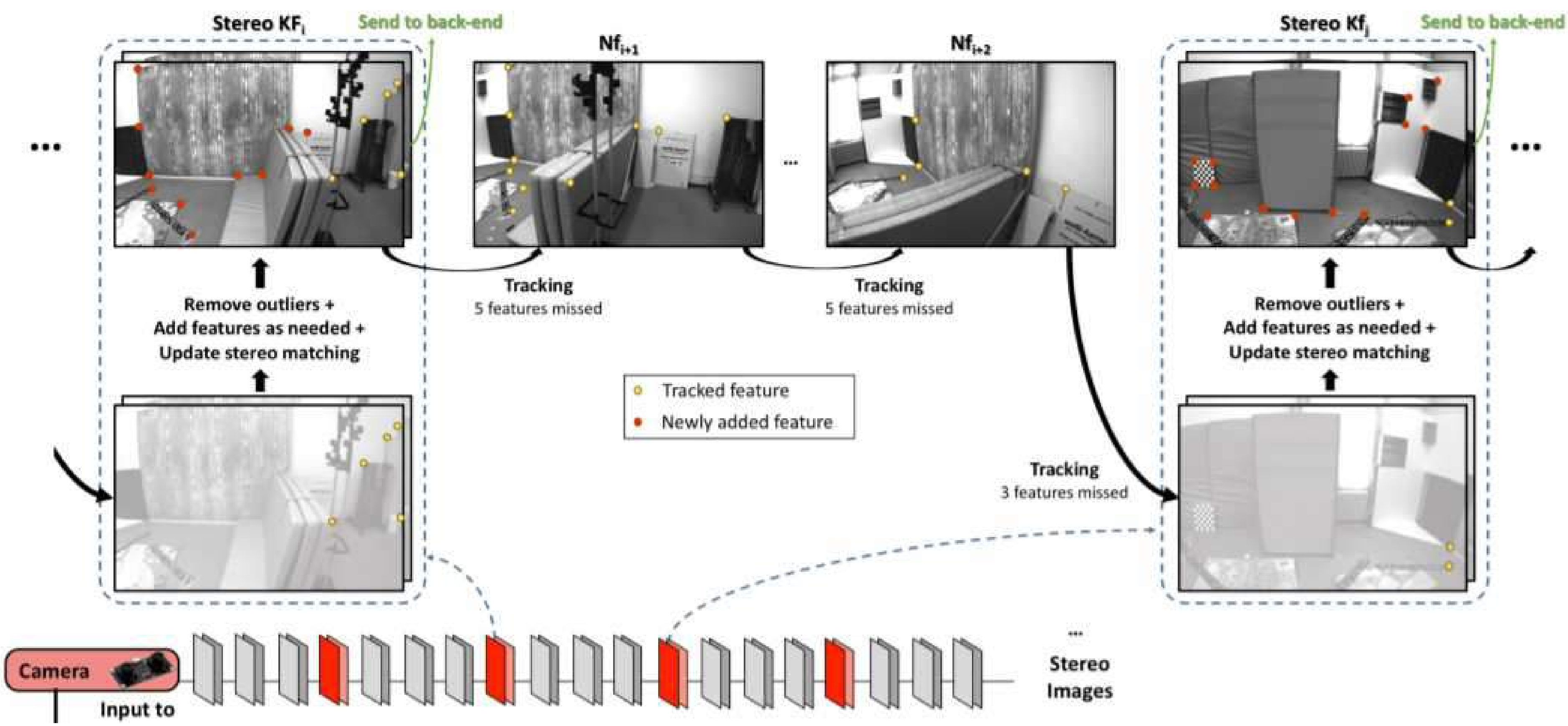
"Octomap" Hornung et. al. 2012

4 But how would we run localization online in a drone that is too small to carry fancy sensors?

Any ideas?

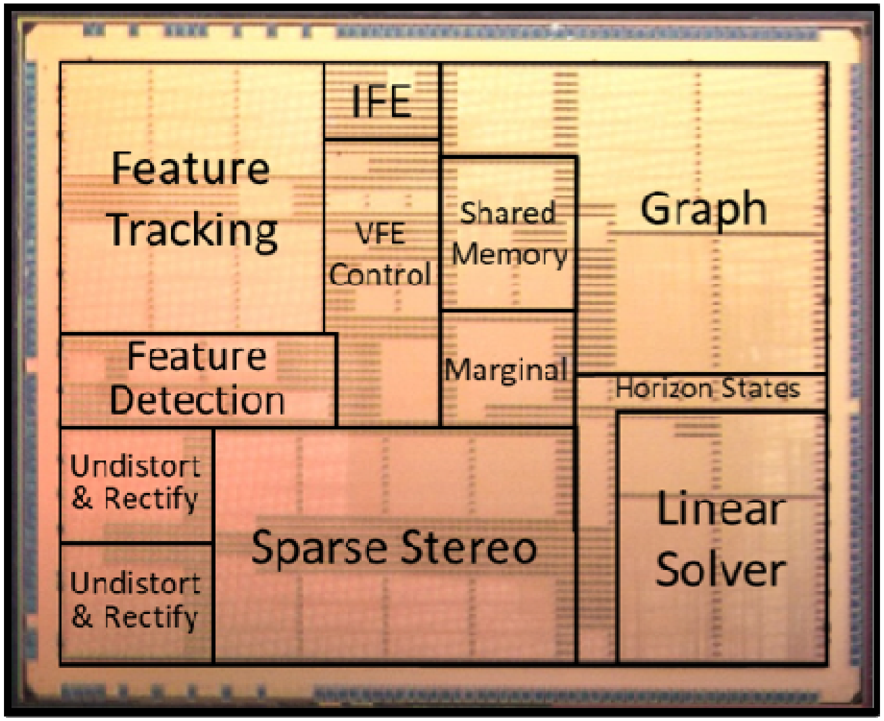
4

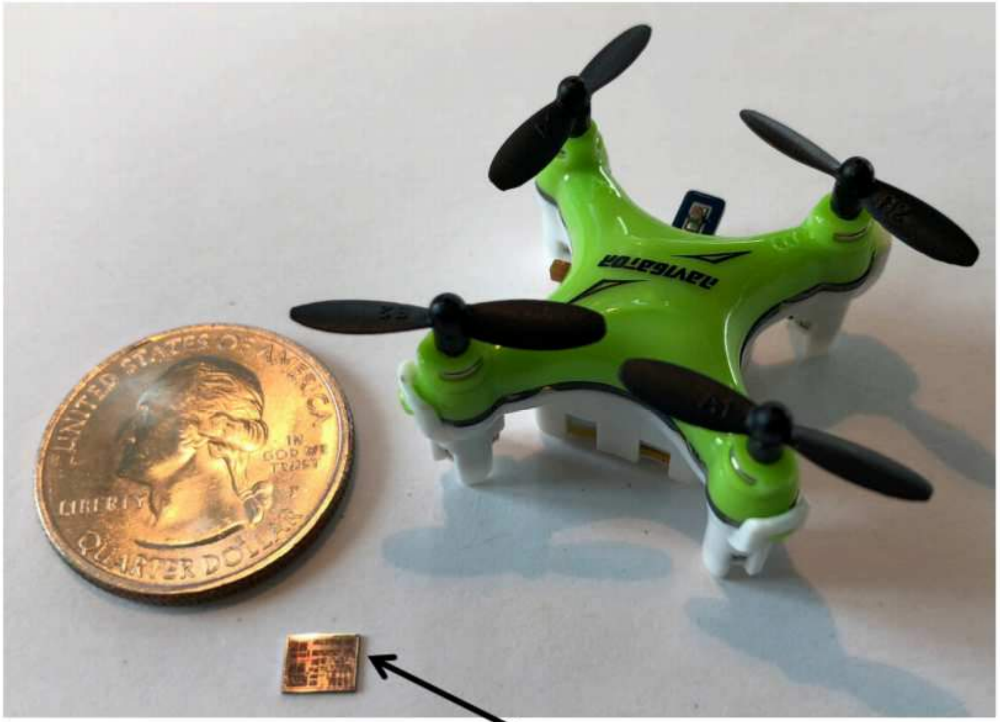
You can estimate the velocity of an object through matching interest points (Visual Odometry)…



4 ...and then build a custom chip to fit it onboard!

http://navion.mit.edu





Navion

Key Takeaways:

1. The Kalman/Particle Filter uses probability to solve the localization problem but modeling and/or approximations are needed for it to run efficiently online
2. Mapping quickly becomes a memory storage problem
3. Constrained form factors (aka tiny drones) will need novel accelerators to allow for autonomy

Your homework for next class

Pre-Reads for Intro to Robotics (Planning and Control)

Published

Edit

Computer Architeecture to Close the Loop in Real-time Optimization: https://ieeexplore.ieee.org/document/7402937

The Architectural Implications of Autonomous Driving: Constraints and Acceleration: https://web.eecs.umich.edu/~shihclin/papers/AutonomousCar-ASPLOS18.pdf

A Summary of Team MIT's Approach to the Virtual Robotics Challenge: https://agile.seas.harvard.edu/files/agile/files/vrc_entry.pdf

Your homework for next class

We have posted a tentative paper list to Canvas (along with PDFs and links)

Start to think about which papers you want as we will be allocating them in a week or two!

If you have an idea for a paper not on the list please run it by us and we may be willing to swap it in!

Pre-Reads for Intro to Robotics (Planning
Control)

Computer Architeecture to Close the Loop in Real-time Optimization: https://ieeexplore.ieee.org/document/7402937

The Architectural Implications of Autonomous Driving: Constraints and Acceleration: https://web.eecs.umich.edu/~shihclin/papers/AutonomousCar-ASPLOS18.pdf

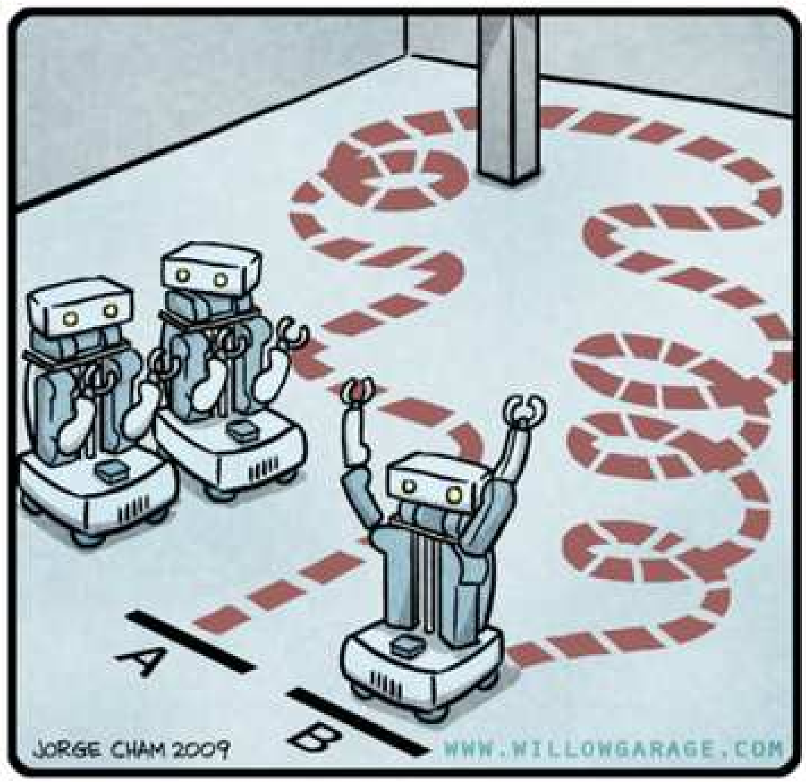
A Summary of Team MIT's Approach to the Virtual Robotics Challenge: https://agile.seas.harvard.edu/files/agile/files/vrc_entry.pdf

I’d love any Feedback!

http://bit.ly/CS249-Feedback-L1

CS 249r: Special Topics in Edge Computing

Intro to Autonomous Systems / Robotics Part 2

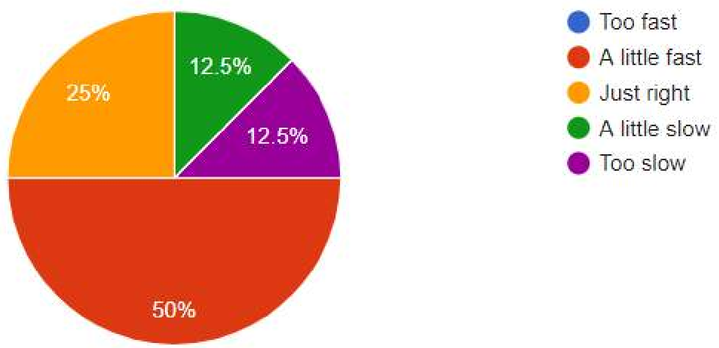


"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

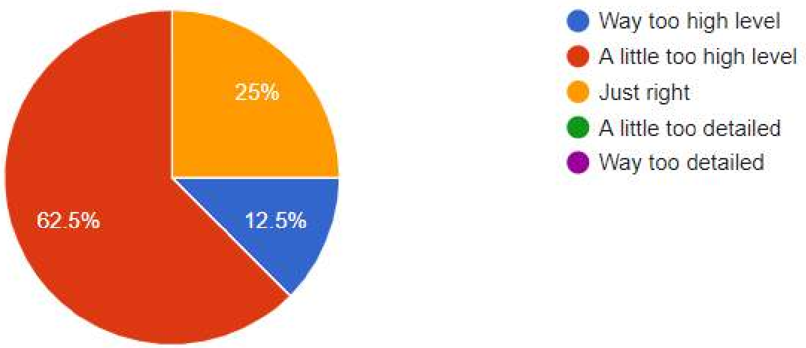
Brian Plancher
Fall 2019

Feedback from last class

How was the pace of class today?



How was the depth of the content covered today?

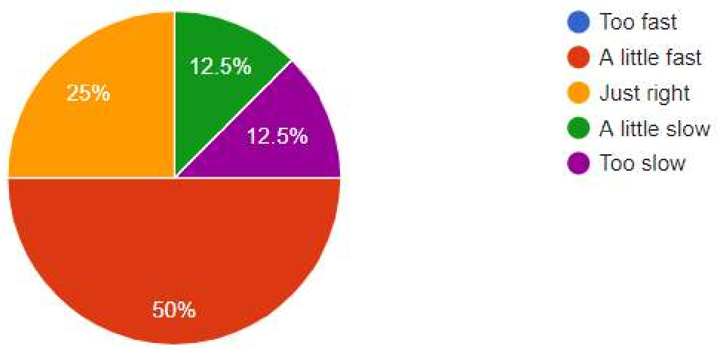


1. Pace is a tad fast
2. Get more technical/depth

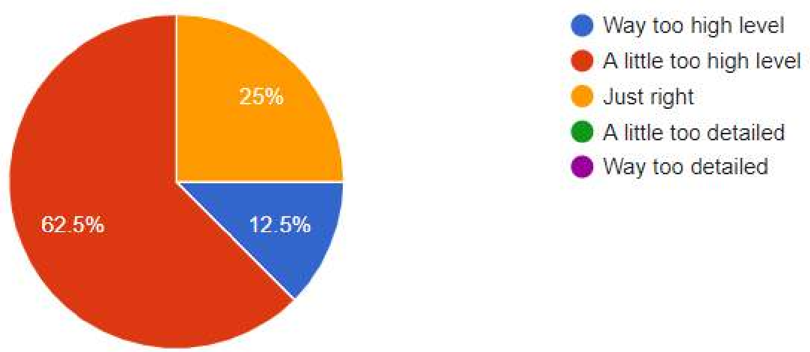
Feedback from last class

Also thanks for the open ended feedback!

How was the pace of class today?



How was the depth of the content covered today?



1. Pace is a tad fast
2. Get more technical/depth

Your homework for next class

We have posted a tentative paper list to Canvas (along with PDFs and links)

Start to think about which papers you want – I will send a link to vote for preferences in a week or so!

If you have an idea for a paper not on the list please run it by us and we may be willing to swap it in!

Pre-Reads for Intro to Domain Specific Architectures

Is dark silicon useful? Harnessing the four horsemen of the coming dar
apocalypse: https://ieeexplore.ieee.org/document/6241647

Turing Lecture: A New Golden Age for Computer Architecture:
https://californiaconsultants.org/wp-content/uploads/2018/04/CNSV-1806-Patterson.pdf (watch the lecture its great!)

Your homework for next class

Pre-Reads for Intro to Doma
Specific Architectures

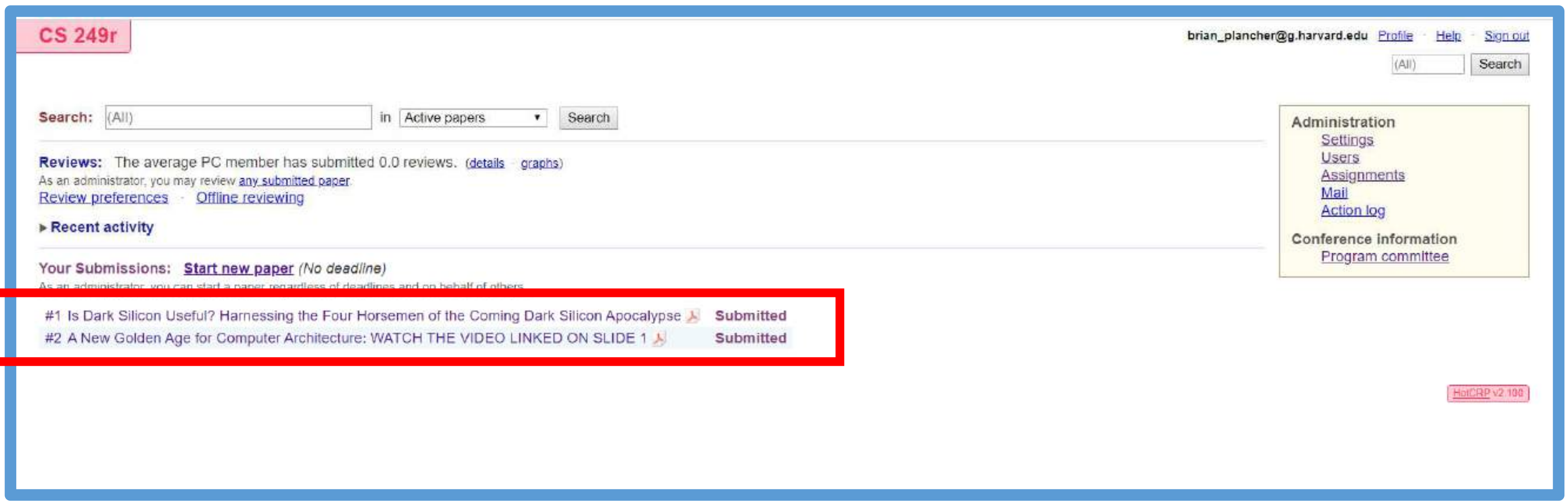
Is dark silicon useful? Harnessing the four horse
apocalypse: https://ieeexplore.ieee.org/docume

Turing Lecture: A New Golden Age for Computer Architecture:
https://californiaconsultants.org/wp-content/uploads/2018/04/CNSV-1806-Patterson.pdf (watch the lecture its great!)

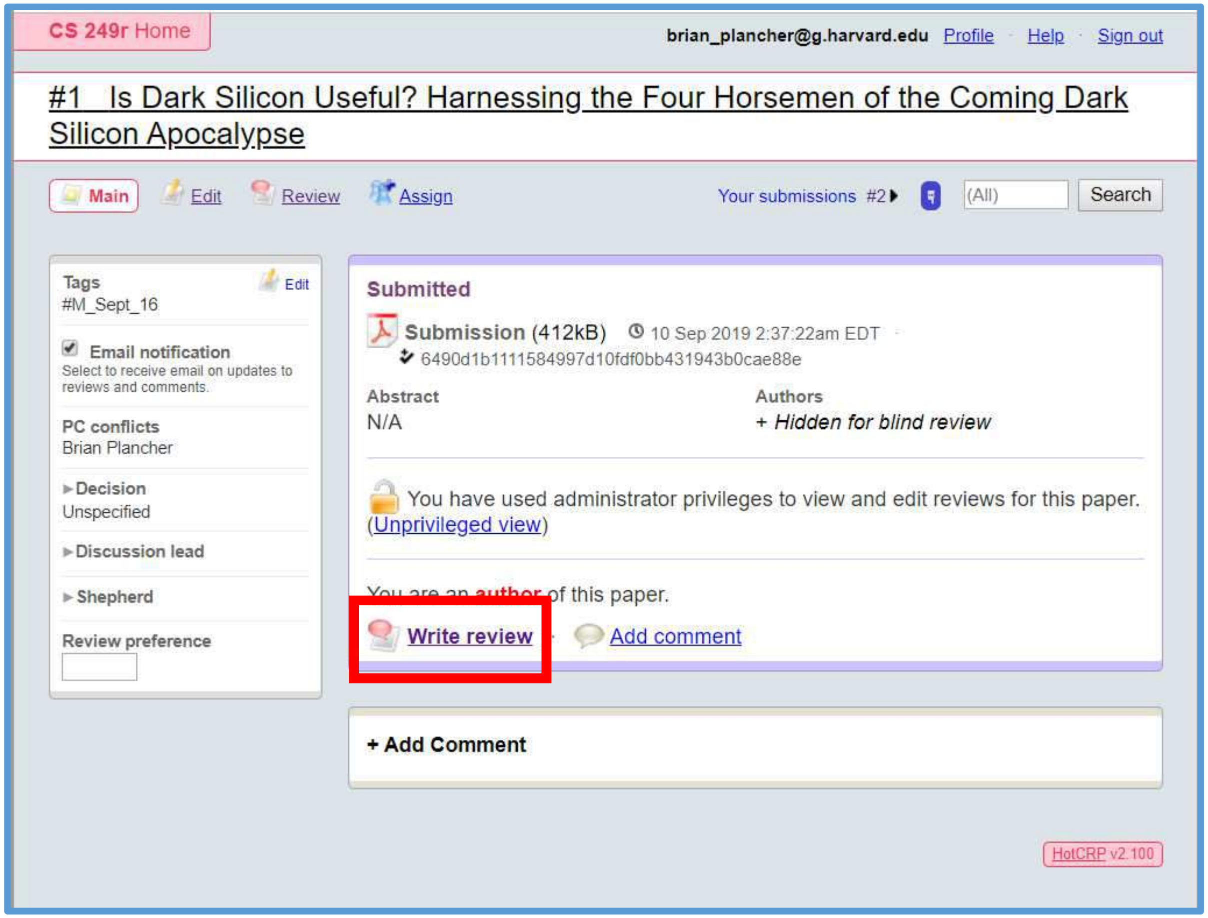
Were going to use **HOTCRP** (linked on Canvas and https://www.eecs.harvard.edu/cs249r/) for these for Monday – you will get an email from **Glenn Holloway** with a Password to access the site. (I am giving him the full roster as of today)

Your homework for next class

Click on a paper to access that paper's page

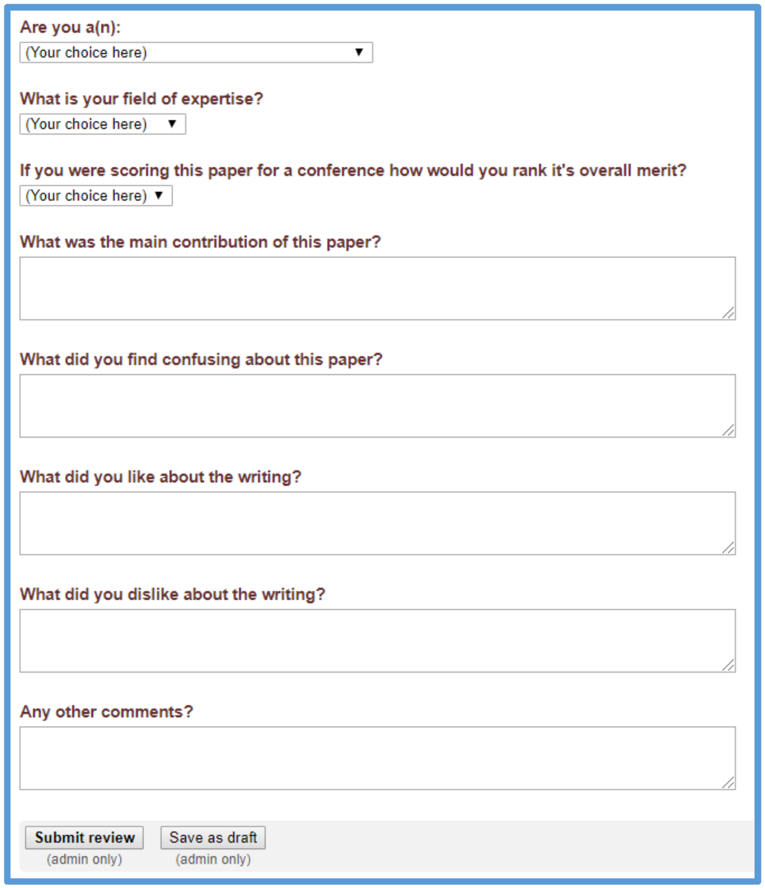


Your homework for next class



Then click “Write review” to open up the form to submit a “review”

Your homework for next class



Are you a(n):

(Your choice here)

What is your field of expertise?

(Your choice here)

If you were scoring this paper for a conference how would you rank it's overall merit?

(Your choice here)

What was the main contribution of this paper?

What did you find confusing about this paper?

What did you like about the writing?

What did you dislike about the writing?

Any other comments?

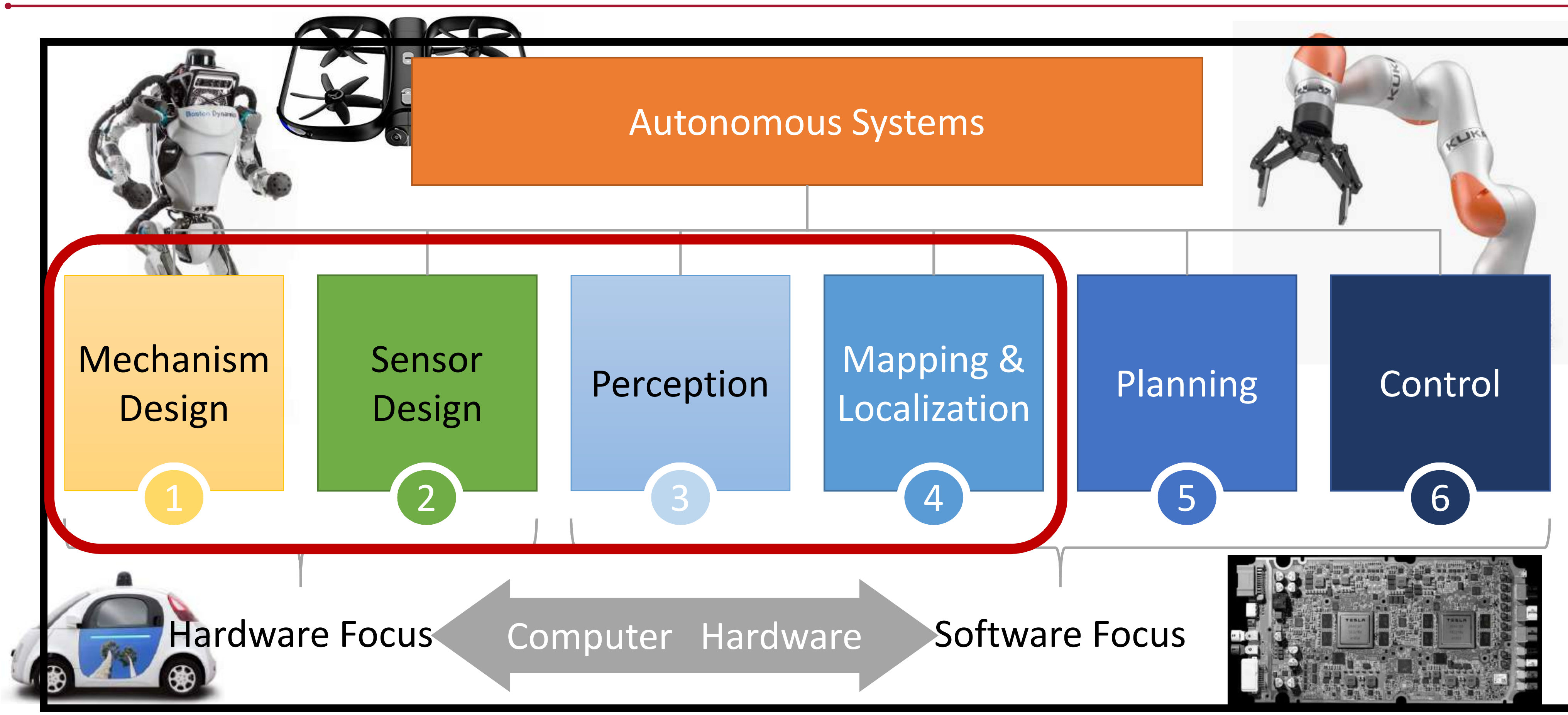
Submit review (admin only) Save as draft (admin only)

Then just fill it out and submit and you'll be good to go!

The goal for the next couple of lectures is to develop a high level understanding of:

1. What is an autonomous system
2. Key problems for autonomous systems
3. Some of the most important (classes of) algorithms in robotics
4. The model based vs. model free tradeoff
5. The online vs offline tradeoff
6. The no free lunch theorem and the need for approximations
7. How computer systems / architecture design has and can play a role in improving autonomous systems

Autonomous Systems / Robotics is a BIG space

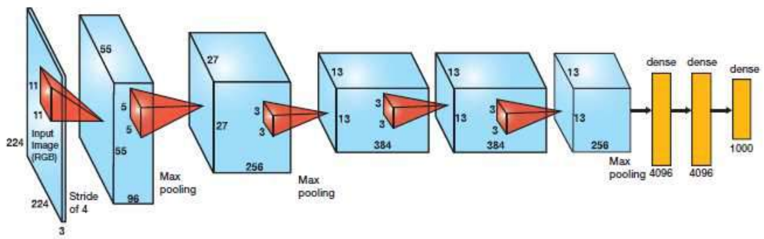


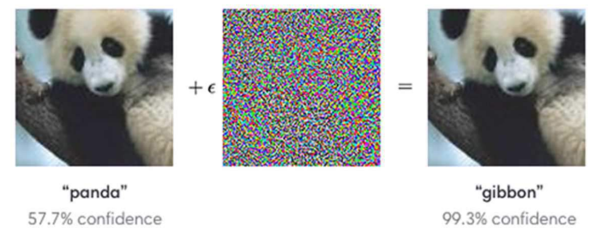
1 2 Key Takeaways:

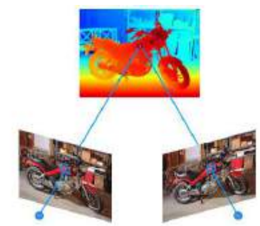



1. When designing algorithms for robots you need to understand the physical capabilities of the robot and you (potentially) need to understand how to model its physical behaviors
2. Different kinds of systems will have different power, weight, and performance budgets for computer hardware

3 Key Takeaways:

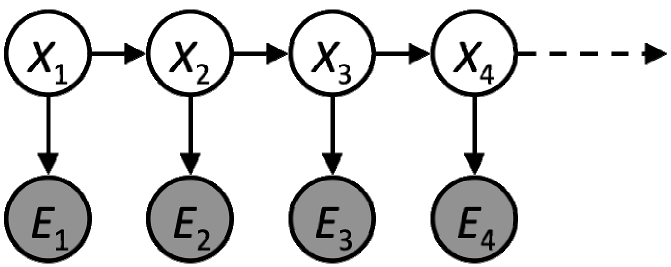


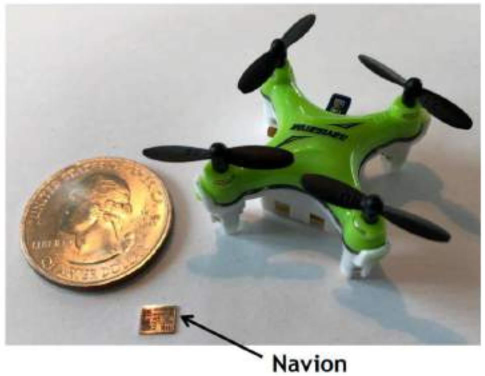


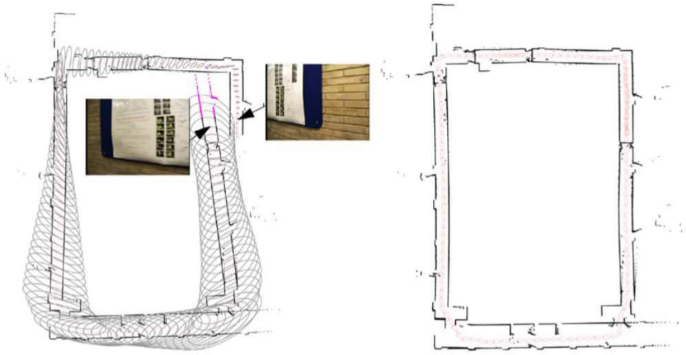


1. As of today it seems like CNNs that automate the design and summary of salient features via convolution are the way to go
   - But/and will need specialized NN running on specialized accelerator chips to get them small enough to fit on small power constrained autonomous systems (e.g., small drones)
   - And we will need to find ways to secure them against attacks!
2. Also, other more targeted problems such as Stereo Depth seem to need accelerators!

④ Key Takeaways:

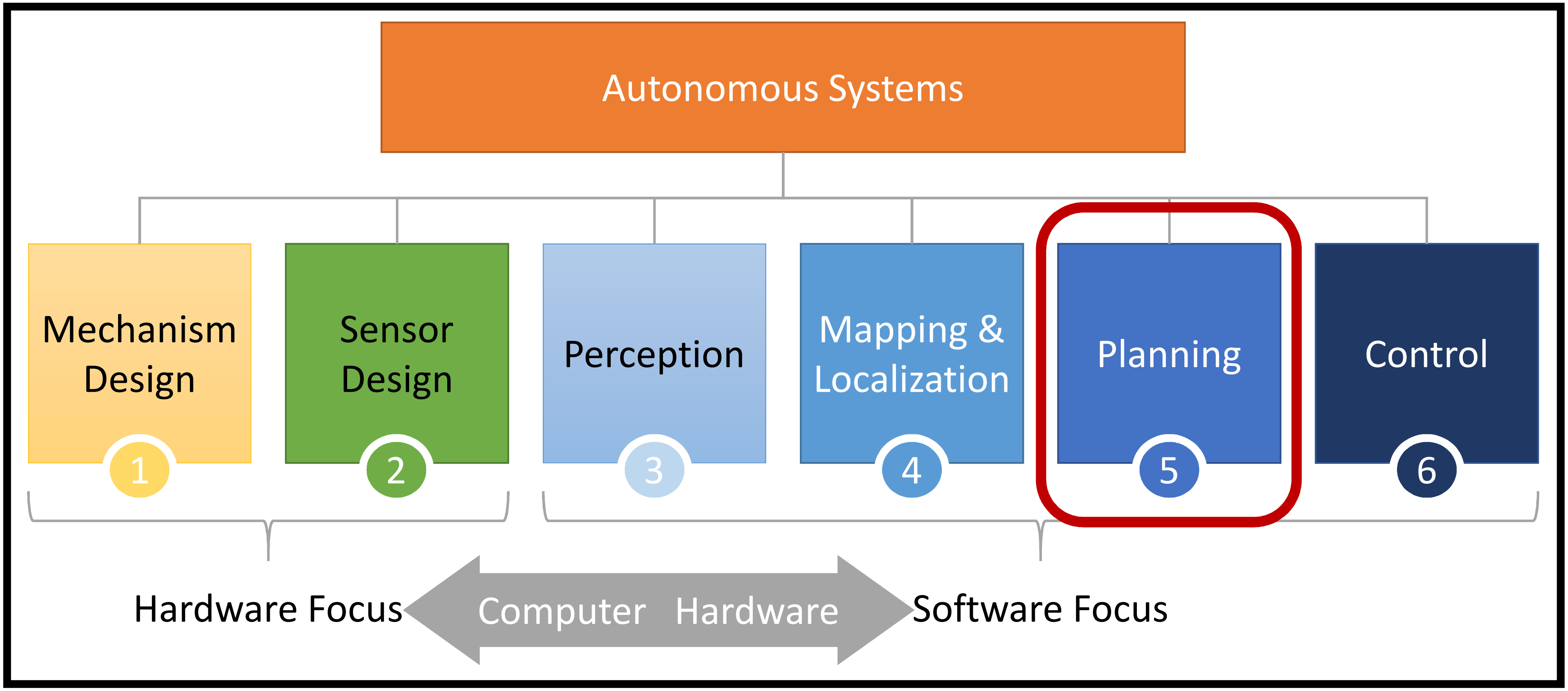






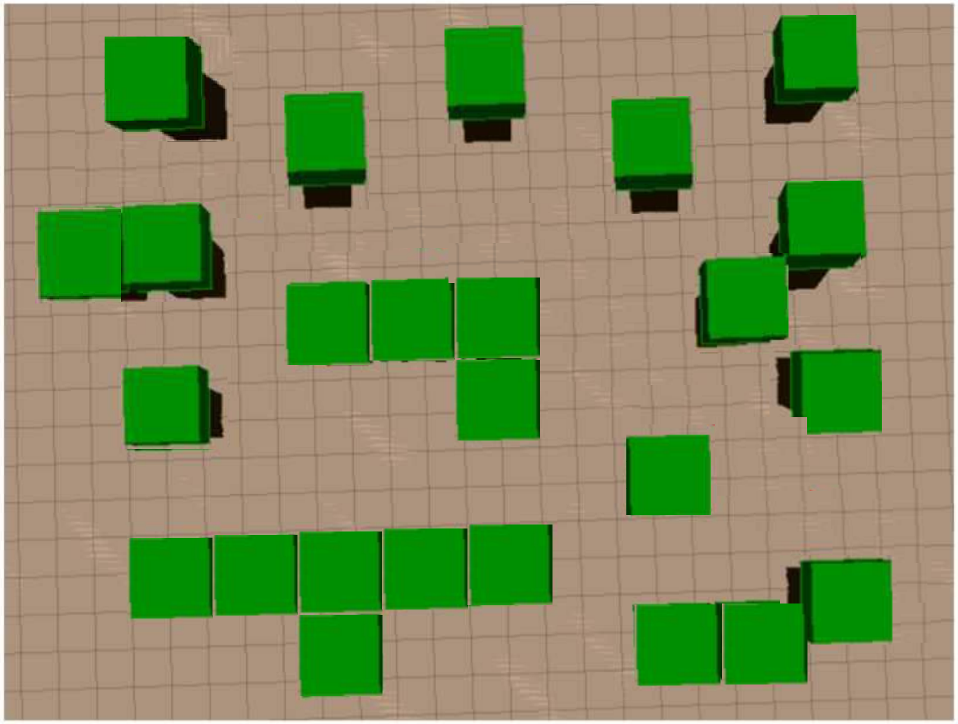
1. The Kalman/Particle Filter uses probability to solve the localization problem but modeling and/or approximations are needed for it to run efficiently online
2. Mapping quickly becomes a memory storage problem
3. Constrained form factors (aka tiny drones) will need novel accelerators to allow for autonomy

Autonomous Systems / Robotics is a BIG space



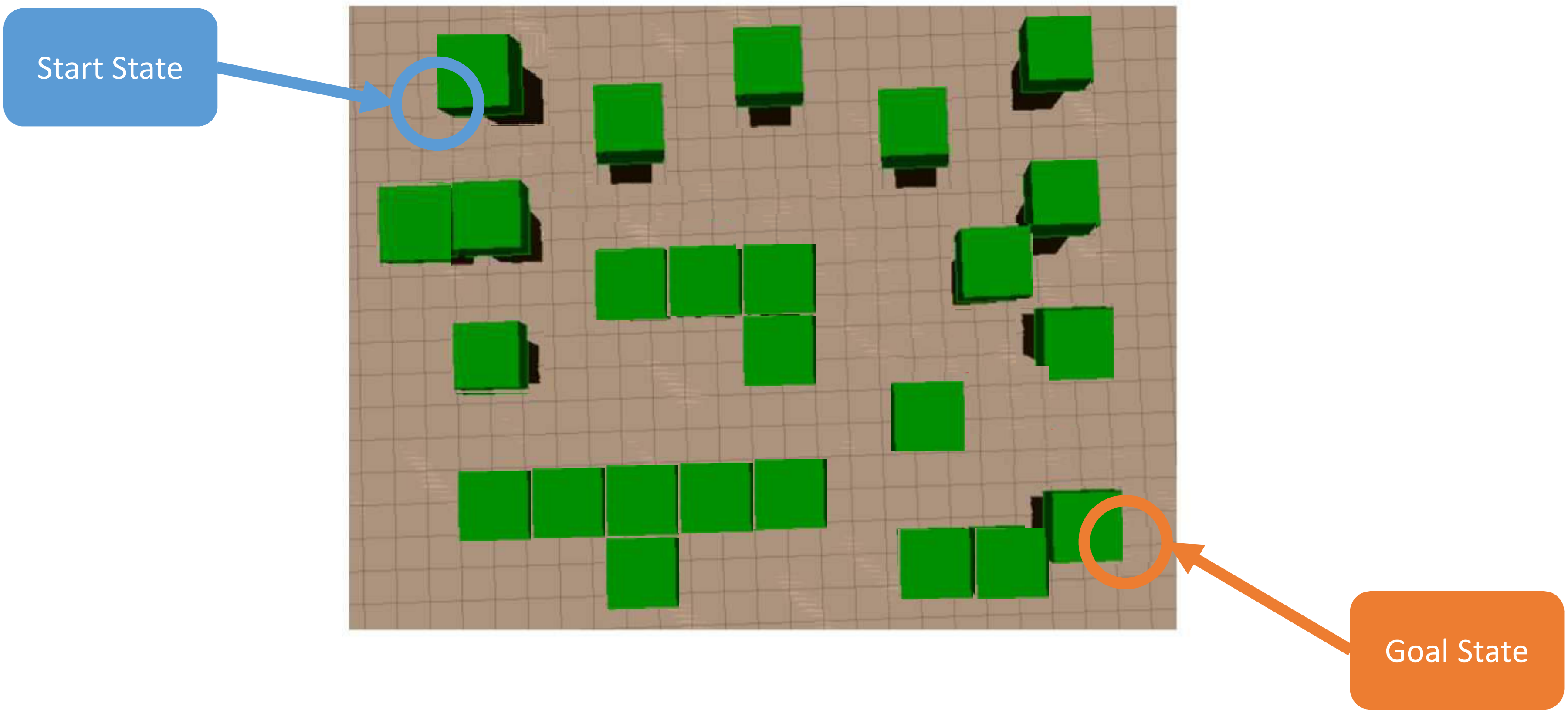
5

Planning is the process of computing an action plan for a robot based on the previously computed map

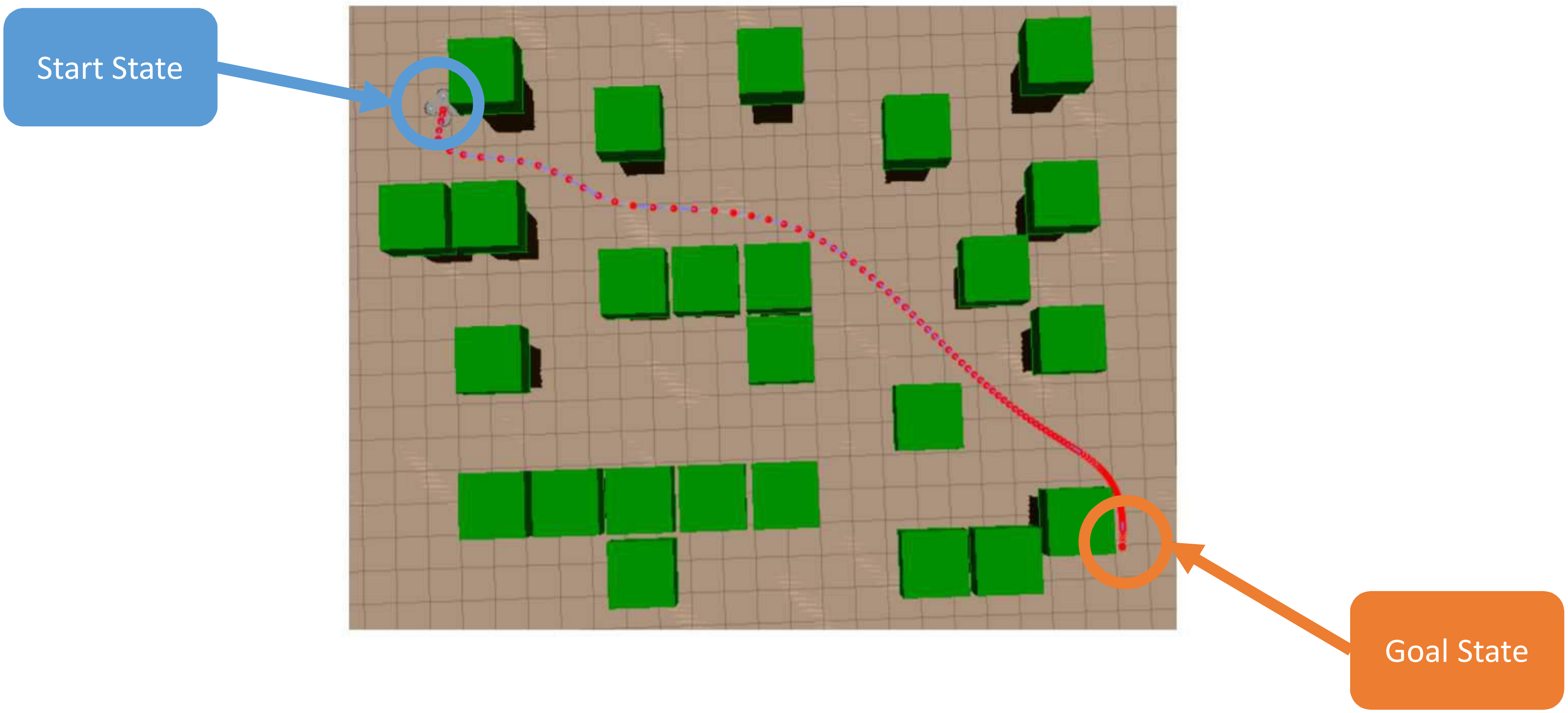


5

Planning is the process of computing an action plan for a robot based on the previously computed map



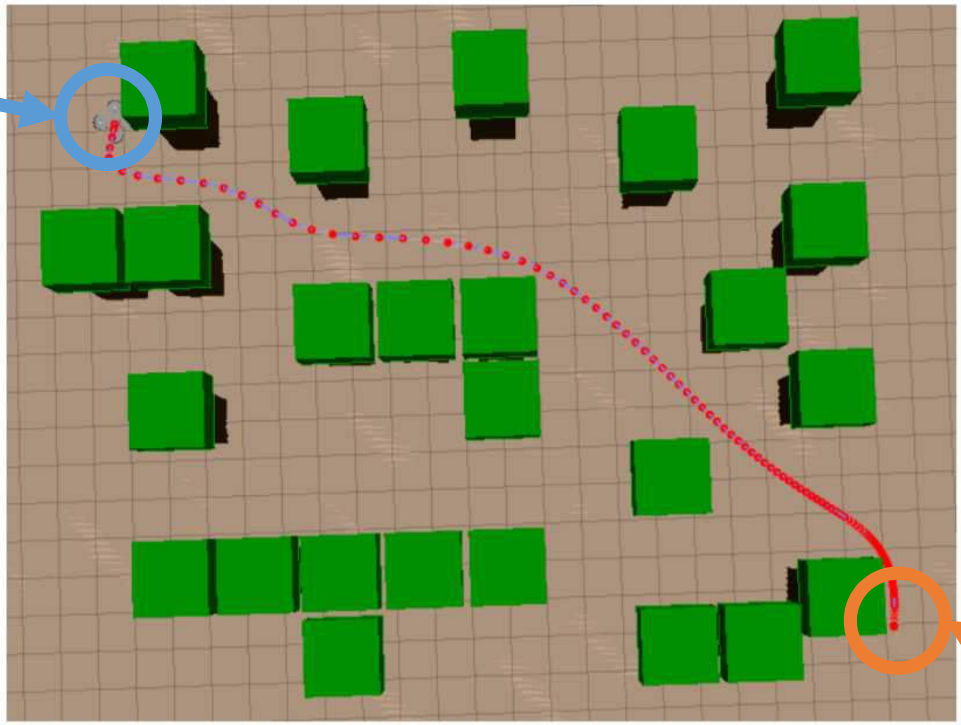
5 Planning is the process of computing an action plan for a robot based on the previously computed map



5

Planning is the process of computing an action plan for a robot based on the previously computed map

Start State

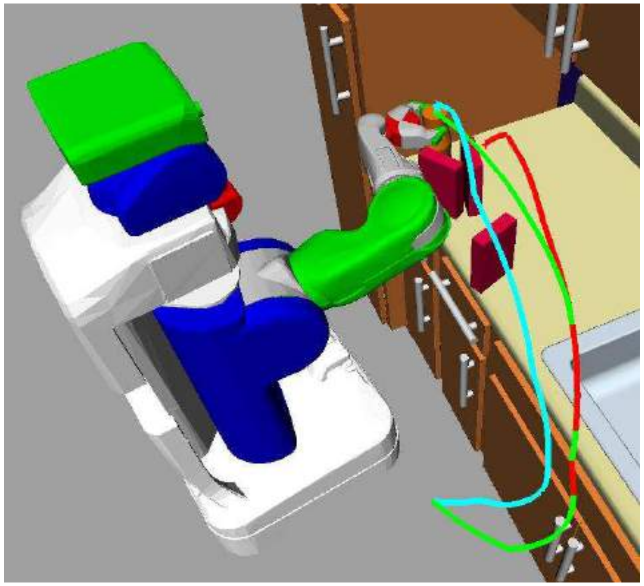


Goal State

Before we can think about how to compute this we need to figure out in what state space are we planning?

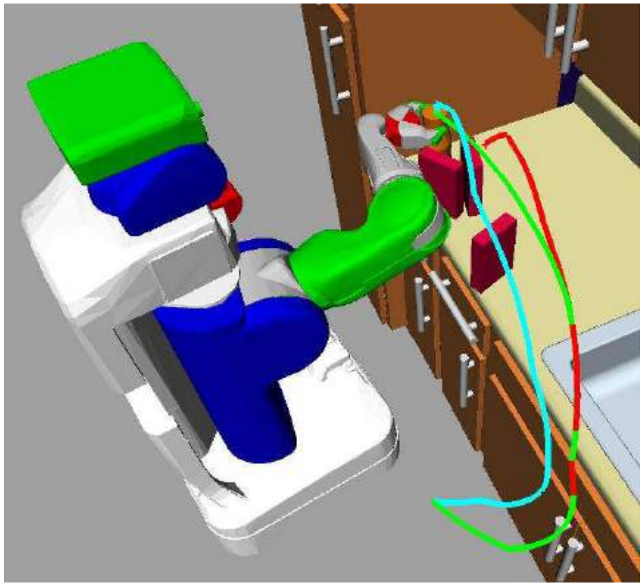
5 Spaces and Transformations (aka where are we planning?)

- Task space: the 6D workspace of the robot
  - E.g., the pose (x,y,z,roll,pitch,yaw) of the robot's hand or an object



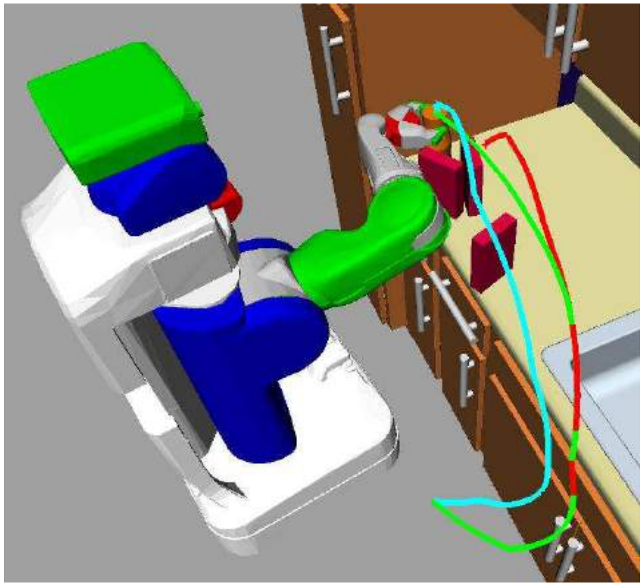
5 Spaces and Transformations (aka where are we planning?)

- Task space: the 6D workspace of the robot
  - E.g., the pose (x,y,z,roll,pitch,yaw) of the robot's hand or an object
- Configuration space: the *n*-dimensional space of joint angles + robot world position
  - Vector $q \in \mathbb{R}^n$



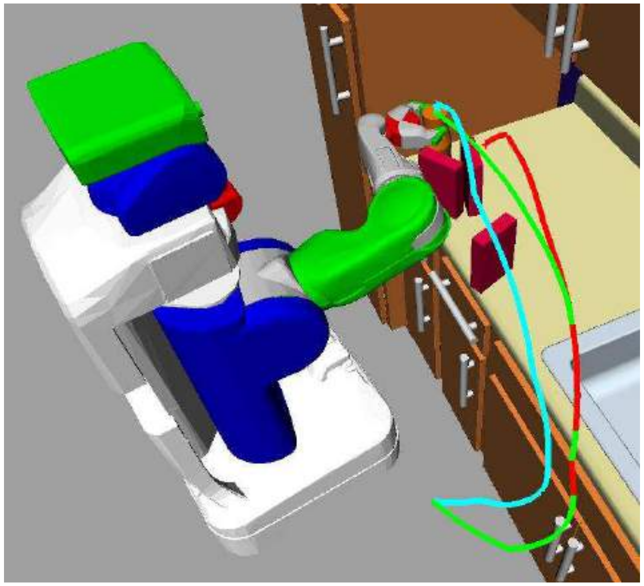
5 Spaces and Transformations (aka where are we planning?)

- Task space: the 6D workspace of the robot
  - E.g., the pose (x,y,z,roll,pitch,yaw) of the robot's hand or an object
- Configuration space: the *n*-dimensional space of joint angles + robot world position
  - Vector $q \in \mathbb{R}^n$
- Forward kinematics: maps *q* to outputs in task space (e.g. hand position)
- Inverse kinematics: maps task space poses to configuration space



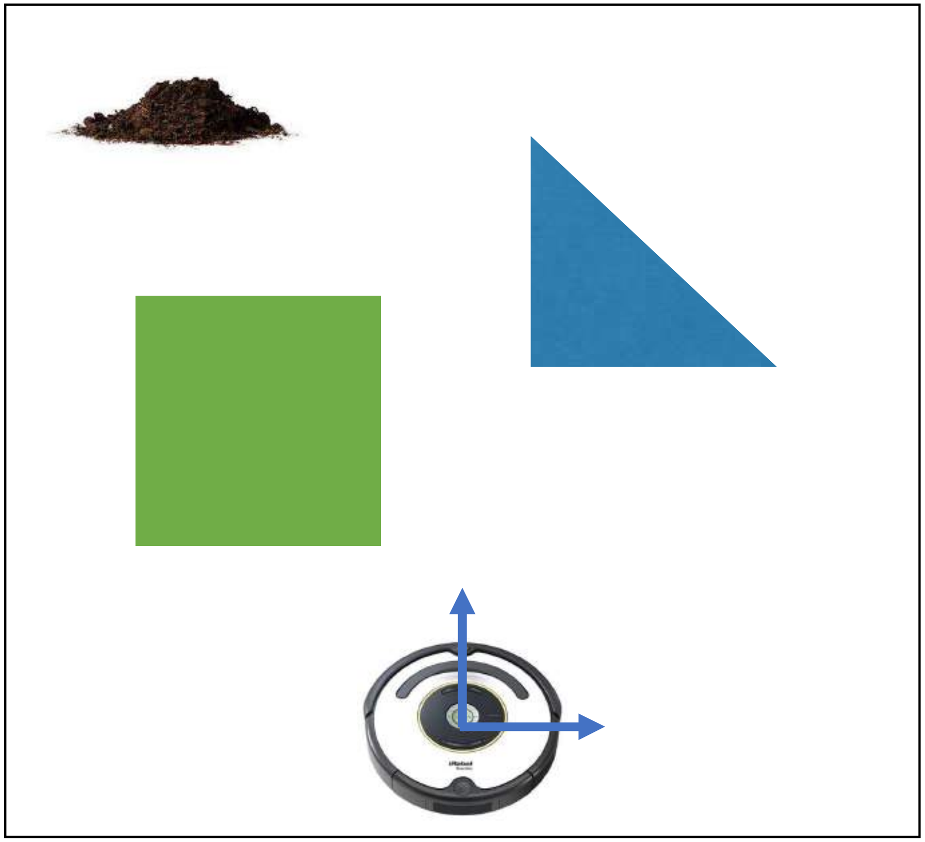
5 Spaces and Transformations (aka where are we planning?)

- Task space: the 6D workspace of the robot
  - E.g., the pose (x,y,z,roll,pitch,yaw) of the robot's hand or an object
- Configuration space: the *n*-dimensional space of joint angles + robot world position
  - Vector $q \in \mathbb{R}^n$
- Forward kinematics: maps *q* to outputs in task space (e.g. hand position)
- Inverse kinematics: maps task space poses to configuration space



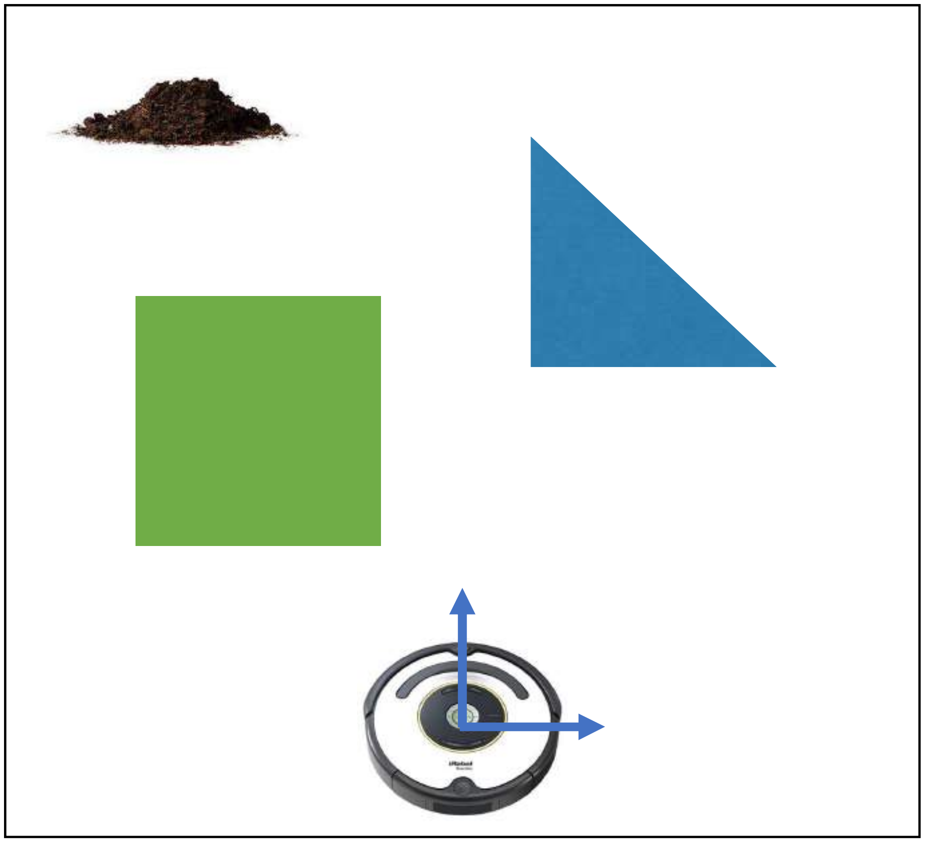
Q: Are forward and inverse kinematics 1 to 1 operations?

5 Configuration Space



Q1: What is the configuration space state for this omnidirectional robot?

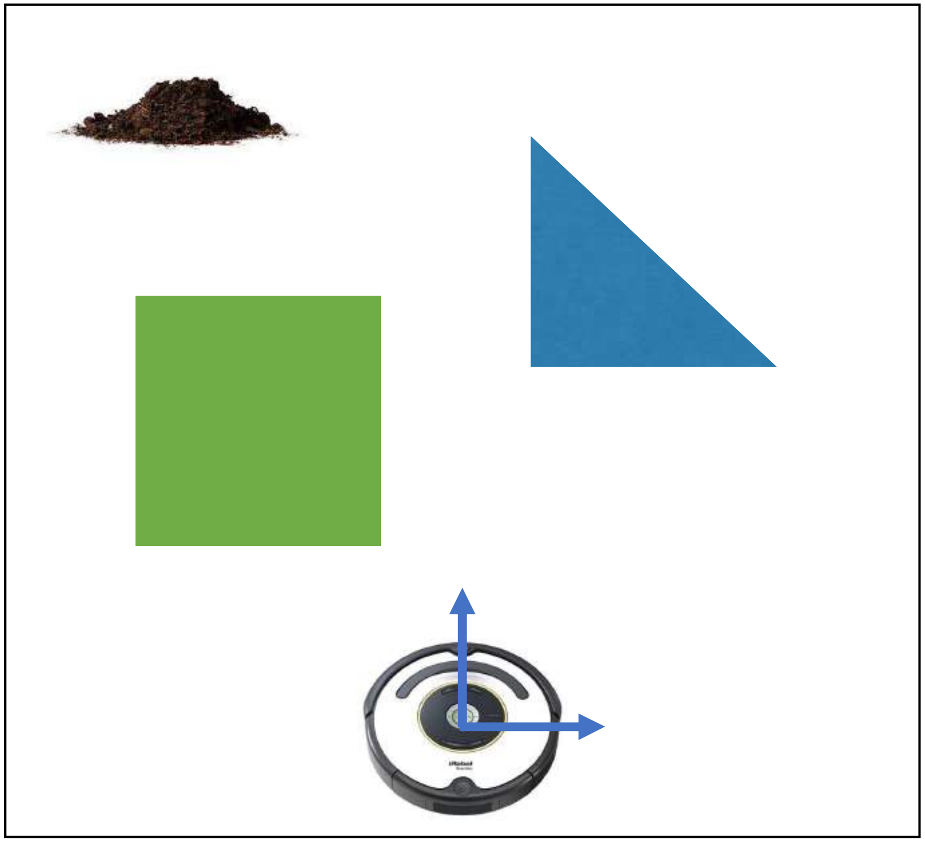
5 Configuration Space



Q1: What is the configuration space state for this omnidirectional robot?

A1: (x,y) position of the center of the robot

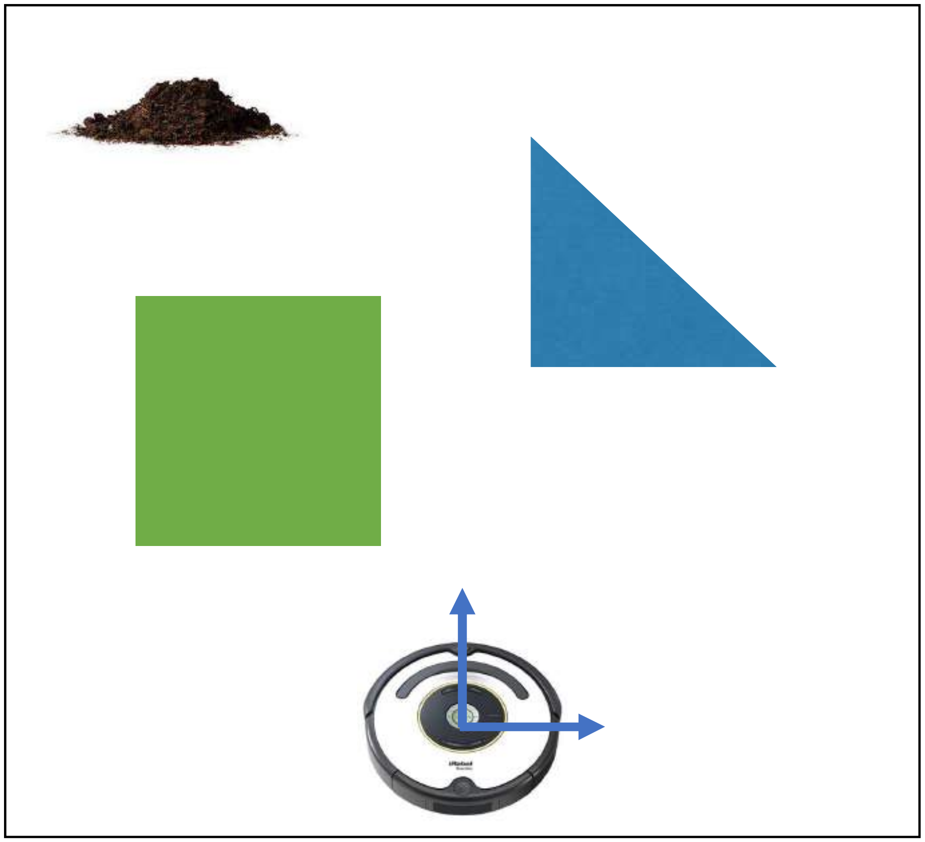
5 Configuration Space

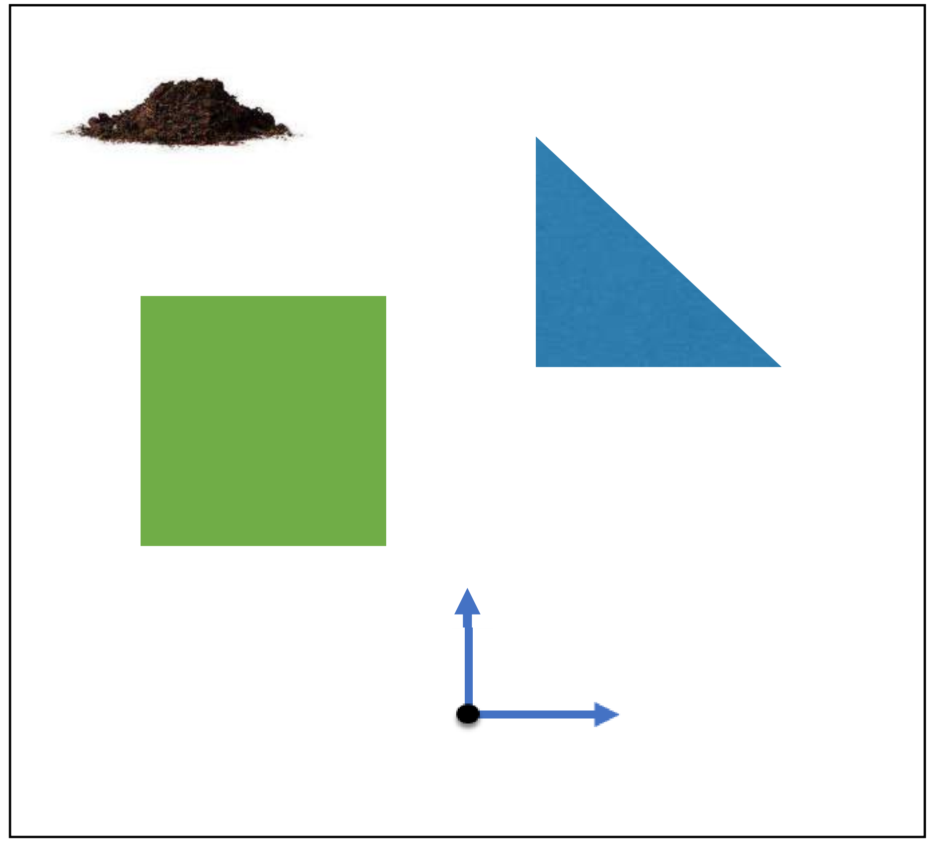


Q2: How can we map this robot’s world into configuration space?

5 Configuration Space

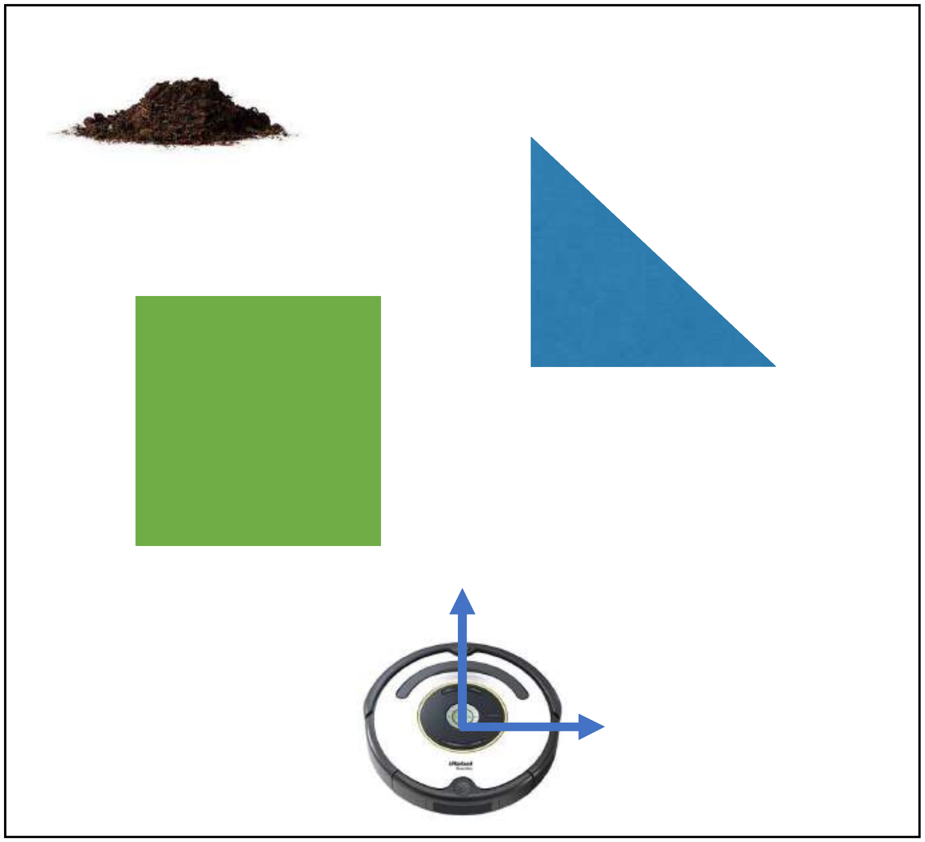
Well we want the robot to become a single (x,y) point

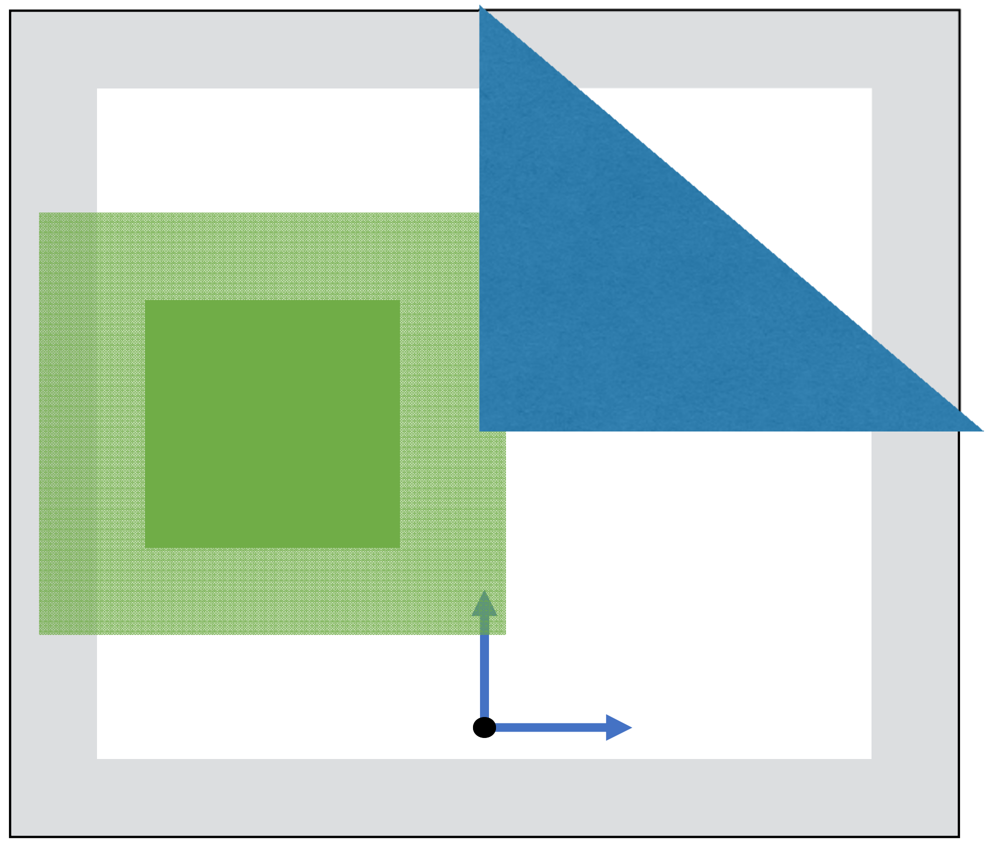




5 Configuration Space

So we need to inflate the obstacles accordinly

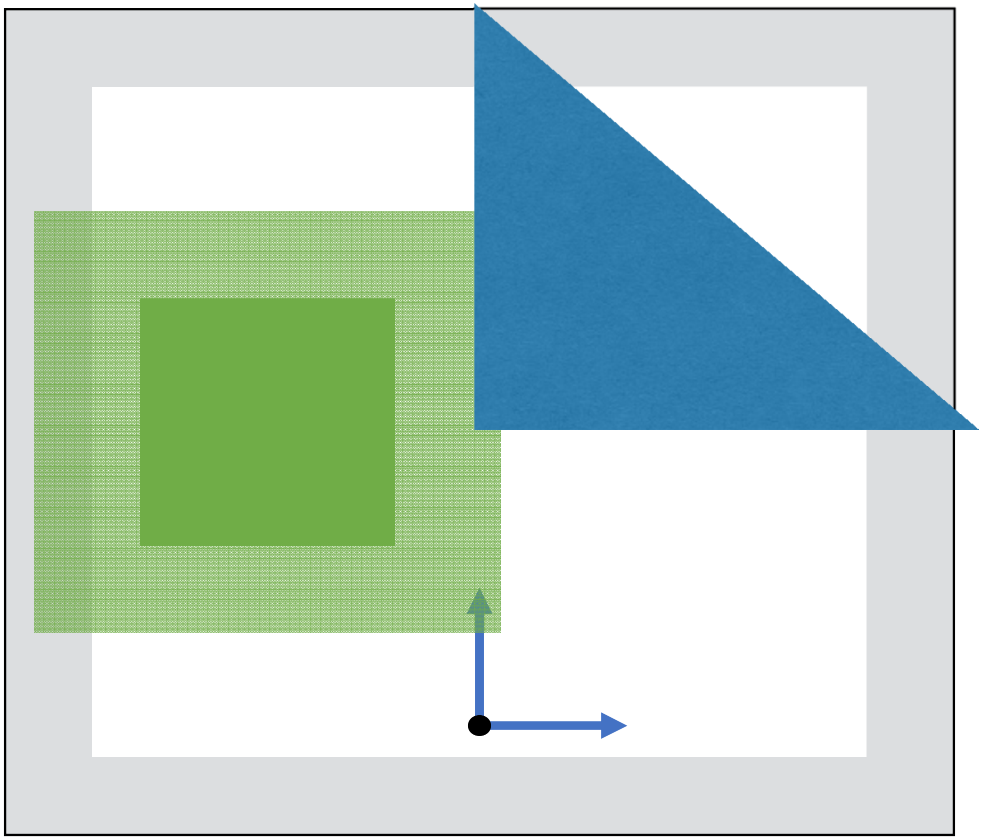




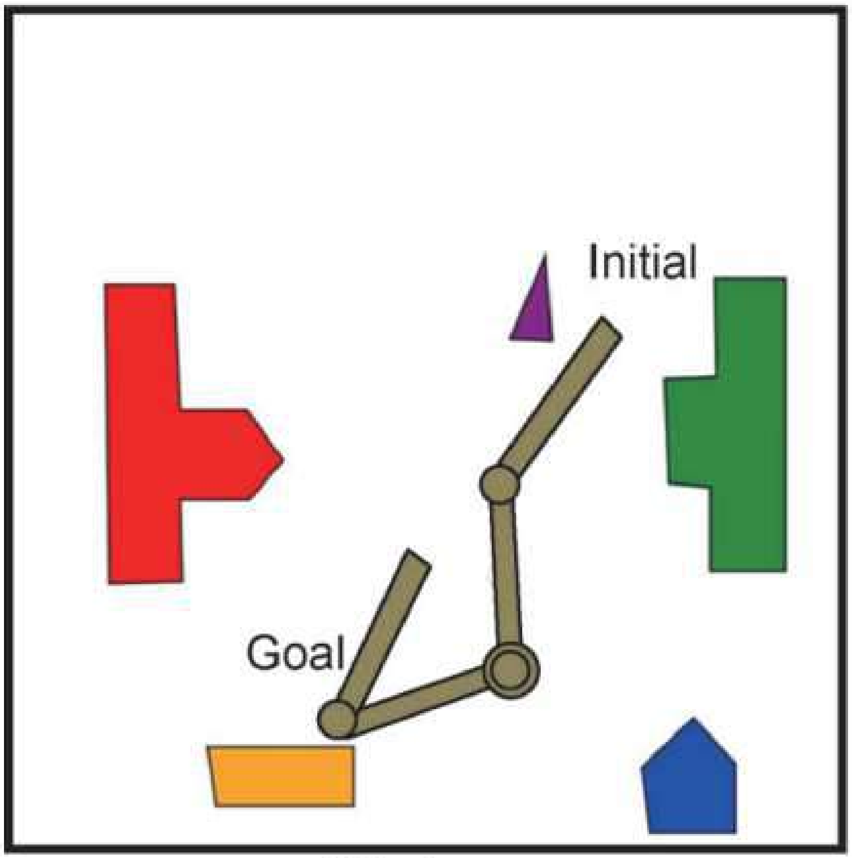
5

Configuration Space

- **Insight:** mapping task space obstacles and goals into configuration space allows us to **plan a path for a single point** instead of worrying about a full robot



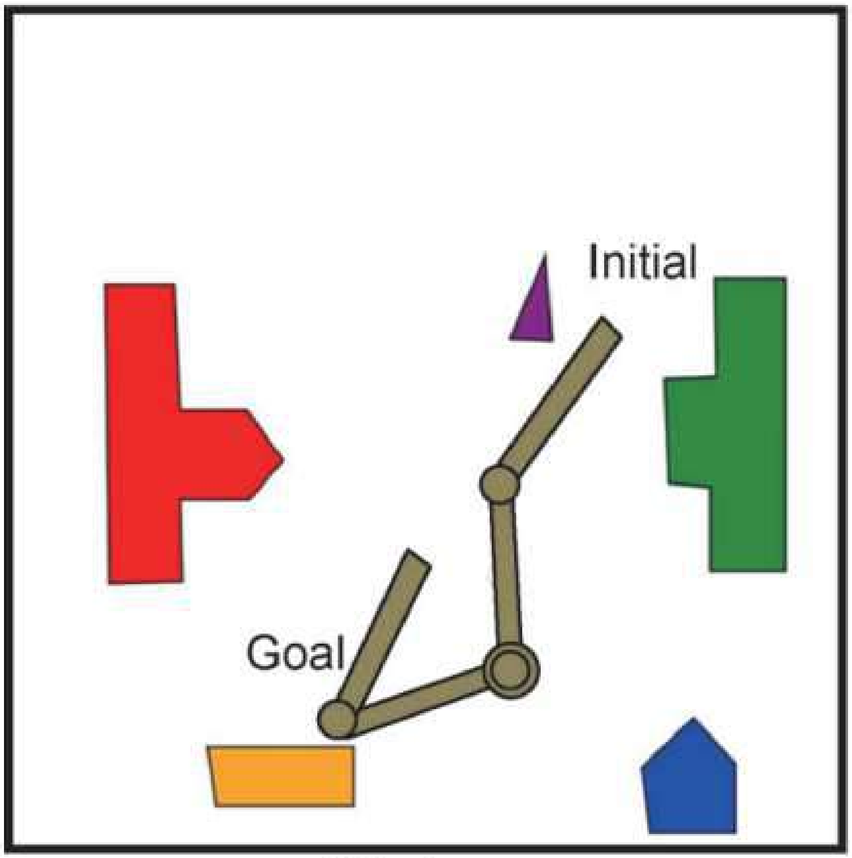
5 Configuration Space



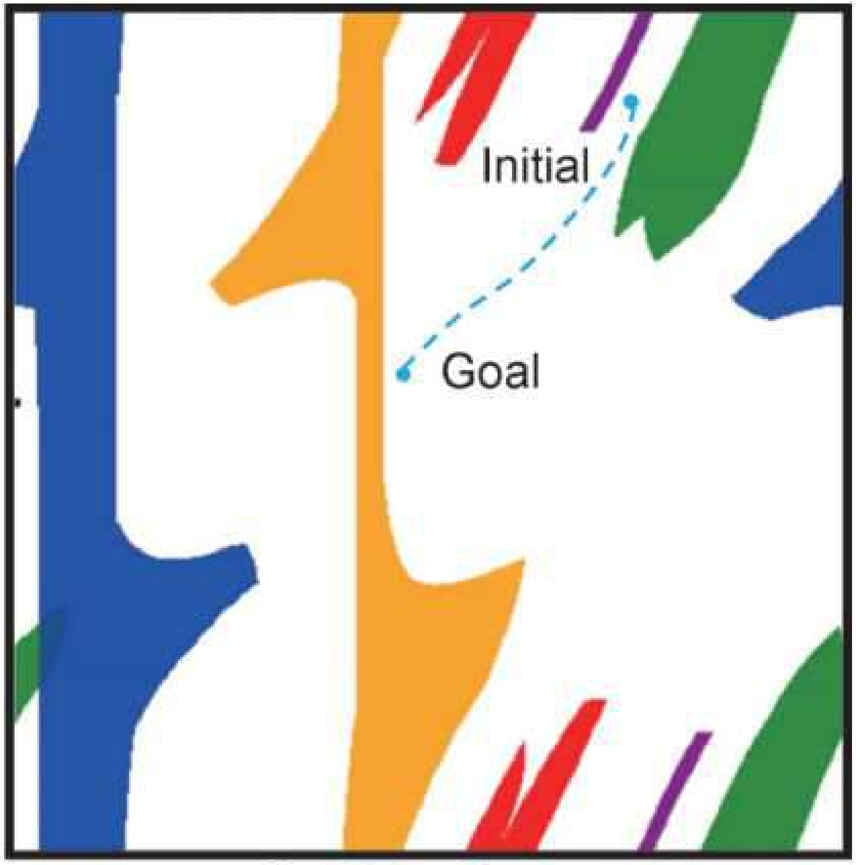
Workspace

How can we map this robot and its world into configuration space?

5 Configuration Space



Workspace

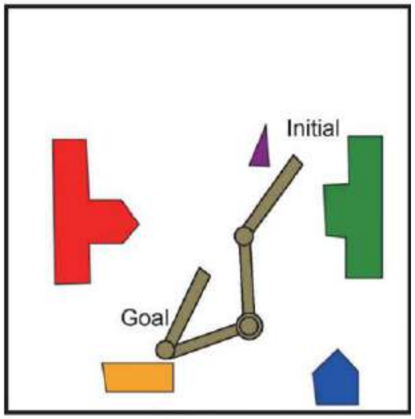


Configuration space

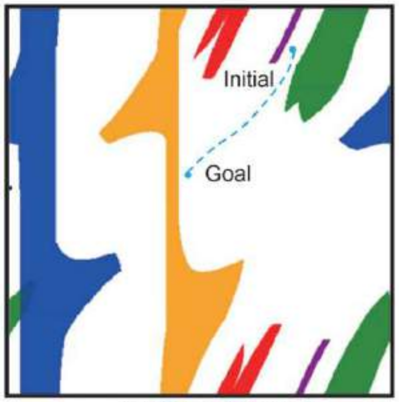
5 How to use configuration space in practice

If we map the obstacles into configuration space we can check whether the configuration point, *q*, is in an obstacle and we have a **unique plan** for the robot

- **Problem:** mapping obstacles into configuration space is hard



Workspace



Configuration space

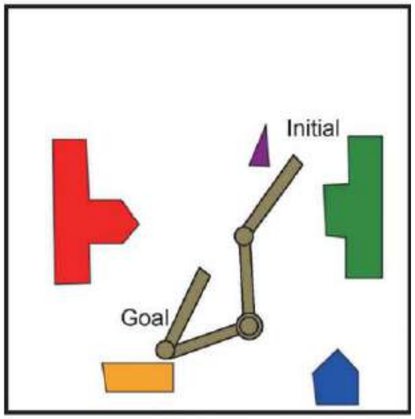
5 How to use configuration space in practice

If we map the obstacles into configuration space we can check whether the configuration point, q, is in an obstacle and we have a **unique plan** for the robot

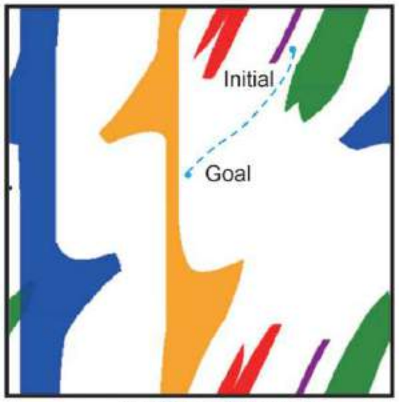
- **Problem:** mapping obstacles into configuration space is hard

Better approach: **use forward kinematics** to check task space obstacle collisions!

Treat the collision checker as a black box function evaluator!



Workspace



Configuration space

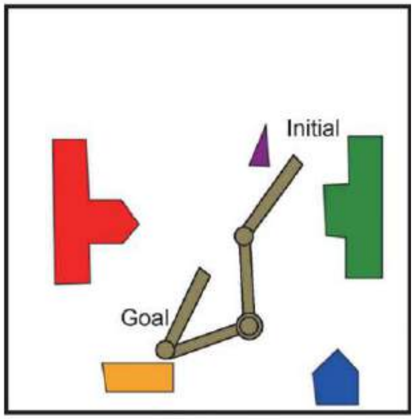
5 How to use configuration space in practice

If we map the obstacles into configuration space we can check whether the configuration point, *q*, is in an obstacle and we have a **unique plan** for the robot

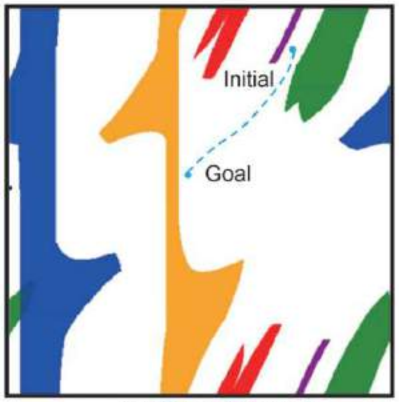
- **Problem:** mapping obstacles into configuration space is hard

Better approach: **use forward kinematics** to check task space obstacle collisions!

- **No free lunch** – Now each collision check requires full kinematics and not a simple lookup

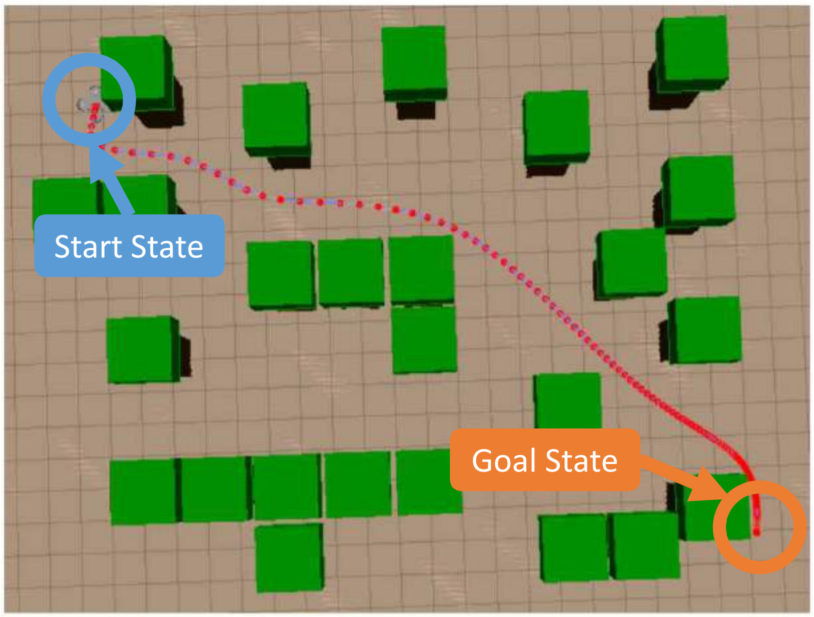


Workspace



Configuration space

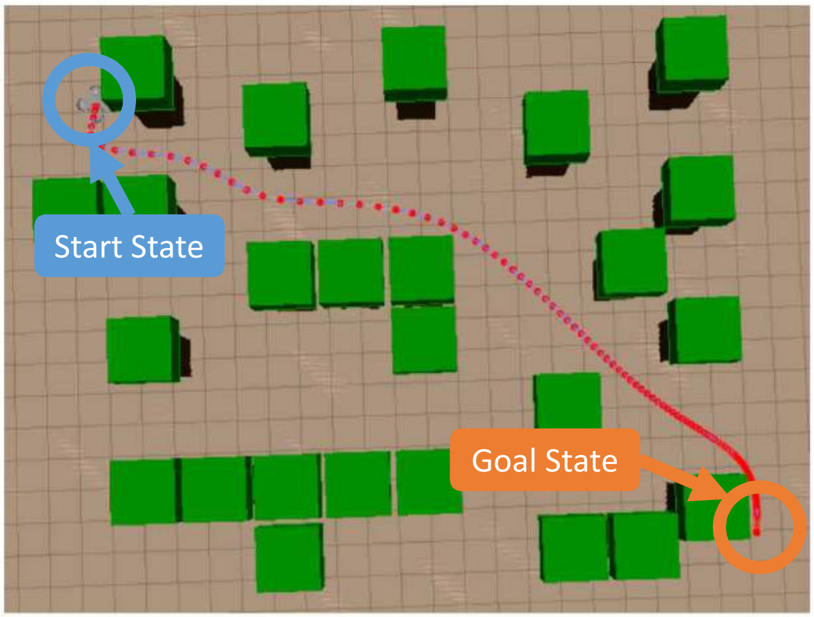
5 Planning in Configuration Space



Goal: Find shortest collision-free path from start to goal

States: configurations $q \in \mathcal{R}^6$ **Actions:** Δq **Transition:** $q' \leftarrow q + \Delta q$

5 Planning in Configuration Space

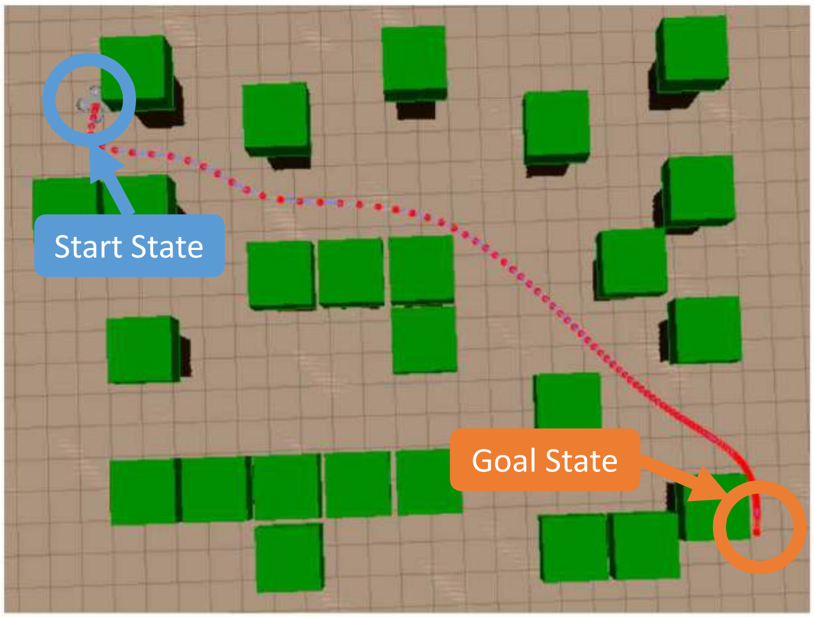


$q \in \mathcal{R}^6$: (x,y,z,θ,ϕ,φ)

Goal: Find shortest collision-free path from start to goal

States: configurations $q \in \mathcal{R}^6$ **Actions:** Δq **Transition:** $q' \leftarrow q + \Delta q$

5 Planning in Configuration Space

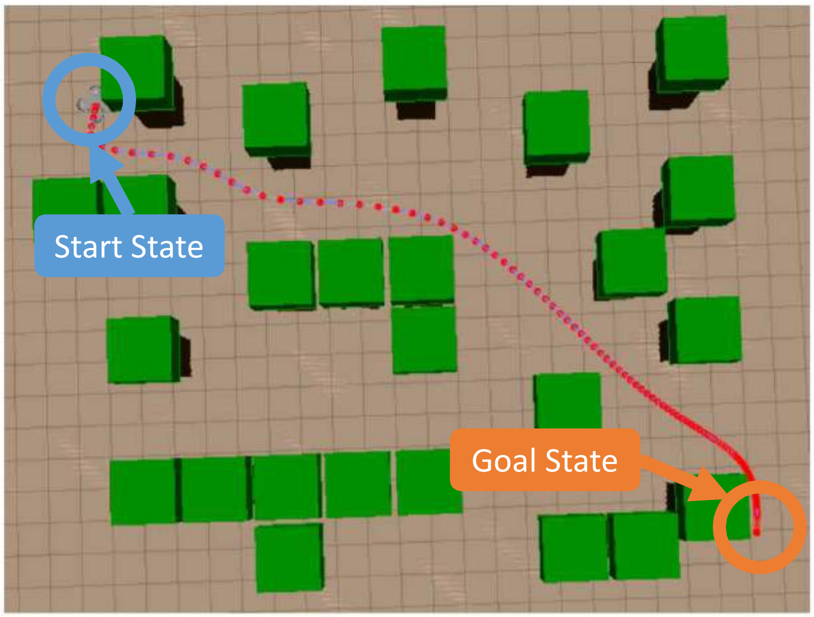


One approach is to ***discretize*** the statespace (grid it) and use graph search (think A* which is known fast)

Goal: Find shortest collision-free path from start to goal

States: configurations $q \in \mathcal{R}^6$ **Actions:** Δq **Transition:** $q' \leftarrow q + \Delta q$

5 Planning in Configuration Space



One approach is to ***discretize*** the statespace (grid it) and use graph search (think A* which is known fast)

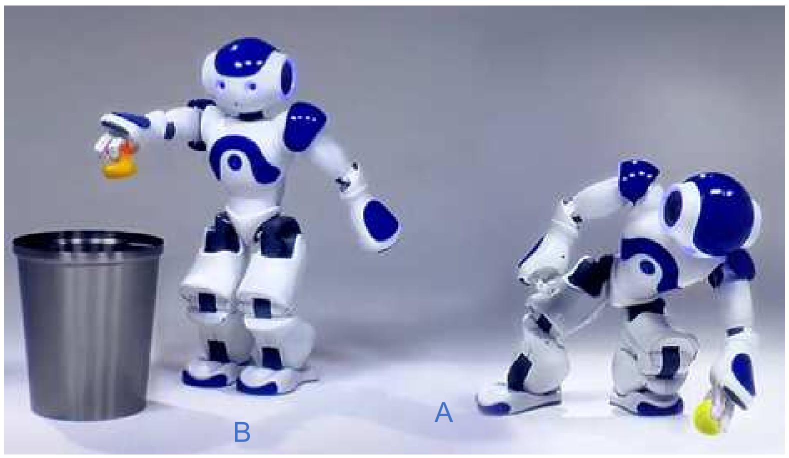
Unfortunately if we use say 100 discrete steps in each direction we get:

$$|S| = 100^6$$

Goal: Find shortest collision-free path from start to goal

States: configurations $q \in \mathcal{R}^6$ **Actions:** Δq **Transition:** $q' \leftarrow q + \Delta q$

5 Planning in Configuration Space



One approach is to ***discretize*** the statespace (grid it) and use graph search (think A* which is known fast)

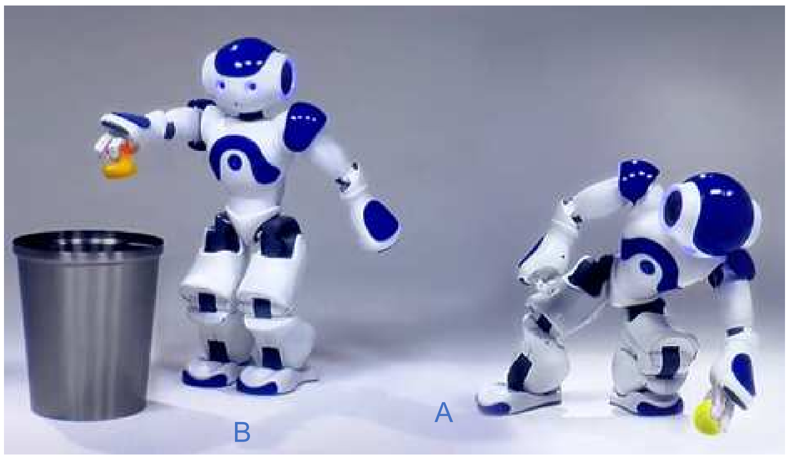
Unfortunately if we use say 100 discrete steps in each direction we get:

(2 ankles + 2 knees + 2 hips + 2 shoulders + 2 elbows + 4 fingers + pose of com) = $\sim$20 variables

Goal: Find shortest collision-free path fr

States: configurations $q \in \mathcal{R}^{20}$ **Actions:** Δ

5 Planning in Configuration Space



One approach is to ***discretize*** the statespace (grid it) and use graph search (think A* which is known fast)

Unfortunately if we use say 100 discrete steps in each direction we get:

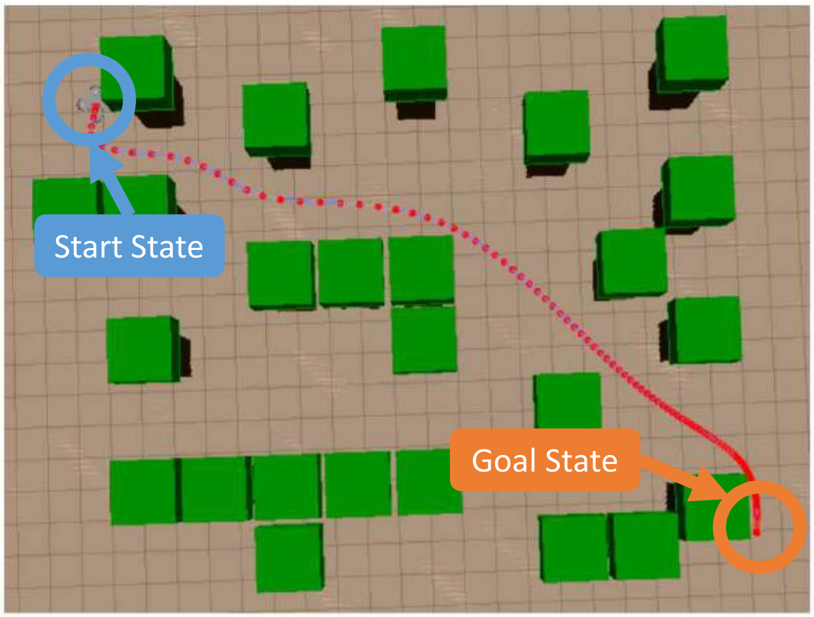
$$|S| = 100^{20}$$

Curse of Dimensionality!

Goal: Find shortest collision-free path from start to goal

States: configurations $q \in \mathcal{R}^{20}$ **Actions:** Δq **Transition:** $q' \leftarrow q + \Delta q$

5 Planning in Configuration Space

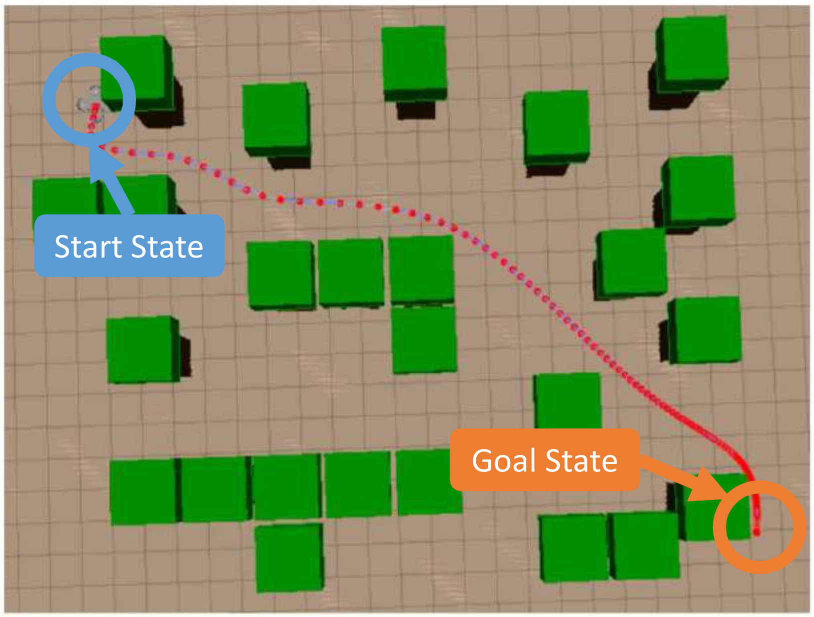


So if we can't explicitly form the graph and search the configuration space what can we do?

Goal: Find shortest collision-free path from start to goal

States: configurations $q \in \mathcal{R}^6$ **Actions:** Δq **Transition:** $q' \leftarrow q + \Delta q$

5 Planning in Configuration Space



What if we **incrementally build up a path** toward the goal?

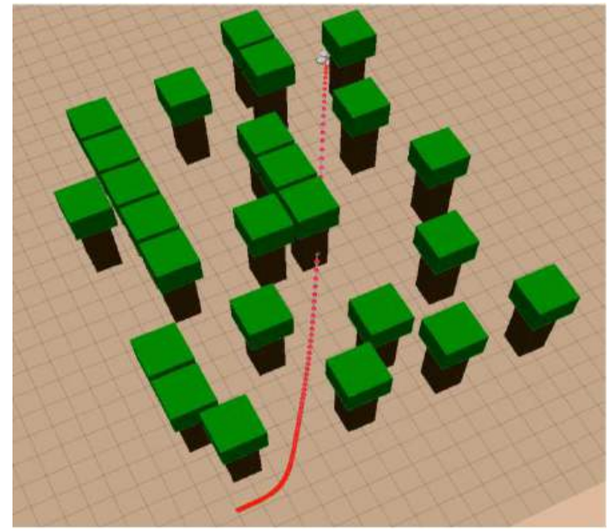
Goal: Find shortest collision-free path from start to goal

States: configurations $q \in \mathcal{R}^6$ **Actions:** Δq **Transition:** $q' \leftarrow q + \Delta q$

5 Planning in Configuration Space

Random Search

Machine Learning

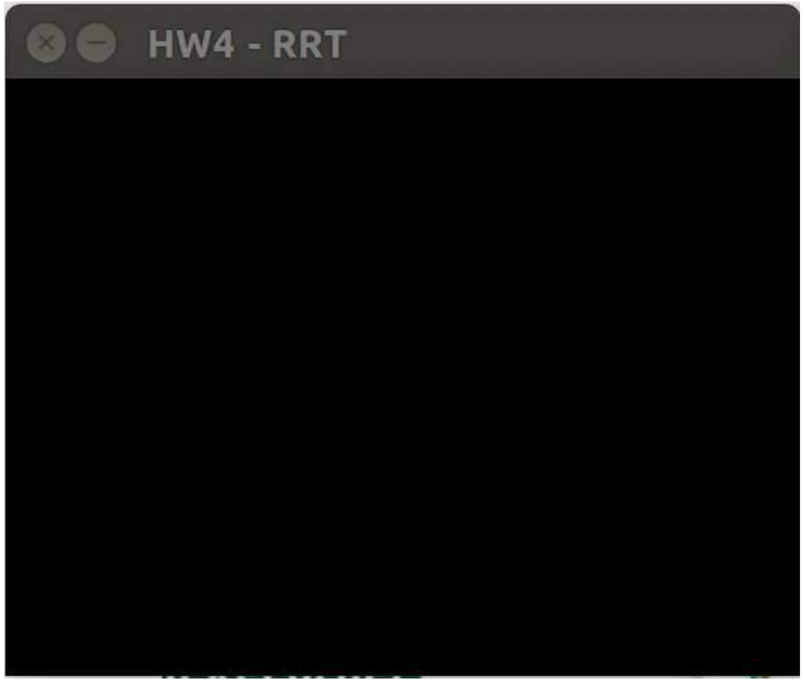


Local Search

5 Rapidly Exploring Random Trees (RRTs)

One of the most famous robot motion planning algorithms is **Rapidly Exploring Random Trees (RRTs)** [Lavalle & Kuffner]

The main idea is to **use randomness** to **rapidly explore** an entire state space to find a path from a given start location to the goal.

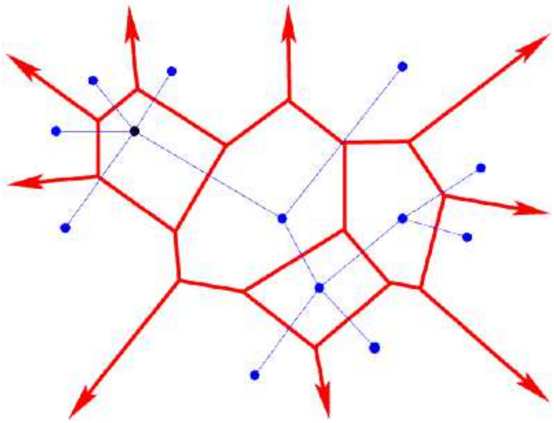


5 Randomness encourages exploration

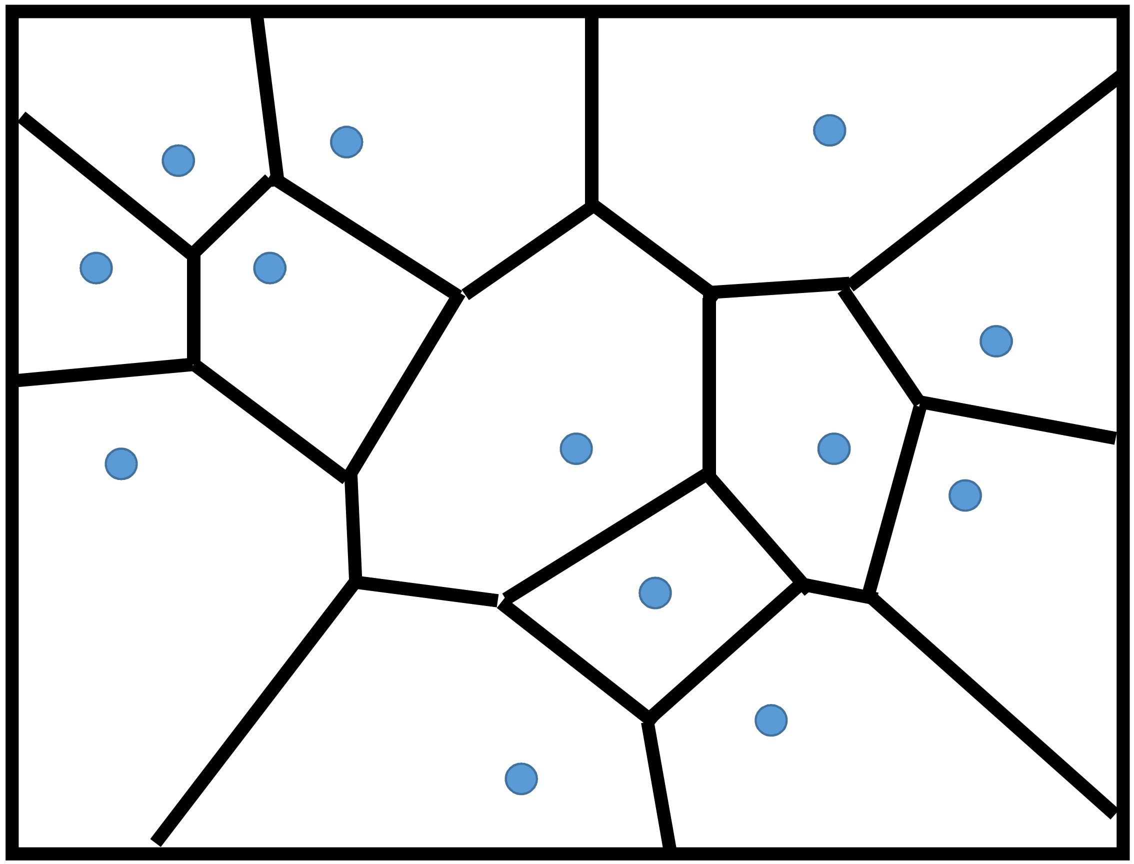
Key idea: uniform random sampling in configuration space is actually a heuristic that encourages exploration!

To see this we use ***Voronoi regions***

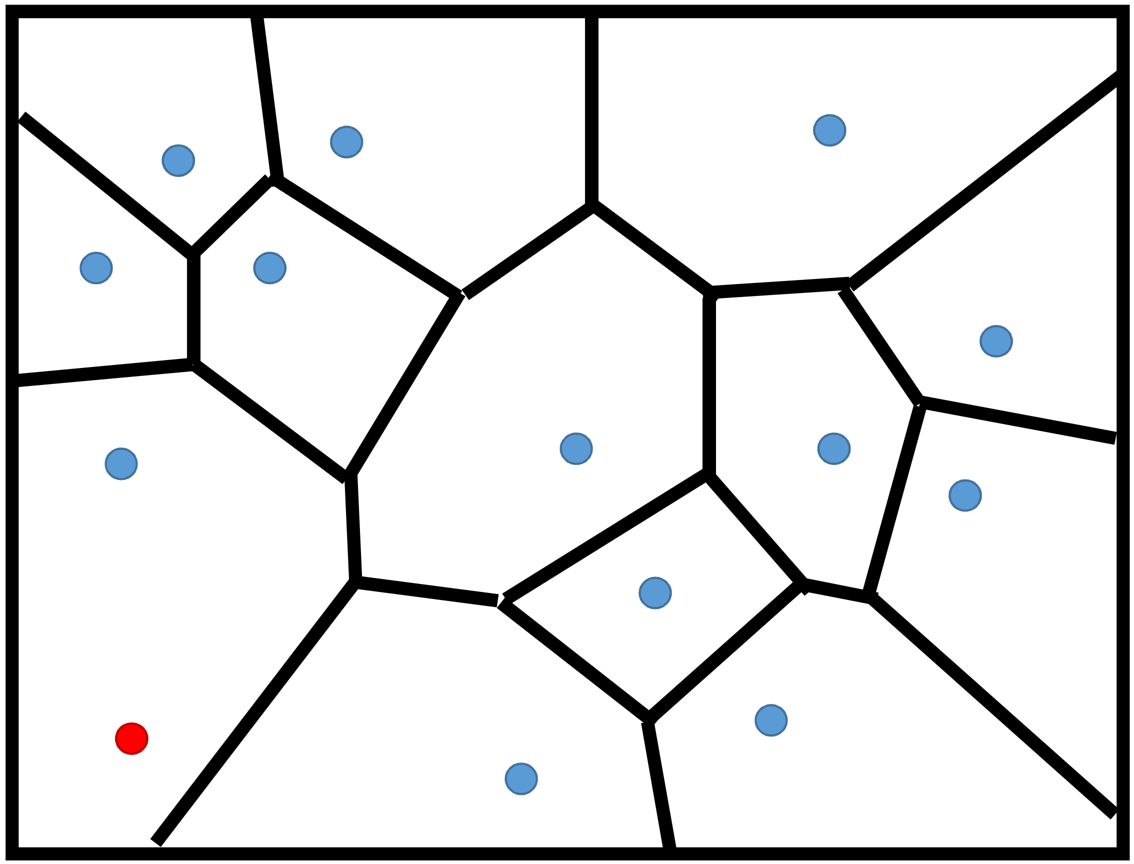
Def: Voronoi region is the set of points in space that are closest to a particular node in the tree:



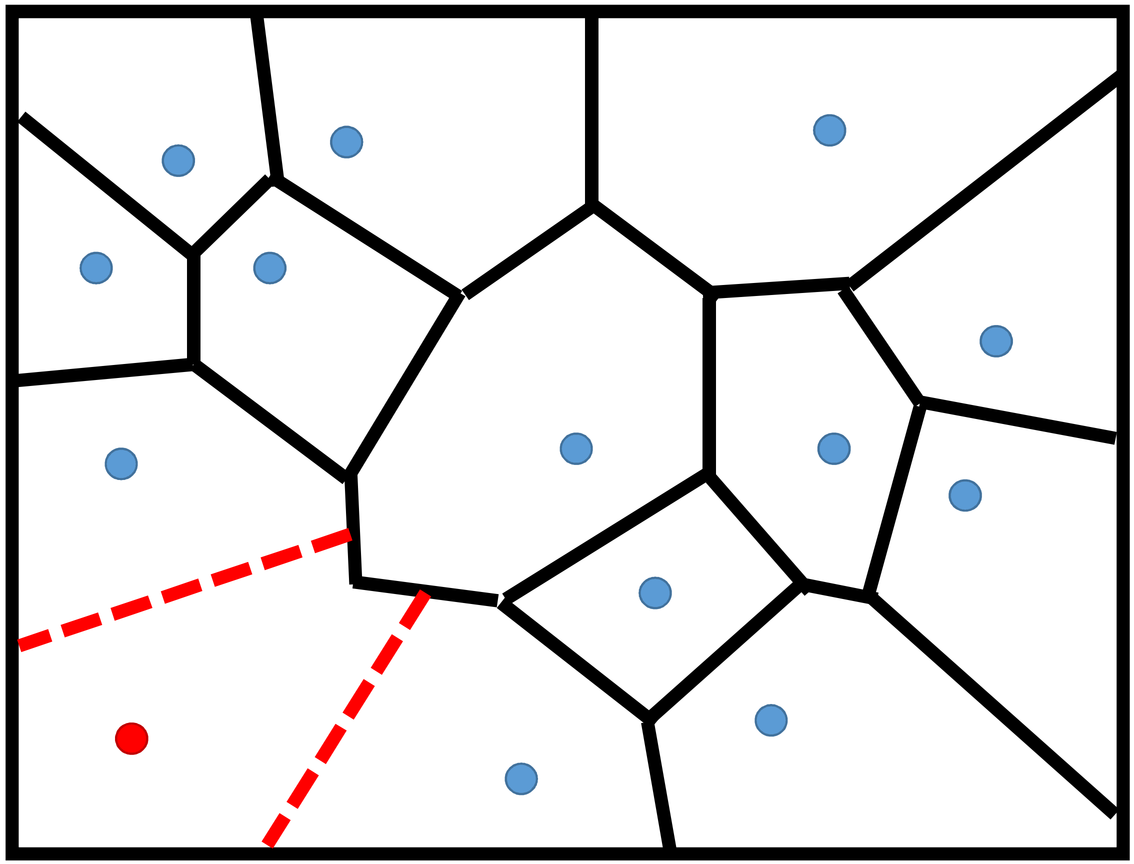
5 Randomness encourages exploration



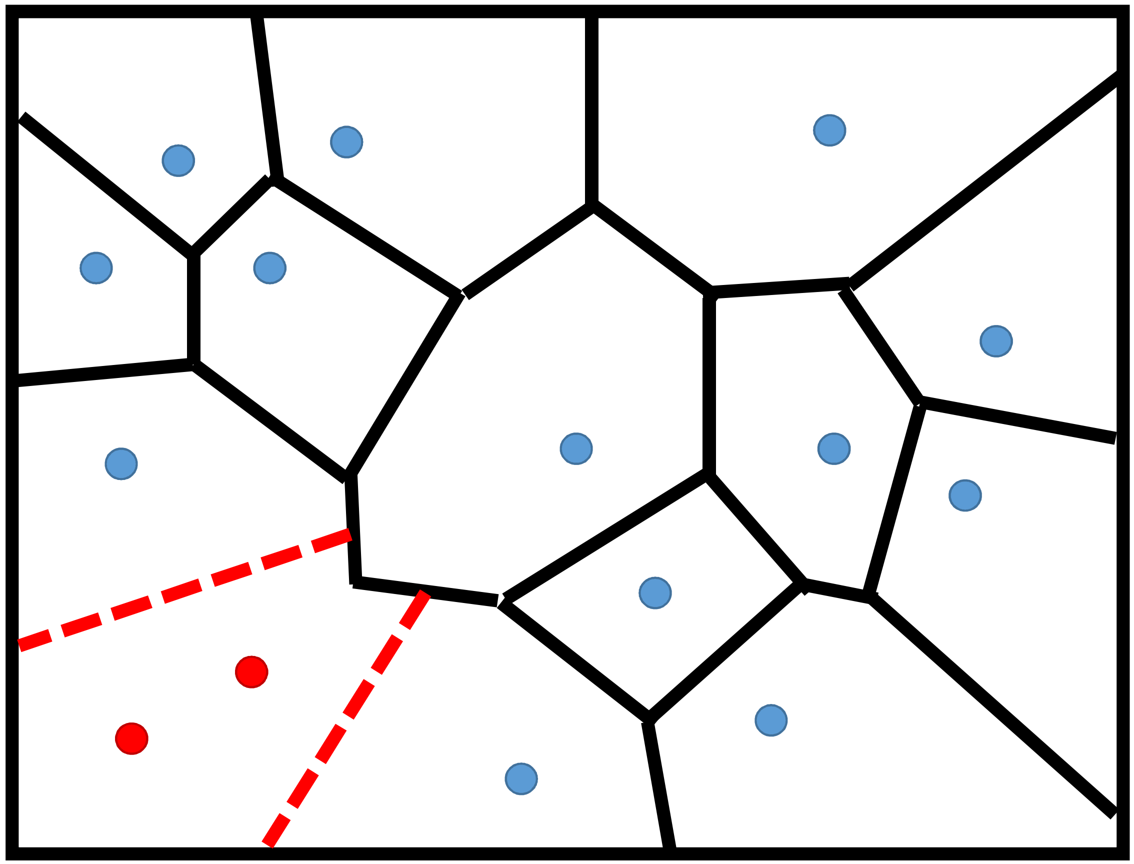
5 Randomness encourages exploration



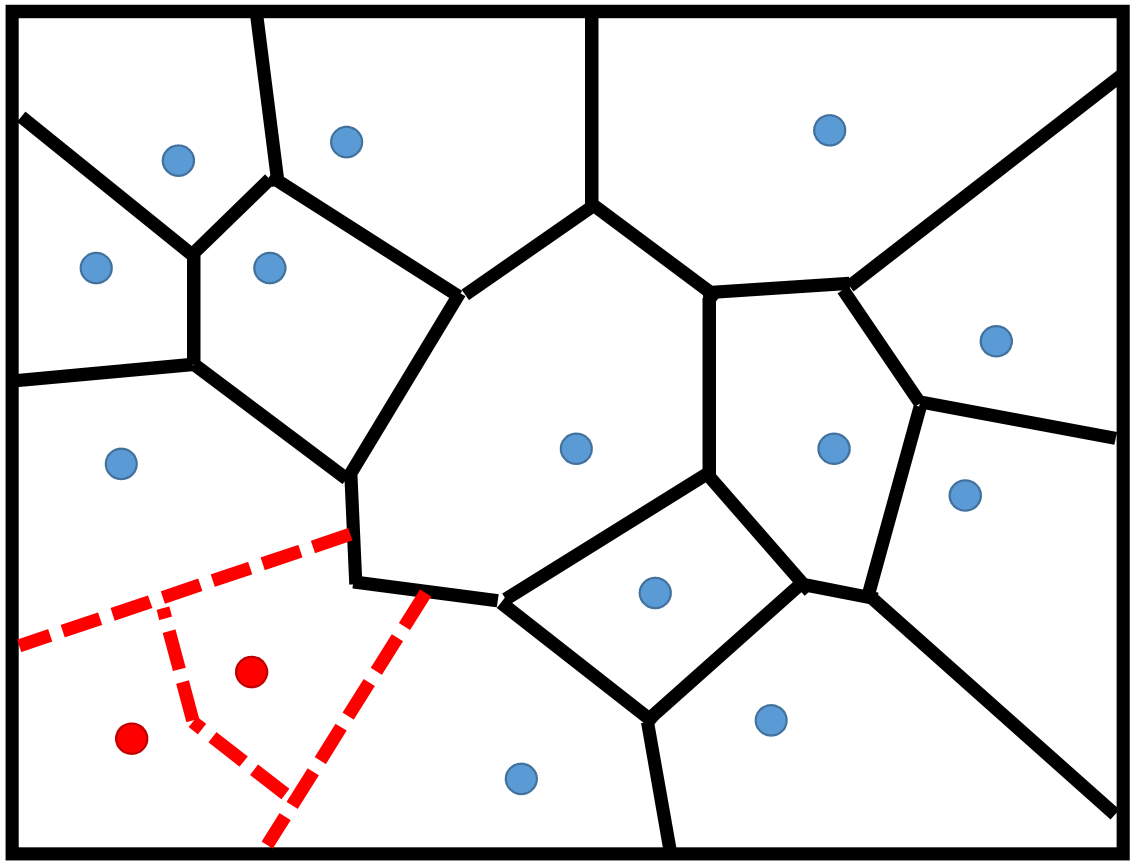
5 Randomness encourages exploration



5 Randomness encourages exploration



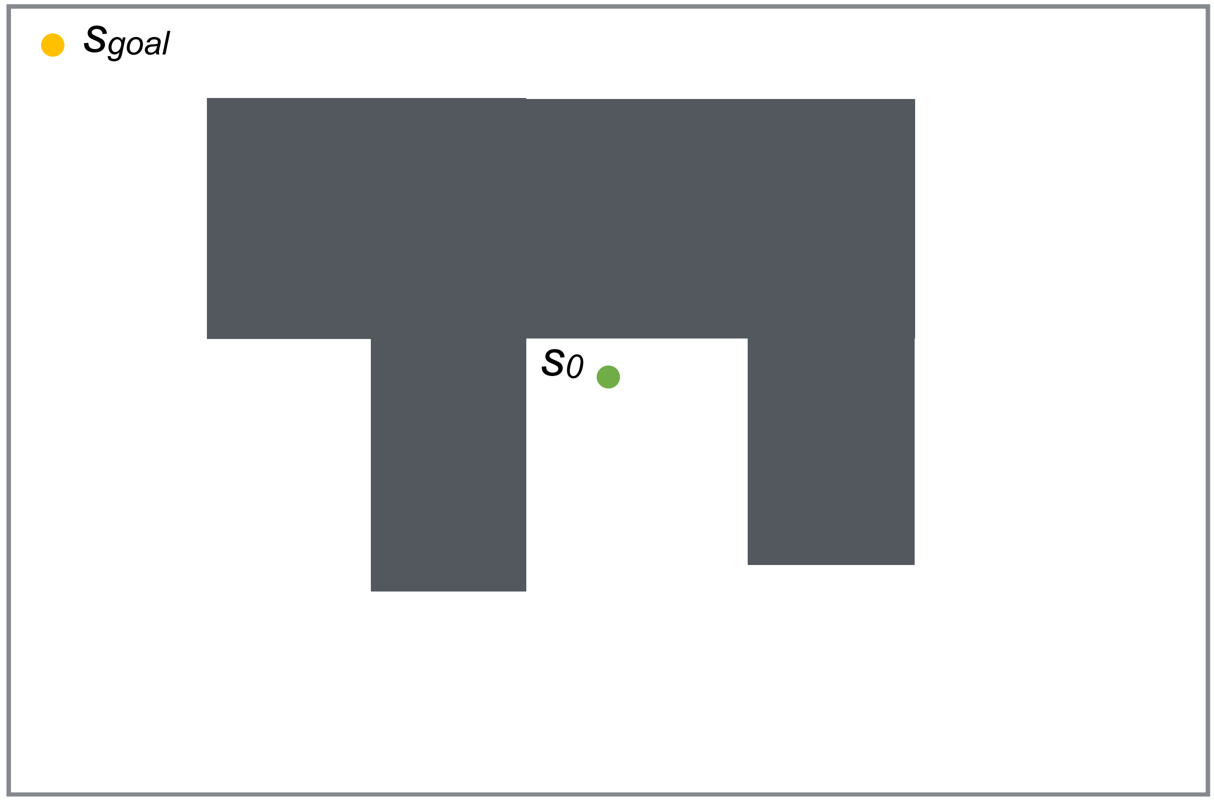
5 Randomness encourages exploration



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

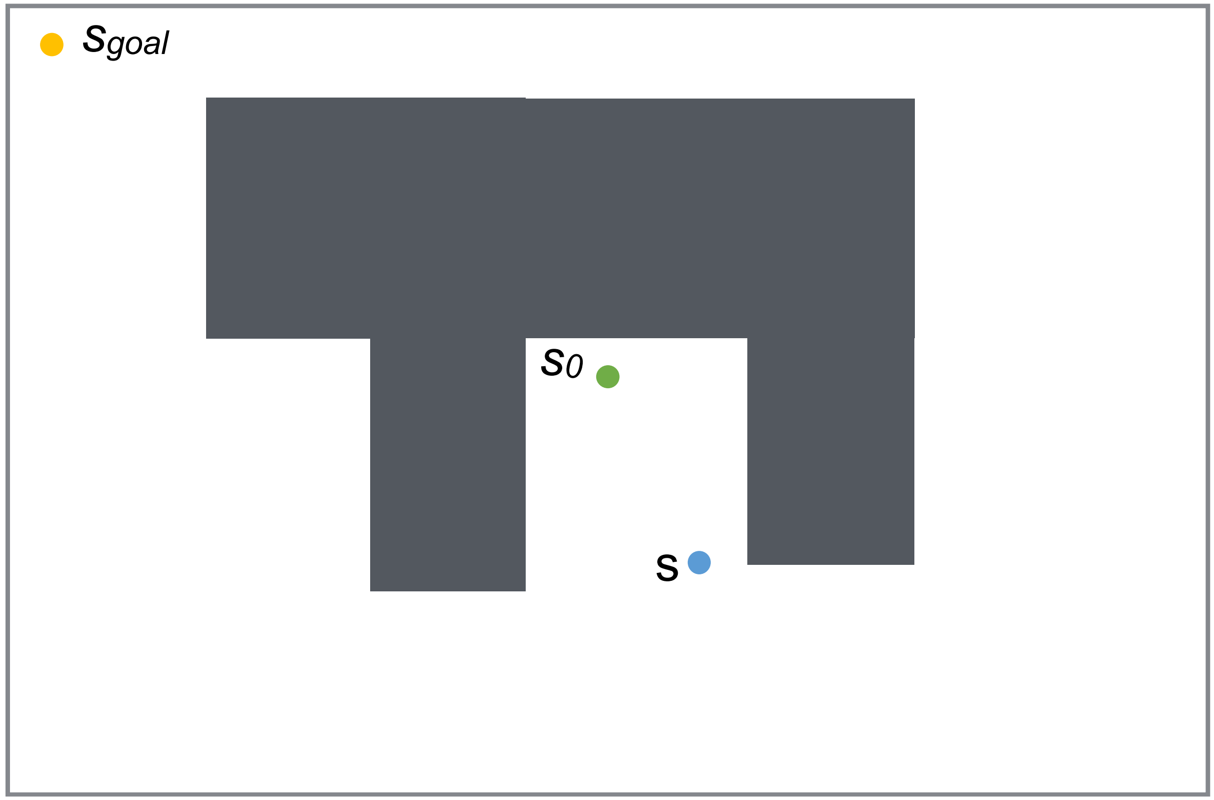
- Sample states $s \in S = R^n$ until s is collision-free
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

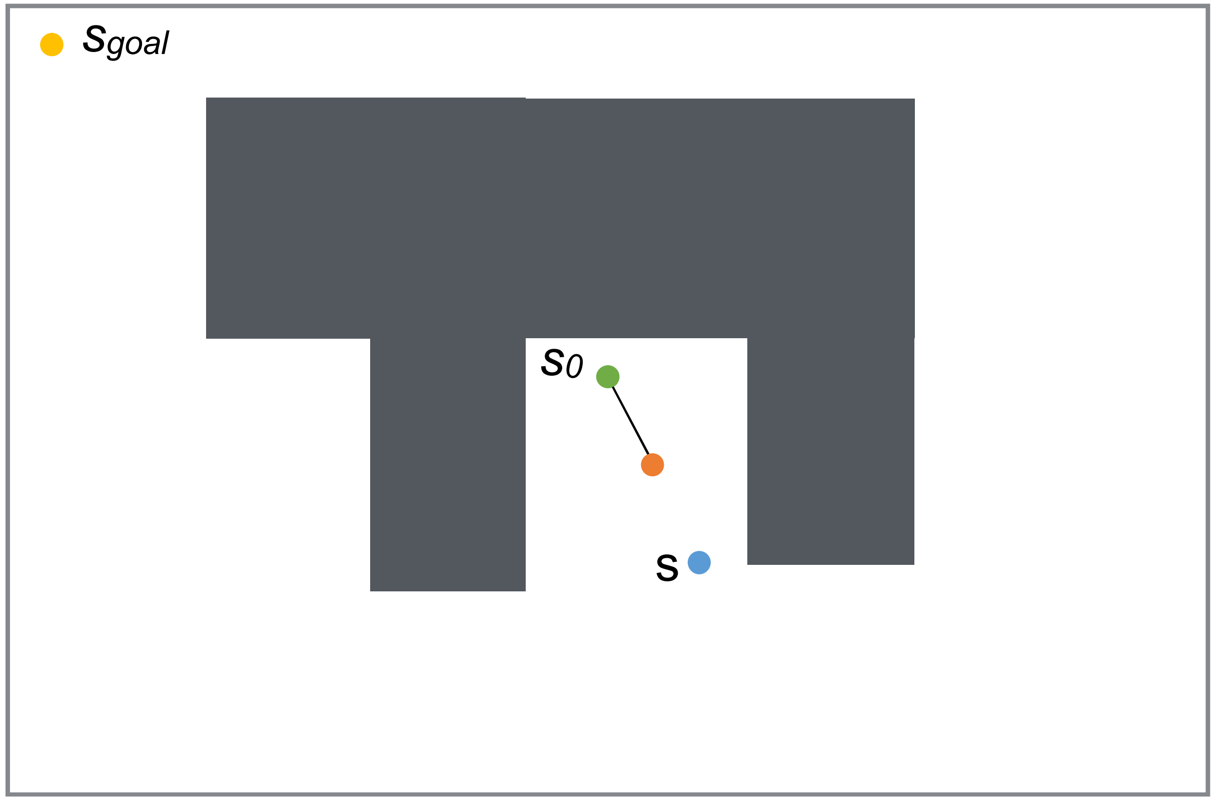
- **Sample states $s \in S = R^n$ until s is collision-free**
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

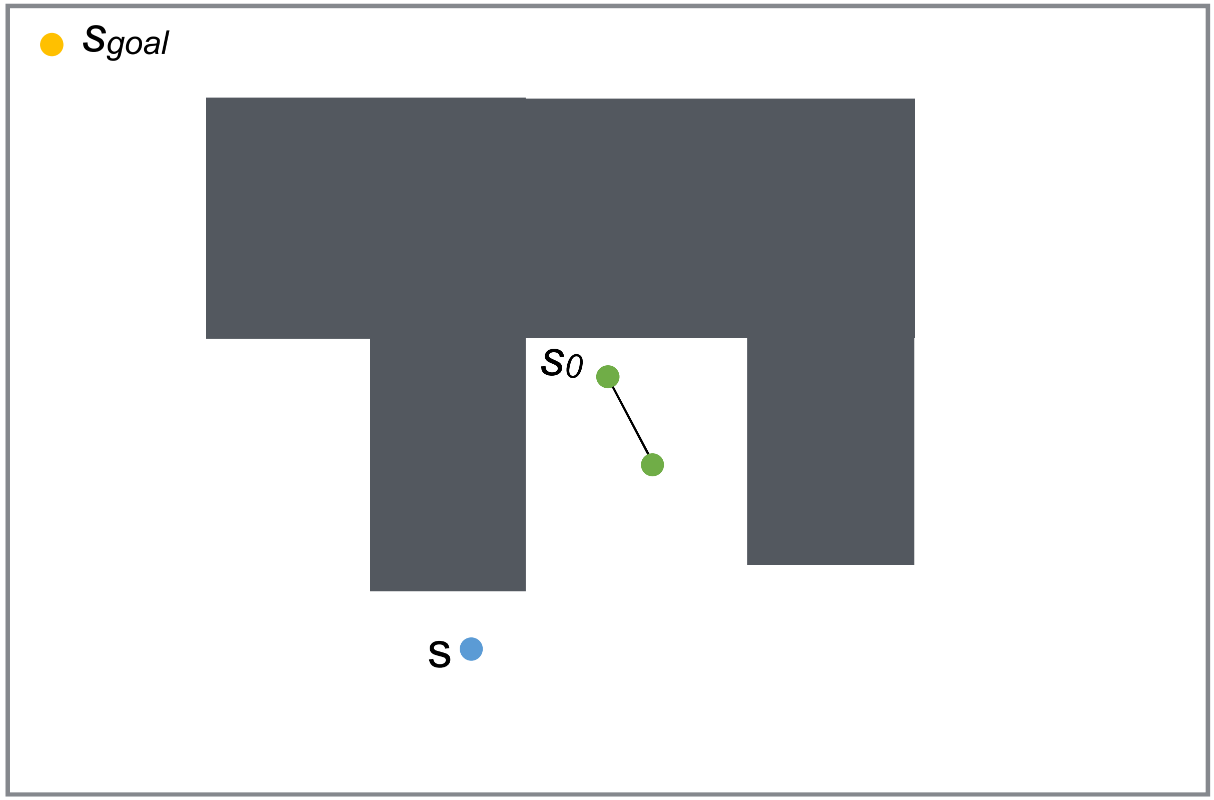
- Sample states $s \in S = R^n$ until s is collision-free
- **Find closest state $s_c \in T$**
- **Extend s_c toward s**
- **Add resulting state s' to T**
- Repeat until T contains a path from s_0 to s_{goal}



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

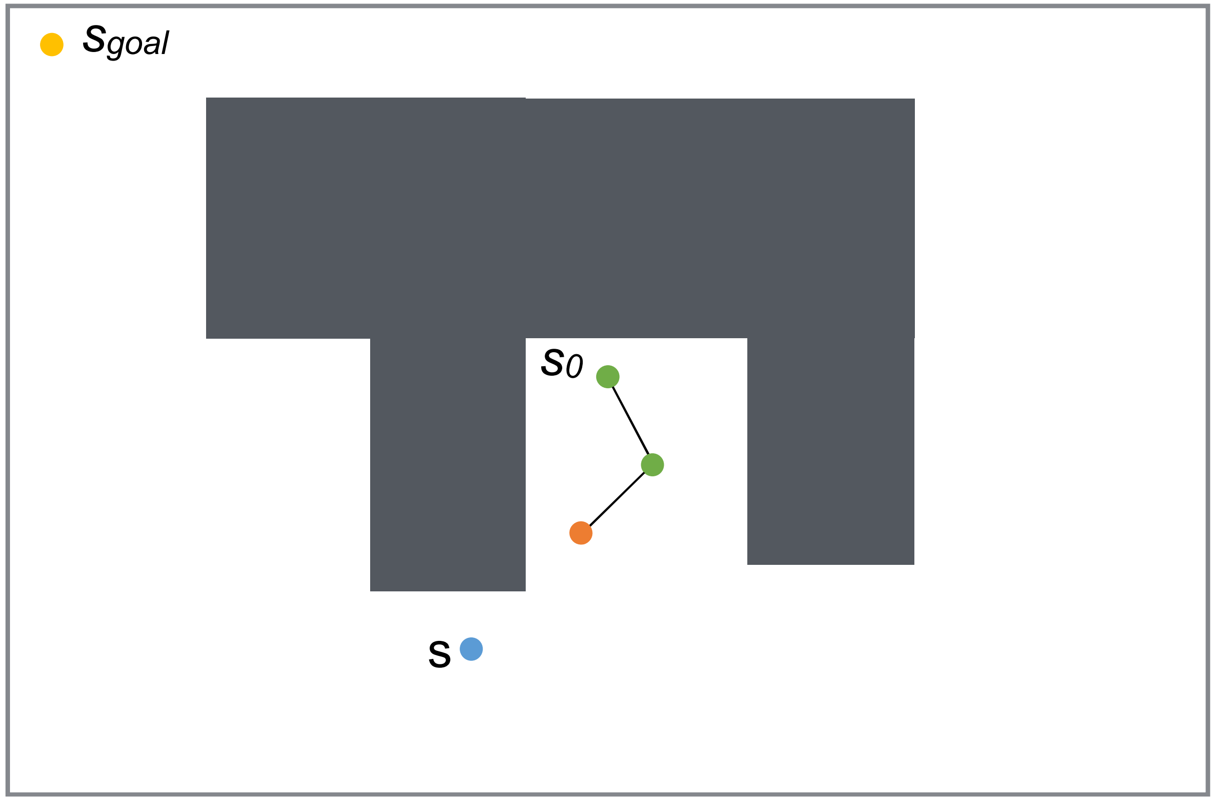
- **Sample states $s \in S = R^n$ until s is collision-free**
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

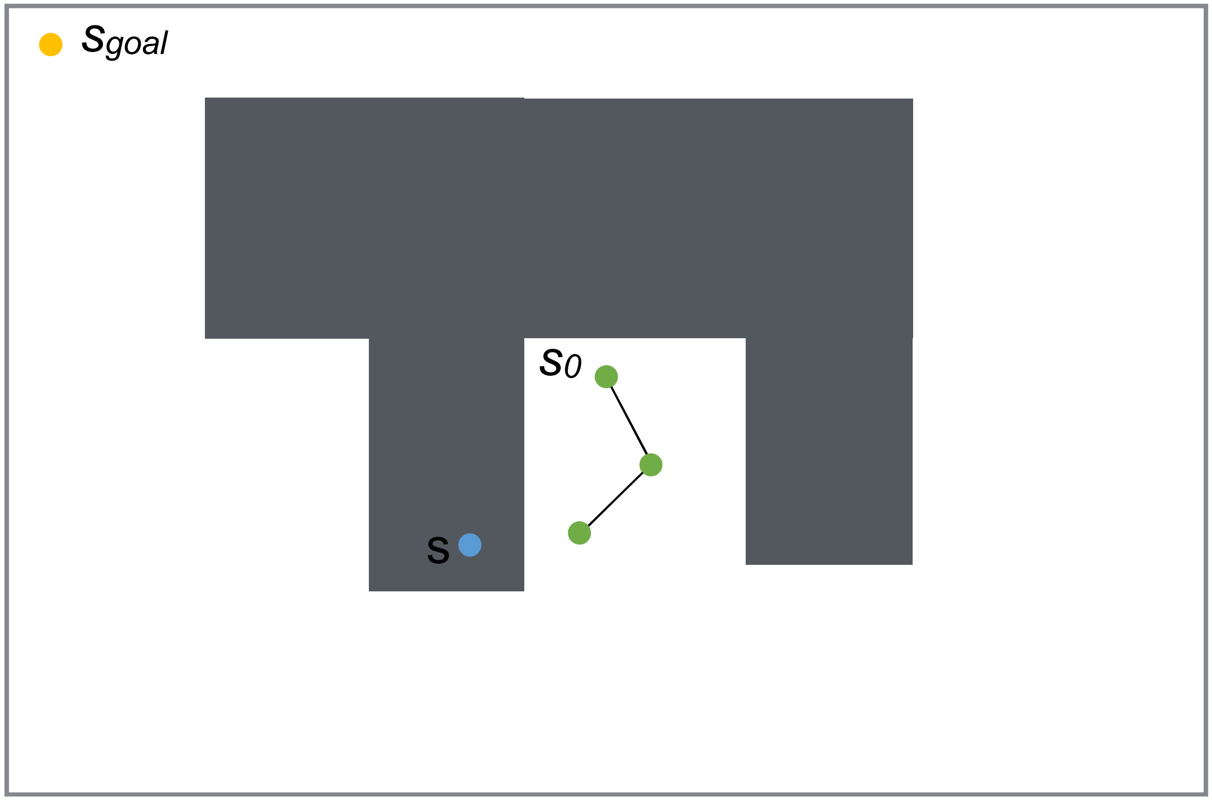
- Sample states $s \in S = R^n$ until s is collision-free
- **Find closest state $s_c \in T$**
- **Extend s_c toward s**
- **Add resulting state s' to T**
- Repeat until T contains a path from s_0 to s_{goal}



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

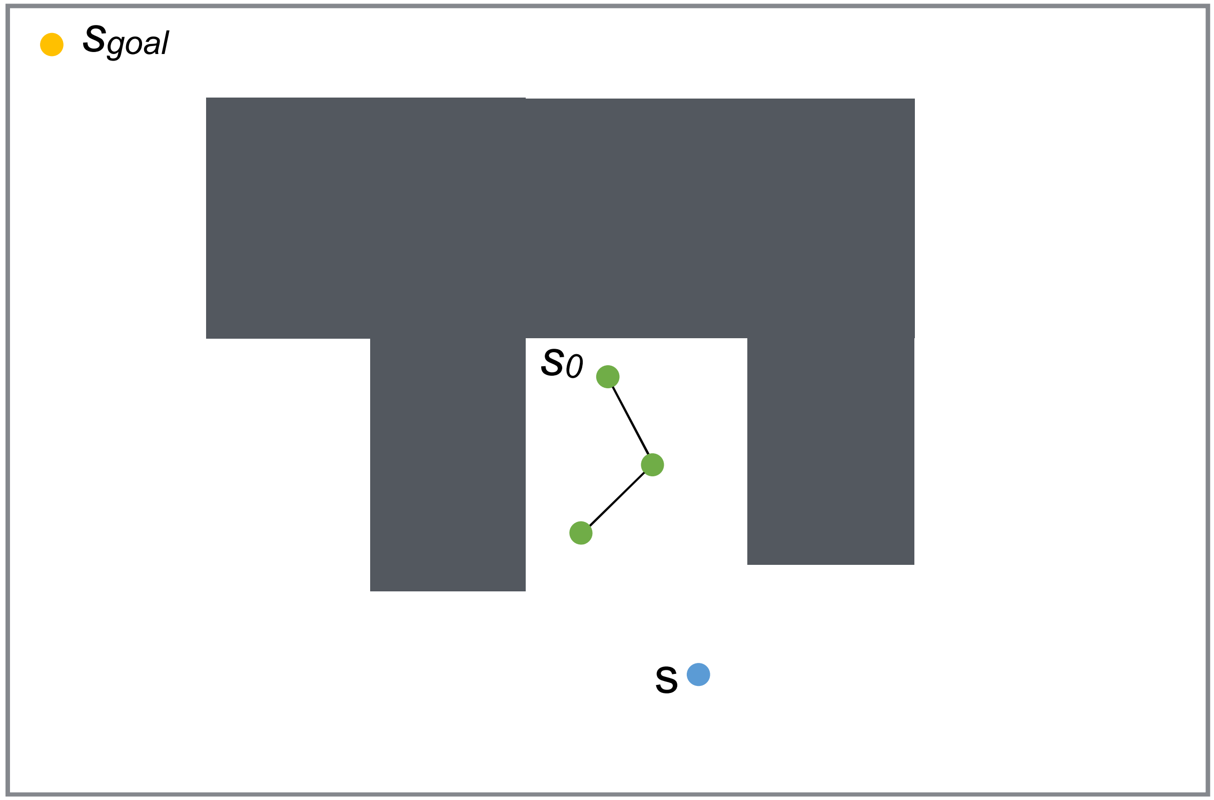
- **Sample states $s \in S = R^n$ until s is collision-free**
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

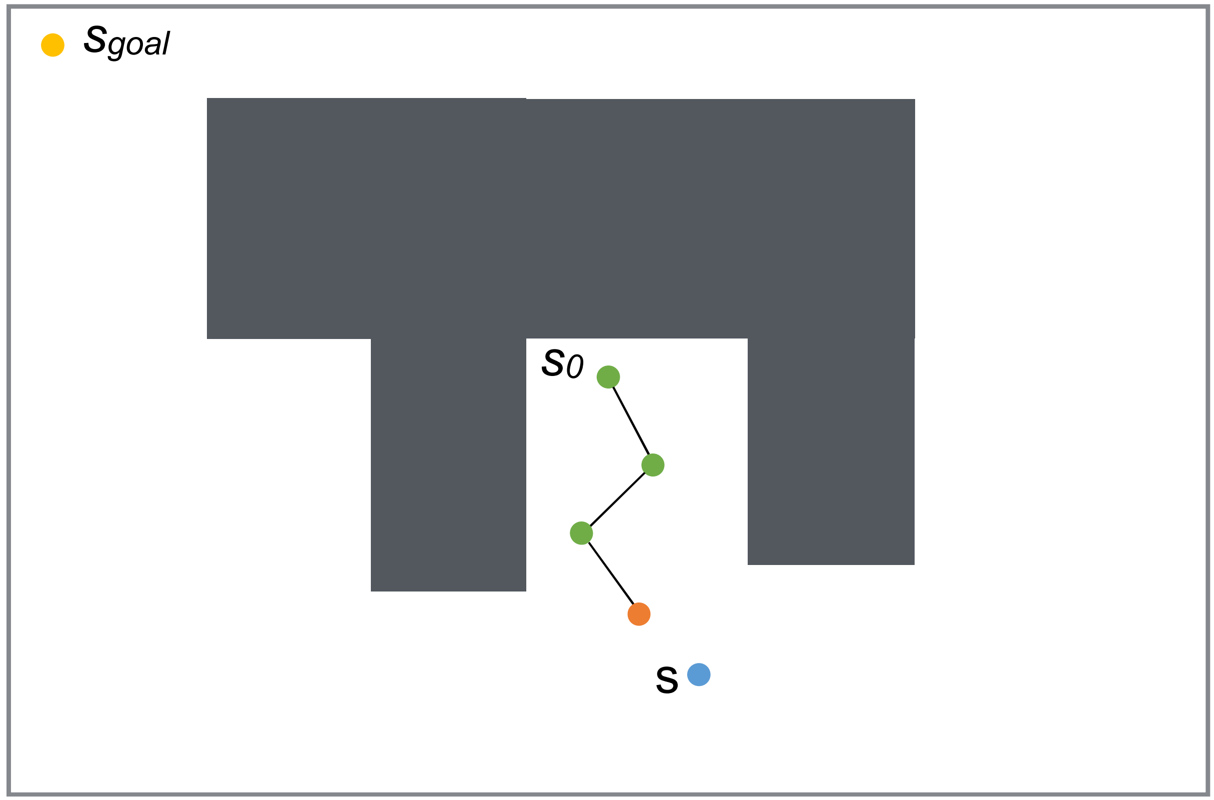
- **Sample states $s \in S = R^n$ until s is collision-free**
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

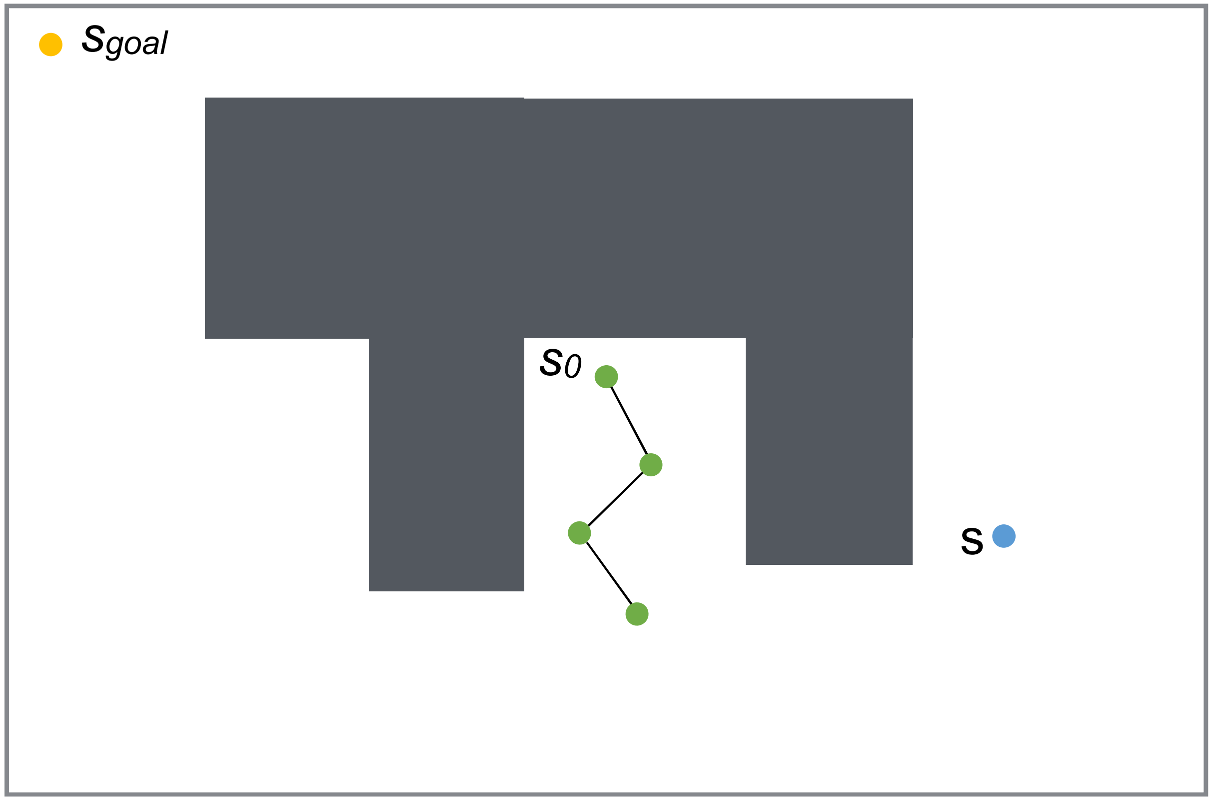
- Sample states $s \in S = R^n$ until s is collision-free
- **Find closest state $s_c \in T$**
- **Extend s_c toward s**
- **Add resulting state s' to T**
- Repeat until T contains a path from s_0 to s_{goal}



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

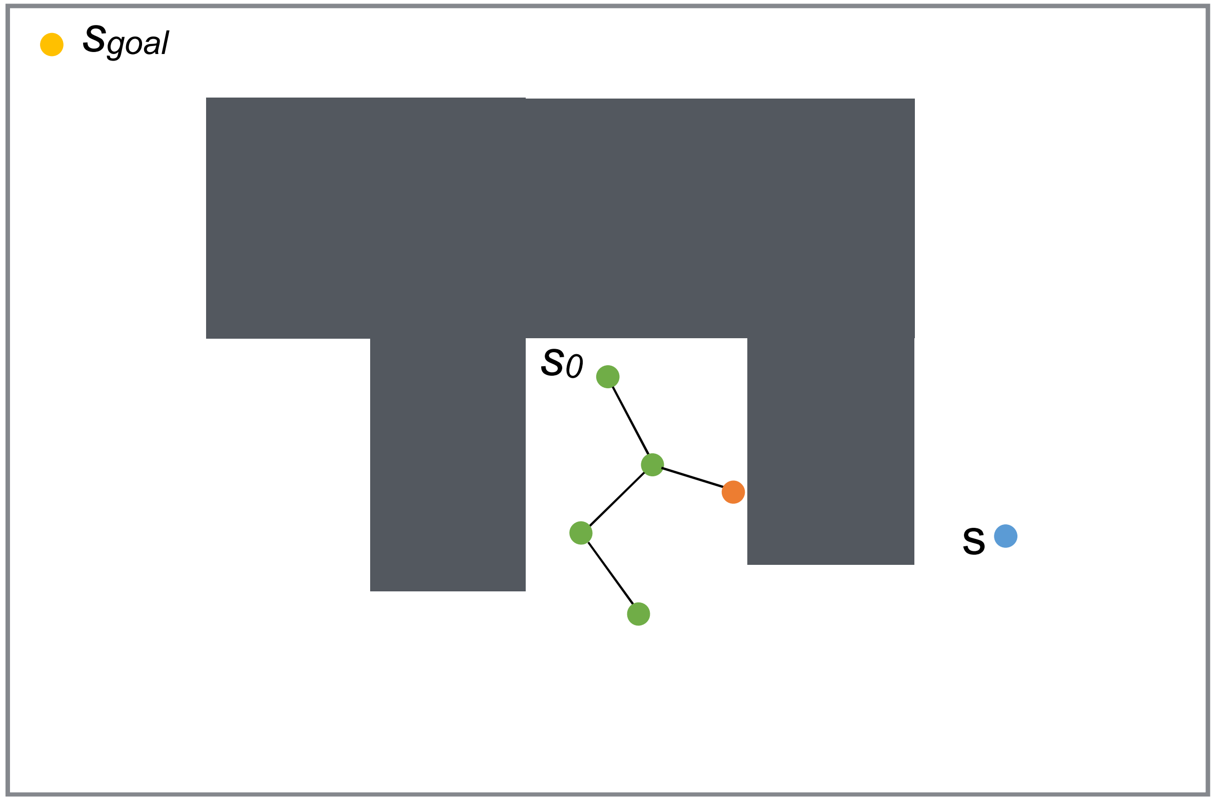
- **Sample states $s \in S = R^n$ until s is collision-free**
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

- Sample states $s \in S = R^n$ until s is collision-free
- **Find closest state $s_c \in T$**
- **Extend s_c toward s**
- **Add resulting state s' to T**
- Repeat until T contains a path from s_0 to s_{goal}

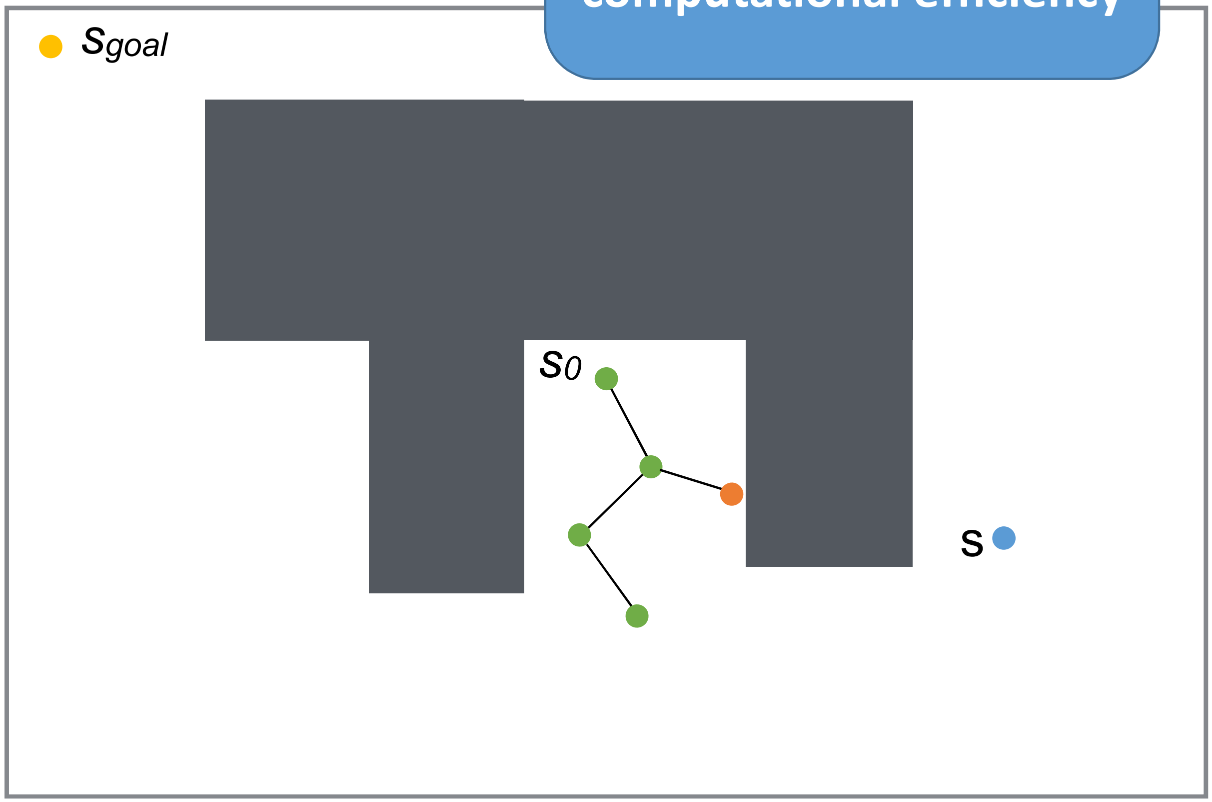


5 Rapidly Exploring Random Trees (RRTs)

Extend distance trades off sample vs. computational efficiency

Algorithm (input: s_0, s_{goal}, initial state tree T)

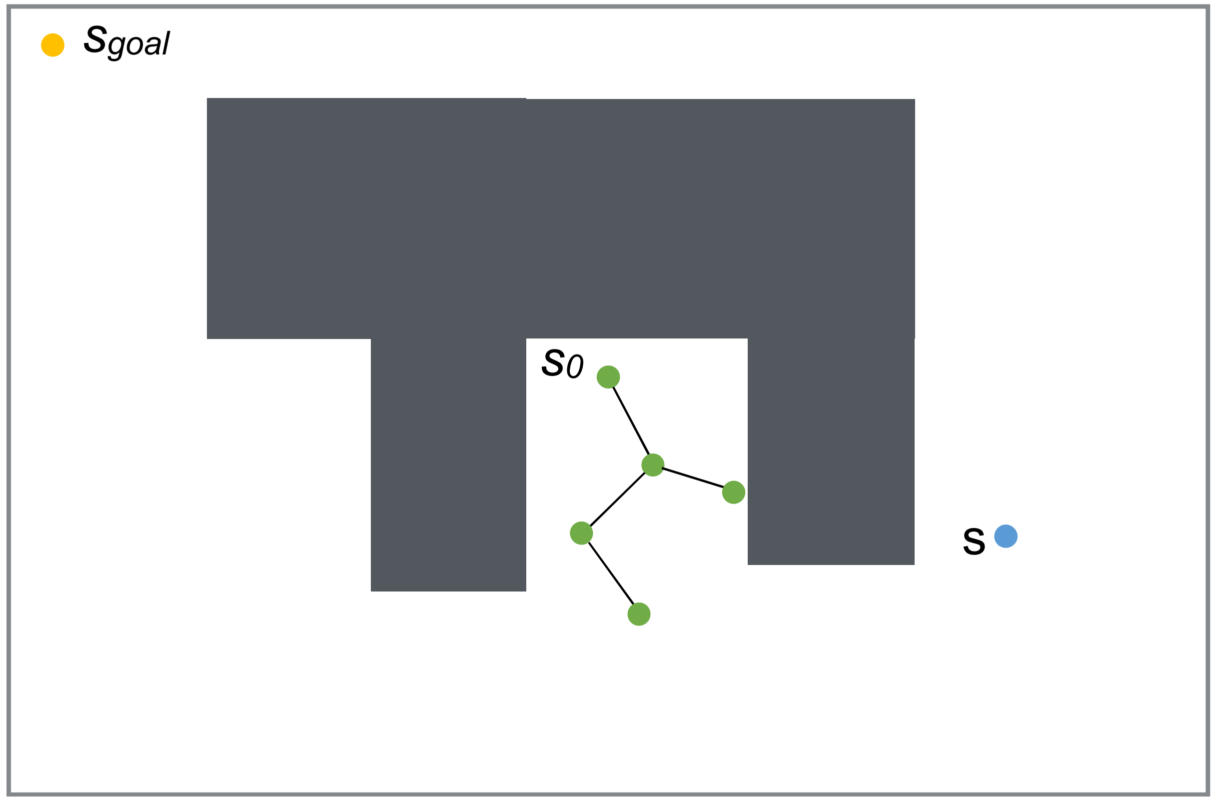
- Sample states $s \in S = R^n$ until s is collision-free
- **Find closest state $s_c \in T$**
- **Extend s_c toward s**
- **Add resulting state s' to T**
- Repeat until T contains a path from s_0 to s_{goal}



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

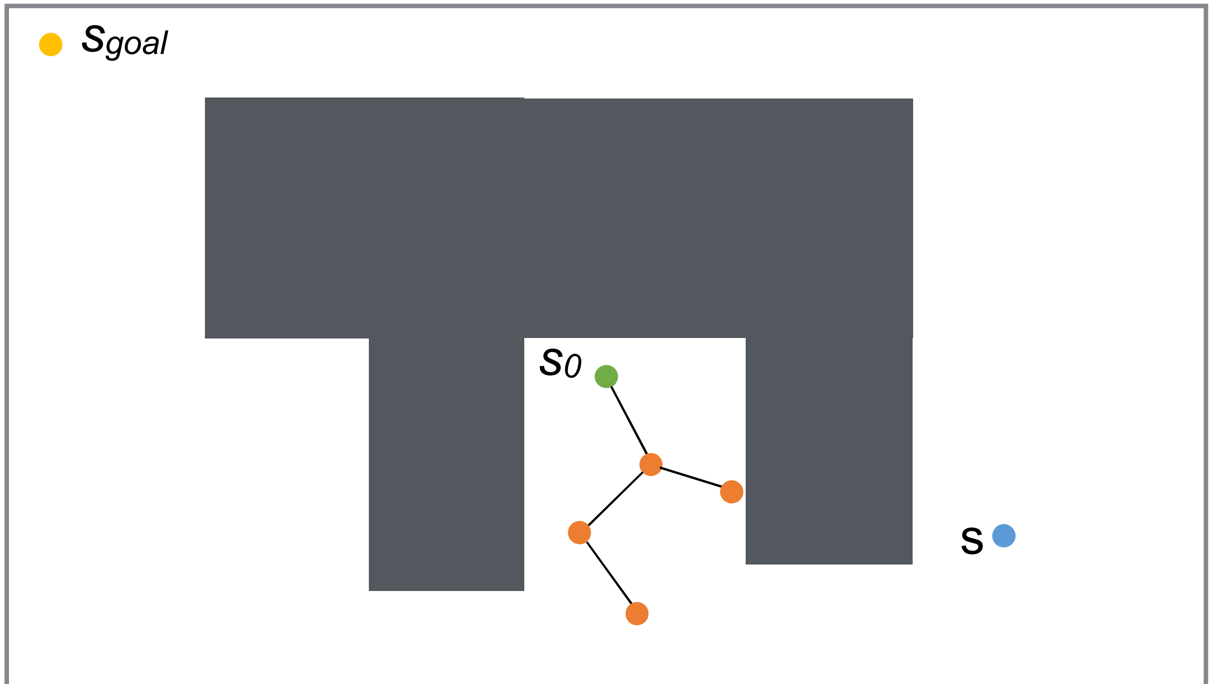
- Sample states $s \in S = R^n$ until s is collision-free
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- **Repeat until T contains a path from s_0 to s_{goal}**



5 Rapidly Exploring Random Trees (RRTs)

Algorithm (input: s_0, s_{goal}, initial state tree T)

- Sample states $s \in S = R^n$ until s is collision-free
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- **Repeat until T contains a path from s_0 to s_{goal}**



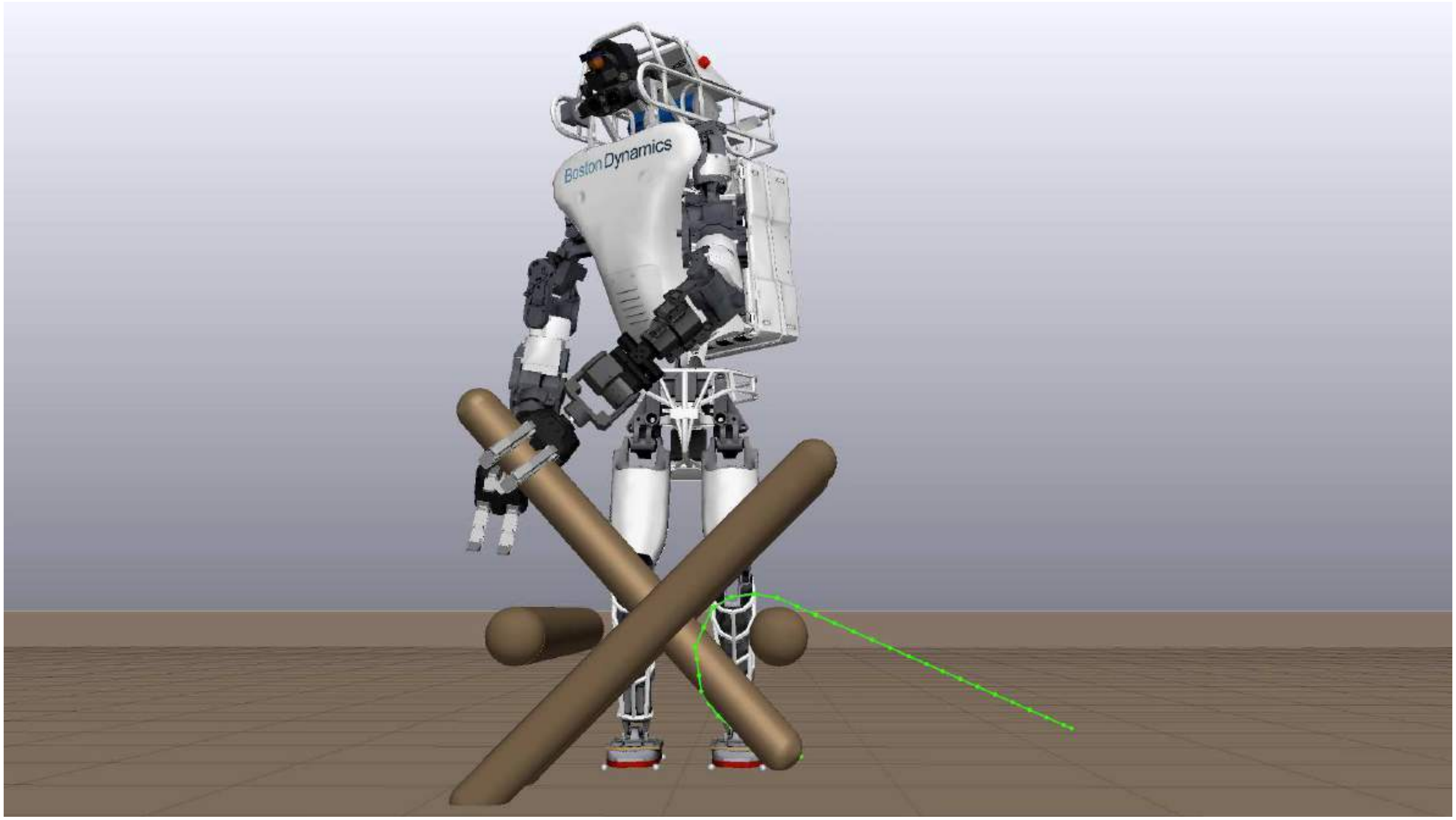
It will always find a solution because it is **probabilistically complete**

5 Rapidly Exploring Random Trees (RRTs)

5 Rapidly Exploring Random Trees (RRTs)

Biased sampling can help!

5 RRTs often works really well in practice



5 RRTs often works really well in practice

204 J. Leonard et al.

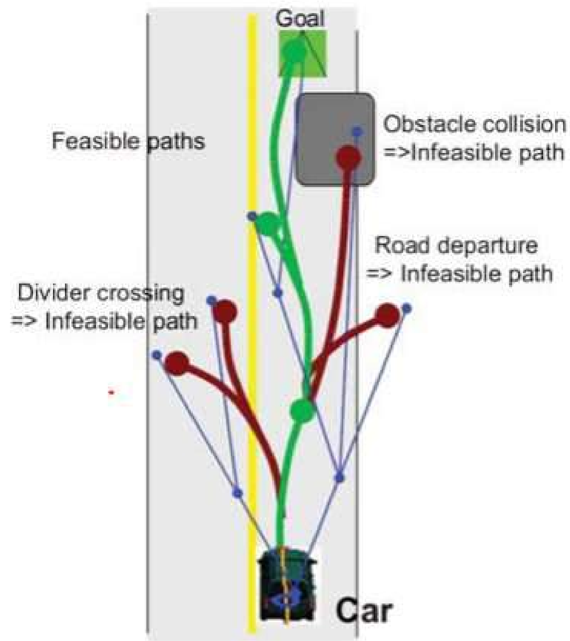
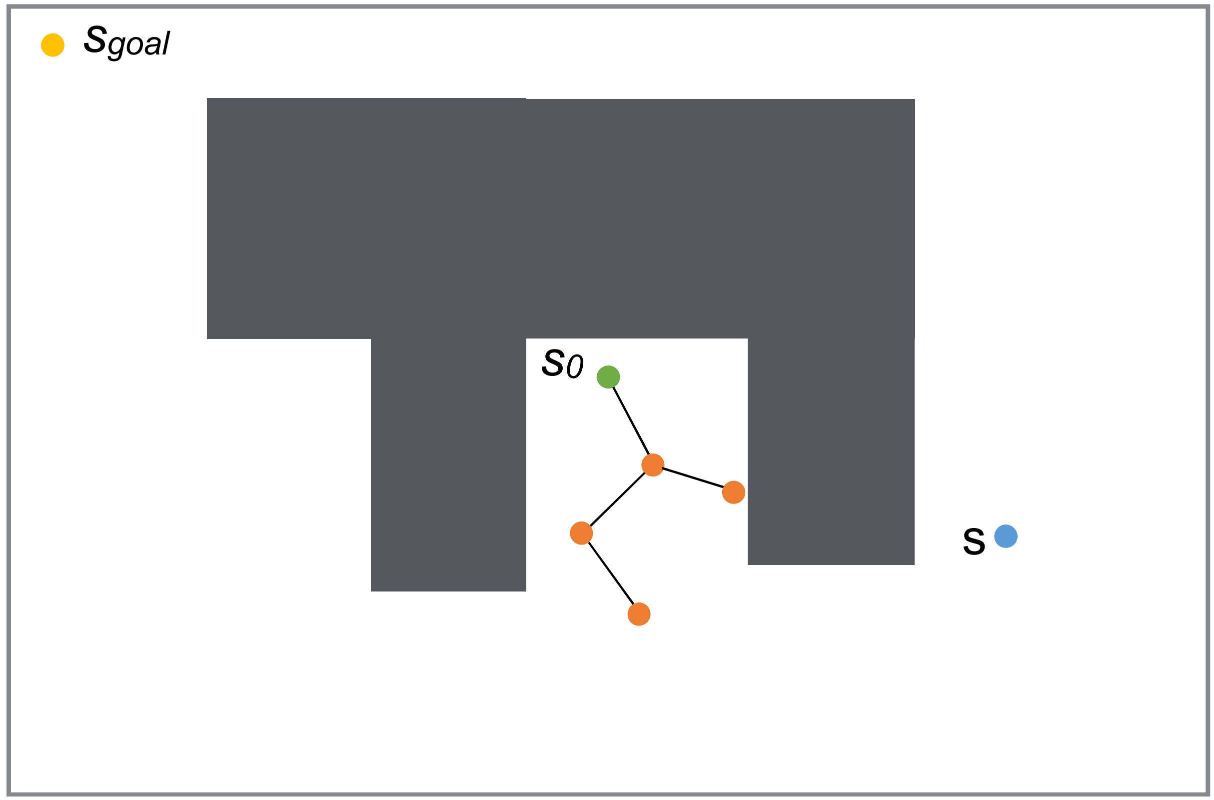


Fig. 22. Illustration of RRT Motion planning. Each leaf of the tree represents a stopping location. The motion control points (in blue) are translated into a predicted path. The predicted paths are checked for drivability (shown in green and red).

5 Questions about the RRT algorithm?

Algorithm (input: s_0, s_{goal}, initial state tree T)

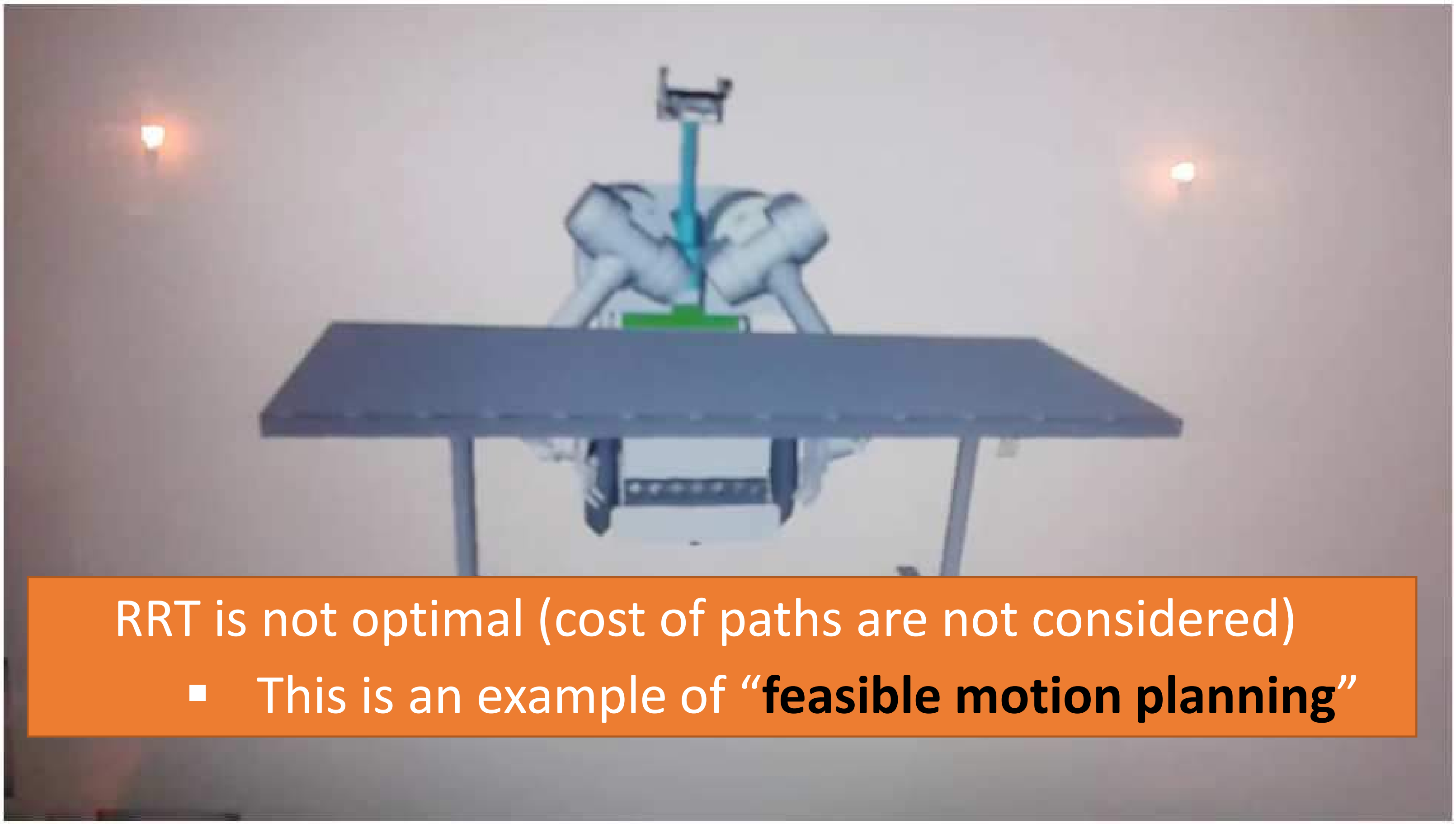
- Sample states $s \in S = R^n$ until s is collision-free
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- **Repeat until T contains a path from s_0 to s_{goal}**



5 But we can get some WEIRD outputs...

5 But we can get some WEIRD outputs…

5 But we can get some WEIRD outputs…



RRT is not optimal (cost of paths are not considered)

- This is an example of "**feasible motion planning**"

5 We solve this problem with RRT*

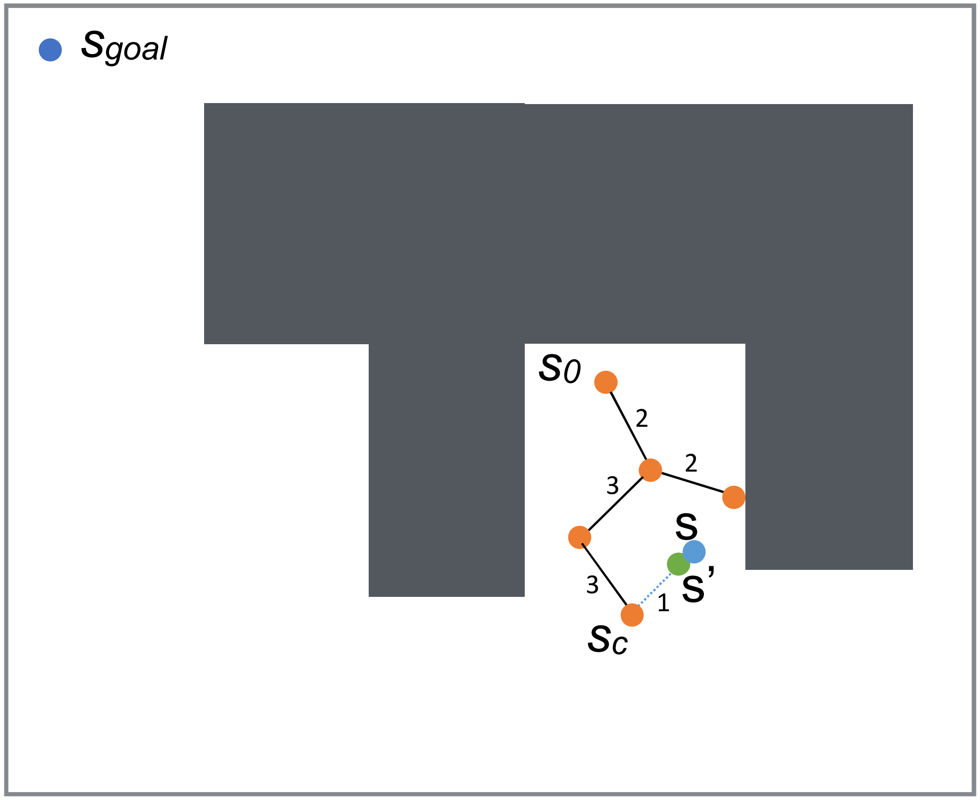
The big trick:

- incrementally "**re-wiring**" the tree to keep locally optimal paths

5 We solve this problem with RRT*

RRT* (input: s_0, s_{goal}, initial state tree T)

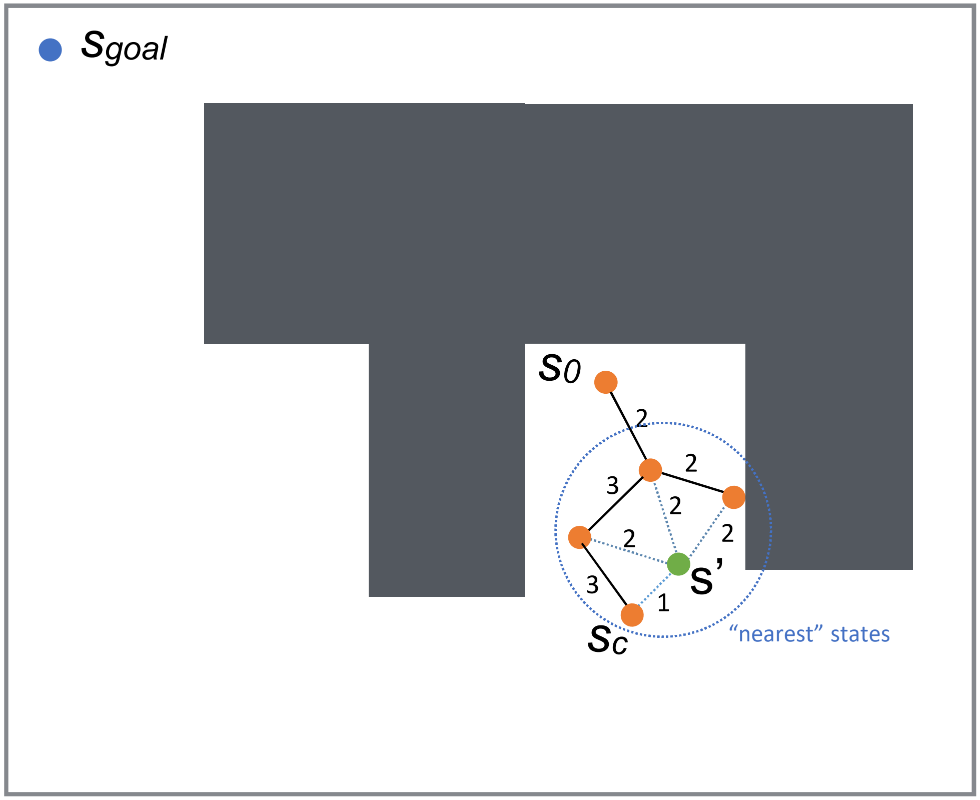
- Sample states $s \in S = R^{15}$ until s is collision-free (often goal directed)
- Find closest state $s_c \in T$
- **Extend s_c toward s resulting in state s'**
- Find all $s_{near} \subseteq T$ within a distance d to s'
- Find $s_{min} \in s_{near}$, that has the lowest *path cost* to s_0 -> s_{min} -> s'
- Add edge s_{min} -> s' to T
- Check path cost through s' to all states in $s \in s_{near}$, if any are lower than existing path cost to s, then "rewire" tree to include edge s' -> s
- Repeat until maximum iterations reached and T contains a path from s_0 to s_{goal}



5 We solve this problem with RRT*

RRT* (input: s_0, s_{goal}, initial state tree T)

- Sample states $s \in S = R^{15}$ until s is collision-free (often goal directed)
- Find closest state $s_c \in T$
- Extend s_c toward s resulting in state s'
- **Find all $s_{near} \subseteq T$ within a distance d to s'**
- Find $s_{min} \in s_{near}$, that has the lowest *path cost* to s_0 -> s_{min} -> s'
- Add edge s_{min} -> s' to T
- Check path cost through s' to all states in $s \in s_{near}$, if any are lower than existing path cost to s, then "rewire" tree to include edge s' -> s
- Repeat until maximum iterations reached and T contains a path from s_0 to s_{goal}

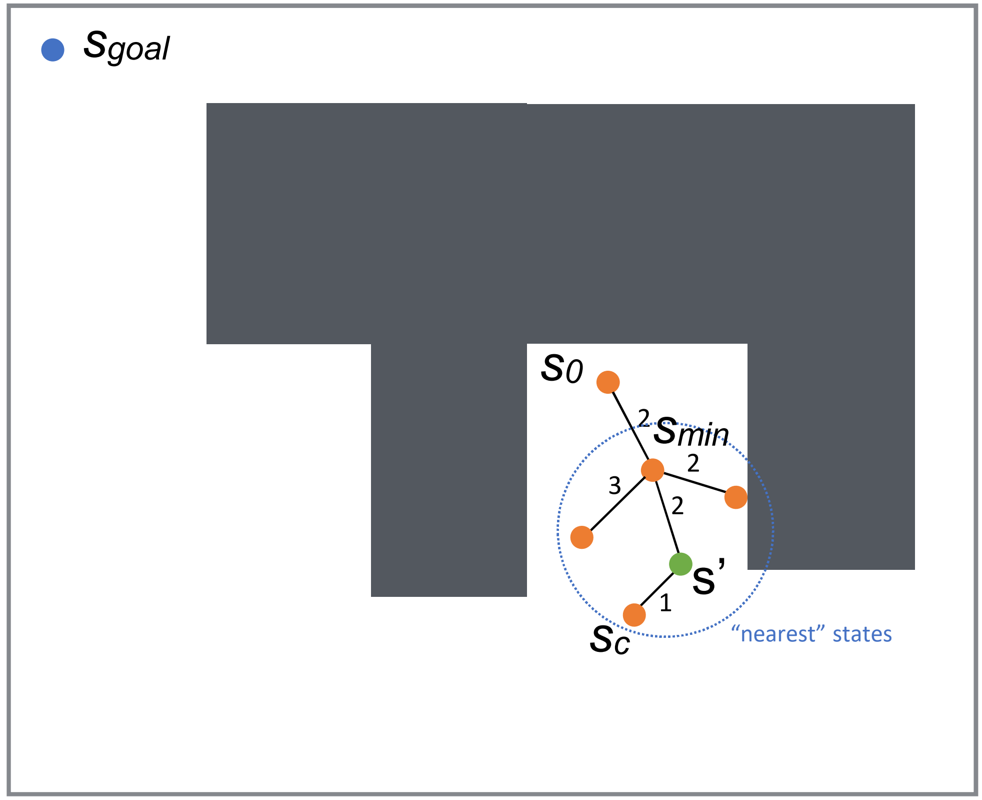


5

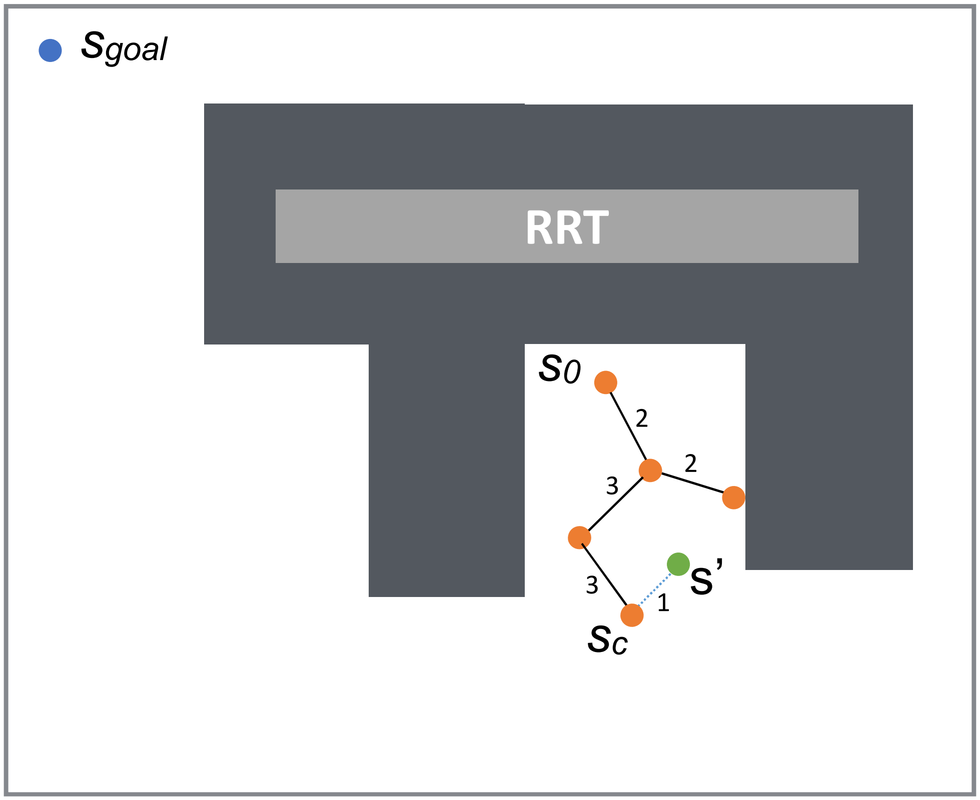
We solve this problem with RRT*

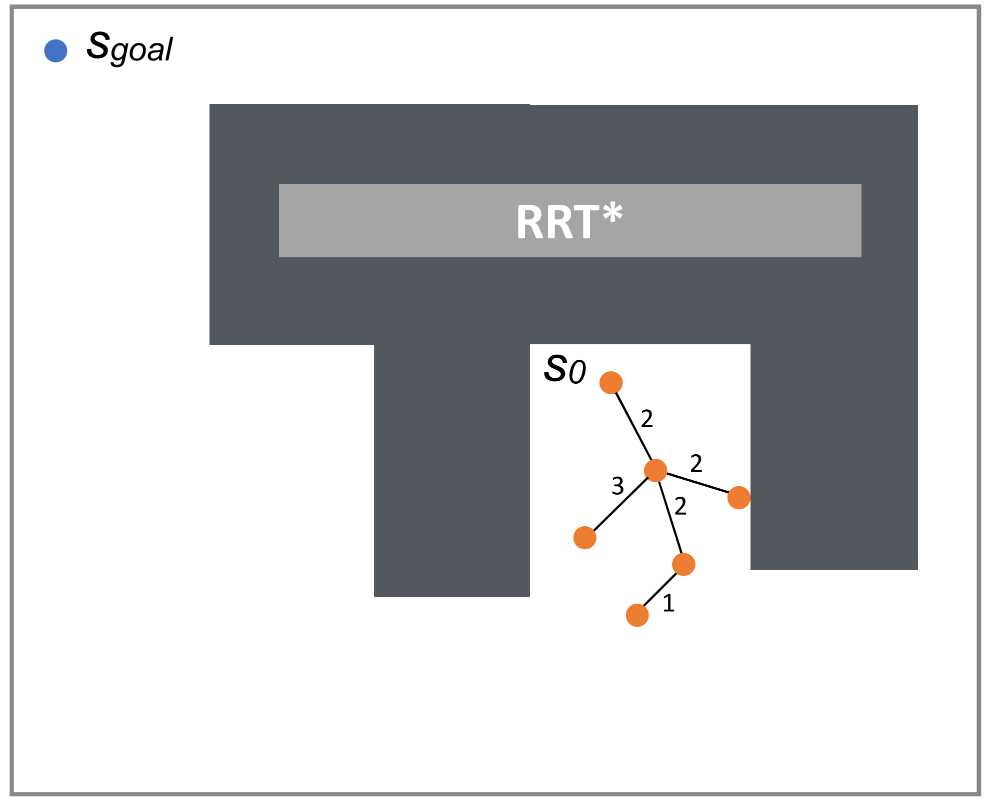
RRT* (input: s_0, s_{goal}, initial state tree T)

- Sample states $s \in S = R^{15}$ until s is collision-free (often goal directed)
- Find closest state $s_c \in T$
- Extend s_c toward s resulting in state s'
- Find all $s_{near} \subseteq T$ within a distance d to s'
- **Find $s_{min} \in s_{near}$, that has the lowest *path cost* to s_0 -> s_{min} -> s'**
- **Add edge s_{min} -> s' to T**
- **Check path cost through s' to all states in $s \in s_{near}$, if any are lower than existing path cost to s, then "rewire" tree to include edge s' -> s**
- Repeat until maximum iterations reached and T contains a path from s_0 to s_{goal}



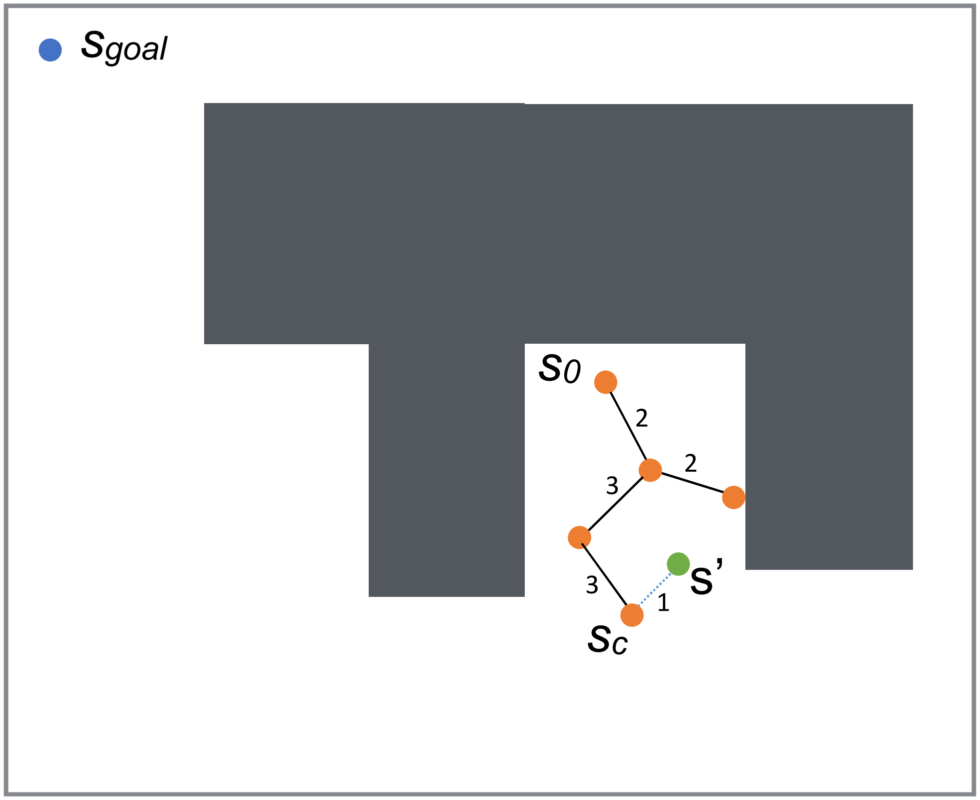
5 We solve this problem with RRT*

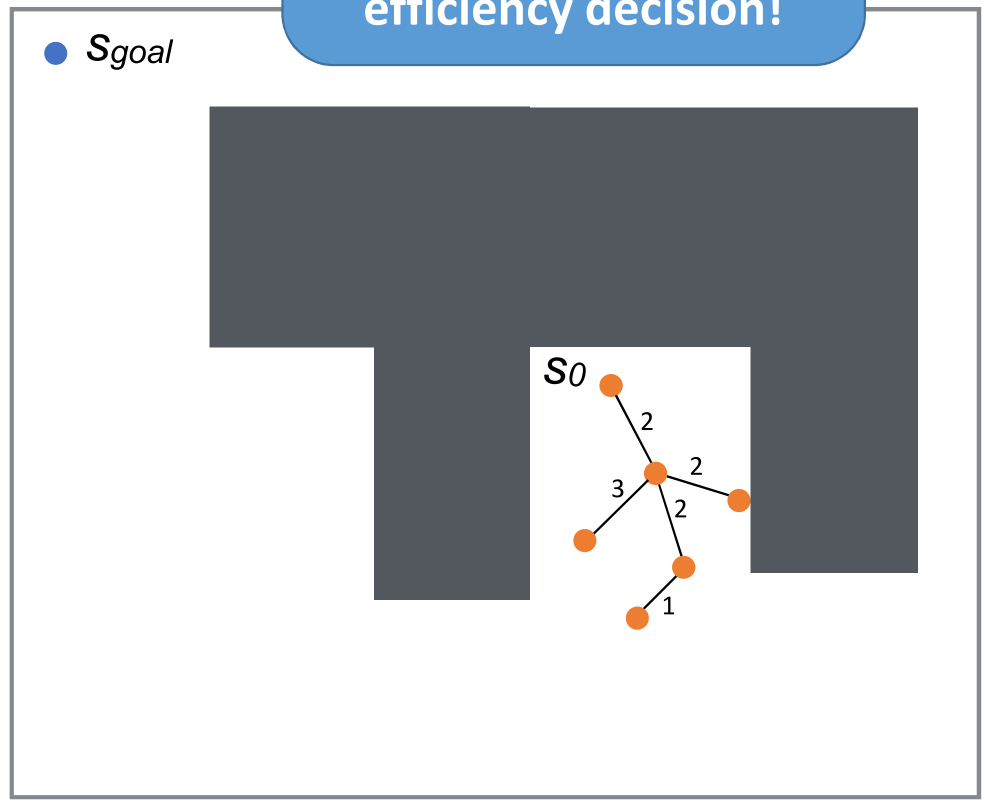




5 We solve this problem with RRT*

Nearest radius size is another sample vs. computational efficiency decision!





5 RRT*

[Karaman & Fazzoli Sampling-based Algorithms for Optimal Motion Planning]

RRT

RRT*

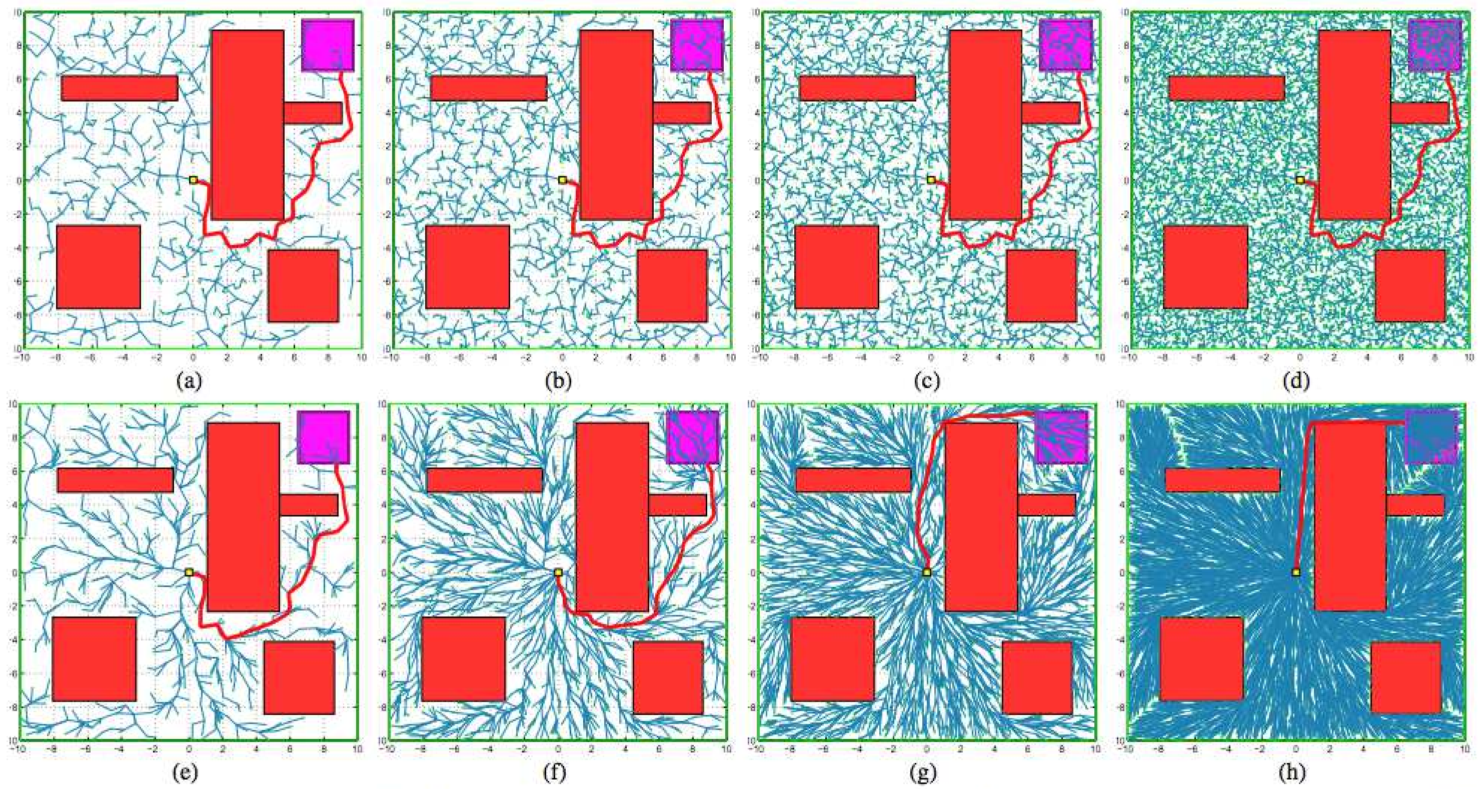


Fig. 1. A Comparison of the RRT* and RRT algorithms on a simulation example. The tree maintained by the RRT algorithm is shown in (a)-(d) in different stages, whereas that maintained by the RRT* algorithm is shown in (e)-(h). The tree snapshots (a), (e) are at 1000 iterations , (b), (f) at 2500 iterations, (c), (g) at 5000 iterations, and (d), (h) at 15,000 iterations. The goal regions are shown in magenta. The best paths that reach the target are highlighted with red.

5 So what have we learned so far?

1. Robot planning usually involves thinking about both **task and configuration spaces**
2. For many real problems, **collision checking** can be expensive
3. **RRT**: a powerful algorithm based on a very simple idea!
   - **Probabilistically complete:** If there's a solution it will find it eventually (but can still be slow for some problems)!
   - BUT RRT is not optimal (cost of paths are not considered)
     - This is an example of "**feasible motion planning**"
     - **RRT*** fixes that by incrementally rewiring the tree

5 To RRT or not to RRT that is the question!

1. Why might RRTs not be the best algorithmic choice for a robot that repeatedly does the same task?
2. How might you adapt RRT to fix this issue?

5 To RRT or not to RRT that is the question!

1. Why might RRTs not be the best algorithmic choice for a robot that repeatedly does the same task?
2. How might you adapt RRT to fix this issue?

1. RRT is a “**single-query**” algorithm – it starts from scratch each time “forgetting” all of the connections it found in previous solves

5 To RRT or not to RRT that is the question!

1. Why might RRTs not be the best algorithmic choice for a robot that repeatedly does the same task?
2. How might you adapt RRT to fix this issue?

1. RRT is a "**single-query**" algorithm – it starts from scratch each time "forgetting" all of the connections it found in previous solves
2. Instead of building a tree lets build a **reusable graph *G***

5 To RRT or not to RRT that is the question!

1. Why might RRTs not be the best algorithmic choice for a robot that repeatedly does the same task?
2. How might you adapt RRT to fix this issue?

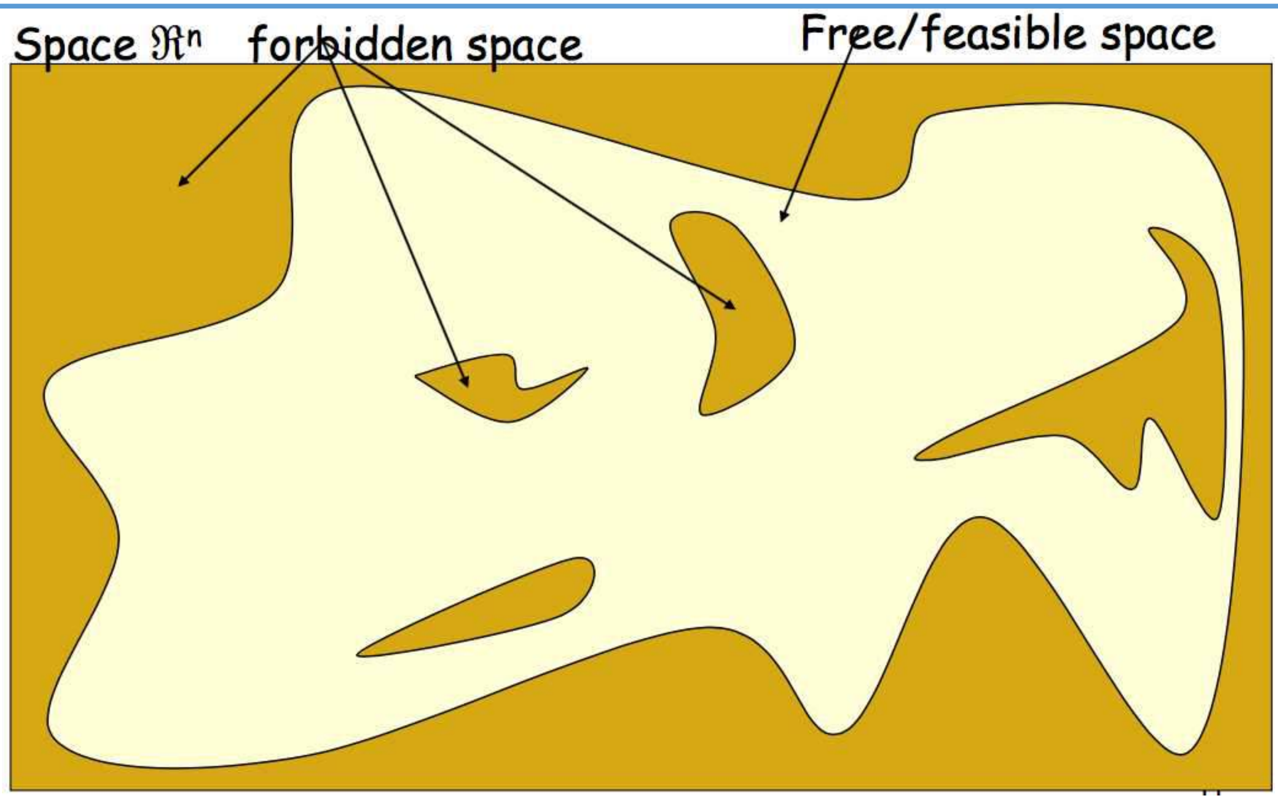
1. RRT is a "**single-query**" algorithm – it starts from scratch each time "forgetting" all of the connections it found in previous solves
2. Instead of building a tree lets build a **reusable graph *G***

This "**multi-query**" approach is called **Probabilistic Roadmaps (PRMs)**

5

Probabilistic Roadmaps (PRMs) leverage an offline and an online computation phase

Step 1: Offline build a random graph ***G*** that covers the state space

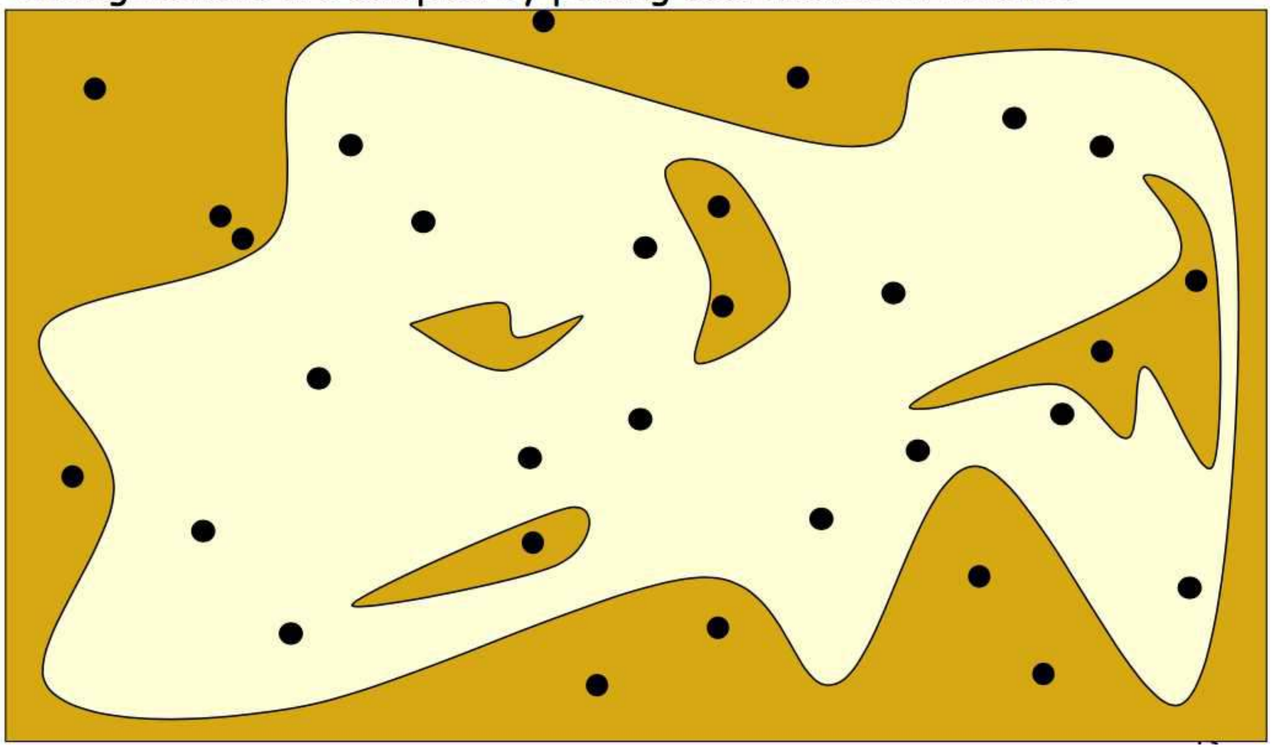


5

Probabilistic Roadmaps (PRMs) leverage an offline and an online computation phase

Step 1: Offline build a random graph ***G*** that covers the state space

Configurations are sampled by picking coordinates at random

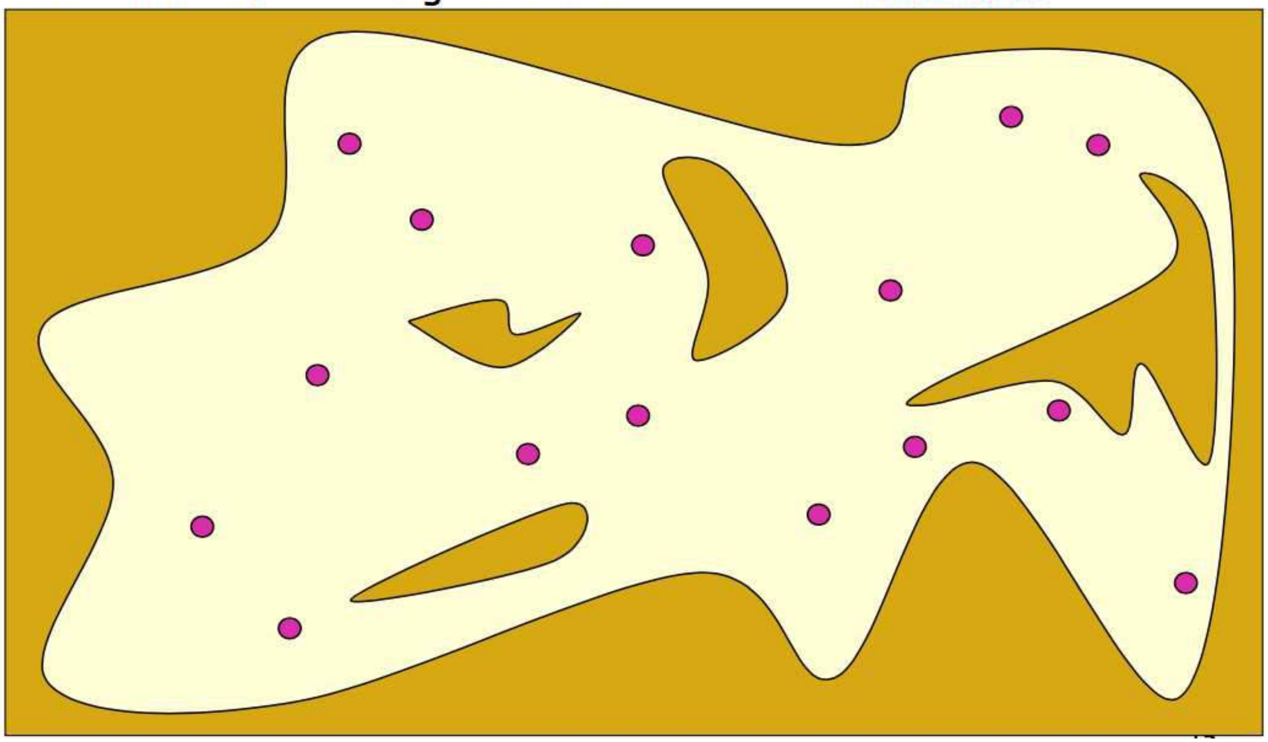


Probabilistic Roadmaps (PRMs) leverage an offline and an online computation phase

5

Step 1: Offline build a random graph ***G*** that covers the state space

The collision-free configurations are retained as milestones

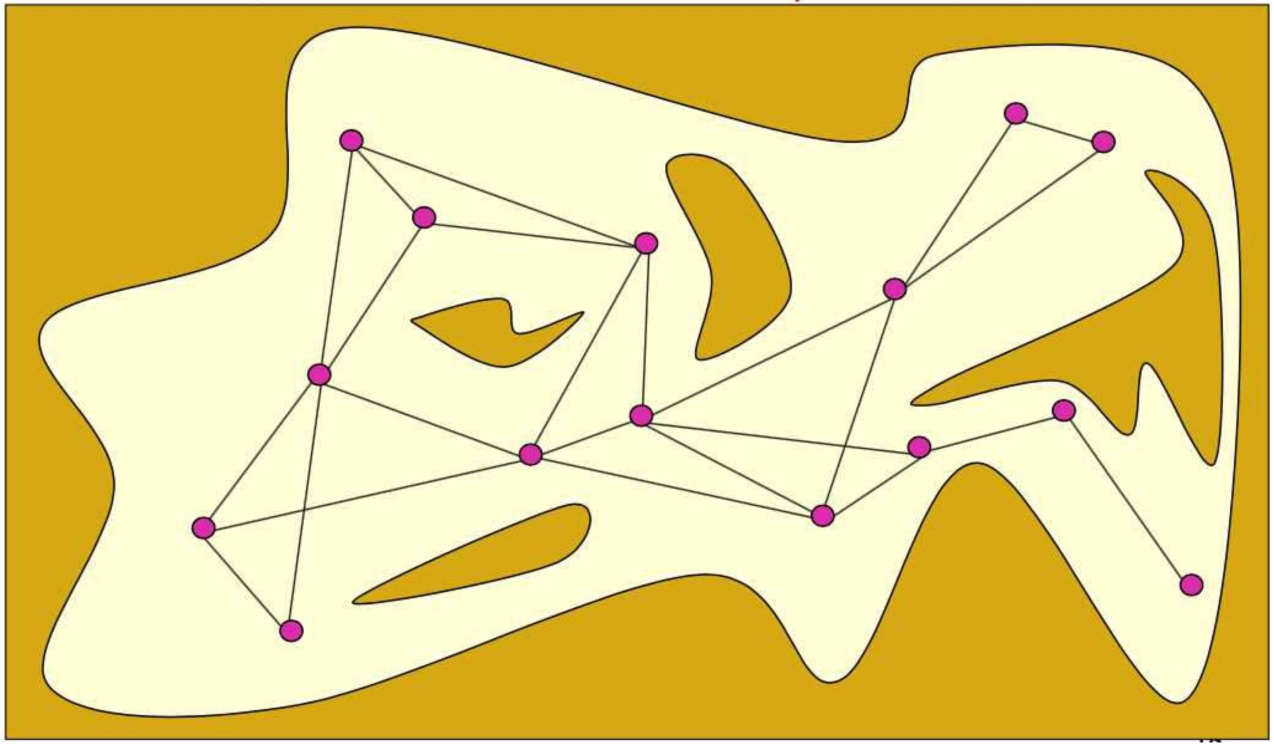


5

Probabilistic Roadmaps (PRMs) leverage an offline and an online computation phase

Step 1: Offline build a random graph ***G*** that covers the state space

The collision-free links are retained as local paths to form the PRM

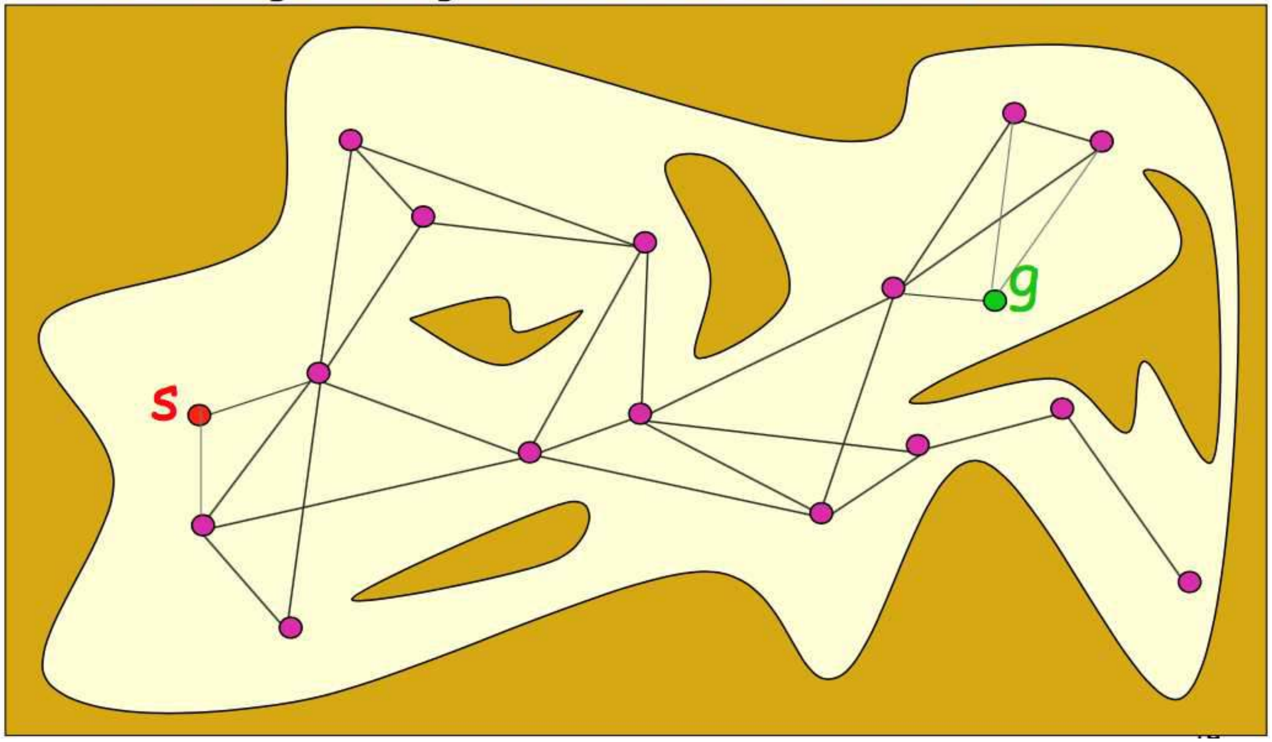


5

Probabilistic Roadmaps (PRMs) leverage an offline and an online computation phase

Step 2: Online connect the start and goal nodes and run graph search

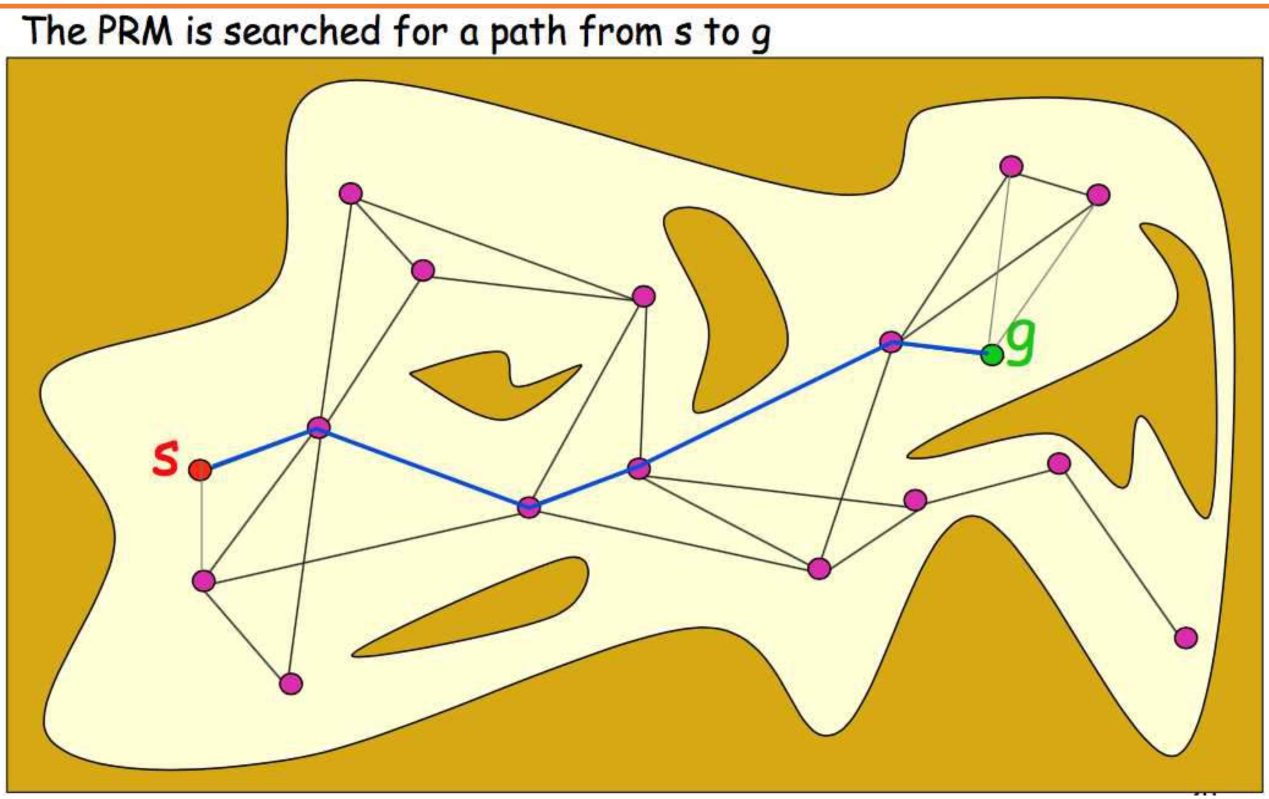
The start and goal configurations are included as milestones



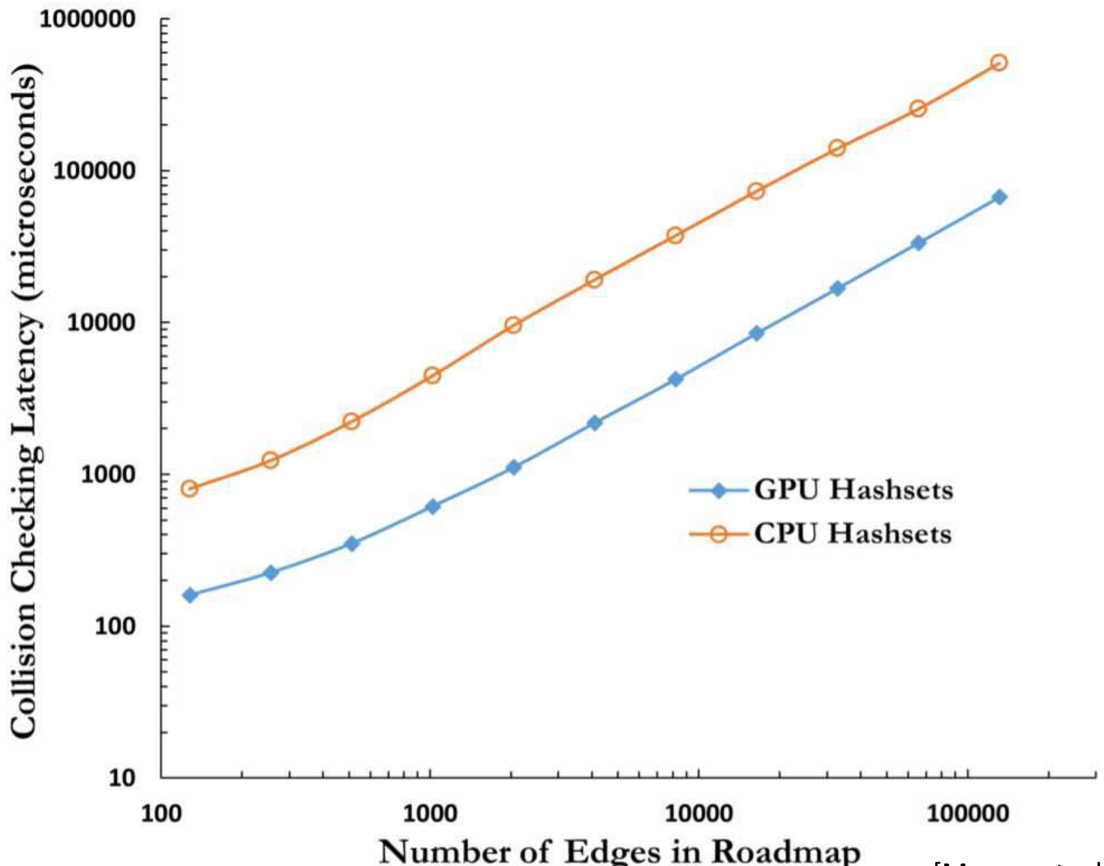
5

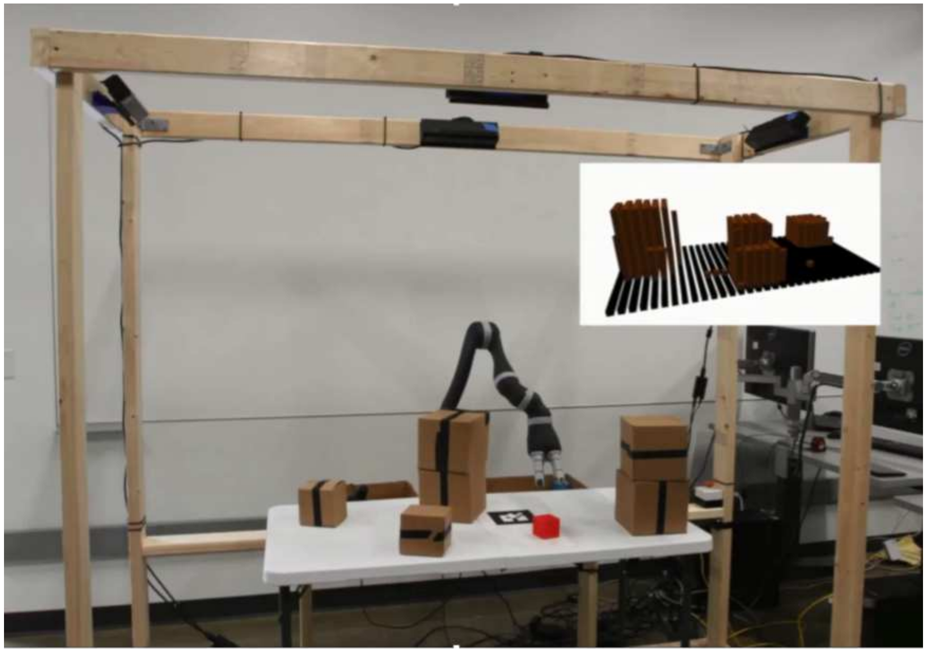
Probabilistic Roadmaps (PRMs) leverage an offline and an online computation phase

Step 2: Online connect the start and goal nodes and run graph search



5 Collision detection for each connecting path in the construction of the PRM can be very expensive

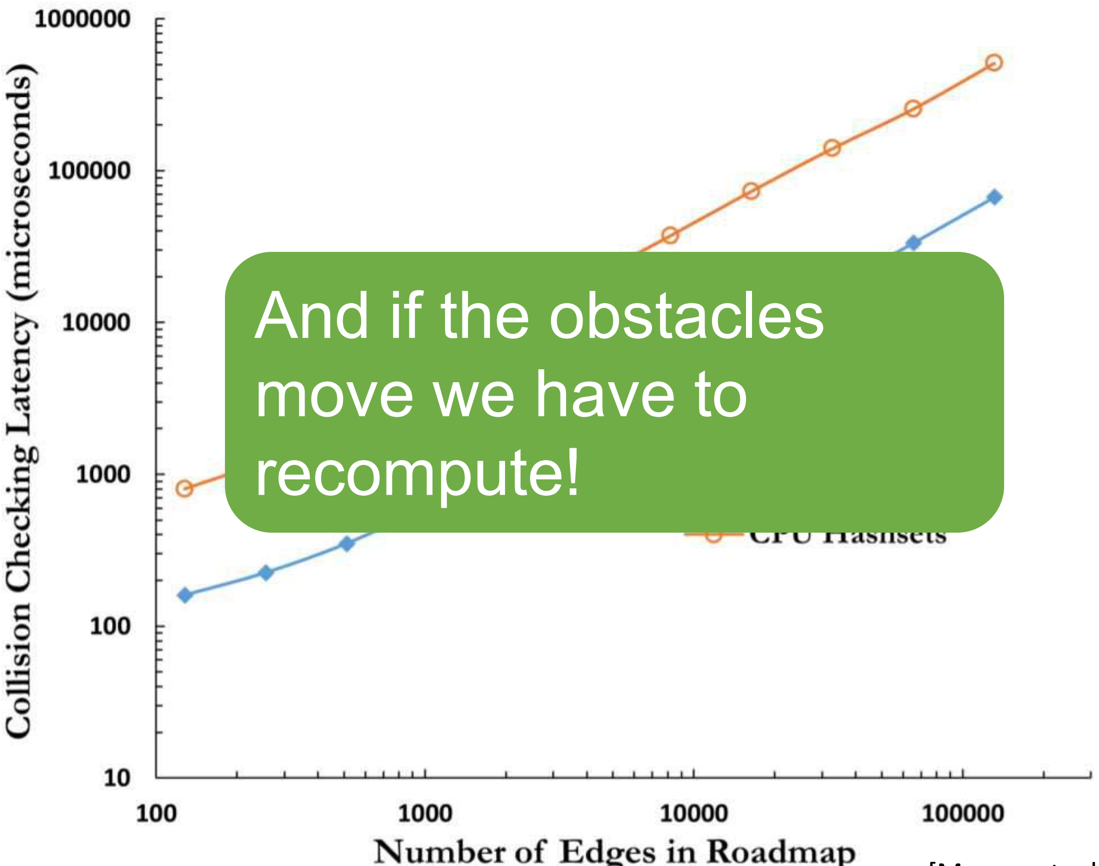




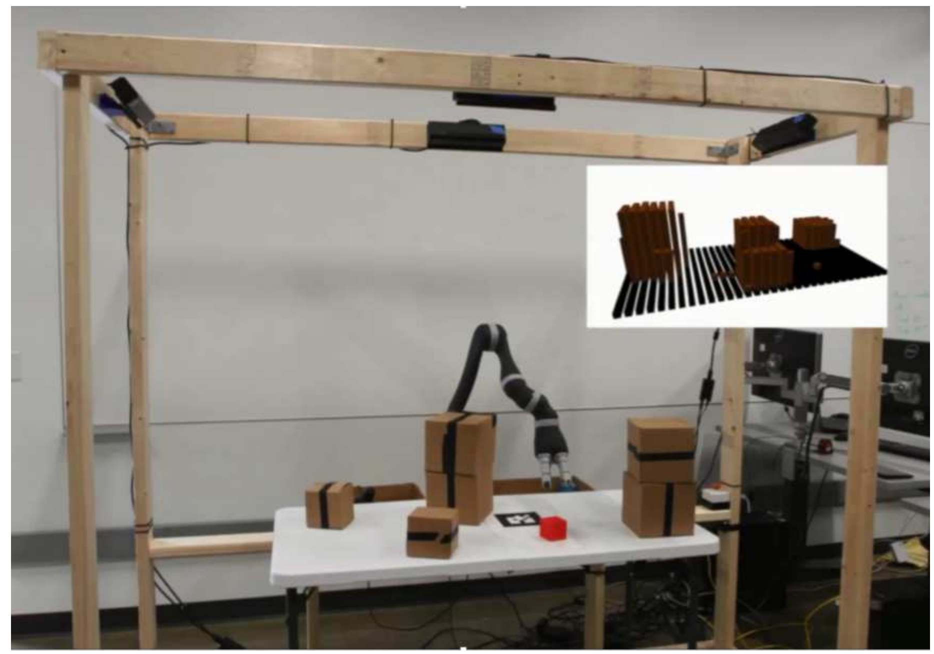
[Murray et. al. The Microarchitecture of a Real-Time Robot Motion Planning Accelerator]

5

Collision detection for each connecting path in the construction of the PRM can be very expensive

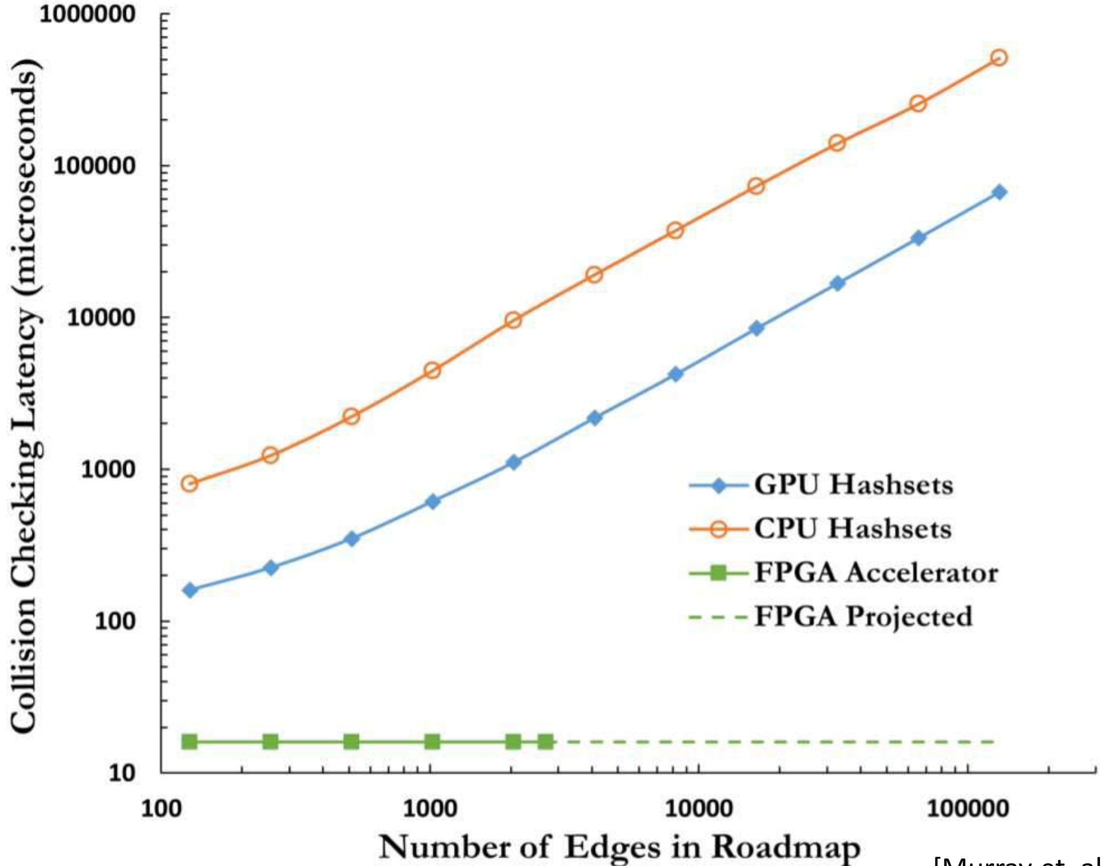


And if the obstacles move we have to recompute!



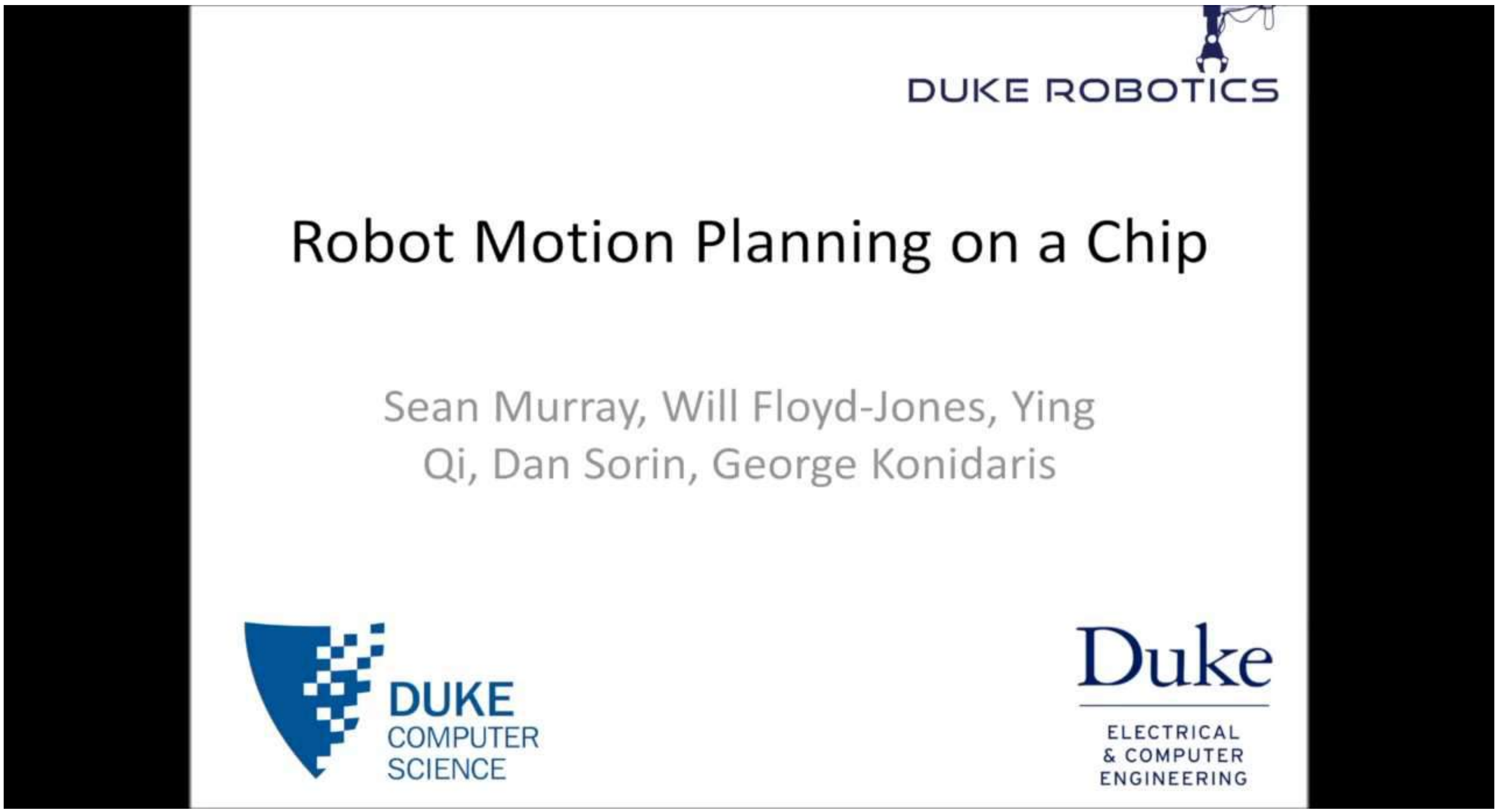
[Murray et. al. The Microarchitecture of a Real-Time Robot Motion Planning Accelerator]

Collision detection for each connecting path in the construction of the PRM can be very expensive



[Murray et. al. The Microarchitecture of a Real-Time Robot Motion Planning Accelerator]

5 Custom hardware can lead to near-instantaneous collision checking!

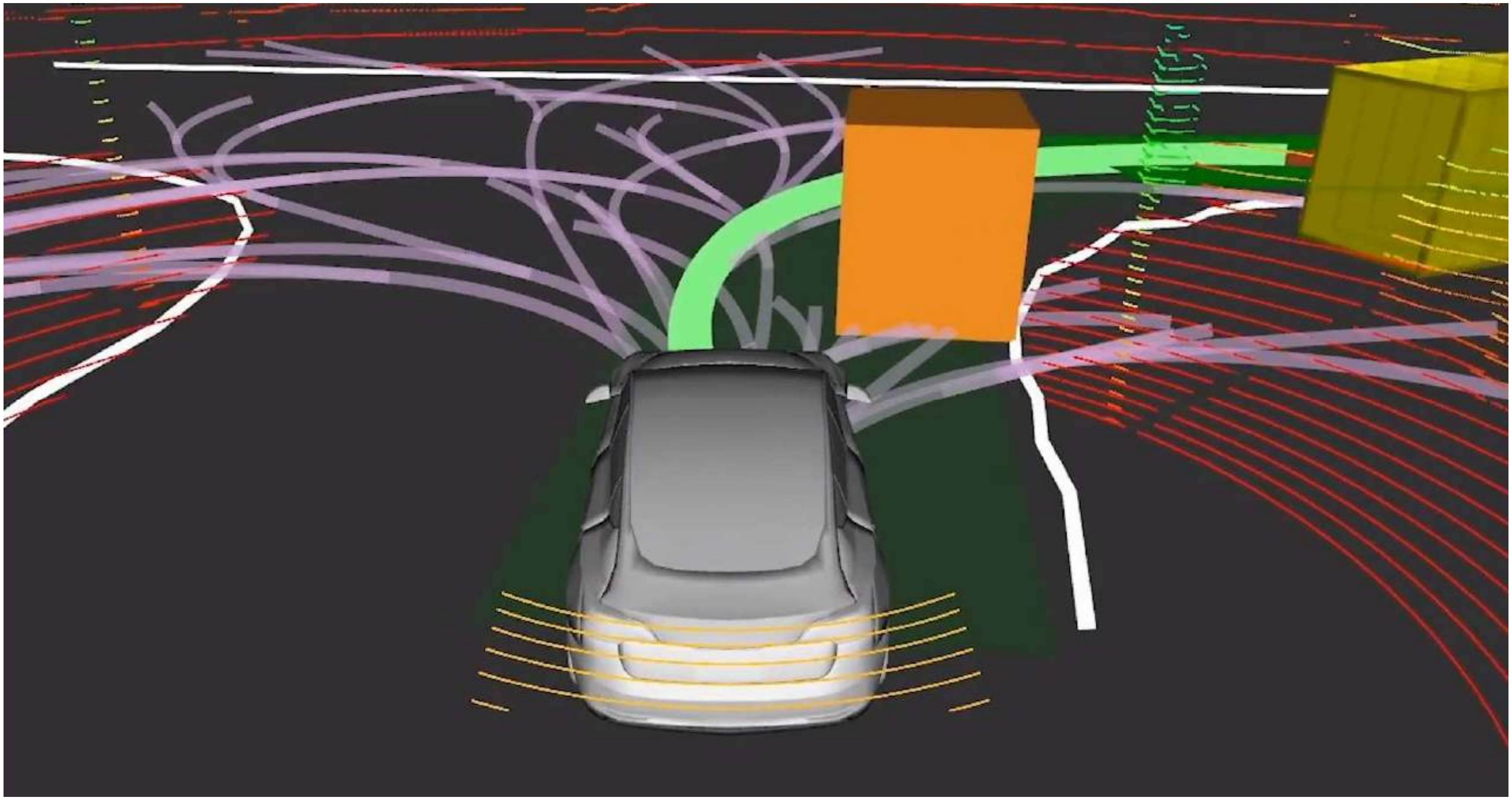


5 Custom hardware can lead to near-instantaneous collision checking!

We'll read this paper later so I'm not going to get into the details!

5 Custom hardware can lead to near-instantaneous collision checking!

Realtime Robotics



5

Ok so why can’t robots use these awesome kinematic planning algorithms all the time and be better at life?!?

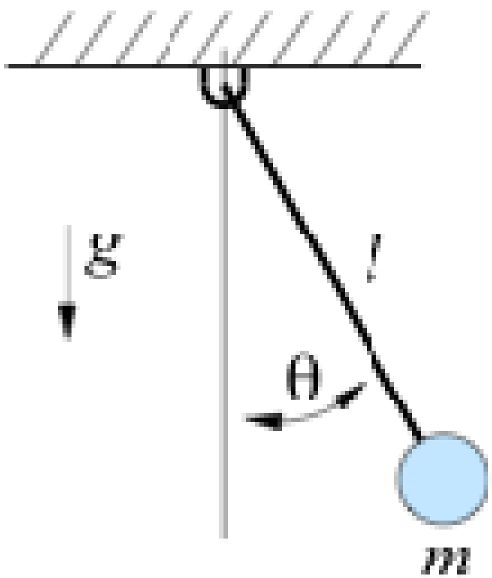
5 Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?

Dynamics (aka Physics)

5

Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?

The Simplest "Robot"

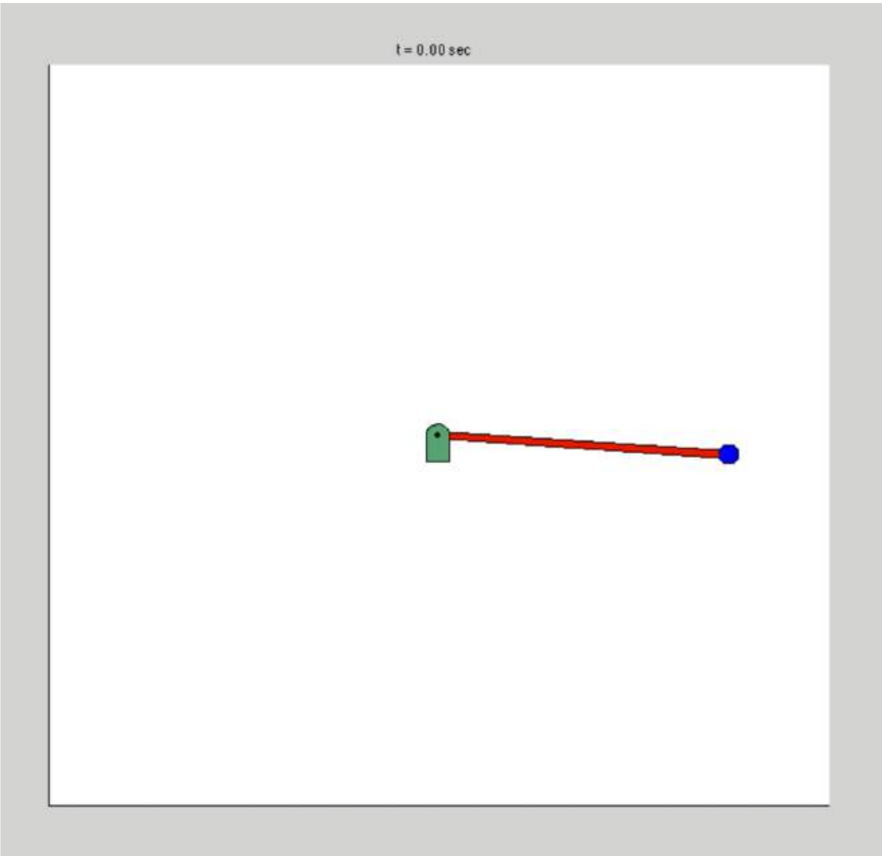


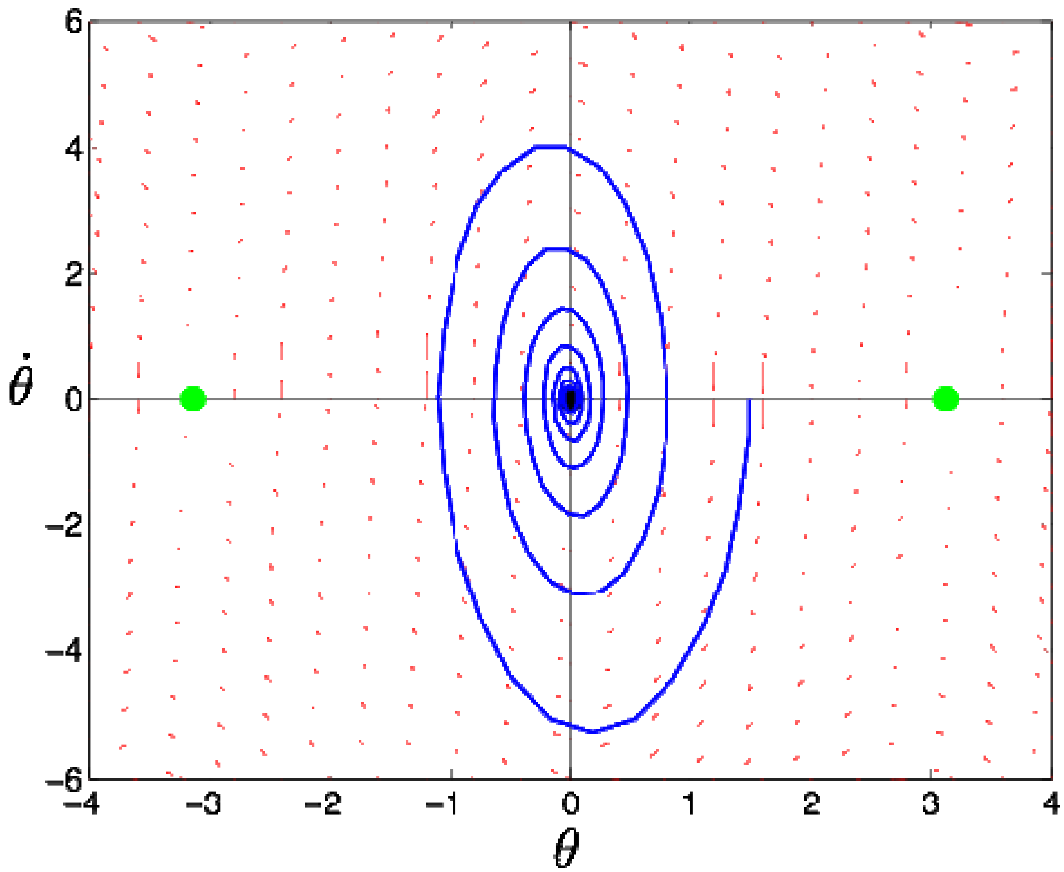
Dynamics (aka Physics)

- States: $s = \{\theta, \dot{\theta}\}$ aka angle and angular **velocity**
- Actions: $a = \tau$ aka torque at joint
- Transitions: $s' = f(s, a)$ aka physics

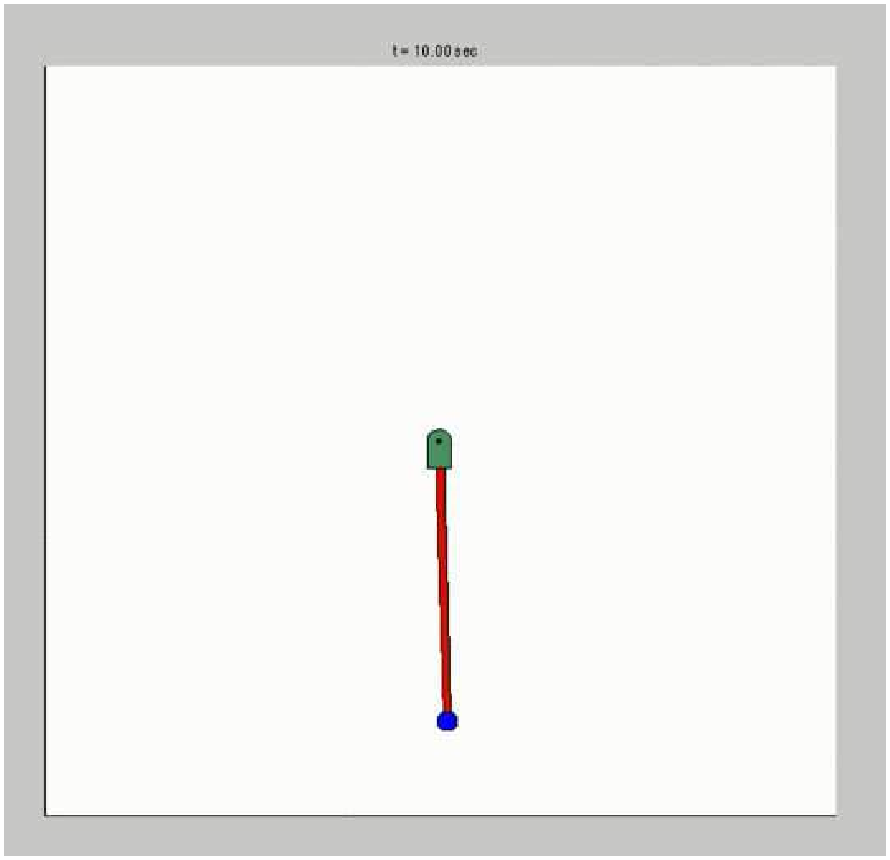
5

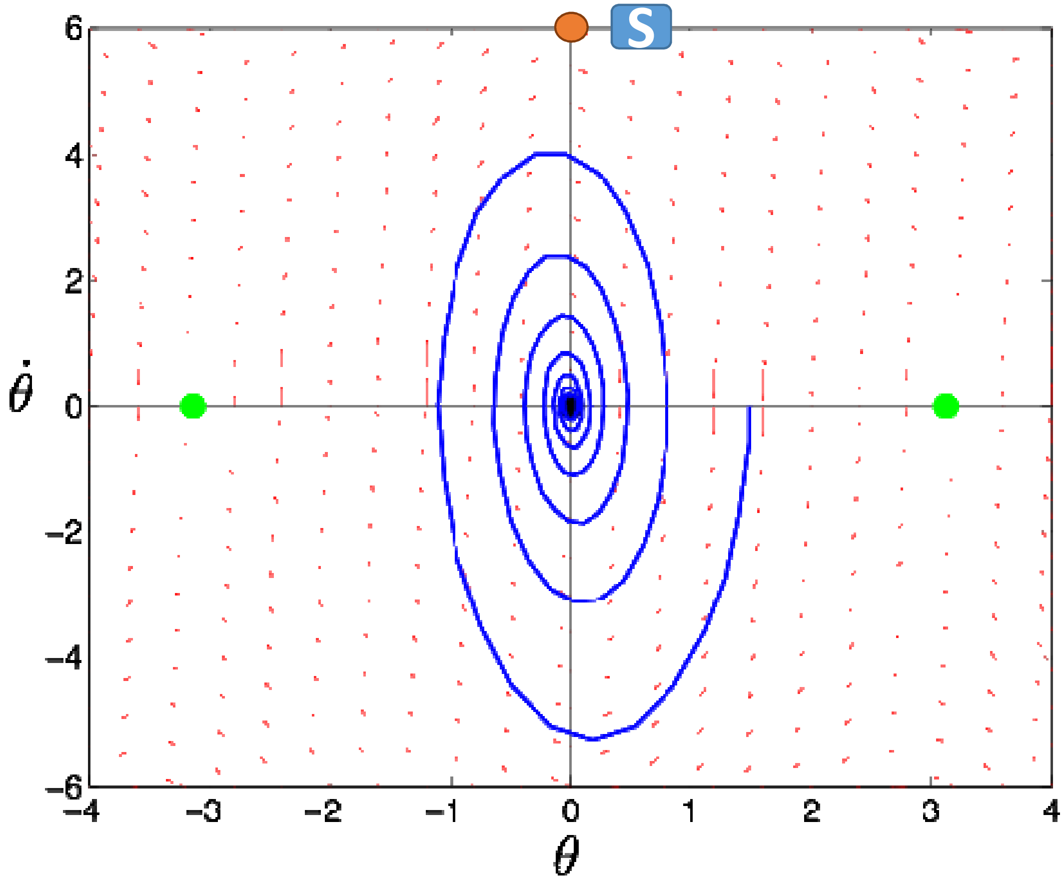
Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?



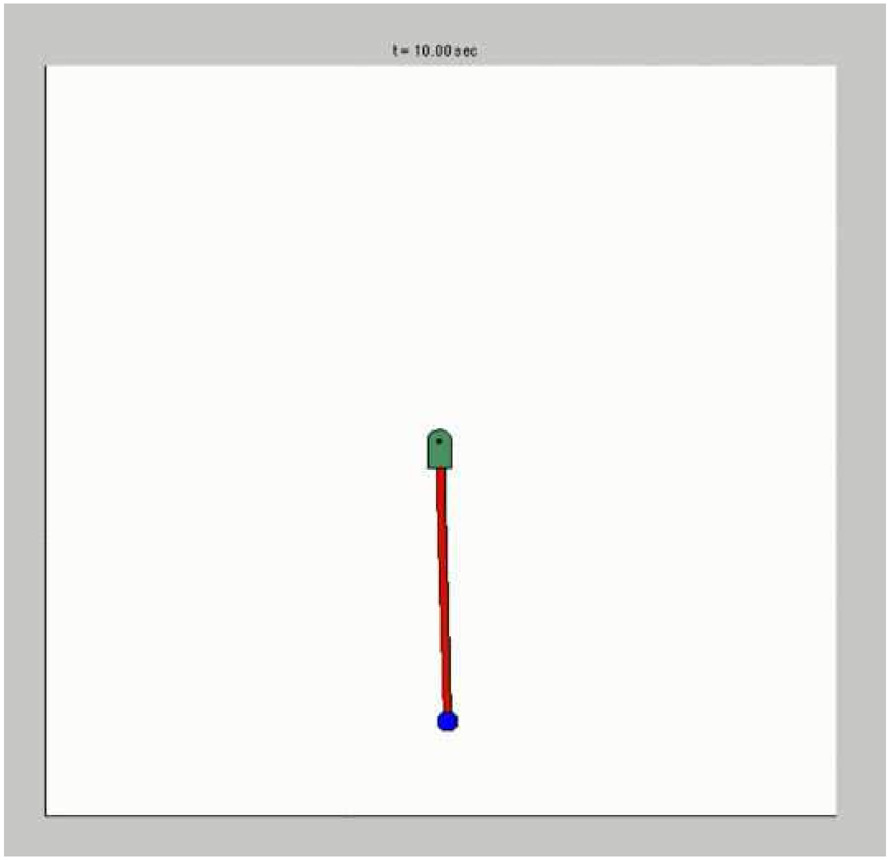


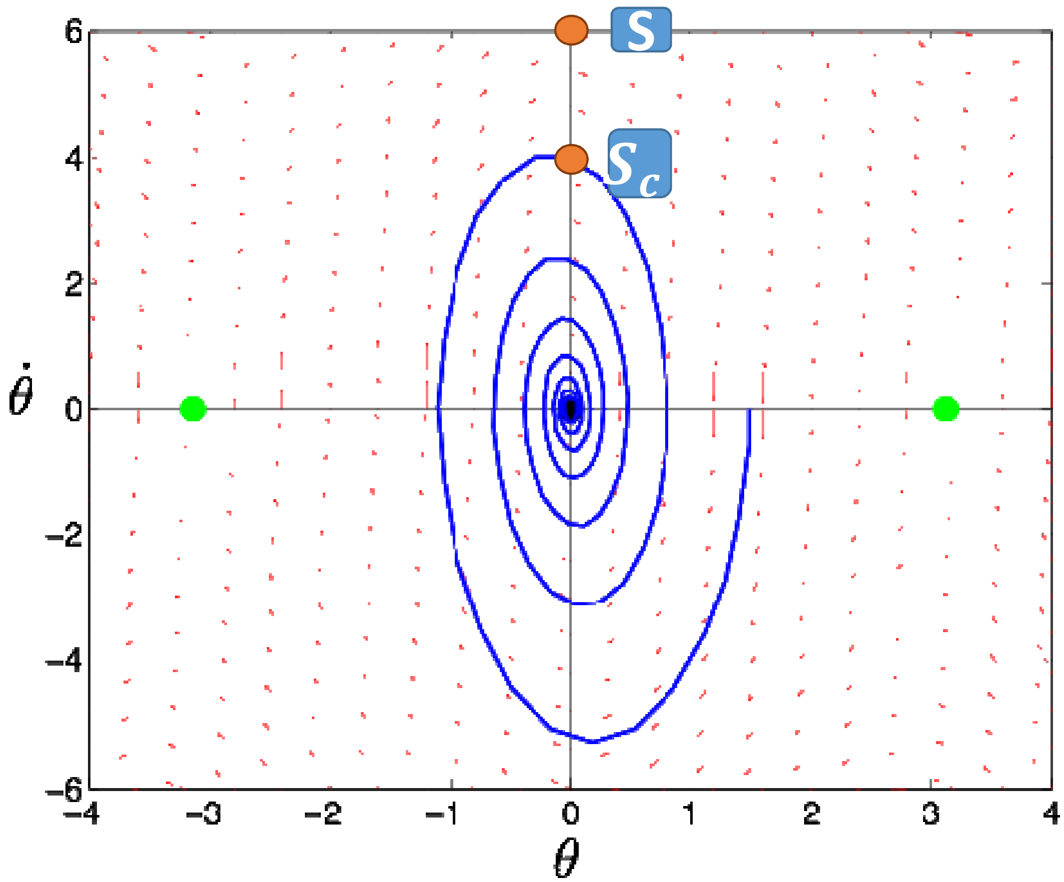
5 Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?





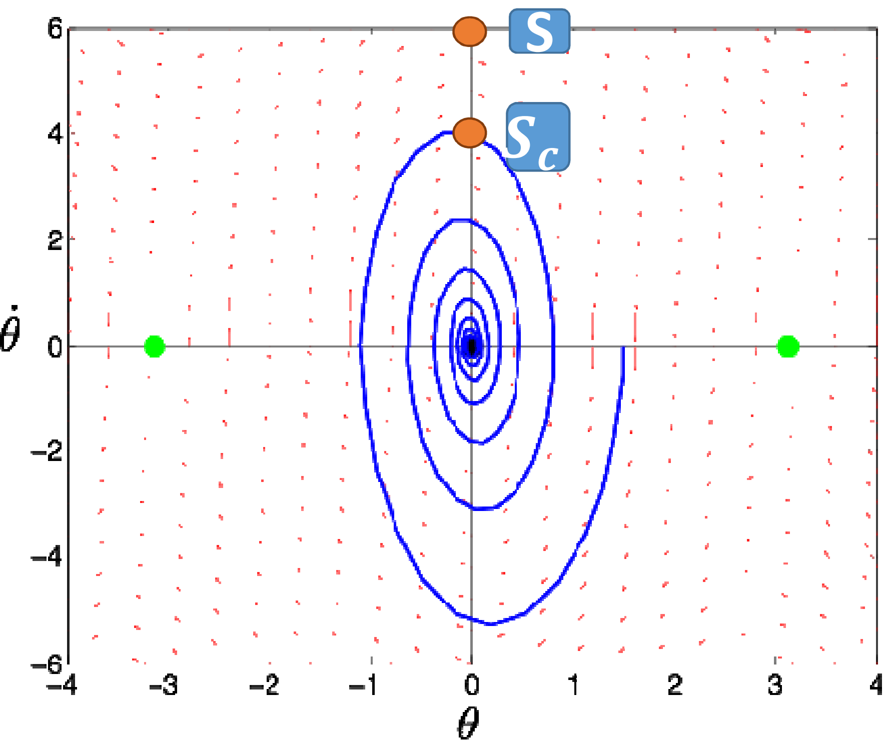
5 Ok so why can’t robots use these awesome kinematic planning algorithms all the time and be better at life?!?





5

Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?



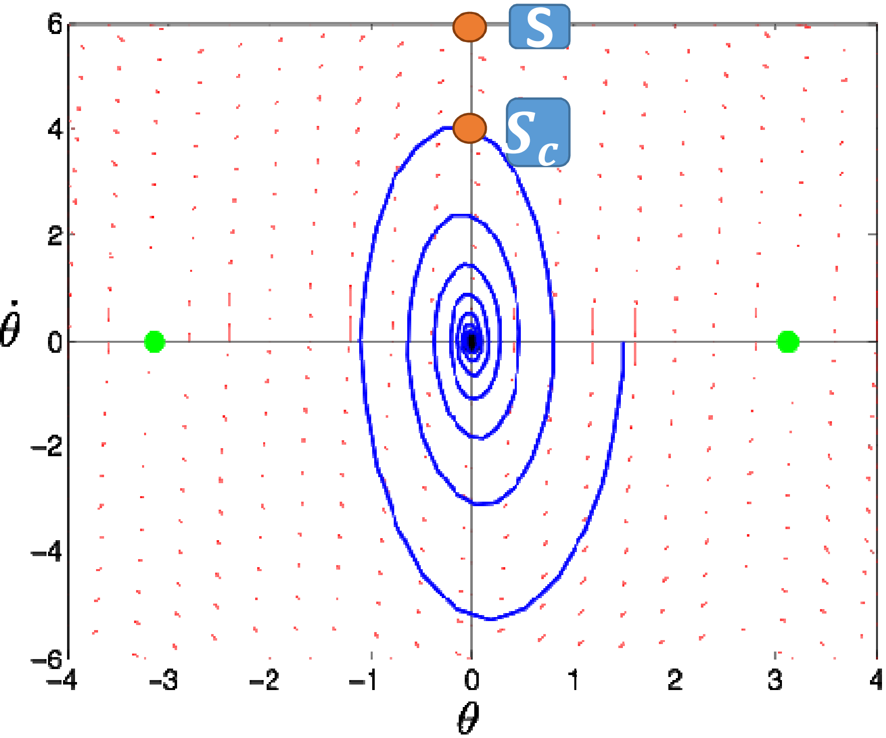
Challenges for Dynamic RRTs

The "extend" operation is complex!

- We need to solve a **boundary value problem** (find a path from sc to s such that it follows the dynamics)
- Basically a "mini" planning problems

5

Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?



Challenges for Dynamic RRTs

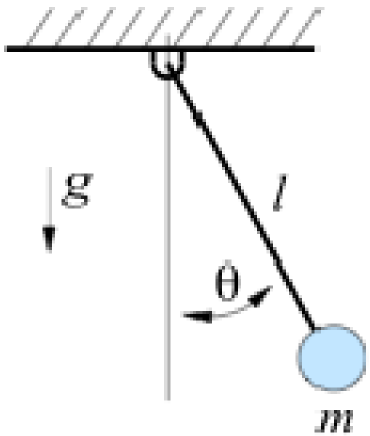
The "extend" operation is complex!

- We need to solve a **boundary value problem** (find a path from sc to s such that it follows the dynamics)
- R … ems

Q: Why don't we just try a discretization of possible actions instead of solving a boundary value problem?

5

Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?

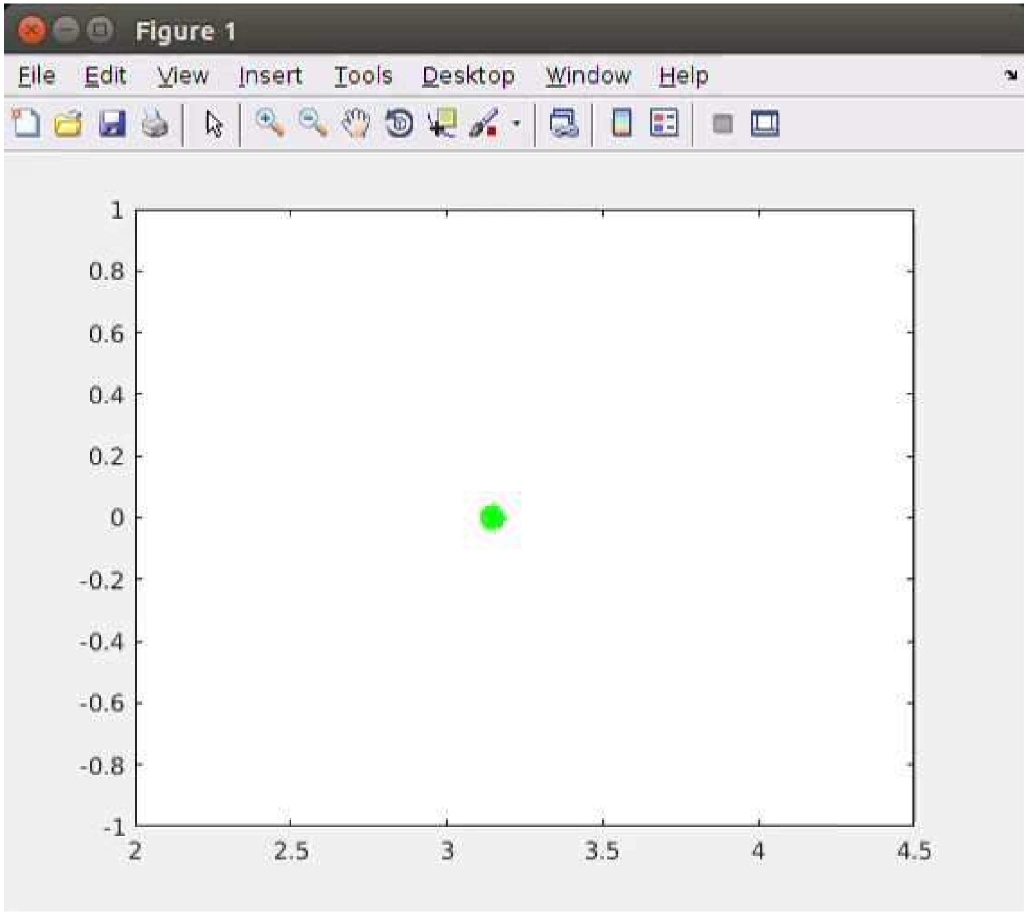


- States: $s = \{\theta, \dot{\theta}\}$ aka angle and angular velocity
- Actions: $a = \tau$ aka torque at joint
- Transitions: $s' = f(s, a)$ aka physics

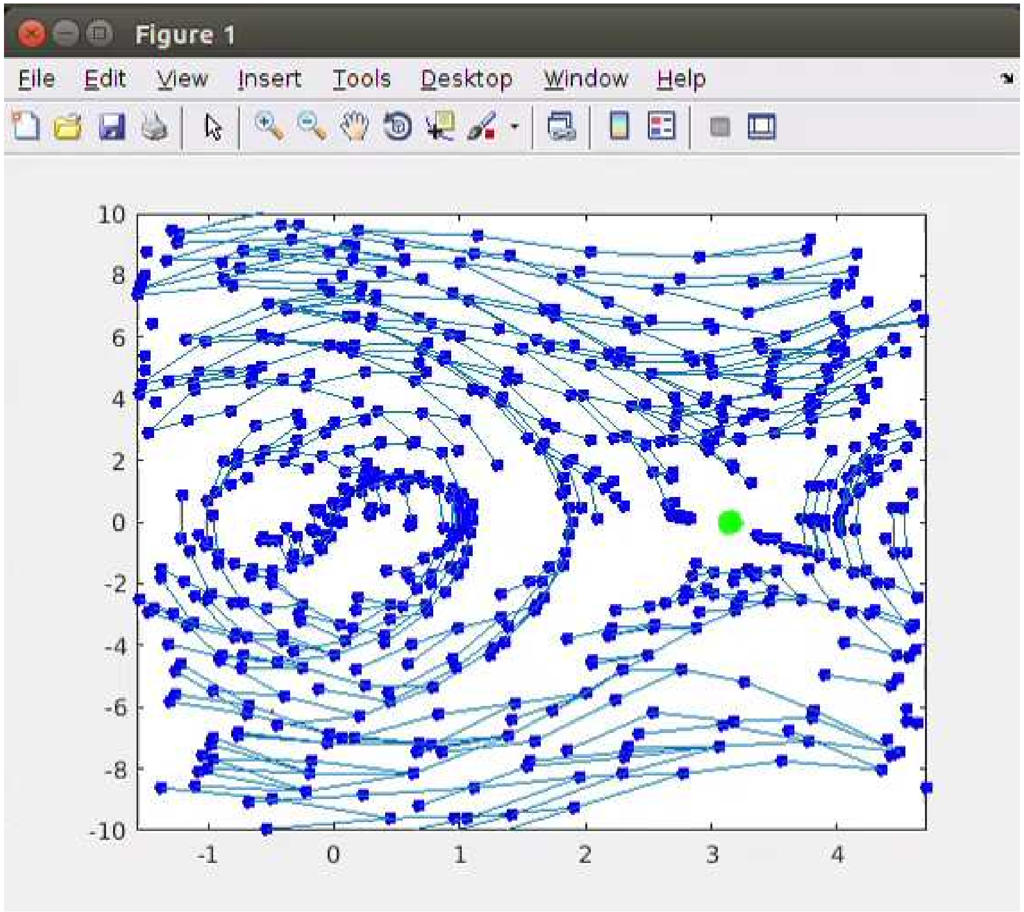
Task: **start** from the **stable downward equilibrium (0,0)** and **swing up** to the **unstable upward equilibrium (π,0)**

5

Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?

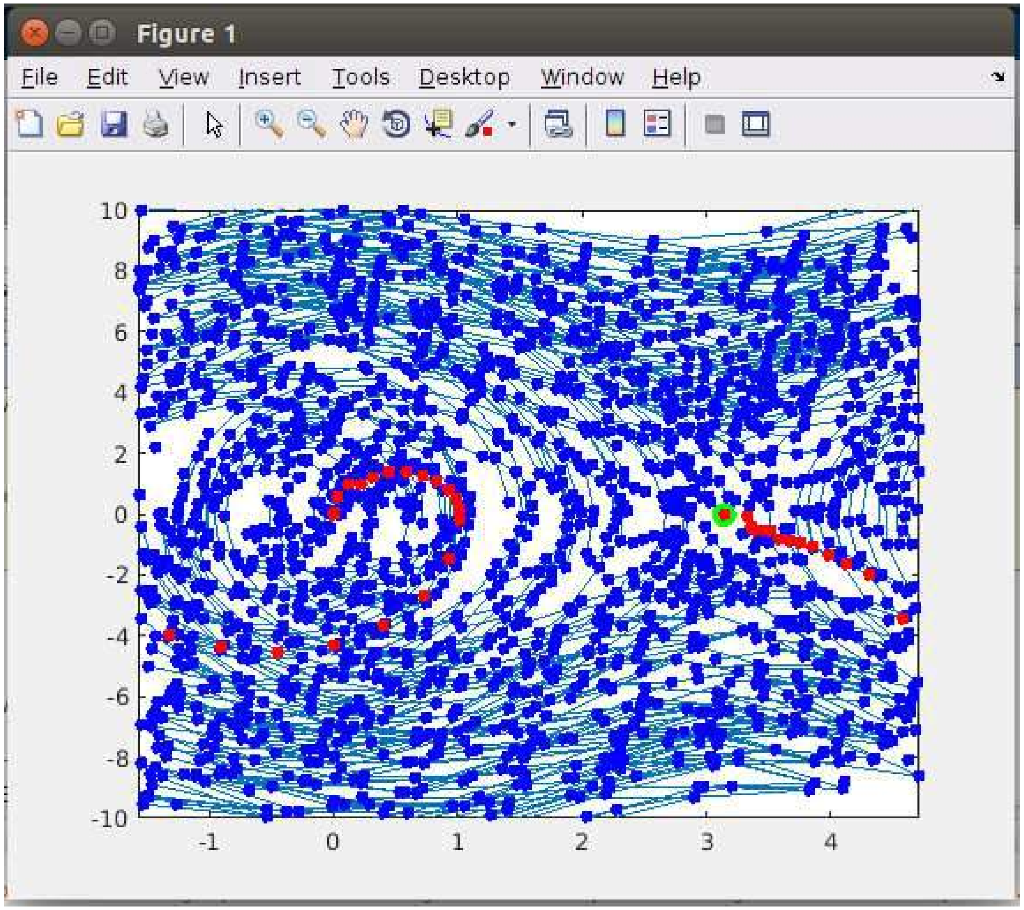


5 Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?



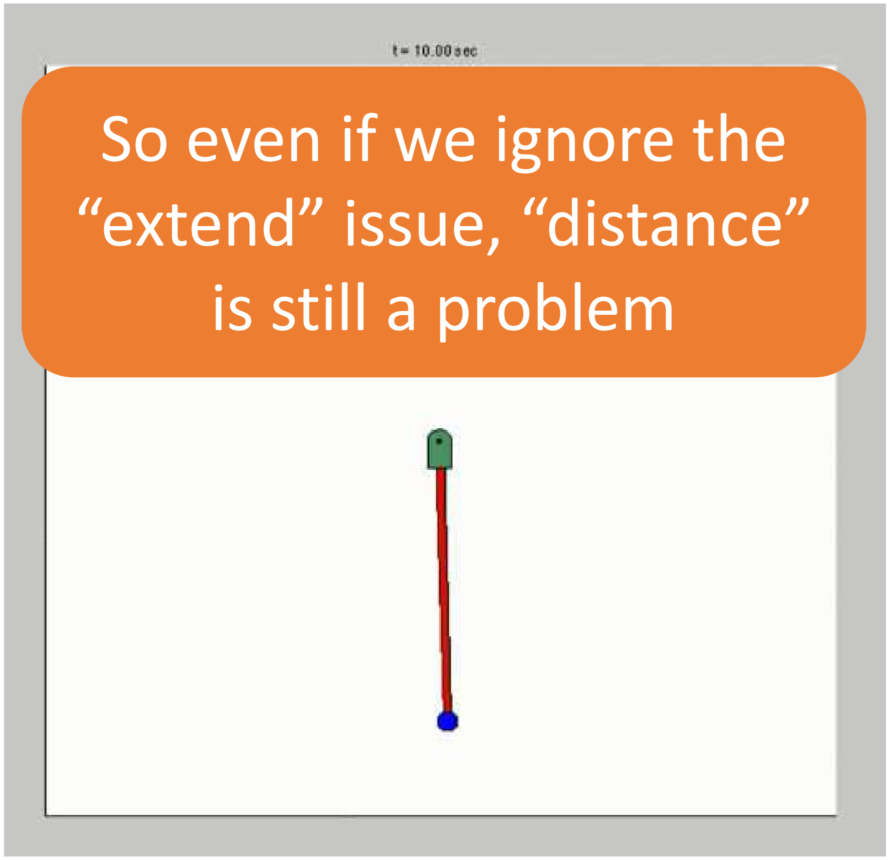
5

Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?

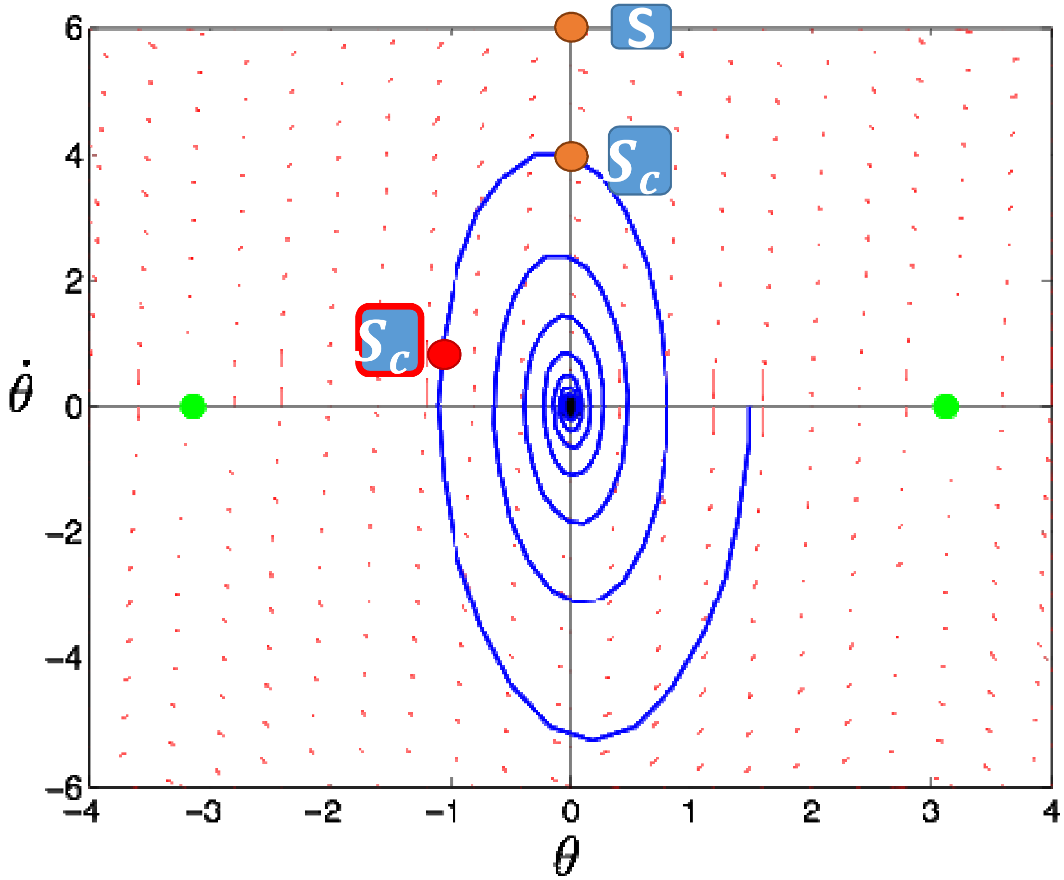


5

Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?

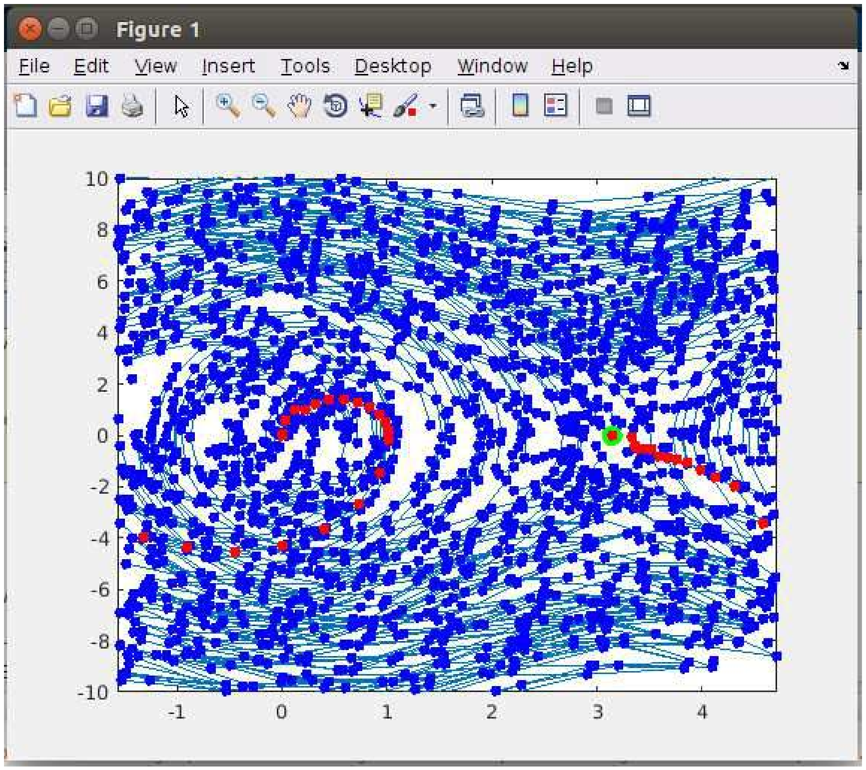


So even if we ignore the "extend" issue, "distance" is still a problem



5

Ok so why can't robots use these awesome kinematic planning algorithms all the time and be better at life?!?



Challenges for Dynamic RRTs

The "extend" operation is complex!

- We need to solve a **boundary value problem** (find a path from sc to s such that it follows the dynamics)
- Basically a "mini" planning problems

What is the "closest state in the tree"

- The "**distance**" between states of dynamical systems is **not well-defined**

5 So what do we do?

5 So what do we do?

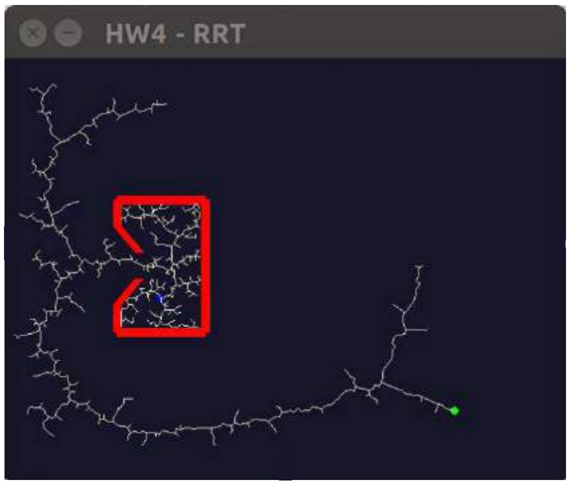
Give up and make the computer solve it for us?

5 So what do we do?

#Learning
#EfficientUseOfHumans

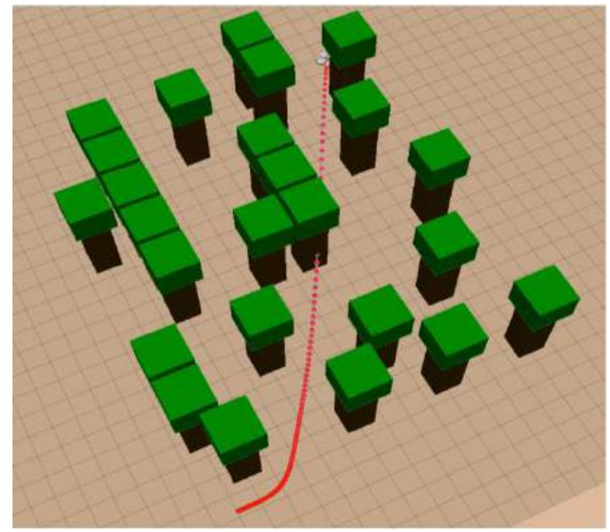
Give up and make the computer solve it for us?

5 Planning in Configuration Space



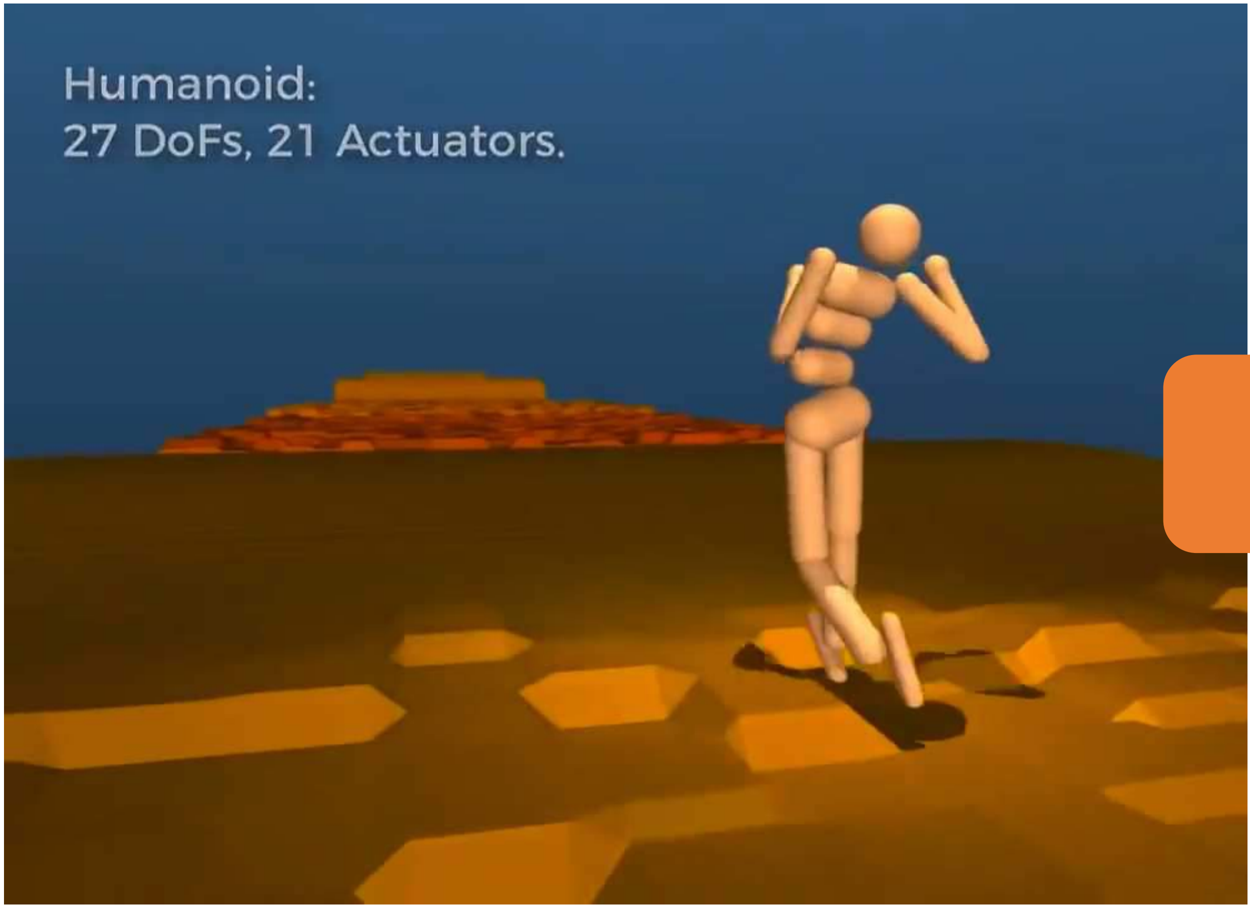
Random Search

Machine Learning



Local Search

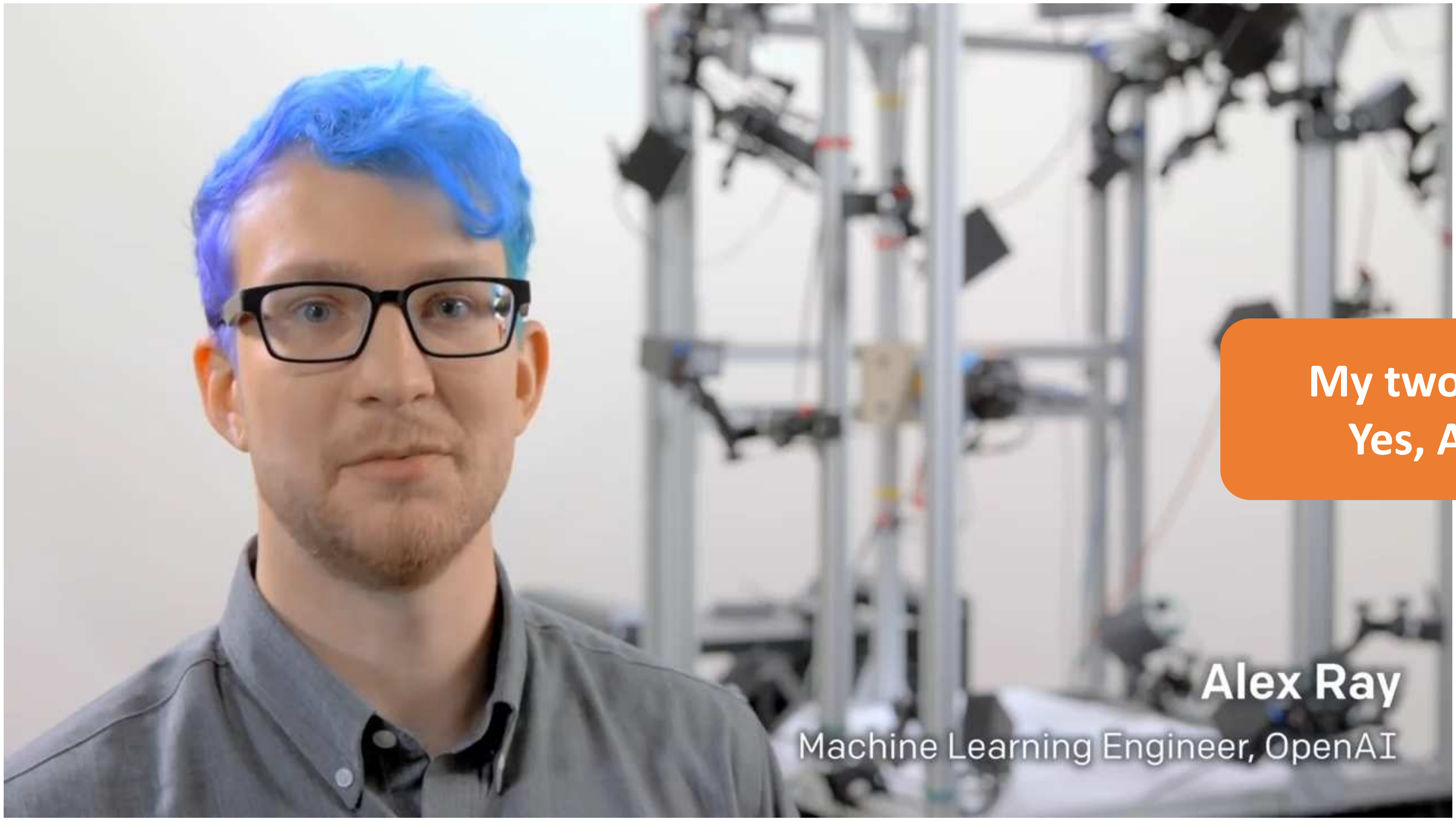
5 Guest Lecture in two weeks: Can I make the computer learn all of this for me automatically?



My two cents:
Yes, And…

5

Guest Lecture in two weeks: Can I make the computer learn all of this for me automatically?



My two cents:
Yes, And...

5 Guest Lecture in two weeks: Can I make the computer learn all of this for me automatically?

My two cents:
Yes, And...

5 So what else can we do?

5

So what else can we do?

Lots of math!

5 So what else can we do?

Lots of math!

5 So what else can we do?

Its actually not that bad and the math isn't actually that scary I promise!

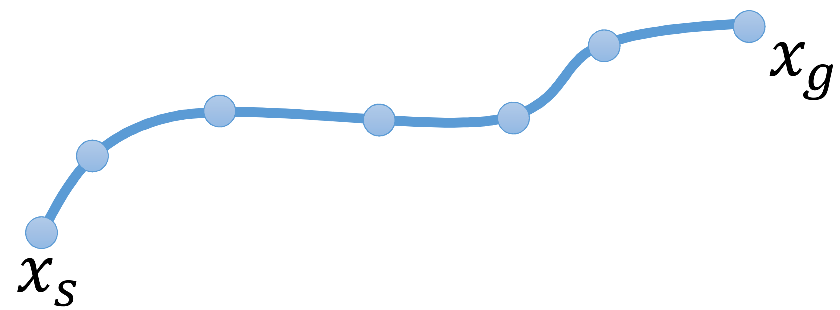
5 Optimization

We can write the planning problem down as an optimization problem!

$$\underset{s_0, a_0, \ldots, s_N, a_N}{\text{minimize}} \sum_{k=0}^{N} c(s_k, a_k)$$

$$\text{subject to} \quad s_{k+1} = f(s_k, a_k)$$

$$s_N = s_{\text{goal}}$$



5 Optimization

We can write the planning problem down as an optimization problem!

$$\underset{s_0, a_0, \ldots, s_N, a_N}{\text{minimize}} \sum_{k=0}^{N} c(s_k, a_k)$$

Minimize a cost in each state (e.g., energy used)

$$\text{subject to} \quad s_{k+1} = f(s_k, a_k)$$

Obey physics

$$s_N = s_{\text{goal}}$$

Get to the goal

5 Optimization

We can use Bellman updates to solve this:

- We can start at the goal state and then work backwards computing the lowest cost actions to get to all states all the way back to the start state

$$\underset{s_0, a_0, \dots, s_N, a_N}{\text{minimize}} \sum_{k=0}^{N} c(s_k, a_k)$$

$$\text{subject to} \quad s_{k+1} = f(s_k, a_k)$$

$$s_N = s_{\text{goal}}$$

$$V_N(s_N) = c(s_N, a_N)$$

$$V_{N-1}(s) = \min_a c(s_{N-1}, a_{N-1}) + V_N\big(f(s_{N-1}, a_{N-1})\big)$$

This leads to the classic ***Value Iteration*** algorithm

5 Optimization

We can use Bellman updates to solve this:

- We can start at the goal state and then work backwards computing the lowest cost actions to get to all states all the way back to the start state

$$\underset{s_0, a_0, \ldots, s_N, a_N}{\text{minimize}} \sum_{k=0}^{N} c(s_k, a_k)$$

$$\text{subject to} \quad s_{k+1} = f(s_k, a_k)$$

$$s_N = s_{\text{goal}}$$

$$V_N(s_N) = c(s_N, a_N)$$

$$V_{k+1}(s) = \min_a c(s, a) + V_k\big(f(s, a)\big)$$

This leads to the classic ***Value Iteration*** algorithm

5 Optimization

We can use Bellman updates to solve this

- We can start at the goal state and then work backwards computing the lowest cost actions to get to all states all the way back to the start state

$$\underset{s_0, a_0, \ldots, s_N, a_N}{\text{minimize}} \sum_{k=0}^{N} c(s_k, a_k)$$

$$\text{subject to} \quad s_{k+1} = f(s_k, a_k)$$

$$s_N = s_{\text{goal}}$$

$$V_N(s_N) = c(s_N, a_N)$$

$$V_{k+1}(s) = \min_a c(s, a) + V_k\big(f(s, a)\big)$$

Sadly again the complexity scales with $\boldsymbol{d}^{|S| = |A|}$ and those can get **HUGE** fast! This is the **"curse of dimensionality"** again

5 Optimization

Lets lower our expectations!
#localOptima #efficientUseOfComputers

We can use Bellman updates to solve this

- We can start at the goal state and then work backwards computing the lowest cost actions to get to all states all the way back to the start state

$$\underset{s_0, a_0, \ldots, s_N, a_N}{\text{minimize}} \sum_{k=0}^{N} c(s_k, a_k)$$

$$\text{subject to} \quad s_{k+1} = f(s_k, a_k)$$

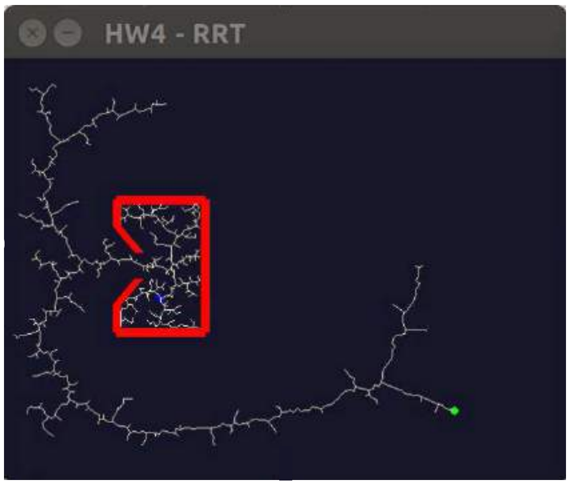
$$s_N = s_{\text{goal}}$$

$$V_N(s_N) = c(s_N, a_N)$$

$$V_{k+1}(s) = \min_a c(s, a) + V_k\big(f(s, a)\big)$$

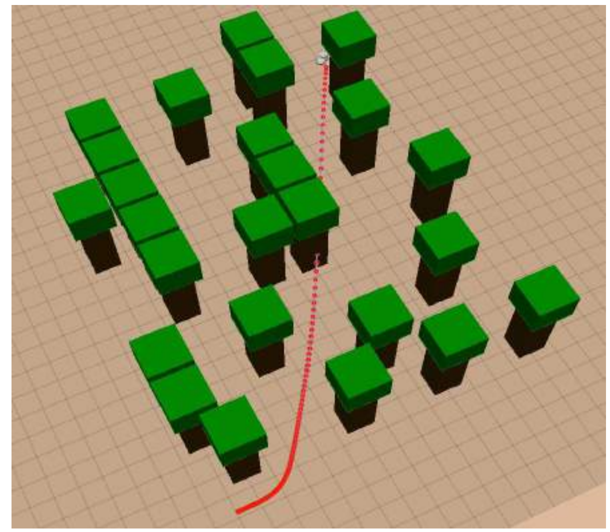
Sadly again the complexity scales with $d^{|S|=|A|}$ and those can get **HUGE** fast! This is the **"curse of dimensionality"** again

5 Planning in Configuration Space



Random Search

Machine Learning



Local Search

5 Trajectory Optimization

What if instead of finding a globally optimal path we search for a locally optimal path (off of some initial condition)?

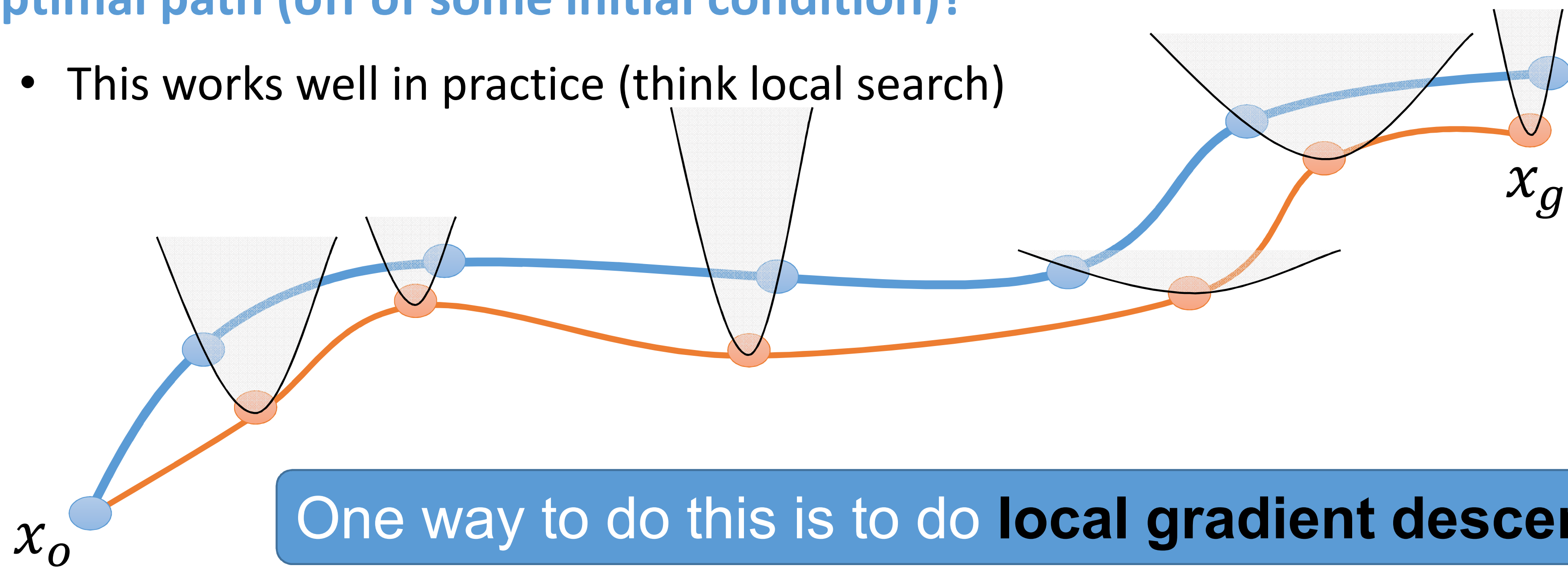
- This works well in practice (think local search)

By making slight perturbations to the current trajectory (blue) we can get to the goal (orange)

5 Trajectory Optimization

What if instead of finding a globally optimal path we search for a locally optimal path (off of some initial condition)?

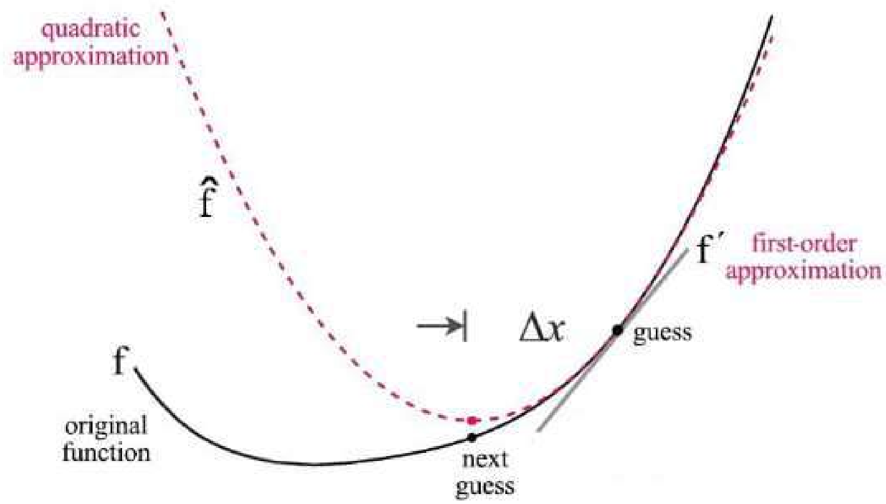
- This works well in practice (think local search)



One way to do this is to do **local gradient descent**

5 Trajectory Optimization

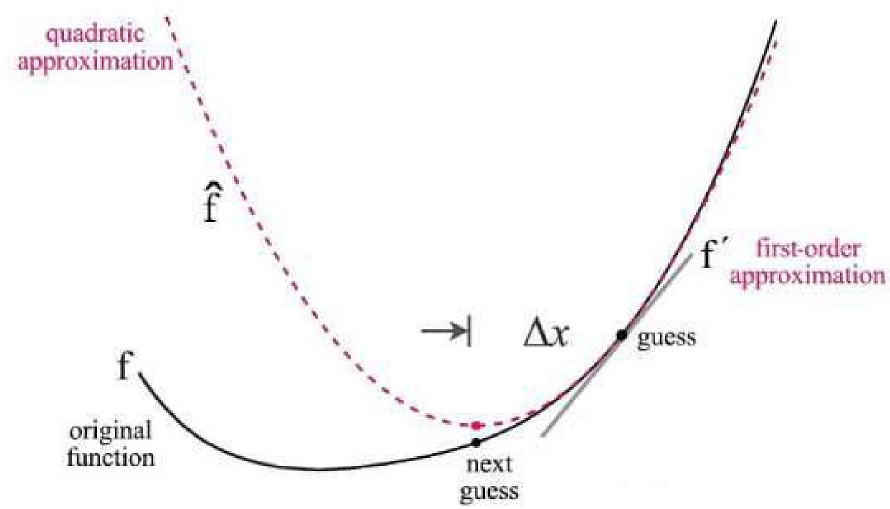
I'm drawing small quadratic bowls because most (if not all) of the practical algorithms make **linear and quadratic approximations** of the nonlinear functions allowing for efficient gradient descent



5 Trajectory Optimization

And convex optimization tells us how to descend to the minima of a quadratic function

I'm drawing small quadratic bowls because most (if not all) of the practical algorithms make **linear and quadratic approximations** of the nonlinear functions allowing for efficient gradient descent



5 Trajectory Optimization

There are also a whole host of algorithms one can use to solve these problems including:

- DDP, SQP, Interior-Point Methods, Trust-Region Methods, Stochastic Gradient Descent Methods, etc.

And you can use off-the-shelf solvers to solve these problems. Popular solvers include:

- SNOPT, IPOPT, NLOPT, fmincon (MATLAB), etc.
- **Most people use off the shelf solvers!**

5 Trajectory Optimization

So trajectory optimization solves everything right?

- Can handle full robot **dynamics**
- No need for distance metrics
- Can use **off the shelf solvers** reducing the coding burden
- Finds a **locally optimal** solution – no weird paths coming out!
  - Extra motions are "optimized away"

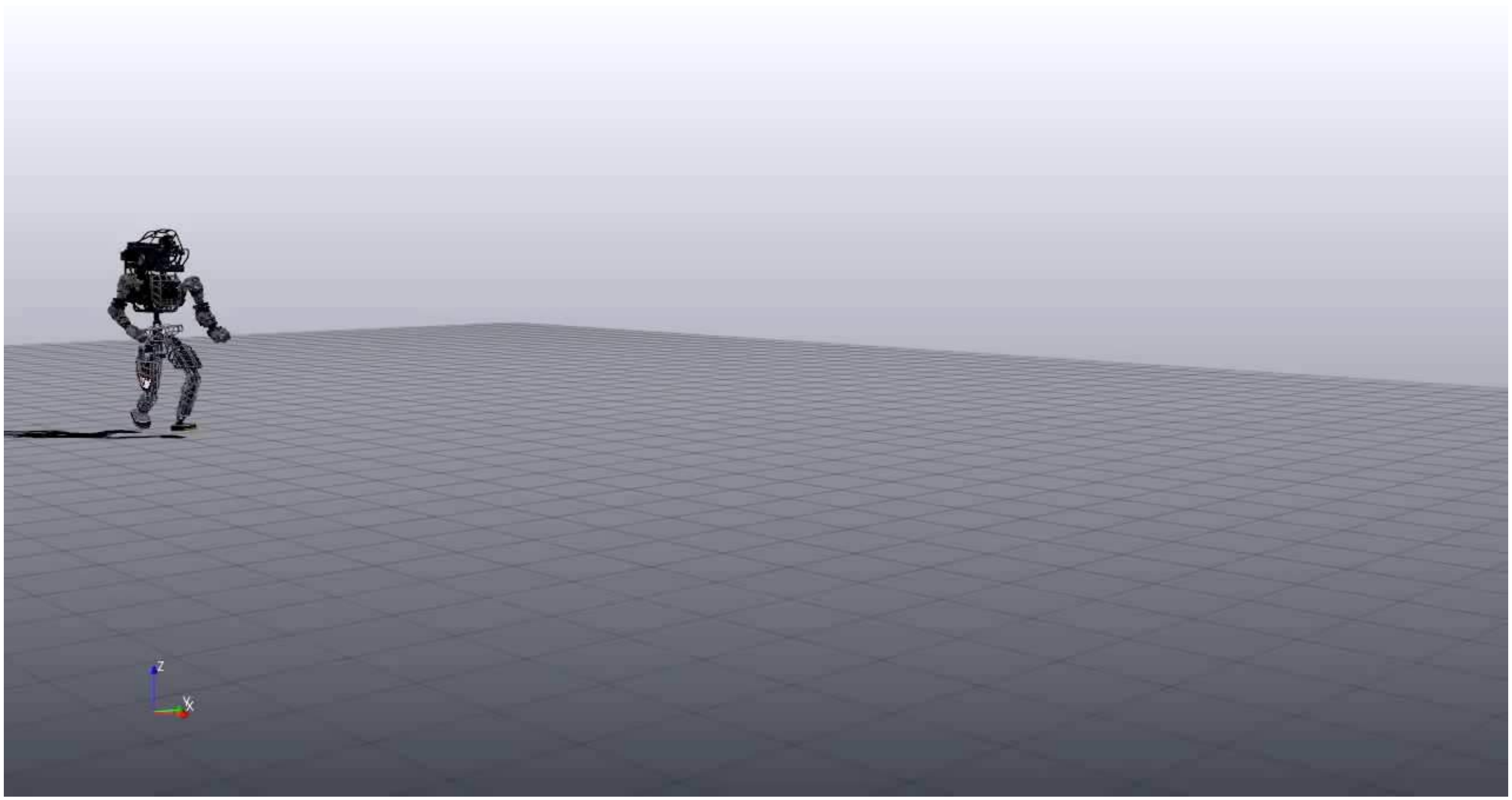
5 Trajectory Optimization

So trajectory optimization solves everything right?

- Can handle full robot **dynamics**
- No need for distance metrics
- Can use **off the shelf solvers** reducing the coding burden
- Finds a **locally optimal** solution – no weird paths coming out!
  - Extra motions are "optimized away"

And optimal motions often look bio-inspired as nature generally uses optimally efficient motions!

5 Atlas 1.0 Trajectory Optimization



5 Trajectory Optimization

So trajectory optimization solves everything right?

- Can handle full robot **dynamics**
- No need for distance metrics
- Can use **off the shelf solvers** reducing the coding burden
- Finds a **locally optimal** solution – no weird paths coming out!

No free lunch strikes again!

But….

- **Not globally optimal** (will often get stuck in local minima)
- **Not even complete** (problems are often non-convex so it may not even find a feasible solution)
- **Also generally slow**

5 Trajectory Optimization

So trajectory optimization solves everything right?

- Can handle full robot **dynamics**
- No need for distance metrics
- Can use **off the shelf solvers** reducing the coding burden
- Finds a **locally optimal** solution – no weird paths coming out!

No free lunch strikes again!

But….

- **Not globally optimal** (will often get stuck in local minima)
- **Not even complete** (problems are often non-convex so it may not even find a feasible solution)
- $\boxed{\textbf{Also generally slow}}$ **Lets dive a little deeper into solvers!**

5 There are two popular classes of solvers

	Shooting Methods (e.g., DDP, iLQR)	**Direct Methods** (e.g., DIRTRAN using SQP or IP)
Pros	• Known fast	• Easy to add constraints (e.g., torque limits, obstacle avoidance) • Easy to leverage **off the shelf solvers** (e.g., SNOPT, IPOPT)
Cons	• Hard to add constraints (e.g., torque limits, obstacle avoidance) • Generally people code it themselves	• Considered slow

5 There are two popular classes of solvers

Pros

Cons

• K

• H
t

• C
t

ice)

f

Technical note: DDP reduces to a specific factorization of the **KKT matrix** solve in a direct method to exploit sparsity!

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x + r \\ \text{subject to} & Ax = b \end{array}$$

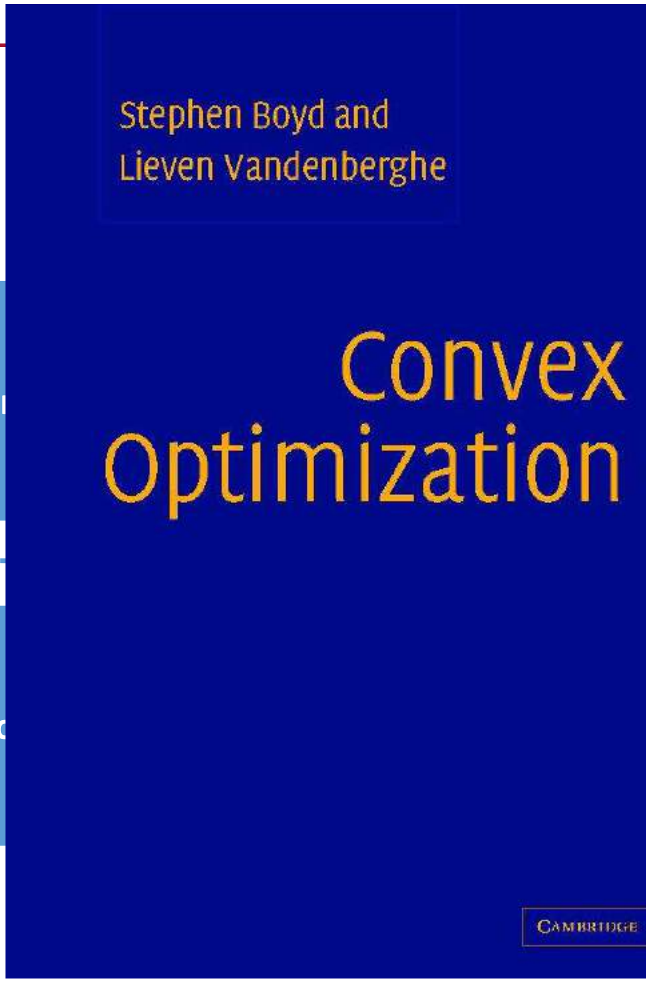
$$\begin{bmatrix} P & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^\star \\ \nu^\star \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix}$$

5 There are two popular classes of solvers

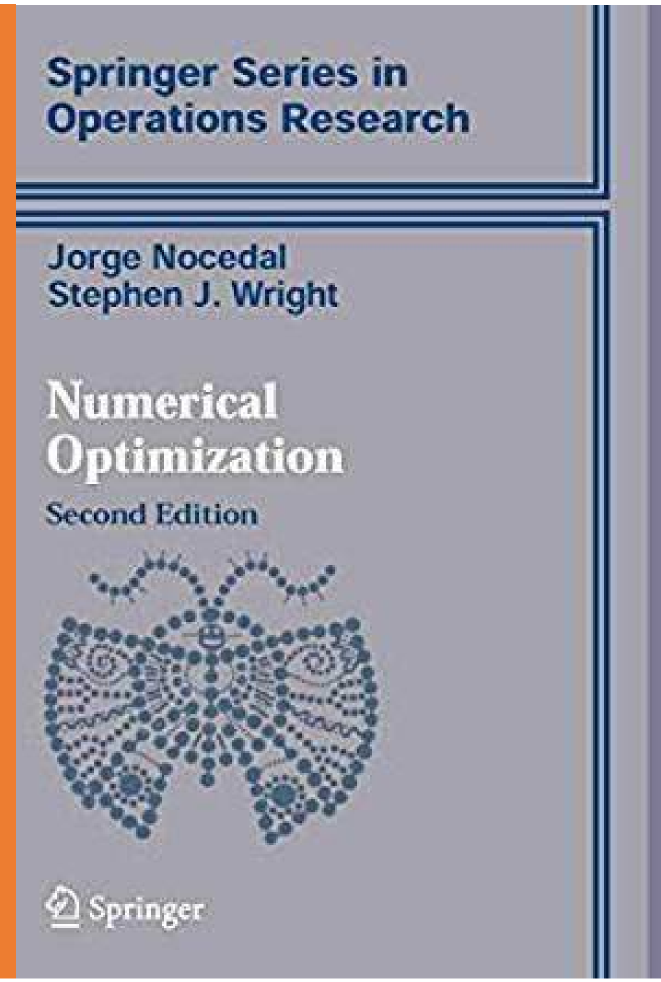
	Shooting M… (e.g., DDP, i…	Direct Methods (e.g., DIRTRAN using SQP or IP)
Pros	• Known fast	• …asy to add constraints (e.g., …orque limits, obstacle avoidance) • …asy to leverage **off the shelf …olvers** (e.g., SNOPT, IPOPT)
Cons	• Hard to add constra… torque limits, obsta… • Generally people c… themselves	• …onsidered slow

I'll dig a little deeper / explain this more in 2 weeks when I present my research on parallel shooting methods

5 There are two popular classes of solvers



But/and these are two great textbooks if you want to learn more about the math!



5 Practical Challenges for Trajectory Optimization: Robustness

Manchester and Kuindersma 2017
Plancher and Kuindersma 2018

1. Solvers are (numerically) sensitive to:
   - Cost function designs and dynamic range
   - Regularization scheme

2. Solutions are sensitive to:
   - Initial state and input trajectories
   - Perturbations (solutions are often on constraint boundaries)

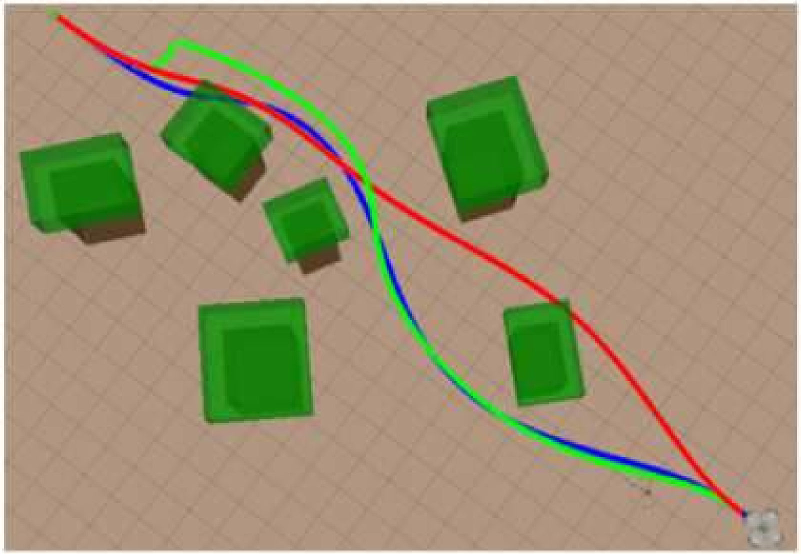


Fig. 4. DIRTRAN (red), DIRTREL-1 (blue), and DIRTREL-2 (green) quadrotor trajectories.

5 Practical Challenges for Trajectory Optimization: Contact

Tedrake Underactuated

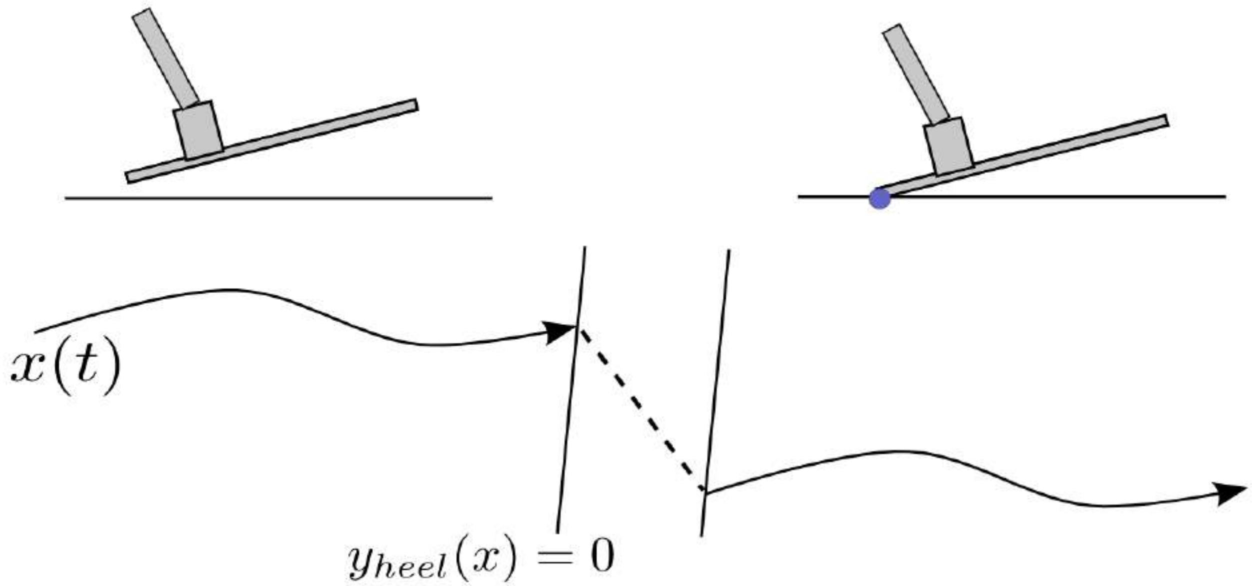
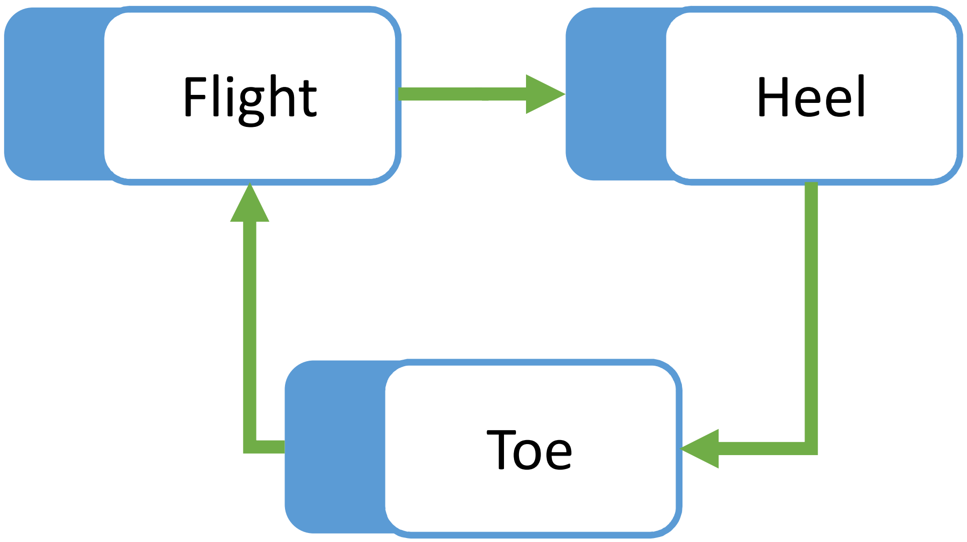


Figure 17.1 - Modeling contact as a hybrid system.

The physics equations are fundamentally different when an object makes or breaks contact

5 Practical Challenges for Trajectory Optimization: Contact

Tedrake Underactuated



For walking these **hybrid modes** form a cyclic graph

If we **pre-specify** the mode sequence and timing we can use our algorithms as before

5 Practical Challenges for Trajectory Optimization: Contact

Manchester and Kuindersma 2017

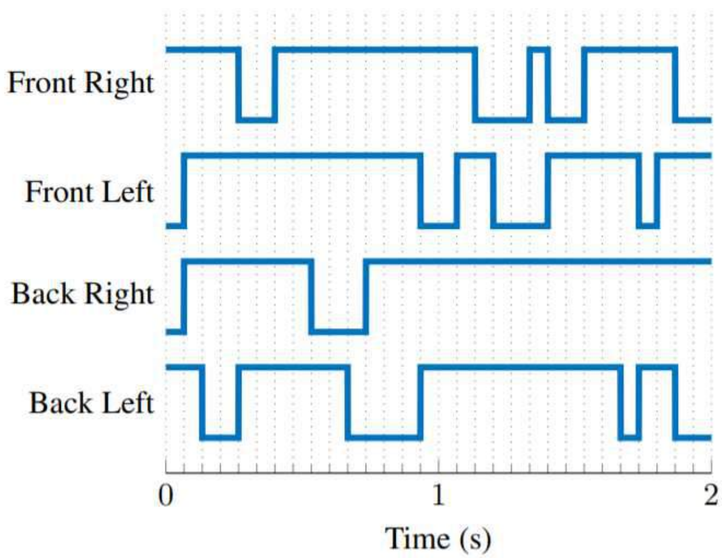
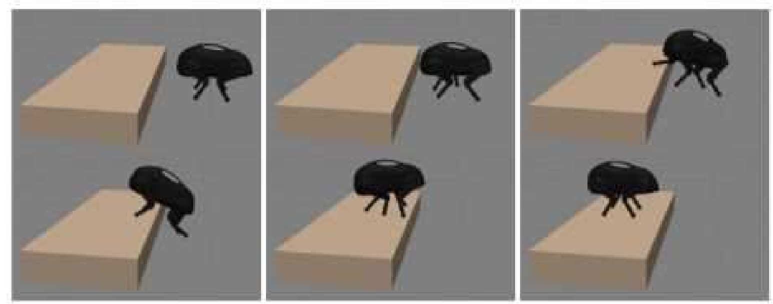


Figure 4. Contact mode sequence for each foot from LittleDog step-climbing example.



But for **complex actions** these modes start to become **hard to pre-specify**

5 Practical Challenges for Trajectory Optimization: Contact

Doshi, et. al. 2018

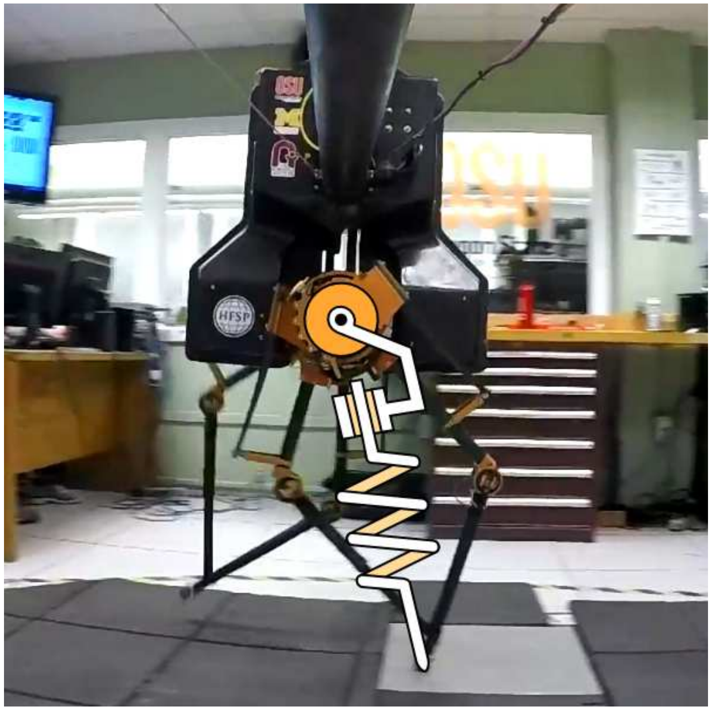
Contact-Implicit Trajectory Optimization includes the contact timings and mode transitions as state variables

2Hz Gaits
(real-time)

But these approaches are computationally very expensive (read **offline**) as the number of modes explodes combinatorically with the number of contact points (Mixed-Integer Programming)!

5 Practical Challenges for Trajectory Optimization: Contact

One approach to avoid solving these large hard problems is to solve the problem on **simpler models** of the system



5 Practical Challenges for Trajectory Optimization: Contact

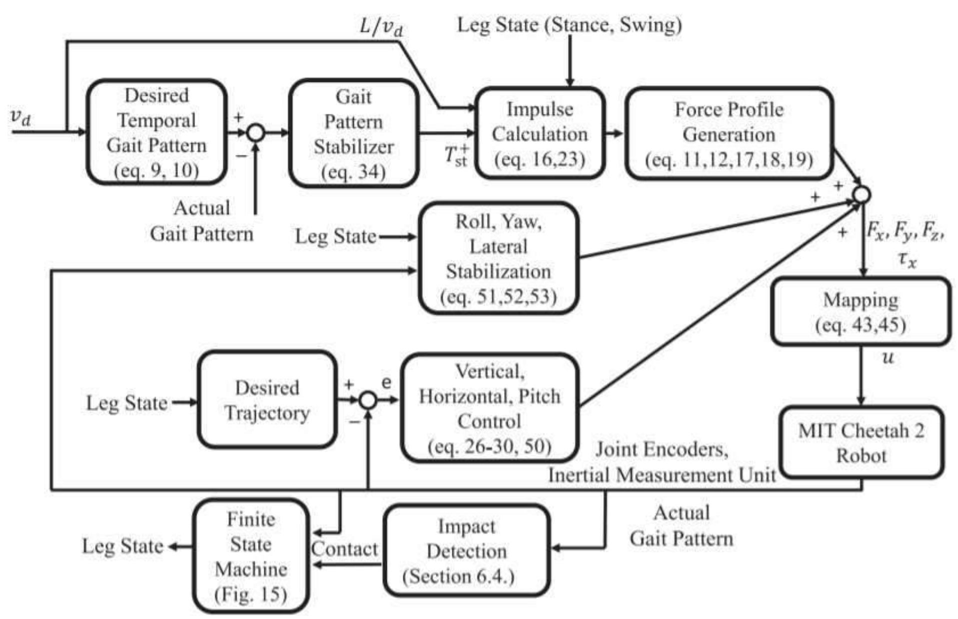
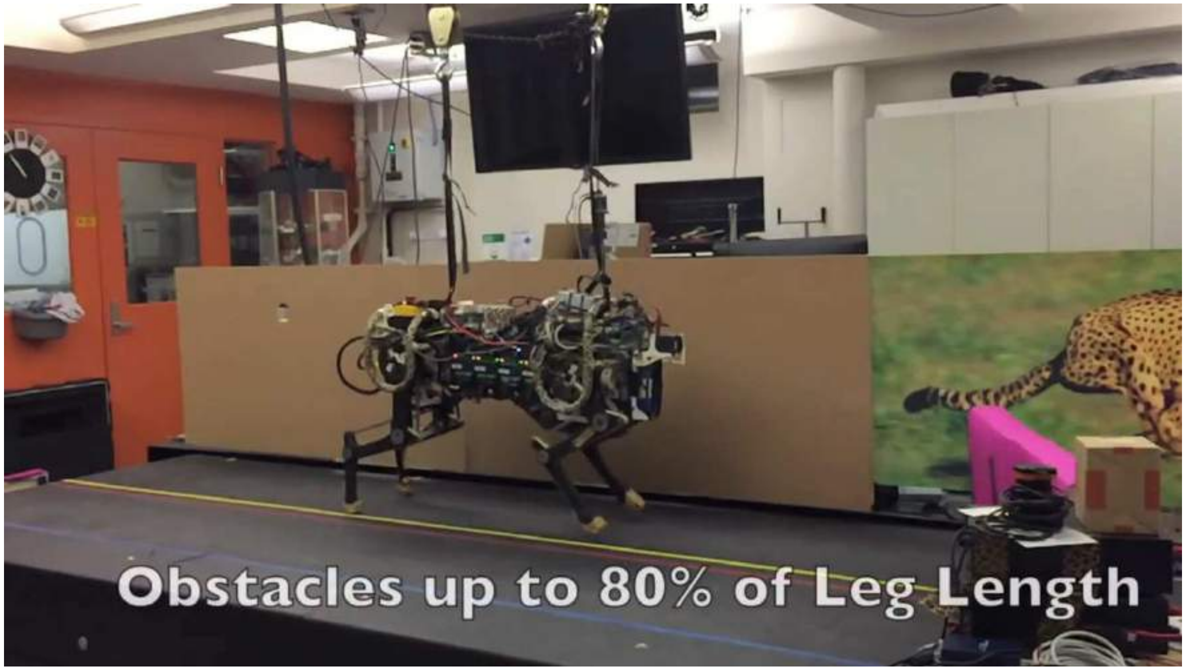


Fig. 11. Overall control system diagram.

And then **combine solutions** to these **(conservative)** simpler problems

5 Practical Challenges for Trajectory Optimization: Contact

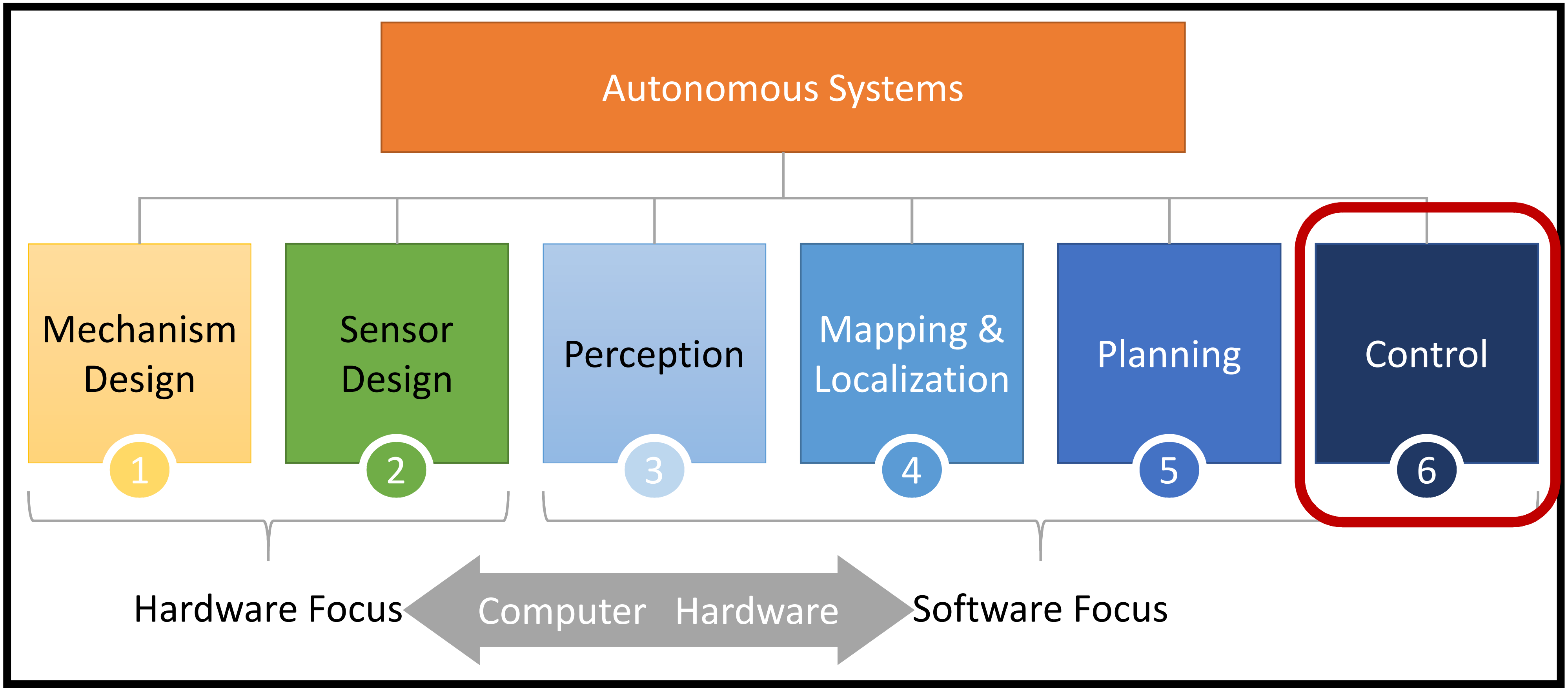


And then **combine solutions** to these **(conservative)** simpler problems

5 Key Takeaways:

1. Robot planning involves both **task and configuration spaces**
2. For many real problems, **collision checking** can be expensive
3. **Sample Based Planners** that leverage random search **(RRT/PRM)**:
   - **Probabilistically complete** (but can still be slow sometimes)
   - **Single-query (RRT) vs. Multi-query (PRM)**
   - **Probabilistically optimal** (RRT*) but generally need smoothers
4. **Trajectory Optimization** leverages local search to find **locally optimal** (generally smooth) solutions
   - Handles dynamics well but **not complete or robust**
   - Can use **off the shelf solvers** (SQP) but **generally slower** than a solver that **exploits sparsity** in the problem (DDP/iLQR)
   - **Contact is hard** and we (sometimes) use **simpler models** for tractability

Autonomous Systems / Robotics is a BIG space

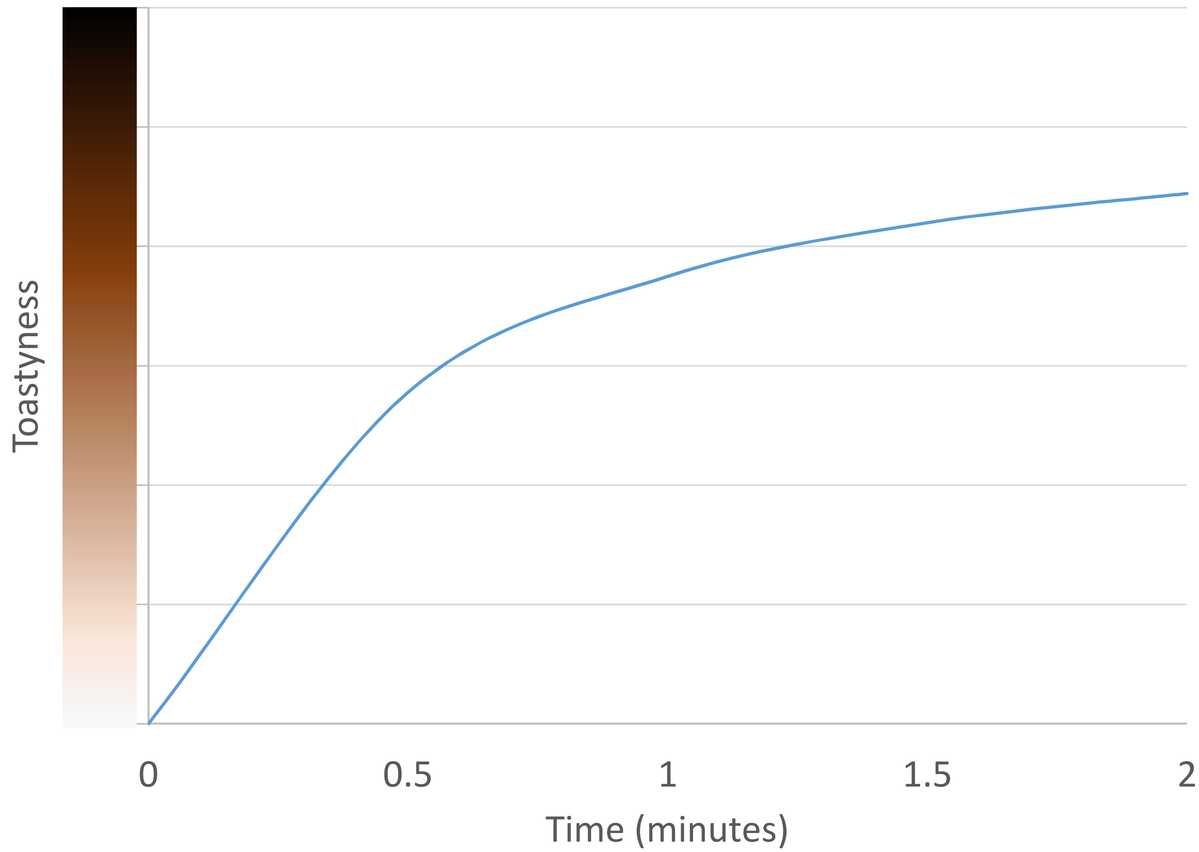


6 Control is the process of executing a plan in the real world

Well the simplest thing we could try would be to just execute the controls from our plan directly on the real system. This is called **Open-Loop Control!**

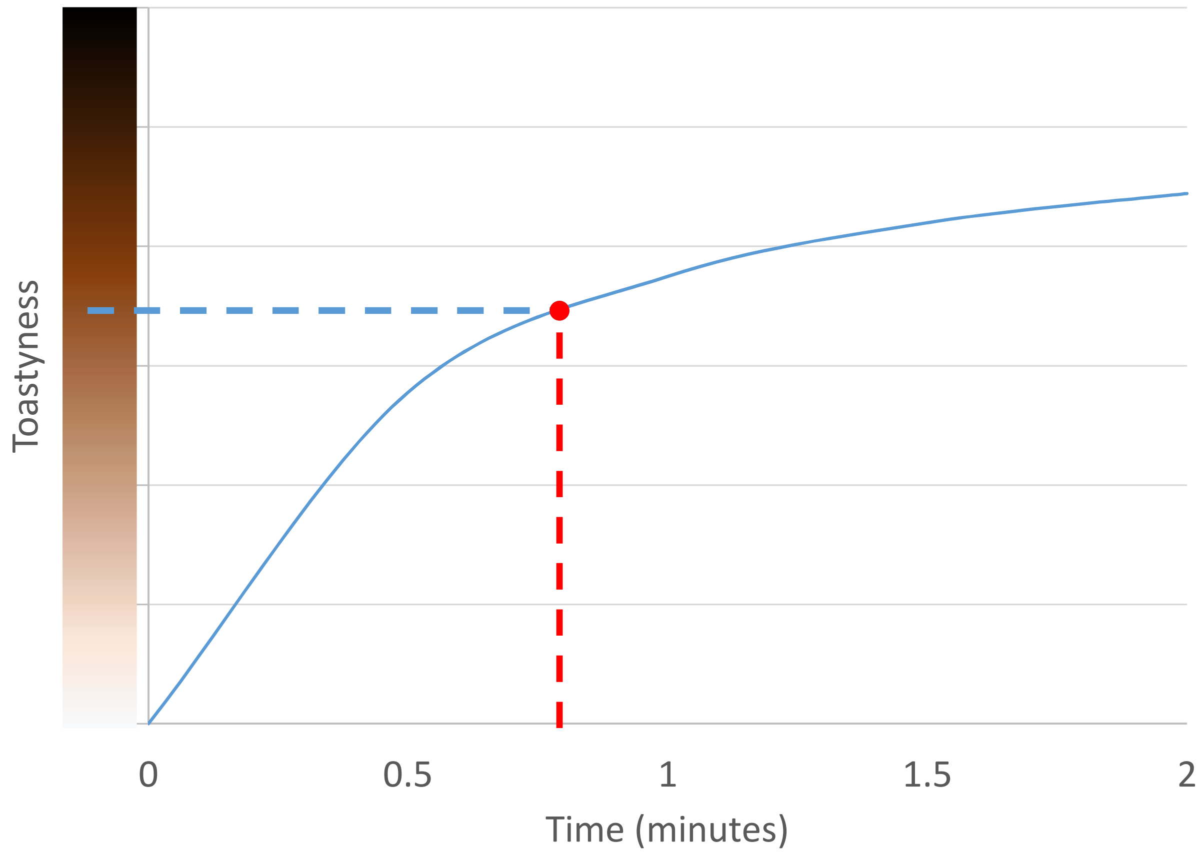
6 Open Loop Control

Adapted from MATLAB Control Toolbox



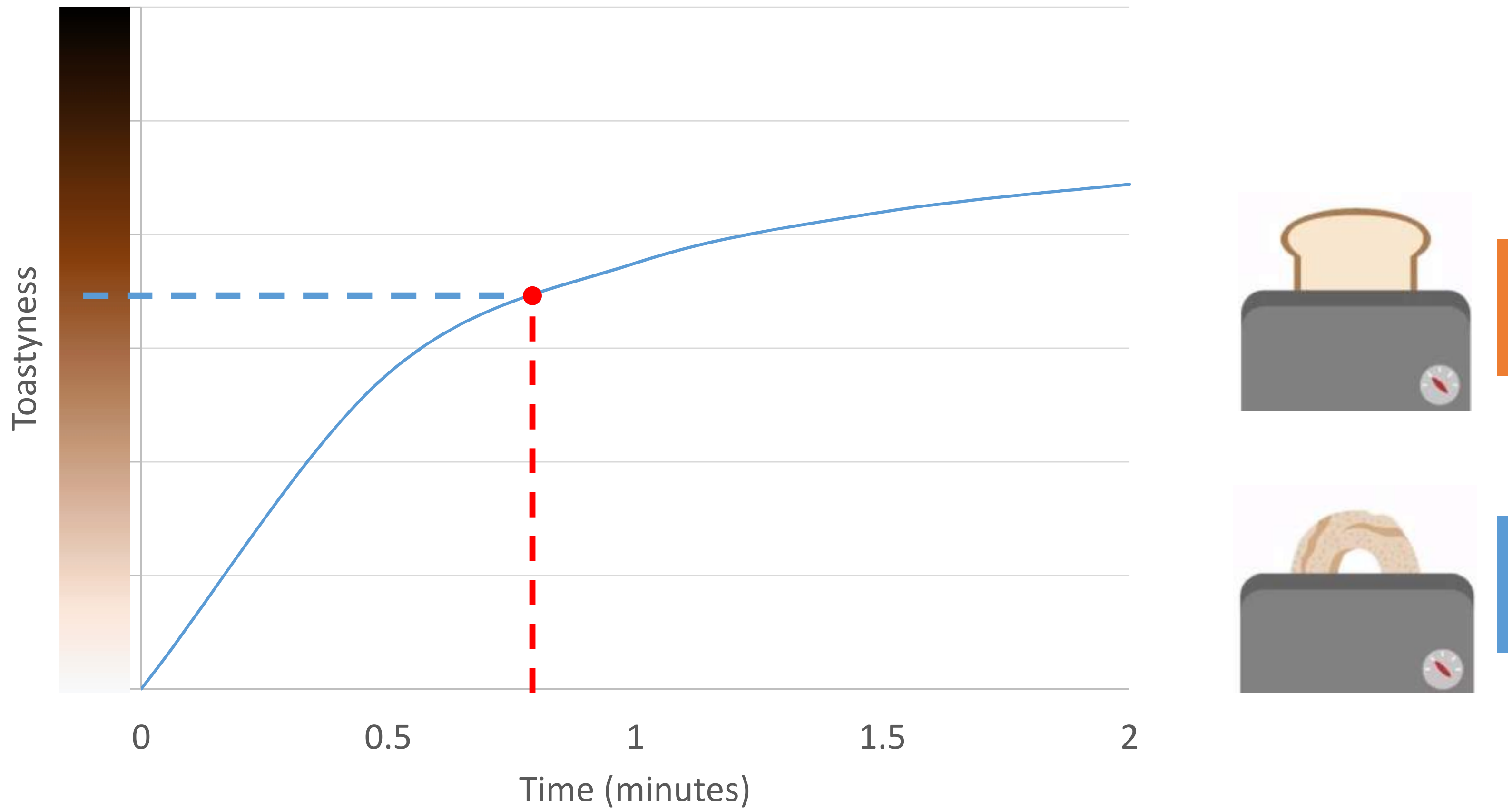
6 Open Loop Control

Adapted from MATLAB Control Toolbox



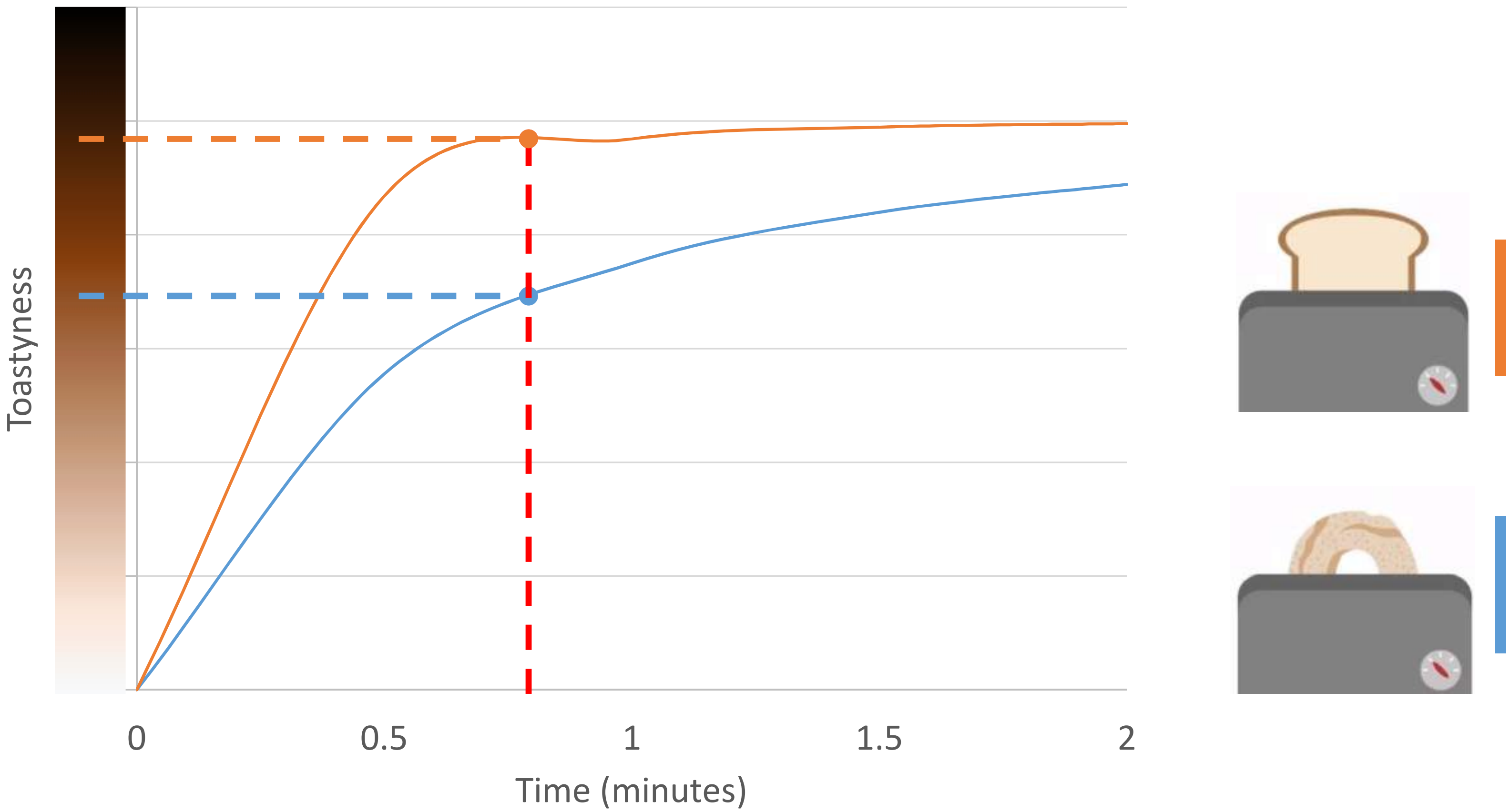
6 Open Loop Control

Adapted from MATLAB Control Toolbox



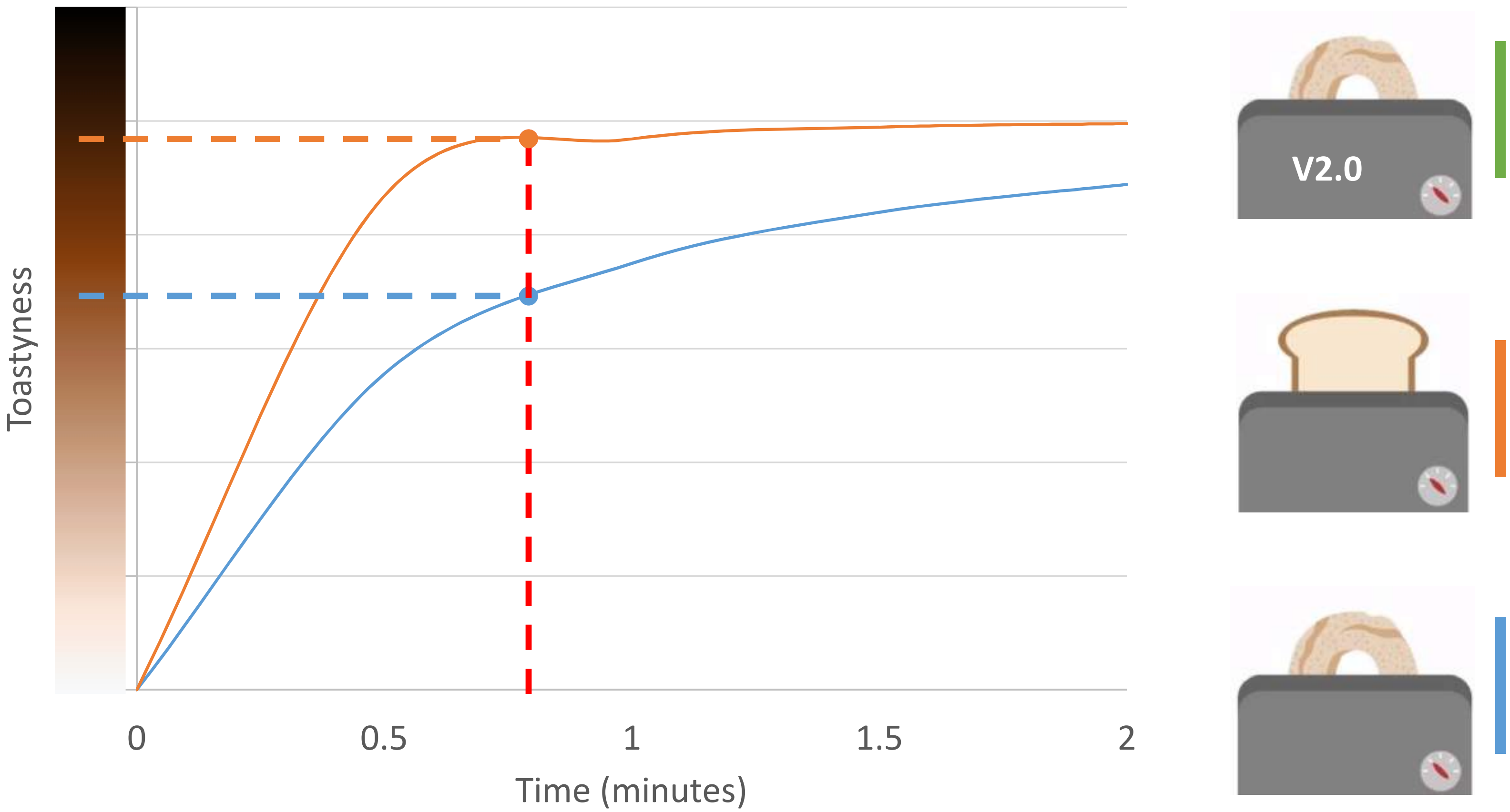
6 # Open Loop Control

Adapted from MATLAB Control Toolbox



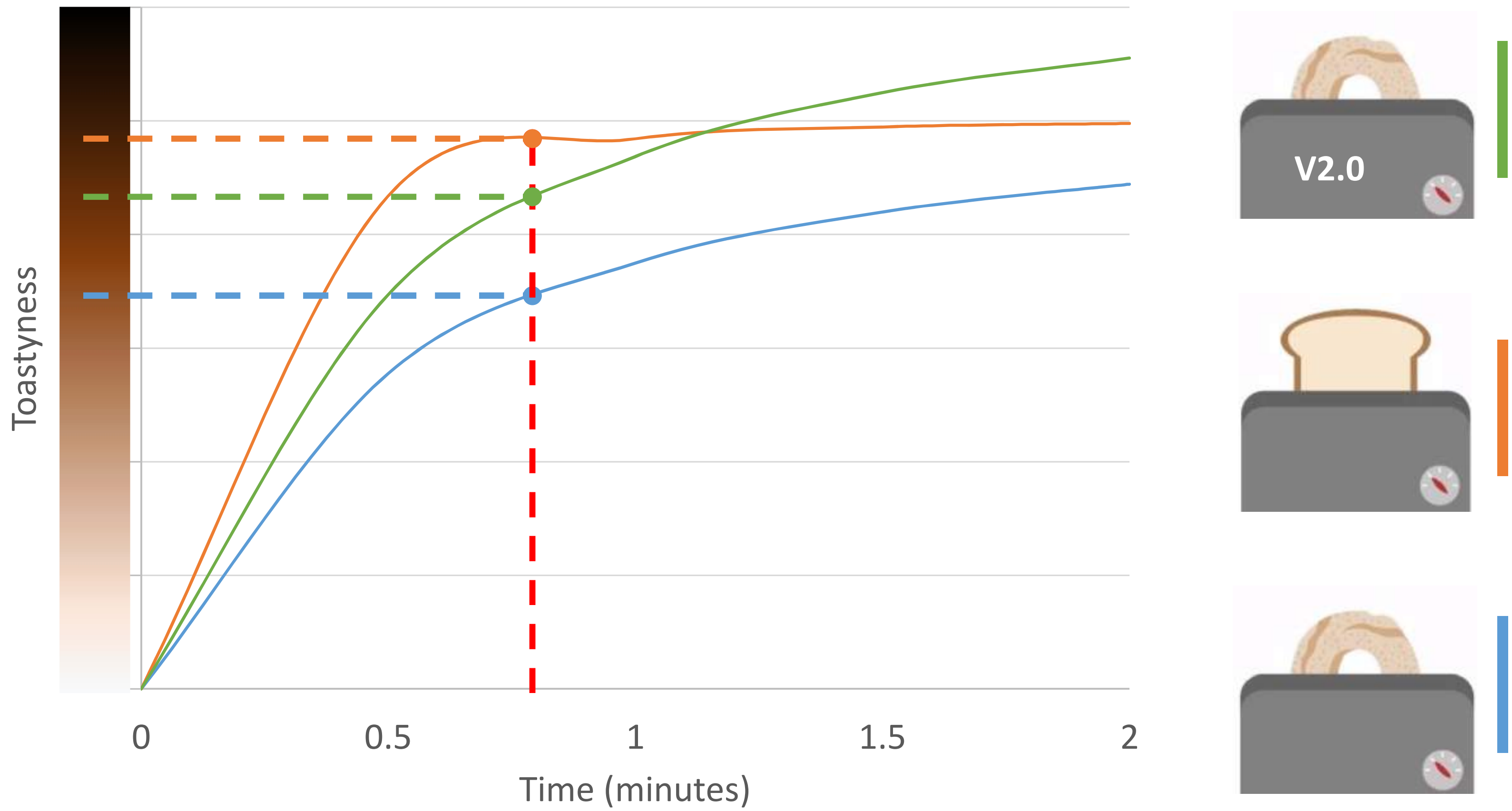
6 Open Loop Control

Adapted from MATLAB Control Toolbox



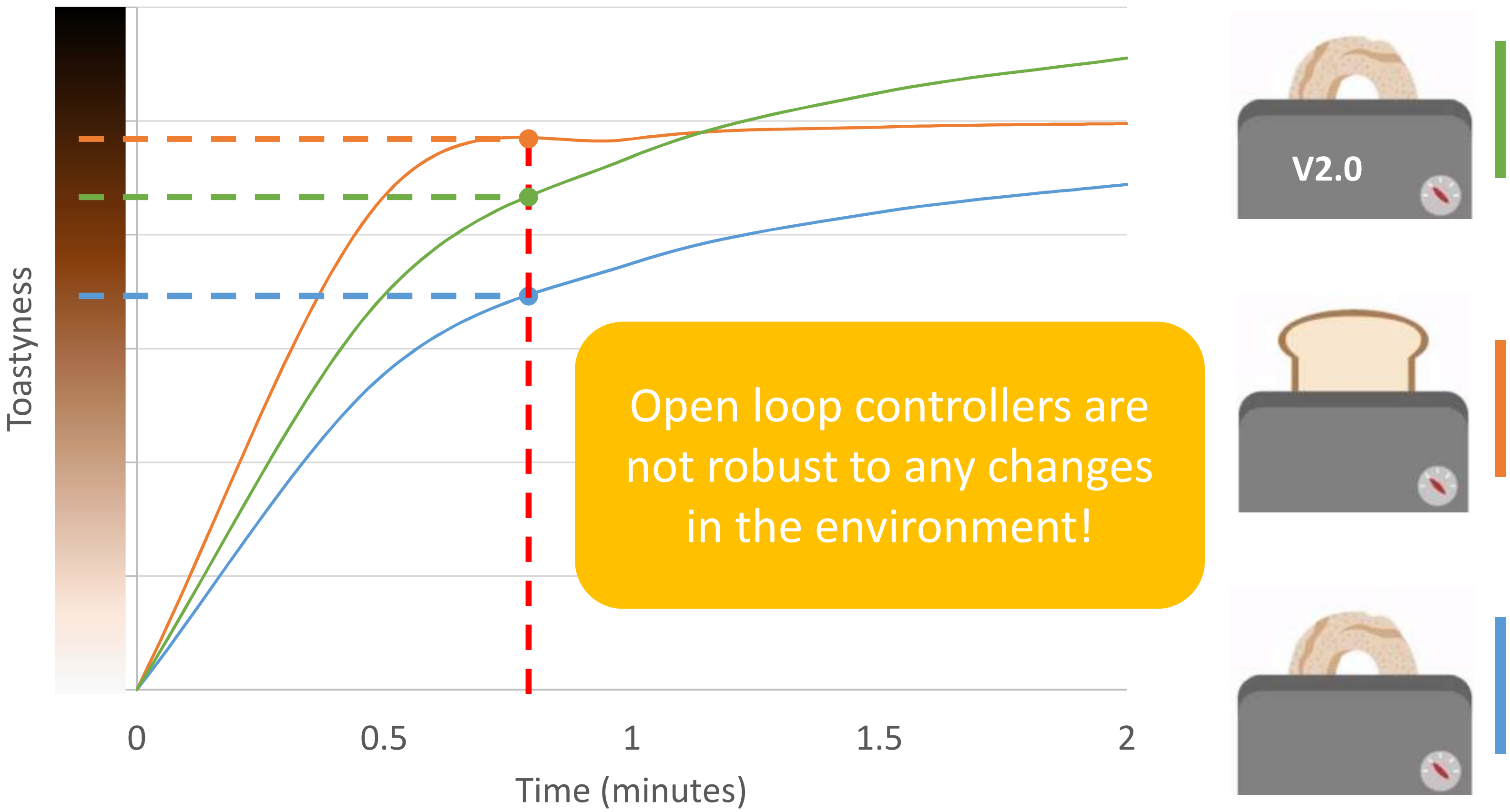
Open Loop Control

Adapted from MATLAB Control Toolbox



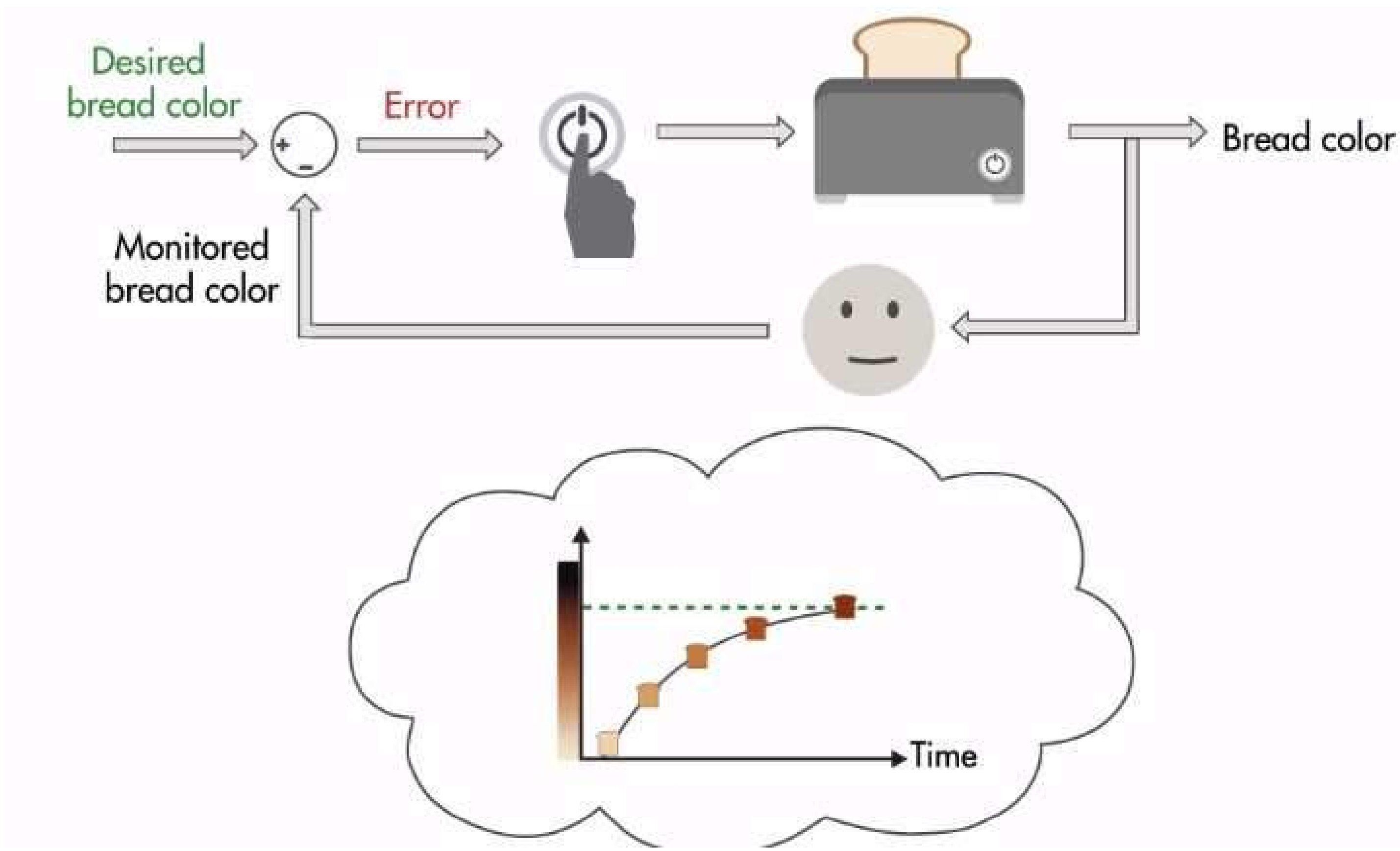
6 Open Loop Control

Adapted from MATLAB Control Toolbox



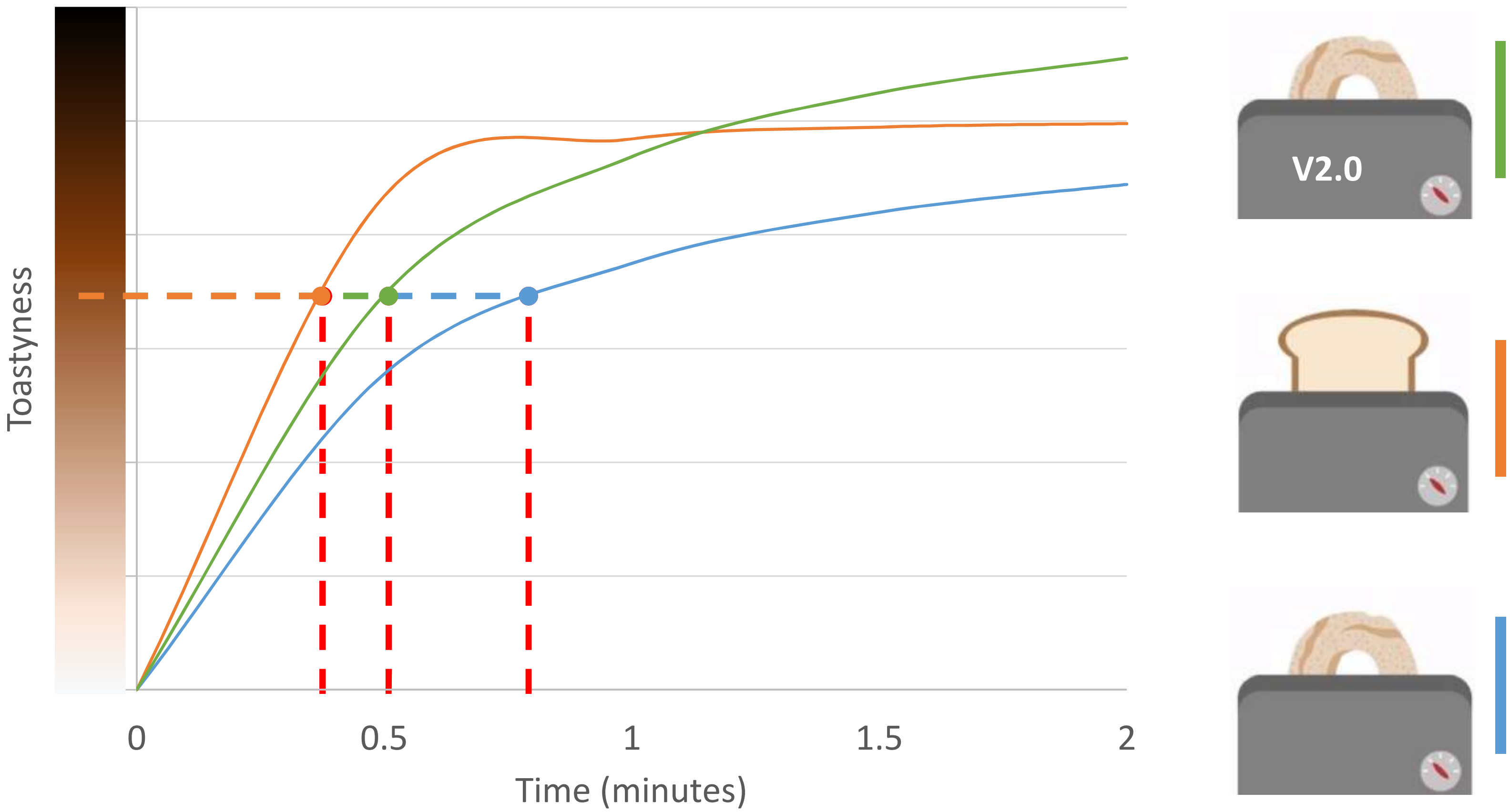
6 Feedback (Closed Loop) Control

Adapted from MATLAB Control Toolbox

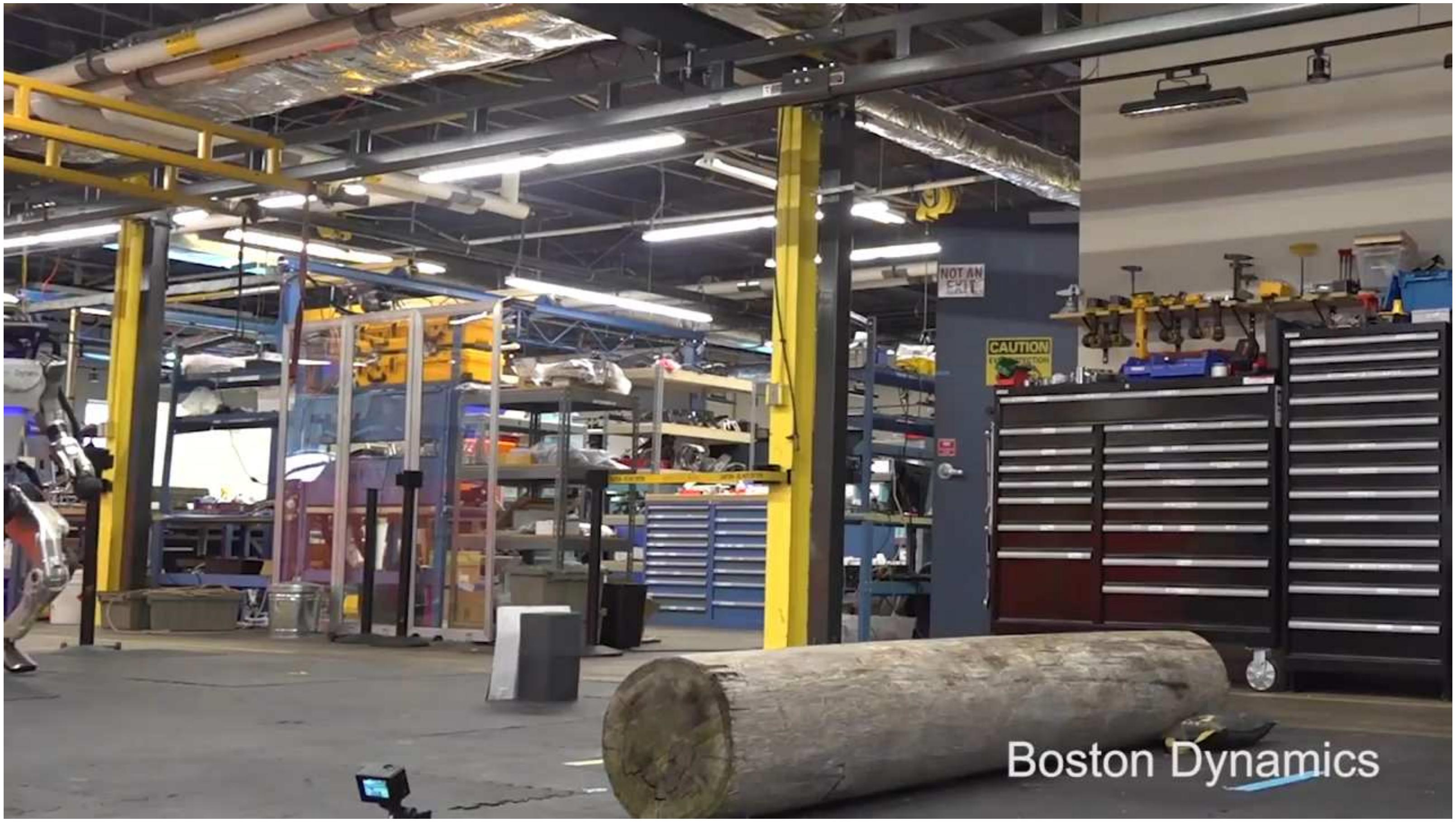


6 Feedback Control

Adapted from MATLAB Control Toolbox

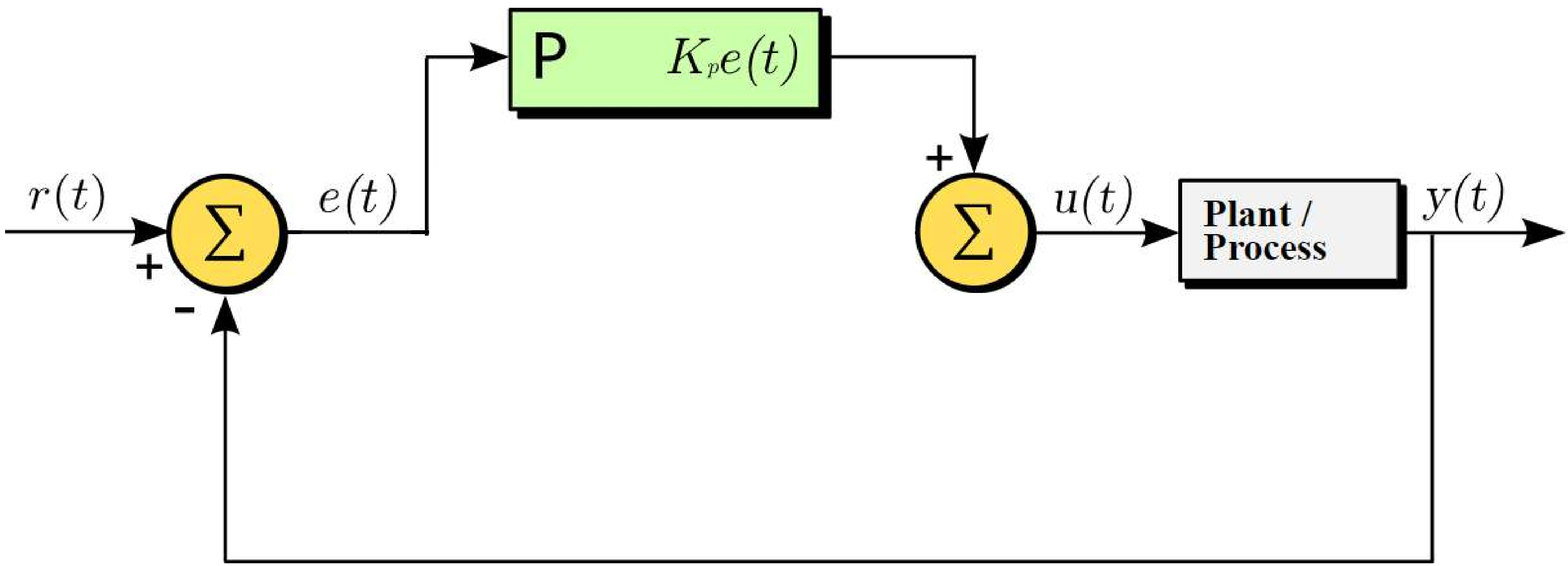


6 Feedback Control can lead to amazing performance!



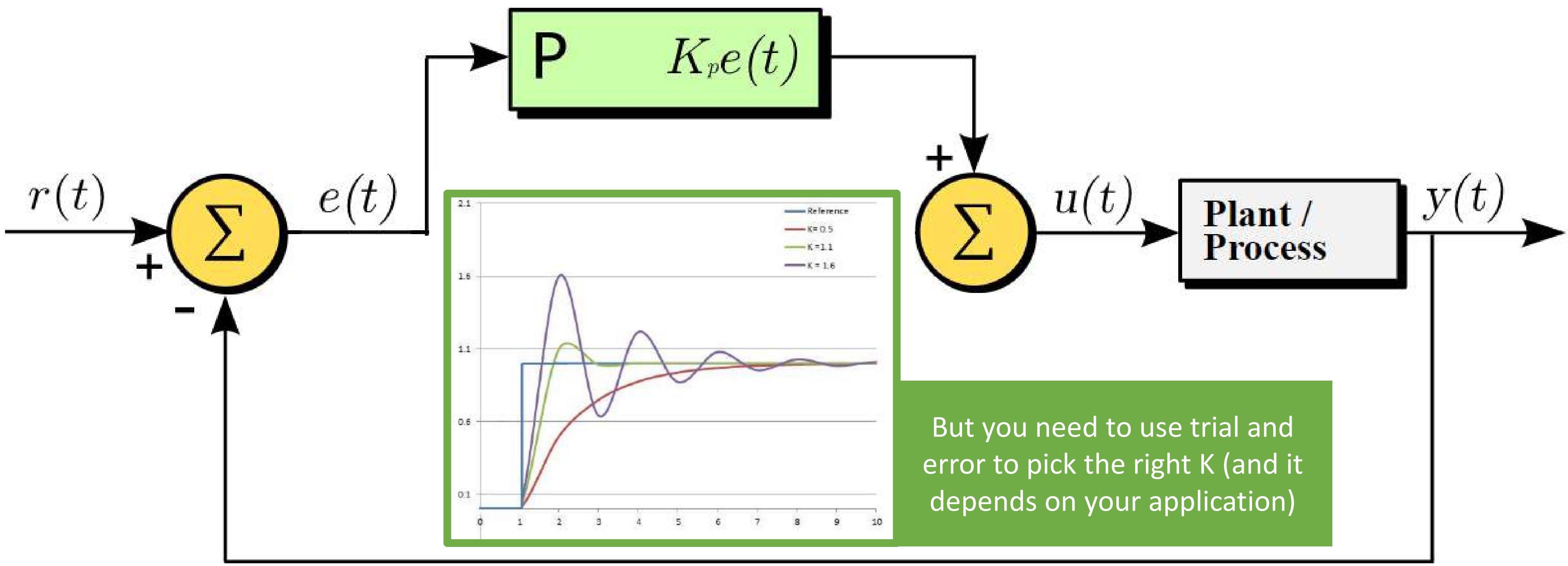
6 So how do we do Feedback Control in practice?

Adapted from Wikipedia



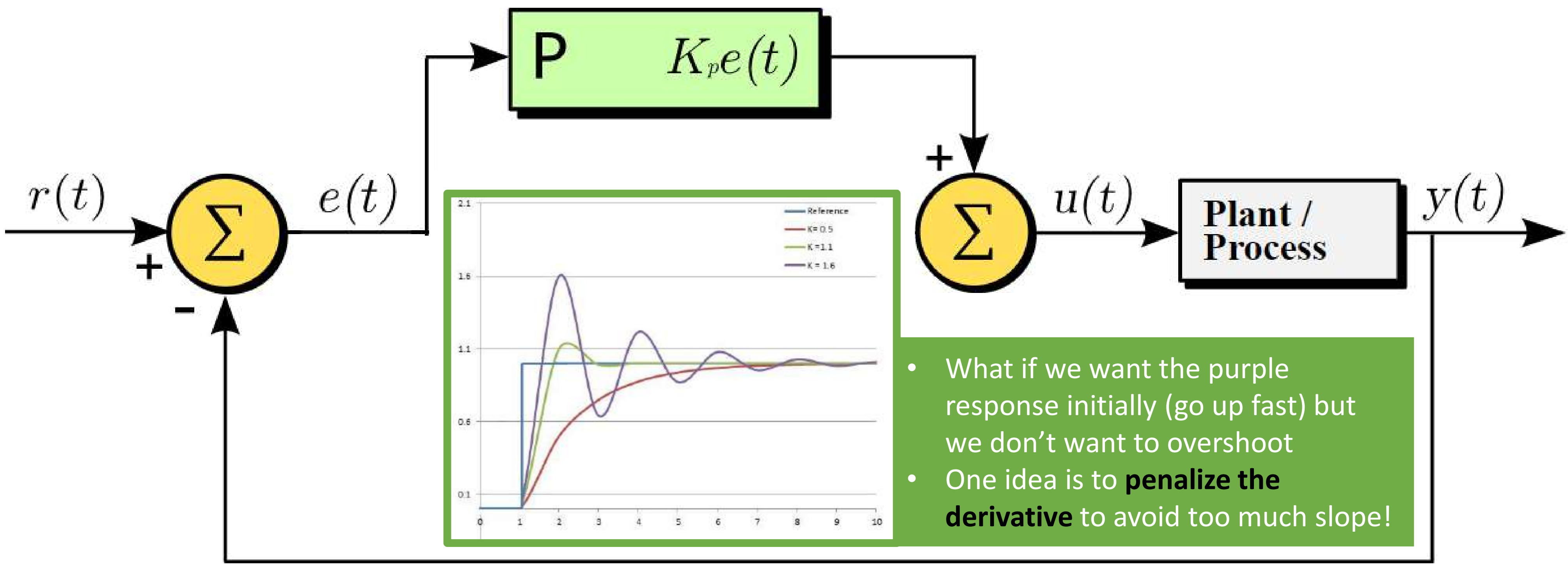
6 So how do we do Feedback Control in practice?

Adapted from Wikipedia



6 So how do we do Feedback Control in practice?

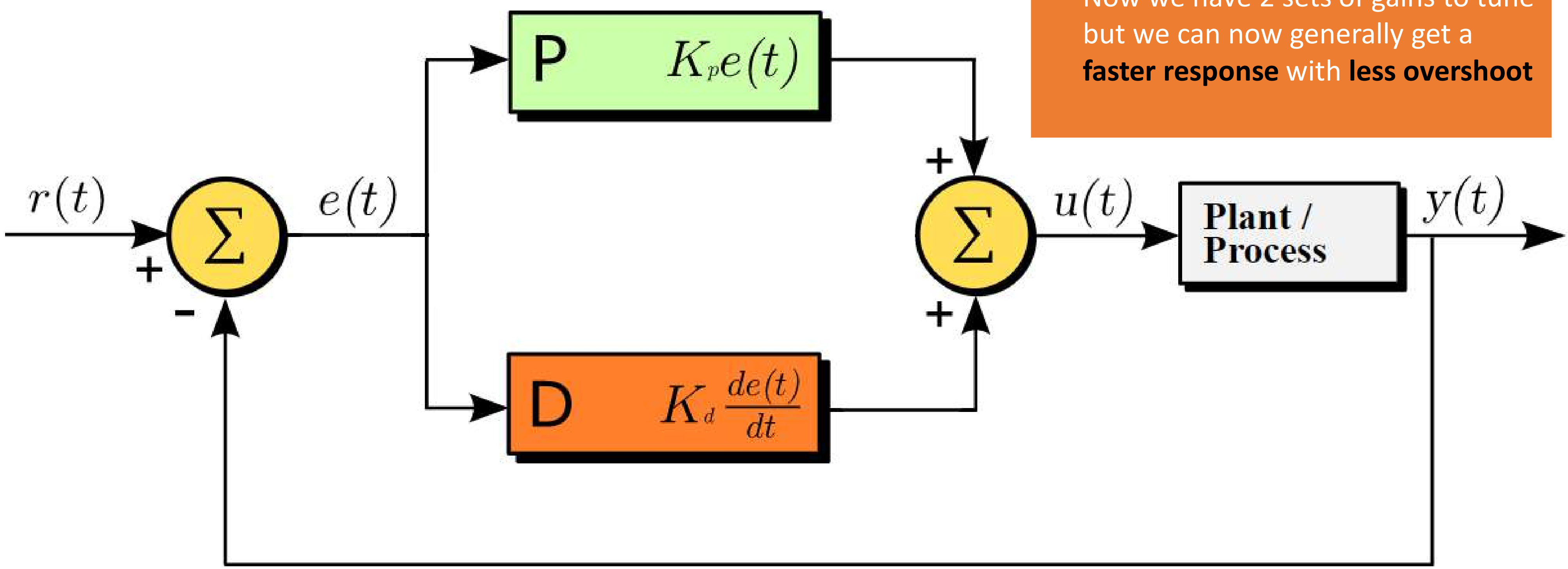
Adapted from Wikipedia



- What if we want the purple response initially (go up fast) but we don't want to overshoot
- One idea is to **penalize the derivative** to avoid too much slope!

6 So how do we do Feedback Control in practice?

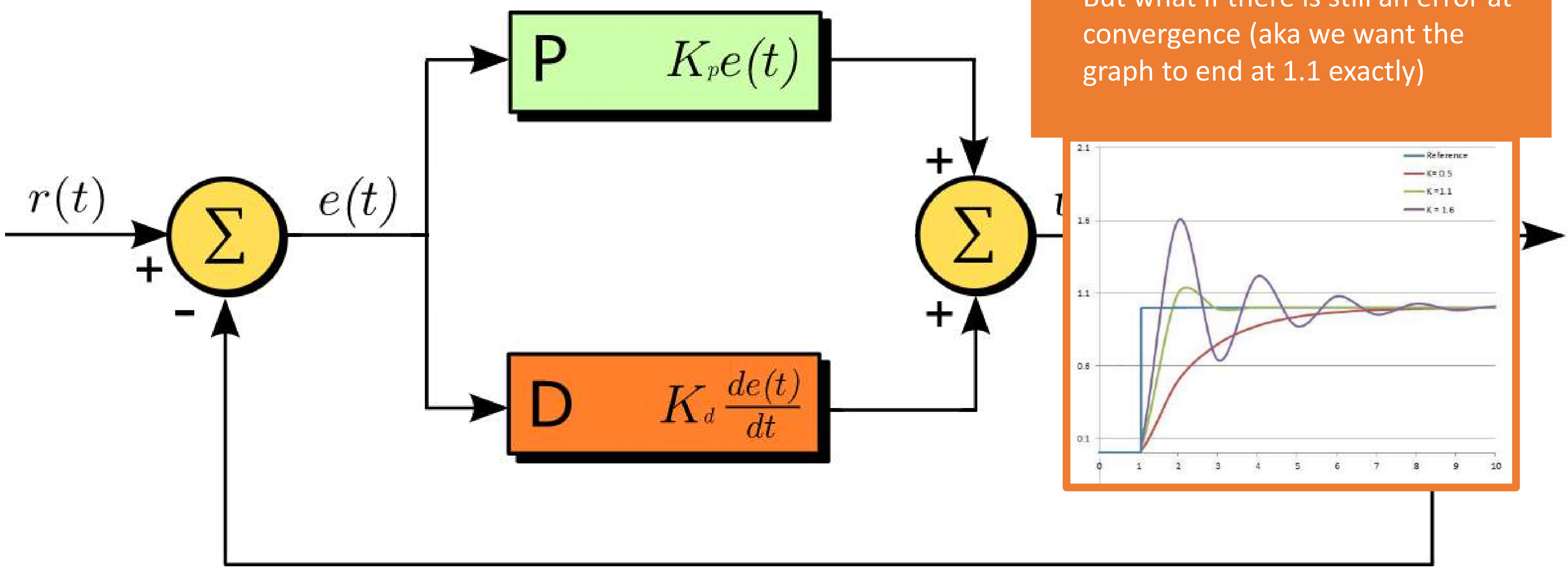
Adapted from Wikipedia



- Now we have 2 sets of gains to tune but we can now generally get a **faster response** with **less overshoot**

❻ So how do we do Feedback Control in practice?

Adapted from Wikipedia

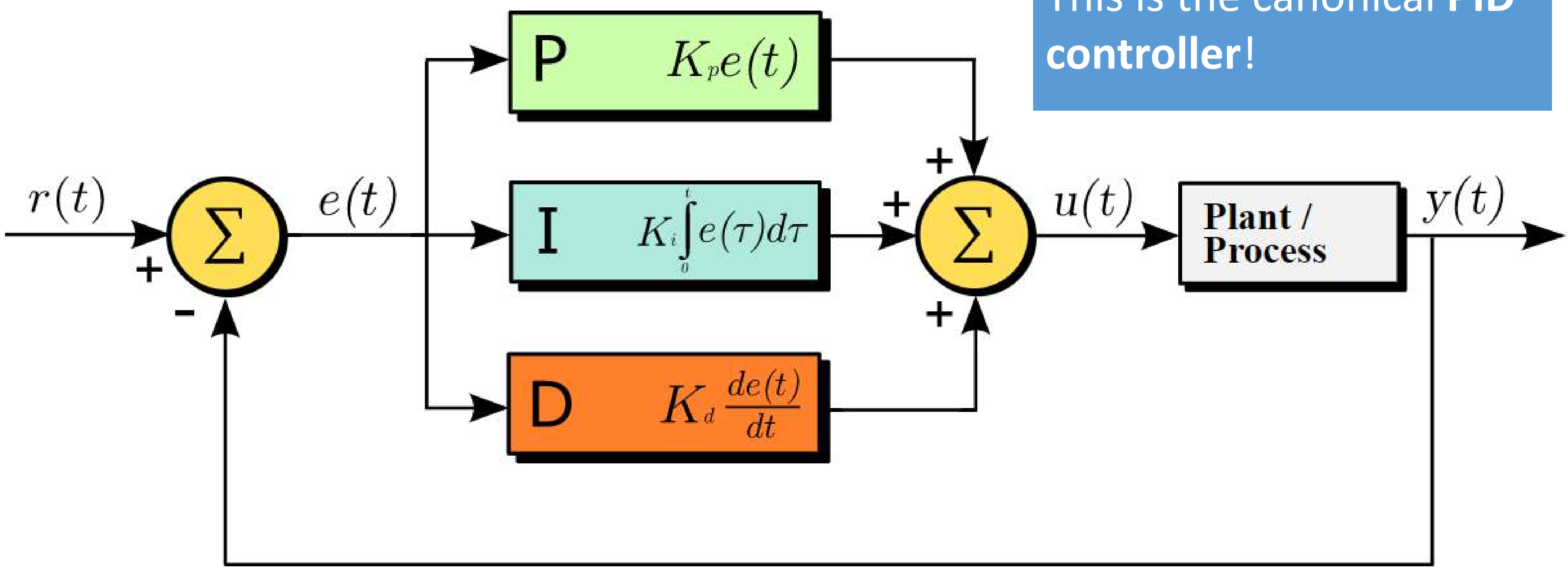


- But what if there is still an error at convergence (aka we want the graph to end at 1.1 exactly)

6 So how do we do Feedback Control in practice?

Adapted from Wikipedia

This is the canonical **PID controller**!



6 So how do we do Feedback Control in practice?

Adapted from Wikipedia

Ziegler–Nichols method

From Wikipedia, the free encyclopedia

Main article: PID controller

The **Ziegler–Nichols tuning method** is a heuristic method of tuning a PID controller. It was developed by John G. Ziegler and Nathaniel B. Nichols. It is performed by setting the *I* (integral) and *D* (derivative) gains to zero. The "P" (proportional) gain, K_p is then increased (from zero) until it reaches the **ultimate gain** K_u, at which the output of the control loop has stable and consistent oscillations. K_u and the oscillation period T_u are used to set the P, I, and D gains depending on the type of controller used:

Ziegler–Nichols method[1]

Control Type	K_p	T_i	T_d	K_i	K_d
P	$0.5K_u$	–	–	–	–
PI	$0.45K_u$	$T_u/1.2$	–	$0.54K_u/T_u$	–
PD	$0.8K_u$	–	$T_u/8$	–	$K_uT_u/10$
classic PID[2]	$0.6K_u$	$T_u/2$	$T_u/8$	$1.2K_u/T_u$	$3K_uT_u/40$
Pessen Integral Rule[2]	$7K_u/10$	$2T_u/5$	$3T_u/20$	$1.75K_u/T_u$	$21K_uT_u/200$
some overshoot[2]	$K_u/3$	$T_u/2$	$T_u/3$	$0.666K_u/T_u$	$K_uT_u/9$
no overshoot[2]	$K_u/5$	$T_u/2$	$T_u/3$	$(2/5)K_u/T_u$	$K_uT_u/15$

Tuning PID gains is an art and there is a whole literature on a variety of methods to get particular types of response curves!

6 PID controllers work really well in practice

6 Tuning gains is hard and non-intuitive is there a better way?

6 Tuning gains is hard and non-intuitive is there a better way?

Of course there is or I wouldn’t need the transition slide!

6 The LQR Controller

What if instead of specifying gains we can specify a **cost function** we want to achieve…

6 The LQR Controller

What if instead of specifying gains we can specify a **cost function** we want to achieve…

Maybe something like track the desired state but don't use too much energy to do it?

6 The LQR Controller

What if instead of specifying gains we can specify a **cost function** we want to achieve…

Maybe something like track the desired state but don't use too much energy to do it?

Instead of tuning gains we can tune cost weights (Q,R) which are often more intuitive

$$L(x,u) = \underbrace{\left(x - x_g\right)^T Q \left(x - x_g\right)}_{\text{Deviation of the state from some goal state}} + \underbrace{u^T R u}_{\text{Effort (torque)}}$$

6 The LQR Controller

It turns out if we minimize this quadratic cost over time with a linear model of the dynamics

$$\min_{x,u} \sum_{k=0}^{N} (x_k - x_g)^T Q (x_k - x_g) + u_k{}^T R u_k$$

$$\text{s.t.} \quad x_{k+1} = A x_k + B u_k$$

There is a closed form solution to the optimal feedback controller! (Riccati Equation)

$$u_k = -K_k x_k$$

6 The LQR Controller

It turns out if we minimize this quadratic cost over time with a linear model of the dynamics

$$\min_{x,u} \sum_{k=0}^{N} (x_k - x_g)^T Q (x_k - x_g) + \; u_k{}^T R u_k$$

$$\text{s.t.} \quad x_{k+1} = A x_k + B u_k$$

There is a closed form solution to the optimal feedback controller! (Riccati Equation)

$$u_k = -K_k x_k$$

This is used widely in practice!

6

We can also use LQR in RRT as a better metric of "distance" and the feedback controller as the best "extend"

Finite-horizon, discrete-time LQR [edit]

For a discrete-time linear system described by: [1]

$$x_{k+1} = Ax_k + Bu_k$$

with a performance index defined as:

$$J = x_N^T Q x_N + \sum_{k=0}^{N-1} \left(x_k^T Q x_k + u_k^T R u_k + 2x_k^T N u_k \right)$$

the optimal control sequence minimizing the performance index is given by:

$$u_k = -F_k x_k$$

where:

$$F_k = (R + B^T P_{k+1} B)^{-1} (B^T P_{k+1} A + N^T)$$

Feedback Controller for "Extend"

and P_k is found iteratively backwards in time by the dynamic Riccati equation:

$$P_{k-1} = A^T P_k A - (A^T P_k B + N)\left(R + B^T P_k B\right)^{-1} (B^T P_k A + N^T) + Q$$

Cost-to-Go as "Distance Metric"

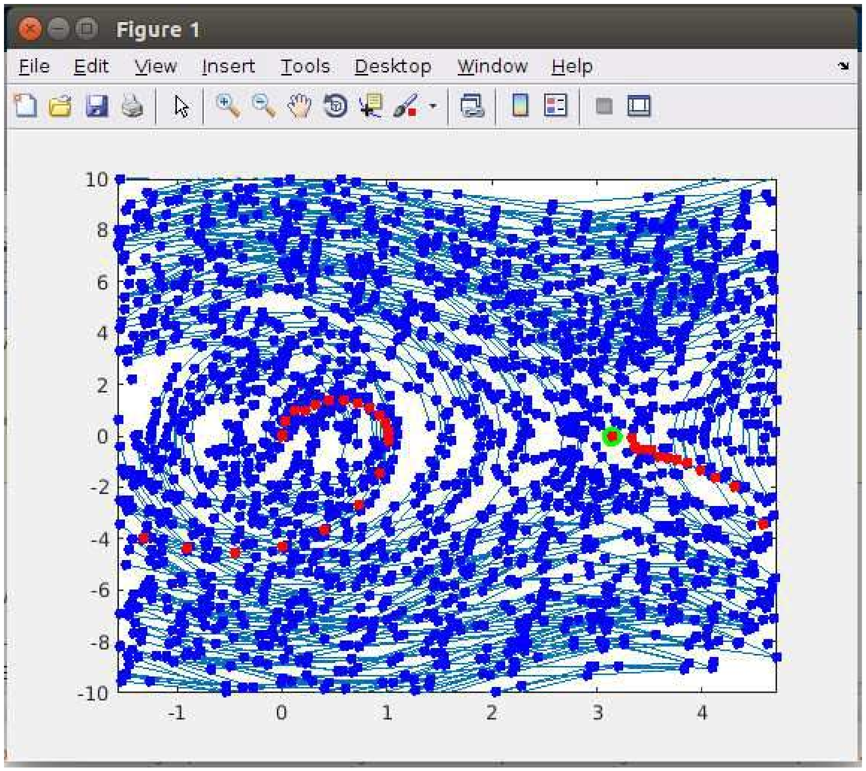
from terminal condition $P_N = Q$. Note that u_N is not defined, since x is driven to its final state x_N by $Ax_{N-1} + Bu_{N-1}$.

Bellman Updates

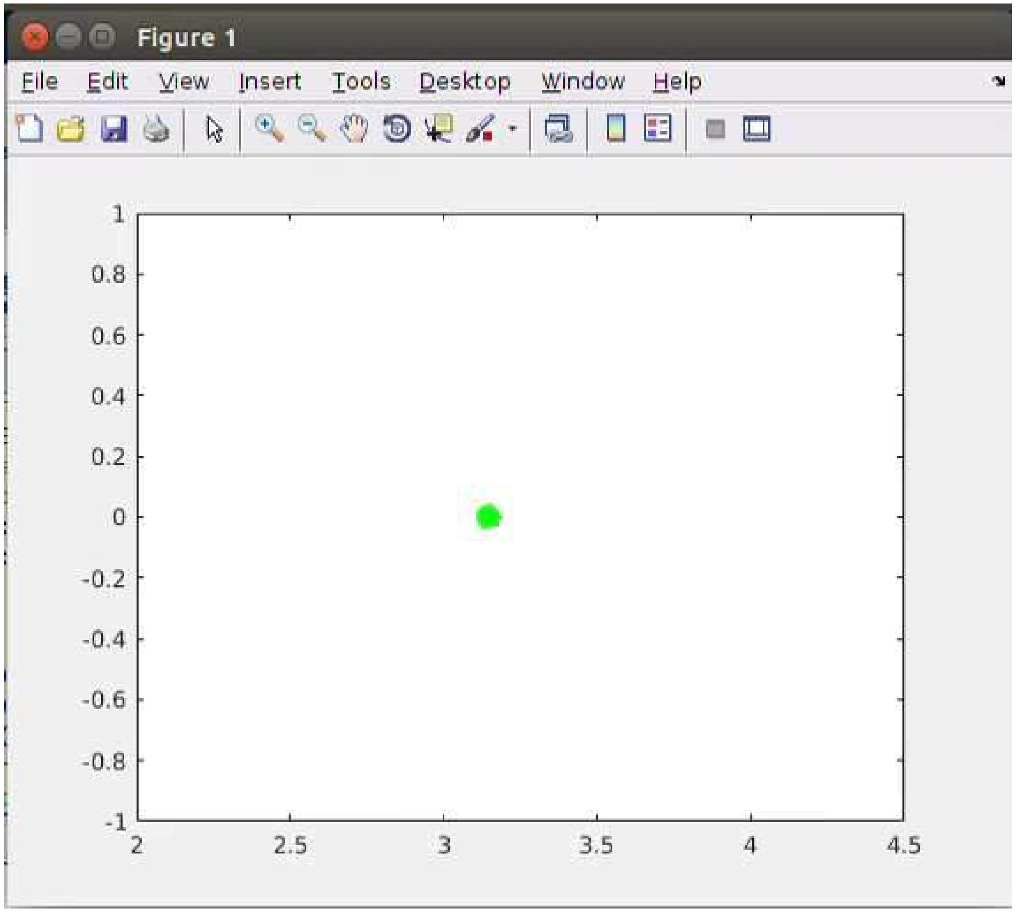
$$V_N(x_N) = c(x_N, u_N)$$

$$V_{k+1}(x) = \min_a c(x,u) + V_k\big(f(x,u)\big)$$

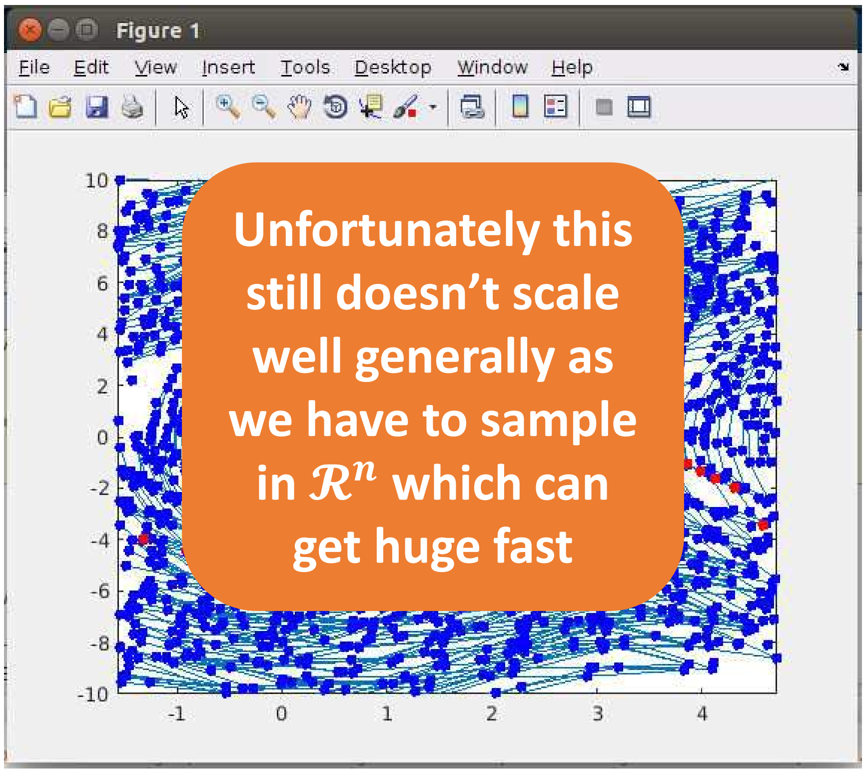
6 We can also use LQR in RRT as a better metric of "distance" and the feedback controller as the best "extend"

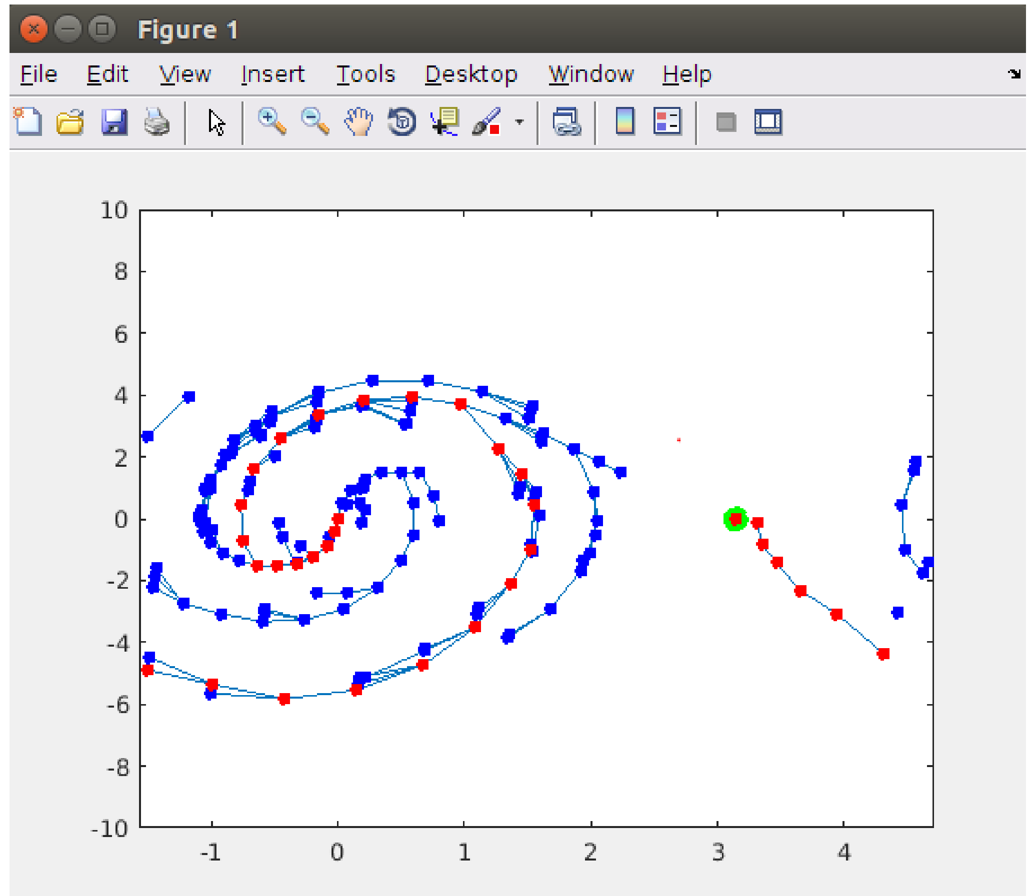


[Perez et. al. LQR-RRT*]



6 We can use LQR in RRT as a better metric of "distance" and the feedback controller as the best "extend"





[Perez et. al. LQR-RRT*]

6 So what have we learned so far?

1. Real world autonomous systems need to use **Feedback Control**
2. **PID** controllers are simple and effective but require **gain tuning**
3. **LQR** controllers allow for **cost function design** instead
4. PID and LQR require a plan to already exist and are simply **tracking controllers**

6 So what have we learned so far?

1. Real world autonomous systems need to use **Feedback Control**
2. **PID** controllers are simple and effective but require **gain tuning**
3. **LQR** controllers allow for **cost function design** instead
4. PID and LQR require a plan to already exist and are simply **tracking controllers**

- But what happens if we **deviate** so much from our original plan that it is **no longer valid**? How do we initiate **re-plans**?

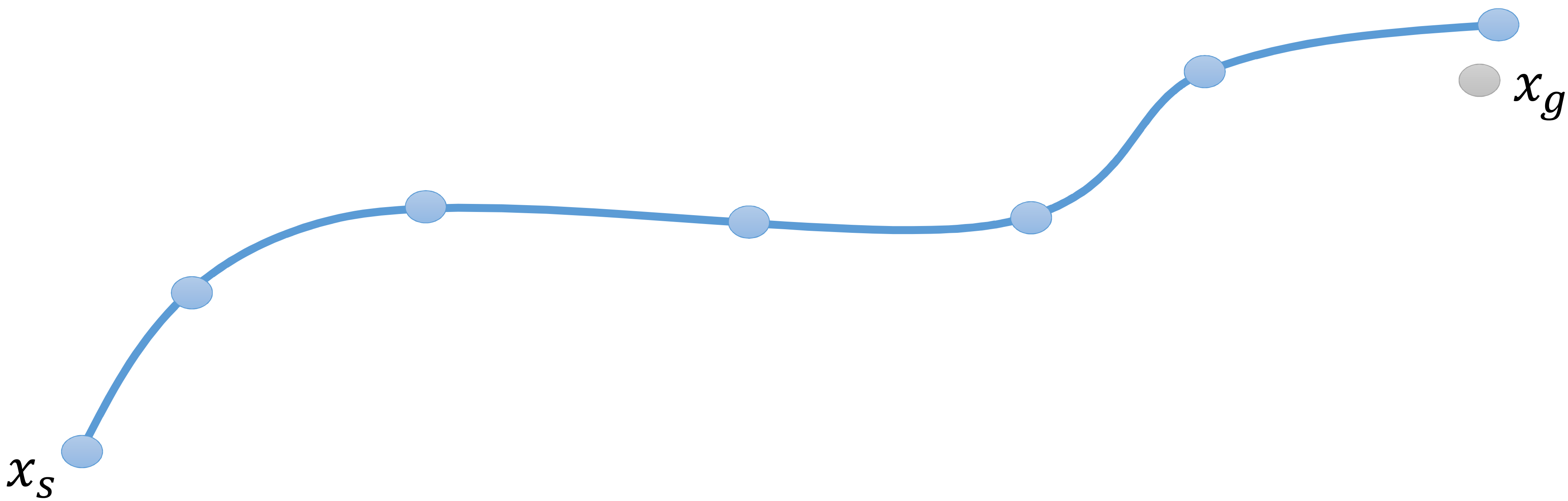
6 So what have we learned so far?

1. Real world autonomous systems need to use **Feedback Control**
2. **PID** controllers are simple and effective but require **gain tuning**
3. **LQR** controllers allow for **cost function design** instead
4. PID and LQR require a plan to already exist and are simply **tracking controllers**

- But what happens if we **deviate** so much from our original plan that it is **no longer valid**? How do we initiate **re-plans**?

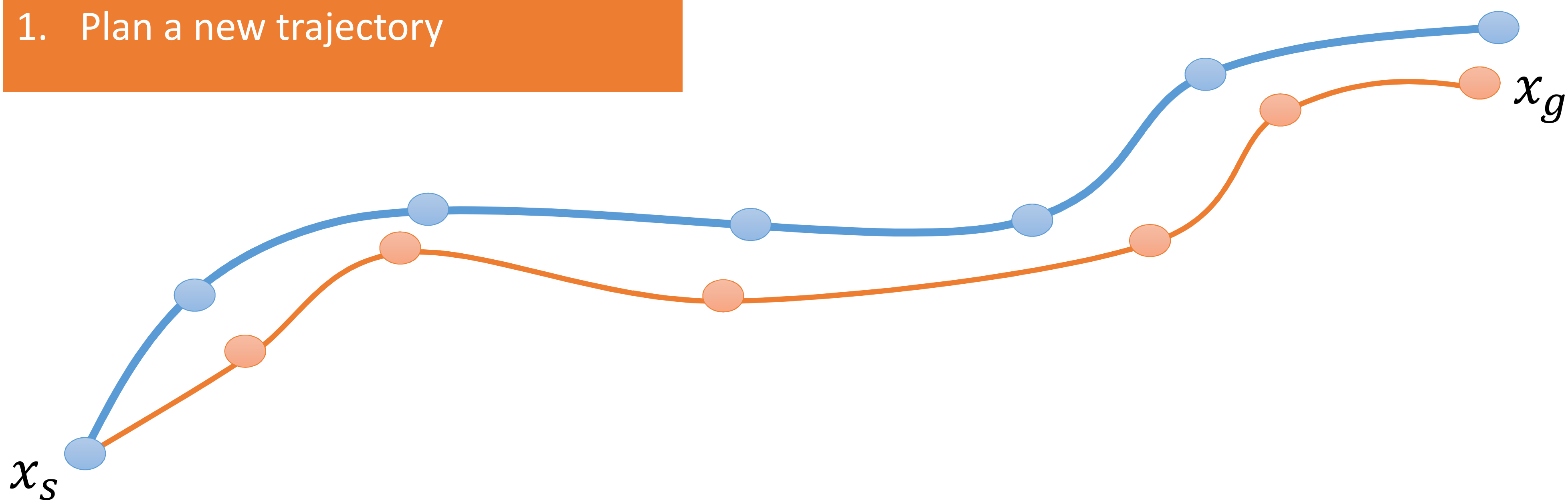
This is an open unsolved problem!

6 Model Predictive Control: re-planning fast enough that the plan becomes the controller!



6 Model Predictive Control: re-planning fast enough that the plan becomes the controller!

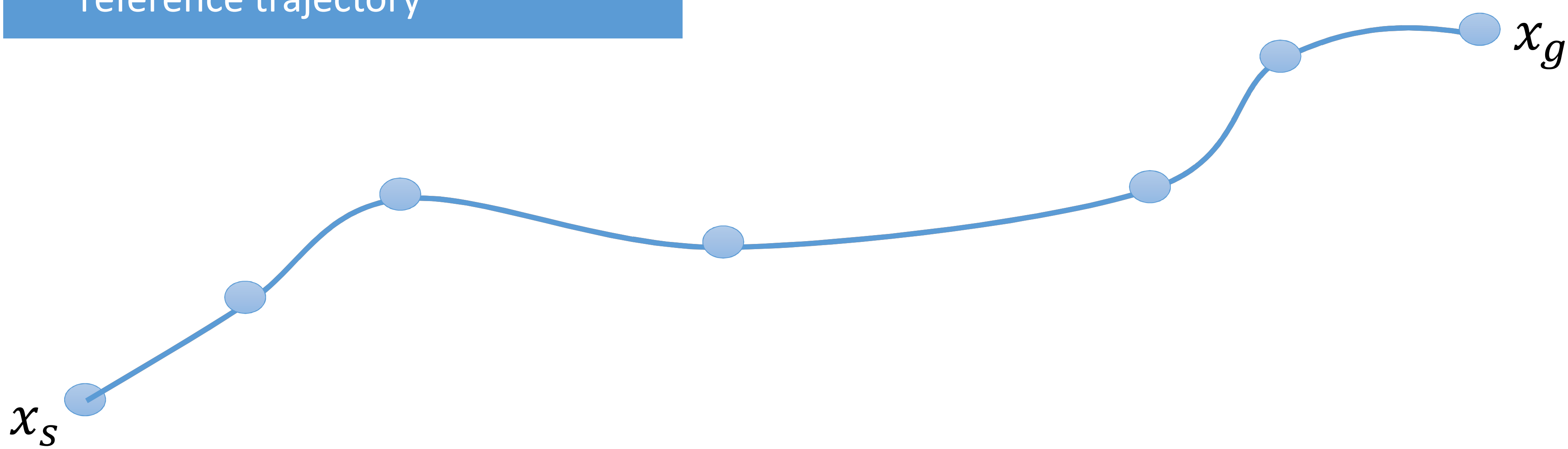
1. Plan a new trajectory



6

Model Predictive Control (MPC): re-planning fast enough that the plan becomes the controller!

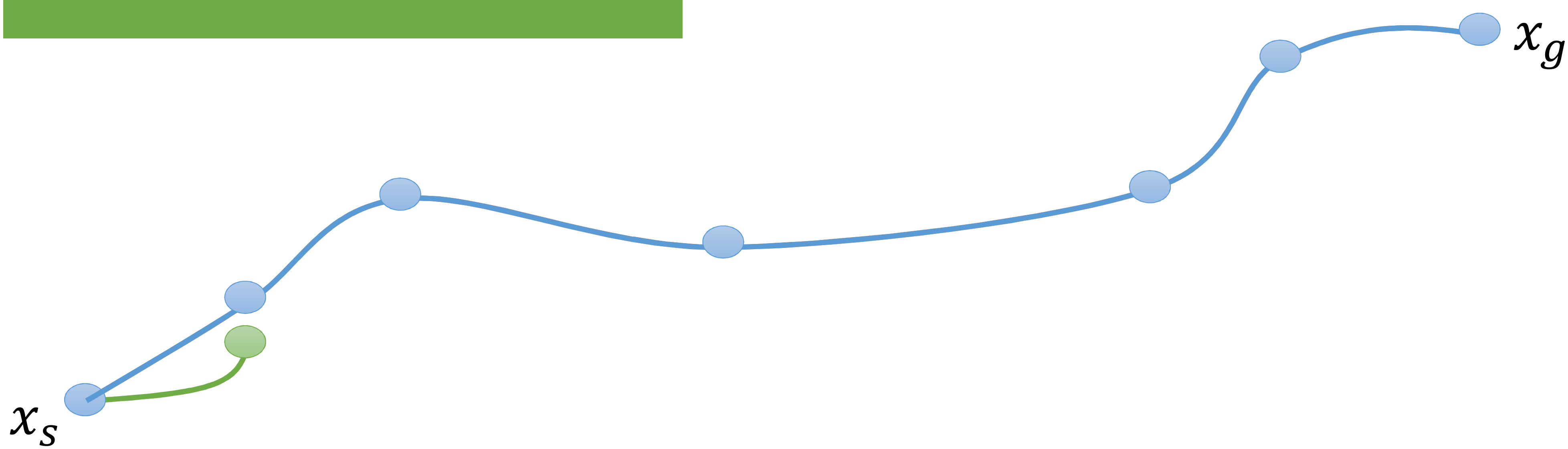
2. The new plan becomes the reference trajectory



6

Model Predictive Control (MPC): re-planning fast enough that the plan becomes the controller!

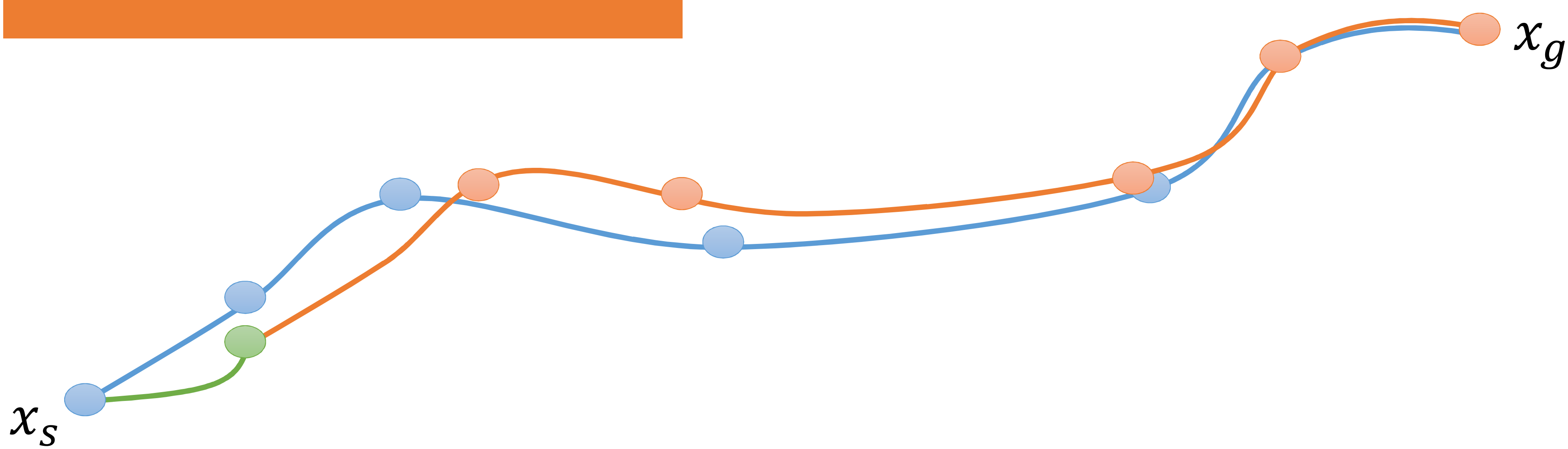
3. Execute the first step of the plan



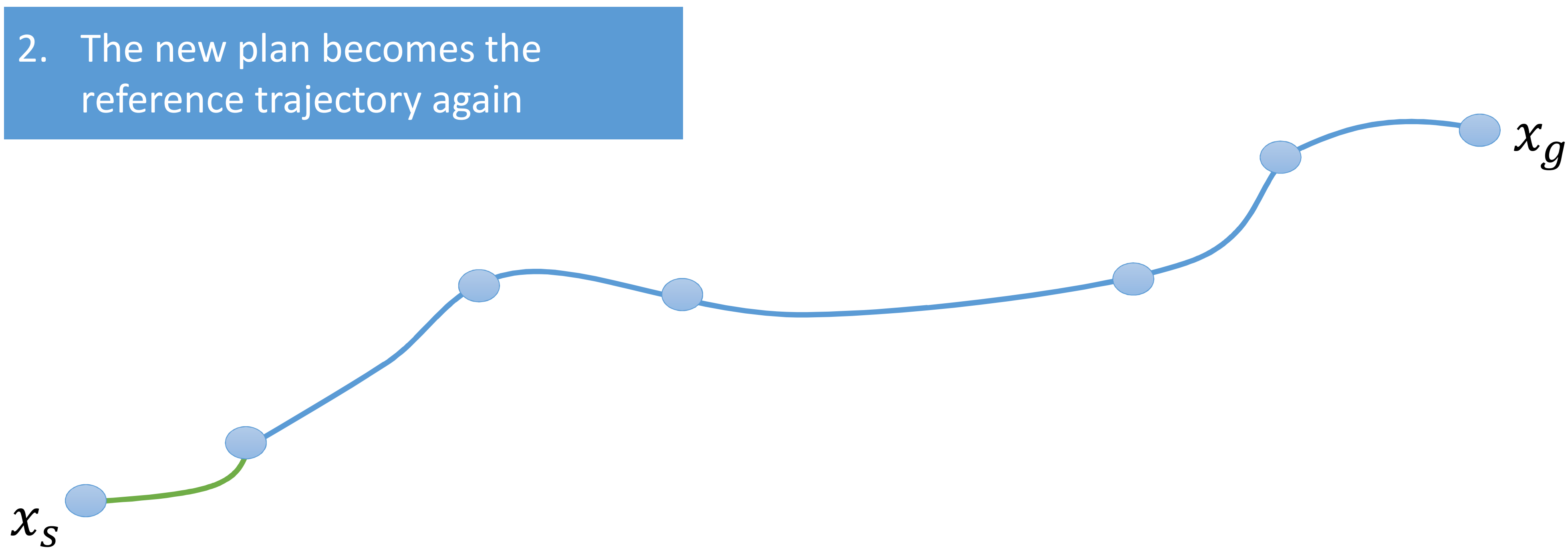
6

Model Predictive Control (MPC): re-planning fast enough that the plan becomes the controller!

1. Re-plan based on that step



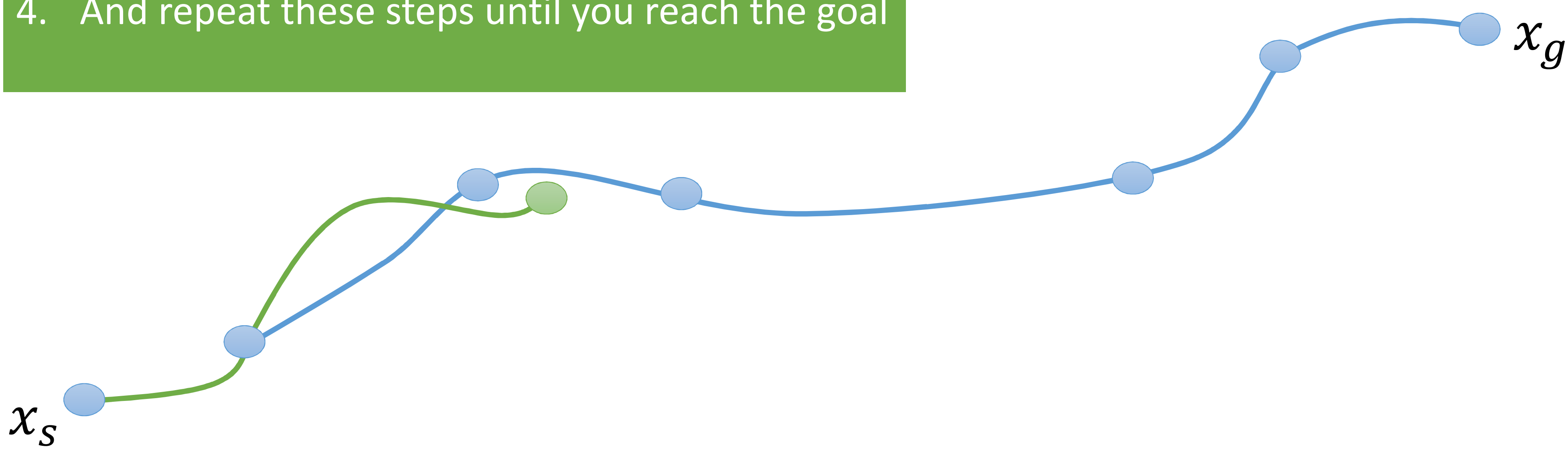
❻ Model Predictive Control (MPC): re-planning fast enough that the plan becomes the controller!



6

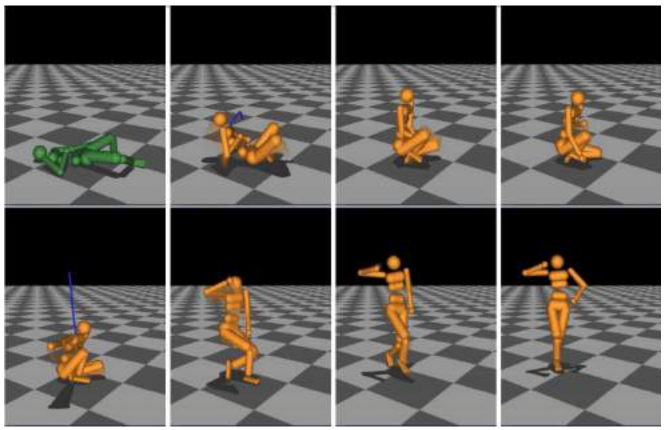
Model Predictive Control (MPC): re-planning fast enough that the plan becomes the controller!

3. Execute the first step of the new plan again
4. And repeat these steps until you reach the goal

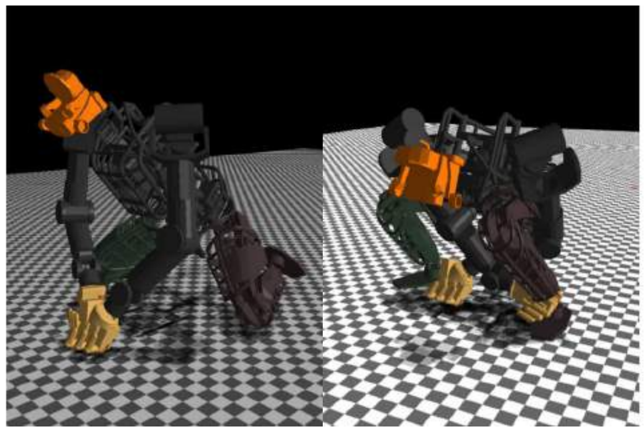


6

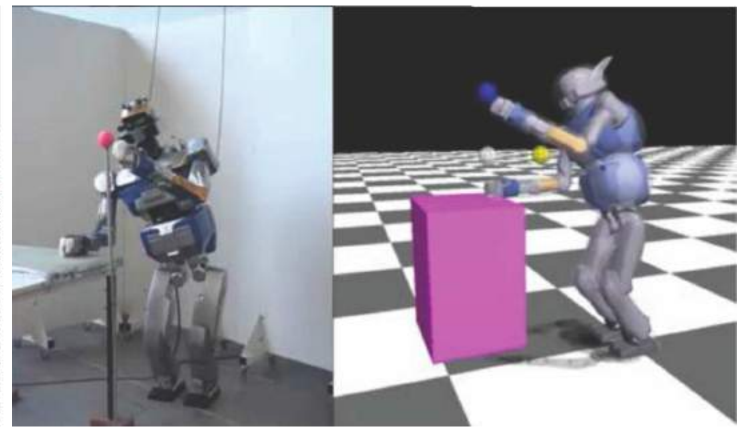
Recently MPC has been used in a variety of complex autonomous systems in simulation and on physical robots



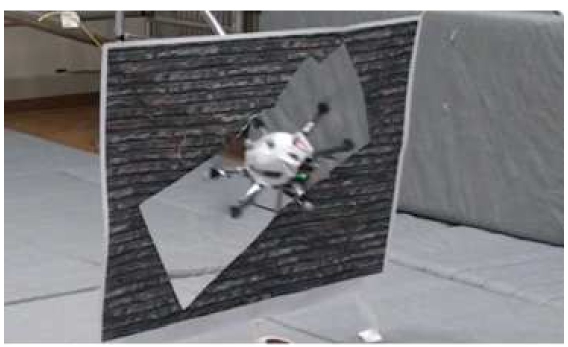
[Tassa et. al. IROS 2012]



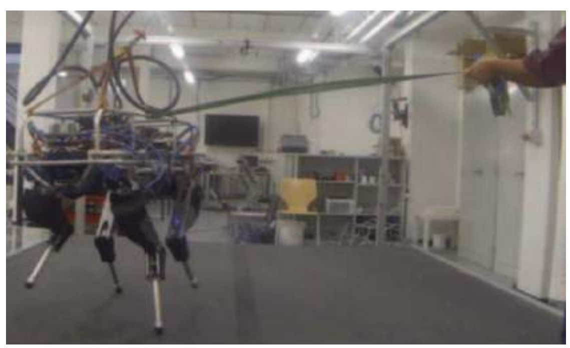
[Erez et. al. Humanoids 2013]



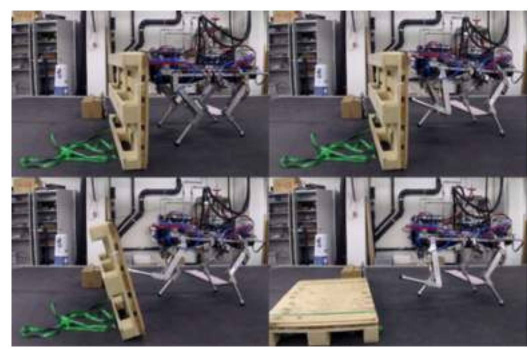
[Koenemann et. al. IROS 2015]



[Neunert et. al. ICRA 2016]



[Neunert et. al. Humanoids 2017]

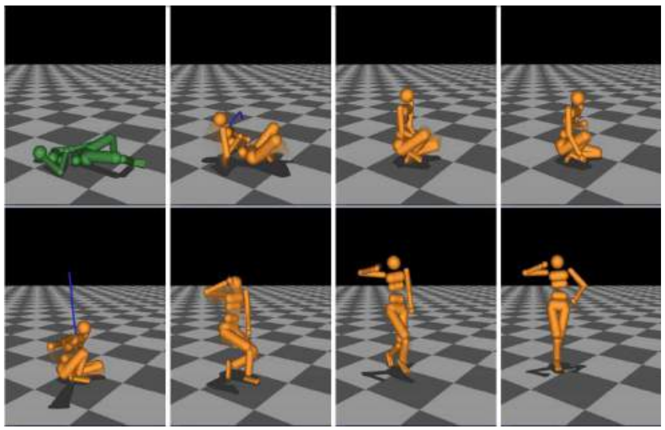


[Farshidian et. al. IEEE RAL 2017]

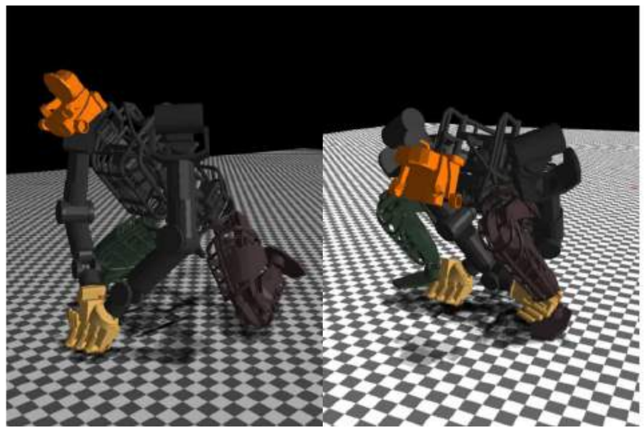
[Plancher et. al. WAFR 2018]
[Plancher et. al. ICRA 2019]

Recently MPC has been used in a variety of complex autonomous systems in simulation and on physical robots

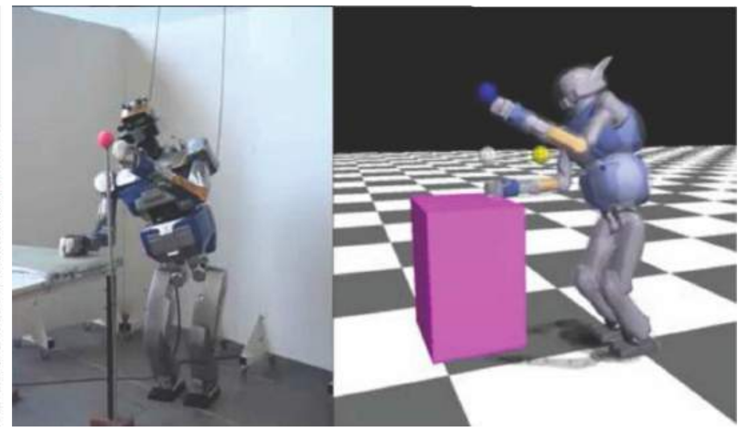
6



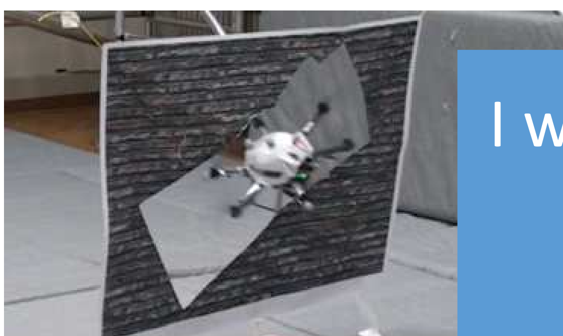
[Tassa et. al. IROS 2012]



[Erez et. al. Humanoids 2013]



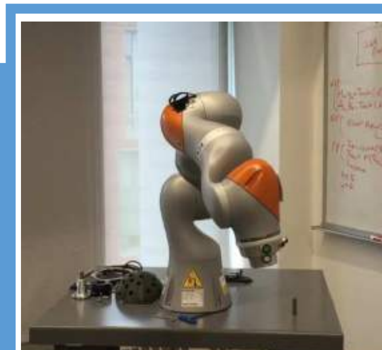
[Koenemann et. al. IROS 2015]



[Neunert et. al. ICRA 2016]

[Neunert et. al. Humanoids 2017]

[Farshidian et. al. IEEE RAL 2017]



[Plancher et. al. WAFR 2018]
[Plancher et. al. ICRA 2019]

I will go into far more detail on this when I present my recent work during the sample paper presentations!

6 Practical Challenges for Control: Contact

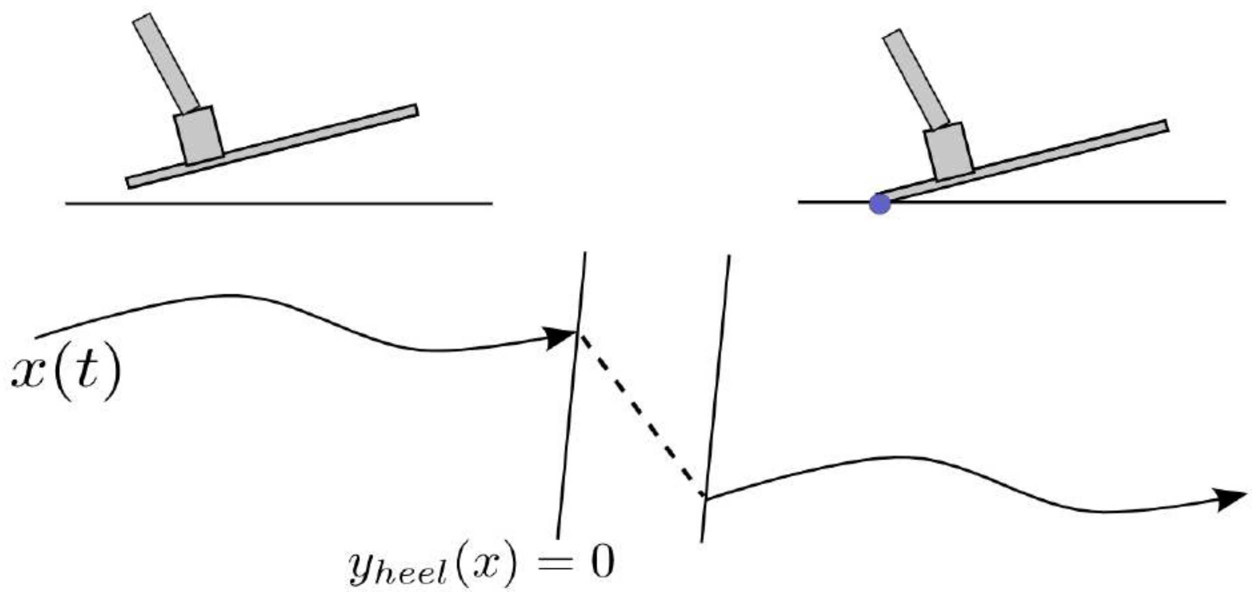


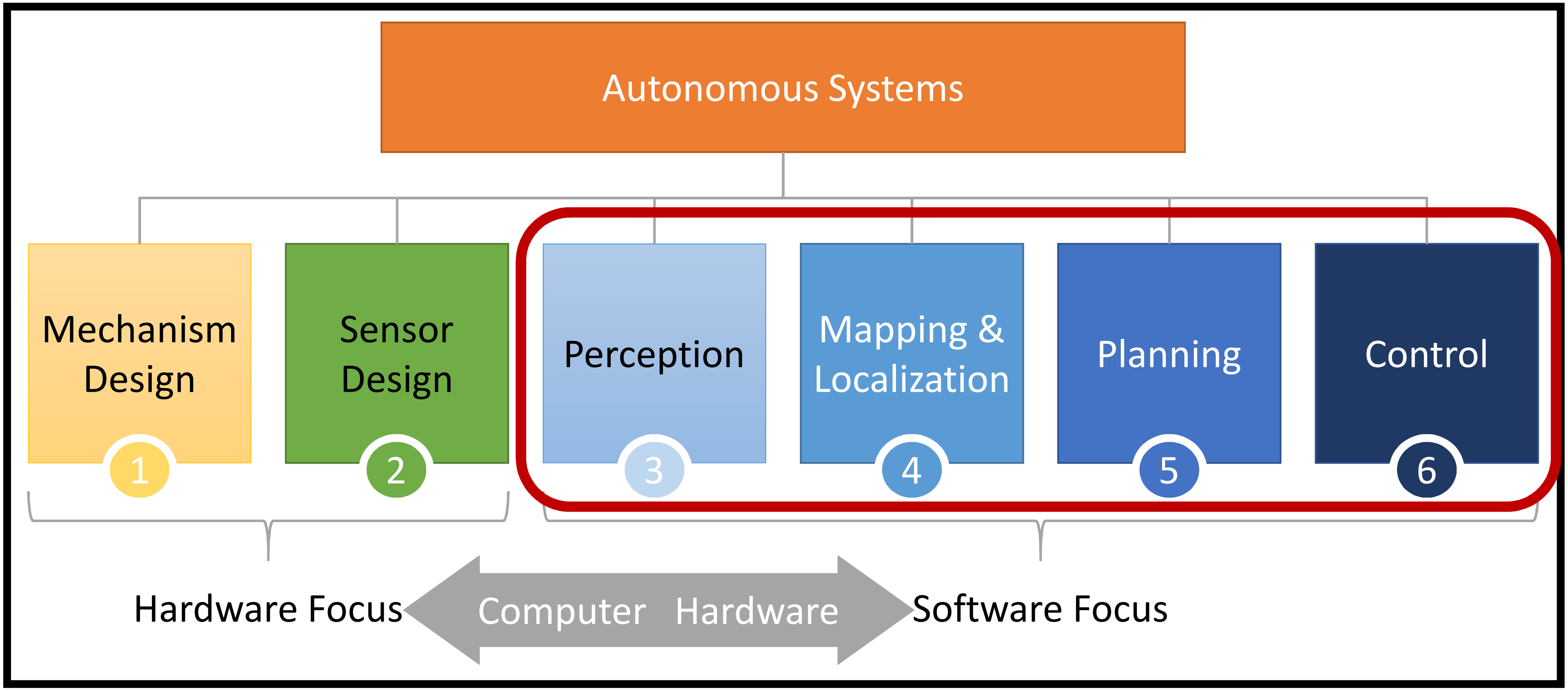
Figure 17.1 - Modeling contact as a hybrid system.

6 Practical Challenges for Control: Contact

6 Key Takeaways:

1. Real world autonomous systems need to use Feedback Control
2. Tracking controllers allow for simple control design and are quite effective in practice. Two common controllers are:
    1. PID with gain tuning
    2. LQR with cost function design
3. Using MPC allows for the planner to be the controller which enables more sophisticated control strategies
4. Contact is really hard!

Autonomous Systems / Robotics is a BIG space



Key Takeaways:

1. NNs running on accelerator chips solve most perception problems
2. The Kalman/Particle Filter uses probability to solve the localization problem but modeling and/or approximations are needed to run online
3. Mapping quickly becomes a memory storage problem
4. Stereo Depth and Visual Odometry also need acceleration to run online
5. Robot planning involves both task and configuration spaces
6. Collision checking can be expensive
7. Sample Based Planners (PRM, RRT, RRT*) leverage random search and are probabilistically complete but do not scale well to high dimensions
8. Trajectory Optimization finds locally optimal paths but is not complete or robust and (often) solved with (slow) off the shelf solvers
9. Tracking controllers (PID, LQR) work well in practice but MPC is a much more powerful (and computationally expensive) approach
10. Contact is hard and we (sometimes) use simpler models for tractability

Key Takeaways:

1. NNs running on accelerator chips solve most perception problems
2. The Kalman/Particle Filter uses probability to solve the localization problem but modeling and/or approximations are needed to run online
3. Mapping qu
4. Stereo Dept … to run online
5. Robot plann … es
6. Collision ch
7. Sample Bas … search and are probabi … dimensions
8. Trajectory Optimization finds locally optimal paths but is not complete or robust and (often) solved with (slow) off the shelf solvers
9. Tracking controllers (PID, LQR) work well in practice but MPC is a much more powerful (and computationally expensive) approach
10. Contact is hard and we (sometimes) use simpler models for tractability

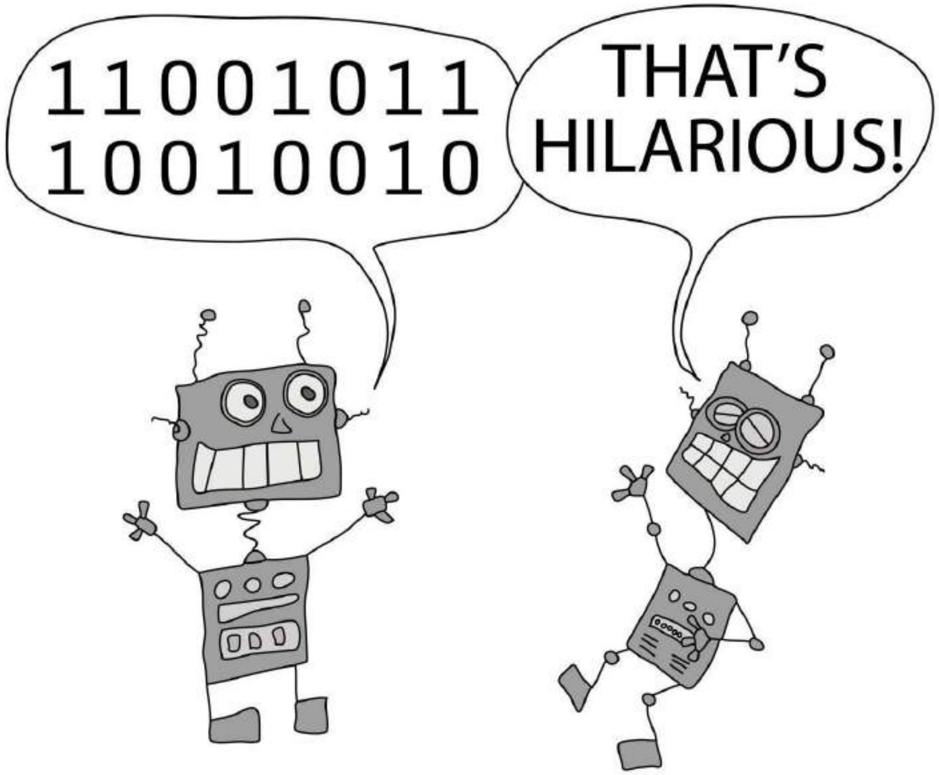
There is SO much room for acceleration!!!!

And that’s everything!

http://bit.ly/CS249-Feedback-L2

CS 249r: Special Topics in Edge Computing

Intro to Autonomous Systems / Robotics Wrap-Up

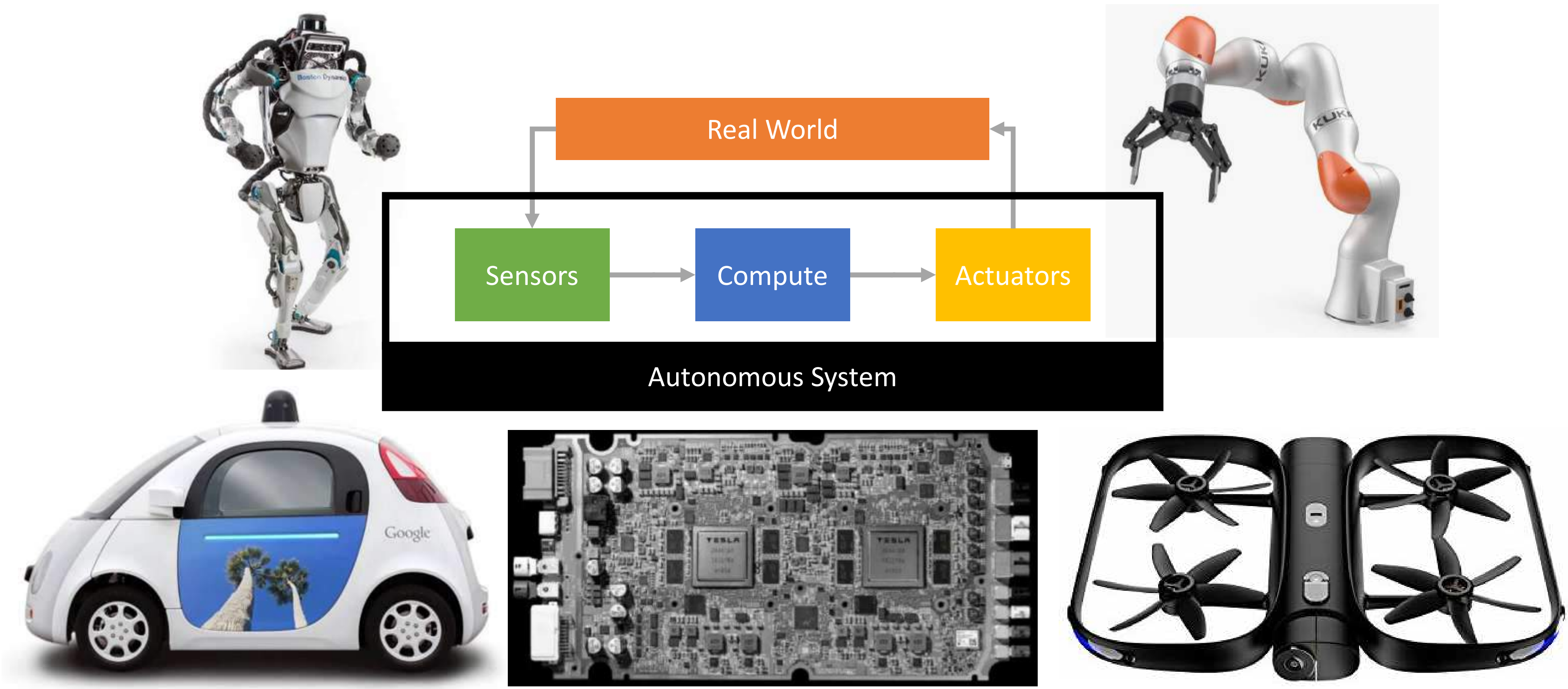


Brian Plancher
Fall 2019

The goal for the next couple of lectures is to develop a high level understanding of:

1. What is an autonomous system
2. Key problems and constraints for autonomous systems
3. Some of the most important (classes of) algorithms in robotics
   A. The model based vs. model free tradeoff
   B. The online vs offline tradeoff
   C. The no free lunch theorem and the need for approximations
4. How computer systems / architecture design has and can play a role in improving autonomous systems

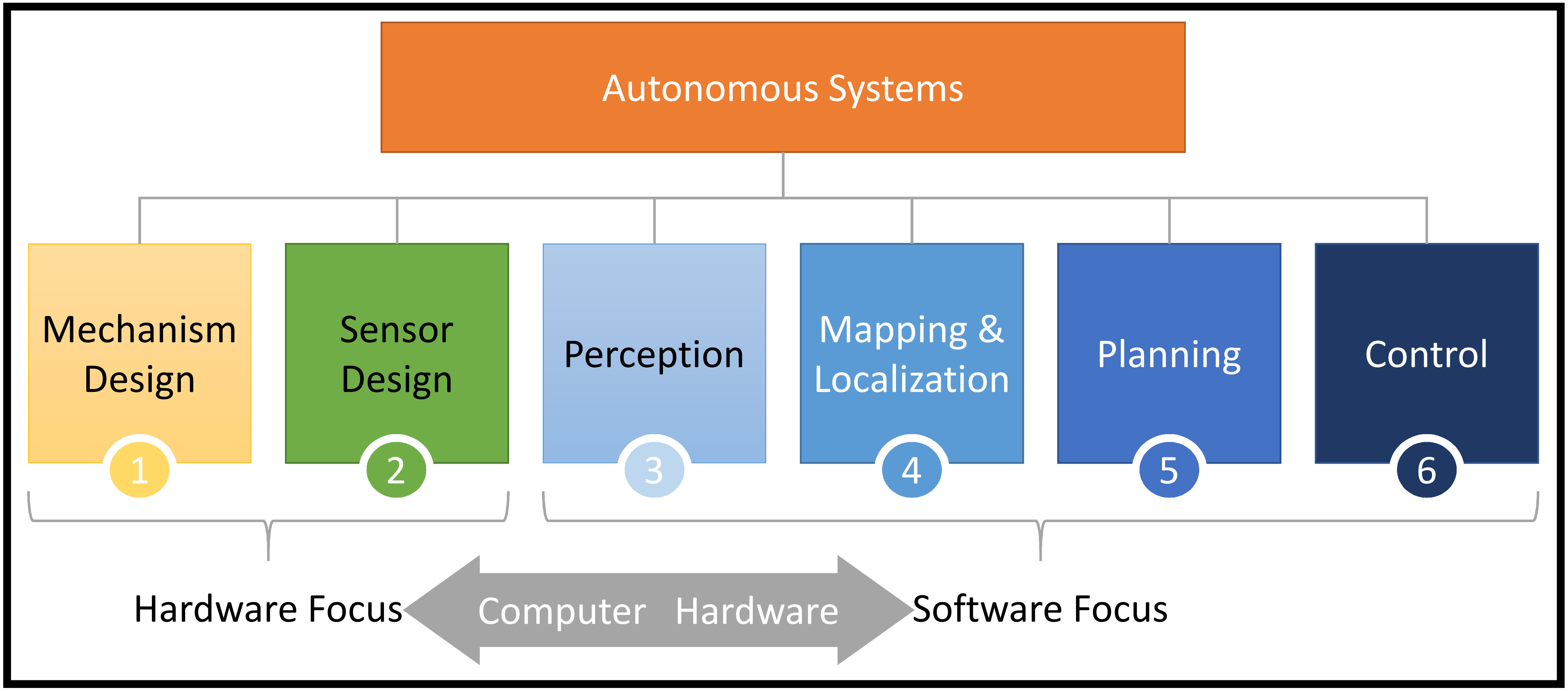
What do we mean by an Autonomous System?



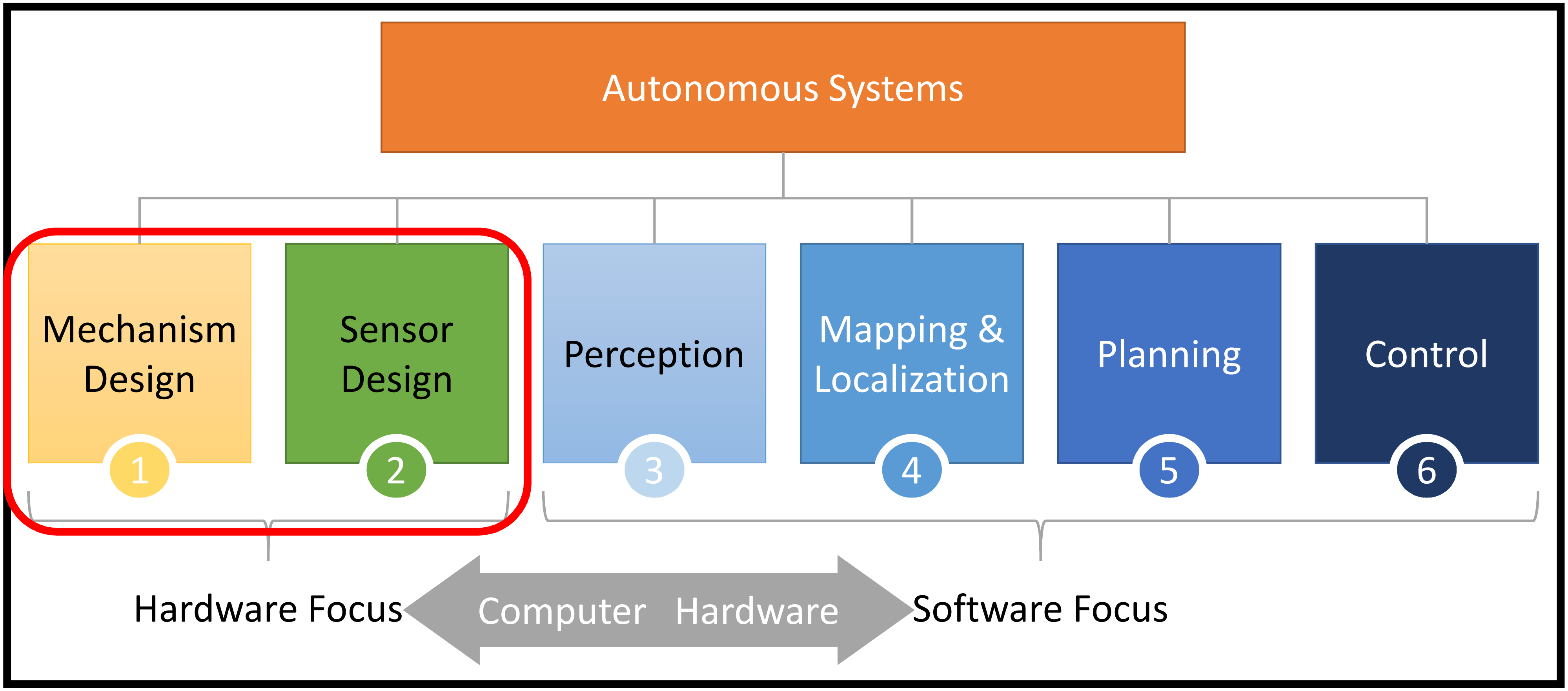
The goal for the next couple of lectures is to develop a high level understanding of:

1. What is an autonomous system
2. Key problems and constraints for autonomous systems
3. Some of the most important (classes of) algorithms in robotics
   A. The model based vs. model free tradeoff
   B. The online vs offline tradeoff
   C. The no free lunch theorem and the need for approximations
4. How computer systems / architecture design has and can play a role in improving autonomous systems

Autonomous Systems / Robotics is a BIG space



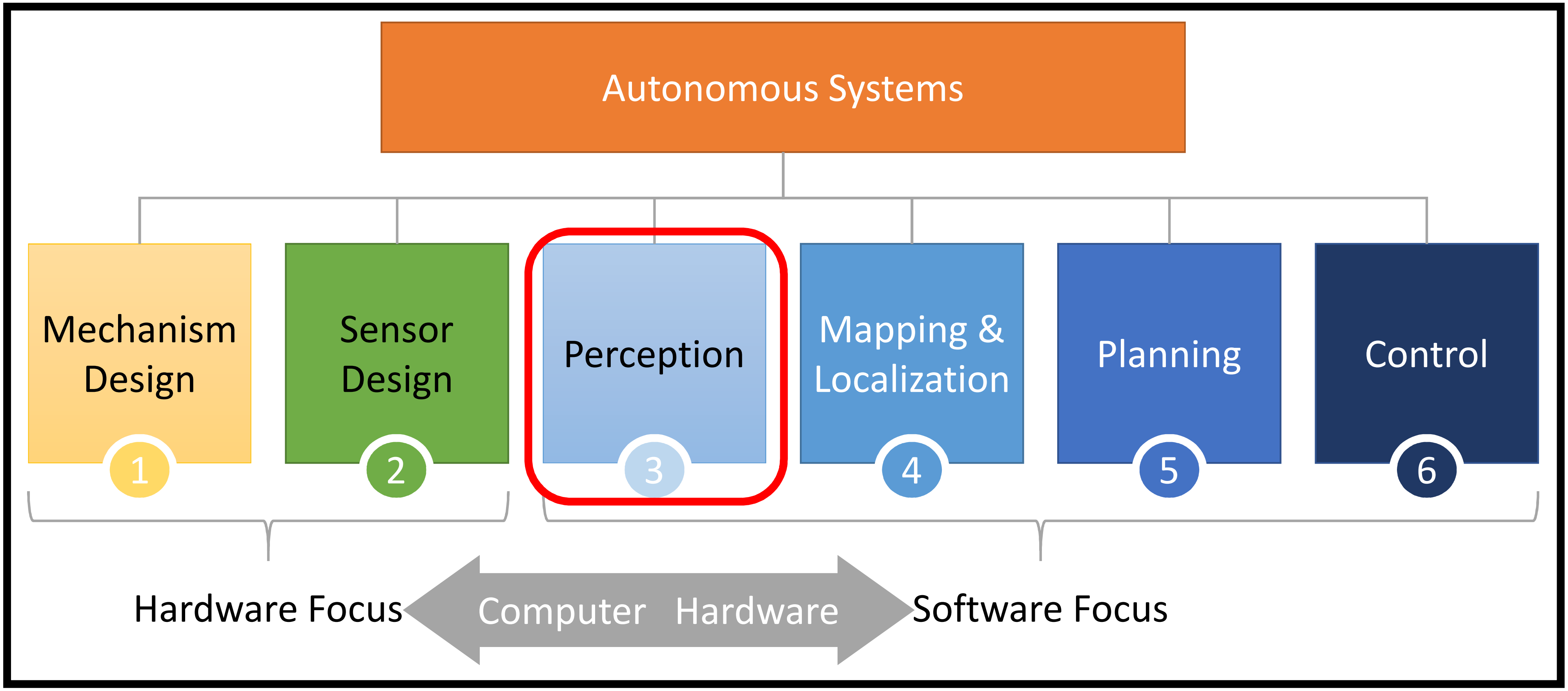
Autonomous Systems / Robotics is a BIG space



1 2 Key Takeaways:

1. When designing algorithms for robots you need to understand the physical capabilities of the robot and you (potentially) need to understand how to model its physical behaviors
2. Different kinds of systems will have different power, weight, and performance budgets for computer hardware

Autonomous Systems / Robotics is a BIG space

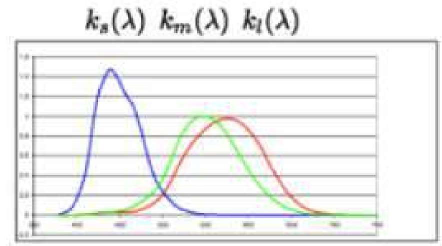


Computer Vision (and Perception in general) is hard

Retinal color

$$\mathbf{c}(\ell(\lambda)) = (c_s, c_m, c_l)$$

$$c_s = \int k_s(\lambda)\ell(\lambda)d\lambda$$

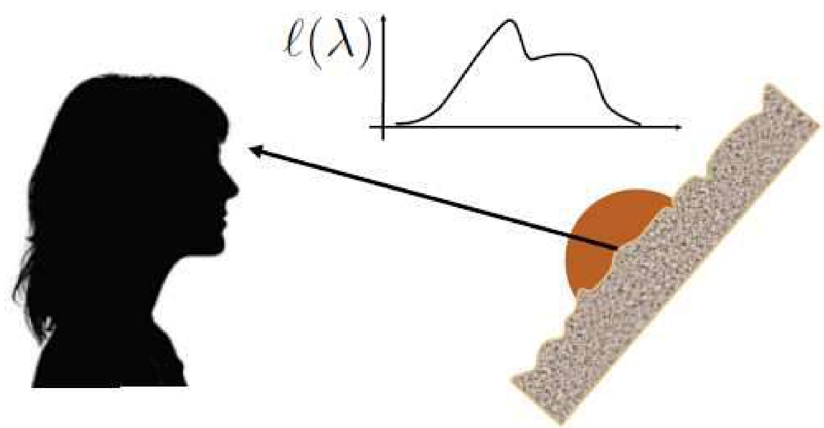


LMS senstivity functions

Perceived color

Object color

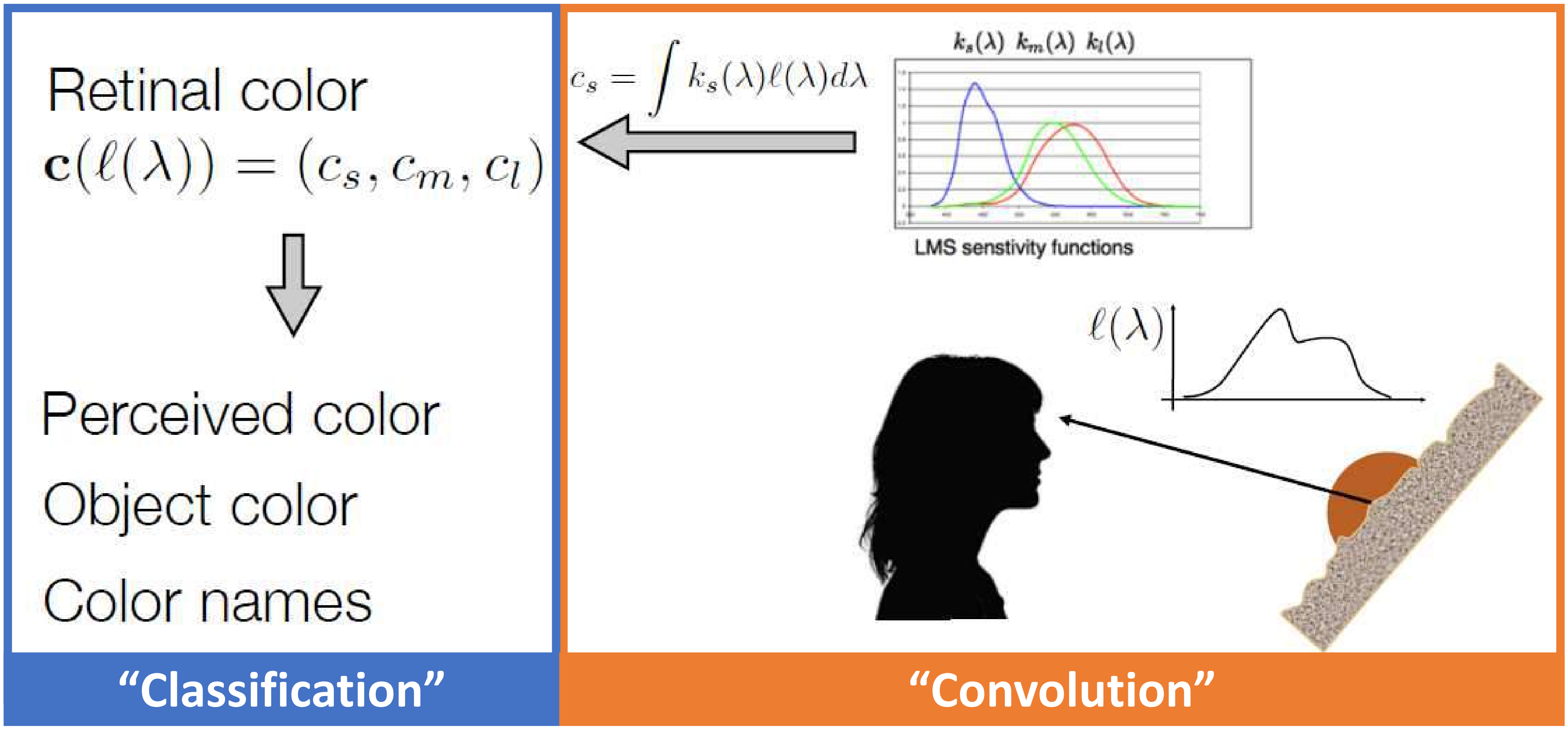
Color names



Slide Credit: Todd Zickler CS 283

3

CV/Perception is solved by modeling and approximating the classification of convolution



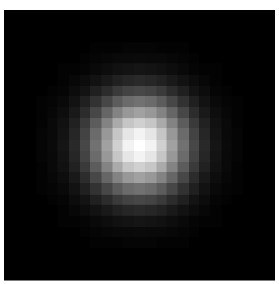
Slide Credit: Todd Zickler CS 283

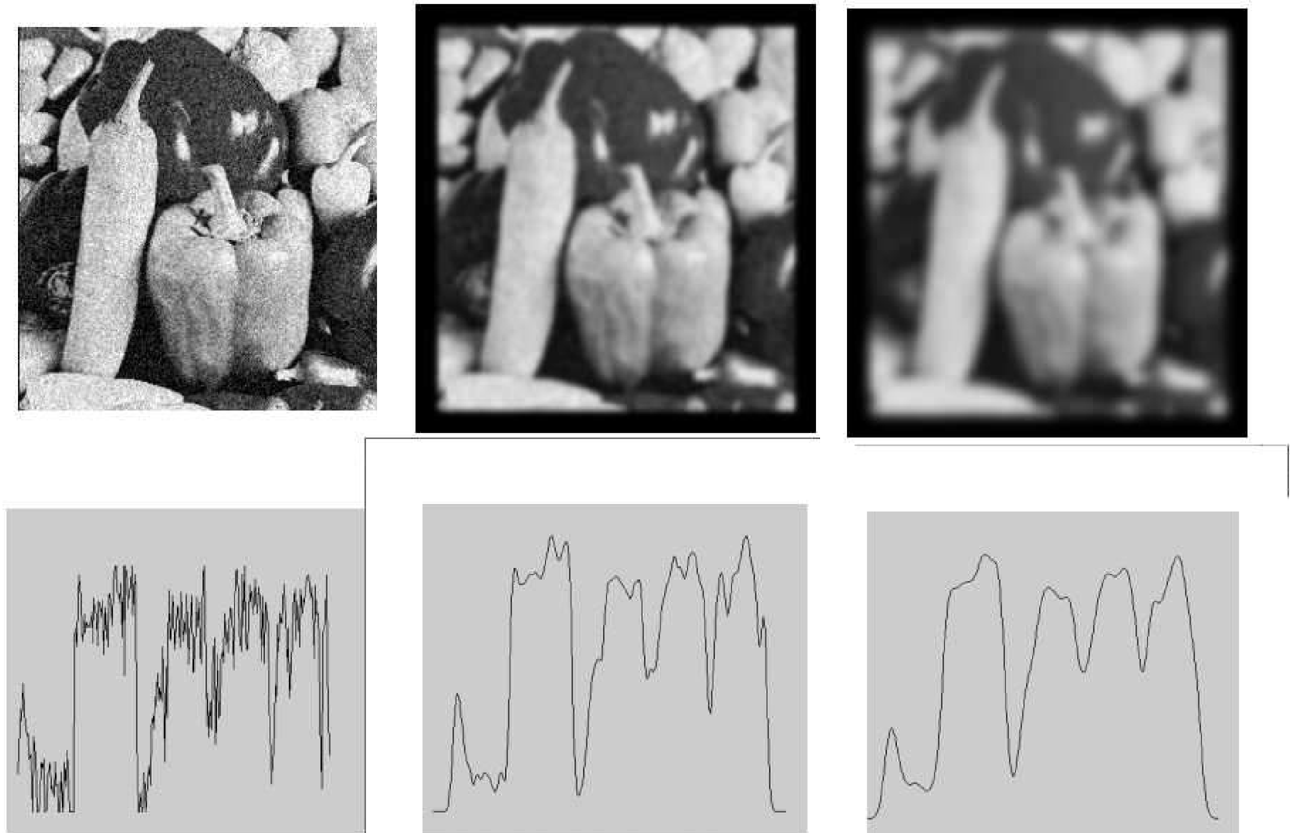
3 We approximate convolution using linear filters

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$





No smoothing

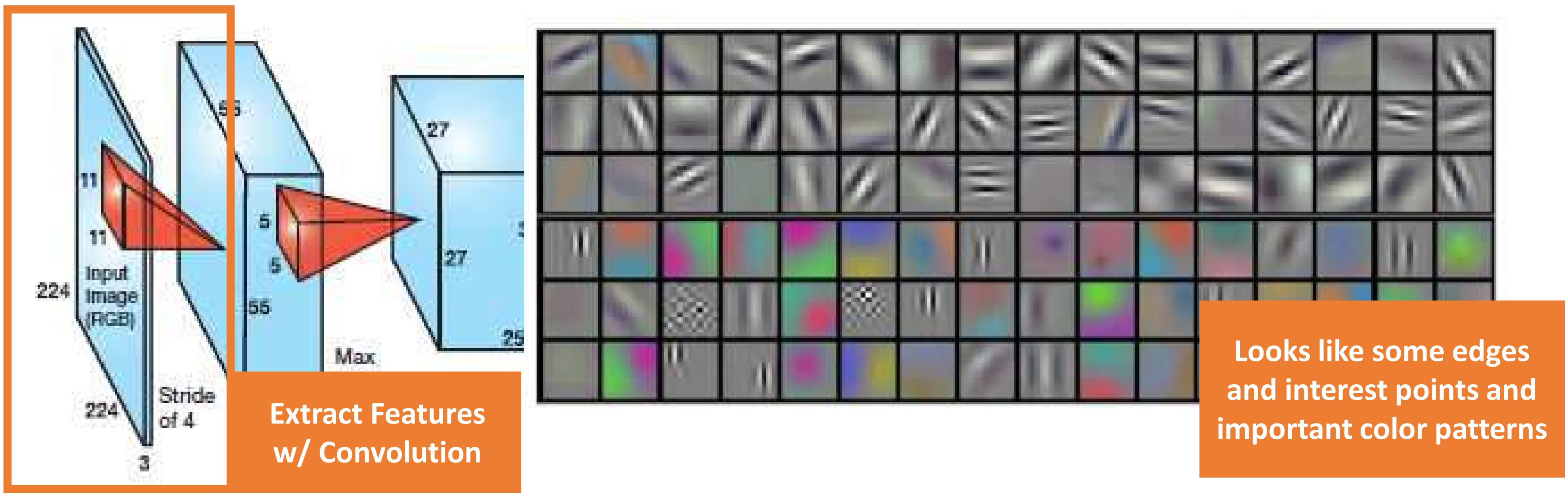
$\sigma = 2$

$\sigma = 4$

3

Deep learning automates the design of filters, and the selection/combination of features for classification

AlexNet: the first widely successful application of deep learning

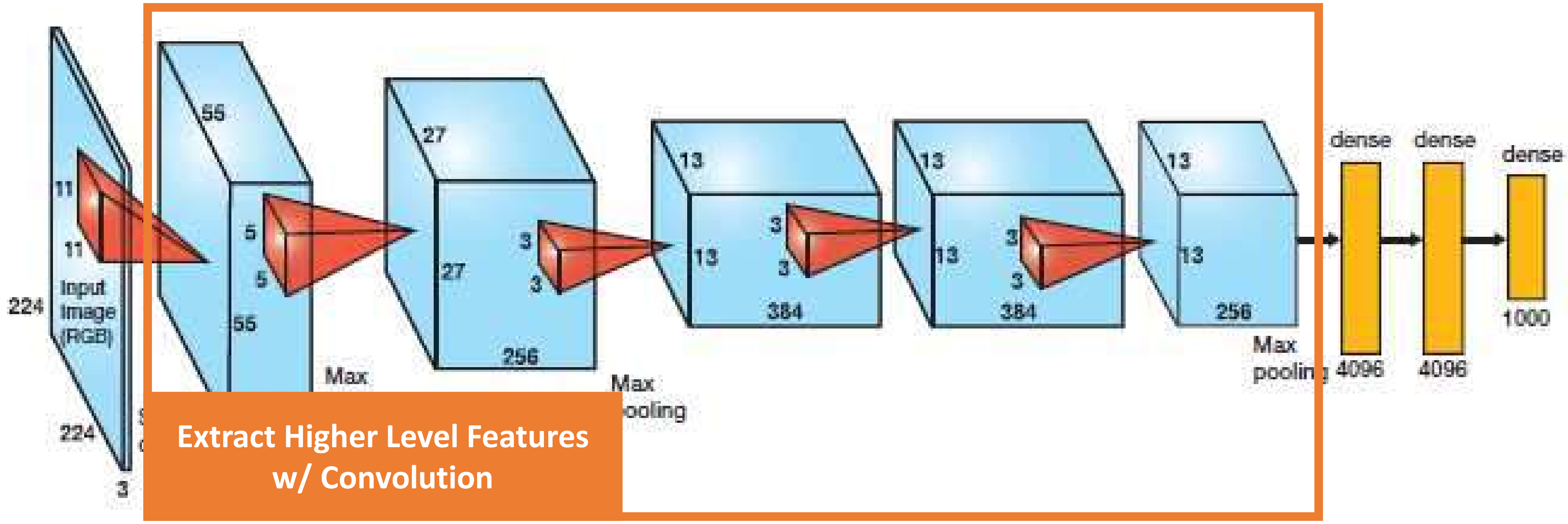


https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

3

Deep learning automates the design of filters, and the selection/combination of features for classification

AlexNet: the first widely successful application of deep learning

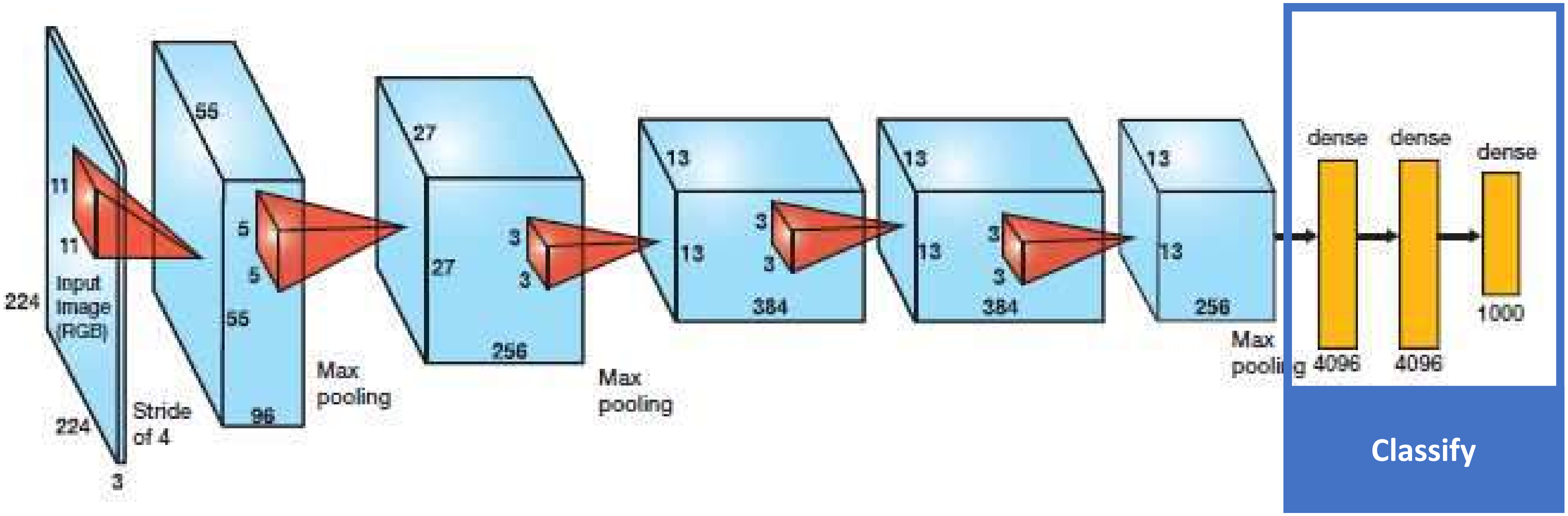


https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

3

Deep learning automates the design of filters, and the selection/combination of features for classification

AlexNet: the first widely successful application of deep learning



https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

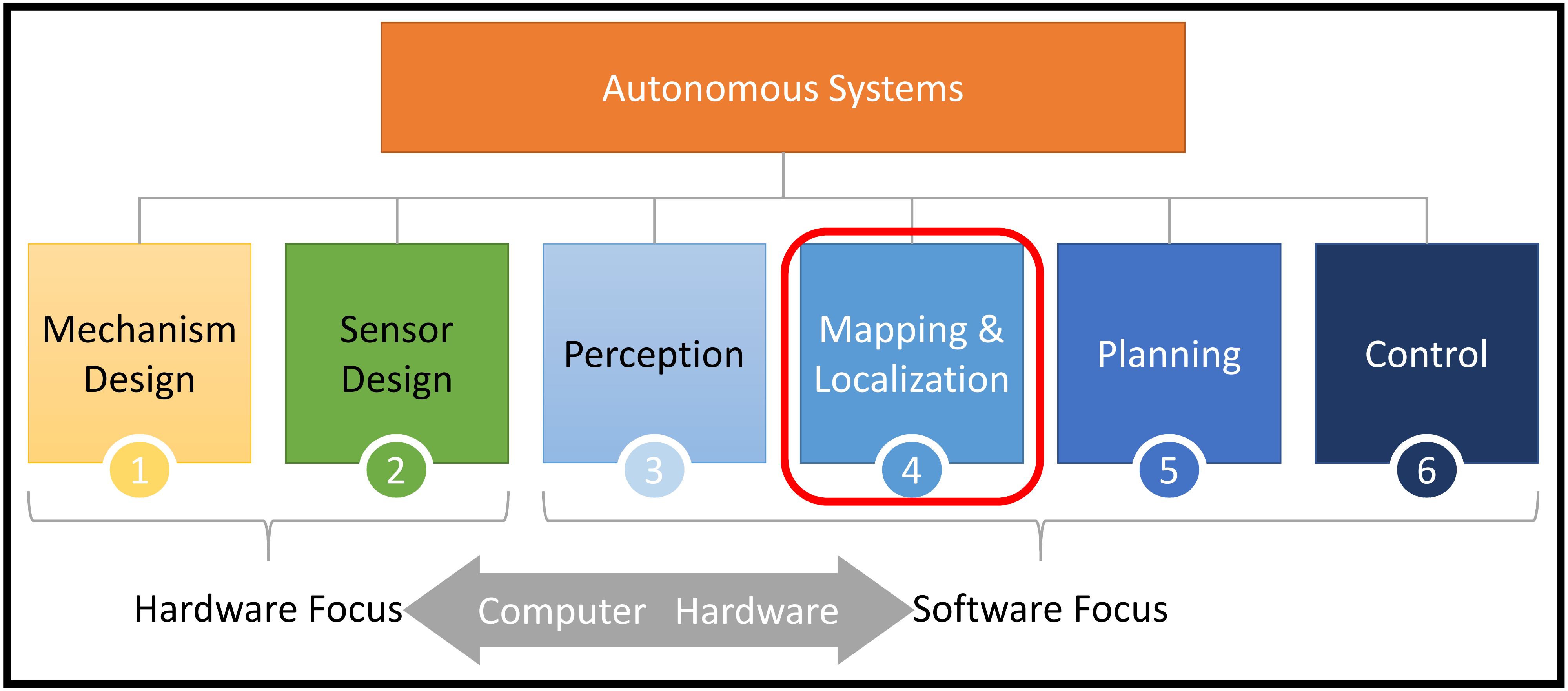
3

Deep learning automates the design of filters, and the selection/combination of features for classification

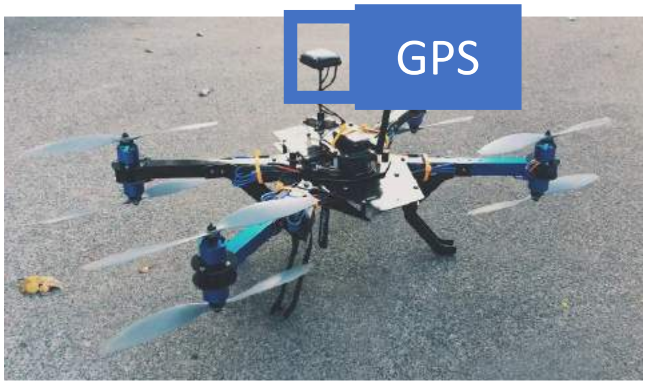
But watch our for adversarial attacks on the math!

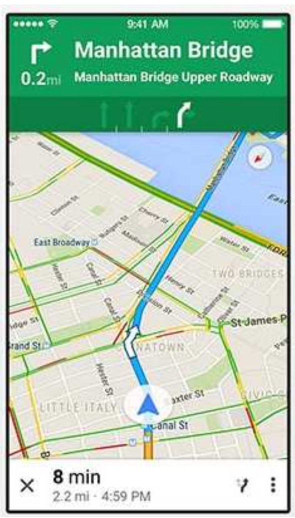
"No Free Lunch!"

Autonomous Systems / Robotics is a BIG space



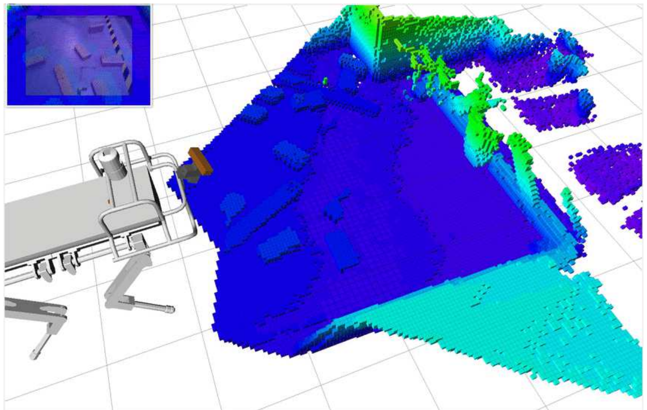
4 Mapping/Localization is hard

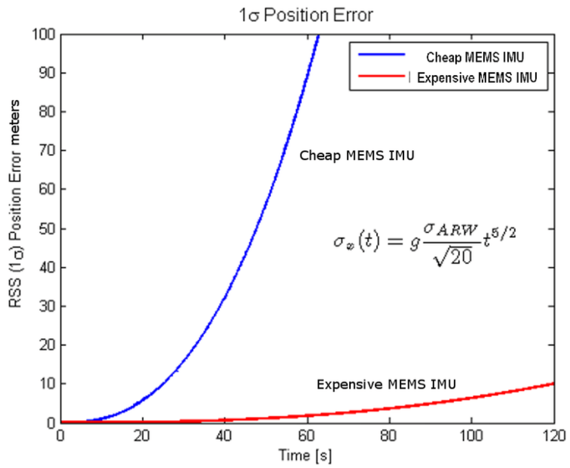




Three Problems

1. GPS is only accurate to O(10m)
2. GPS relies on already having a perfect map of the environment (unrealistic often)
3. Other sensor data is also quite noisy!





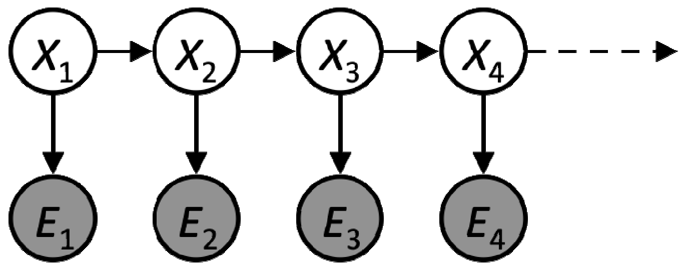
4 Mapping/Localization is solved by modeling the world as an HMM and using modeling and approximating to solve it

Track the **Belief State B_t** of the **state and landmarks**

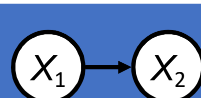
$$\boldsymbol{B_t = p(X_t|X_o, E_o \cdots E_{t-1})}$$

Hidden Markov Model (HMM)

States X update in time but we only observe the effects E

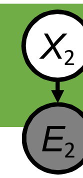


Time Update



$$P(x_{t+1}|x_0, e_1 \dots e_t) = \int P(x_t|x_0, e_1 \dots e_t) * P(x_{t+1}|x_t)$$

Evidence Update



$$P(x_{t+1}|x_0, e_1 \dots e_{t+1}) \propto \int P(x_{t+1}|x_0, e_1 \dots e_t) * P(e_{t+1}|x_{t+1})$$

4 Mapping/Localization is solved by modeling the world as an HMM and using modeling and approximating to solve it

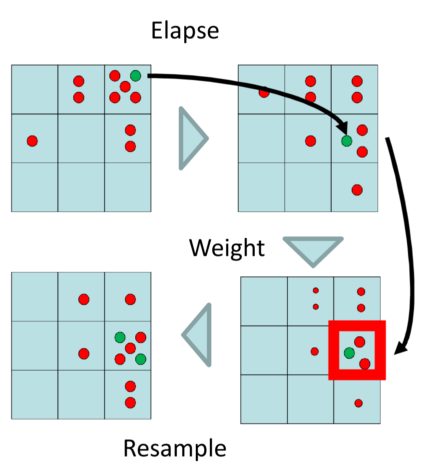
Time Update $X_1 \rightarrow X_2$

$$P(x_{t+1}|x_0, e_1 \dots e_t) = \int P(x_t|x_0, e_1 \dots e_t) * P(x_{t+1}|x_t)$$

Evidence Update $X_2 \rightarrow E_2$

$$P(x_{t+1}|x_0, e_1 \dots e_{t+1}) \propto \int P(x_{t+1}|x_0, e_1 \dots e_t) * P(e_{t+1}|x_{t+1})$$

Particle Filter



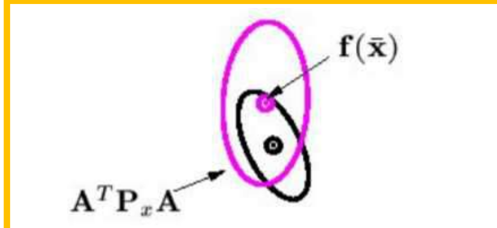
Approximate with Samples

Model With Gaussian

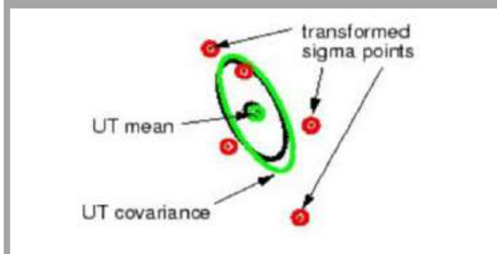
Approximate as Linear

Approximate with Samples

Extended Kalman Filter (EKF)



Unscented Kalman Filter - UKF



4

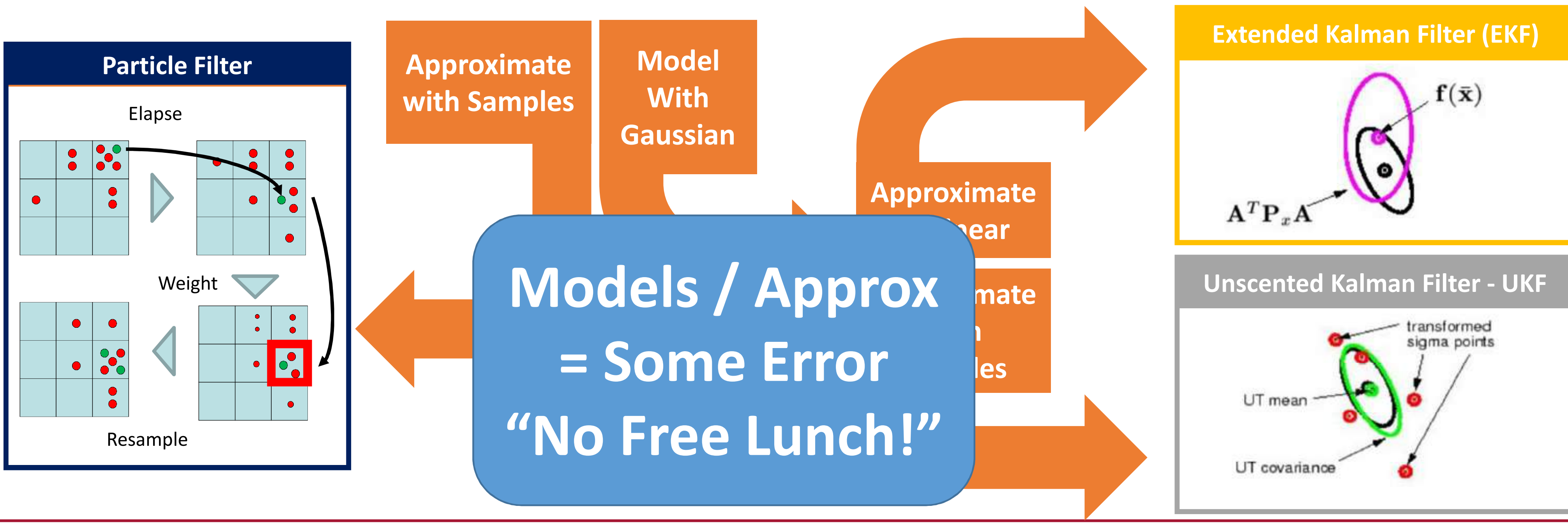
Mapping/Localization is solved by modeling the world as an HMM and using modeling and approximating to solve it

Time Update $X_1 \rightarrow X_2$

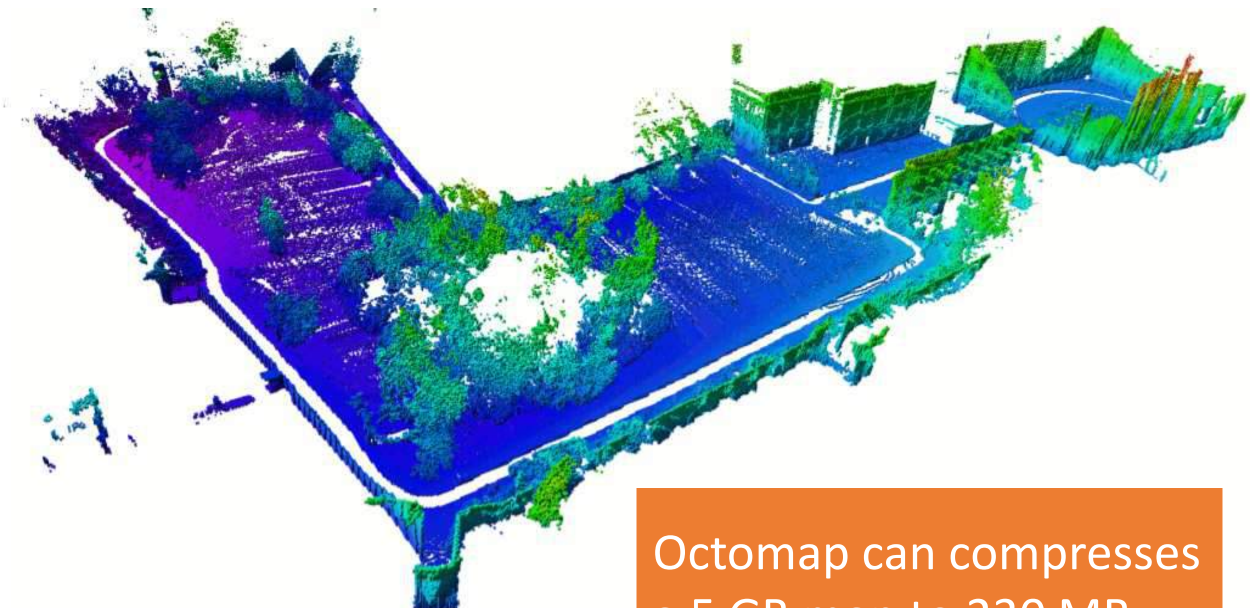
$$P(x_{t+1}|x_0, e_1 \dots e_t) = \int P(x_t|x_0, e_1 \dots e_t) * P(x_{t+1}|x_t)$$

Evidence Update $X_2 \rightarrow E_2$

$$P(x_{t+1}|x_0, e_1 \dots e_{t+1}) \propto \int P(x_{t+1}|x_0, e_1 \dots e_t) * P(e_{t+1}|x_{t+1})$$



4 Also we need to approximate the resolution of our maps and store them intelligently to fit them in memory



Octomap can compresses a 5 GB map to 230 MB

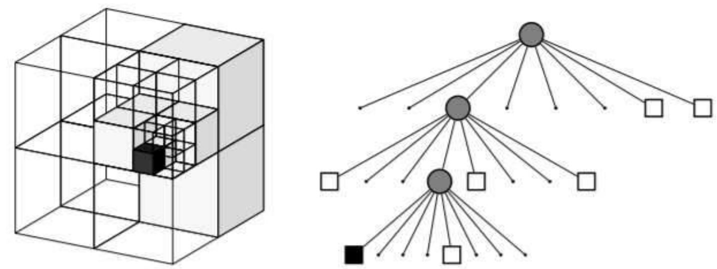


Fig. 2 Example of an octree storing free (shaded white) and occupied (black) cells. The volumetric model is shown on the left and the corresponding tree representation on the right.

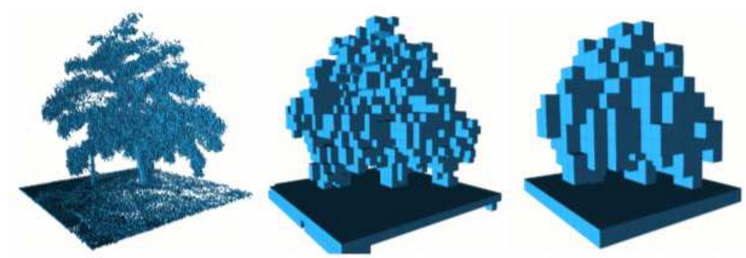
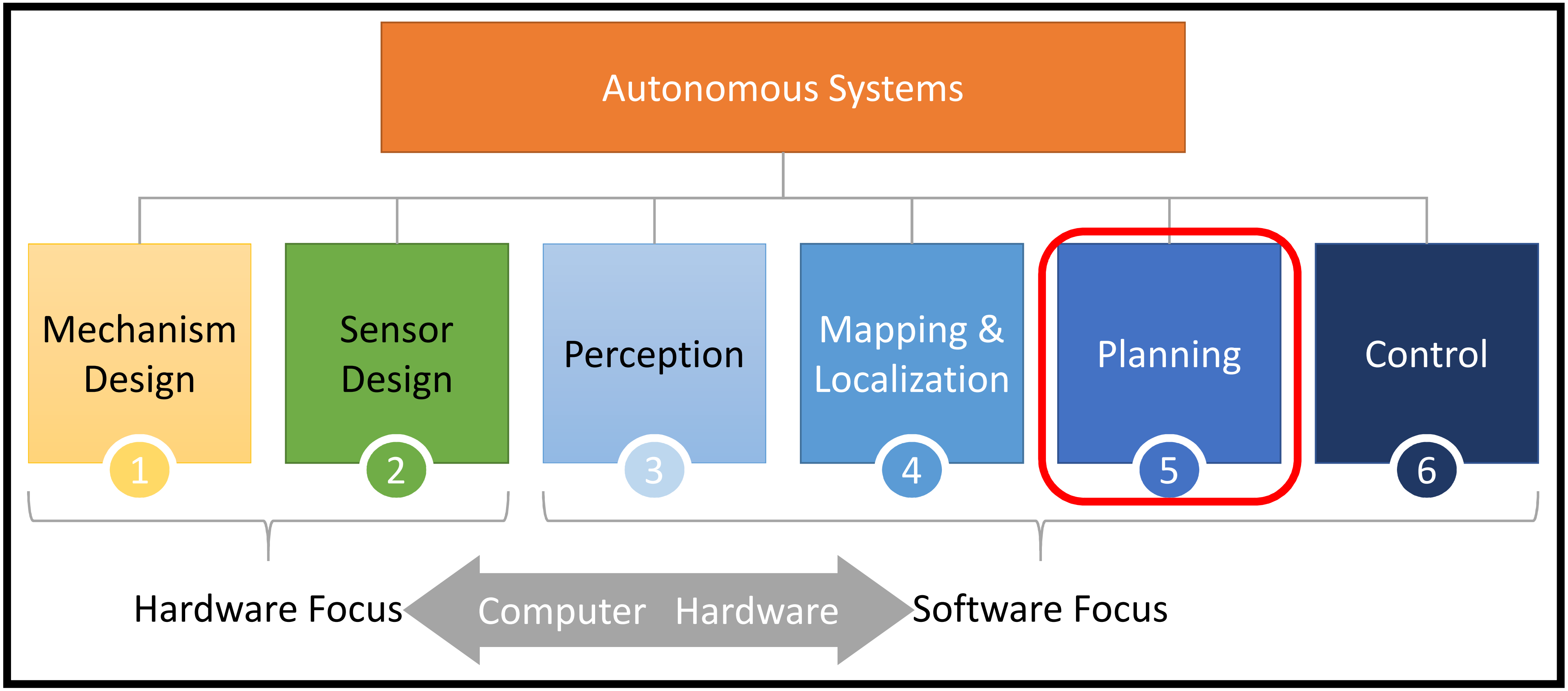


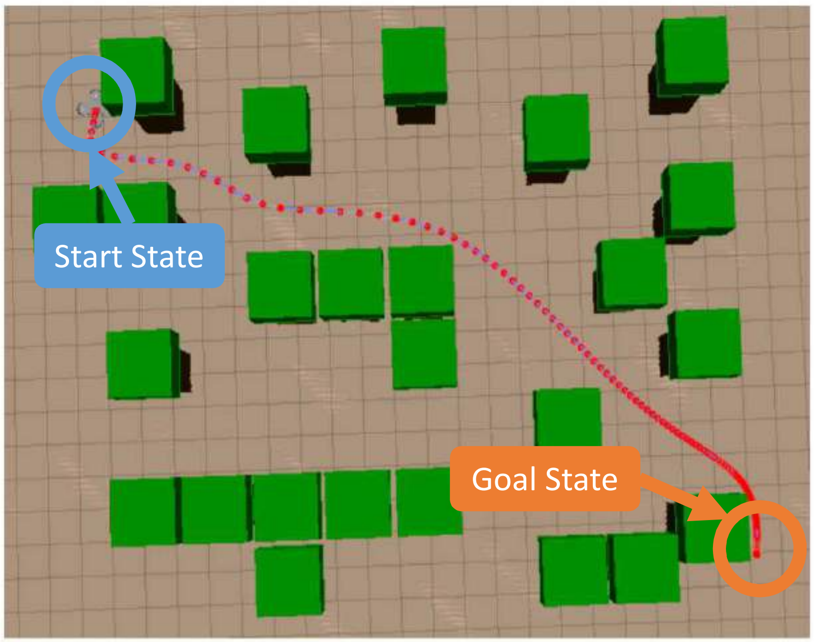
Fig. 3 By limiting the depth of a query, multiple resolutions of the same map can be obtained at any time. Occupied voxels are displayed in resolutions 0.08 m, 0.64 , and 1.28 m.

“Octomap” Hornung et. al. 2012

Autonomous Systems / Robotics is a BIG space



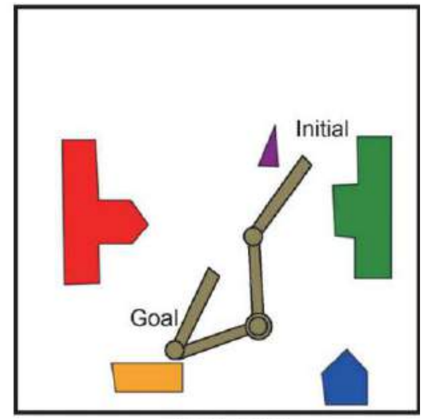
5 Planning (in Configuration Space) is hard



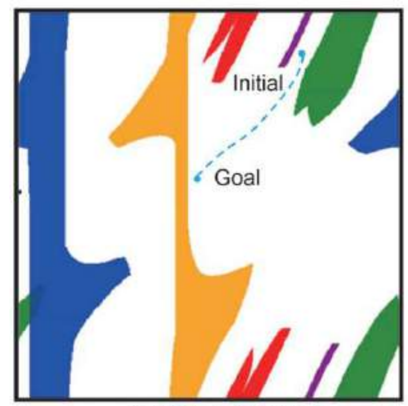
One approach is to ***discretize*** the statespace (grid it) and use graph search (A* = fast)

Another is to solve a global optimization problem:

$$\begin{aligned} \underset{s_0, a_0, \ldots, s_N, a_N}{\text{minimize}} \quad & \sum_{k=0}^{N} c(s_k, a_k) \\ \text{subject to} \quad & s_{k+1} = f(s_k, a_k) \\ & s_N = s_{\text{goal}} \end{aligned}$$



Workspace

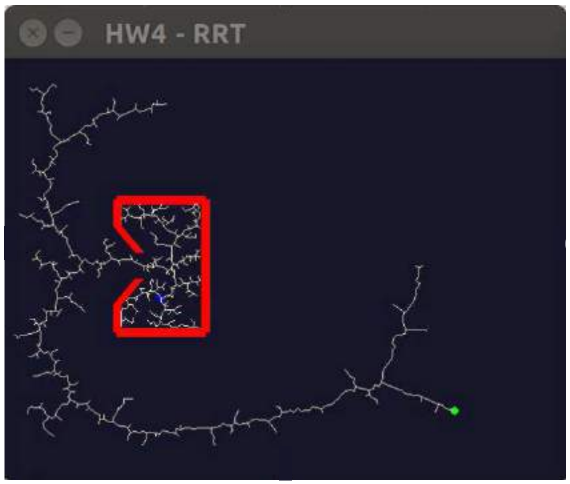


Configuration space

Complexity scales with $d^{|S| = |A|}$: **Curse of Dimensionality**

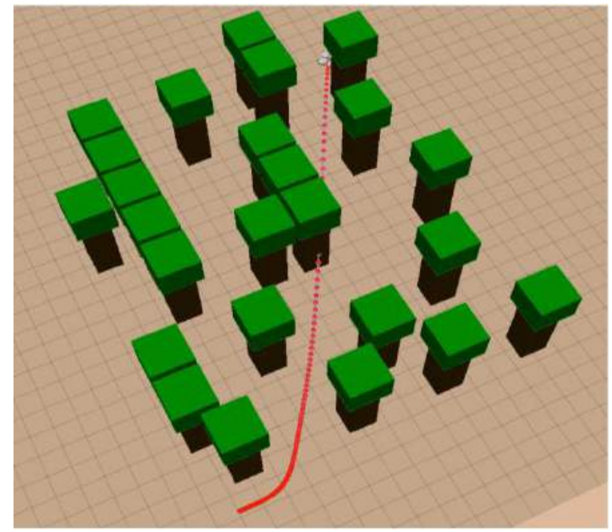
5

There are three main ways to approximately plan in Configuration Space



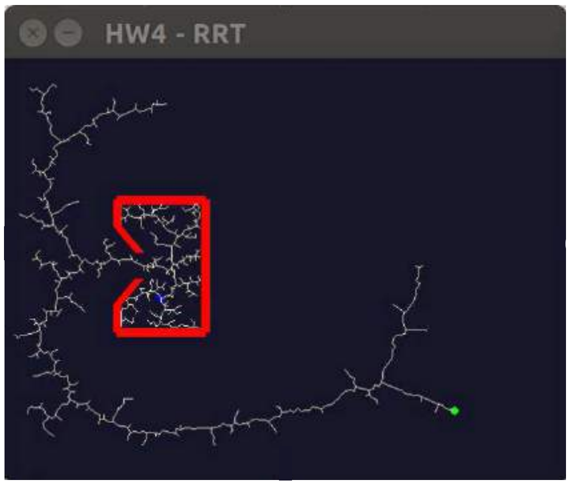
Random Search

Machine Learning



Local Search

5 We can approximately plan locally optimal plans in Configuration Space in three ways



Random Search

Online probabilistic completeness and optimality with RRT*

Offline/Online for "multi-query" with PRM

RRT

RRT*

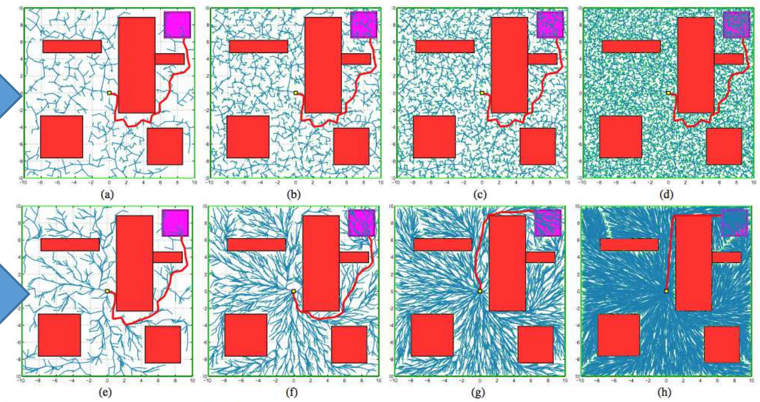
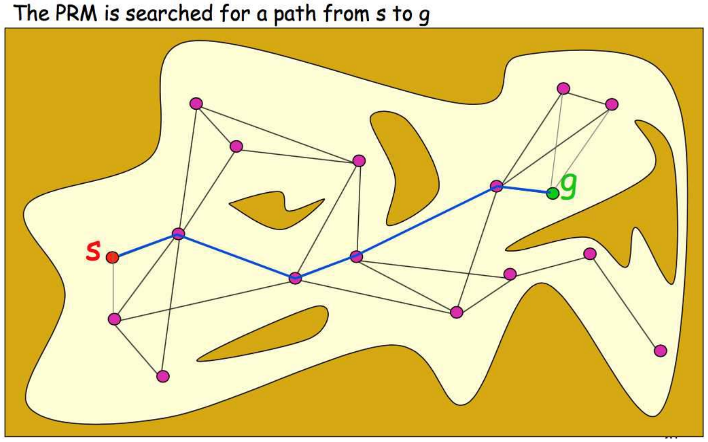
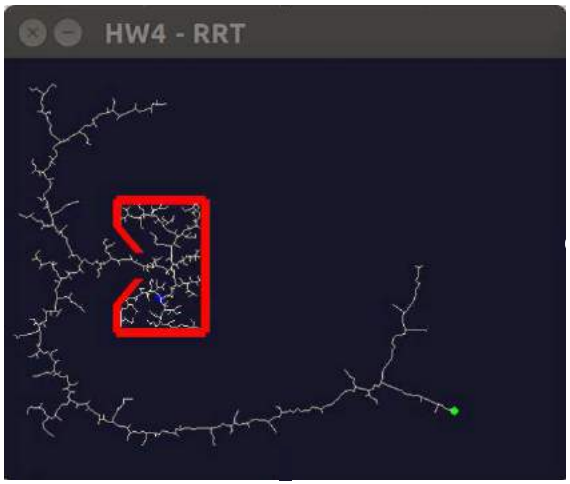


Fig. 1. A Comparison of the RRT* and RRT algorithms on a simulation example. The tree maintained by the RRT algorithm is shown in (a)-(d) in different stages, whereas that maintained by the RRT* algorithm is shown in (e)-(h). The tree snapshots (a), (e) are at 1000 iterations , (b), (f) at 2500 iterations, (c), (g) at 5000 iterations, and (d), (h) at 15,000 iterations. The goal regions are shown in magenta. The best paths that reach the target are highlighted with red.

The PRM is searched for a path from s to g



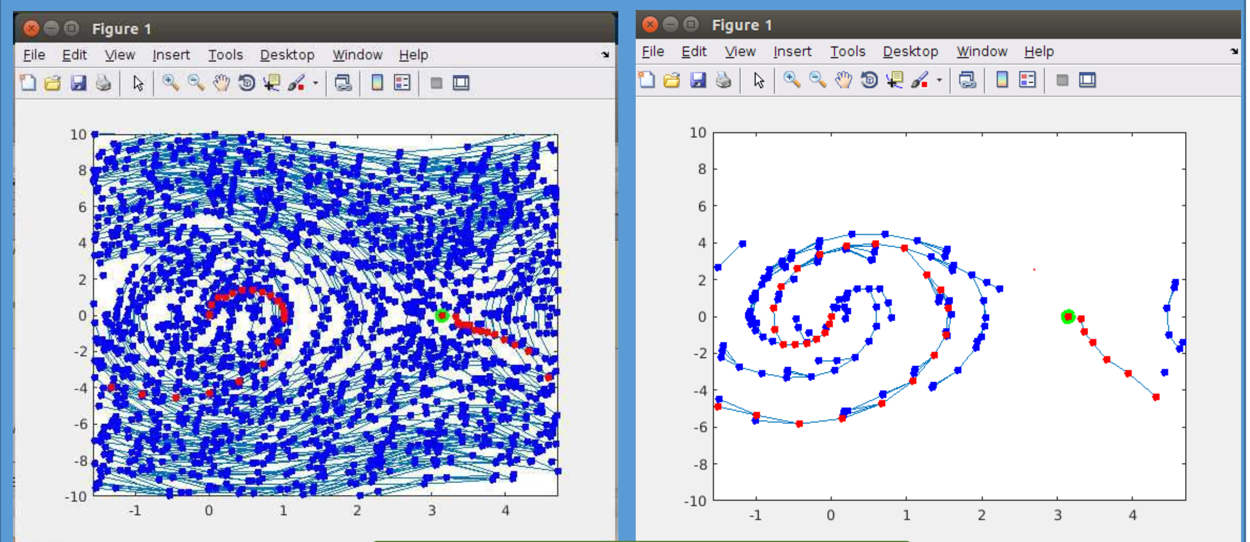
5 We can approximately plan locally optimal plans in Configuration Space in three ways



Random Search

The PRM is searched for a path from s to g

Note: Can scale to low-dimensional dynamical systems with LQR-RRT*



"No Free Lunch!"

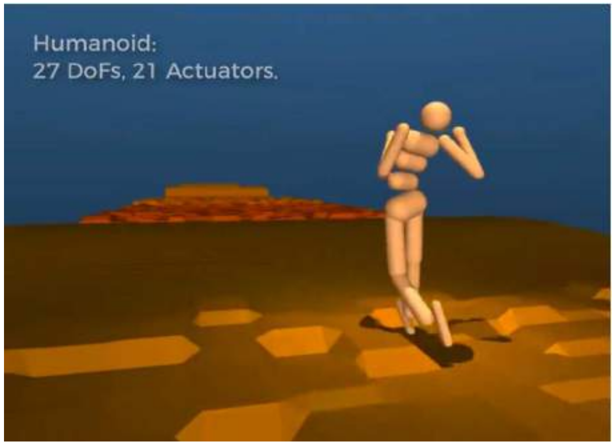
(g) (h)

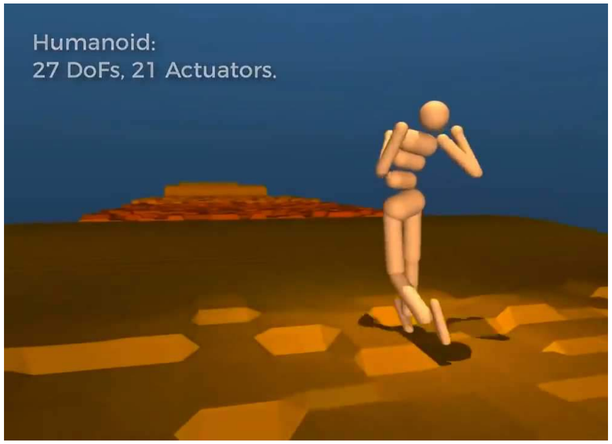
...ee maintained by the RRT algorithm is shown in (a)-(d) in different ...apshots (a), (e) are at 1000 iterations , (b), (f) at 2500 iterations, (c), (g) at 5000 iterations, and (d), (h) at 15,000 iterations. The goal regions are shown in magenta. The best paths that reach the target are highlighted with red.

5

We can approximately plan locally optimal plans in Configuration Space in three ways

Machine Learning





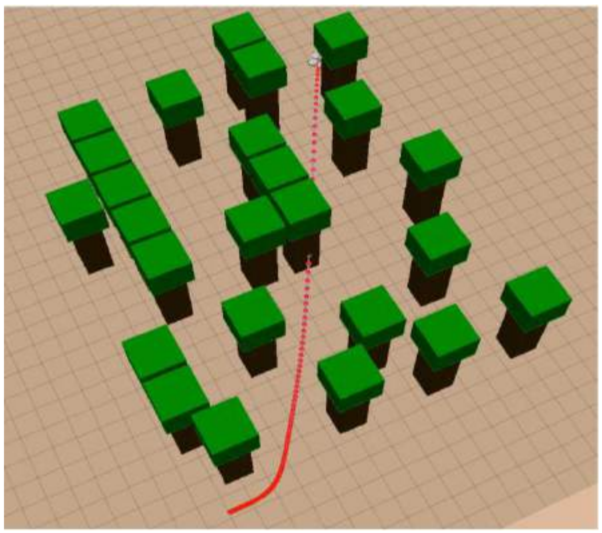
My two cents: Yes, and no free lunch!

Needs to re-lean physics and suffers from sample complexity

In two weeks more on this!

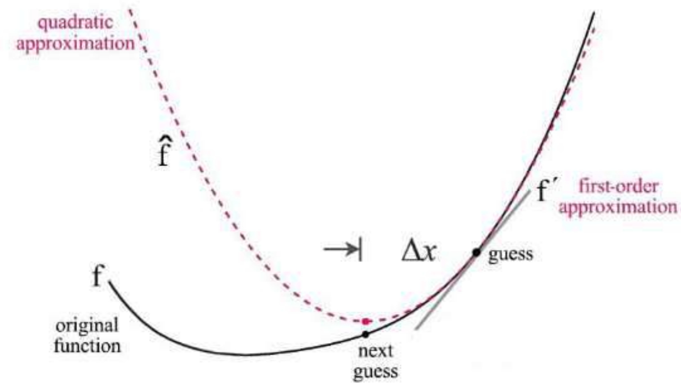
5

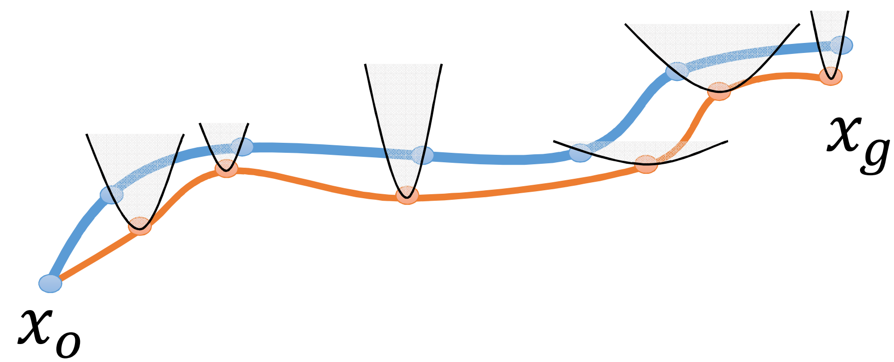
We can approximately plan locally optimal plans in Configuration Space in three ways



Local Search

Solve math locally with linear & quadratic approximations



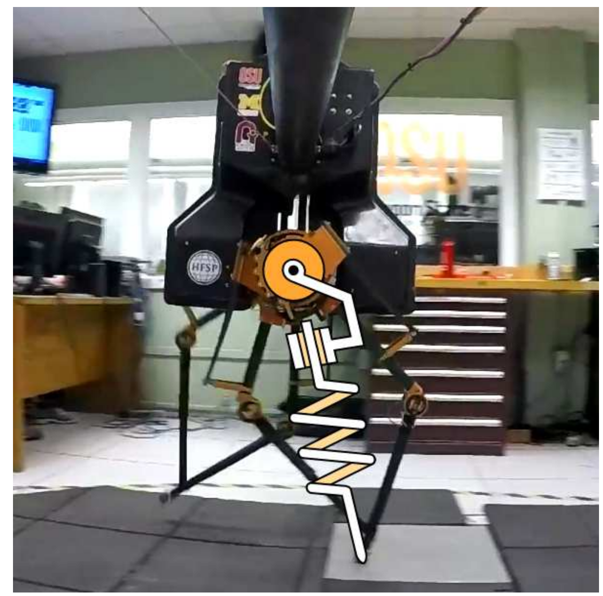


5

Practical Challenges for Trajectory Optimization: Not Complete, Not Robustness and Contact = No Free Lunch!

1. **Not complete** (aka no guaranteed solution) and often slow!
2. Solvers are **numerically sensitive**
3. Solutions are sensitive to initial trajectories and **perturbations**
4. The physics equations are fundamentally different when an object makes or breaks **contact** leading to a **combinatorial explosion**

One approach to avoid solving these large hard problems is to solve the problem by **combining simpler models** of the system although this leads to **conservative** behavior



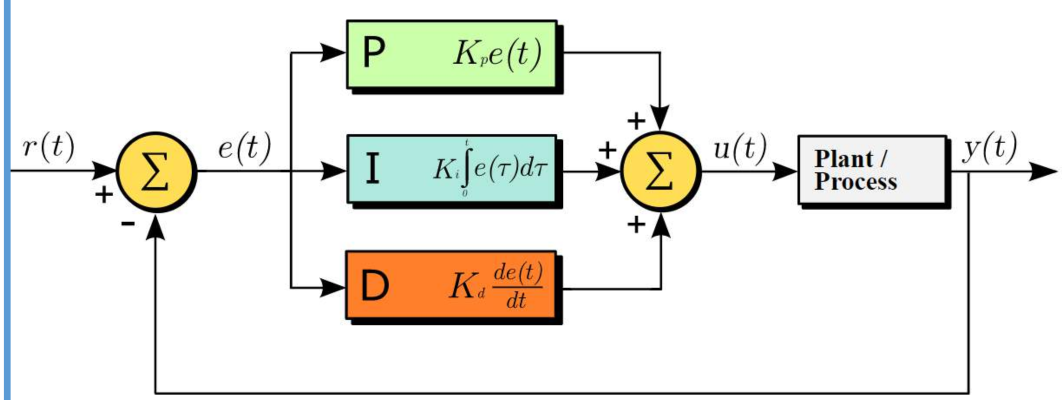
5 Control is hard (even for the experts)

6

We use feedback tracking controllers to run our plans in the real world (and handle the differences encountered)

Model as linear combination of errors and approximate gains

This is the canonical **PID controller**!



LQR: Quadratic Cost with Linear Dynamics

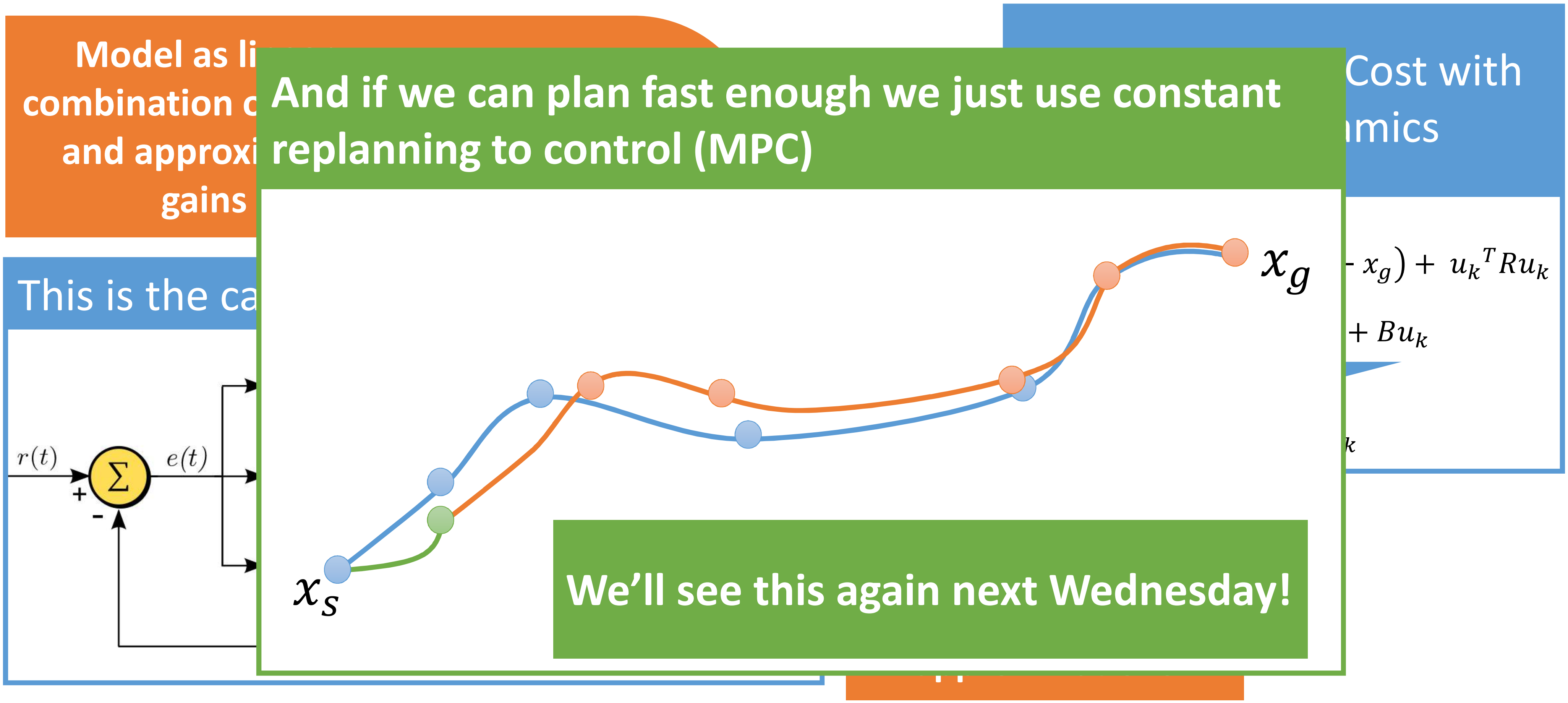
$$\min_{x,u} \sum_{k=0}^{N} (x_k - x_g)^T Q (x_k - x_g) + u_k{}^T R u_k$$

$$\text{s.t.} \quad x_{k+1} = A x_k + B u_k$$

$$u_k = -K_k x_k$$

Solve math locally with linear & quadratic approximations

6 We use feedback tracking controllers to run our plans in the real world (and handle the differences encountered)



6 Practical Challenges for Control: Contact

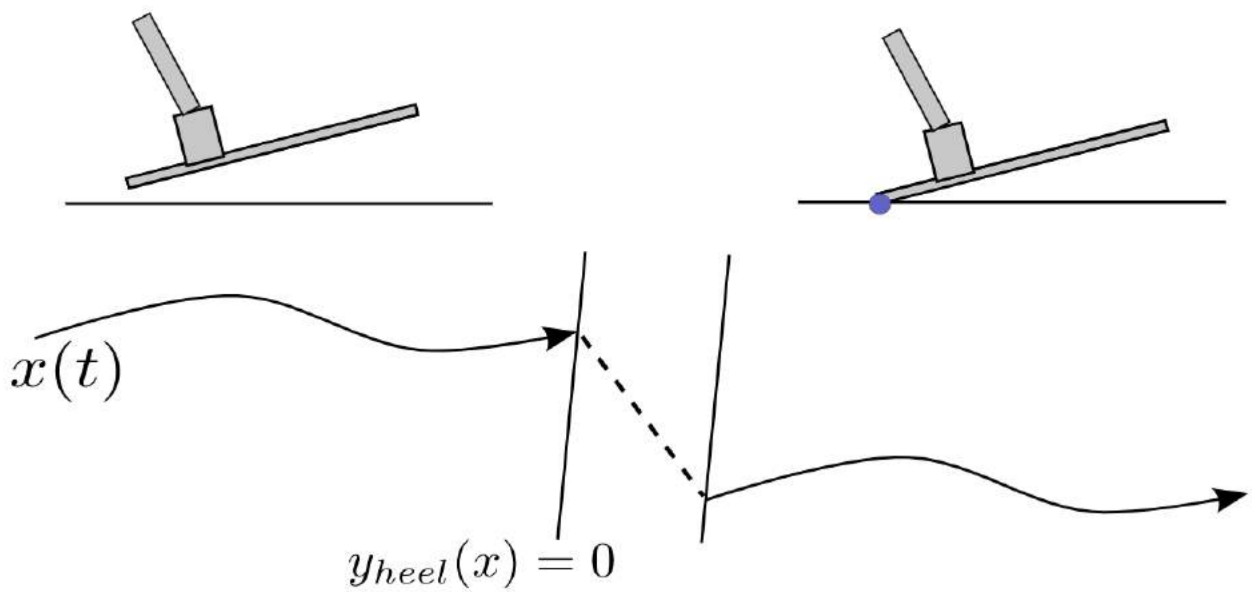


Figure 17.1 - Modeling contact as a hybrid system.

The goal for the next couple of lectures is to develop a high level understanding of:

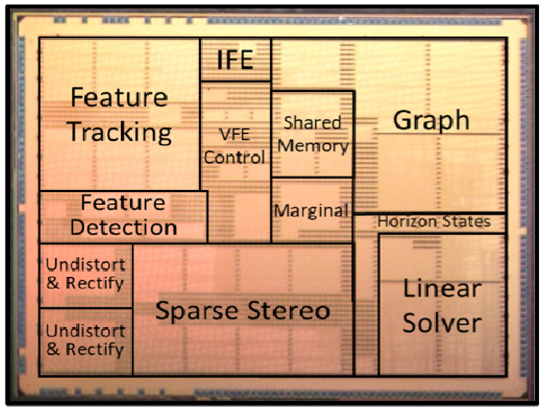
1. What is an autonomous system
2. Key problems and constraints for autonomous systems
3. Some of the most important (classes of) algorithms in robotics
   A. The model based vs. model free tradeoff
   B. The online vs offlin
   C. The no free lunch t mations
4. How computer systems / architecture design has and can play a role in improving autonomous systems

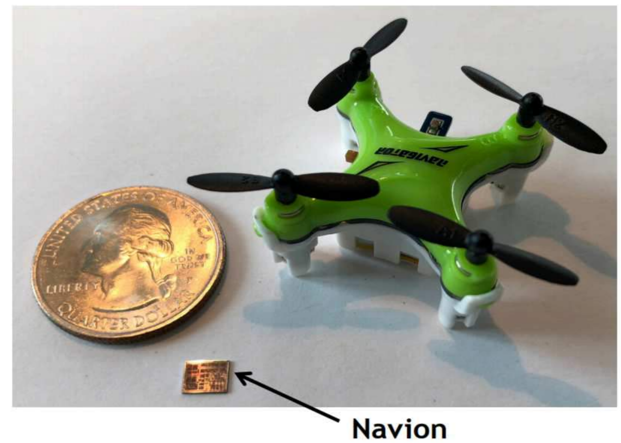
This is what we will explore in all of the papers!

The goal for the next couple of lectures is to develop a high level understanding of:

1. What is an autonomo
2. Key problems and con
3. Some of the most imp
   A. The model based vs. r
   B. The online vs offlin
   C. The no free lunch t mations
4. How computer systems / architecture design has and can play a role in improving autonomous systems

http://navion.mit.edu





This is what we will explore in all of the papers!

Your homework – get on HOTCRP

Email **Glenn Holloway:**
holloway@eecs.harvard.edu

He will send you a password (username is that email address) after which I can assign you access to review papers