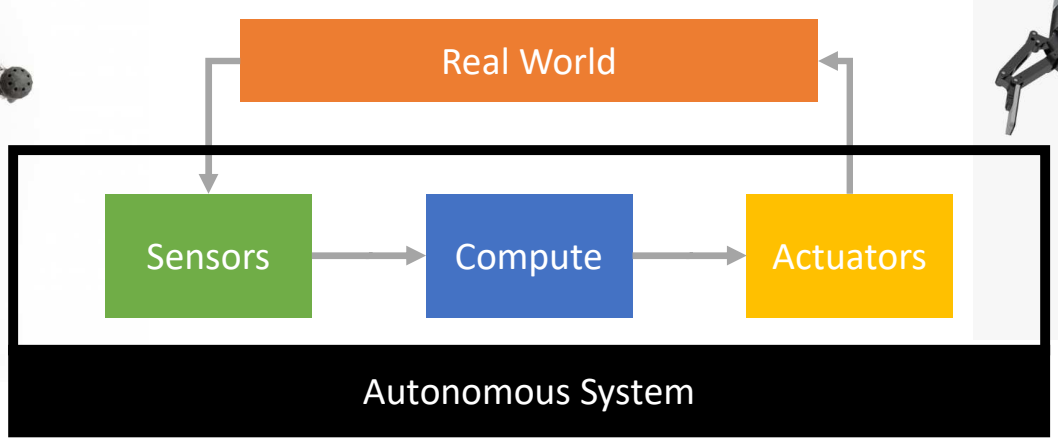
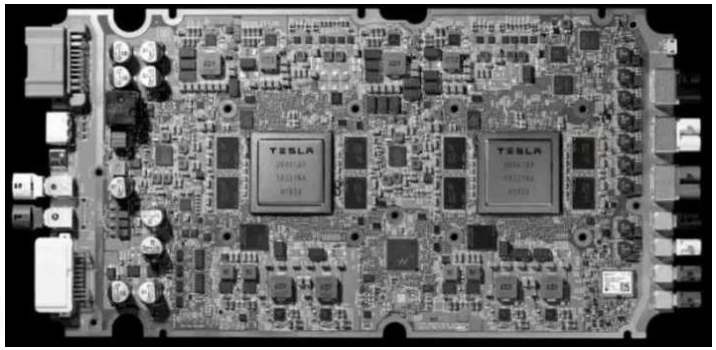
CS 249r: Special Topics in Edge Computing

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%

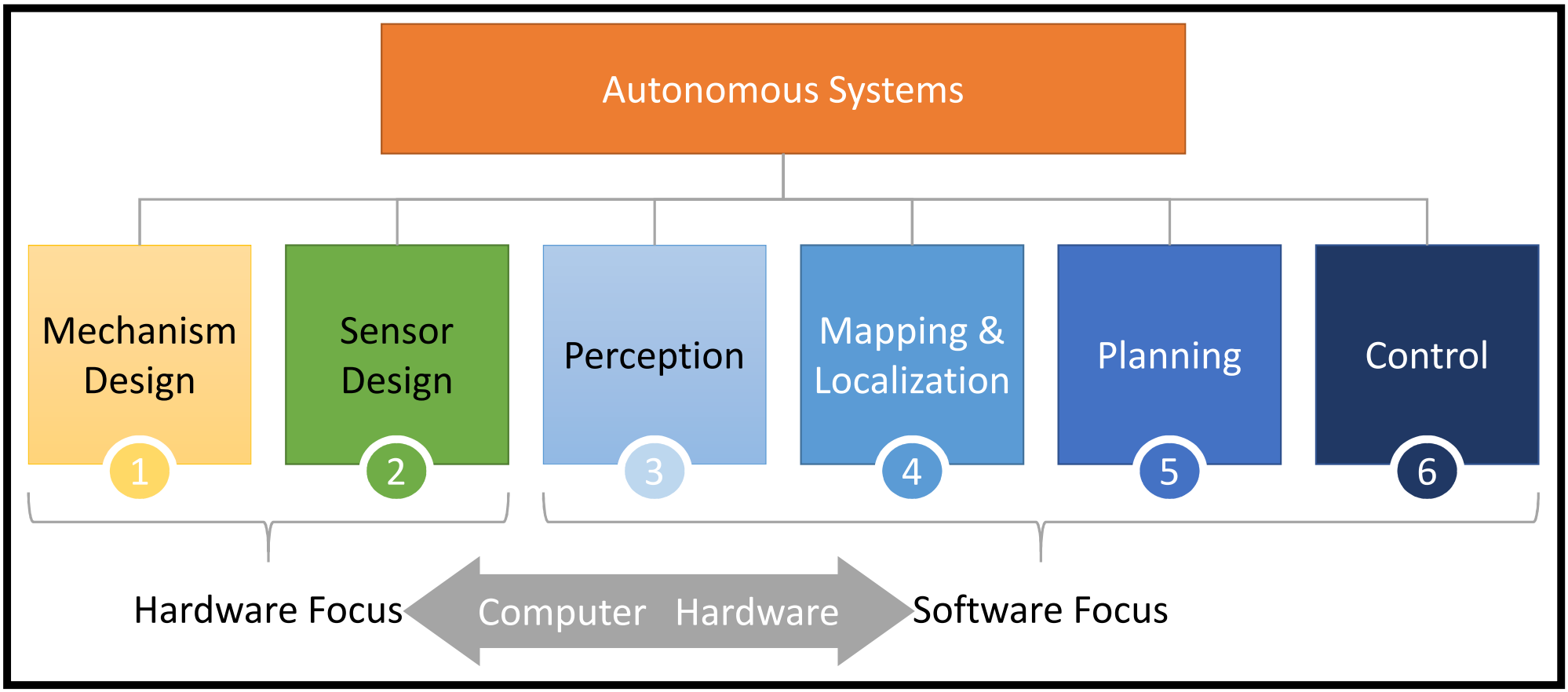
What are the prerequisites for CS 249r?

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?

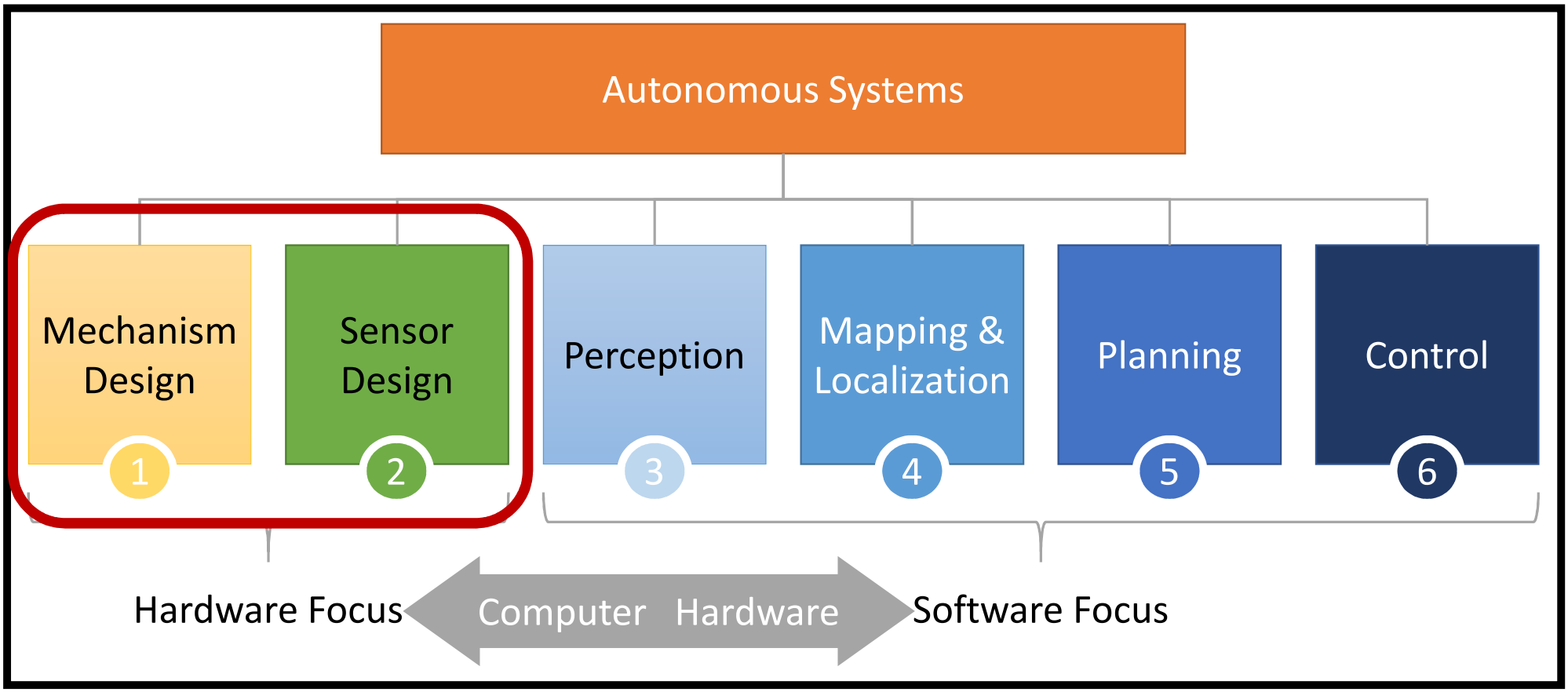
Autonomous Systems / Robotics is a **BIG** space



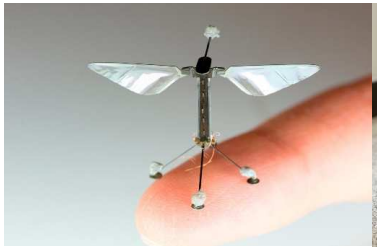
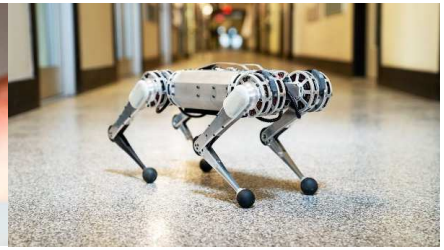
The goal for the next couple of lectures is to develop a high level understanding of:

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

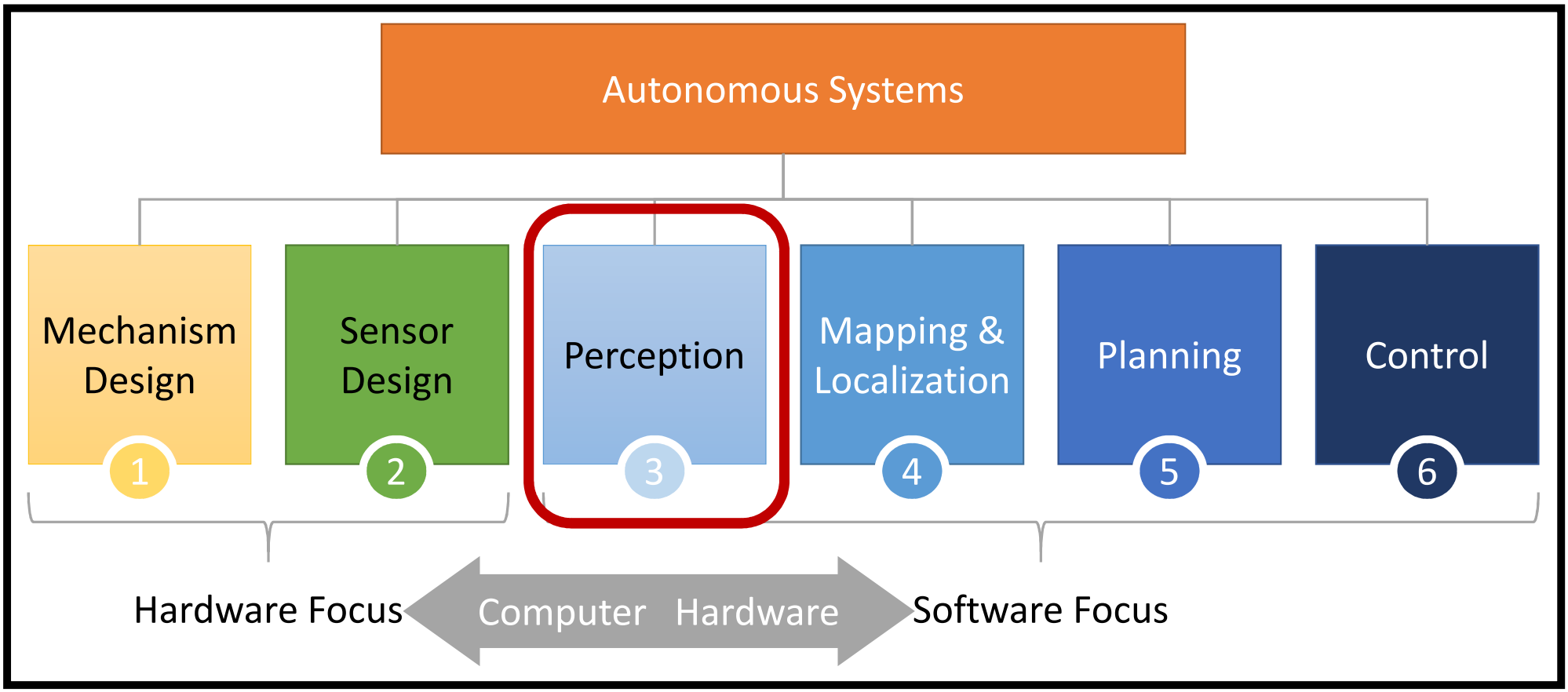


12 Key Takeaways:

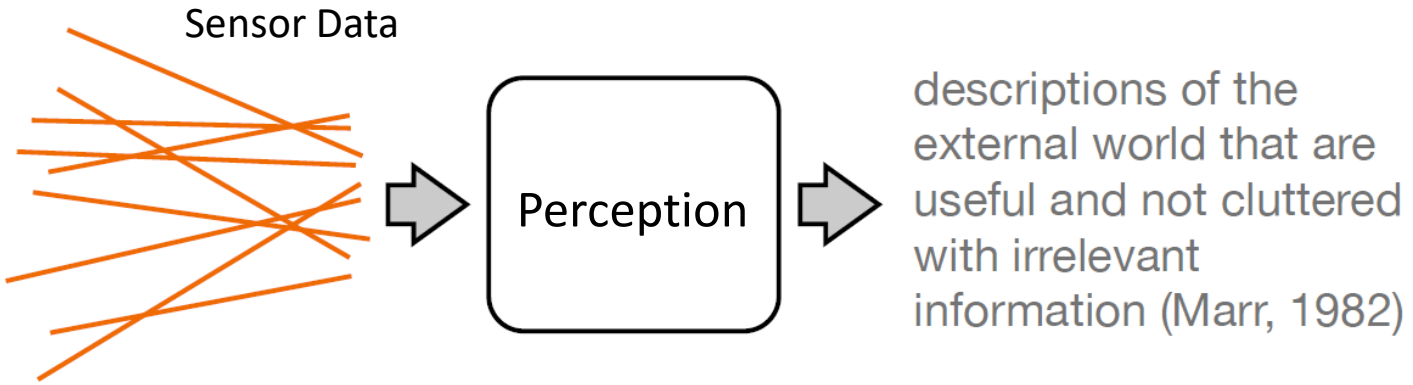




1. When designing algorithms for robots you need to understand the physical capabilities of the robot and you (potentially) need to understand how to model its physical behaviors
2. Different kinds of systems will have different power, weight, and performance budgets for computer hardware

Autonomous Systems / Robotics is a BIG space

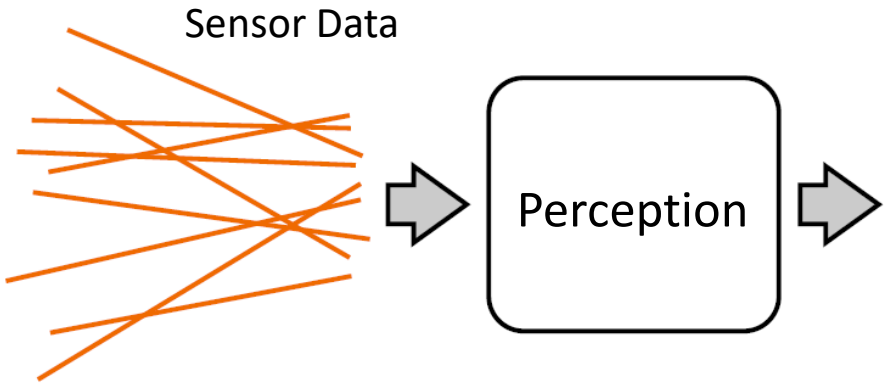


Perception is the processing of sensor data to understand the world around the robot



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)

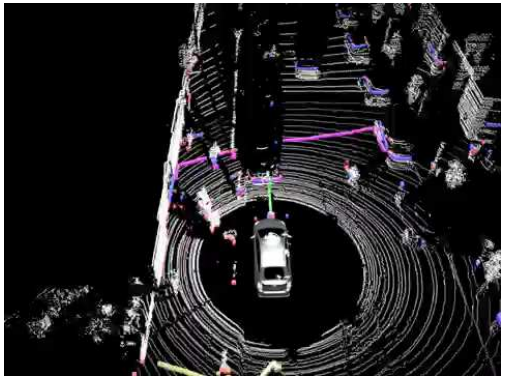
- Geometric information (3D shape, position)

Slide Credit: Todd Zickler CS 283

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.)

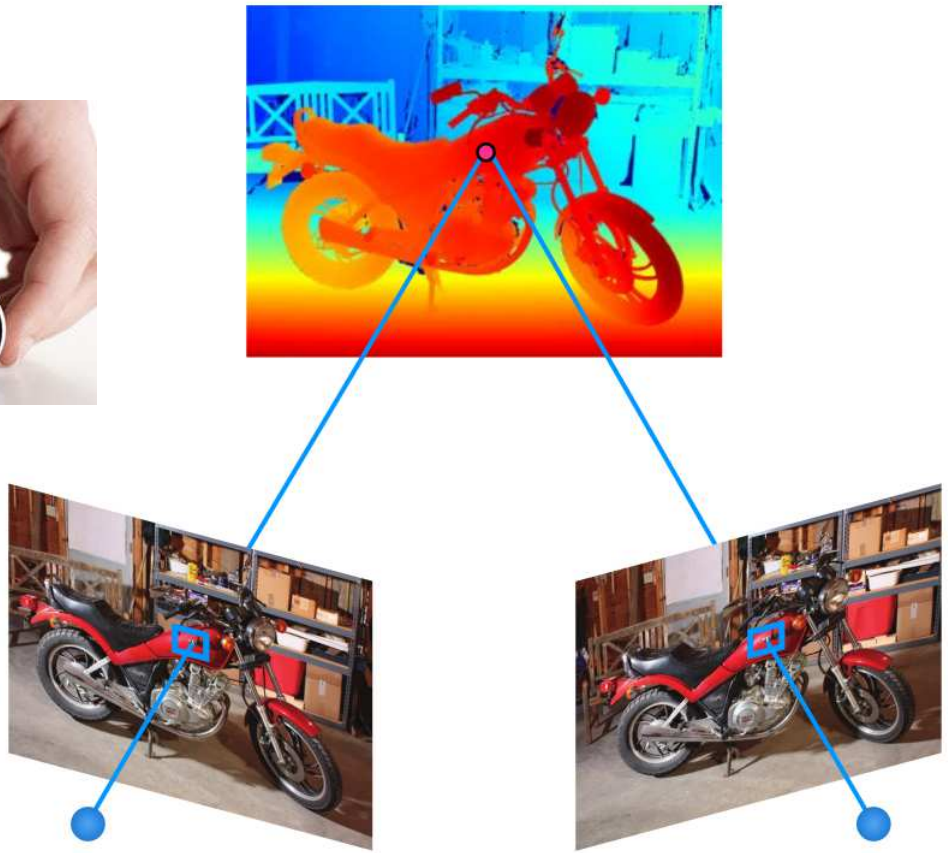


Unstructured Light (aka Cameras)

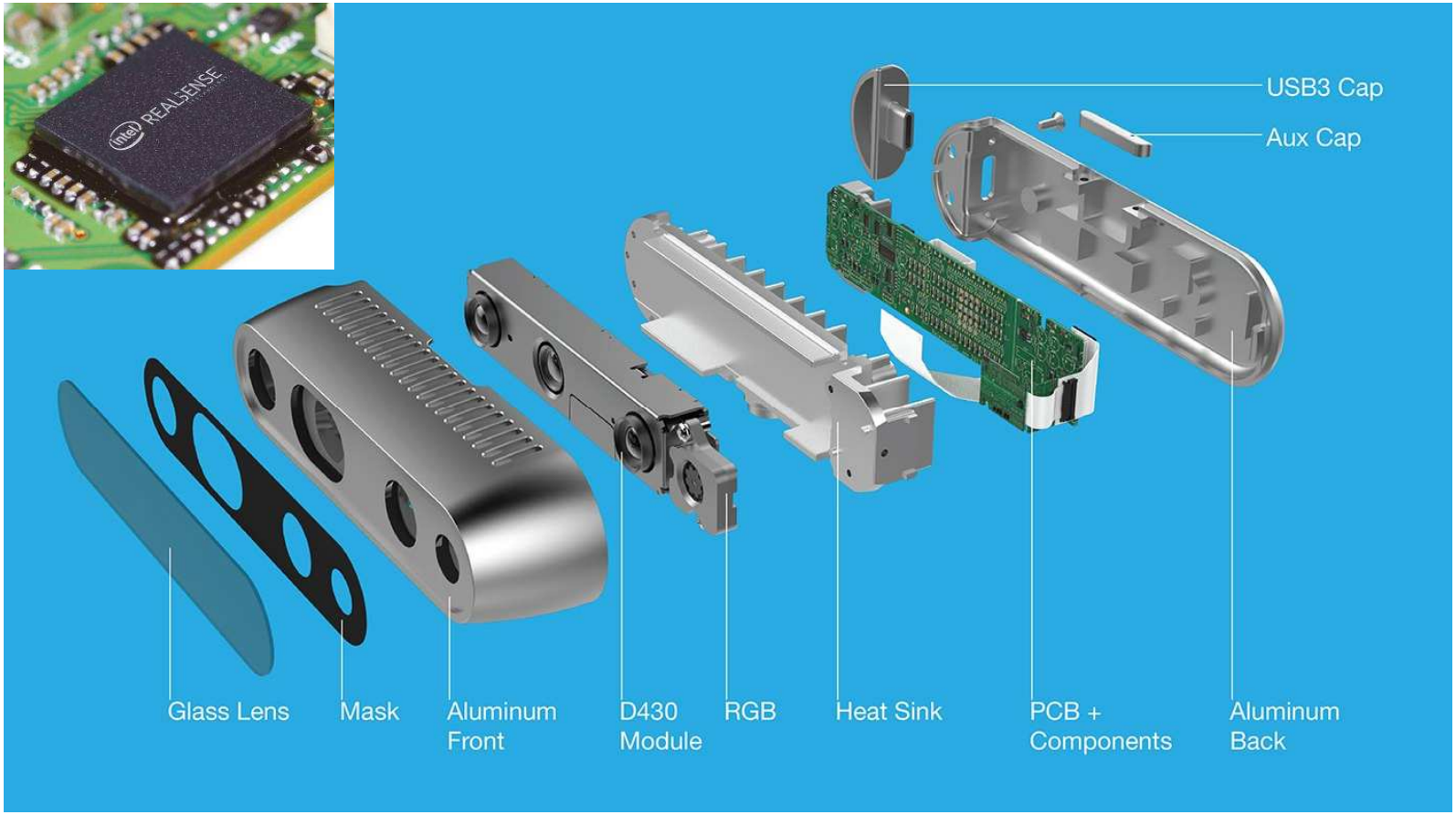
Computer Vision

Usable Data

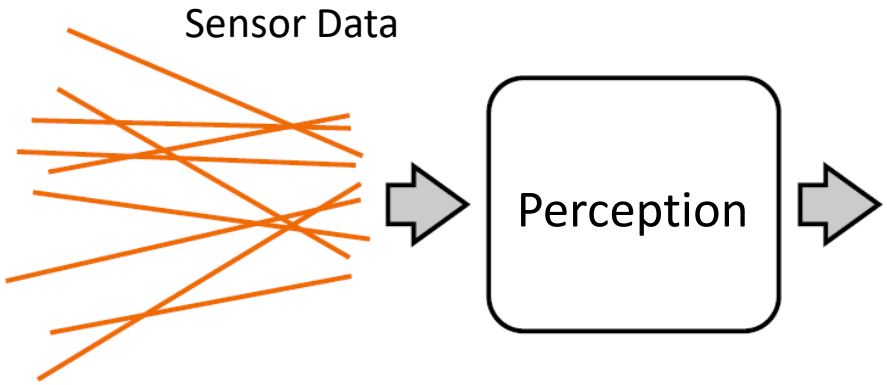
We can compute the depth to objects by using geometry and physics



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

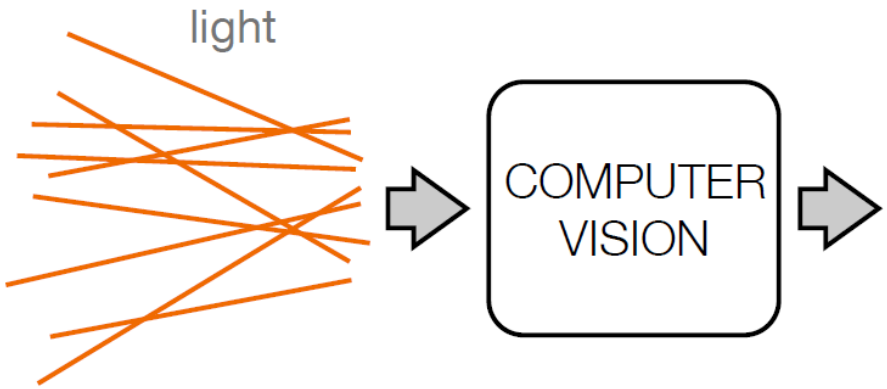


descriptions of the external world that are useful and not cluttered with irrelevant information (Marr, 1982)

- Geometric information (3D shape, position)

Slide Credit: Todd Zickler CS 283

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

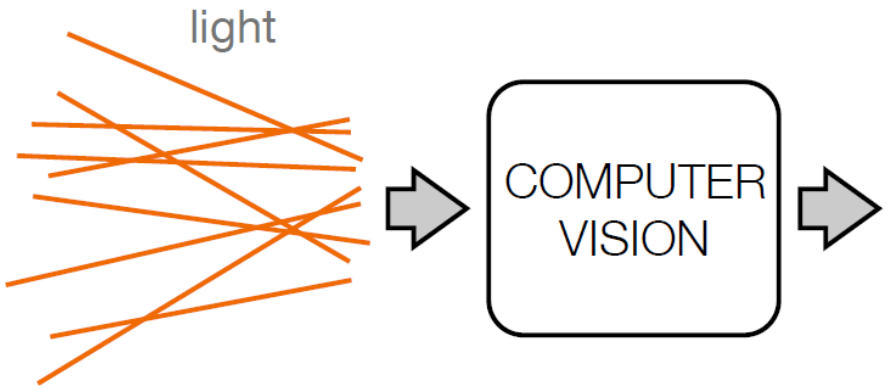


descriptions of the external world that are useful and not cluttered with irrelevant information (Marr, 1982)

- Geometric information (3D shape, position)

Slide Credit: Todd Zickler CS 283

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

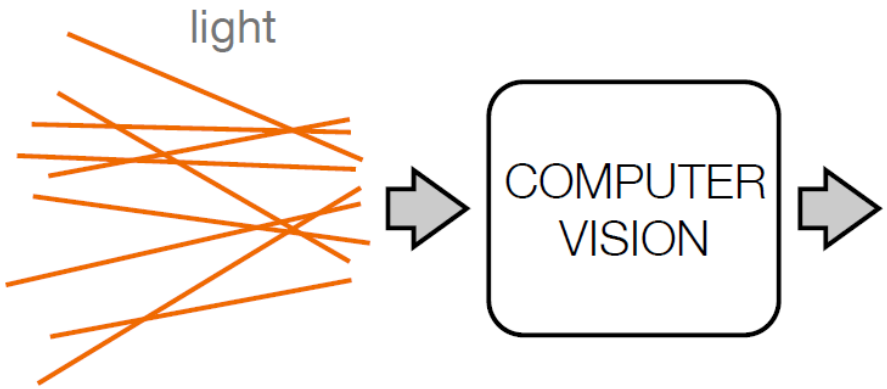


descriptions of the external world that are useful and not cluttered with irrelevant information (Marr, 1982)

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

Slide Credit: Todd Zickler CS 283

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)
- Semantic information (object, scene categories)

Slide Credit: Todd Zickler CS 283

3 Computer Vision is a hard problem

Computer Vision is a hard problem

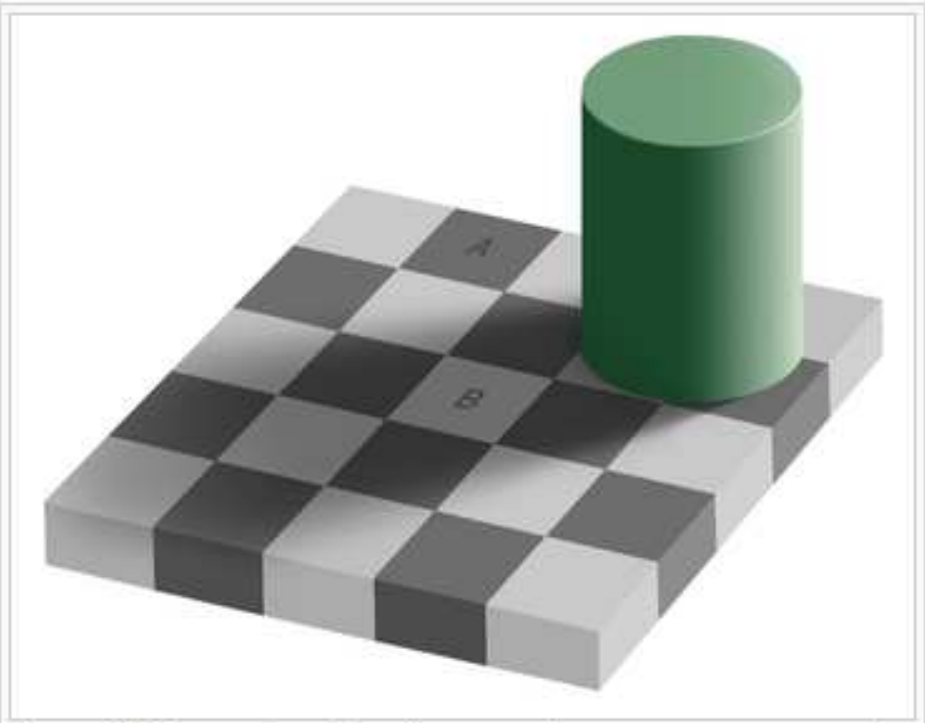
What color(s) are this shirt and these pants?

Slide Credit: Hamilton Chong

3 Computer Vision is a hard problem

Slide Credit: Hamilton Chong

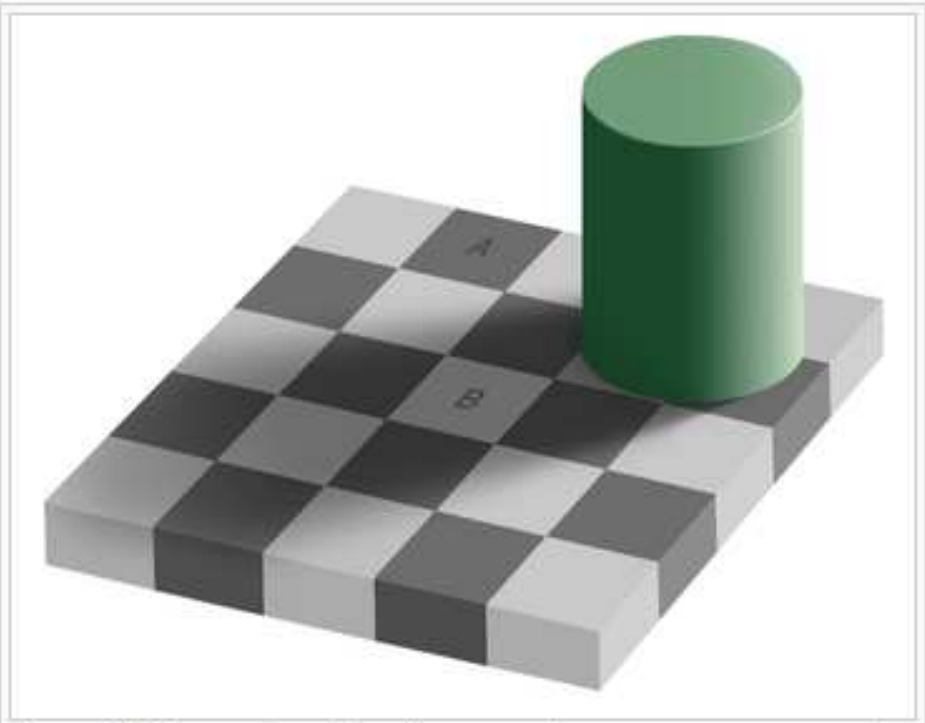
3 Computer Vision is a hard problem



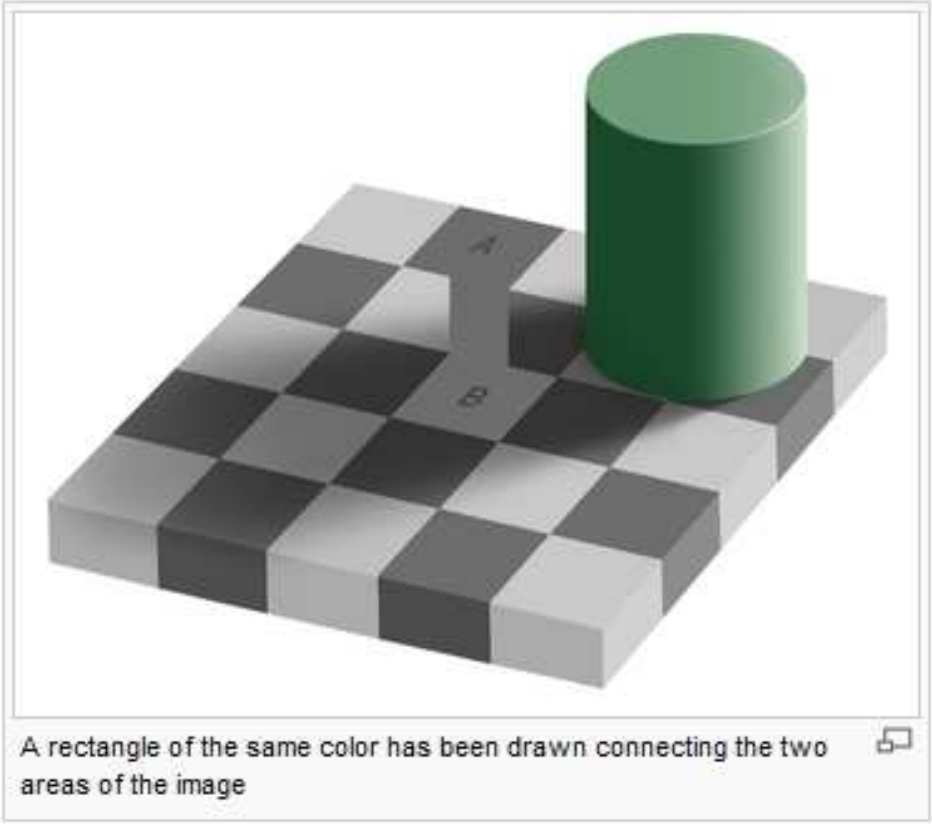
Areas of the image A and B are the same color

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

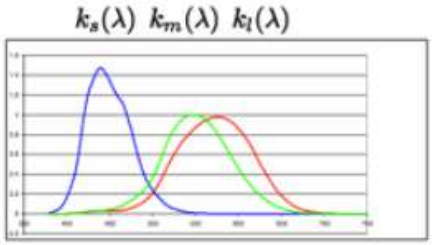
Sinha *et al.*: Face Recognition by Humans: Nineteen Results Researchers Should Know About

3 Computer Vision is a hard problem

Retinal color

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

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

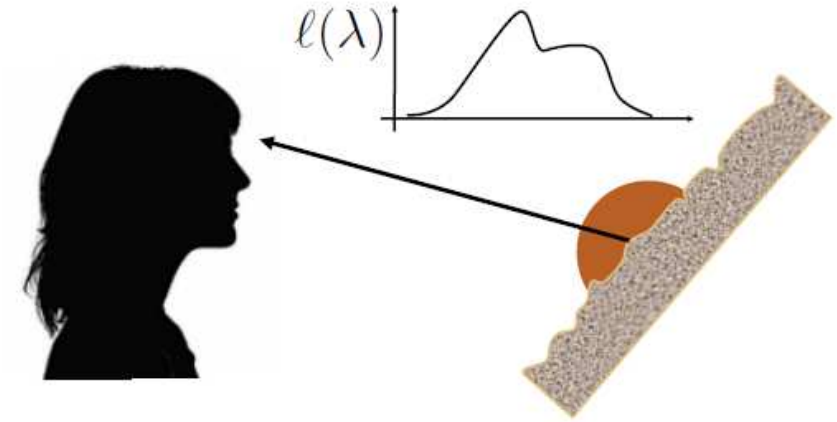


LMS senstivity functions

Perceived color

Object color

Color names



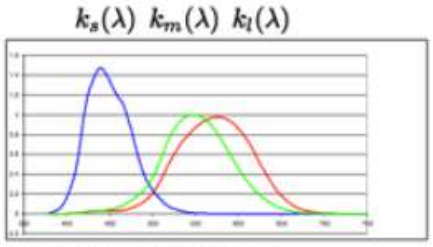
Slide Credit: Todd Zickler CS 283

3 Computer Vision is a hard problem

Retinal color

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

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



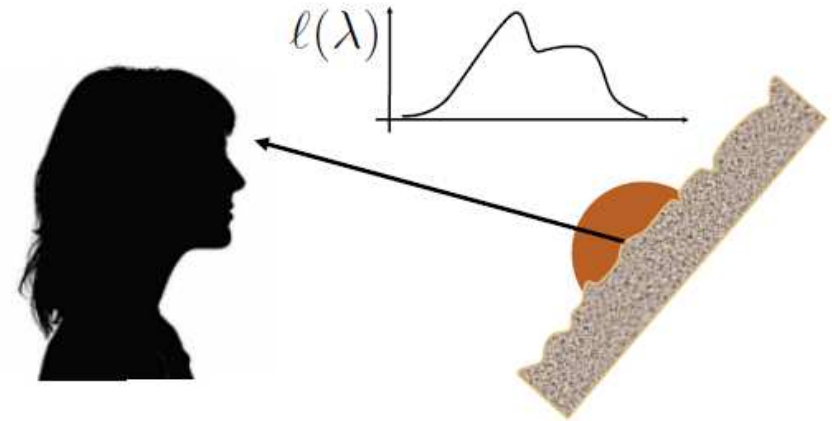
LMS senstivity functions

So how can we represent this with an algorithm?

Perceived color

Object color

Color names



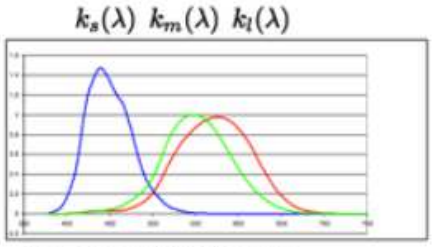
Slide Credit: Todd Zickler CS 283

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

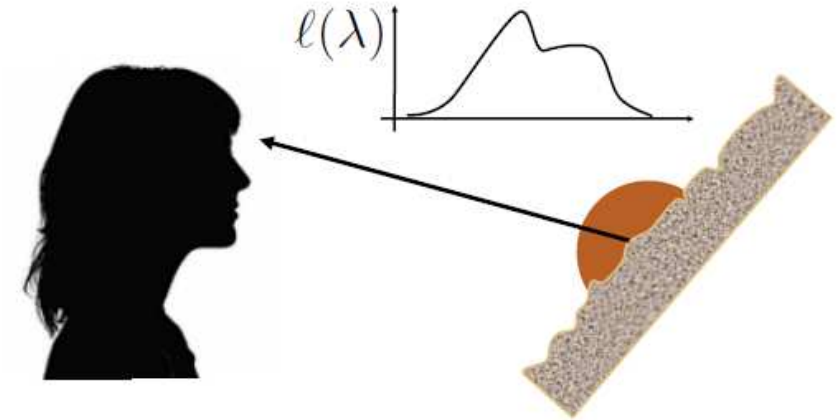
So how can we represent this with an algorithm?

Perceived color

Object color

Color names

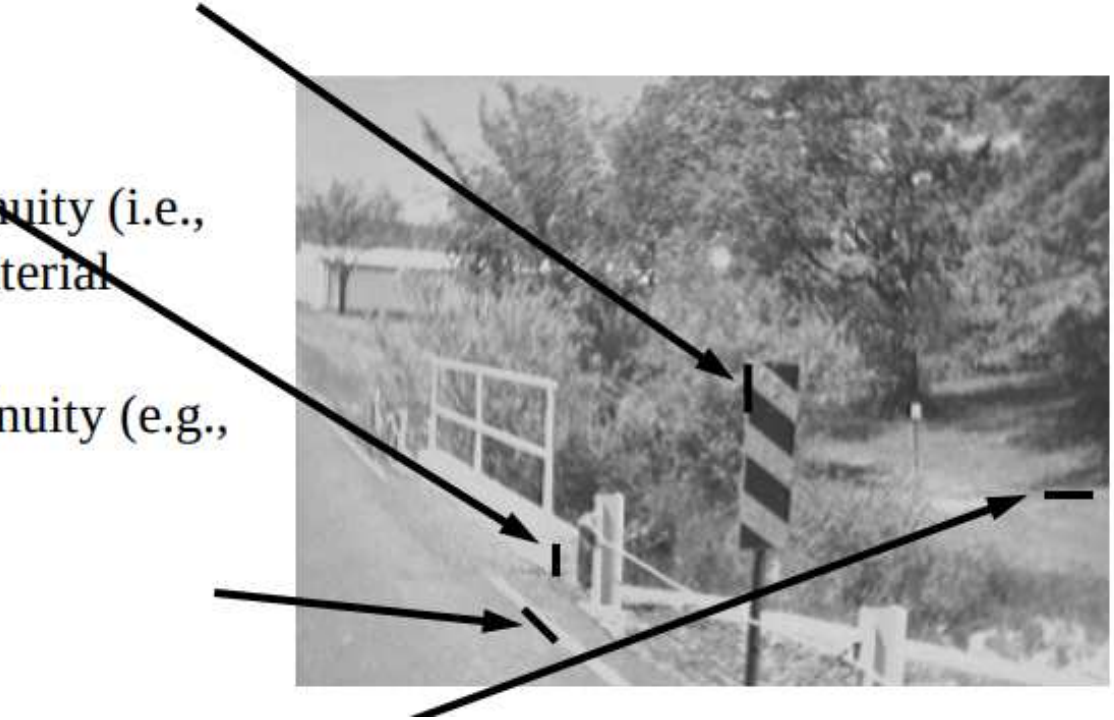
Well lets start by building up some intuition for how to find an edge!



Slide Credit: Todd Zickler CS 283

Edges are where discontinuities occur in images

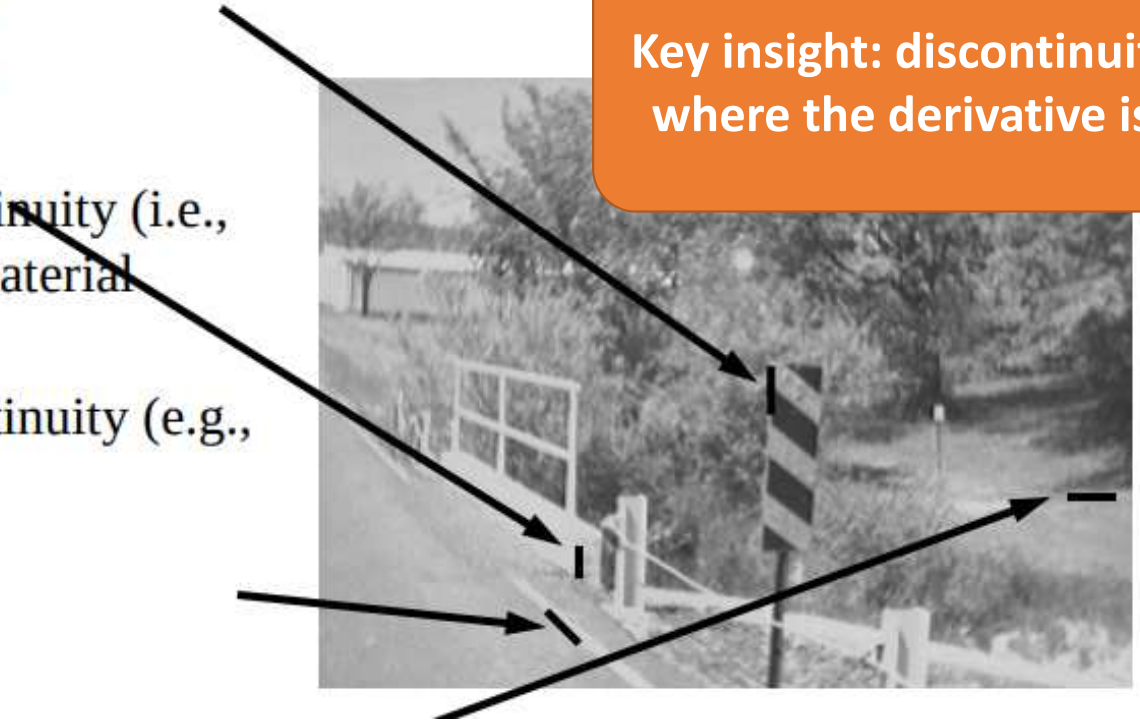
- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)
- Illumination discontinuity (e.g., shadow)



Slide credit: Christopher Rasmussen

3 Edges are where discontinuities occur in images

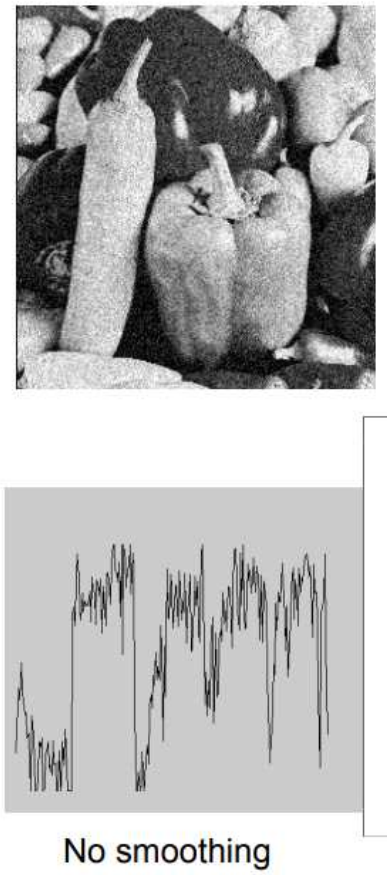
- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)
- Illumination discontinuity (e.g., shadow)



Key insight: discontinuities are where the derivative is high!

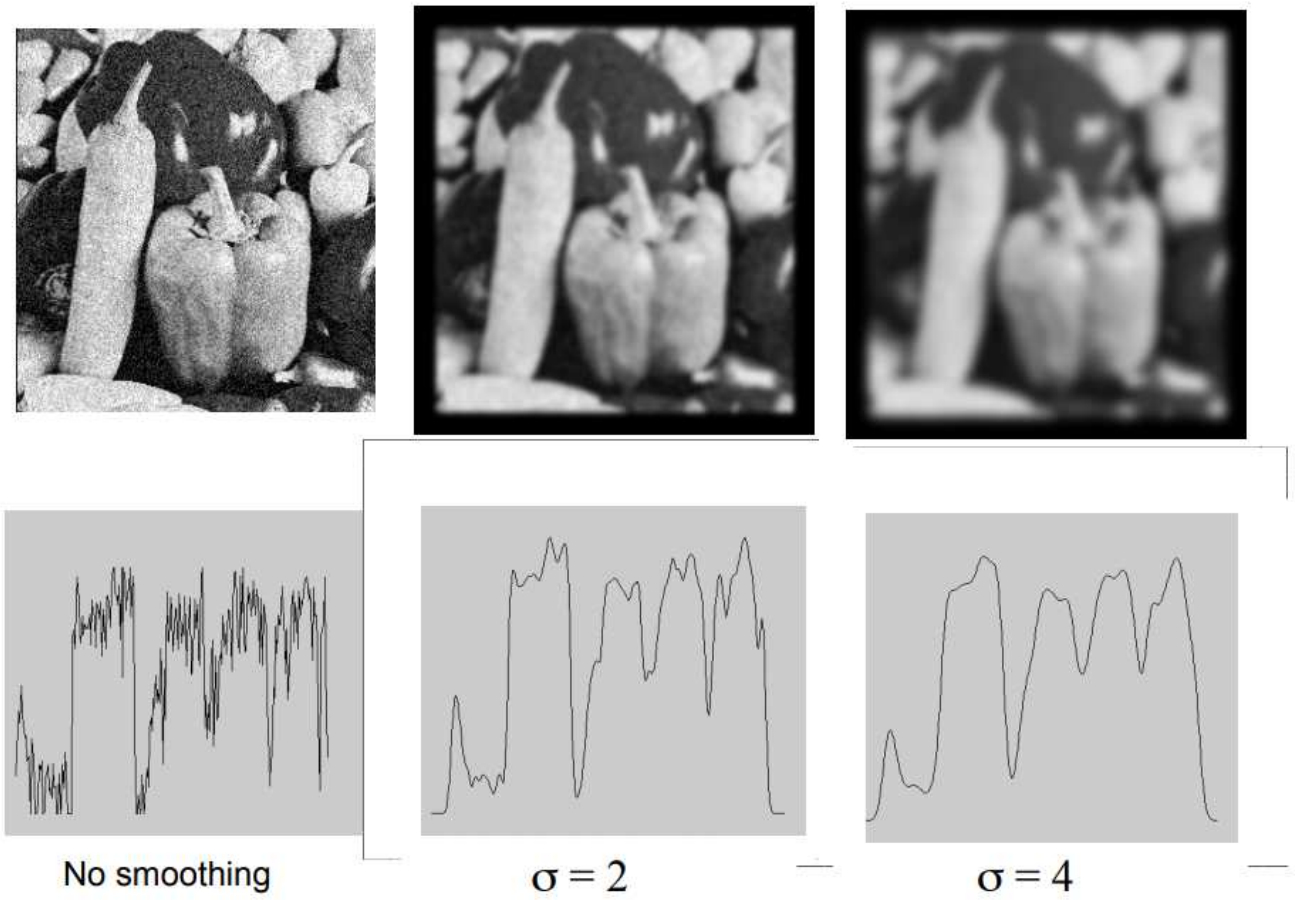
Slide credit: Christopher Rasmussen

3 Noise will corrupt our derivative computation



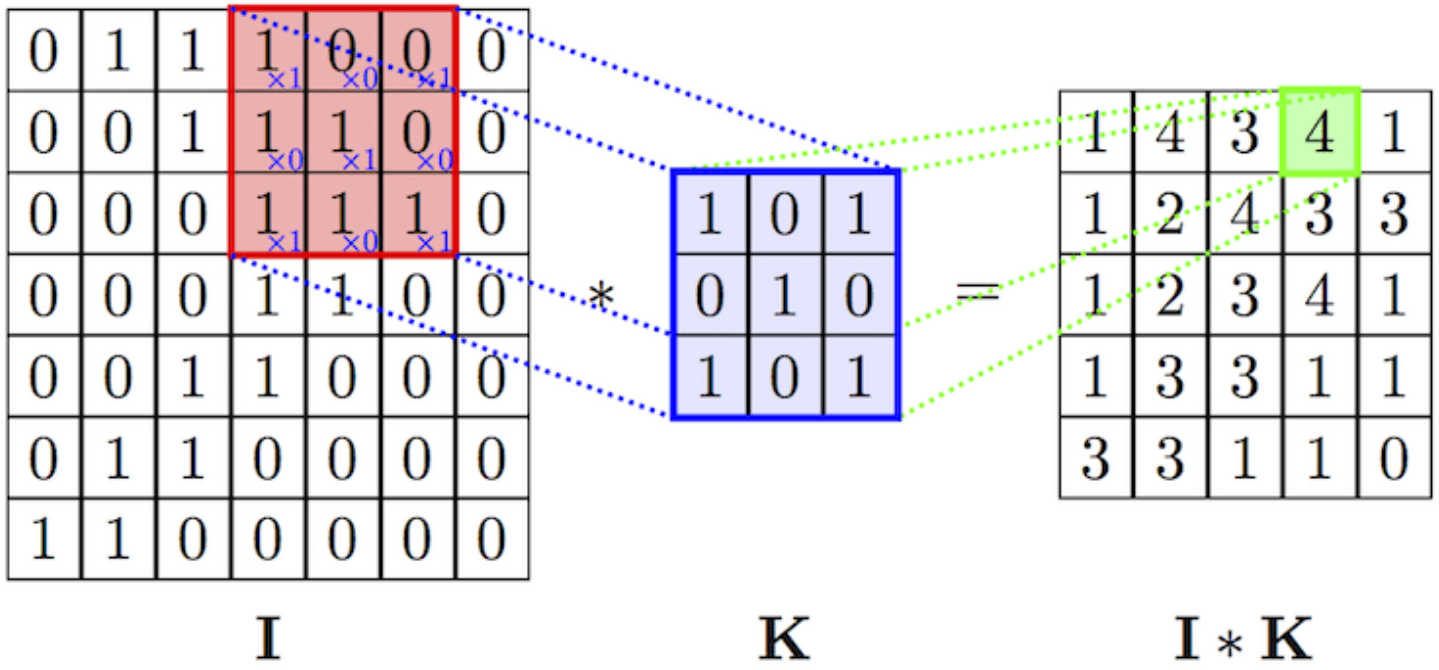
No smoothing

3 “Spatially local averaging” reduces noise



The traditional Computer Vision approach is through convolution of linear filters

So what is a convolution of a linear filter?

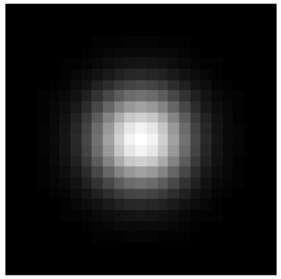


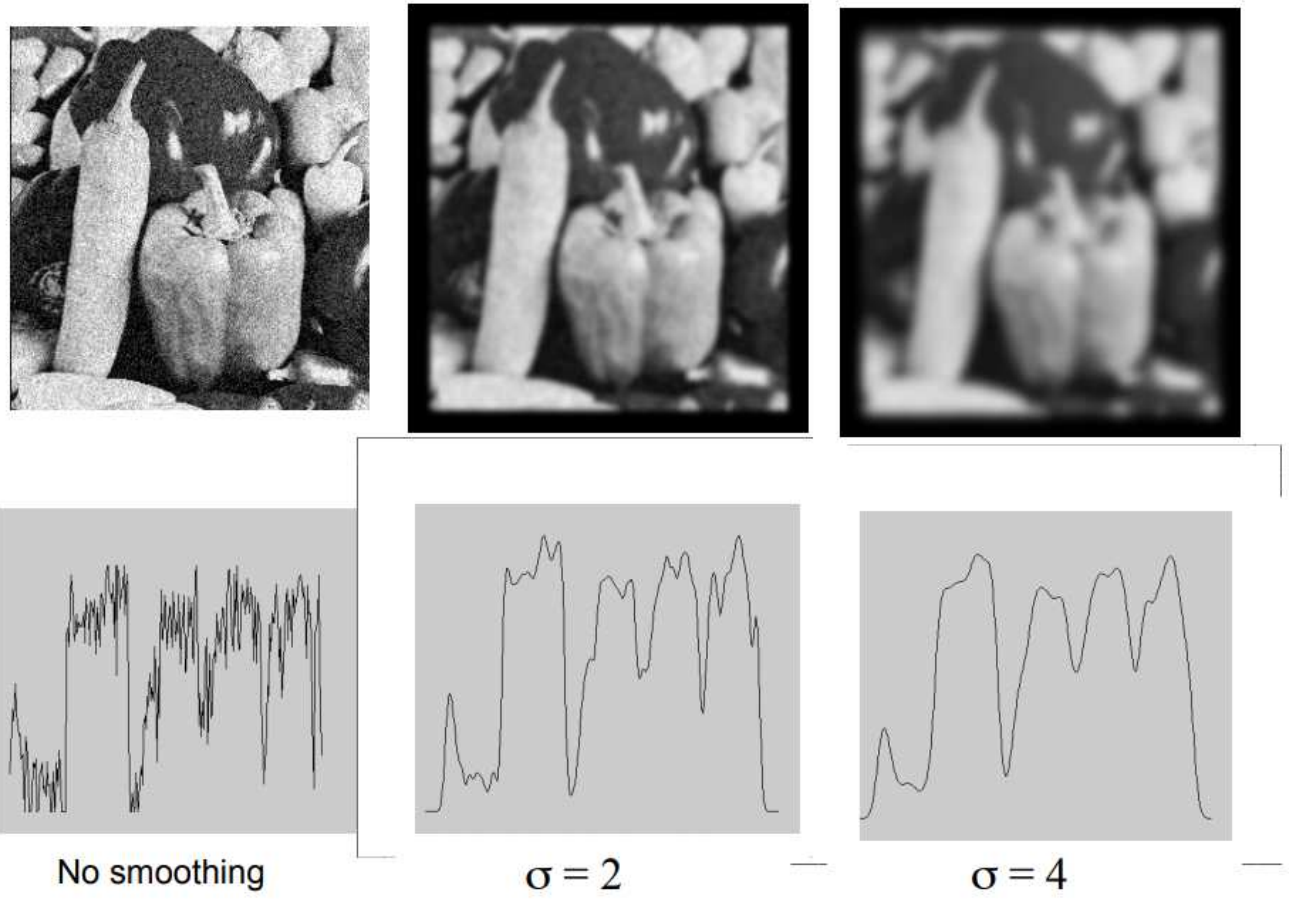
3 “Spatially local averaging” reduces noise

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

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, σ = 1



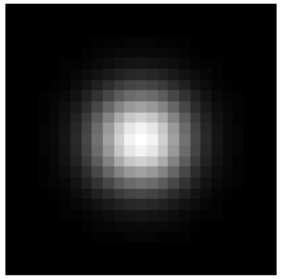


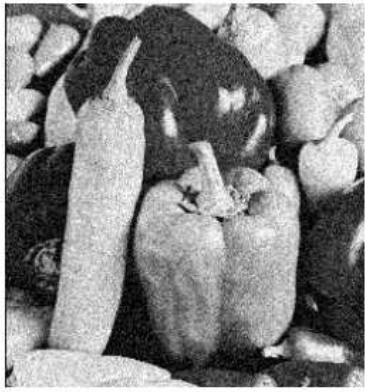
3 “Spatially local averaging” reduces noise

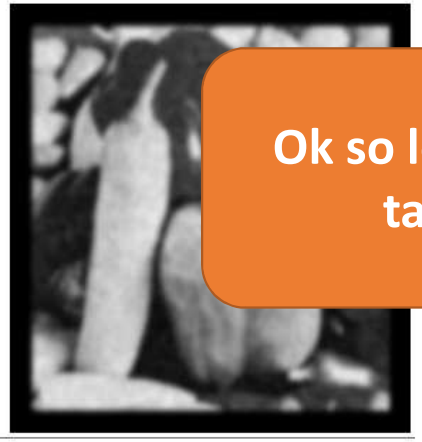
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

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

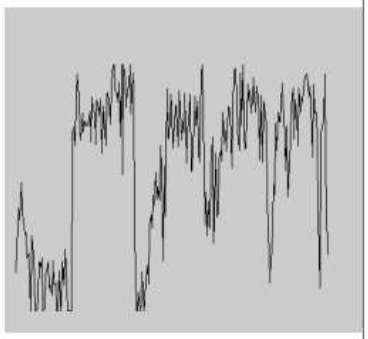
5 x 5, σ = 1



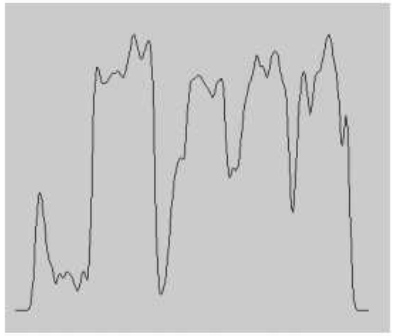




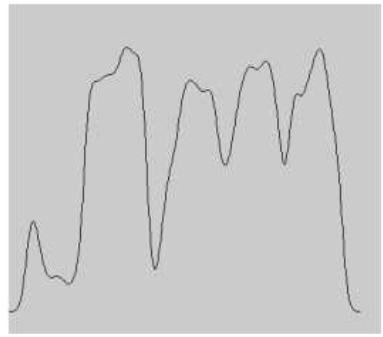
Ok so lets smooth and then take a derivative!



No smoothing



σ = 2



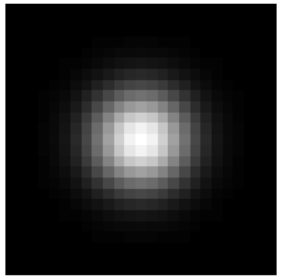
σ = 4

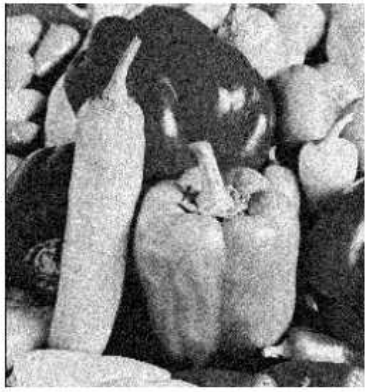
3 “Spatially local averaging” reduces noise

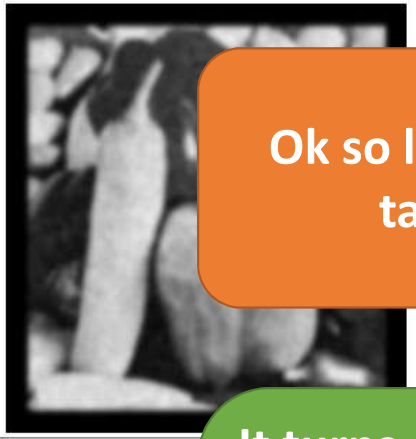
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

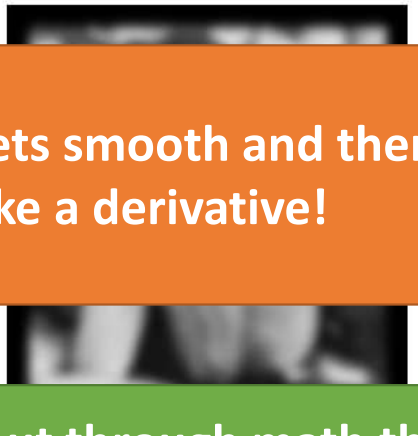
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

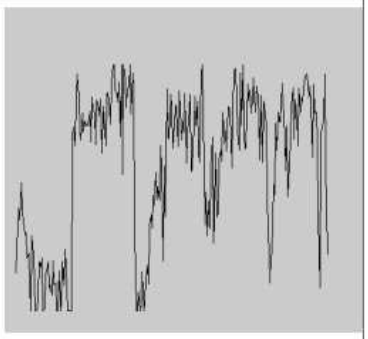


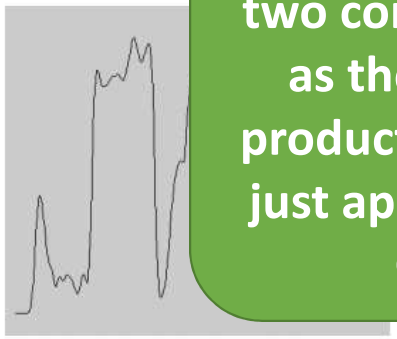


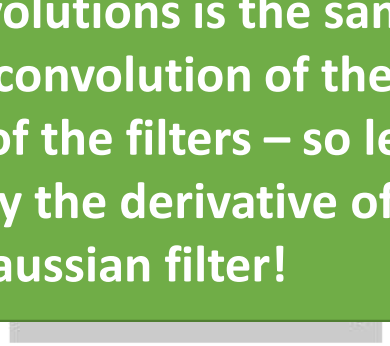




Ok so lets smooth and then take a derivative!







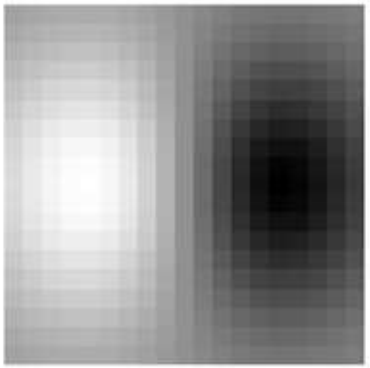
It turns out through math that two convolutions is the same as the convolution of the product of the filters – so lets just apply the derivative of a Gaussian filter!

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

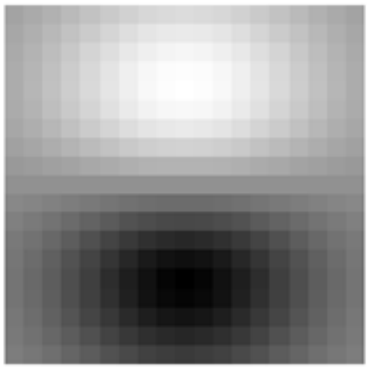
Derivatives increase noise so we can find edges using a derivative of Gaussian Filter

Applying the first derivative of Gaussian

$\frac{d}{dx}$

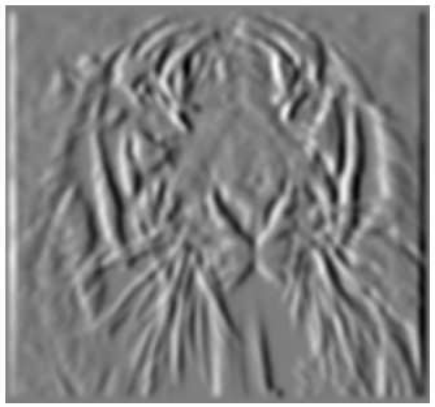


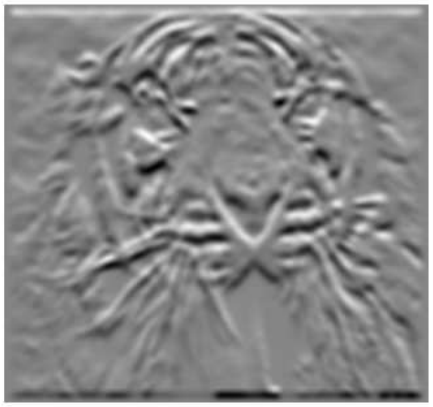
$\frac{d}{dy}$



I

$\frac{\partial I}{\partial x}$



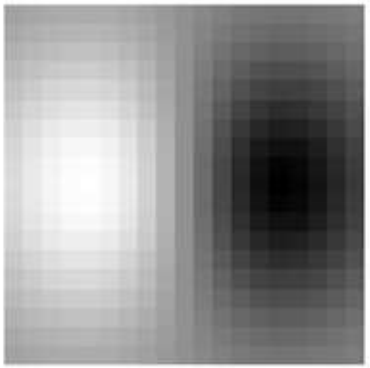


$\frac{\partial I}{\partial y}$

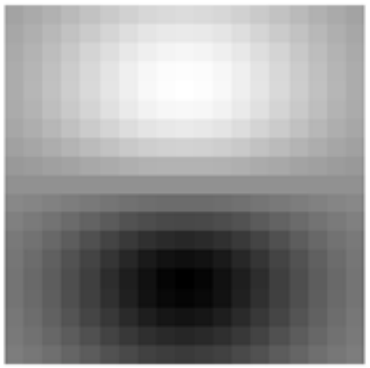
Derivatives increase noise so we can find edges using a derivative of Gaussian Filter

Applying the first derivative of Gaussian

$\frac{d}{dx}$

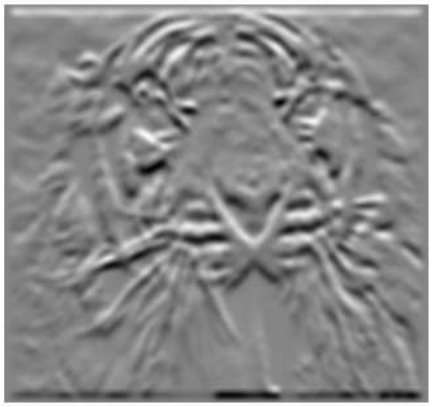


$\frac{d}{dy}$



I

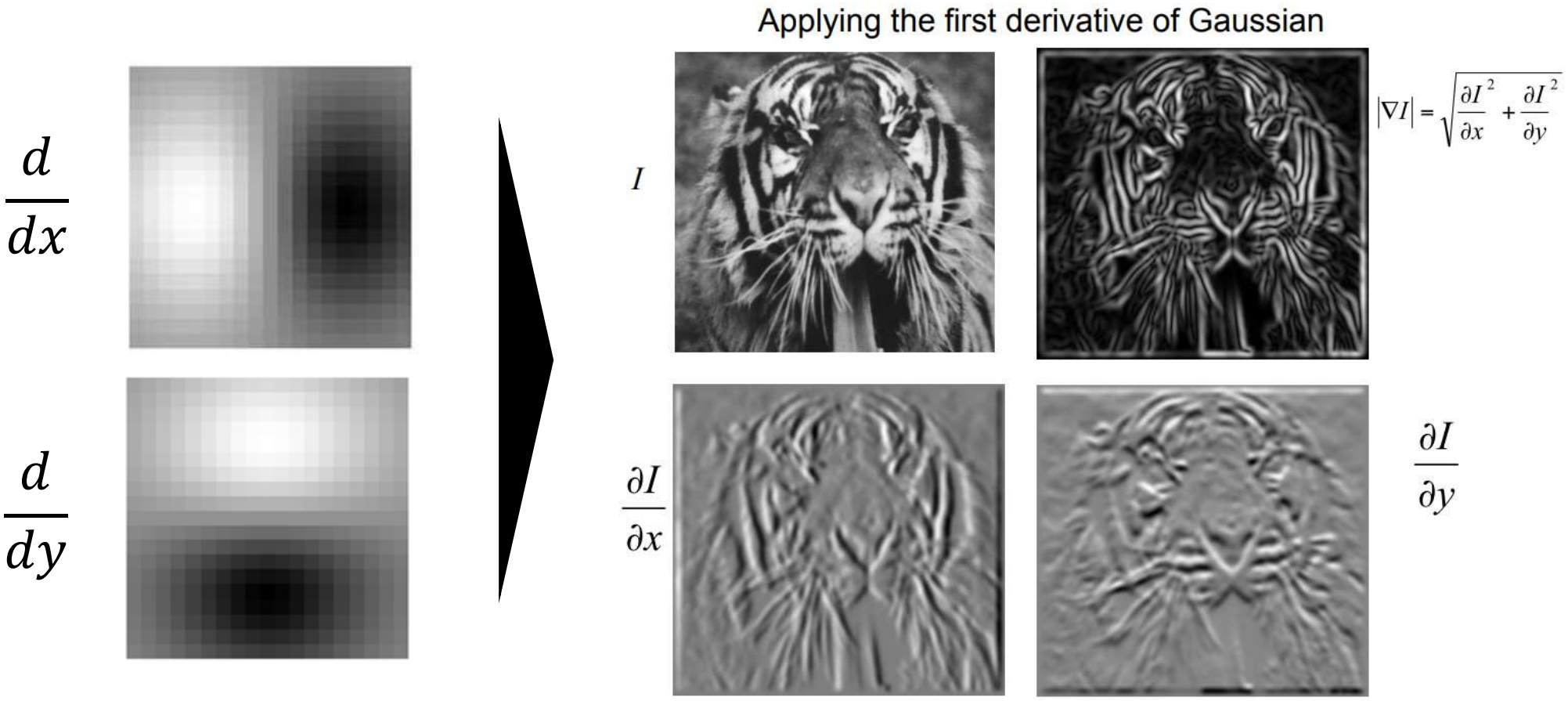
$\frac{\partial I}{\partial x}$



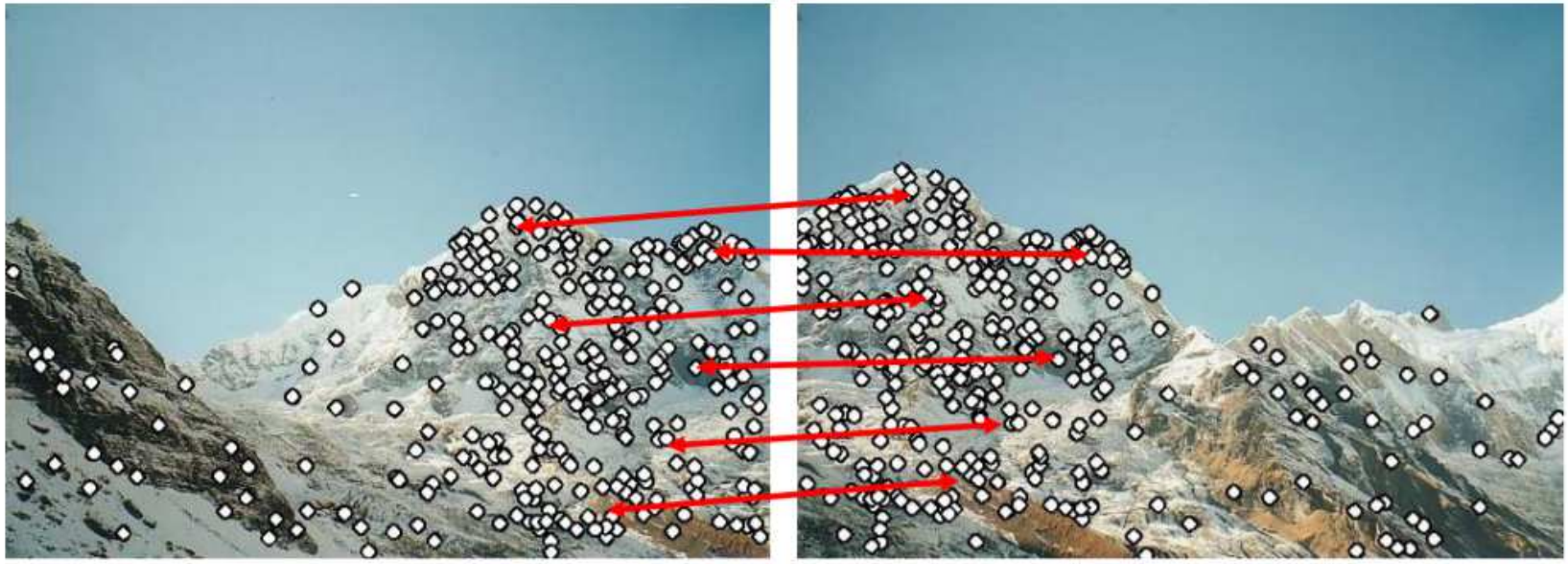
$\frac{\partial I}{\partial y}$

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

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



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



Step 1: extract features

Step 2: match features

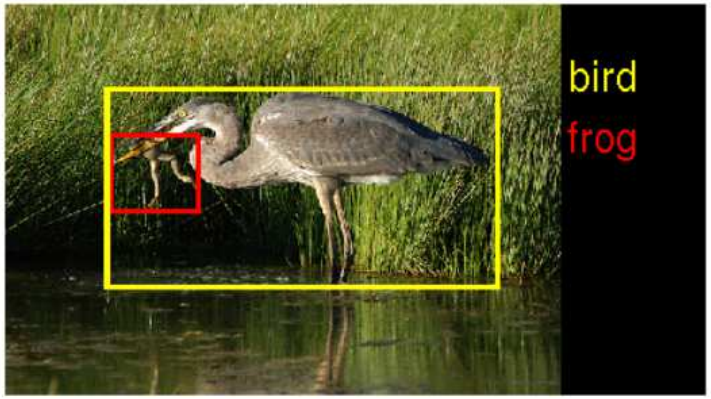
Various filters can be used to extract features to e.g., stich panoramas

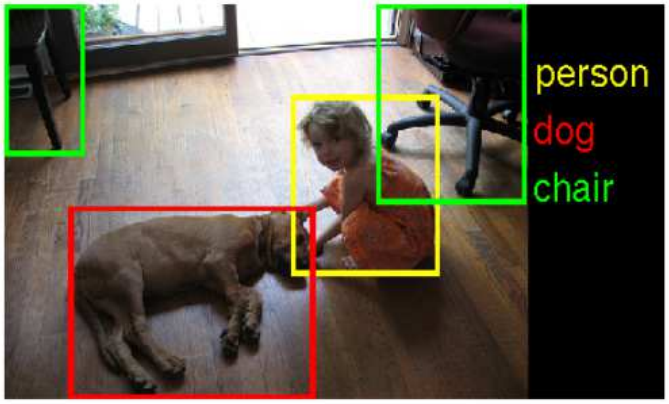
Step 1: extract features

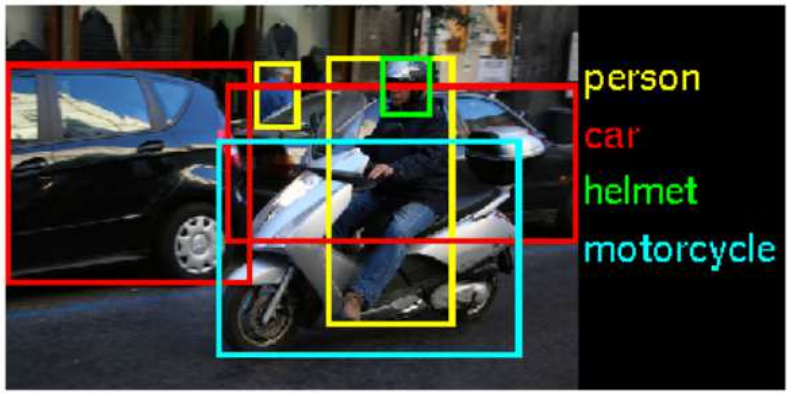
Step 2: match features

Step 3: align images

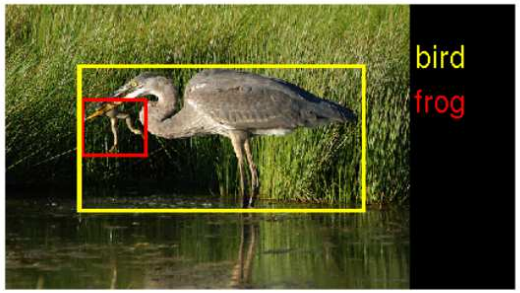
But what features should we use for object recognition?

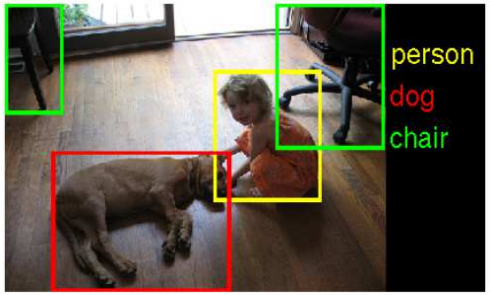


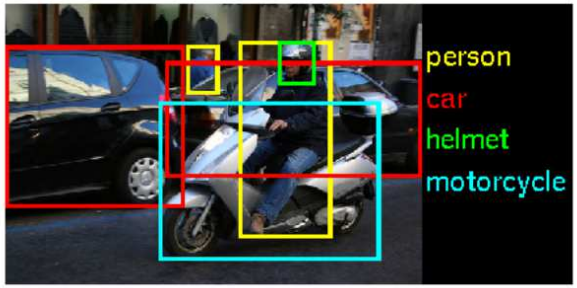




3 The ImageNet Challenge

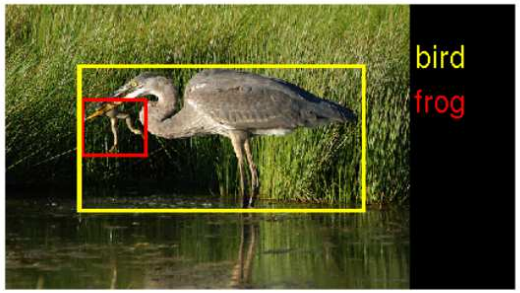


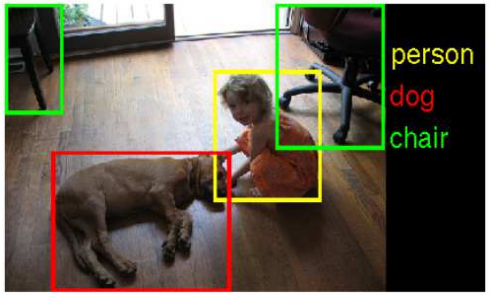


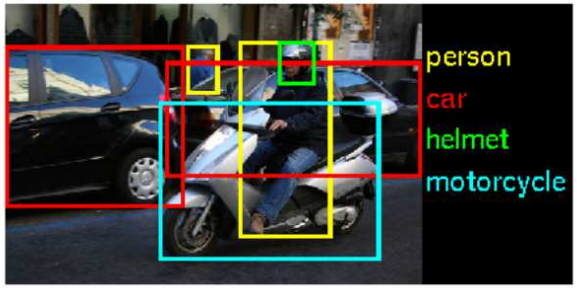


The ImageNet Challenge provided 1.2 million examples of 1,000 **labeled** items and challenged algorithms to learn from the data and then was tested on another 100,000 images

3 The ImageNet Challenge



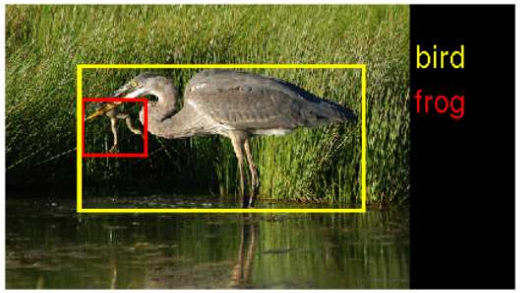


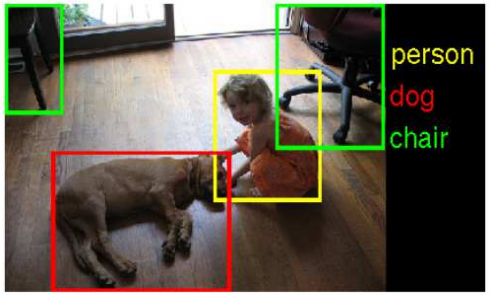


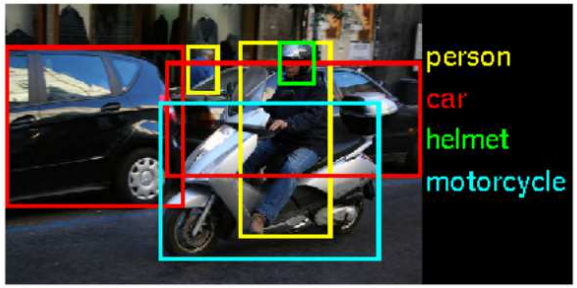
In 2010 teams had
75-50% error

In 2011 teams had
75-25% error

3 The ImageNet Challenge



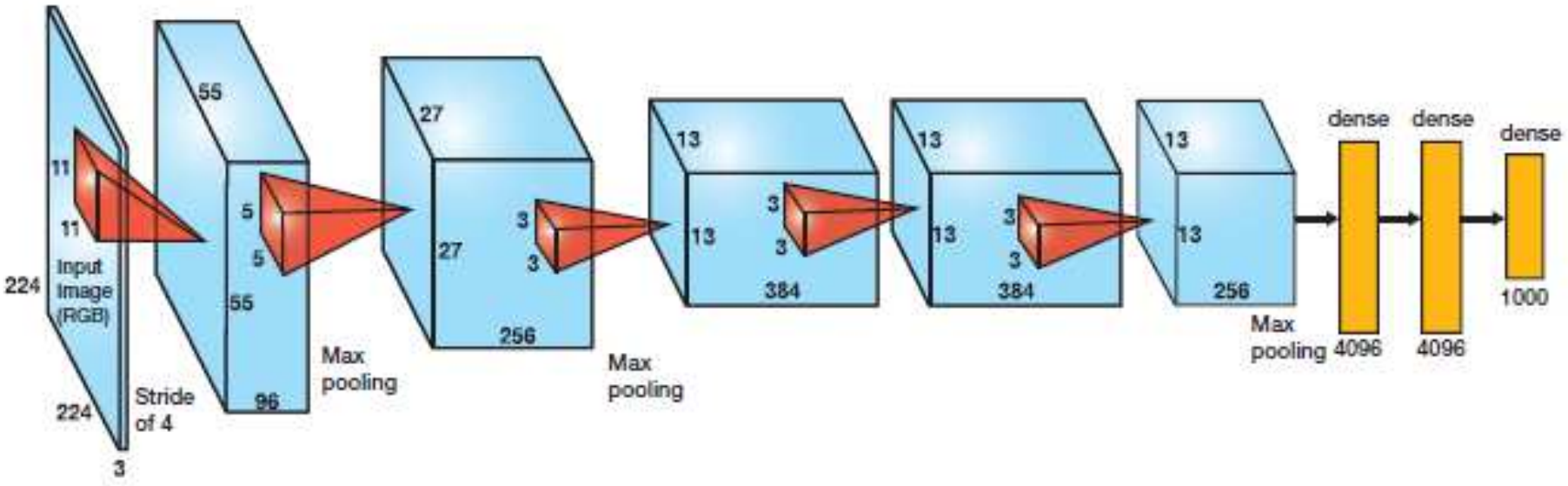




In 2012 still no team had less than 25% error barrier except **AlexNet at 15%**

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

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/

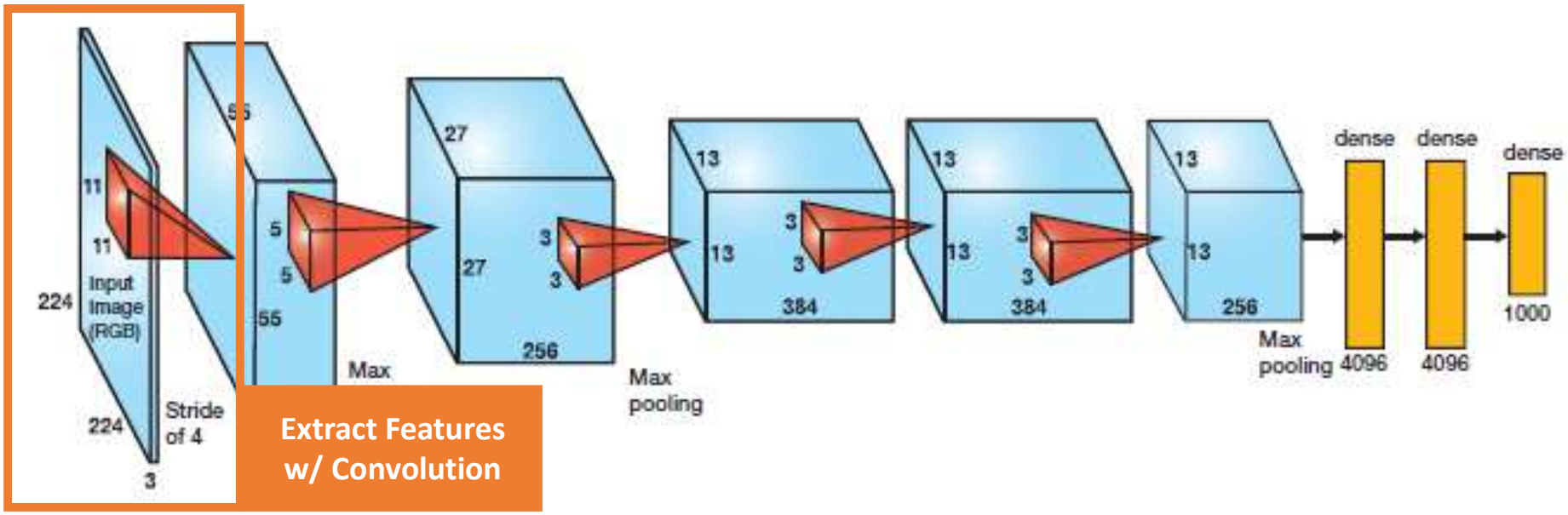
Deep learning automates the design, selection and extraction of features, with amazing results

Traditional Computer Vision Flow

Deep Learning Flow

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

AlexNet: the first widely successful application of deep learning

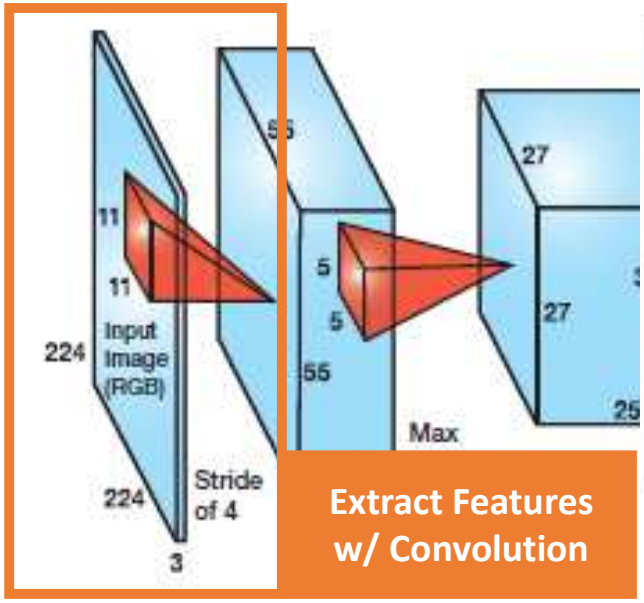


https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

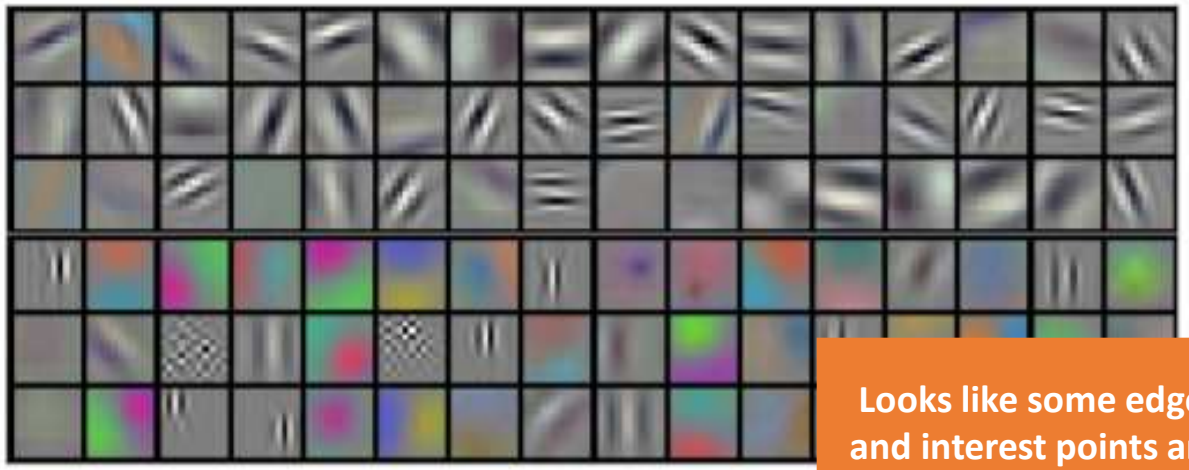
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

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

AlexNet: the first widely successful application of deep learning



Extract Features w/ Convolution



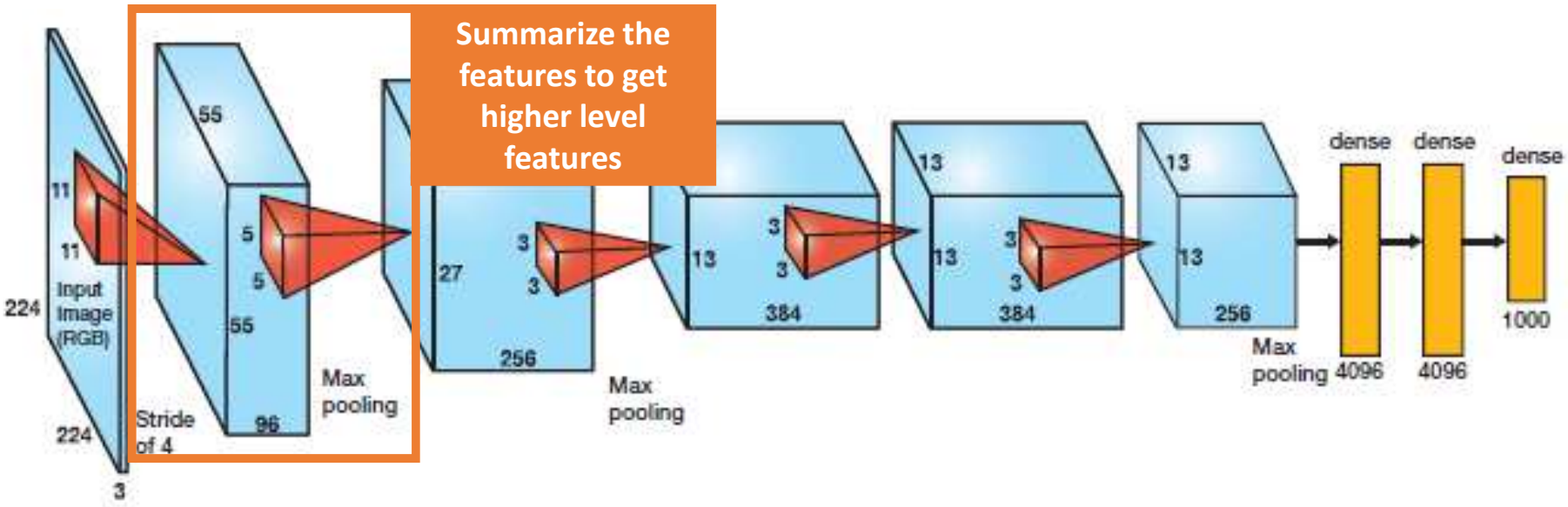
Looks like some edges and interest points and important color patterns

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

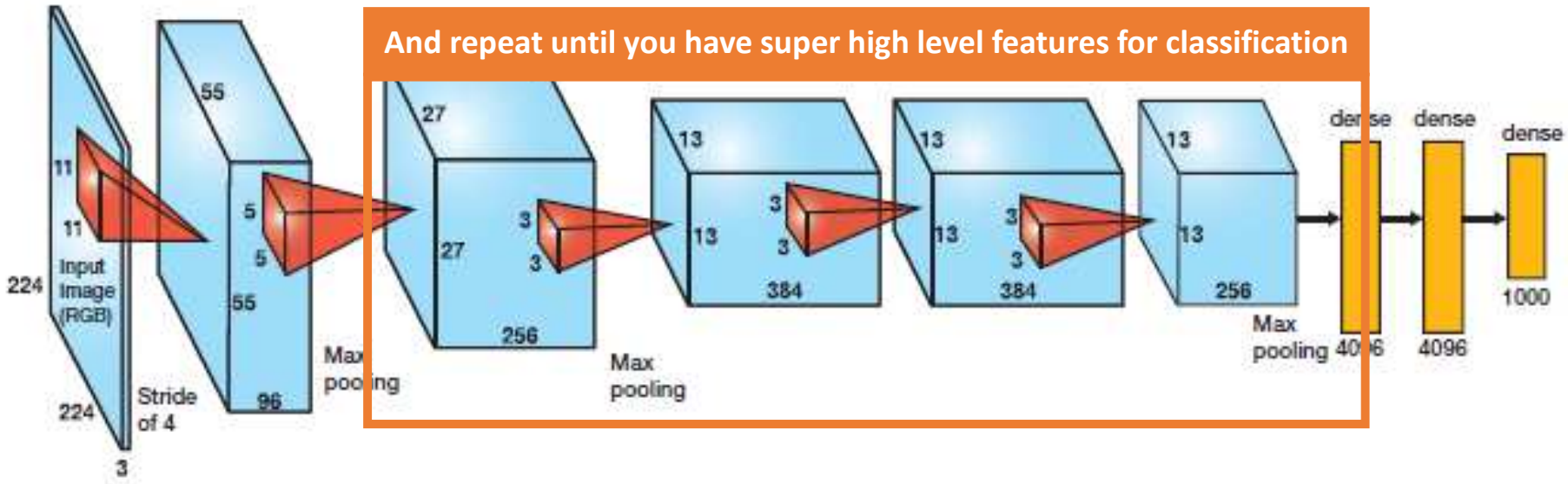


https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

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

AlexNet: the first widely successful application of deep learning

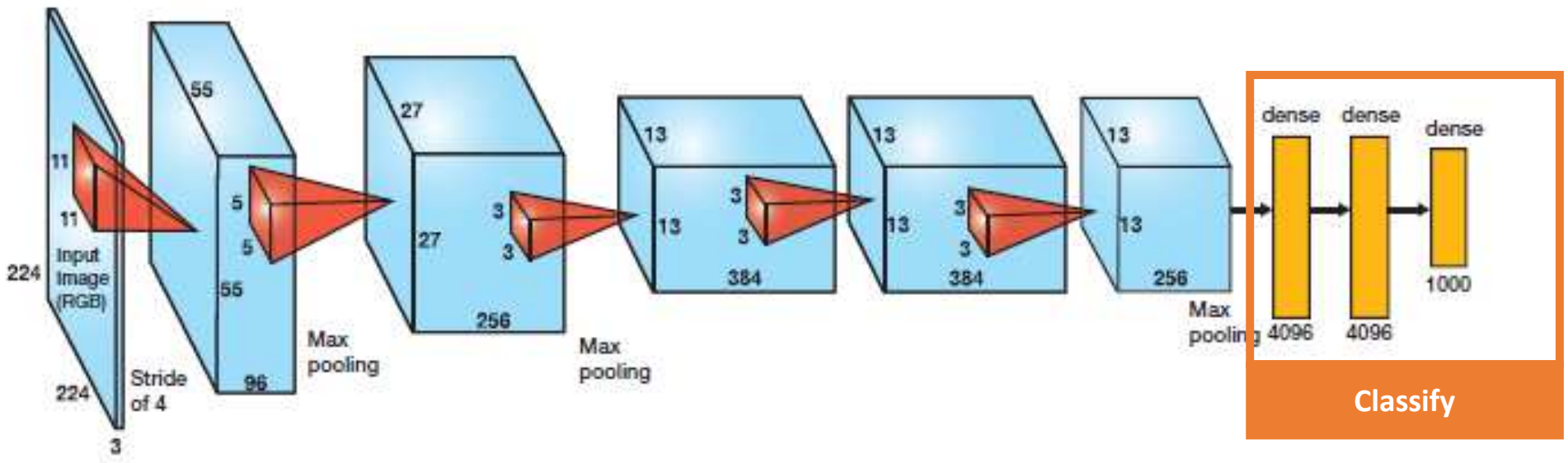


https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

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

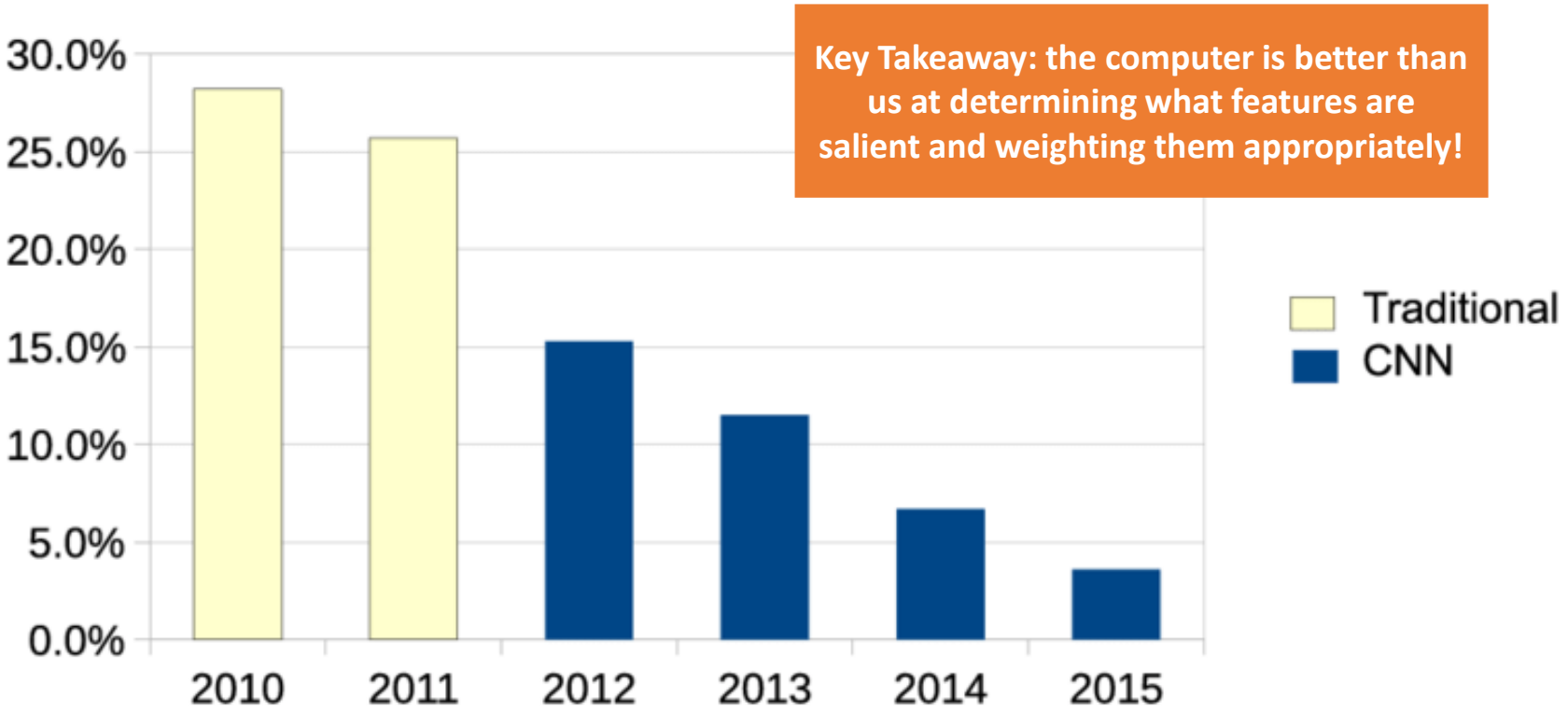
AlexNet: the first widely successful application of deep learning



https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

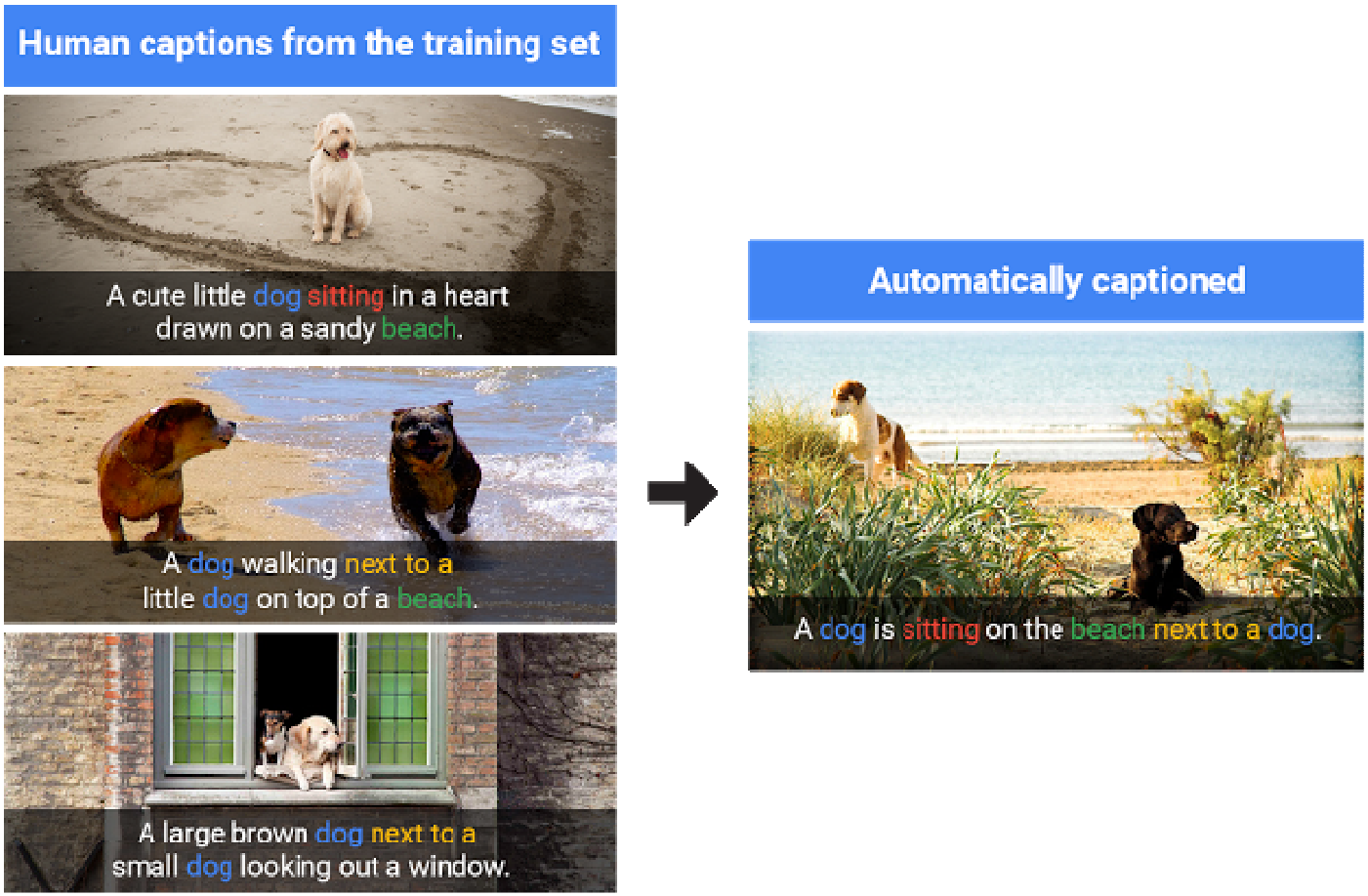
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

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



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!



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

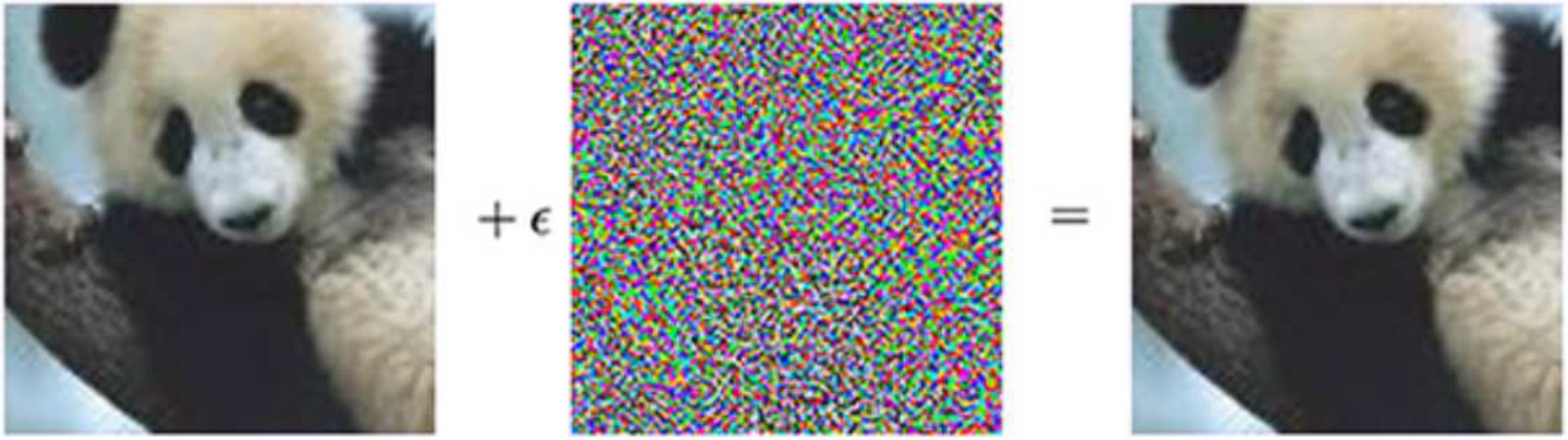
3 And can be used to track objects in real time

3 What might be the downside to using NNs?

3 For one, NNs can be tricked by adversarial markings

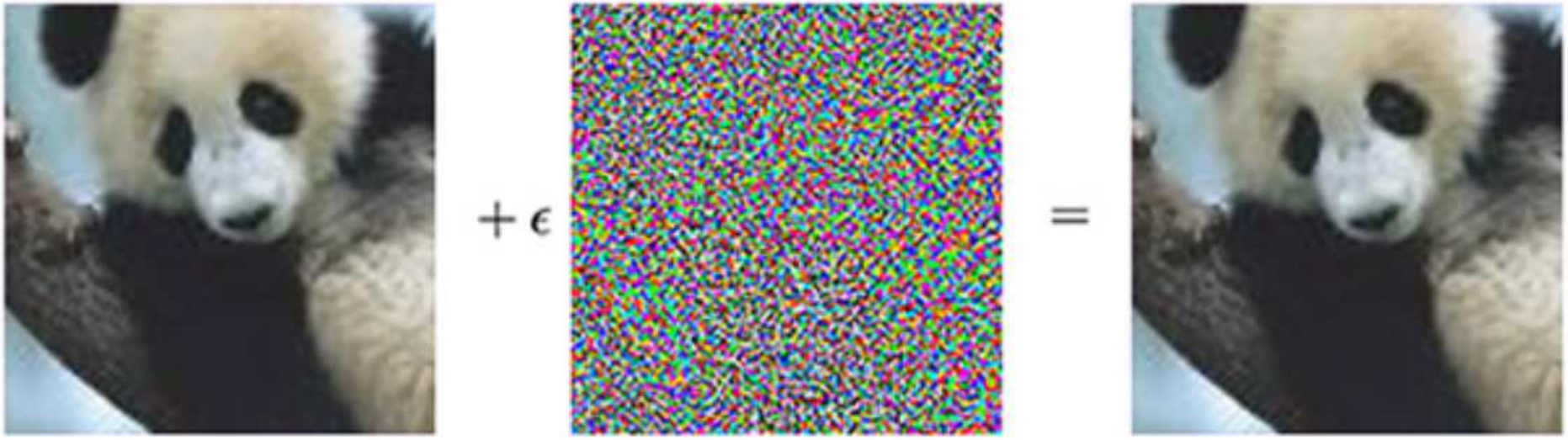
3 For one, NNs can be tricked by adversarial markings

Ackerman “Hacking the Brain With Adversarial Images”



3 For one, NNs can be tricked by adversarial markings

Ackerman “Hacking the Brain With Adversarial Images”



“panda”

57.7% confidence

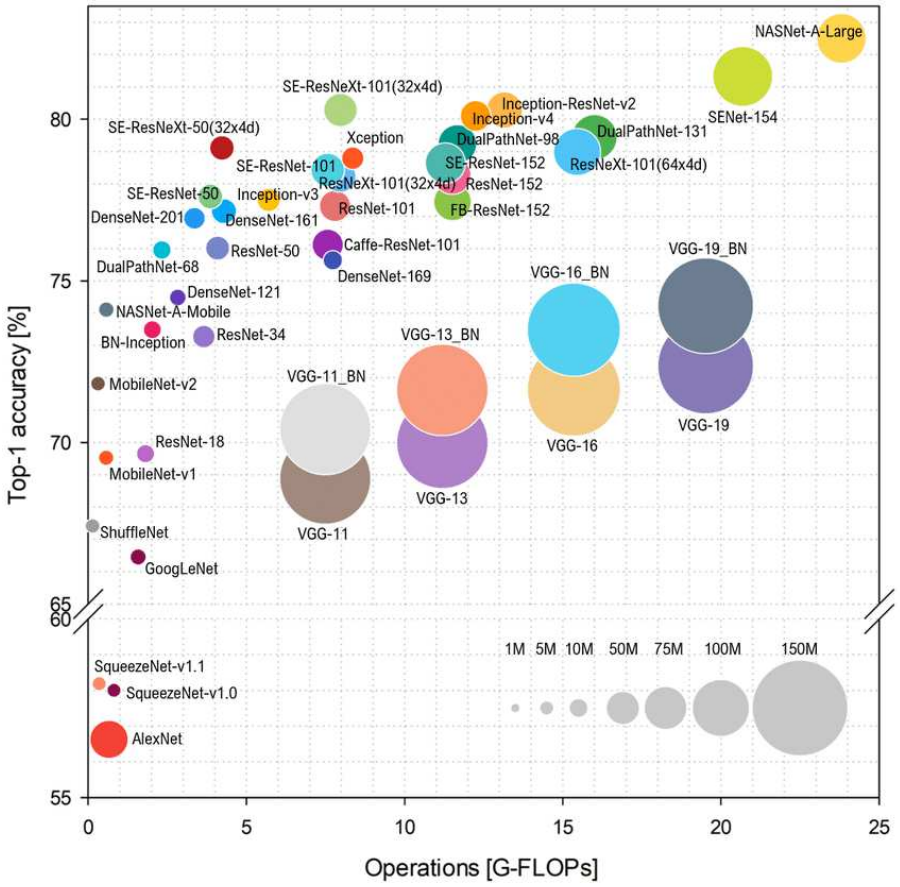
There is **no model** of the world semantically just mathematically

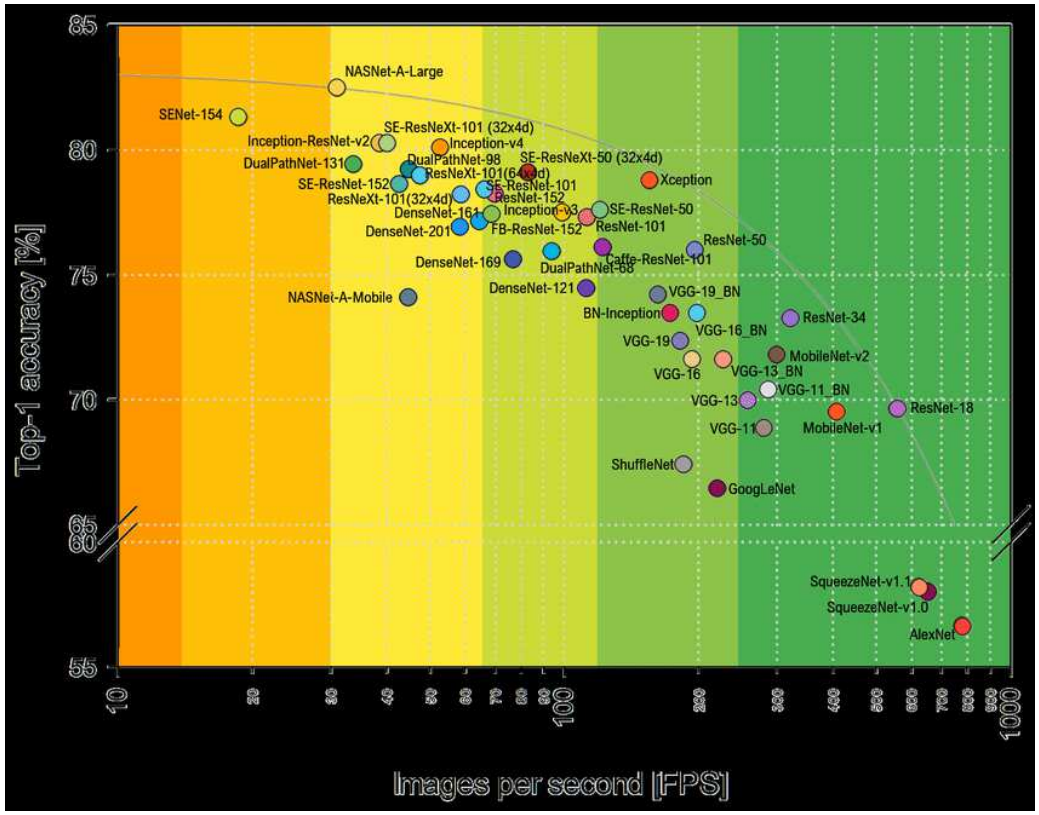
“gibbon”

99.3% confidence

3

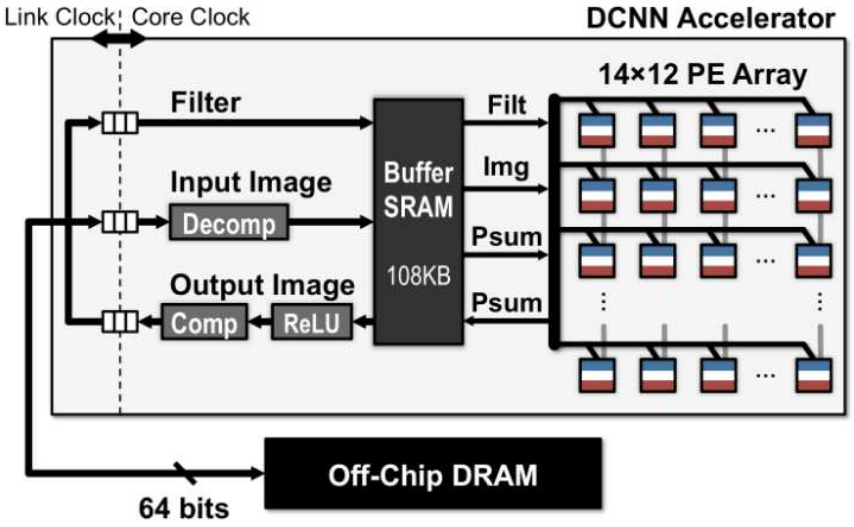
Second, (good) NN models are (often) large and expensive to train and compute



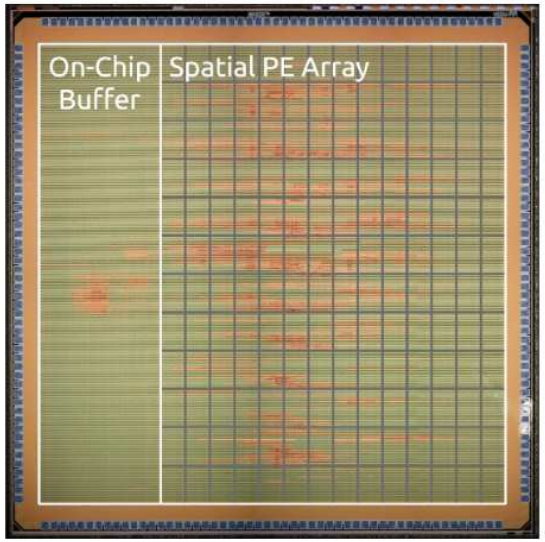


[Bianco et. al. Benchmark Analysis of Representative Deep Neural Network Architectures]

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



Eyeriss Architecture



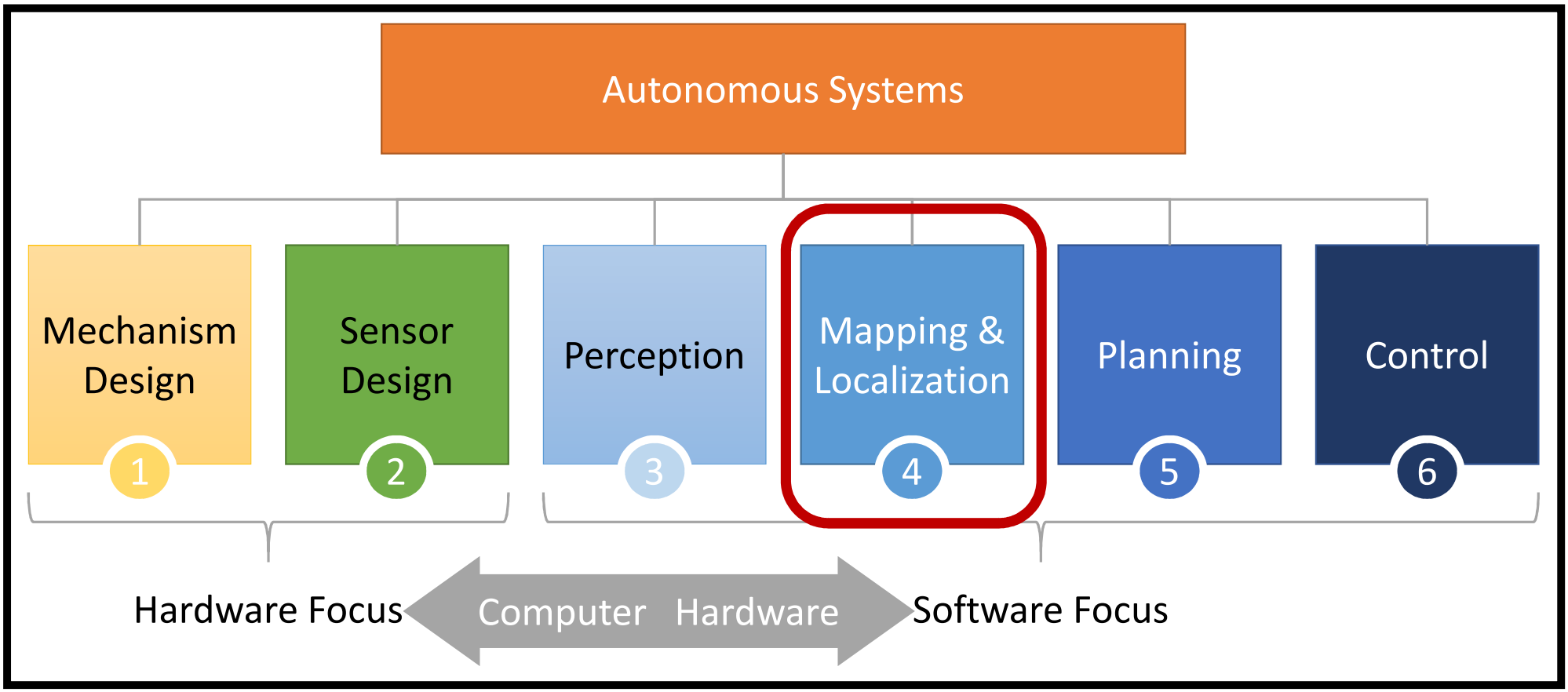
Die Photo

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

3 Key Takeaways:

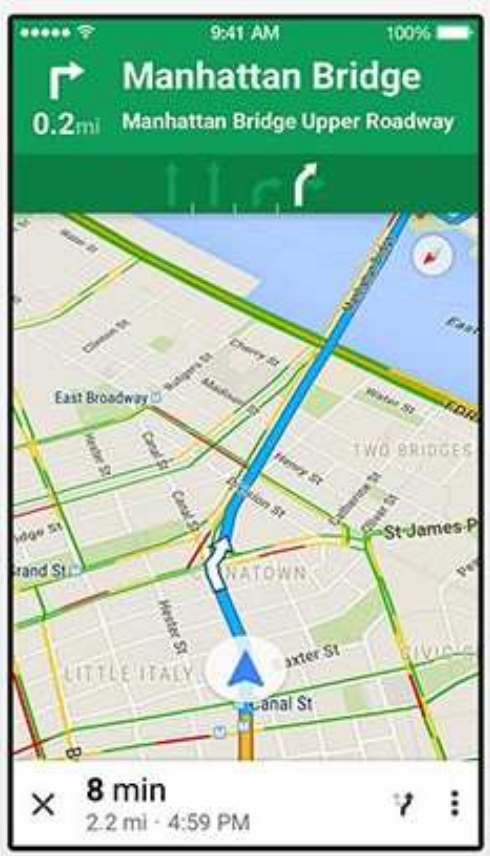
1. As of today it seems like CNNs that automate the design and summary of salient features via convolution are the way to go
   - But/and will need specialized NN running on specialized accelerator chips to get them small enough to fit on small power constrained autonomous systems (e.g., small drones)
   - 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

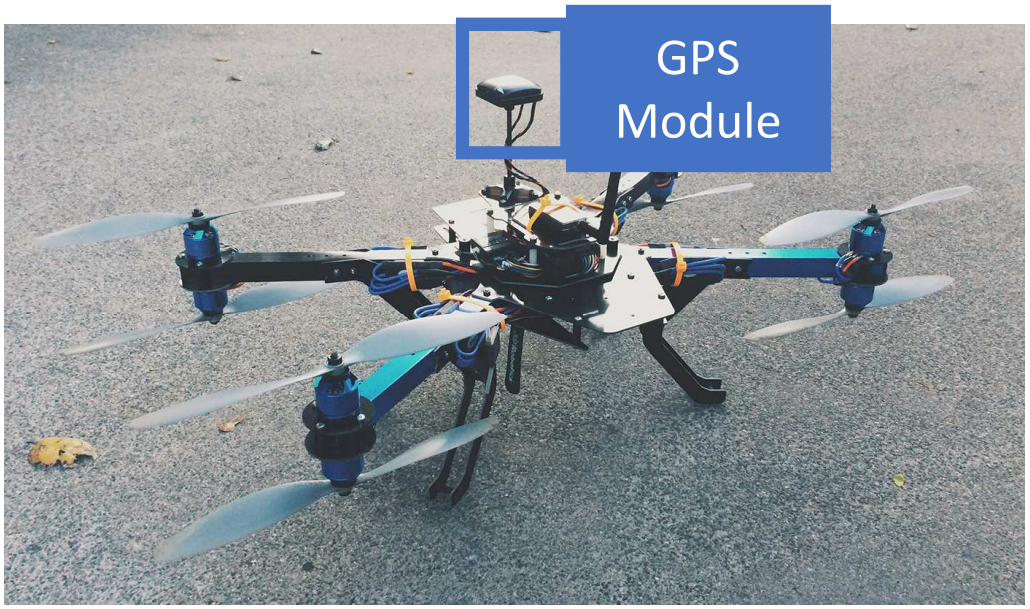


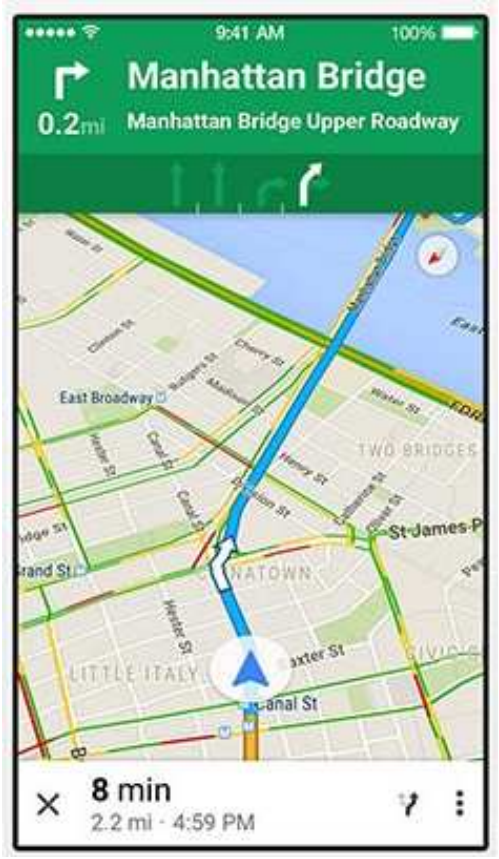
4

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

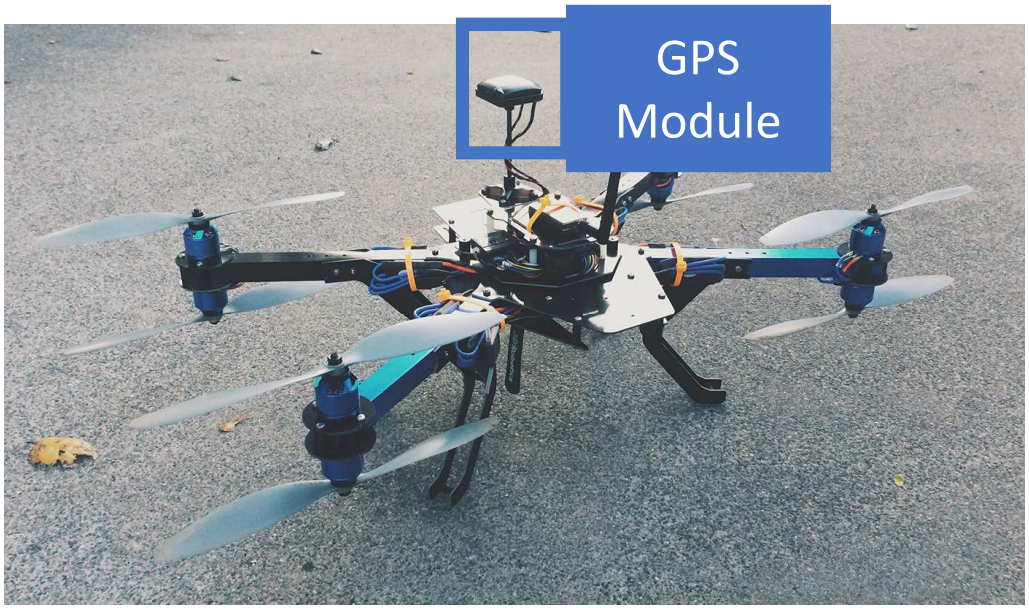


4 GPS provides a good idea of where a robot is globally…





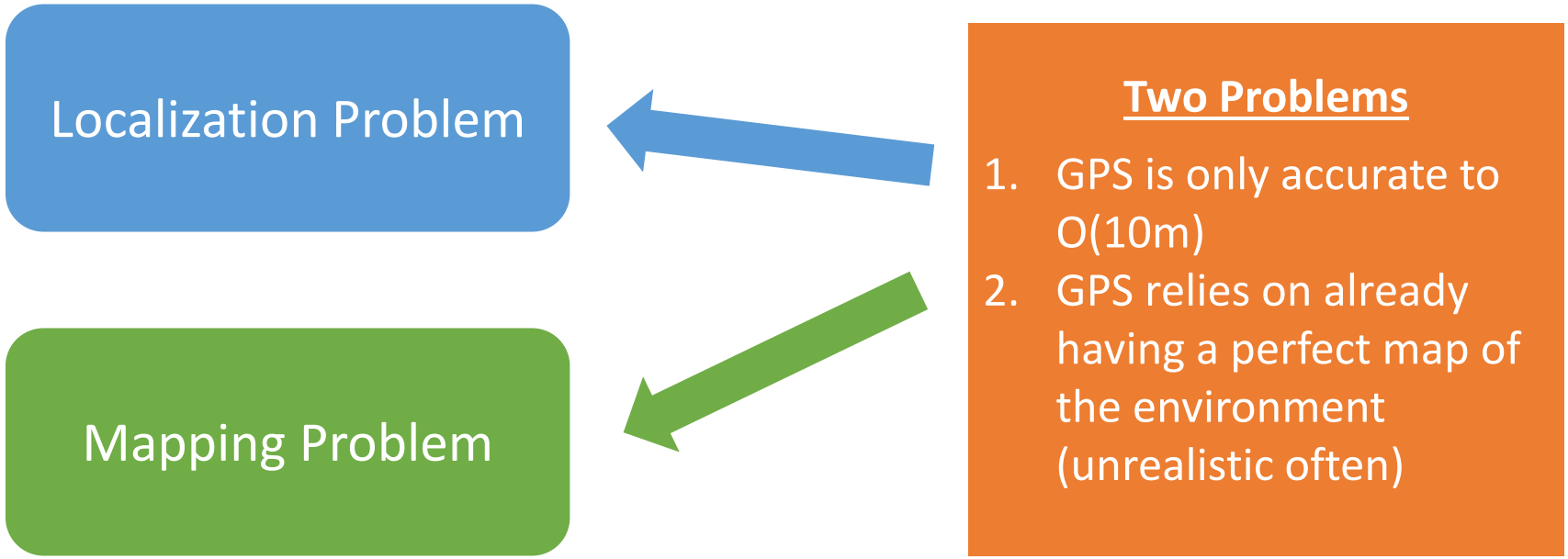
4 … but isn't very accurate locally and requires a map



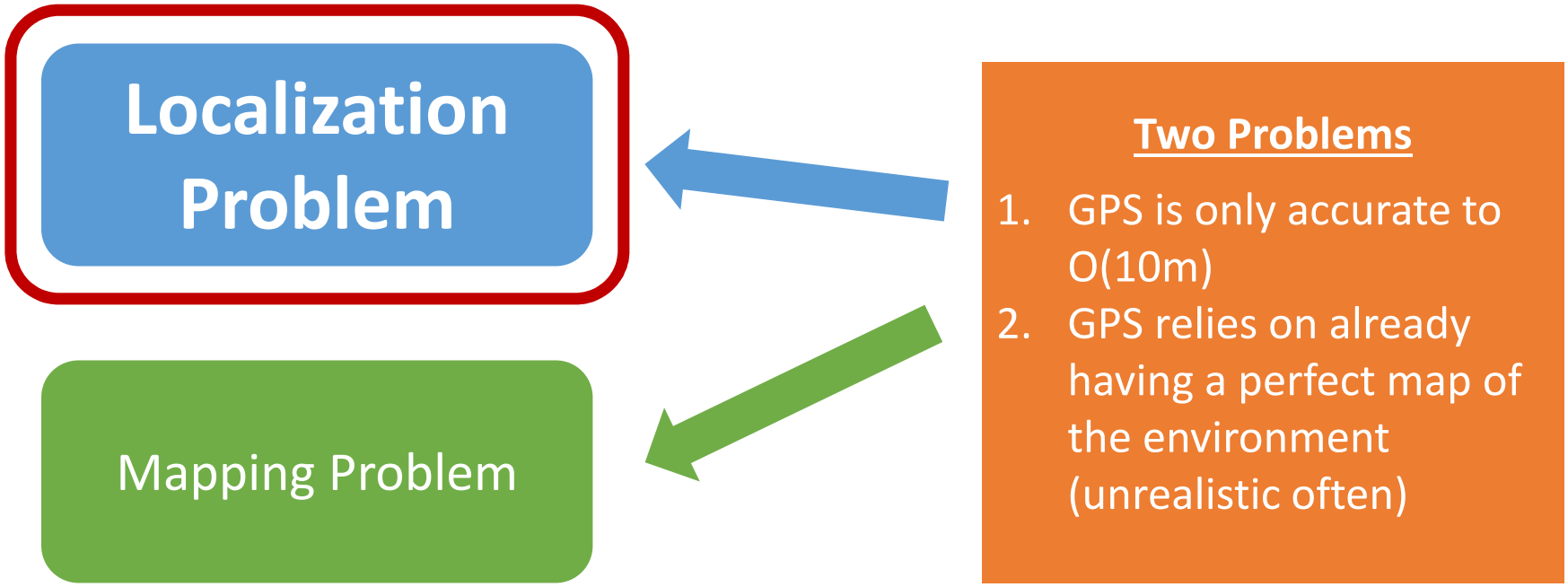
Two Problems

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

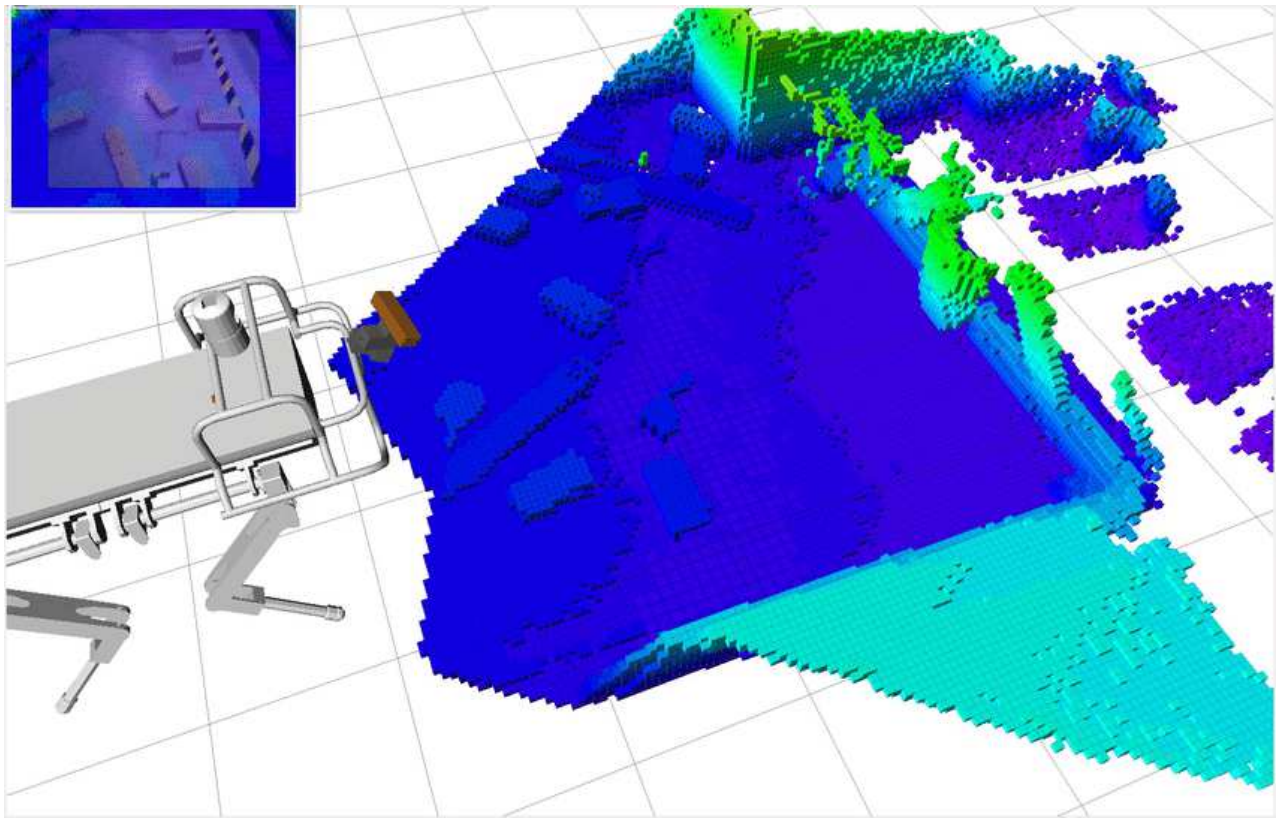
4 … but isn't very accurate locally and requires a map



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



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

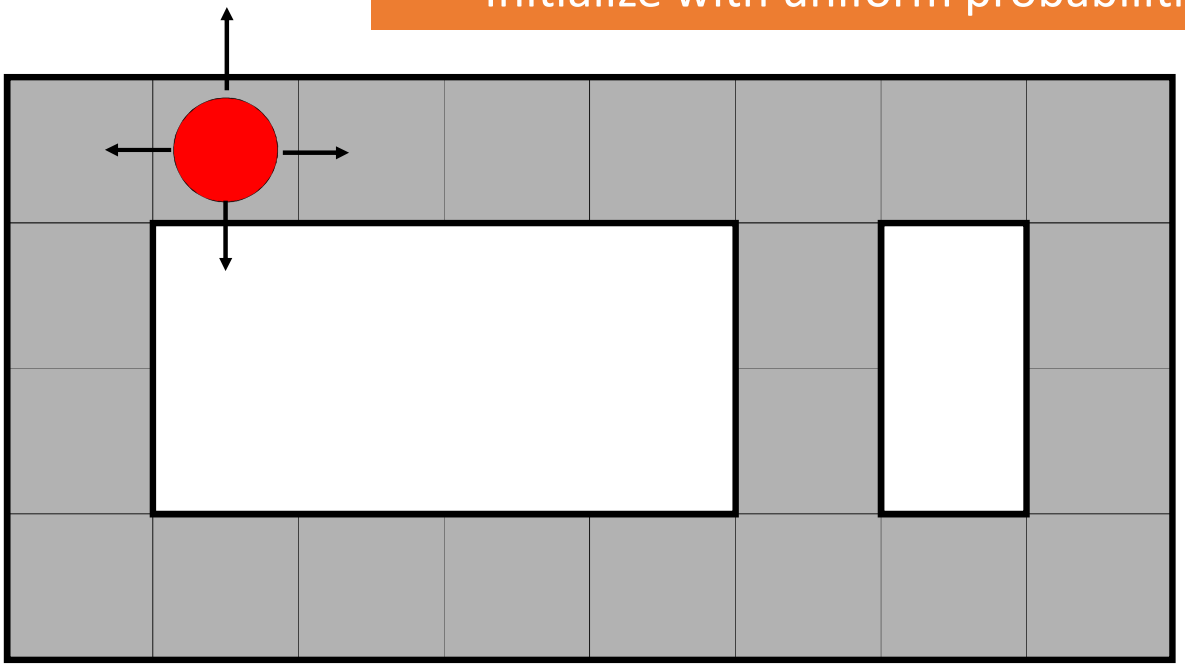


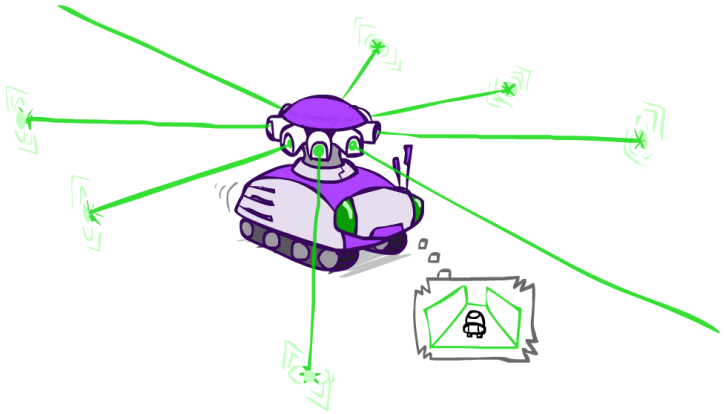
4 So what can we do?

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

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

Initialize with uniform probabilities everywhere





t=0

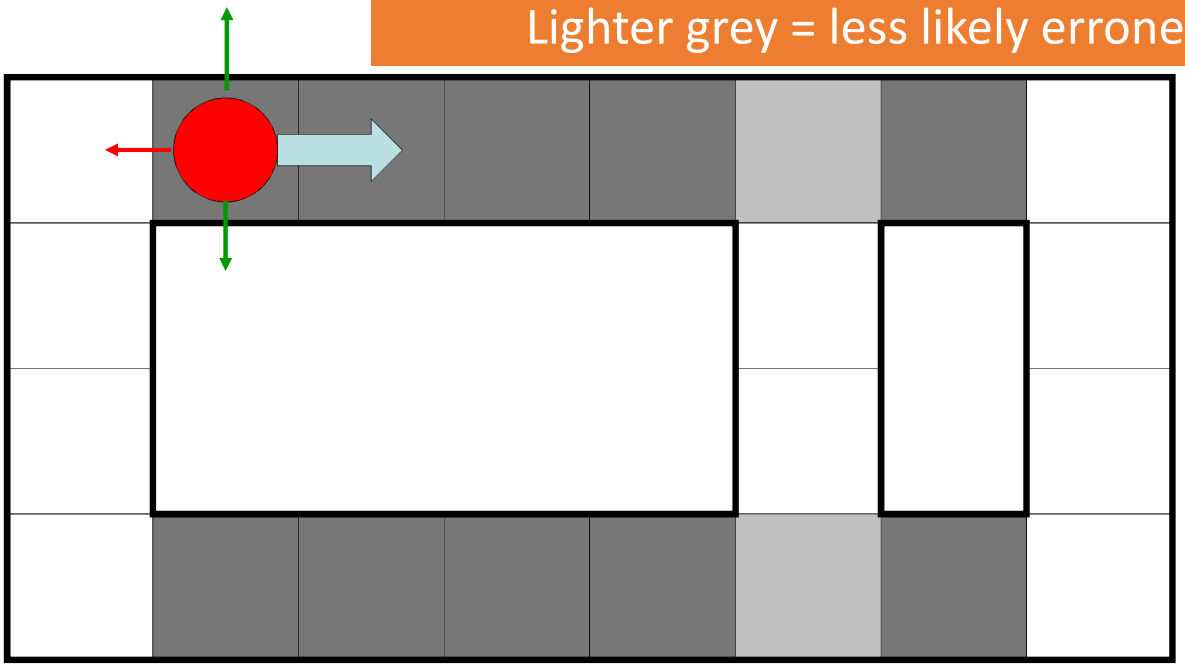
Probabilities

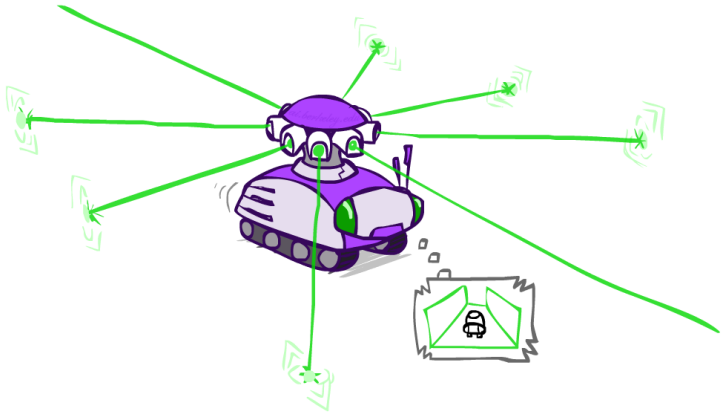
0 1

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





t=1

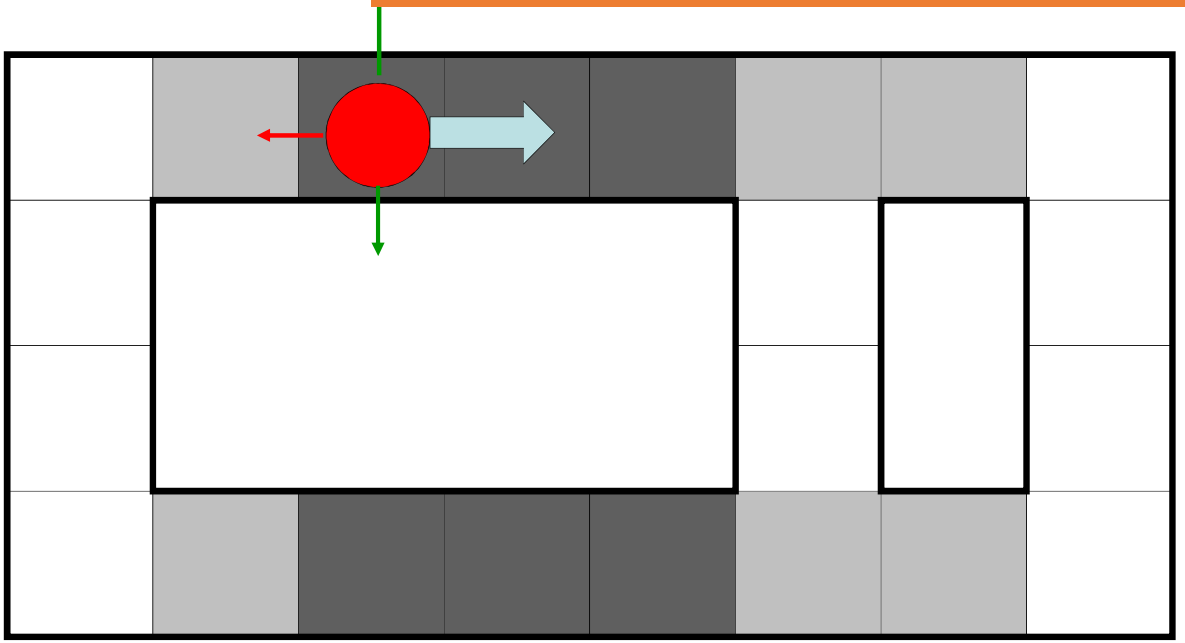
Probabilities

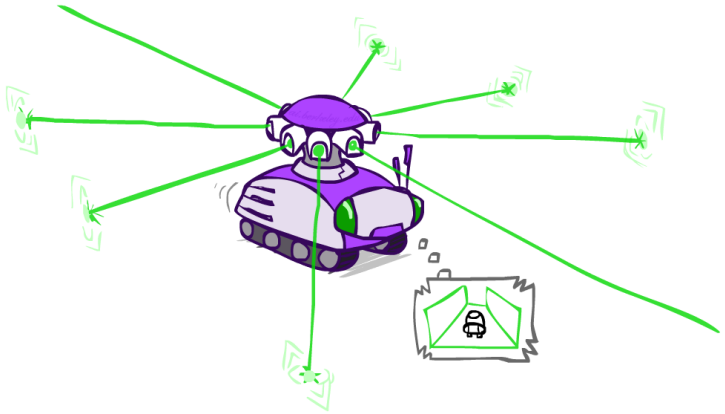
0 1

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

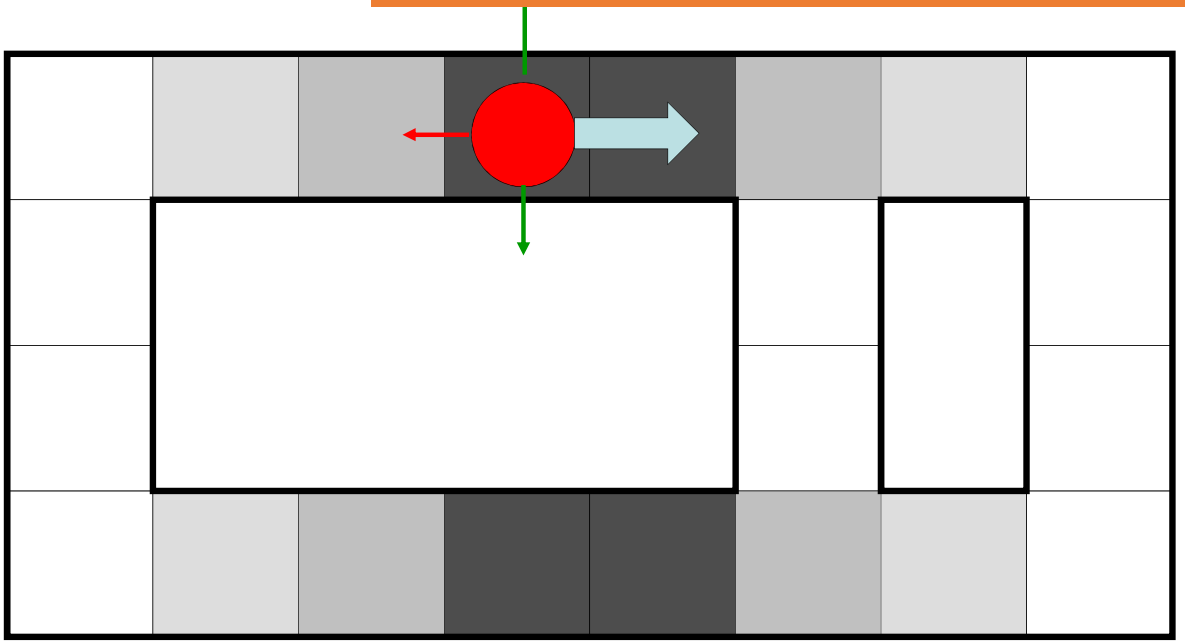
Probabilities

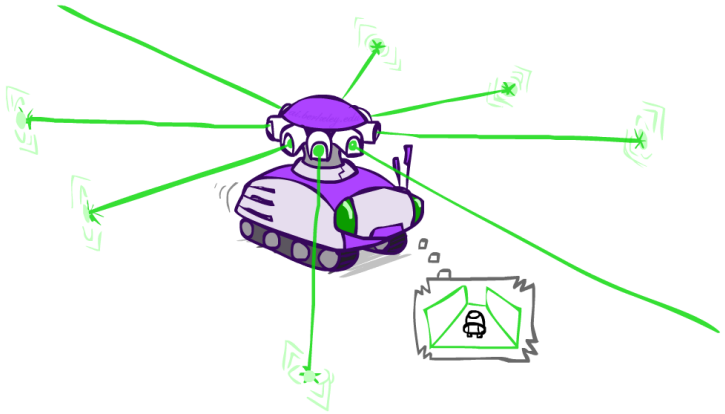
0 1

Example from Michael Pfeiffer

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

Update for motion and next sensor reading





t=3

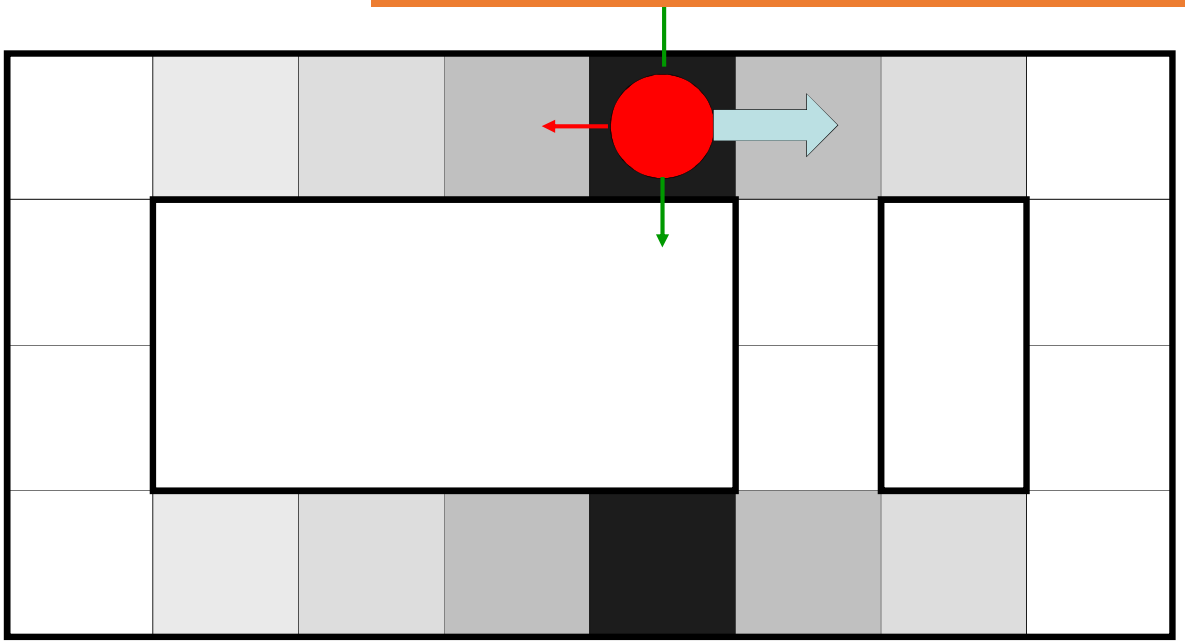
Probabilities

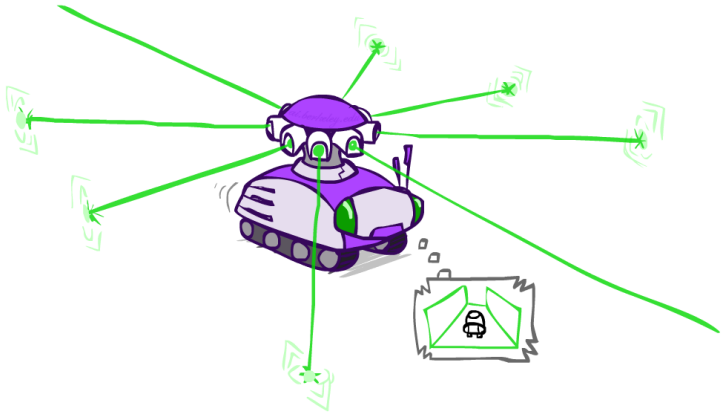
0 1

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

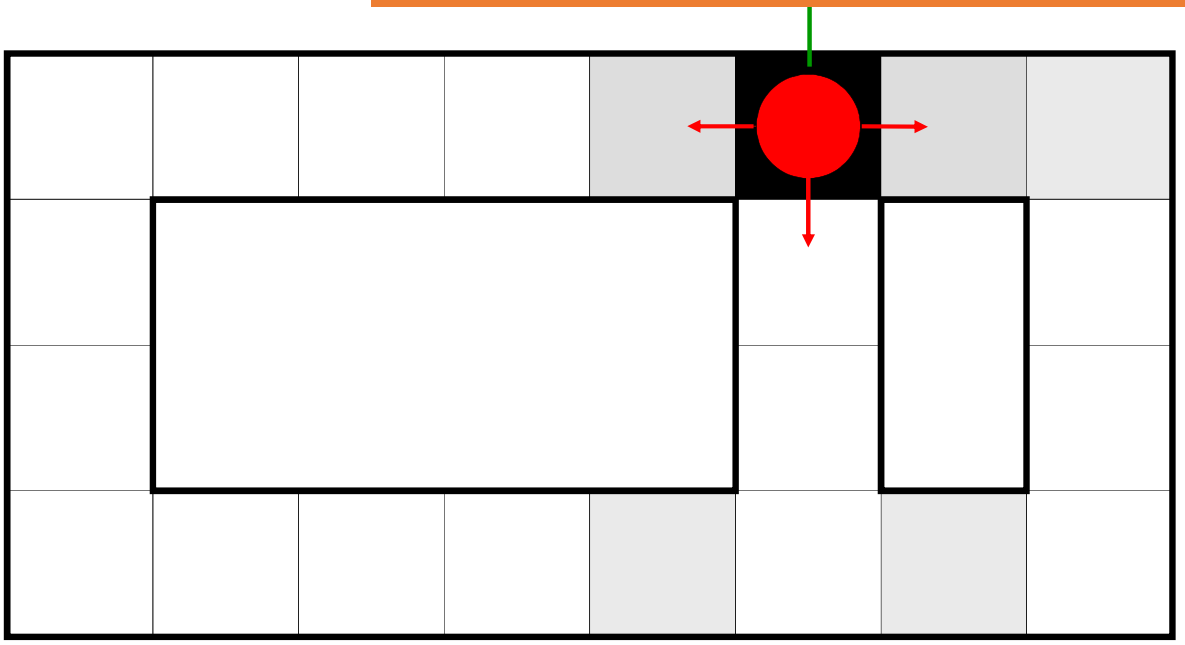
Probabilities

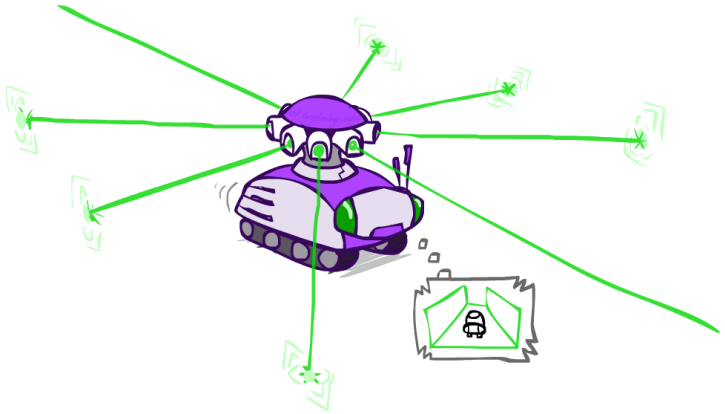
0 1

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

0 1

Example from Michael Pfeiffer

4

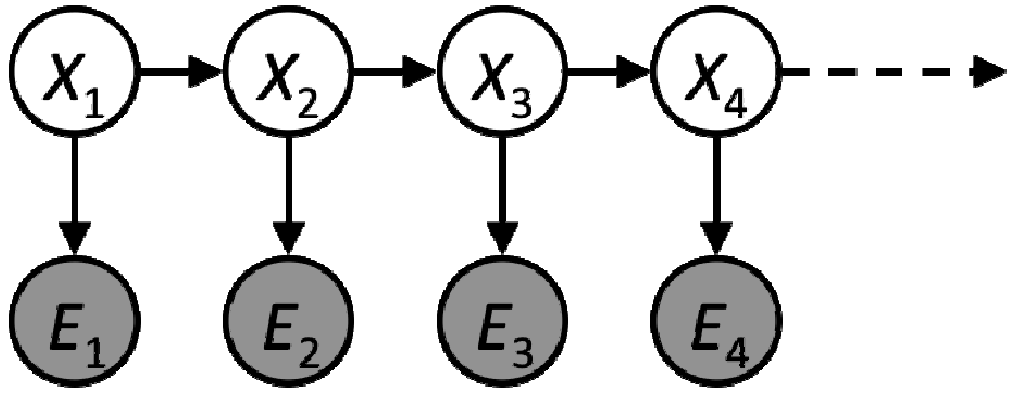
One approach is to model the probability of being in a given state with a Hidden Markov Model

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

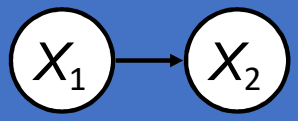


Report the **mean** of the **Belief State** (which is a probability distribution) as our current best estimate of the state

4

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

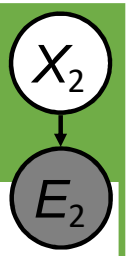
Time Update



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

Based on a model of physics usually

Evidence Update



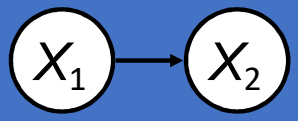
$$P(x_{t+1}|x_0, e_1 \dots e_{t+1}) \propto \int P(x_{t+1}|x_0, e_1 \dots e_t) * P(e_{t+1}|x_{t+1})$$

Based on a model of the sensor data usually

4

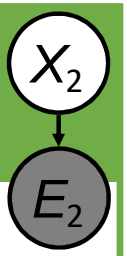
The Kalman Filter updates the belief state (a probability distribution) for the passage of time and for evidence

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)

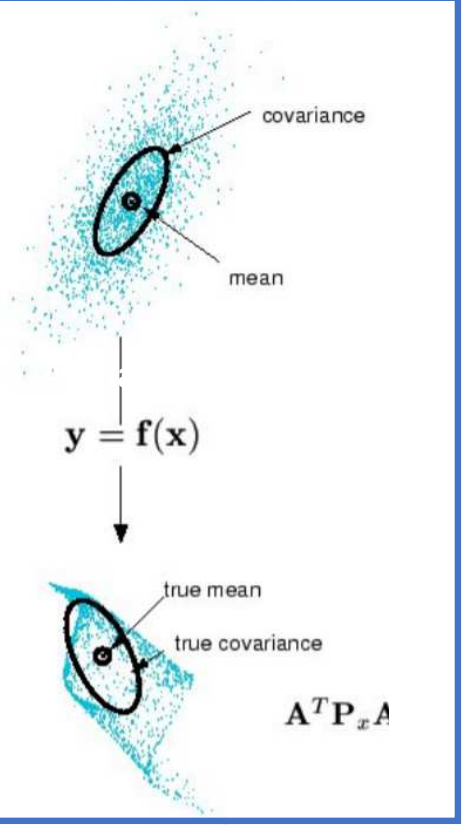
usually

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)

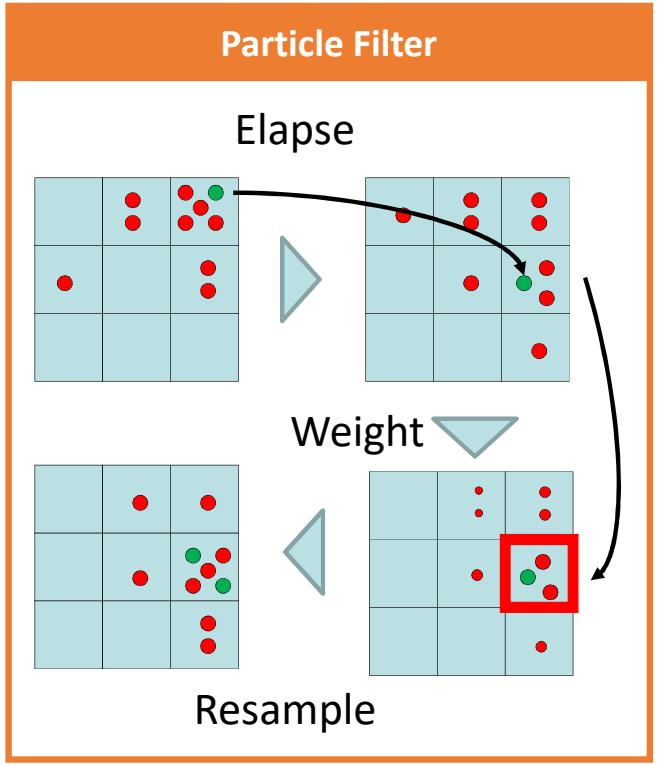


Most accurate but computationally very expensive (often intractable)

4 There are four popular ways to compute this in practice

Berkely AI Material and Scott Kuindersma

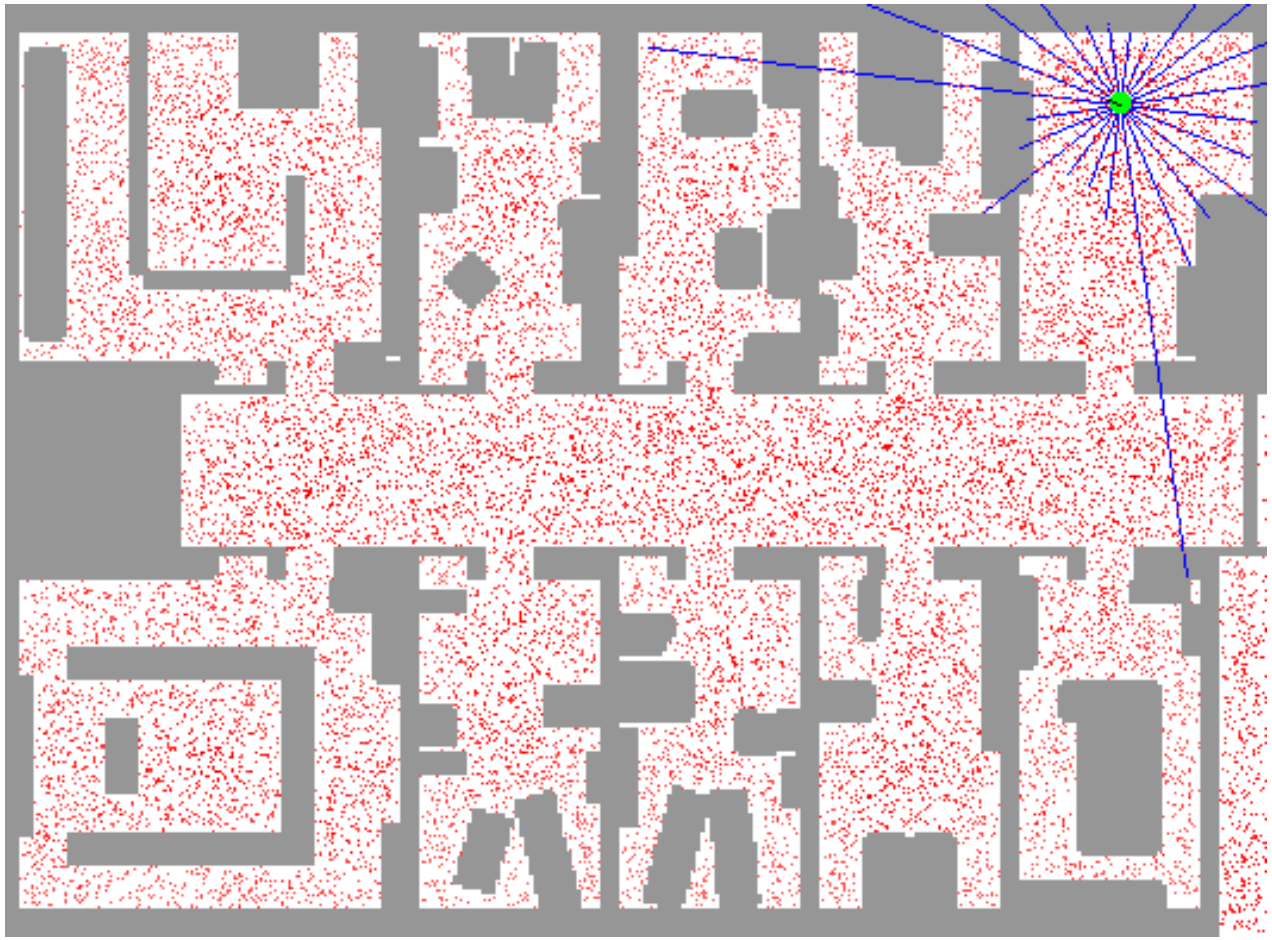
2. **Pass many samples** through the **nonlinear** equations for the motion update (physics) and the sensor update and use the samples as a discrete approximation of the probability distribution



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

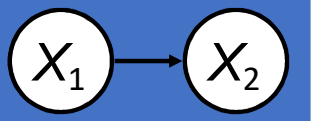
4 There are four popular ways to compute this in practice

Dieter Fox



4 There are four popular ways to compute this in practice

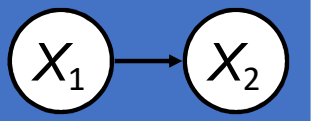
What if we don't want to sample?



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

4 There are four popular ways to compute this in practice

Lets do some math for a minute

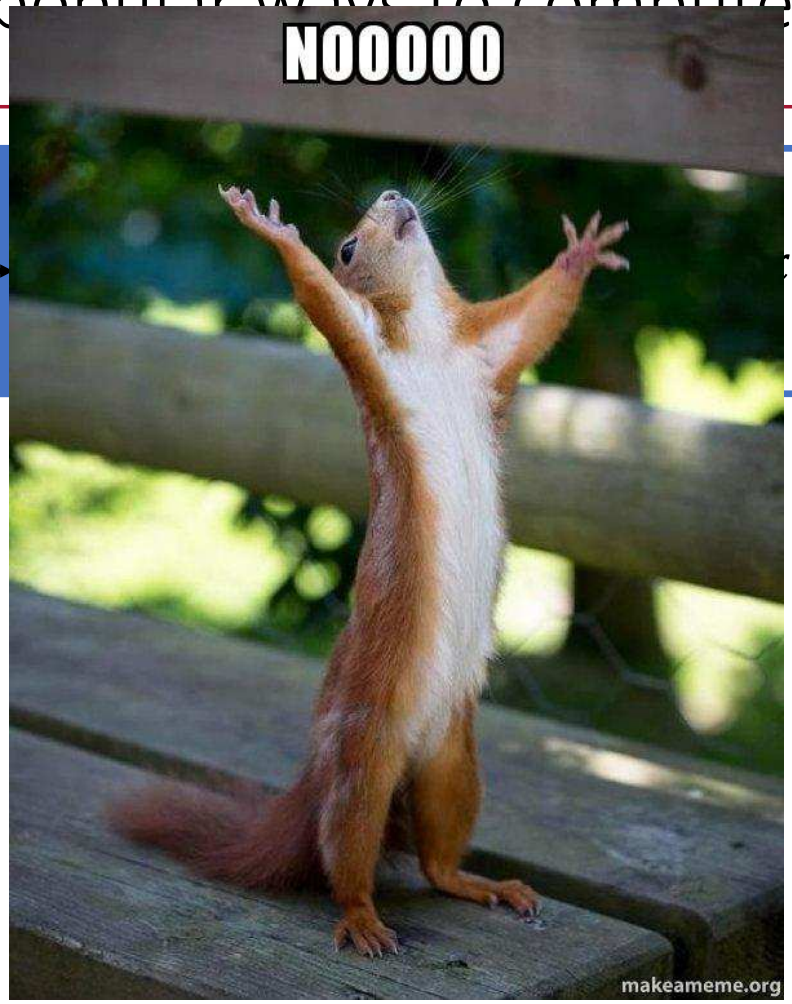


$$P(x_{t+1}|x_0, e_1 \dots e_t) = \int P(x_t|x_0, e_1 \dots e_t) * P(x_{t+1}|x_t)$$

4 There are four popular ways to compute this in practice

Lets do some math for a minute

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

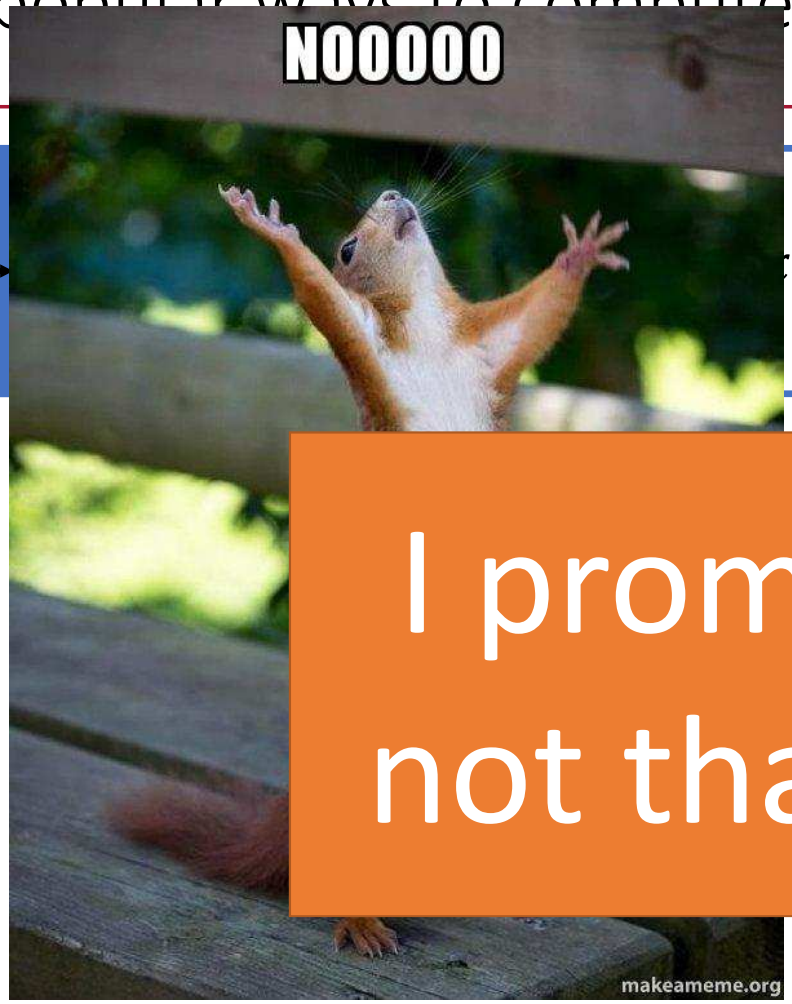


4 There are four popular ways to compute this in practice

Lets do some math for a minute

X_1

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



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

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

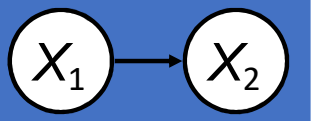
ise its

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



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

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

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

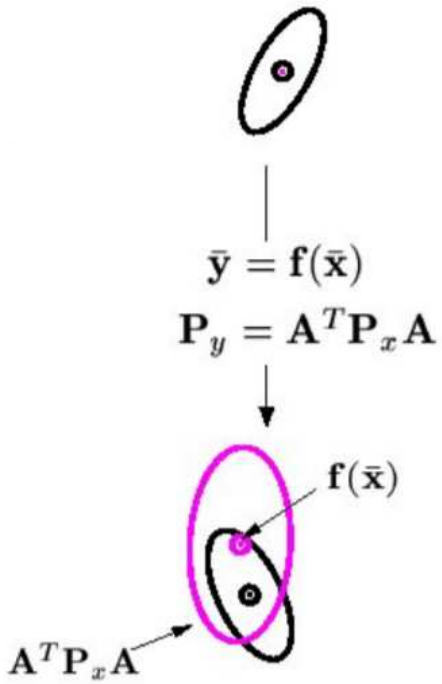
Well if we represent the transition from one state to the next by a linear equation (just linearize physics) and represent P(x) as Gaussian then we can just use this simple linear transformation to do all of the math super fast!

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



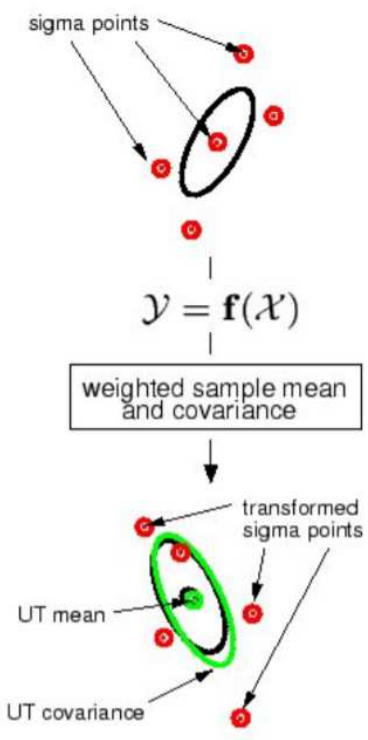
Simplest and least accurate as assumes Gaussian + linear

4 There are four popular ways to compute this in practice

van der Merwe and Wan (2001)

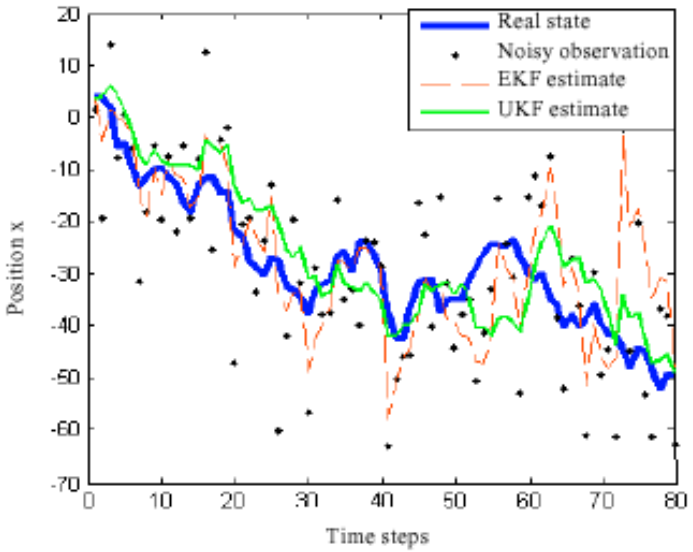
4. Assume the belief PDF is **Gaussian** and **pass limited samples** through the **nonlinear** equations for the motion update (physics) and the sensor update and reconstruct the Gaussian on the other side

Unscented Kalman Filter - UKF



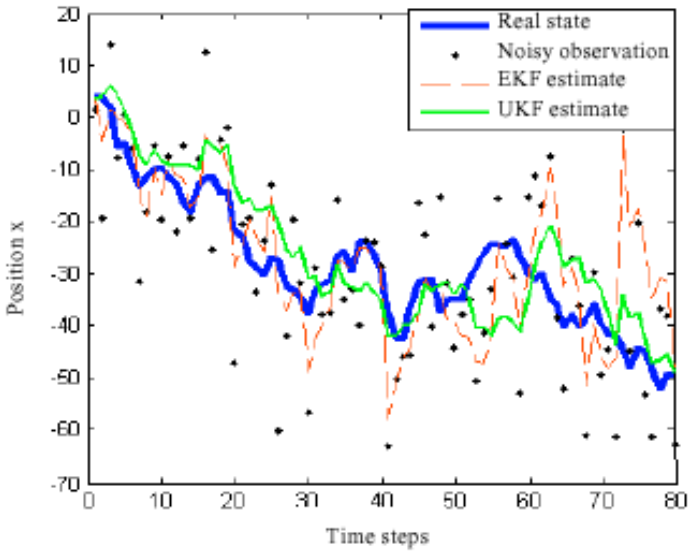
Moderately accurate but assumes Gaussian

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



Hassanzadeh and Fallah 2008

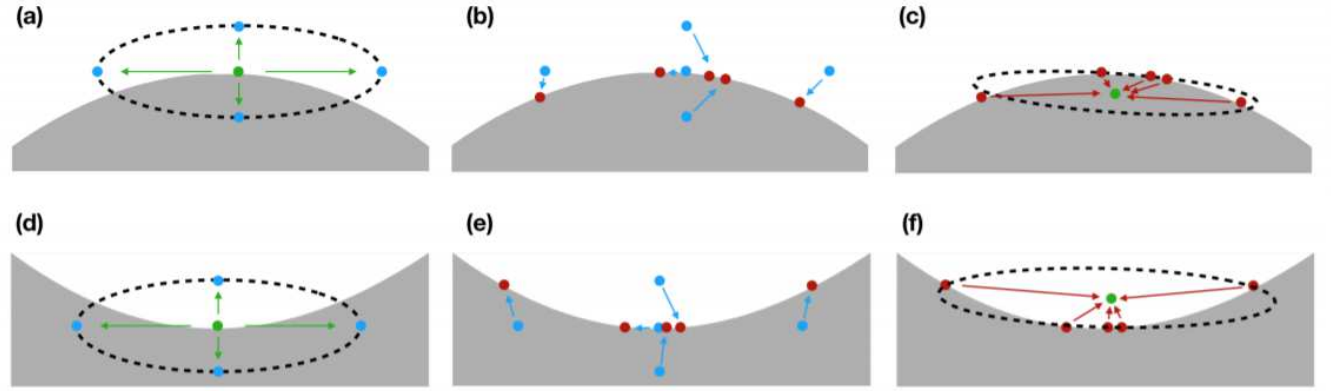
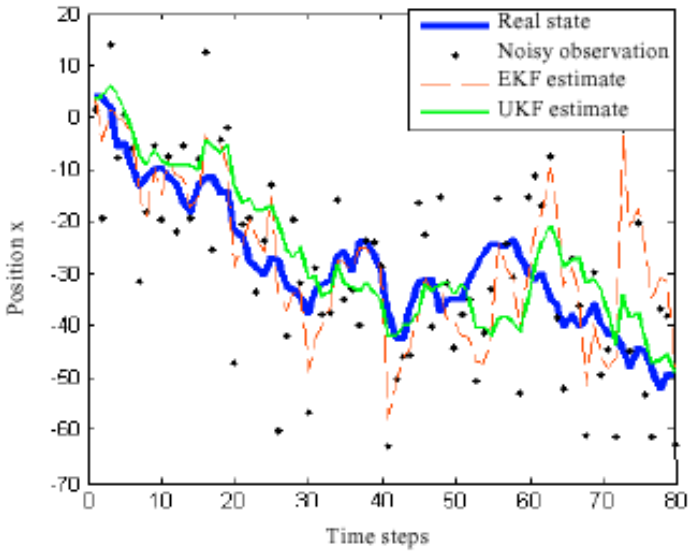


Fig. 1: Two illustrations of fundamental problems associated with the UKF in the presence of the inequalities associated with contact. When sample points are generated (a and d) samples are either infeasible or have different contact modes than the mean estimate. In the first sequence (a-c) the resulting 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.

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**!



Hassanzadeh and Fallah 2008

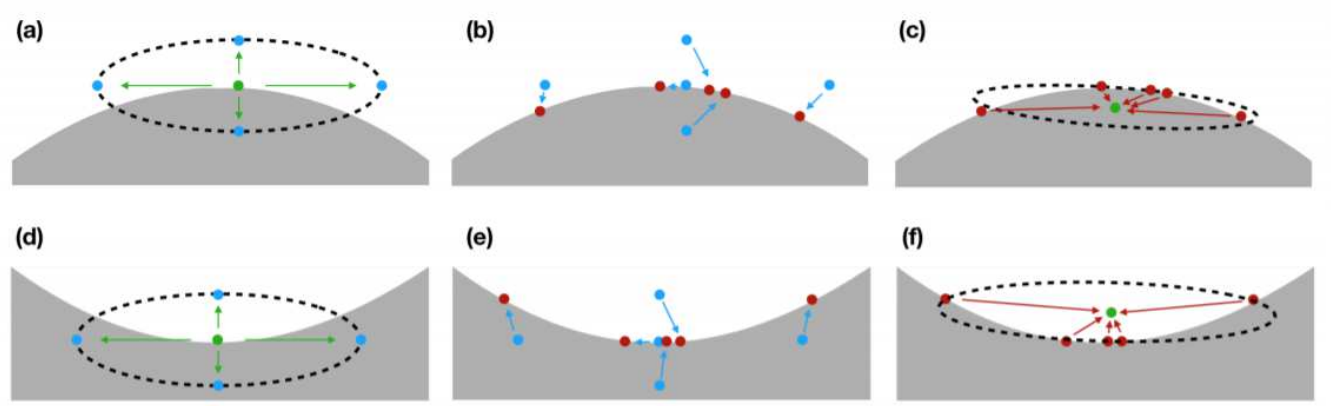


Fig. 1: Two illustrations of fundamental problems associated with the UKF in the presence of the inequalities associated with contact. When sample points are generated (a and d) samples are either infeasible or have different contact modes than the mean estimate. In the first sequence (a-c) the resulting 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.

Varin and Kuindersma 2018

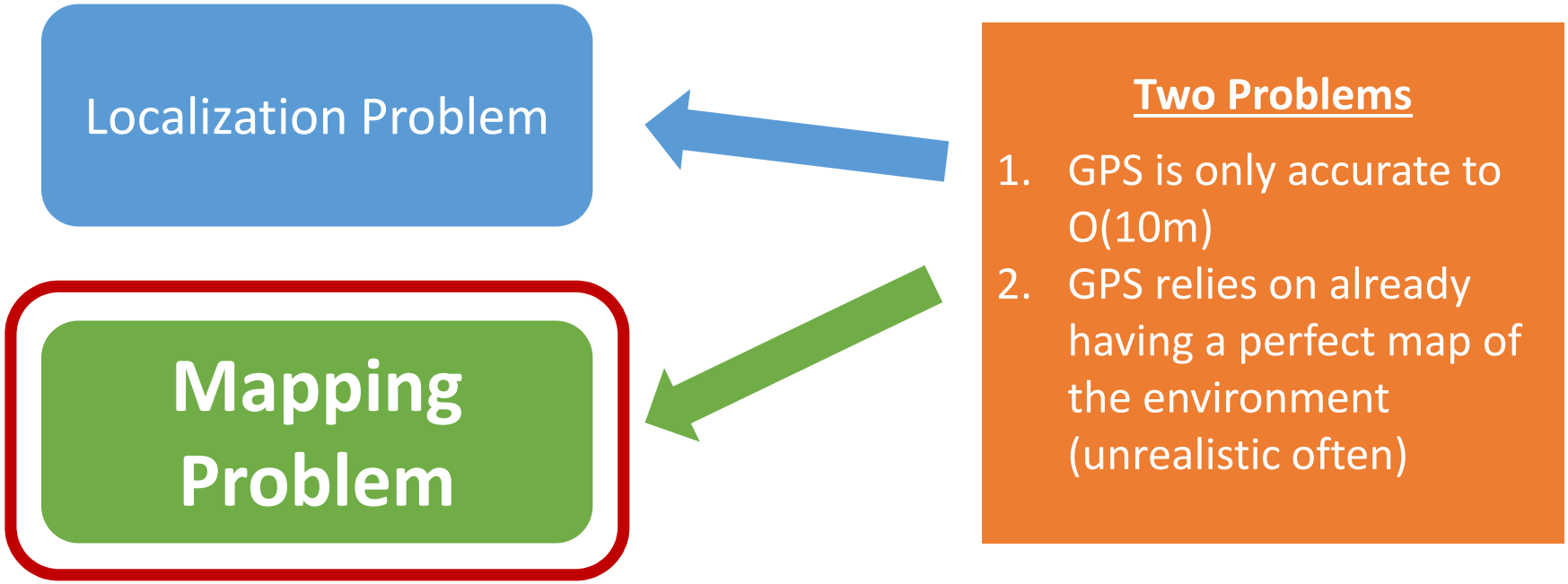
4 There are four popular ways to compute this in practice

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

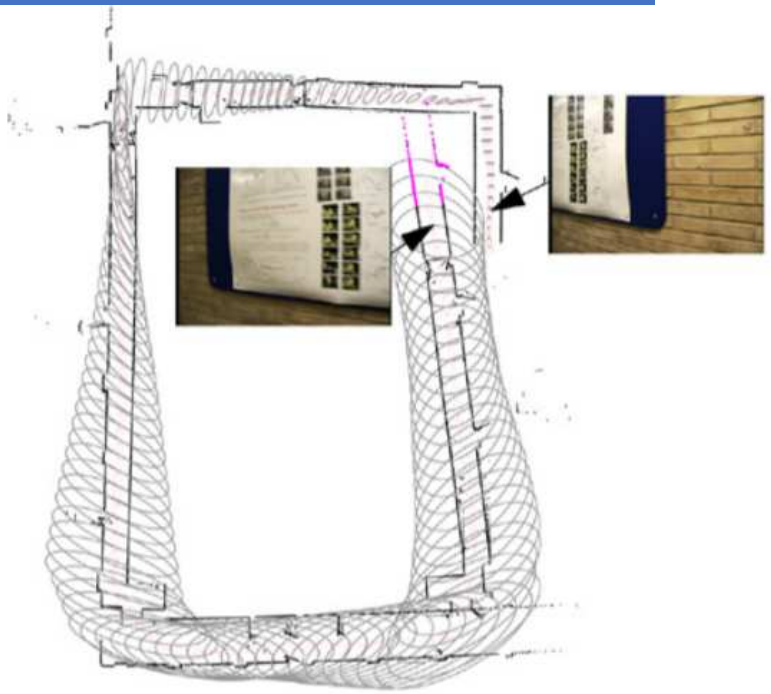
1. Pass the full belief PDF through **nonlinear** equations for the motion update (physics) and the sensor update	Most accurate but computationally very expensive (often intractable)
2. Pass **many samples** through the **nonlinear** equations for the motion update (physics) and the sensor update and 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?



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.)

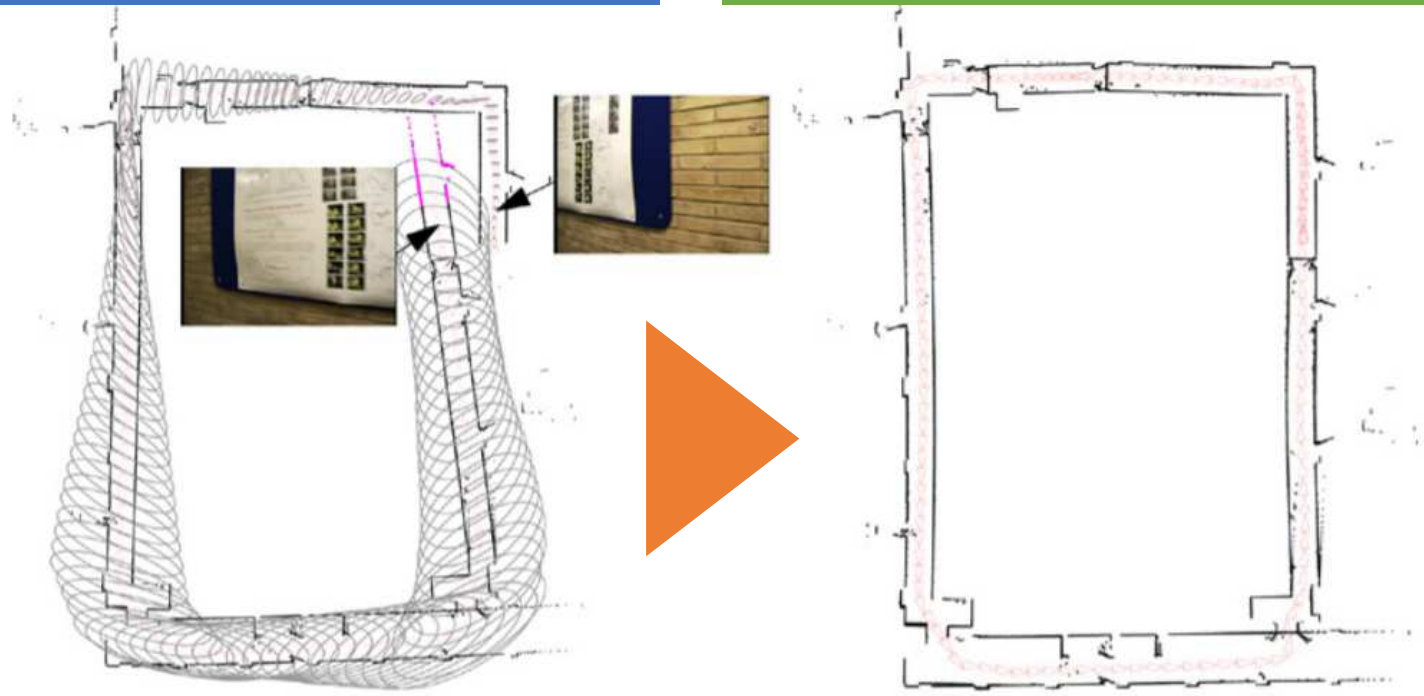


Ho and Newman 2006

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.)

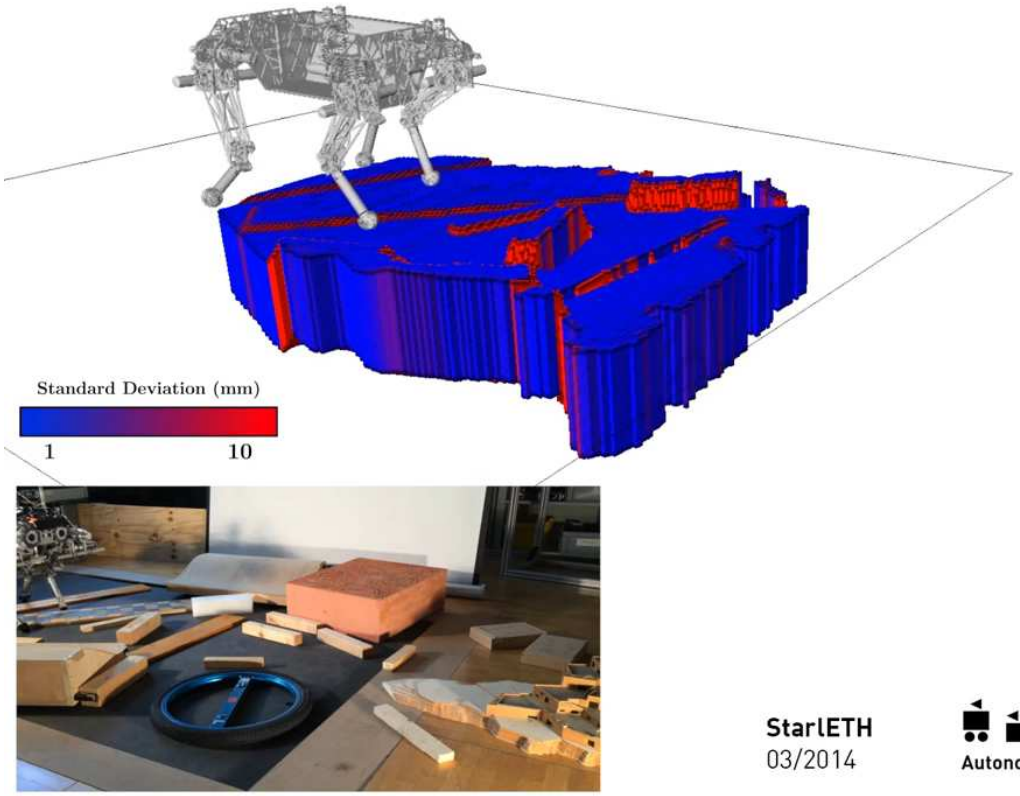
The real hard part is figuring out when you have been somewhere before as measurements drift (the **loop closure** problem)



Ho and Newman 2006

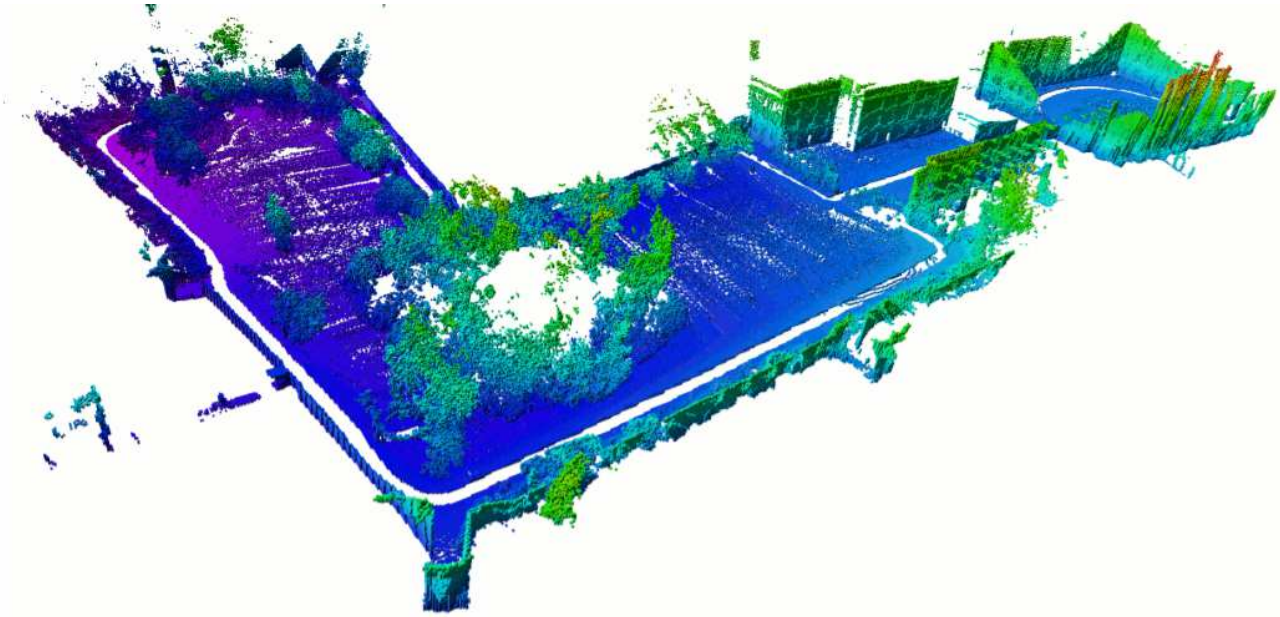
4 SLAM with Loop Closure

4 Mapping can even be done in 3D!



4

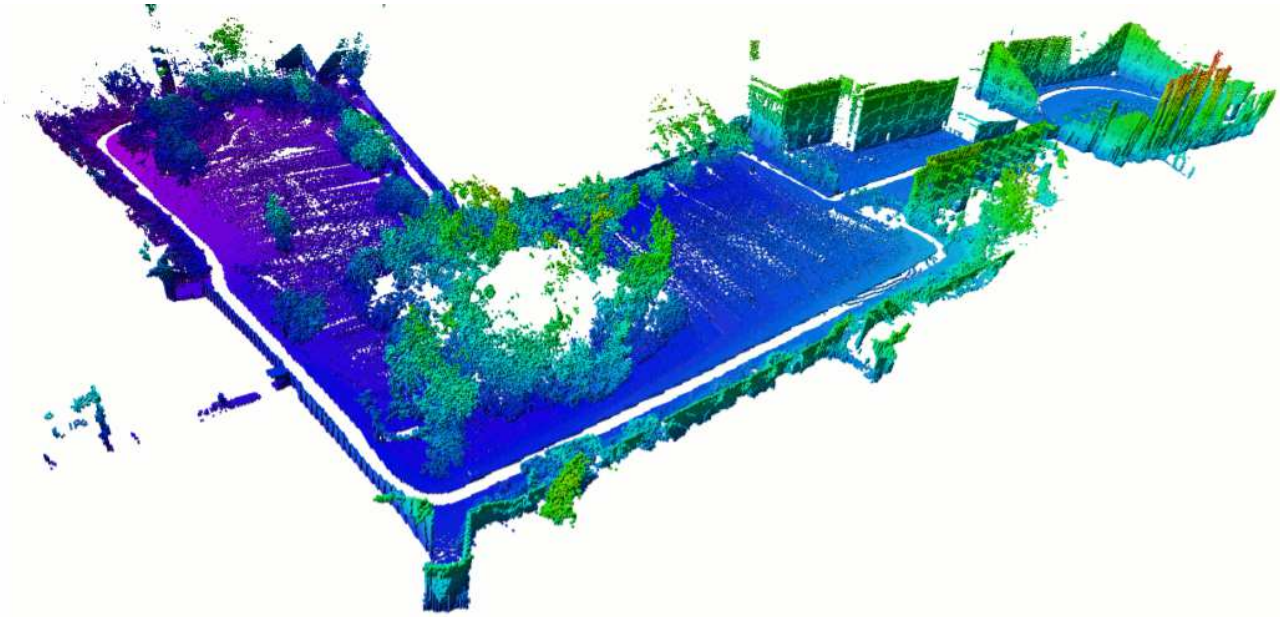
However building (and even storing) maps leads to a huge memory problem especially on small mobile systems



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

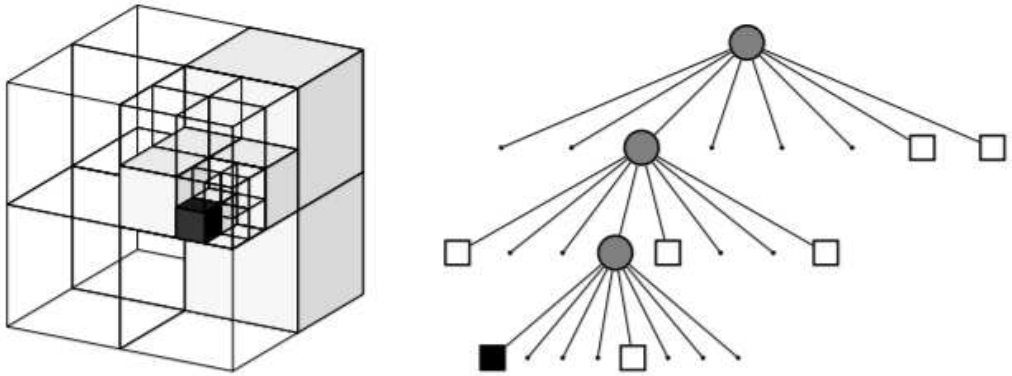


Fig. 2 Example of an octree storing free (shaded white) and occupied (black) cells. The volumetric model is shown on the left and the corresponding tree representation on the right.

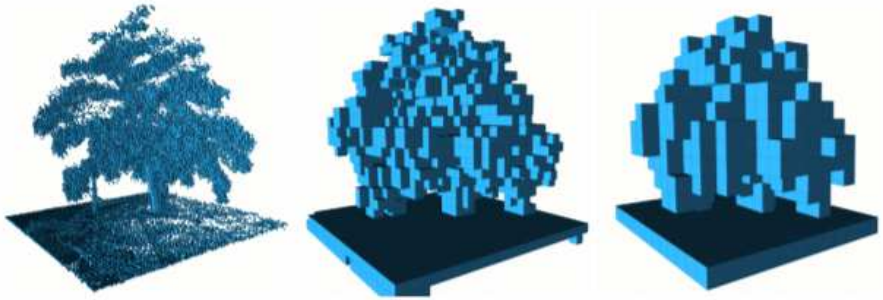


Fig. 3 By limiting the depth of a query, multiple resolutions of the same map can be obtained at any time. Occupied voxels are displayed in resolutions 0.08 m, 0.64 , and 1.28 m.

"Octomap" Hornung et. al. 2012

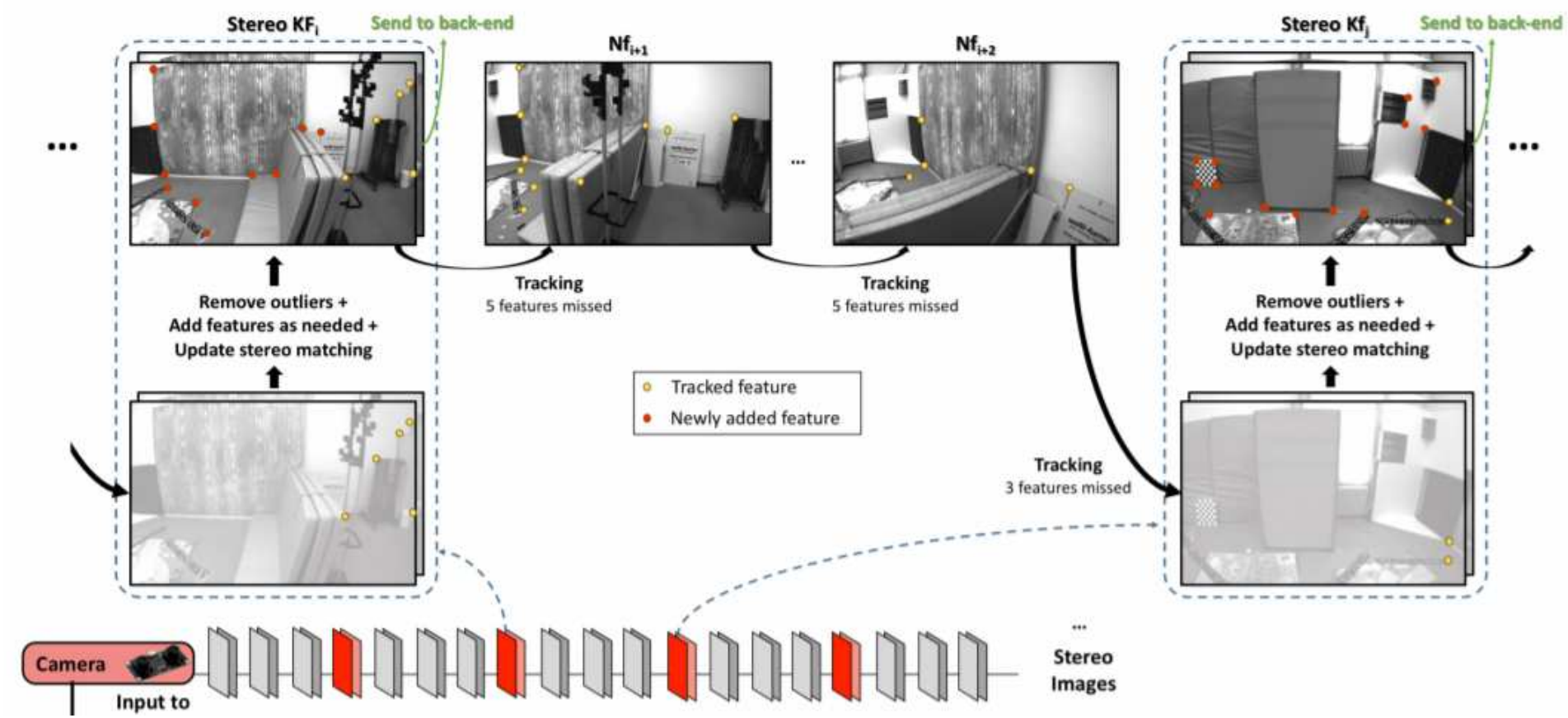
4

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

Any ideas?

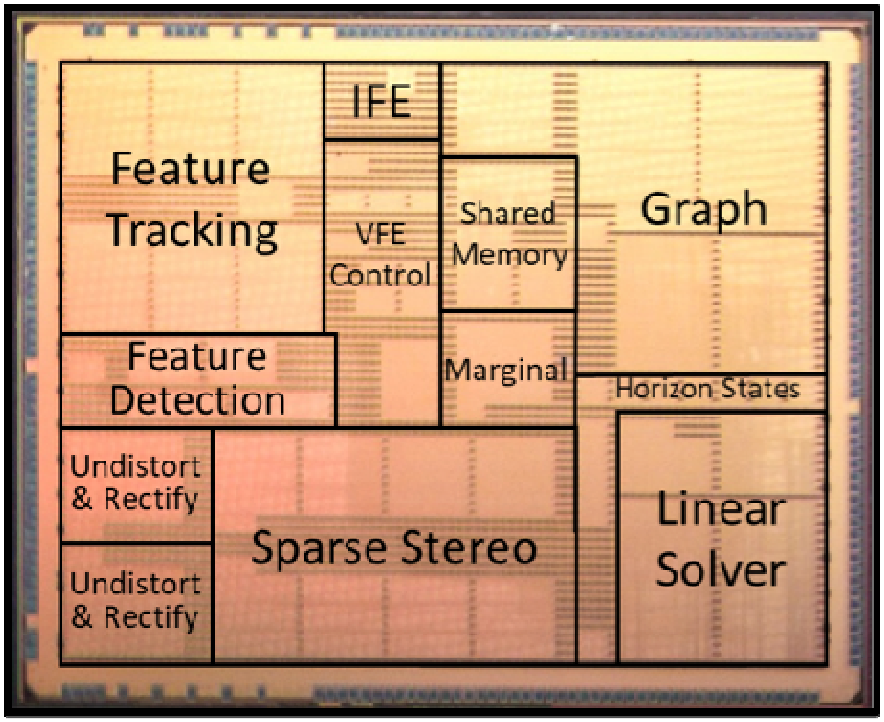
4

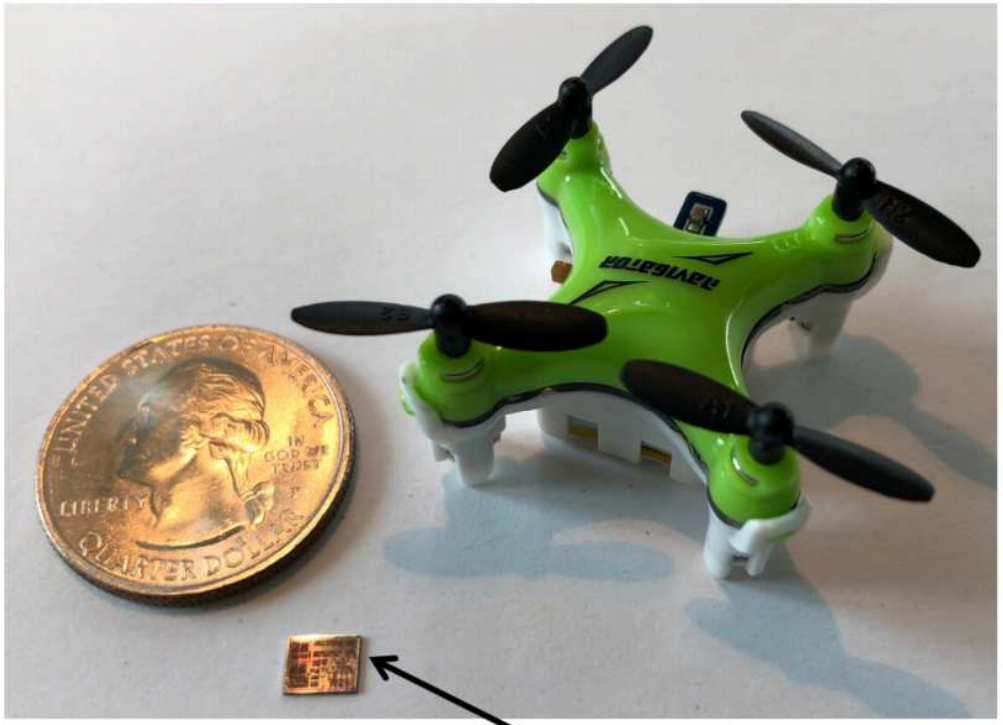
You can estimate the velocity of an object through matching interest points (Visual Odometry)…



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

http://navion.mit.edu





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

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