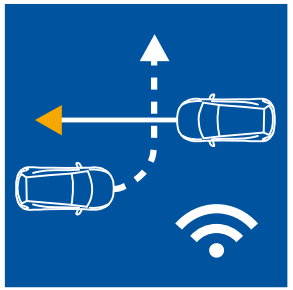


Lecture

Control and Perception in Networked and Autonomous Vehicles

Alexandru Kampmann, M. Sc. | Dr. Bassam Alrifaee | Patrick Scheffe, M. Sc. | Simon Schäfer, M. Sc.

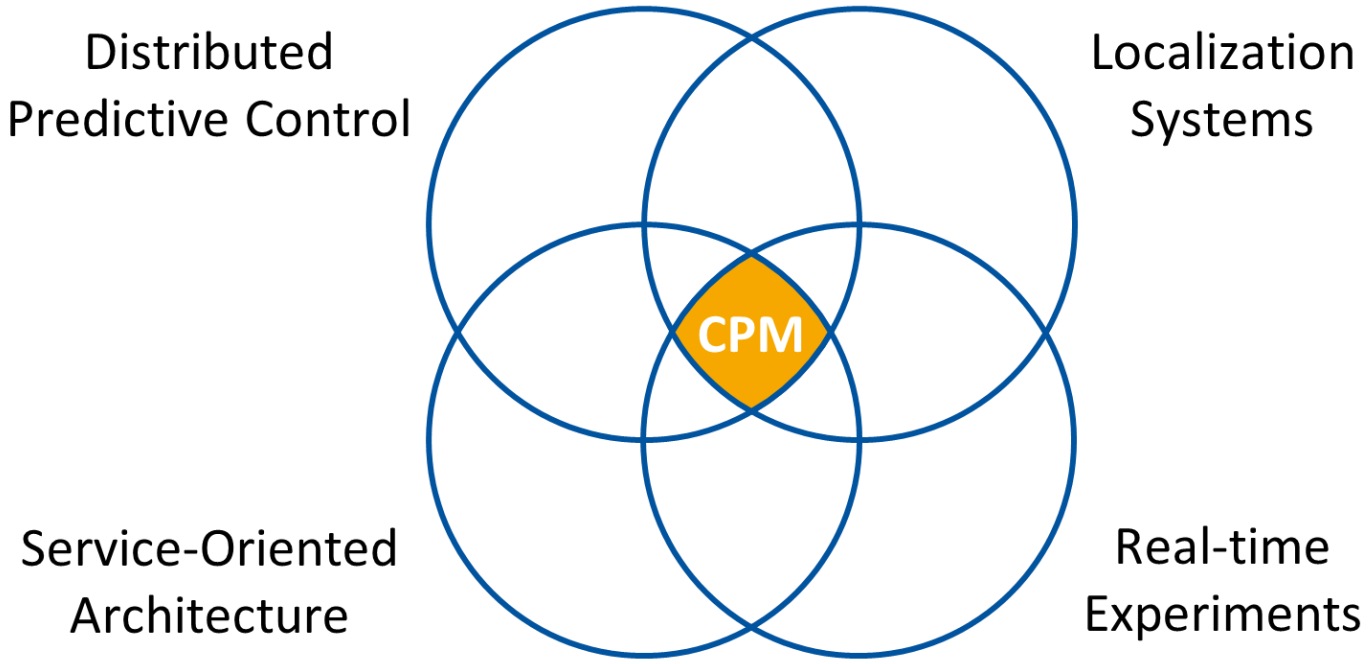
Winter Semester 2023/2024

Part 5

Machine Perception

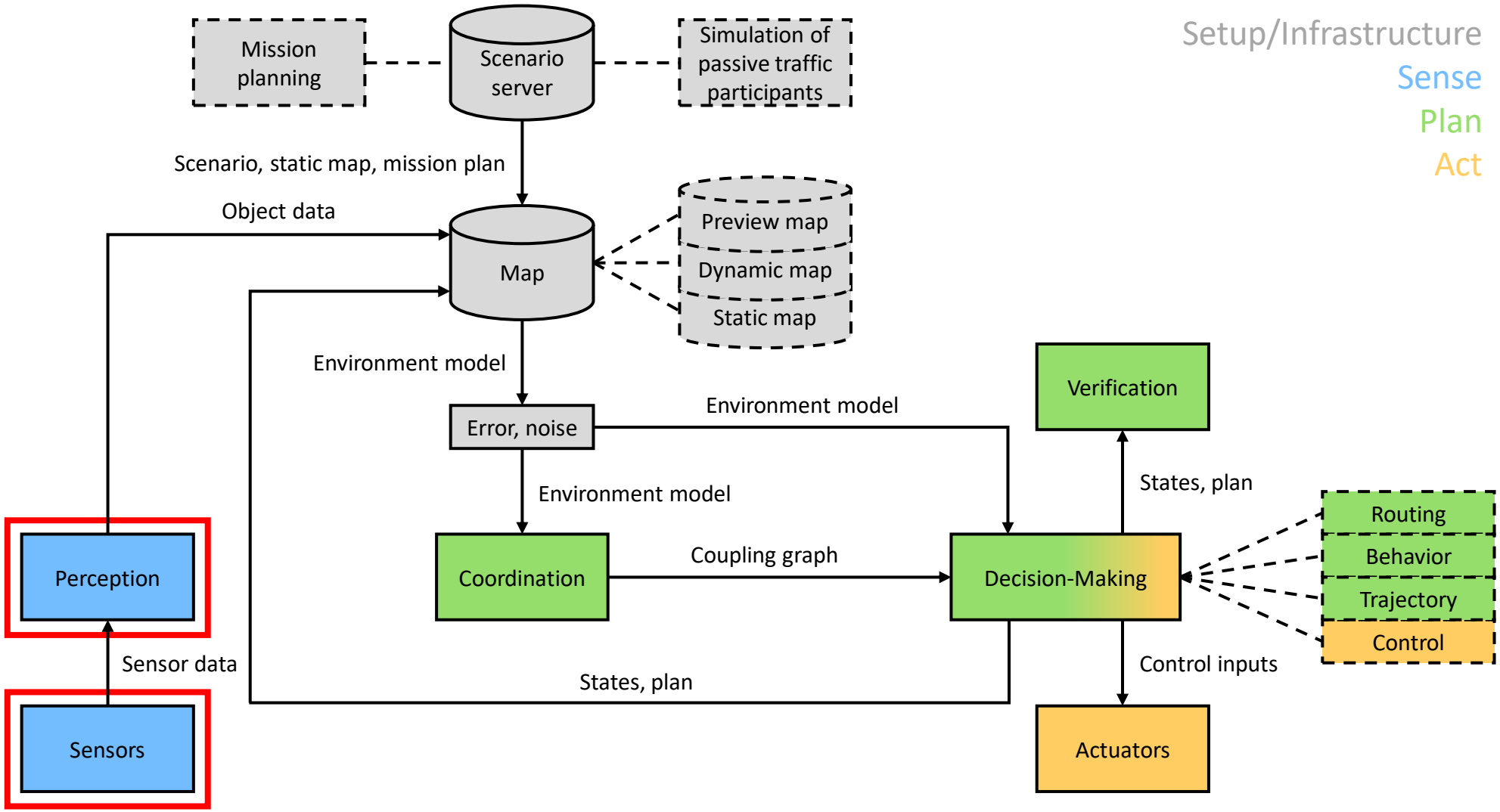
Course contents (CPM group course)

- Vehicle models
- Control and optimization
- Network and distribution
- Machine perception
- Software architectures and testing concepts
- Case study



*CPM: Cyber-Physical Mobility

Lab architecture

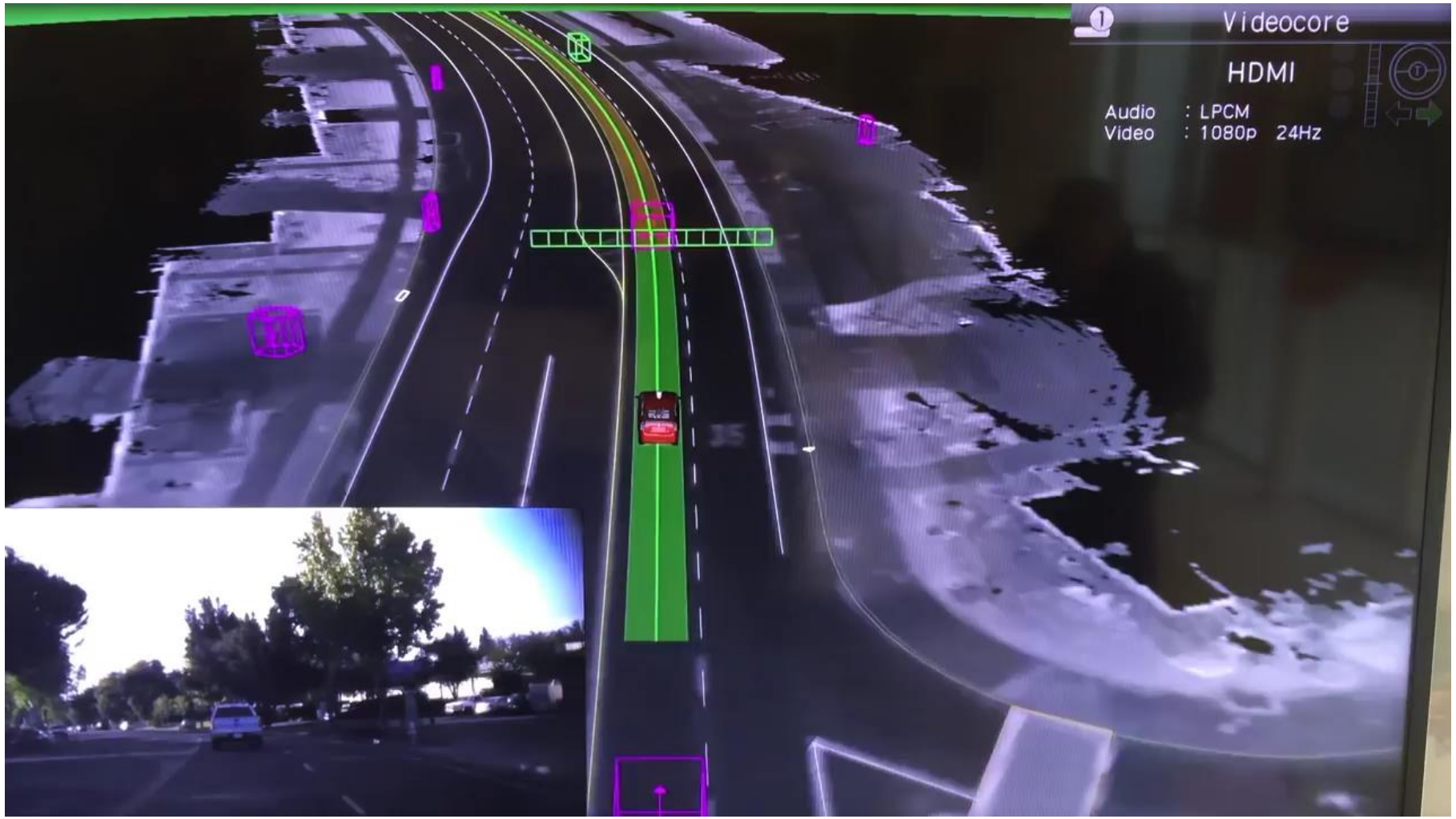


Agenda

- Sensors & Environment Models
- Computer Vision Basics: Classic Problems & Approaches
- Machine Learning Basics
- Deep Learning Basics

Perception Basics: Sensors & Environment Models

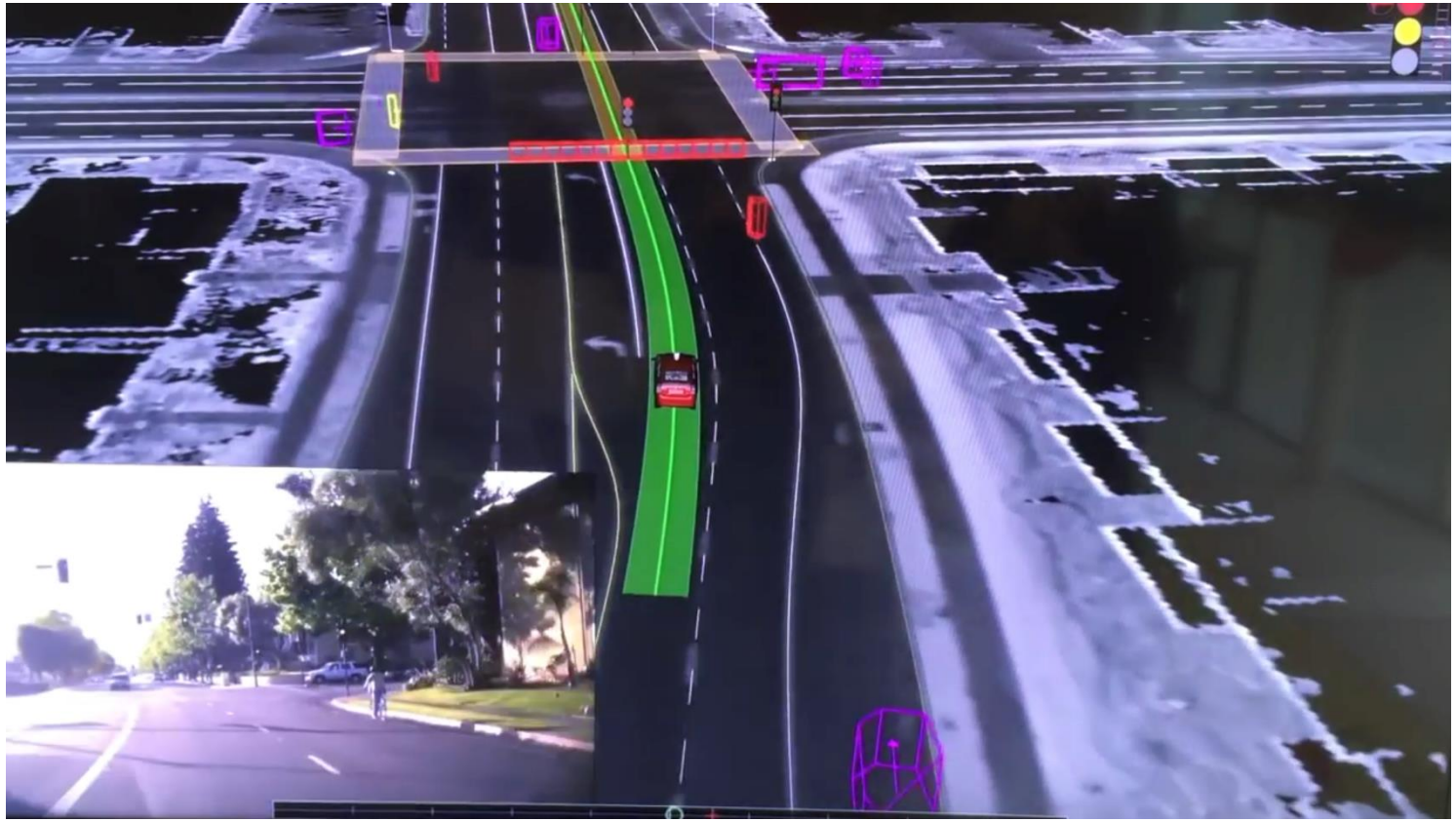
Perception Basics



How Google self-driving car sees a road - https://www.youtube.com/watch?v=MqUbdd7ae54

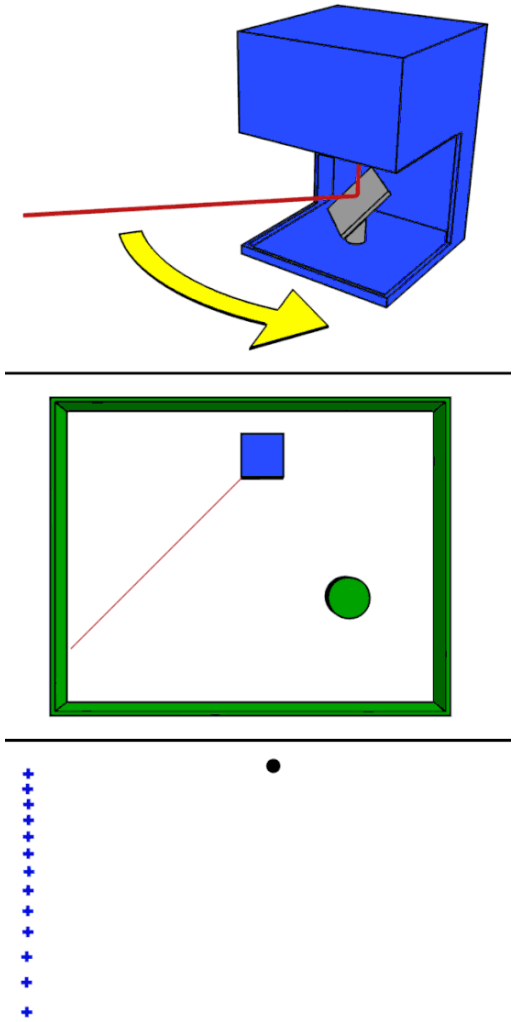
Perception Basics

- Environment model with **dynamic** and **static** elements
- Input form sensors and offline maps (localization)
- Basis for decision making, trajectory planning, control etc.



Sensors – Light Detection and Ranging (LIDAR)

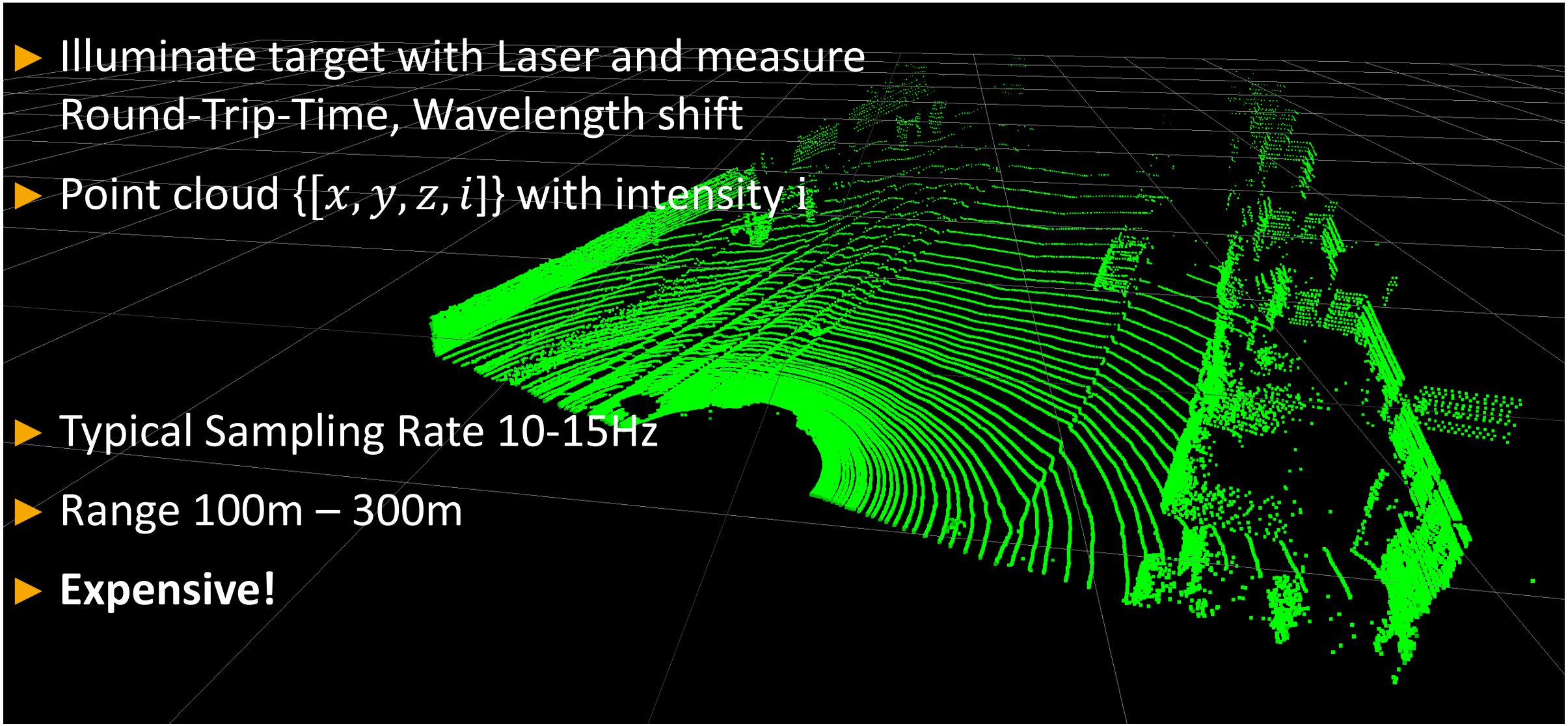
- Illuminate target with laser and measure round-trip-time, wavelength shift
- Point cloud $\{[x, y, z, i]\}$ with intensity i



- Typical sampling rate 10-15Hz
- Range 100m – 300m
- **Expensive!**

Sensors – Light Detection and Ranging (LIDAR)

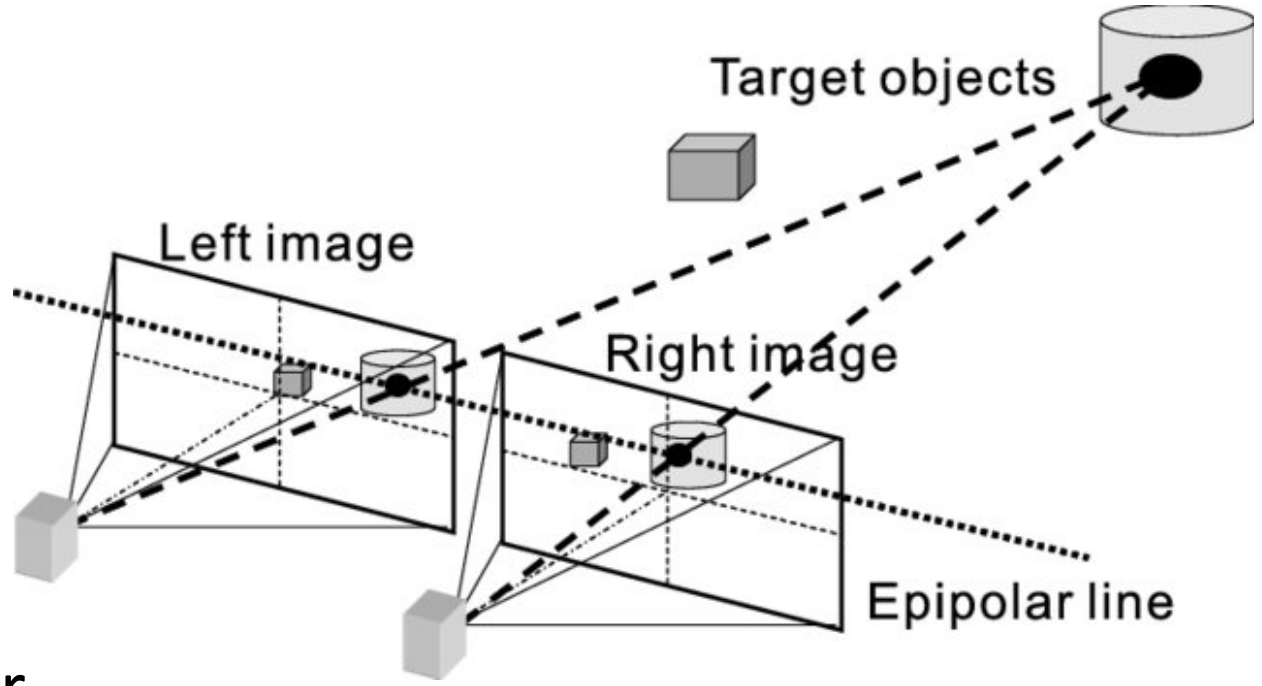
- Illuminate target with Laser and measure Round-Trip-Time, Wavelength shift
- Point cloud $\{[x, y, z, i]\}$ with intensity i



- Typical Sampling Rate 10-15Hz
- Range 100m – 300m
- **Expensive!**

Sensors – Stereo Vision

- Depth *estimation* from two mono cameras
- Correspondences in Left/Right Images yields depth estimation
- Issues with matching can lead to wrong depth estimates
- In contrast to LIDAR no **active** distance measurement, but cheaper and higher frequency



Sensors – Radar

- Emmits pulsed electromagnetic wave
- Objects reflect wave back to radar antenna
- Provides accurate measurement of
  - relative position of object
  - range to object
  - velocity

LRR3	MRR Front	MRR Rear
		
SOP: 2009	SOP: 2013	SOP: 2014
• Range: up to 250 m • SiGe MMICs (bare chip) • Opening Angle: 30° • Dimensions (HxWxD) 77 x 74 x 58 mm • Weight: 285 g	• Range: up to 160 m • SiGe MMICs (packaged chip) • Opening Angle: 45° • Dimensions (HxWxD) 60 x 70 x 30 mm • Weight: 200 g	• Range: up to 100 m • SiGe MMICs (packaged chip) • Opening Angle: 150° • Dimensions (HxWxD) 60 x 70 x 30 mm • Weight: 190 g

Bosch

No Sensor is perfect - Combination necessary

LIDAR	Stereo Camera	RADAR
sparse point cloud, becomes worse for larger distances	rich semantic information	no semantic information
precise range measurement	estimated range information	precise range measurement, mostly for metal objects
robust at day and night, but deteriorates in rain or snow	deteriorating performance at night	insensitive to weather conditions
very expensive	comparatively cheap	comparatively cheap
low update rate, 10 Hz	high update rate	high update rate

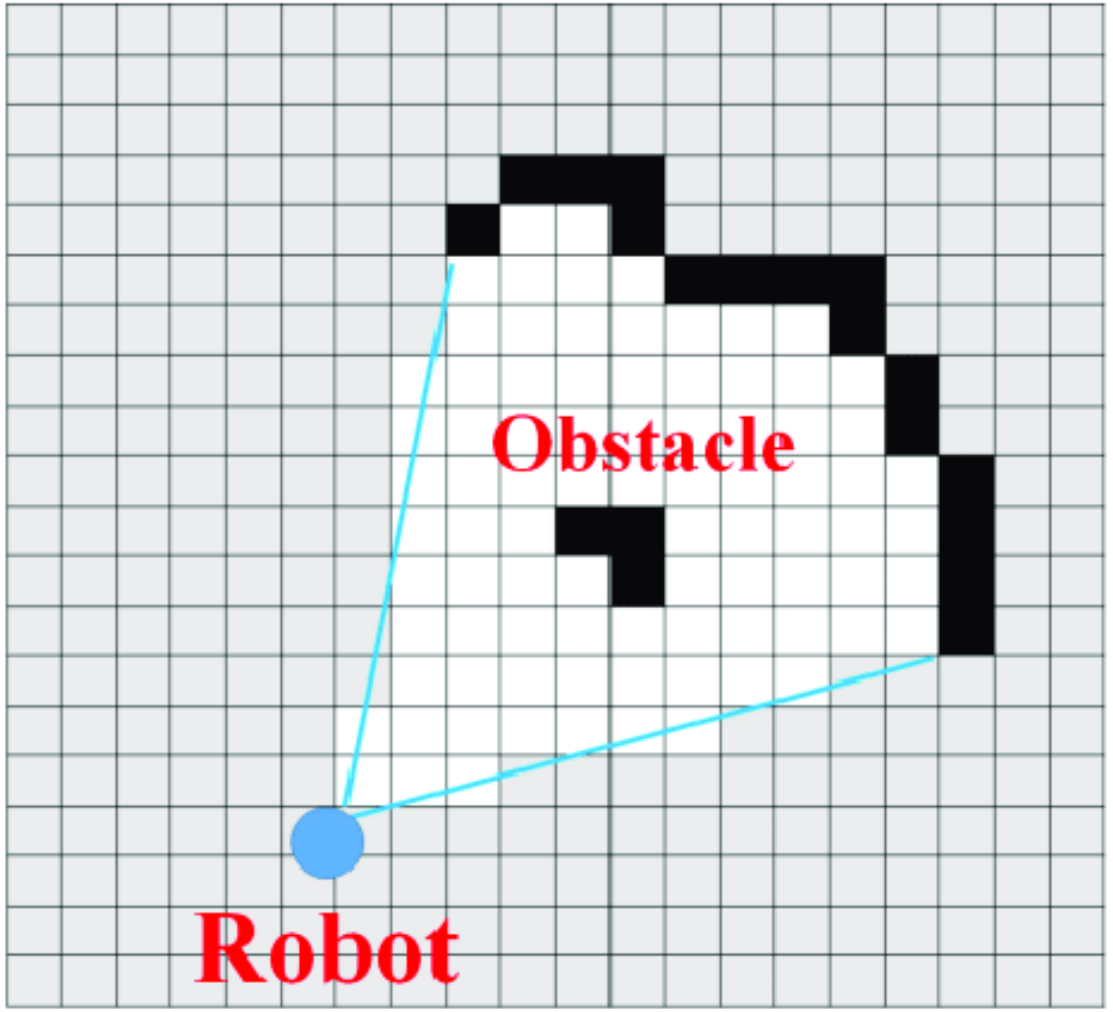
DARPA Urban Challenge Winner – Boss (2006)

Environment Models

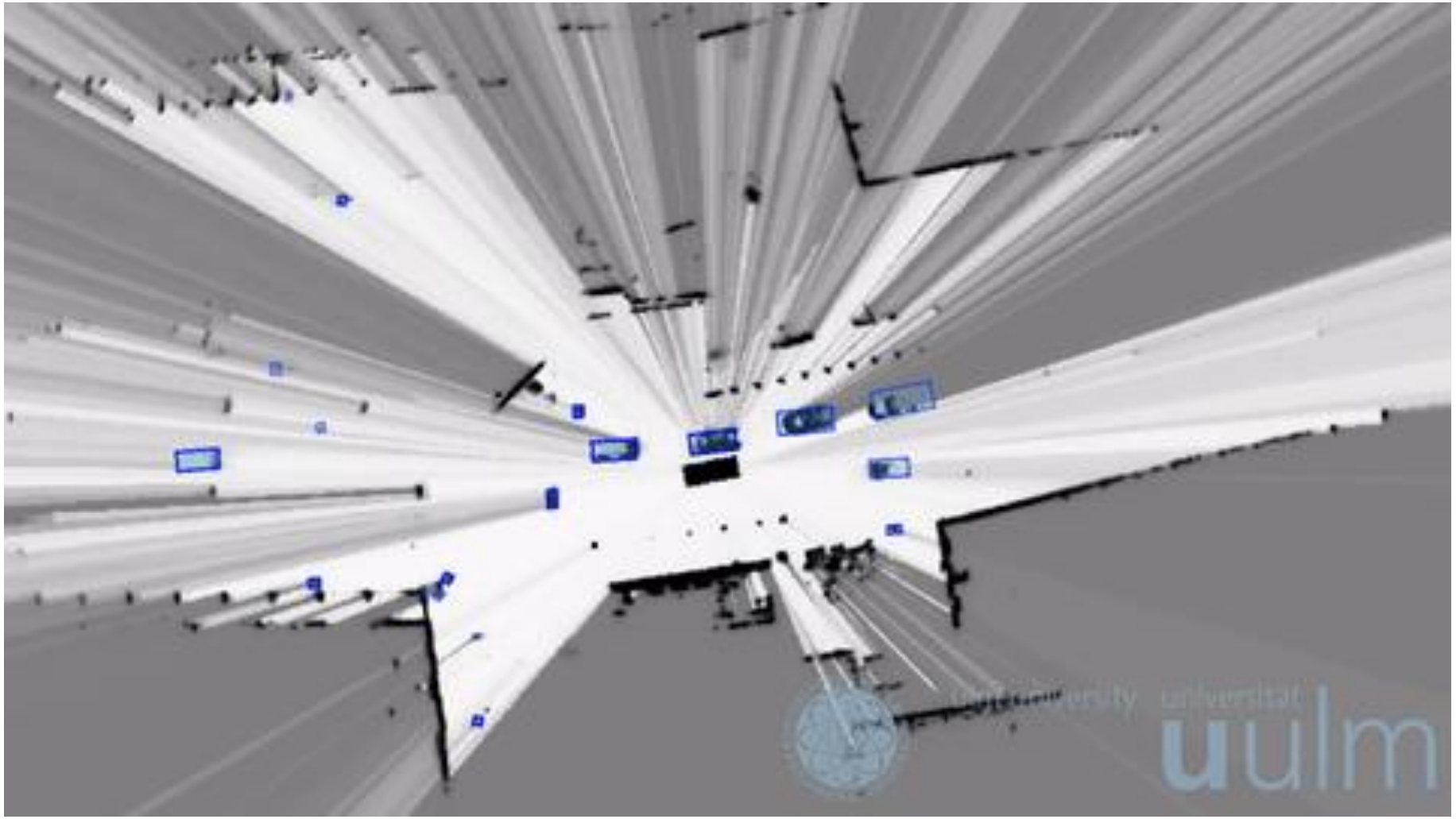
- Model describes environment in a suitable way for further algorithmic processing (i.e. decision making, path planning, control, …)
- We will learn about two common environment representations
  - **Occupancy Grid Mapping**
  - **Object Lists**

Occupancy Grid Mapping

- Discretize world in cells of, e.g., 5x5cm
- Each cell is either **occupied** or **free**
- Use range finder (LIDAR, Stereo Camera, Ultrasound, …) to determine state of cell while robot moves through the environment



Occupancy Grid Mapping



S. Hoermann, P. Henzler, M. Bach and K. Dietmayer, "Object Detection on Dynamic Occupancy Grid Maps Using Deep Learning and Automatic Label Generation," *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, 2018, pp. 826-833, doi: 10.1109/IVS.2018.8500677.

Occupancy Grid Mapping

- Assumptions here: cell is either completely free or occupied, world is static, individual cells are independent of each other
- **Sensor noise**: cell occupancy modelled as binary random variable $m_t^{[x,y]}$
- Occupancy measurements are obtained at time t as $z_t^{[x,y]} \sim \{0, 1\}$
  - 0 – free, 1 – occupied

$$m_t^{[x,y]}: \{free, occupied\} \rightarrow \{0,1\}$$

$$p\left(m_t^{[x,y]} = 1 \mid z_{1:t}^{[x,y]}\right)$$

$m_{1,1}$	$m_{1,2}$	$m_{1,3}$	…	
		$m_{2,3}$	⋮	
		Robot		$m_{x,y}$

Occupancy Grid Mapping

- Probabilistic measurement model $\mathrm{p}\left(z_t^{[x,y]} \middle| m_t^{[x,y]}\right)$
  - $\mathrm{p}\left(z_t^{[x,y]} = 1 \middle| m_t^{[x,y]} = 1\right)$ – True occupied
  - $\mathrm{p}\left(z_t^{[x,y]} = 0 \middle| m_t^{[x,y]} = 1\right)$ – False free
  - $\mathrm{p}\left(z_t^{[x,y]} = 1 \middle| m_t^{[x,y]} = 0\right)$ – False occupied
  - $\mathrm{p}\left(z_t^{[x,y]} = 0 \middle| m_t^{[x,y]} = 0\right)$ – True free

Occupancy Grid Mapping

- Probabilistic measurement model $\mathrm{p}\left(z_t^{[x,y]} \middle| m_t^{[x,y]}\right)$
  - $\mathrm{p}\left(z_t^{[x,y]} = 1 \middle| m_t^{[x,y]} = 1\right) = 0.9$
  - $\mathrm{p}\left(z_t^{[x,y]} = 0 \middle| m_t^{[x,y]} = 1\right) = 1 - \mathrm{p}\left(z_t^{[x,y]} = 1 \middle| m_t^{[x,y]} = 1\right) = 0.1$
  - $\mathrm{p}\left(z_t^{[x,y]} = 1 \middle| m_t^{[x,y]} = 0\right) = 0.2$
  - $\mathrm{p}\left(z_t^{[x,y]} = 0 \middle| m_t^{[x,y]} = 0\right) = 1 - \mathrm{p}\left(z_t^{[x,y]} = 1 \middle| m_t^{[x,y]} = 0\right) = 0.8$

Occupancy Grid Mapping – Basic Framework

- Recursive State Estimation of $p\left(m_t^{[x,y]} \mid z_{1:t}^{[x,y]}\right)$

$$p\left(m_t^{[x,y]} \mid z_{1:t}^{[x,y]}\right) = \frac{\mathrm{p}\left(z_t^{[x,y]} \mid m_t^{[x,y]}, z_{1:t-1}^{[x,y]}\right) \mathrm{p}\left(m_t^{[x,y]} \mid z_{1:t-1}^{[x,y]}\right)}{\mathrm{p}\left(z_t^{[x,y]} \mid z_{1:t-1}^{[x,y]}\right)}$$

$$= \mathrm{p}\left(z_t^{[x,y]} \mid m_t^{[x,y]}, z_{1:t-1}^{[x,y]}\right) \mathrm{p}\left(m_t^{[x,y]} \mid z_{1:t-1}^{[x,y]}\right) \eta$$

already incorporated in $m_t^{[x,y]}$

$$= \mathrm{p}\left(z_t^{[x,y]} \mid m_t^{[x,y]}\right) \mathrm{p}\left(m_t^{[x,y]} \mid z_{1:t-1}^{[x,y]}\right) \eta$$

$$= \mathrm{p}\left(z_t^{[x,y]} \mid m_t^{[x,y]}\right) \mathrm{p}\left(m_{t-1}^{[x,y]} \mid z_{1:t-1}^{[x,y]}\right) \eta$$

Prior from previous timestep

Occupancy Grid Mapping – Example

- Measurement model
  - $\mathrm{p}\left(z_t^{[x,y]} = 1 \,\middle|\, m_t^{[x,y]} = 1\right) = 0.9, \quad \mathrm{p}\left(z_t^{[x,y]} = 0 \,\middle|\, m_t^{[x,y]} = 1\right) = 0.1$
  - $\mathrm{p}\left(z_t^{[x,y]} = 1 \,\middle|\, m_t^{[x,y]} = 0\right) = 0.2, \quad \mathrm{p}\left(z_t^{[x,y]} = 0 \,\middle|\, m_t^{[x,y]} = 0\right) = 0.8$
- **Reminder** $p\left(m_t^{[x,y]} \mid z_{1:t}^{[x,y]}\right) = \mathrm{p}\left(z_t^{[x,y]} \mid m_t^{[x,y]}\right) \mathrm{p}\left(m_{t-1}^{[x,y]} \mid z_{1:t-1}^{[x,y]}\right)$

0.5	0.5	0.5	…	
		0.5	⋮	
		Robot		$m_{x,y}$

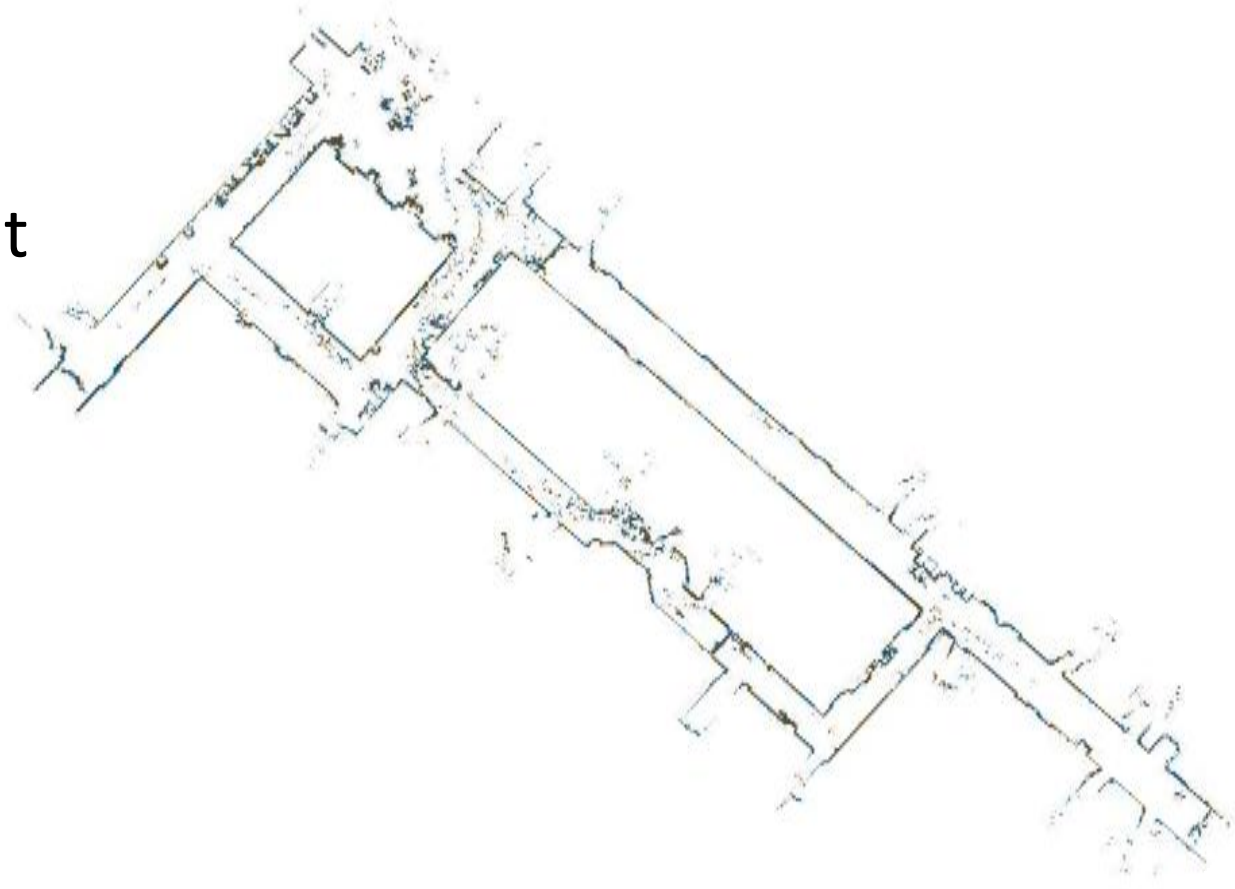
$$p\left(m_1^{1,3} = 0 \mid z_1^{1,3} = 1\right) = \eta * 0.2 * 0.5 = 0.19$$
$$p\left(m_1^{1,3} = 1 \mid z_1^{1,3} = 1\right) = \eta * 0{,}9 * 0.5 = 0.81$$
$$\eta = \frac{1}{0.5 * 0.2 + 0.9 * 0.5} = \frac{1}{0.55}$$

$$p\left(m_1^{2,3} = 1 \mid z_1^{2,3} = 0\right) = \mu * 0.1 * 0.5 = 0.12$$
$$p\left(m_1^{2,3} = 0 \mid z_1^{2,3} = 0\right) = \mu * 0.8 * 0.5 = 0.88$$
$$\eta = \frac{1}{0.1 * 0.5 + 0.8 * 0.5} = \frac{1}{0.45}$$

0.5	0.5	**0.81**	…	
		0.12	⋮	
		Robot		$m_{x,y}$

Summary

- Allows to build model of environment using known robot poses
- In practice more complex inverse measurement models
- Dynamic Occupancy Grid Maps



Object List

Object $o_i = \left(T = [t_x, t_y, t_z],\ D = [d_x, d_y, d_z],\ R = [\theta, \phi, \beta], c, \boldsymbol{x}\right)$

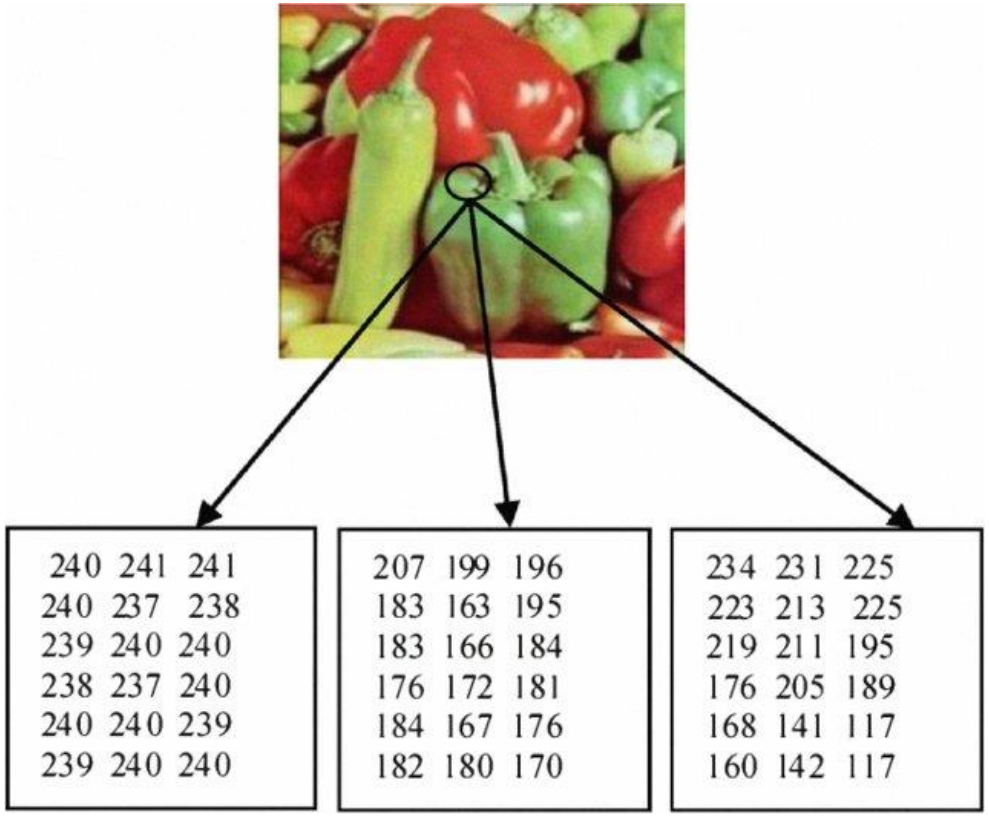
- T center
- D dimensions
- R orientation
- c object class

- $\boldsymbol{x}$ object state (e.g. velocity, acceleration, angular acceleration,…)
- Tracking for maintaining object state over time
- Extracted through application of **Machine Learning Algorithms**

Computer Vision Basics: Classic Problems & Approaches

What is a camera image?

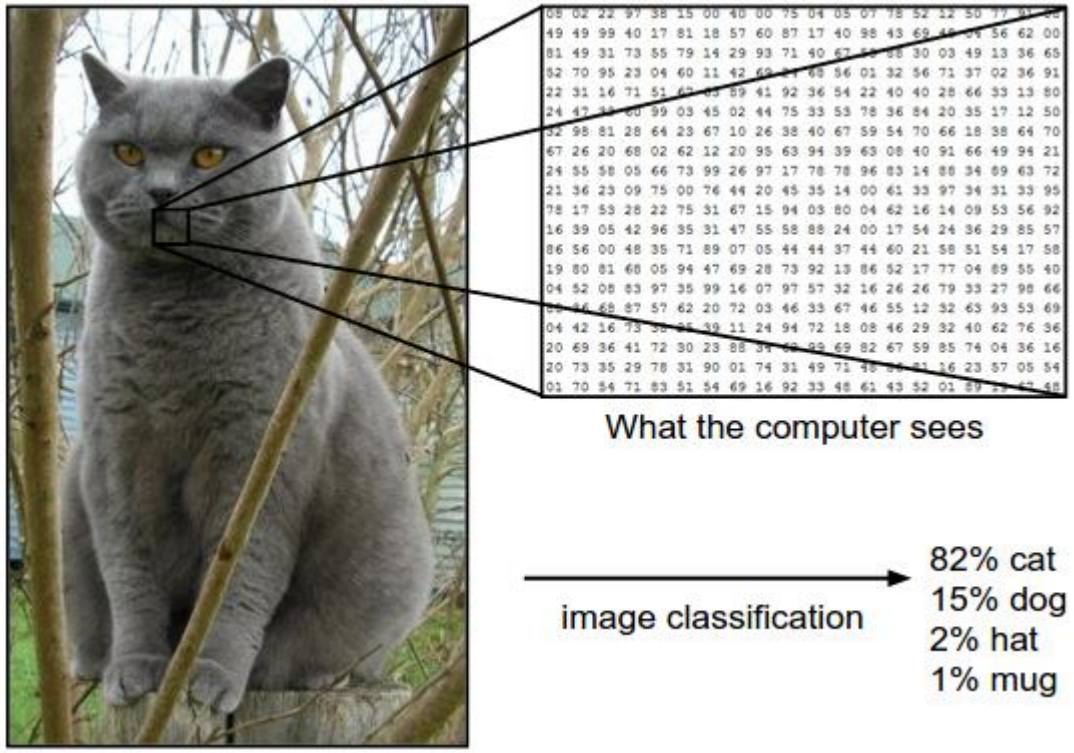
- RGB: three-dimensional array, one channel per color



S. M. Masud Karim, M. S. Rahman and M. I. Hossain, "A new approach for LSB based image steganography using secret key," *14th International Conference on Computer and Information Technology (ICCIT 2011)*, Dhaka, 2011, pp. 286-291.

Task – Classification

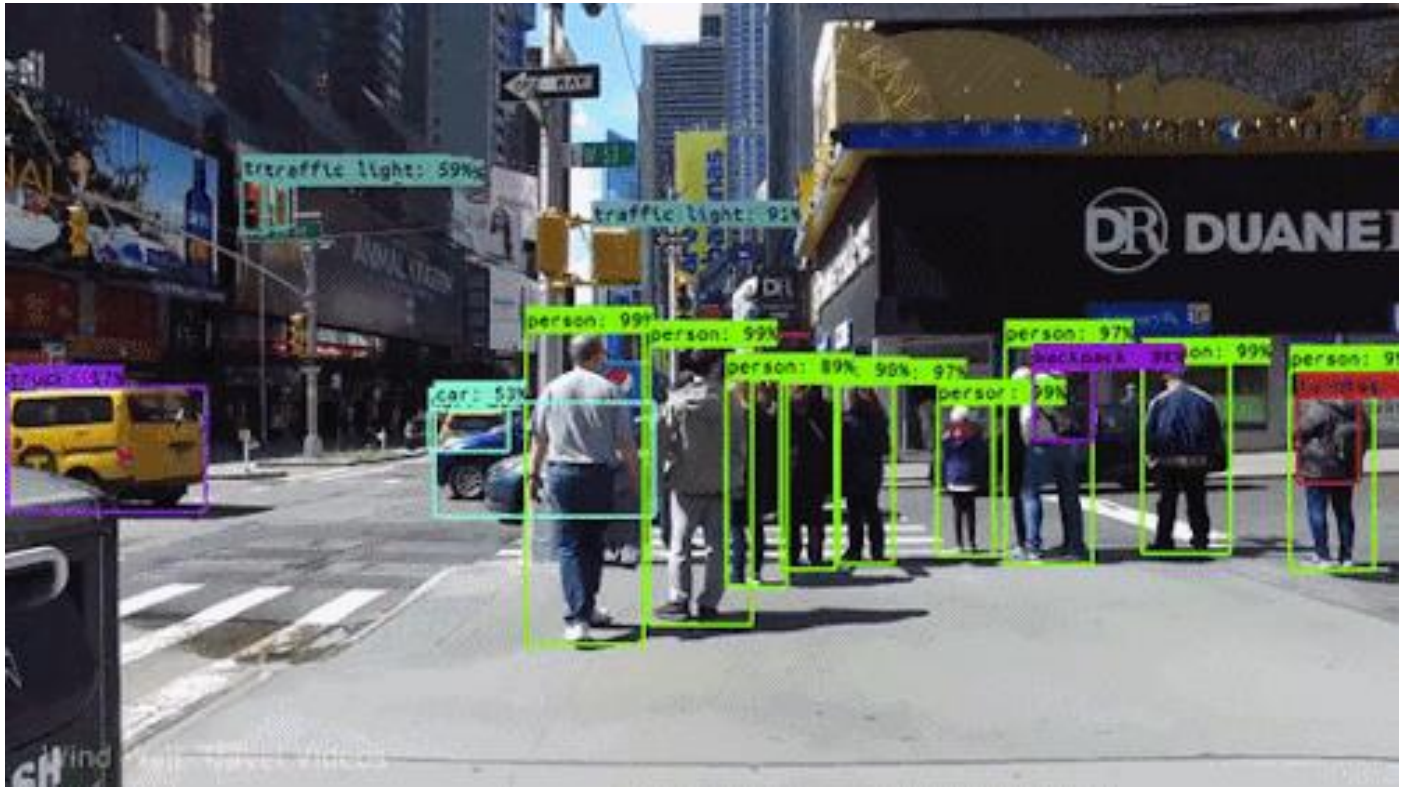
Map image to a set classes, e.g. $\{cat, dog, hat, mug\}$



CS231n Convolutional Neural Networks for Visual Recognition – Stanford University

Task – Object Detection (Bounding Boxes)

Where in the image are objects, what **class** do they belong to?



CS231n Convolutional Neural Networks for Visual Recognition – Stanford University

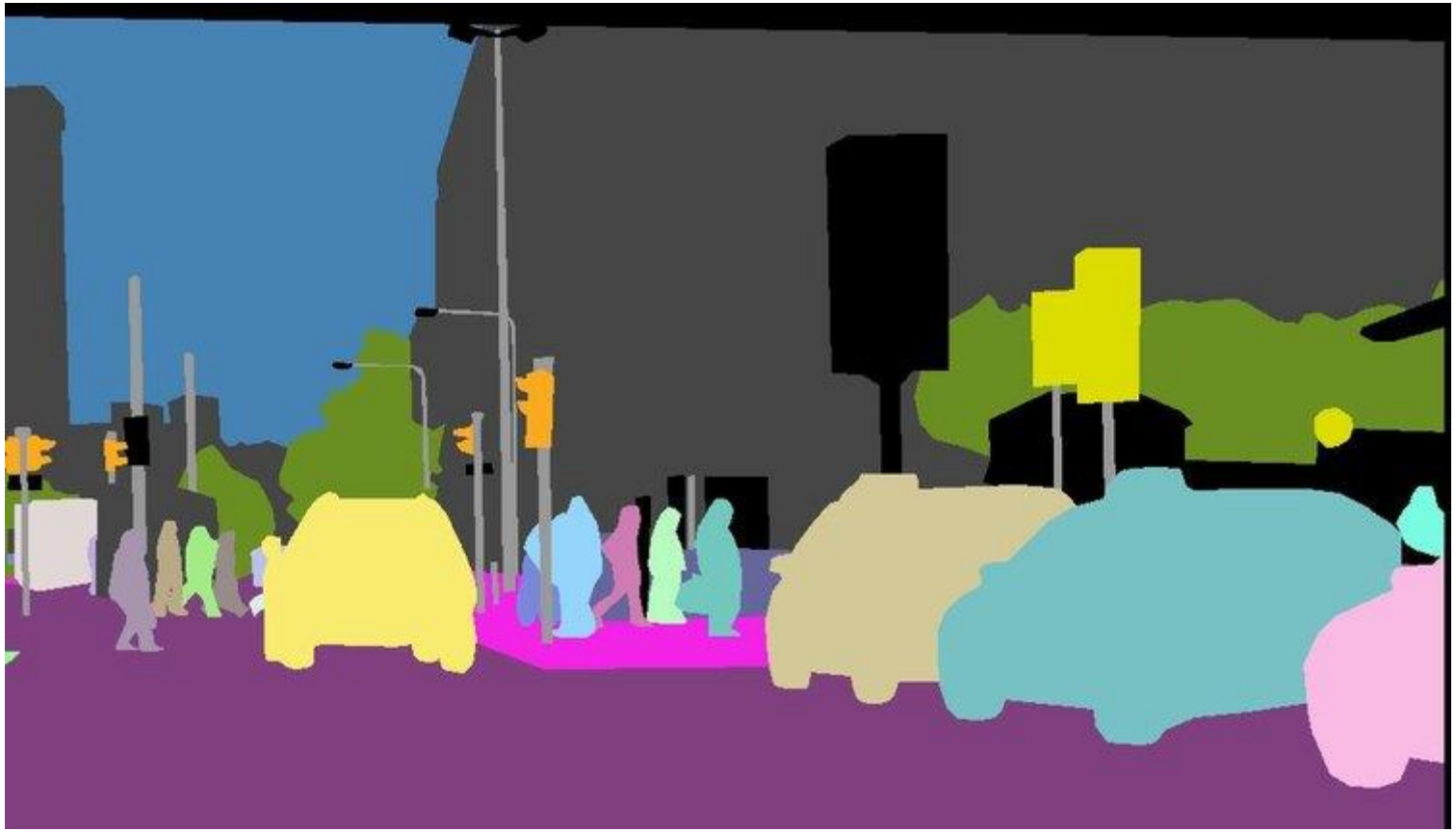
Task – Semantic Segmentation

Assign **class** to each pixel (e.g. road surface, vehicle, pedestrian)

Cordts, Marius, et al. "The cityscapes dataset for semantic urban scene understanding." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

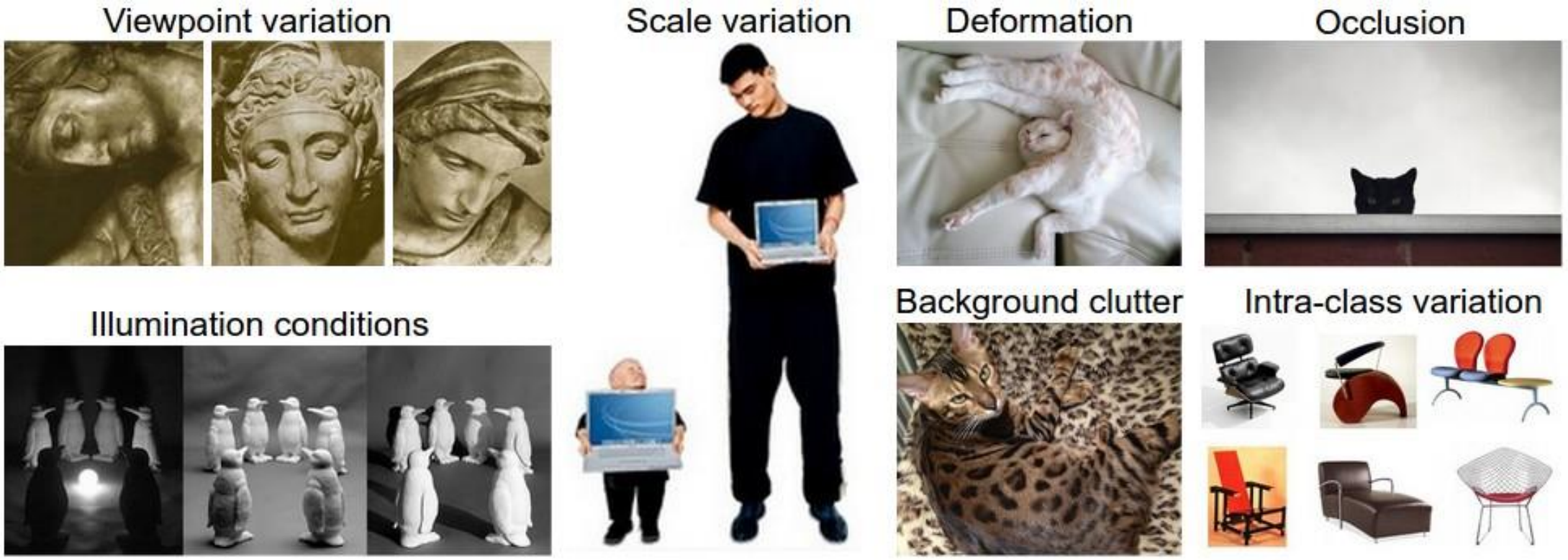
Task – Instance Segmentation

Assign **class** and **object instance** to each pixel



Cordts, Marius, et al. "The cityscapes dataset for semantic urban scene understanding." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Challenges

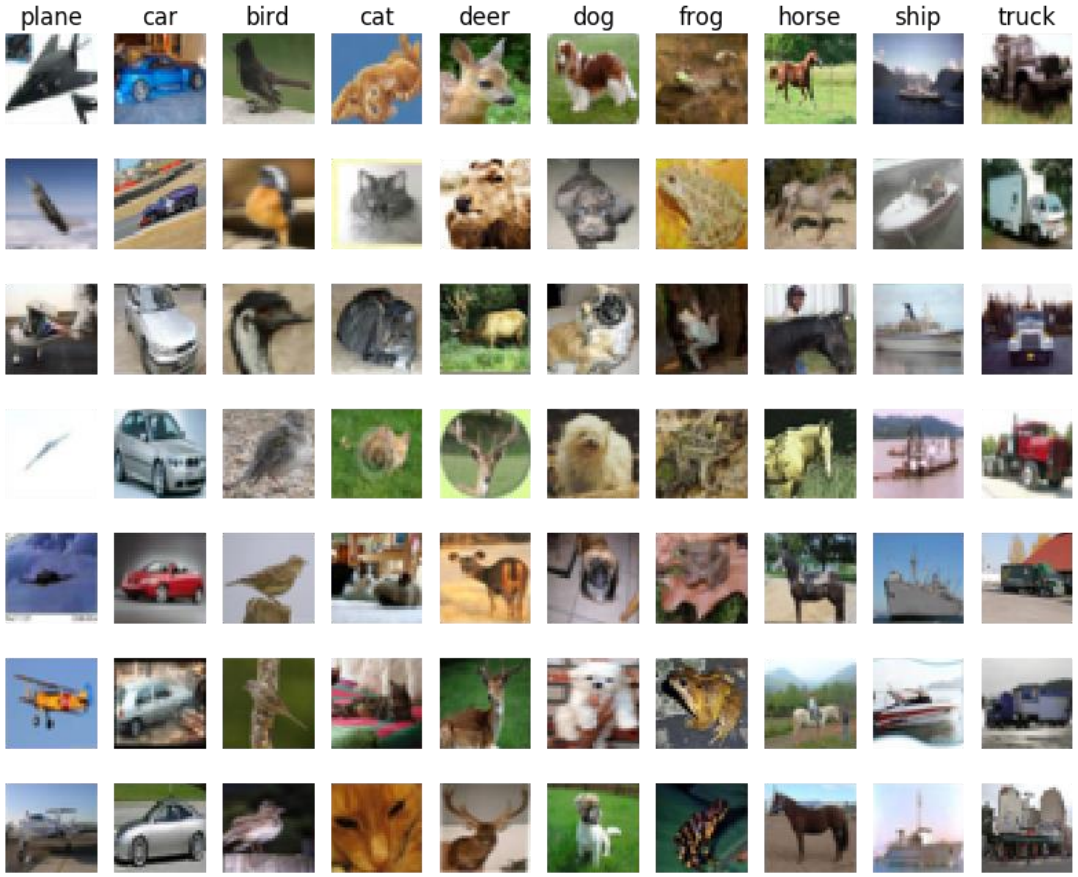


- How to write an algorithm that handles all edge cases?

CS231n Convolutional Neural Networks for Visual Recognition – Stanford University

Computer Vision - Data Driven Approach

1. Collect Dataset $\mathcal{D} = \{(y_n, x_n)\}_{i=1}^{N}$
   $x_n \in X$ - input data (Camera image, LIDAR, …)
   $y_n \in \mathcal{C}$ - class of x_n
   $\mathcal{C} = \{C_j\}_{j=1}^{L}$ - classes (plane, car, bird, cat)
2. Choose model $f_{\boldsymbol{\theta}}(x): X \mapsto \mathcal{C}$
3. **Training**: learn "best" parameter $\boldsymbol{\theta}_*$ from $\mathcal{D}$
4. Deployment: Apply $f_{\boldsymbol{\theta}_*}$ to new input $\boldsymbol{x}$



Data Driven Approach – Toy Example

- Linear classifier $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$
  - $\boldsymbol{x}$ – 64x64 input image (grayscale)
  - $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{b})$ – parameters of classifier
  - $\mathcal{C} = \{Cat, Dog, Human\}$

64x64 = 4096 Pixel

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underset{3x4096}{\boldsymbol{W}}\boldsymbol{x} + \underset{3x1}{\boldsymbol{b}} = \begin{array}{|c|l} \hline 100 & Cat \\ \hline 20 & Dog \\ \hline -3 & Human \\ \hline \end{array}$$

Reshape to
1x4096 matrix

How to find $\boldsymbol{W}$ and $\boldsymbol{b}$?

Data Driven Approach - Training

Training set (~80%) $\mathcal{D}_{Train} = \{(y_i, x_i)\}_{i=1}^{K}$	Test set (~20%) $\mathcal{D}_{Test} = \{(y_i, x_i)\}_{i=K+1}^{N}$

- **Training:** find ”best” values for $\boldsymbol{W}$ and $\boldsymbol{b}$
- **Loss** $J(\theta) \in \mathbb{R}$ measures performance of f_θ on $\mathcal{D}_{Train}$

$$J(\theta) = \sum_{i=1}^{K} \left(y_i - f_\theta(x_i)\right)^2$$

- Minimize Loss using optimization

$$\boldsymbol{\theta}_* = arg\min_{\theta} J(\theta)$$

Use-case Dependent! Here: Least-Squares, closed-form solution exists for linear f_θ

Data Driven Approach - Testing

Training set (~80%) $\mathcal{D}_{Train} = \{(y_i, x_i)\}_{i=1}^{K}$	Test set (~20%) $\mathcal{D}_{Test} = \{(y_i, x_i)\}_{i=K+1}^{N}$

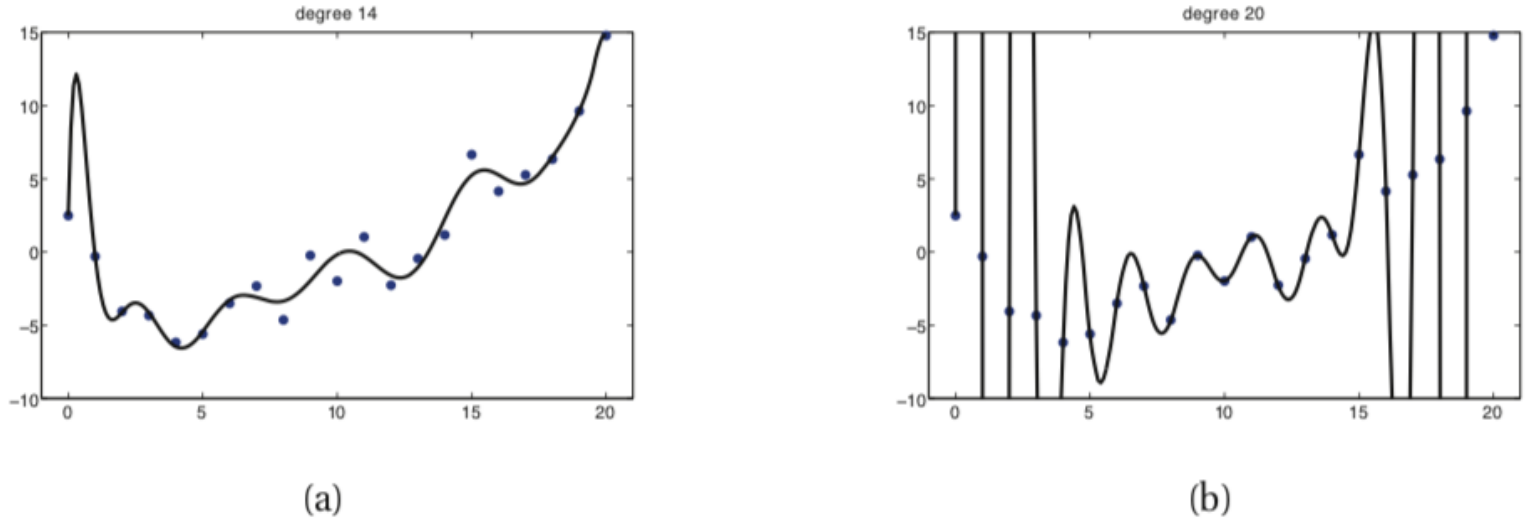
- Measure performance of f_{θ_*} on $\mathcal{D}_{Test}$

$$J(\theta_*) = \sum_{i=K+1}^{N} \left(y_i - f_{\theta_*}(x_i)\right)^2$$

- Performance of f_{θ_*} on $\mathcal{D}_{train}$ may be great, but horrible on $\mathcal{D}_{Test}$

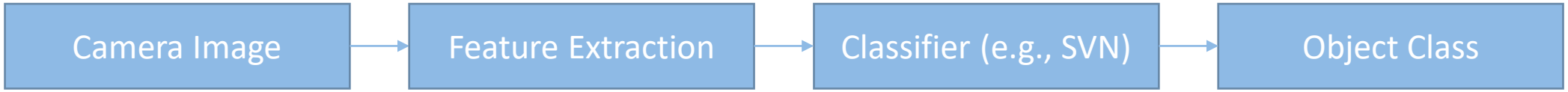
Data Driven Approach – Overfitting

- Choosing the right model $f_{\boldsymbol{\theta}}$ is application specific (trial-and-error)
- Too “simple”: cannot capture complex relationships
- Too “flexible”: danger of overfitting to noise
  - e.g. Trainig set may be memorized during training

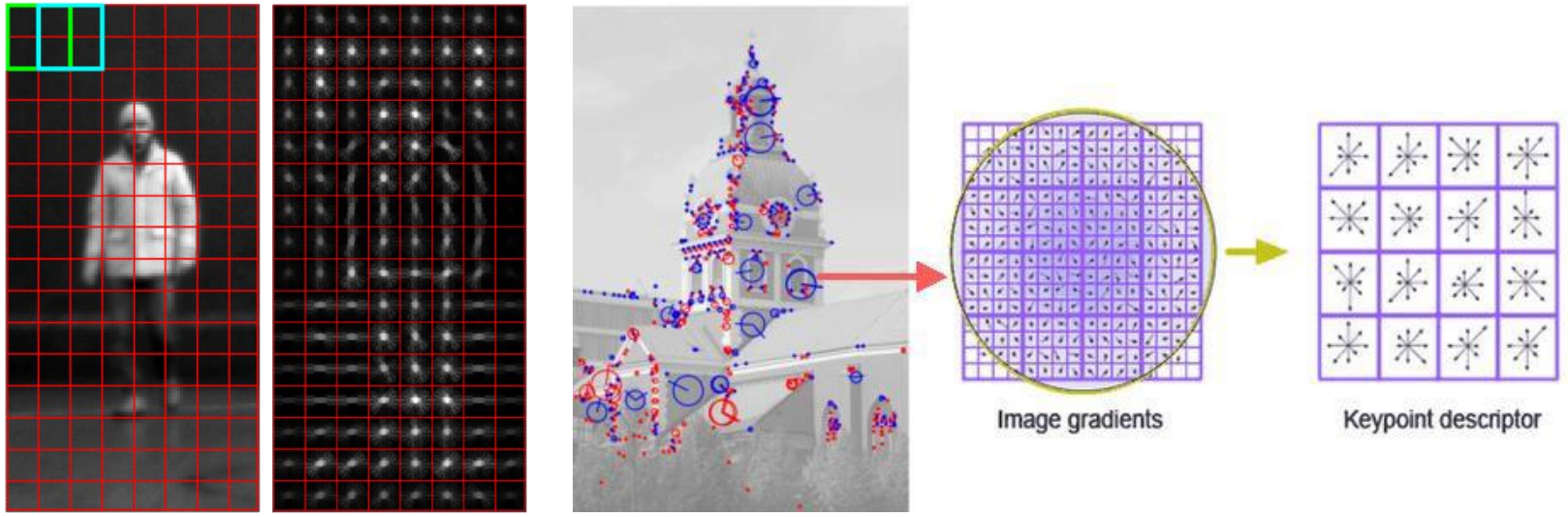


Robert, Christian. "Machine learning, a probabilistic perspective." (2014)

Computer Vision Pipeline – Before Deep Learning



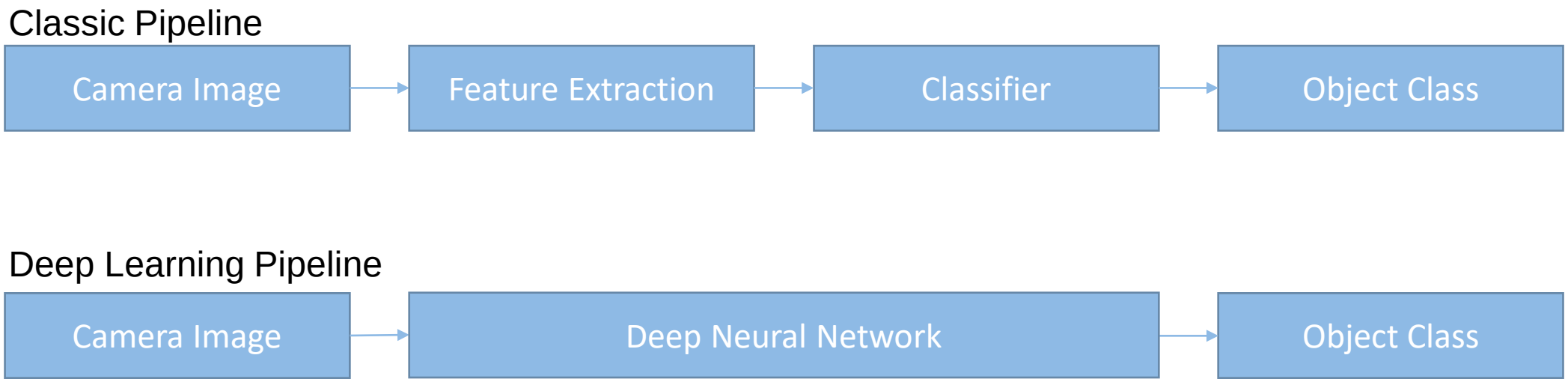
- Features: Histogram of Oriented Gradients (HOG), SIFT, SURF, ORB, ...
- **Problem**: features are highly application specific, “hand-crafted features”



N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*

Deep Learning Basics

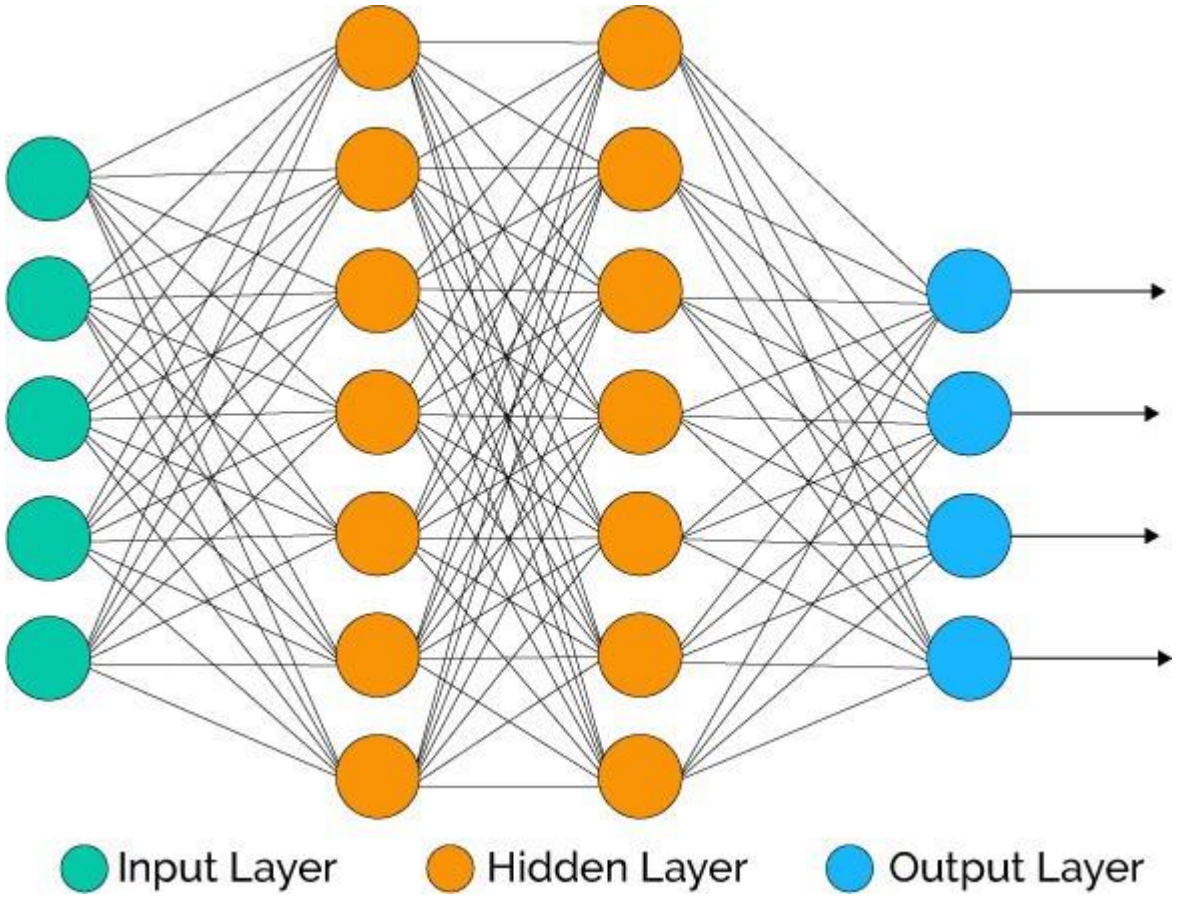
Deep Learning Computer Vision Pipeline



- Feature Extraction and Classification performed in one step
- Milestone Paper: Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in Neural Information Processing Systems*. 2012.

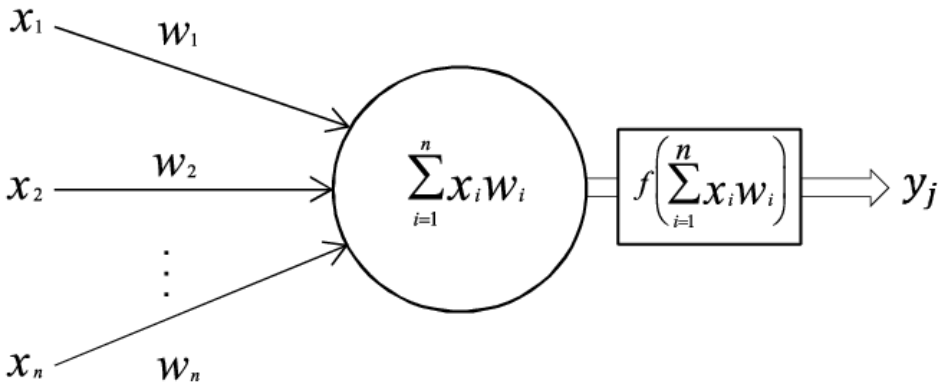
Deep Learning – Neural Network Building Blocks

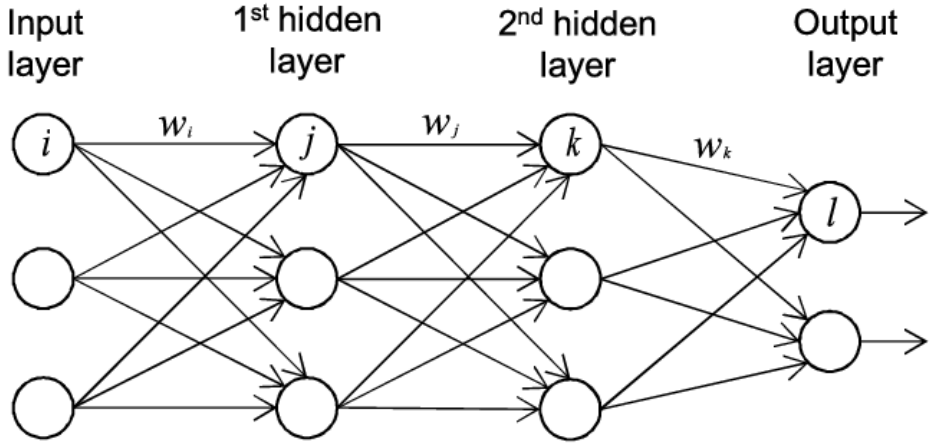
- Input layer, e.g. RGB Values
- Sequence of hidden layer
  - Different intermediate layer types: Convolutional, Fully-Connected, Pooling, …
- Output Layer indicates class



Deep Learning – Building Blocks

- Weights $\boldsymbol{\theta} = (\mathbf{w_1}, \ldots, \boldsymbol{w_n})$ learnable parameter of the network
- Output is weighted sum of outputs from previous layers with activation function
- Non-linear *activation function* f
  - e.g. ReLU $f(x) = \max(0, x)$
- Output y_i serves as input to neurons in next layer





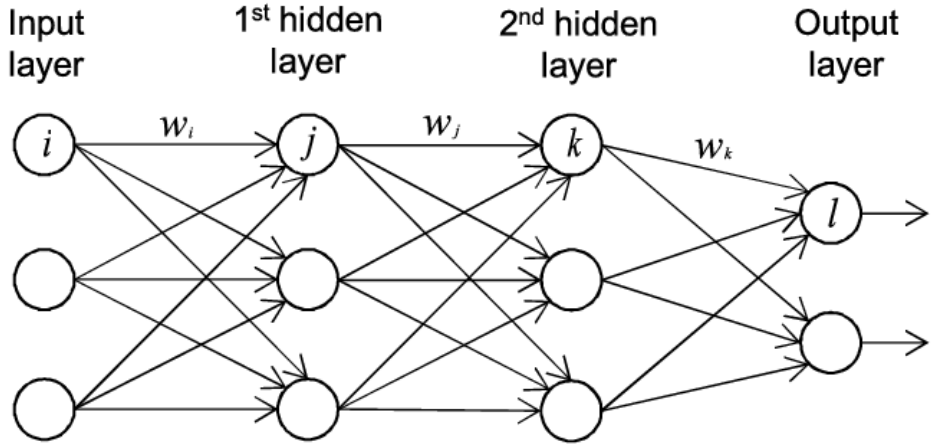
Vieira, Sandra, Walter HL Pinaya, and Andrea Mechelli. "Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications." *Neuroscience & Biobehavioral Reviews* 74 (2017): 58-75.

Training Neural Networks – Loss Function

- **Use-case**: Classification with classes $C_1, \dots, C_L$
- Output of final layer $o_1, \dots, o_L$
- Normalize output into probability distribution using **softmax** function

$$\sigma(o_i) = \frac{e^{o_i}}{\sum_{j=1}^{L} e^{o_j}} \in (0,1)$$

$$\sum_{j=1}^{L} \sigma(o_i) = 1$$



26	Softmax →		~0.72
25			~0.268
20			~0.001

- Network outputs probability distribution for data sample x_i over all possible classes

Training Neural Networks – Loss Function

- Reminder: true class for sample x_i is y_i
- Predicted probability distribution $P(\mathrm{C_i} \mid x_i) = \sigma(o_i)$
- True probability distribution for sample $Q(\mathrm{C_i} \mid x_i) = \begin{cases} 1, & C_i = y_i \\ 0, & C_i \neq y_i \end{cases}$

Example

	$P(C_i \mid x_i)$	$Q(C_i \mid x_i)$
C_1	0.72	0
$C_2 = C_{GT}$	0.268	1
C_3	0.001	0

- Loss-Function is a measure of “distance” between predicted probability distribution and “true” probability distribution

Training Neural Networks – Cross-Entropy Loss

- Cross-entropy measures distance between two probability distributions

$$H_{x_i}(Q,P) = -\sum_{C_i \in \mathcal{C}} Q(C_i \mid x_i) log P(C_i \mid x_i)$$

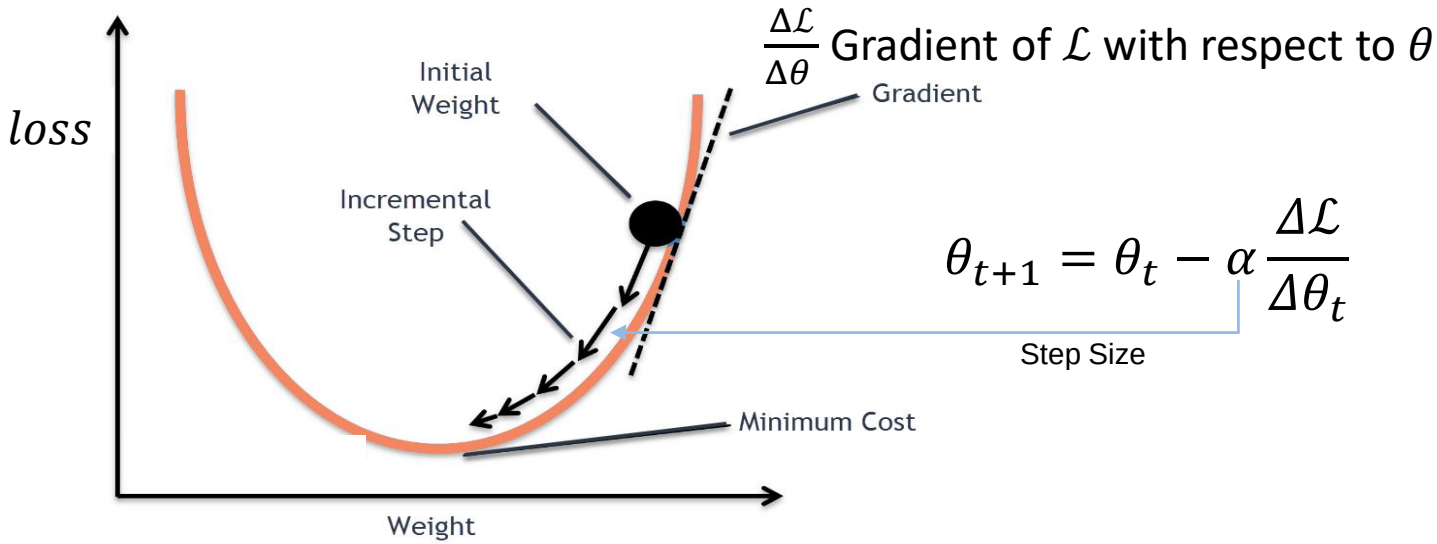
$$= -Q(C_{GT} \mid x_i) \log P(C_{GT} \mid x_i) = -\log P(C_{GT} \mid x_i) = -\boldsymbol{log}(\frac{e^{o_i}}{\sum_{j=1}^{L} e^{o_j}})$$

- Putting it all together, the **loss** for sample x_i and given network weights θ is

$$\mathcal{L}(\theta, x_i) = -\boldsymbol{log}(\frac{e^{o_i}}{\sum_{j=1}^{L} e^{o_j}})$$

Deep Learning – Optimization

- Apply Gradient Descent to minimize $\mathcal{L}(\theta, x_i)$



- Backpropagation: efficient algorithm for derivative calculation
- Loss function not convex; no guarantee for global minimum

Deep Learning – Training Overview

- Choose Neural Network Architecture
  - many proven architectures, e.g. AlexNet, VGG16, ResNet
- Obtain Dataset $\mathcal{D}_{Train} = \{(y_i, x_i)\}_{i=1}^{K}$, $\mathcal{D}_{Test} = \{(y_i, x_i)\}_{i=K+1}^{N}$

Initialize θ with random values
while(true)
for $(y_i, x_i) \in \mathcal{D}_{Train}$

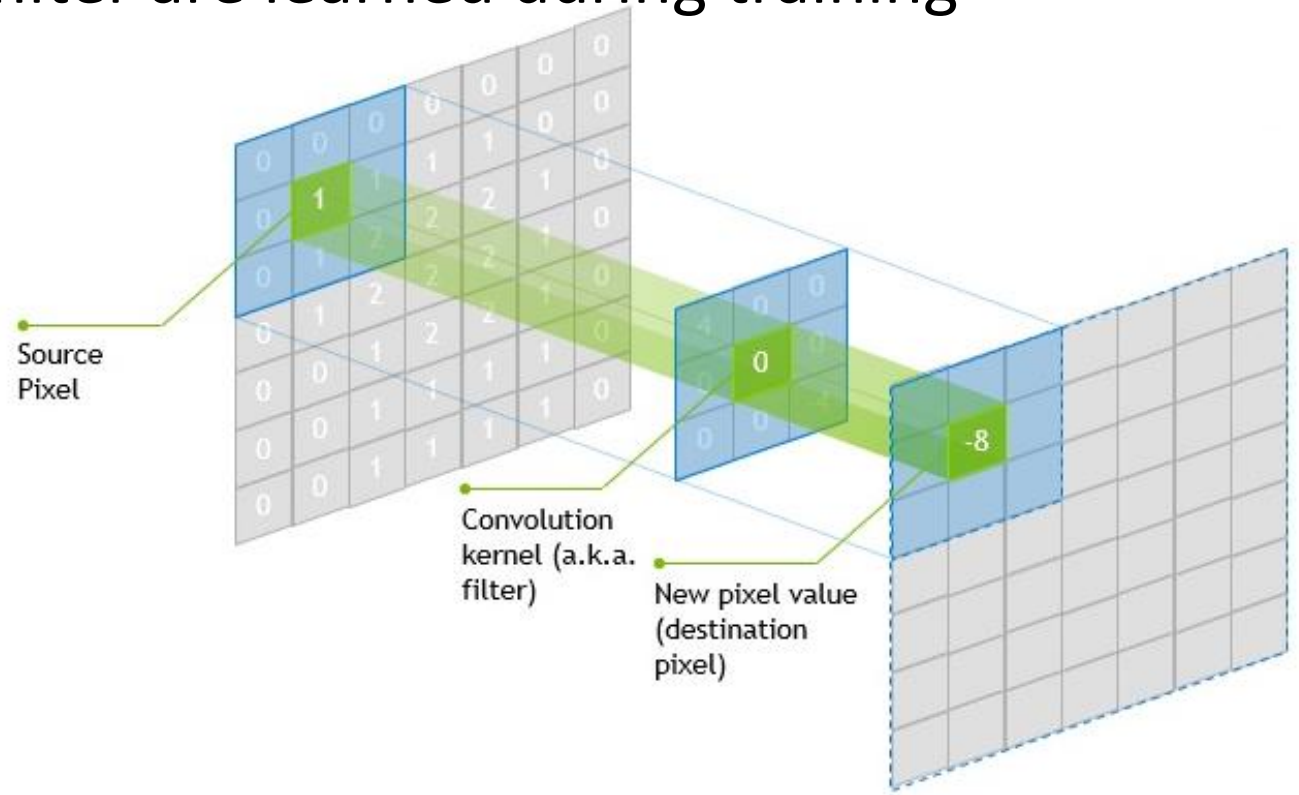
1. Compute forward pass *(inference)*, i.e. compute network output for x_i
2. Compute derivate of loss $\frac{\Delta \mathcal{L}_i}{\Delta \theta_t}$ using backpropagation
3. Update $\theta_{t+1} = \theta_t - \alpha \frac{\Delta \mathcal{L}_i}{\Delta \theta_t}$ (step size in practice ~ 0.01)

Compute current accuracy on $\mathcal{D}_{Test}$ using current best θ
Terminate if accuracy does not improve, iteration limit, …

- Training requires GPU, training/inference on CPU usually too slow

Deep Learning – Convolutional Layers

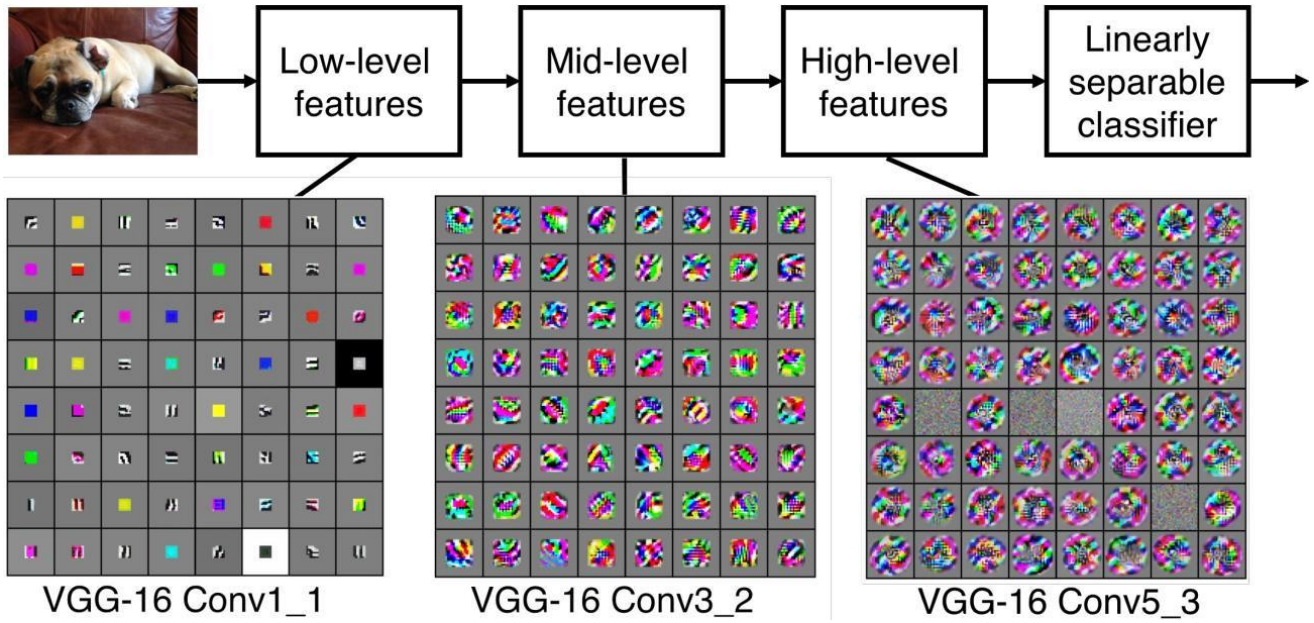
- Layer consists of learnable convolution filters
- Output of layer $n-1$ filtered in layer n, serves as input to layer $n+1$
- Weights of the filter are learned during training



https://www.youtube.com/watch?v=f0t-OCG79-U

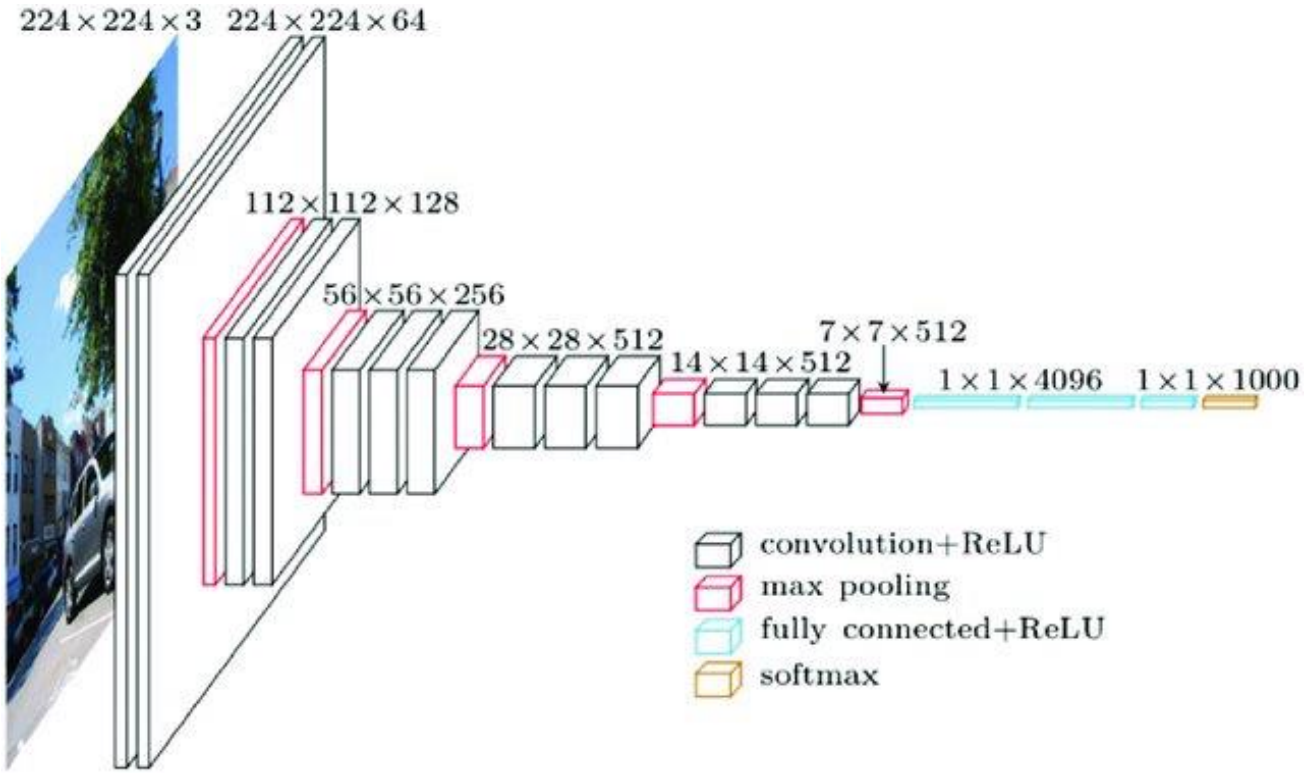
Deep Learning – Convolutional Neural Network

- Filter react to specific features (edges, blobs, faces, hands, people,…)
- Tends to become more abstract for layers deeper within the network



Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.

Case Study – VGG16 Architecture



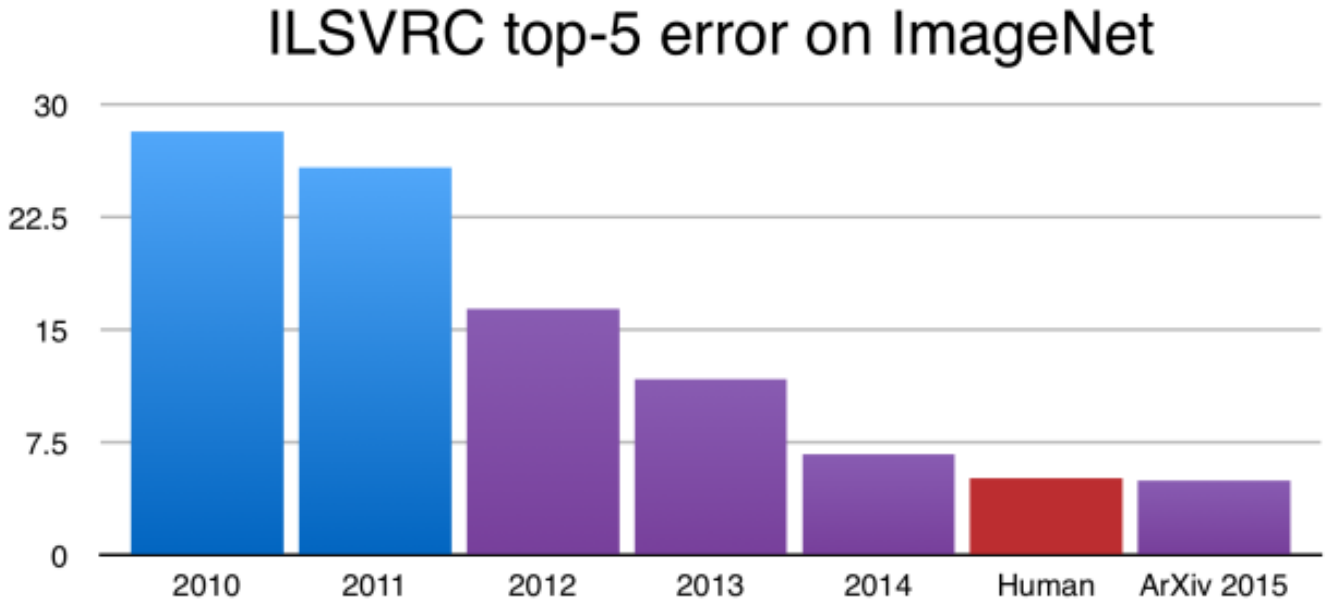
Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *ICLR 2015*

Deep Learning – Scratching the Surface

- Layer Types
  - Pooling
  - Upsampling
  - Batching
  - Dropout/Regularization
- Losses
  - Loss for Semantic Segmentation
  - Loss for Object Detection (Bounding Boxes)
  - …
- Training
  - Transfer Learning
  - …

Deep Learning – Performance

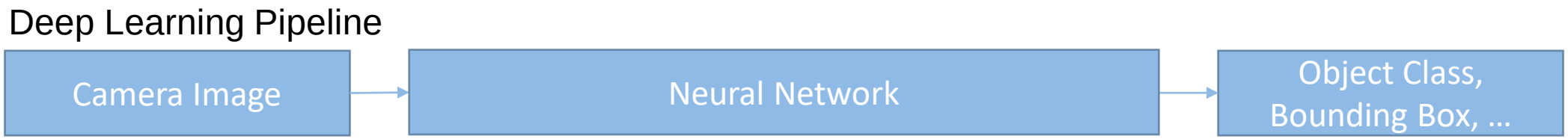
- Deep Neural Networks have become state-of-the-art for perception problems



https://devblogs.nvidia.com/mocha-jl-deep-learning-julia/

- Many other applications: Visual odometry, LIDAR point clouds, Radar, depth from mono camera, prediction, …

Deep Learning für Computer Vision



- No more „Feature Engineering“
- End-To-End Learning of Feature Extraction & Classifier
- Massive improvements on performance of perception systems
- New applications become possible
- Requires lots of data and GPUs
- State-of-the-art often intractable for real-time applications

Deep Learning für Computer Vision – How to start?

Open Datasets: Cityscapes, KITTI, Apollo, Oxford RoboCar, …

Frameworks

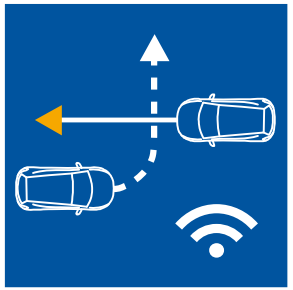
- Tensorflow, Keras, PyTorch, …
- GPU for Training/Inference

Open-source culture: many papers release code on github!

Top Conferences: CVPR, ECCV, ICRA, NIPS, ITSC, IV, …

Further literature (1)

- Thrun, S., Burgard, W., & Fox, D. (2000). *Probabilistic robotics*.
- Robert, C. (2014). *Machine learning, a probabilistic perspective*.
- Theodoridis, S. (2015). *Machine learning: a Bayesian and optimization perspective*. Academic Press.



Lecture

Control and Perception in Networked and Autonomous Vehicles

Simon Schäfer, M. Sc. | Dr. Bassam Alrifaee | Patrick Scheffe, M. Sc.

Winter Semester 2023/2024

Part 5

State Estimation and Kalman Filter with KARL the Little

Agenda

- State Estimation
- Kalman Filter
- Hands-on Example

State Estimation

Definition: State Estimation

- Open-Loop Observers
- Luenberger Observer
- Kalman Filter
- Extended Kalman Filter
- Unscented Kalman Filter
- Particle Filter
- …

State Estimation

State estimation refers to the process of determining or approximating the current internal state of a system based on available information. This process combines **mathematical models**, **input data**, and **observations** to infer the most likely state of the system at a given moment, even when the complete information might not be directly measurable or available.

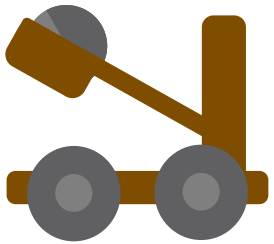
Definition: State Estimation

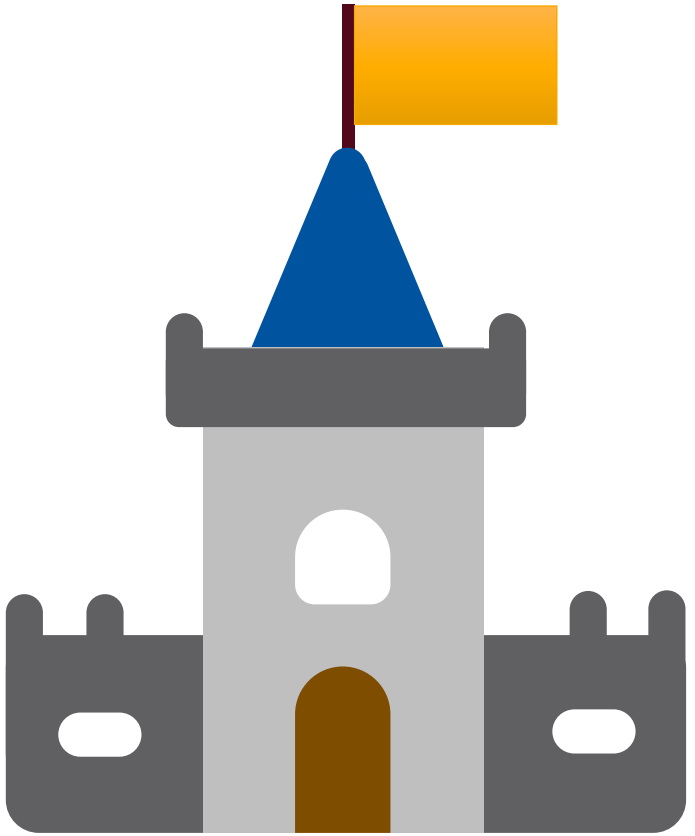
- **Open-Loop Observers**
- Luenberger Observer
- **Kalman Filter**
- Extended Kalman Filter
- Unscented Kalman Filter
- Particle Filter
- …

State Estimation

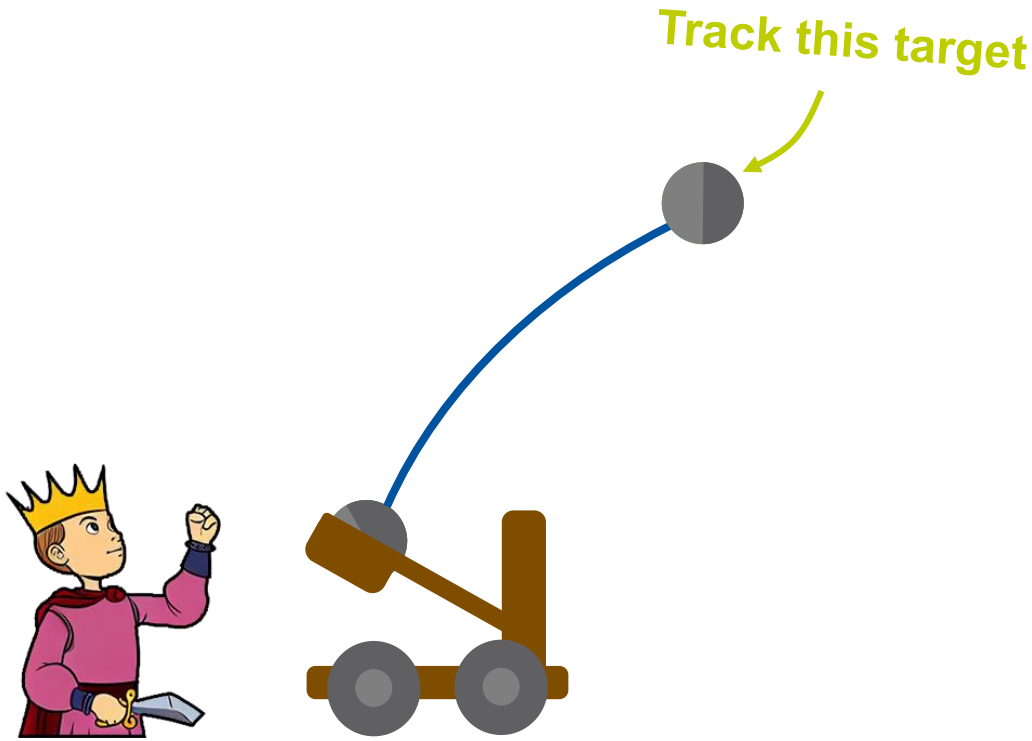
State estimation refers to the process of determining or approximating the current internal state of a system based on available information. This process combines **mathematical models**, **input data**, and **observations** to infer the most likely state of the system at a given moment, even when the complete information might not be directly measurable or available.

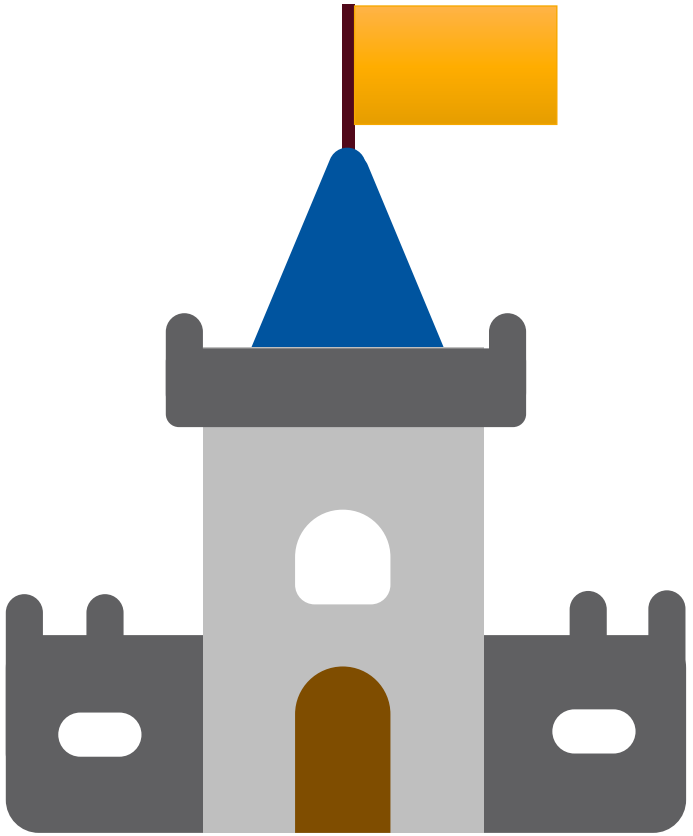
Hands-on Example



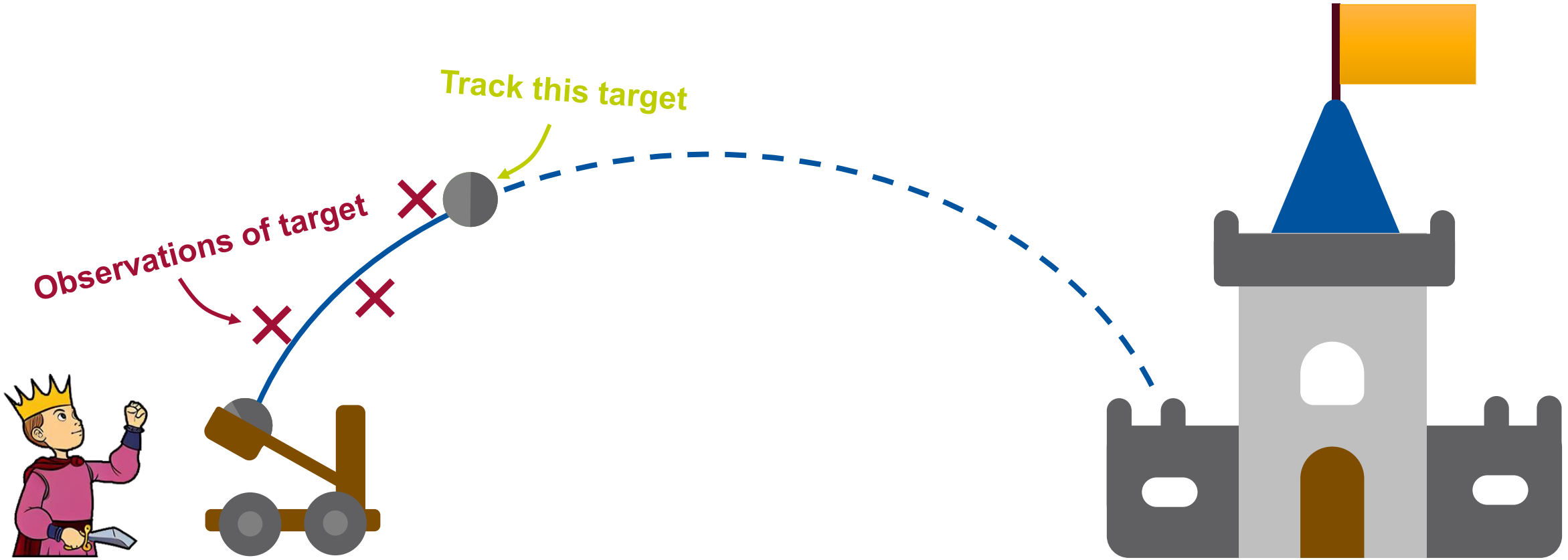


Hands-on Example





Hands-on Example



Hands-on Example

What do we want to know?

“Where is the stone at a given point in time?”

- Wanted:
  - State estimation of target
  - Predict arrival point

What do we know?

“We observe the trajectory of the stone?”

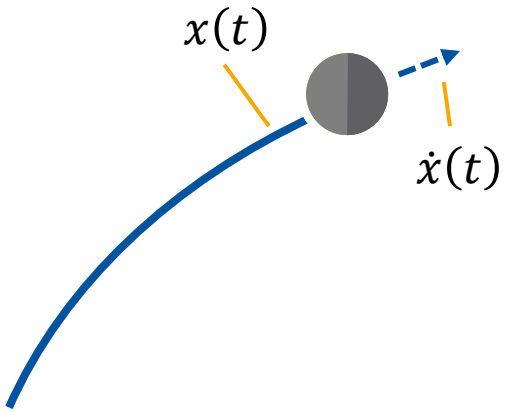
- Given:
  - Observations of target

There is a catch.

“The observations are not perfect.”

- Needed:
  - Probabilistic model to allow for uncertainties
  - Model of the dynamics of the target

Definition: Model dynamics



Model Dynamics of Target

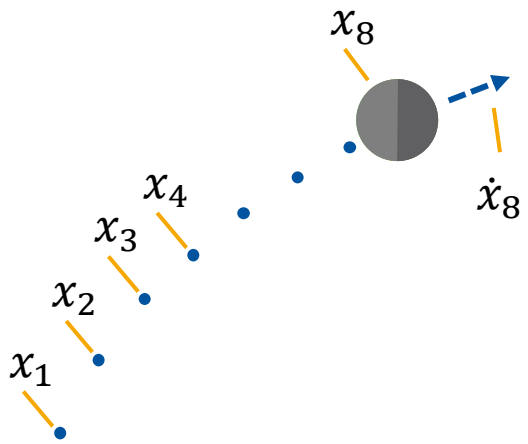
The state of a linear, continues-time system can be described with a state equation like

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) + \omega(t)$$

The system is observed through a linear equation of form

$$z(t) = H(t)x(t) + \eta(t)$$

Definition: Model dynamics



Model Dynamics of Target

The state of a linear, discrete-time system can be described with a state equation like

$$x_{k+1} = A_k x_k + B_k u_k + \omega_k$$

The system is observed through a linear equation of form

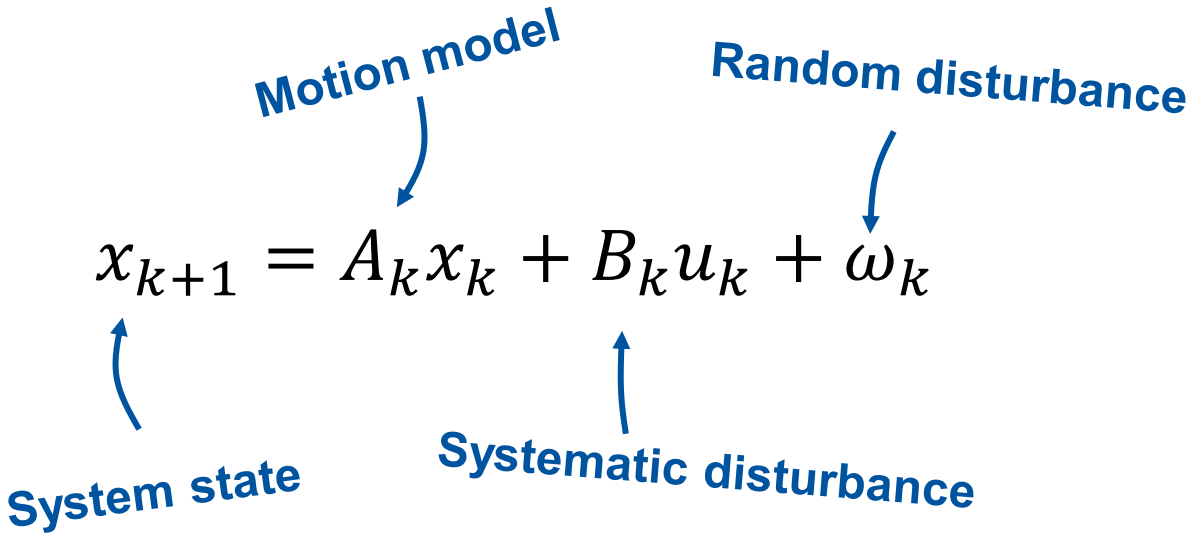
$$z_k = H_k x_k + \eta_k$$

Euler Method

Numerical integration method

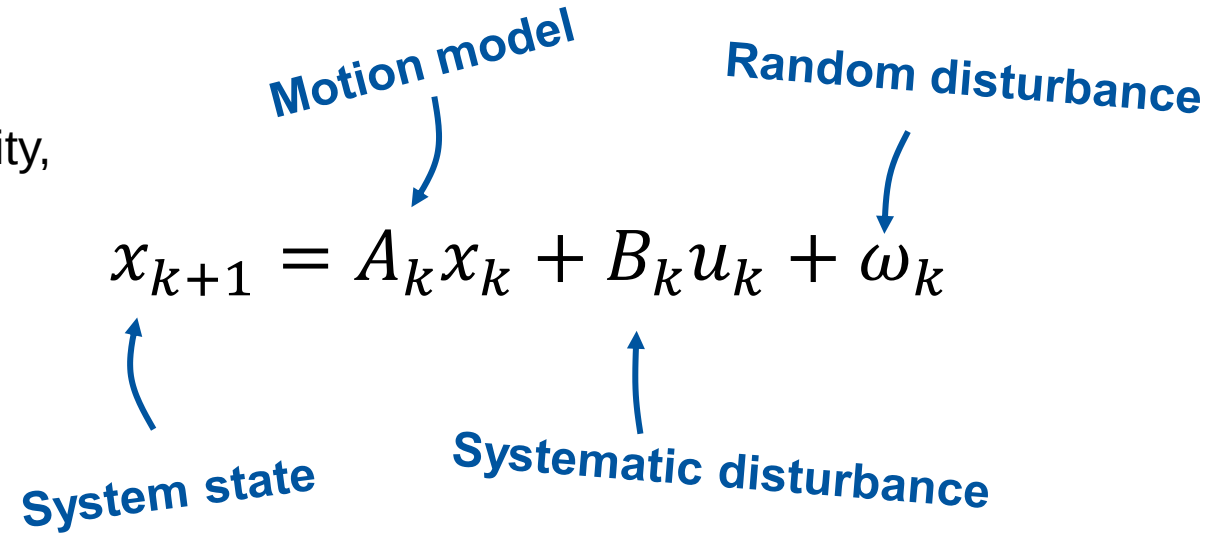
$$x_{k+1} \approx x_k + \Delta t \cdot \dot{x}_k$$

Definition: Model dynamics



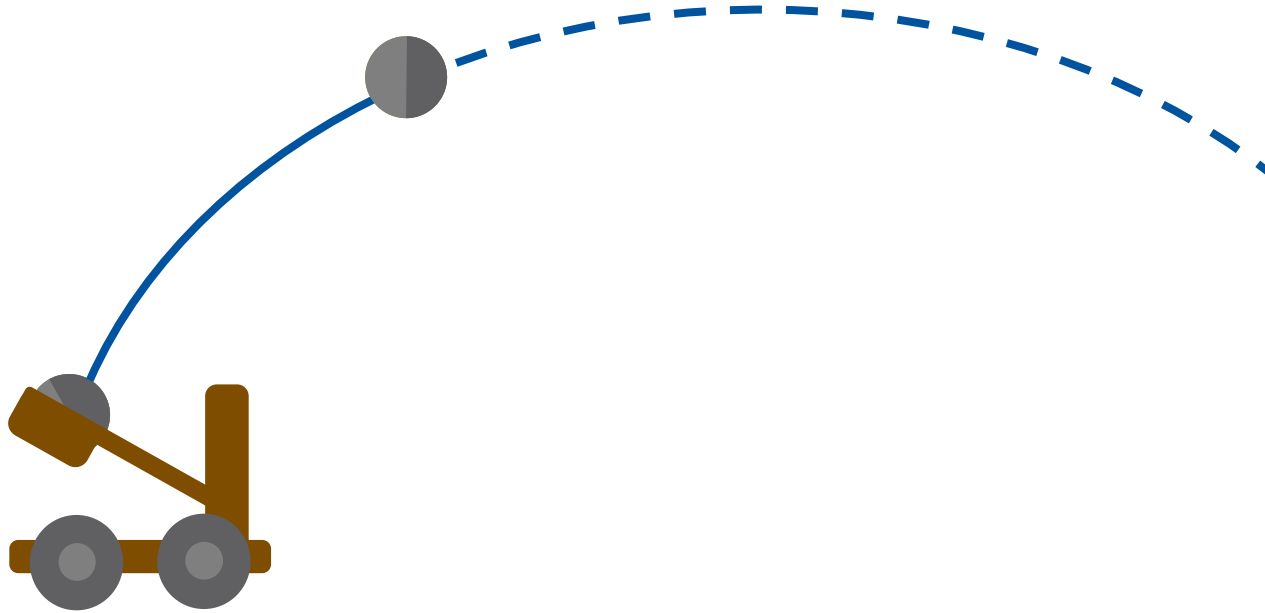
Definition: Model dynamics

- System state
  - Estimation of the current state of the target e.g., position, velocity, orientation, …
- Motion model
  - Description of the targets motion
  - Contains integration
- Systematic disturbance
  - Disturbances that are known in advance e.g., gravity, drag, …
- Random disturbance
  - Disturbances that are unknow or of random nature e.g., wind, friction, …



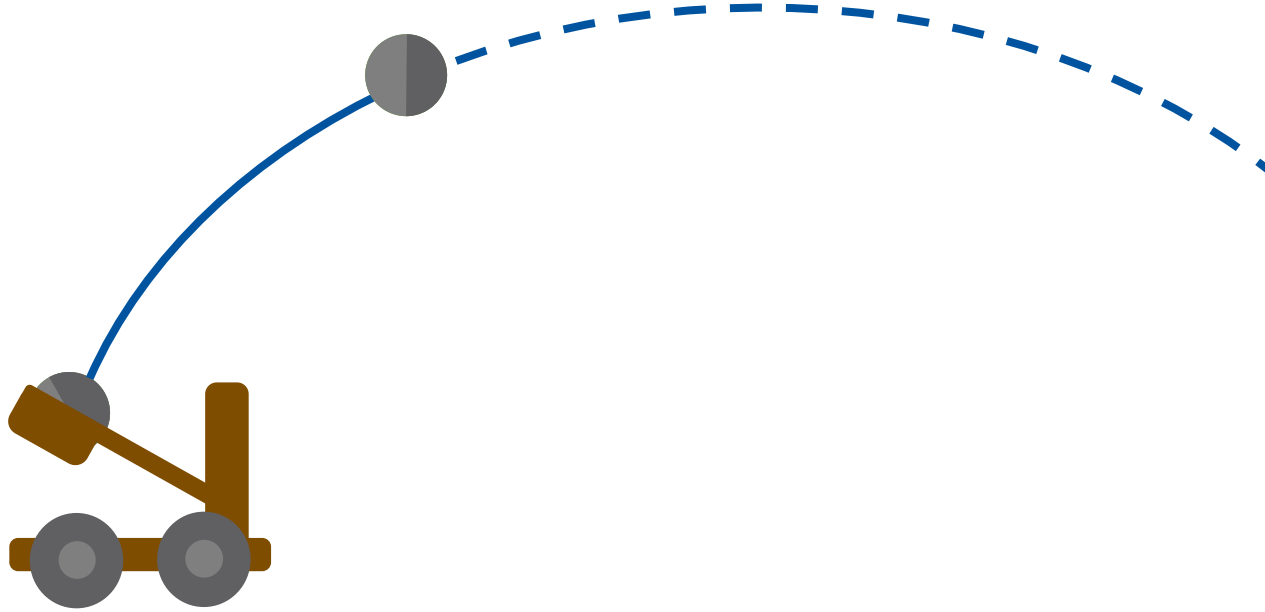
Hands-on Example

- System state
  -
- Motion model
  -
- Systematic disturbance
  -
- Random disturbance
  -

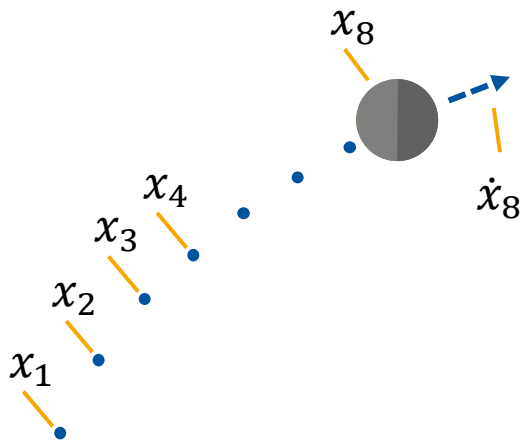


Hands-on Example

- System state
  - Position and velocity: $x = [s_x, s_y, v_x, v_y]^T$
- Motion model
  - Constant Velocity: $A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Systematic disturbance
  - Gravity: $B = [0 \quad 0 \quad 0 \quad \Delta t]^T [-g]$
- Random disturbance
  - Present but unknown



Definition: Model dynamics



Model Dynamics of Target

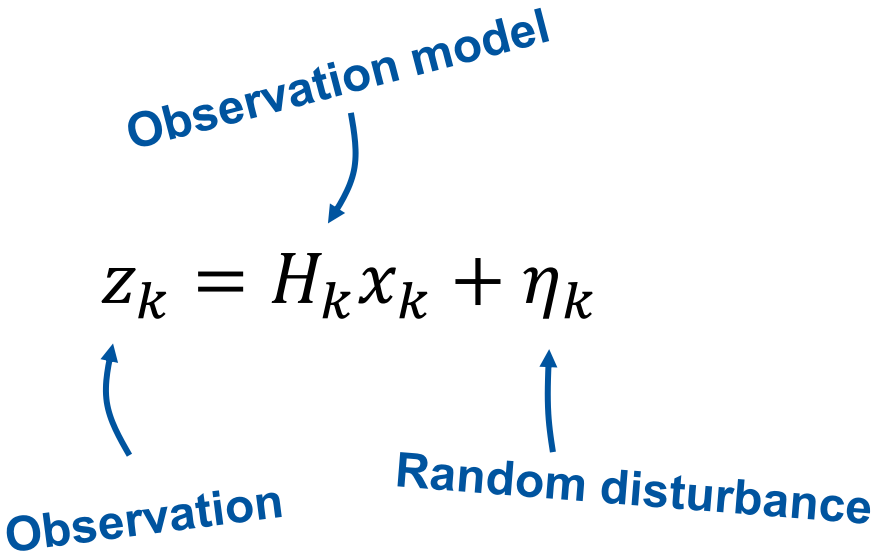
The state of a linear, discrete-time system can be described with a state equation like

$$x_{k+1} = A_k x_k + B_k u_k + \omega_k$$

The system is observed through a linear equation of form

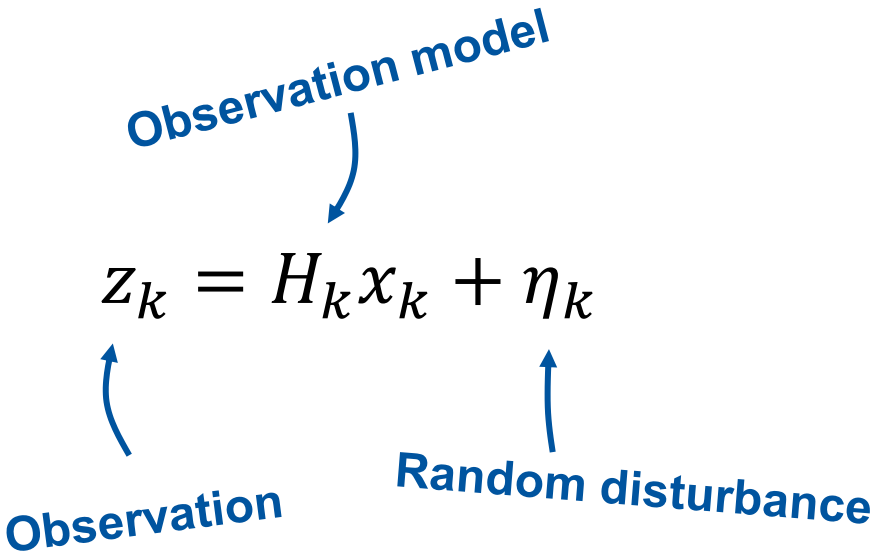
$$z_k = H_k x_k + \eta_k$$

Definition: Model dynamics



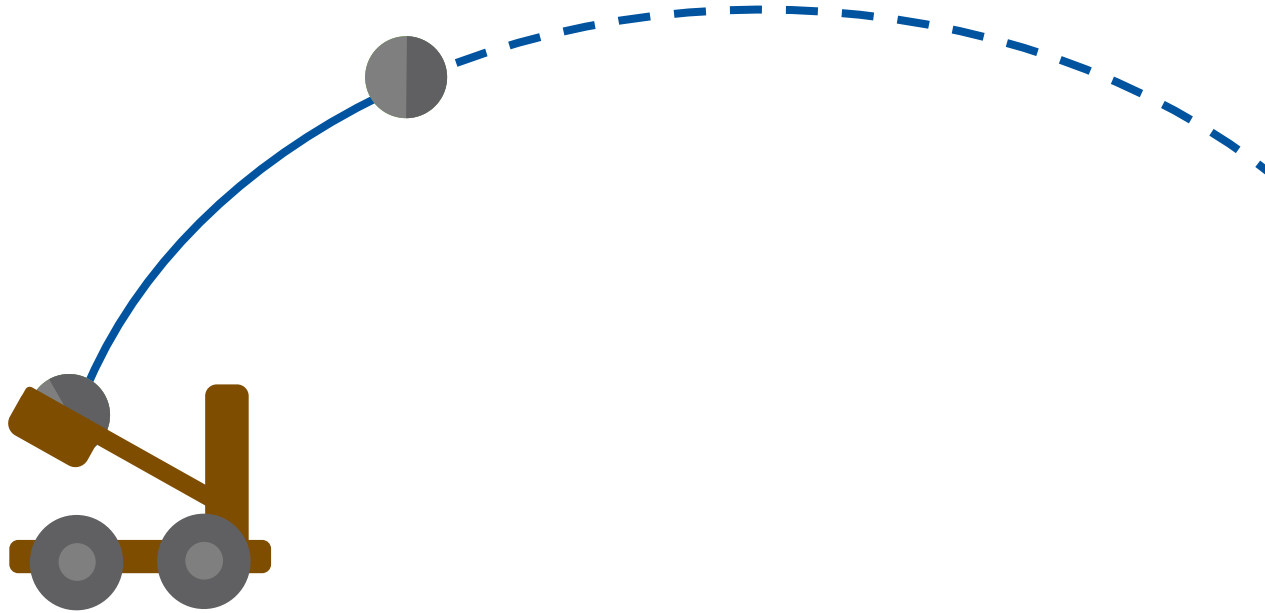
Definition: Model dynamics

- Observation
  - Measurement of the current state of the target e.g., position, velocity, orientation, …
- Observation model
  - Description the relation between the measured variable and the system state e.g., level of a tank, …
- Random disturbance
  - Disturbances that are unknow or of random nature e.g. sensor noise, reflections, …



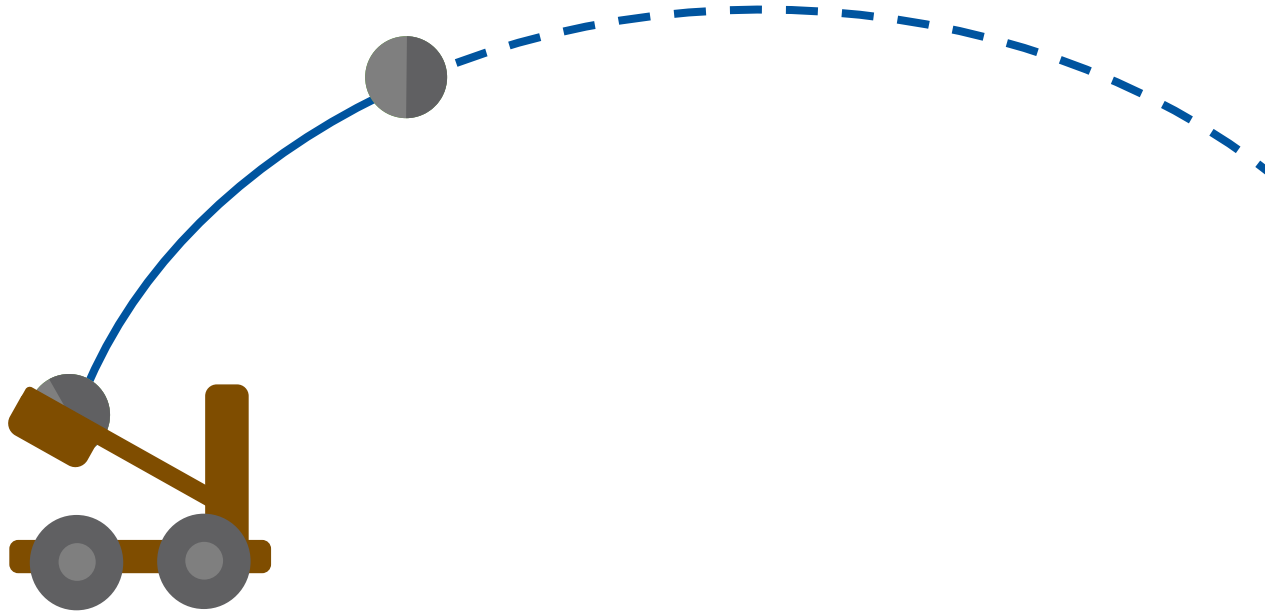
Hands-on Example

- Observation
  -
- Observation model
  -
- Random disturbance
  -



Hands-on Example

- Observation
  - Position: $\mathrm{z} = \left[s_x, s_y\right]^T$
- Observation model
  - Mapping: $\mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$
- Random disturbance
  - Present but unknown



Hands-on Example

- Observation
  - Position: $\mathrm{z} = \left[s_x, s_y\right]^T$
- Observation model
  - Mapping: $\mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$
- Random disturbance
  - Present but unknown

- System state
  - Position and velocity: $x = \left[s_x, s_y, v_x, v_y\right]^T$
- Motion model
  - Constant Velocity: $A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Systematic disturbance
  - Gravity: $B = [0 \quad 0 \quad 0 \quad \Delta t]^T [-g]$
- Random disturbance
  - Present but unknown

Hands-on Ex

Open-Loop Observer

Estimate state based on initial condition and dynamics.

$$x_{k+1} = A_k x_k + B_k u_k$$

$$= \begin{bmatrix} & & & \\ & & & \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- Random disturbance
  - Present but unknown

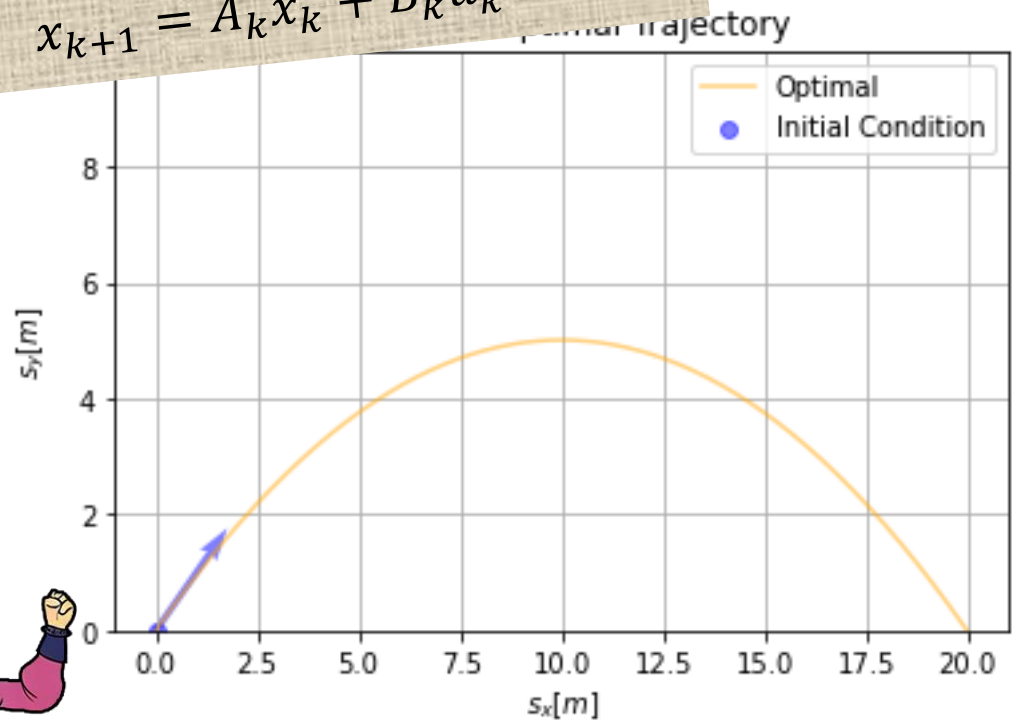
- System state
  - Position and velocity: $x = [s_x, s_y, v_x, v_y]^T$
- Motion model
  - Constant Velocity: $A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Systematic disturbance
  - Gravity: $B = [0 \quad 0 \quad 0 \quad \Delta t]^T [-g]$
- Random disturbance
  - Present but unknown

Hands-on Ex

Open-Loop Observer

Estimate state based on initial condition and dynamics.

$$x_{k+1} = A_k x_k + B_k u_k$$

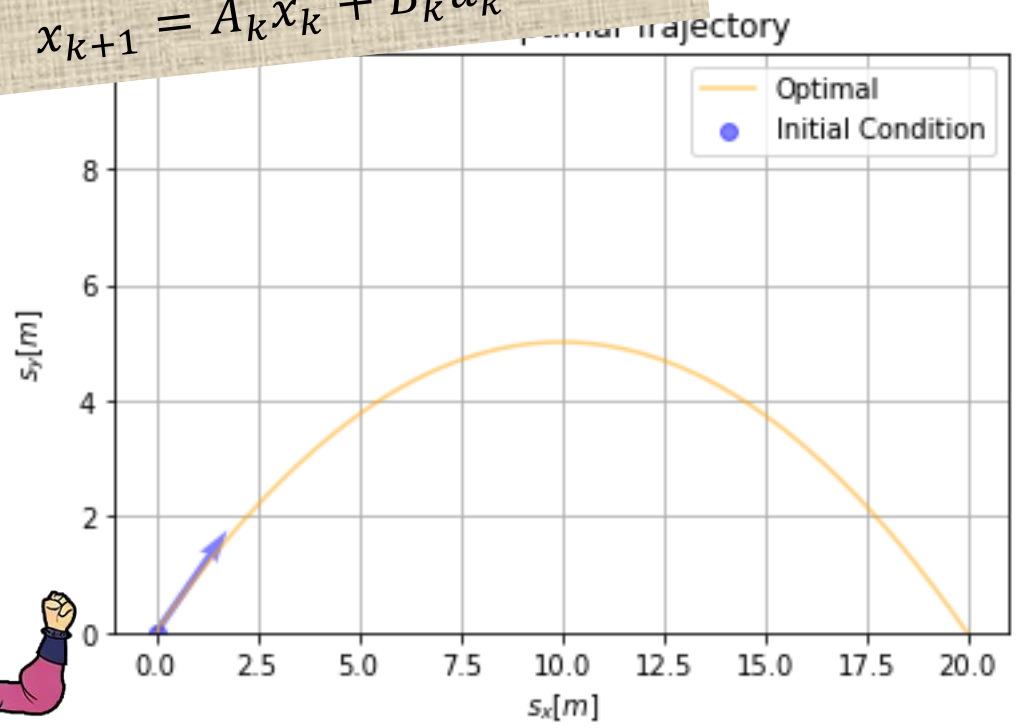


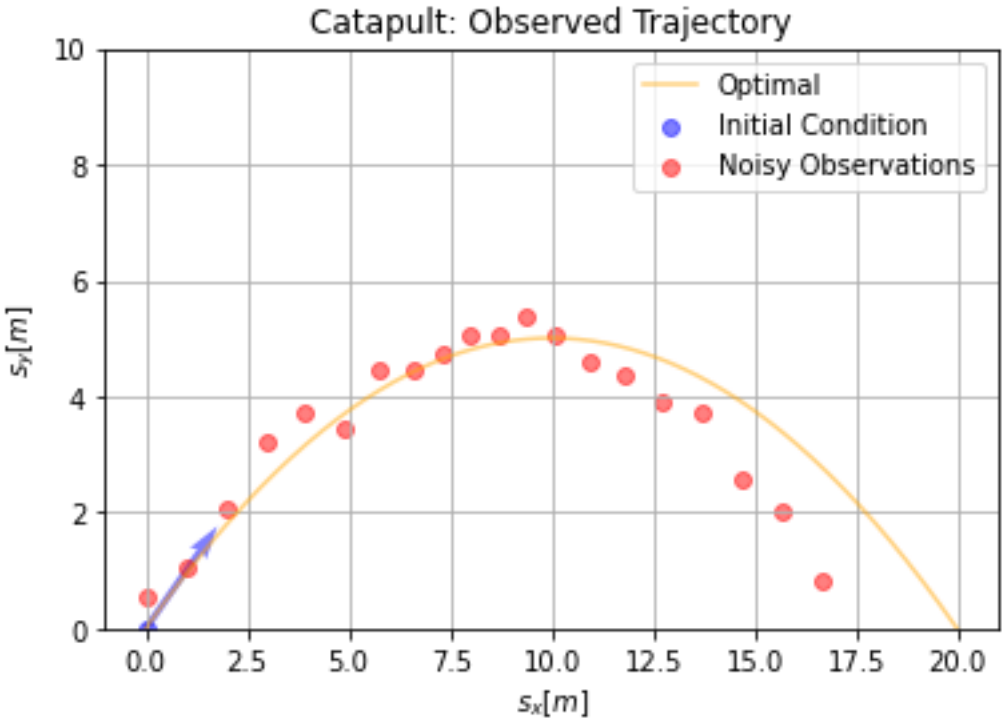
Hands-on Ex

Open-Loop Observer

Estimate state based on initial condition and dynamics.

$$x_{k+1} = A_k x_k + B_k u_k$$



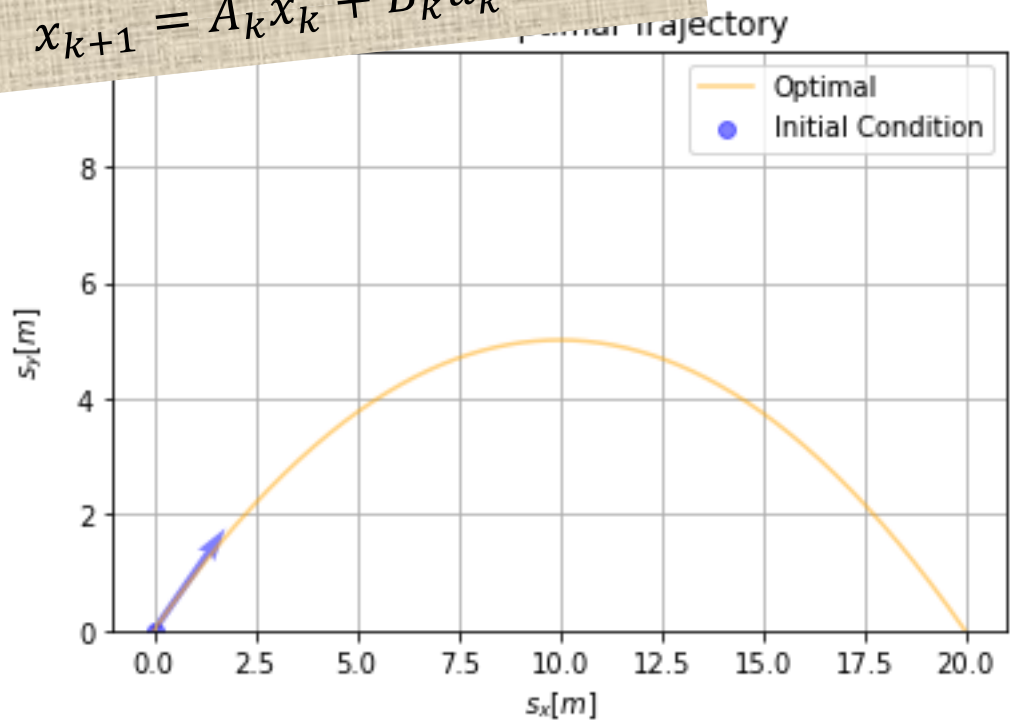


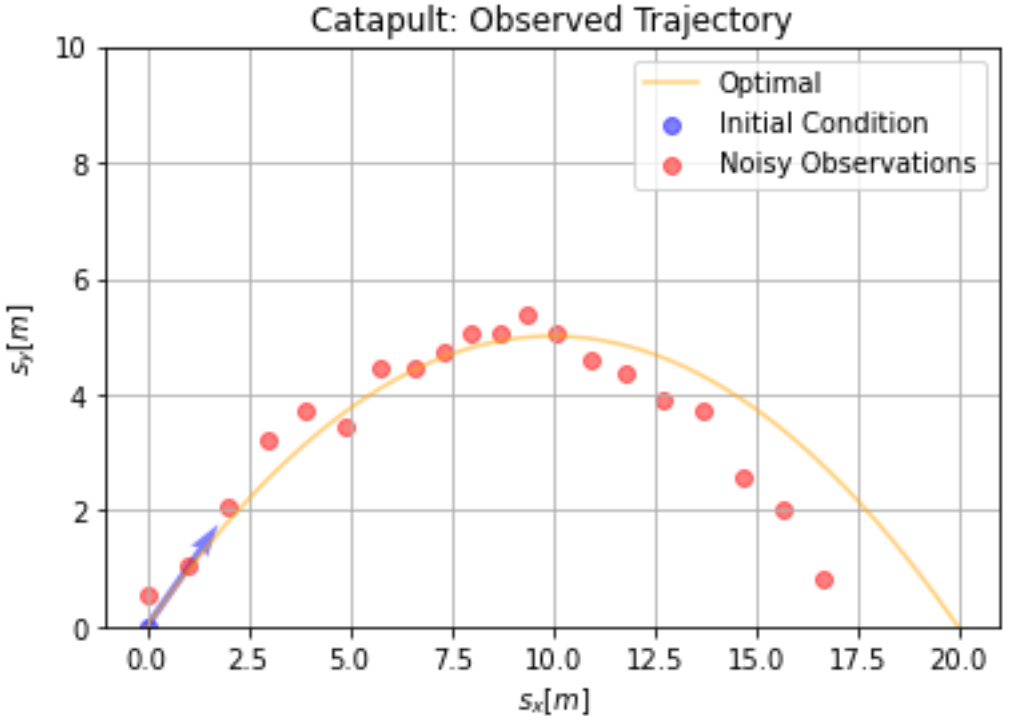
Hands-on Ex

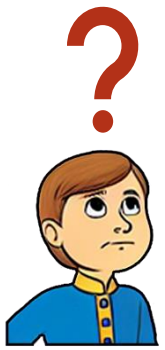
Open-Loop Observer

Estimate state based on initial condition and dynamics.

$$x_{k+1} = A_k x_k + B_k u_k$$





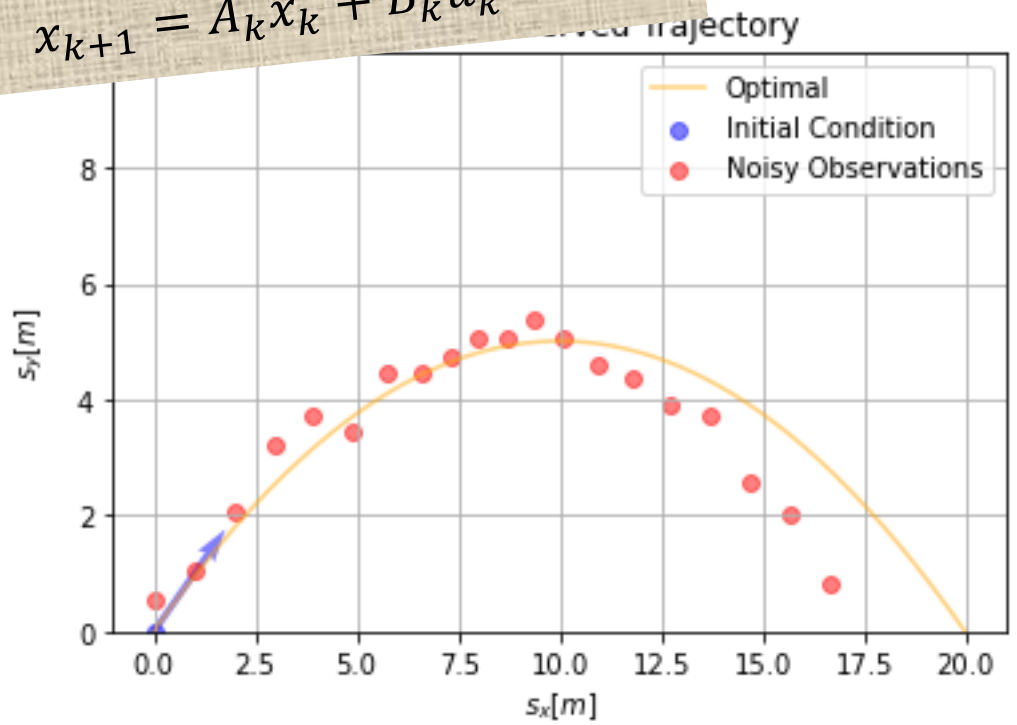


Hands-on Ex

Open-Loop Observer

Estimate state based on initial condition and dynamics.

$$x_{k+1} = A_k x_k + B_k u_k$$



Error of Open-Loop Observers

If a state of a linear, discrete-time system can be described with a state equation like:

$$x_{k+1} = A_k x_k + B_k u_k$$

The evolution of error is given by:

$$x_{k+1} - \tilde{x}_{k+1} = A_k^T (x_0 - \tilde{x}_0)$$

Stability of system matrix

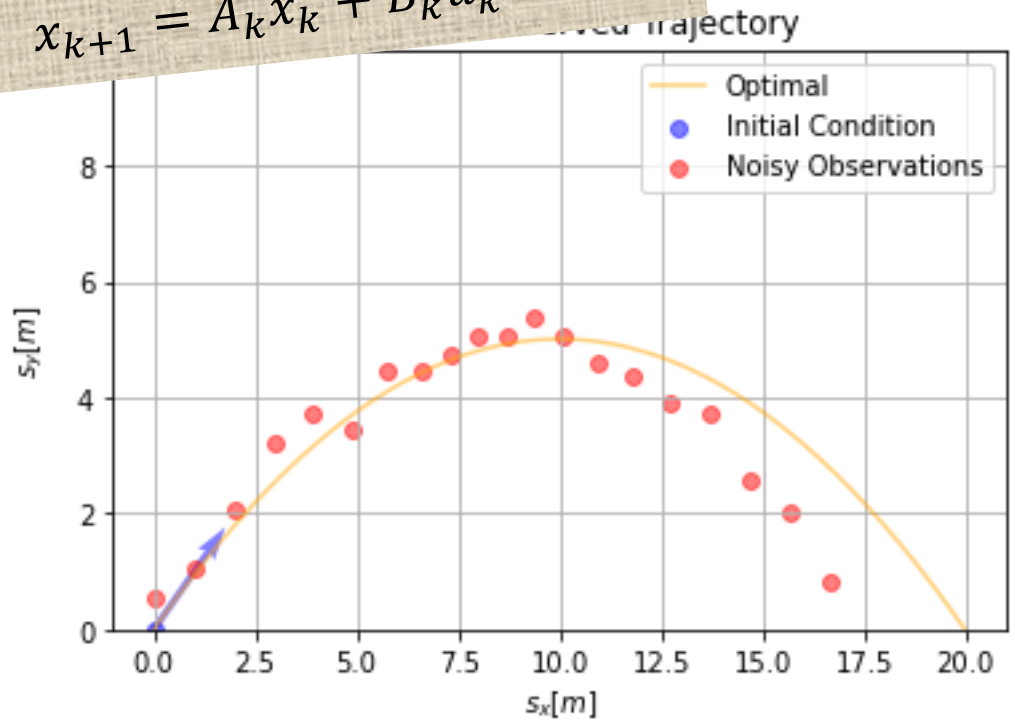
Error in initial estimation.

Hands-on Ex

Open-Loop Observer

Estimate state based on initial condition and dynamics.

$$x_{k+1} = A_k x_k + B_k u_k$$



Approximation of dynamics

$$x_{k+1} = A_k x_k, + B_k u_k + \omega_k$$

$$z_k = H_k x_k + \eta_k$$

Observations

Random disturbance

Kalman Filter

Model uncert

Open-Loop Observer

Estimate state based on initial condition and dynamics.

$$x_{k+1} = A_k x_k + B_k u_k$$

Random disturbance

$$x_{k+1} = A_k x_k + B_k u_k + \omega_k$$

Model uncertanties

Random disturbance

$$x_{k+1} = A_k x_k + B_k u_k + \omega_k$$

assuming now that

$$\omega_k \Rightarrow \mathcal{N}(0, Q_k)$$

$$x_k \Rightarrow \mathcal{N}(x_k, P_k)$$

Gaussian distribution

Model uncertanties

Random disturbance

$$x_{k+1} = A_k x_k + B_k u_k + \omega_k$$

assuming now that

$$\omega_k \Rightarrow \mathcal{N}(0, Q_k)$$

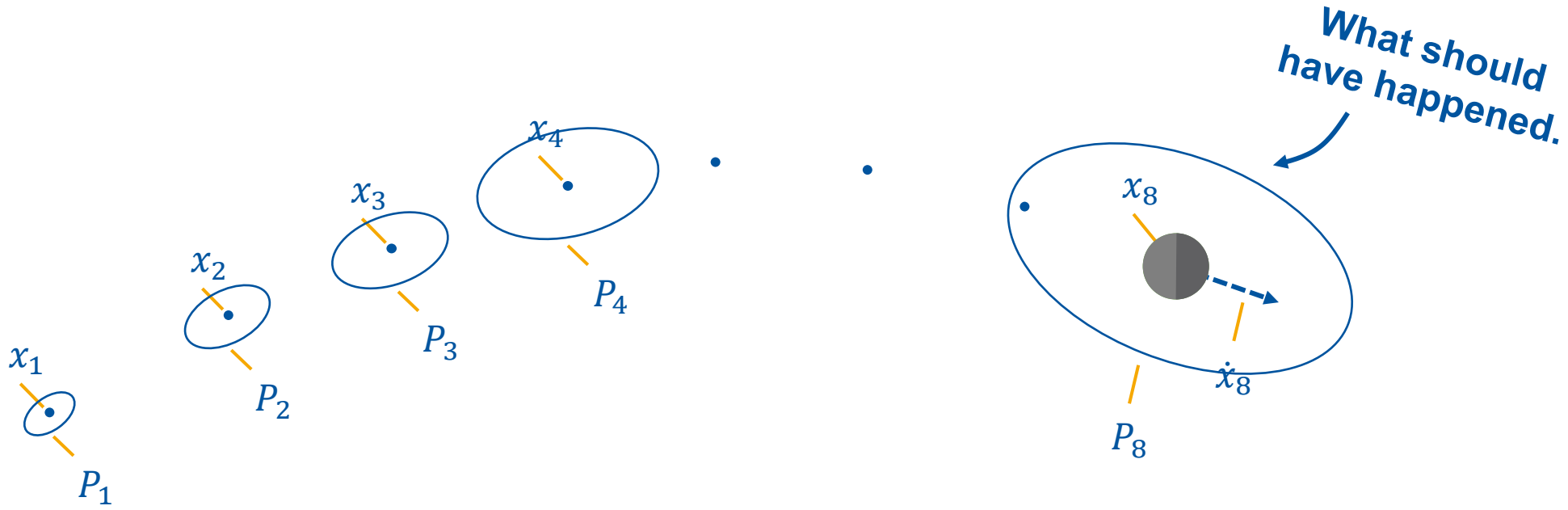
$$x_k \Rightarrow \mathcal{N}\ (x_k, P_k)$$

Gaussian distribution

$$\hat{x}_{k+1} = A_k x_k + B_k u_k$$
$$\hat{P}_{k+1} = A_k P_k A_k^T + Q_k$$

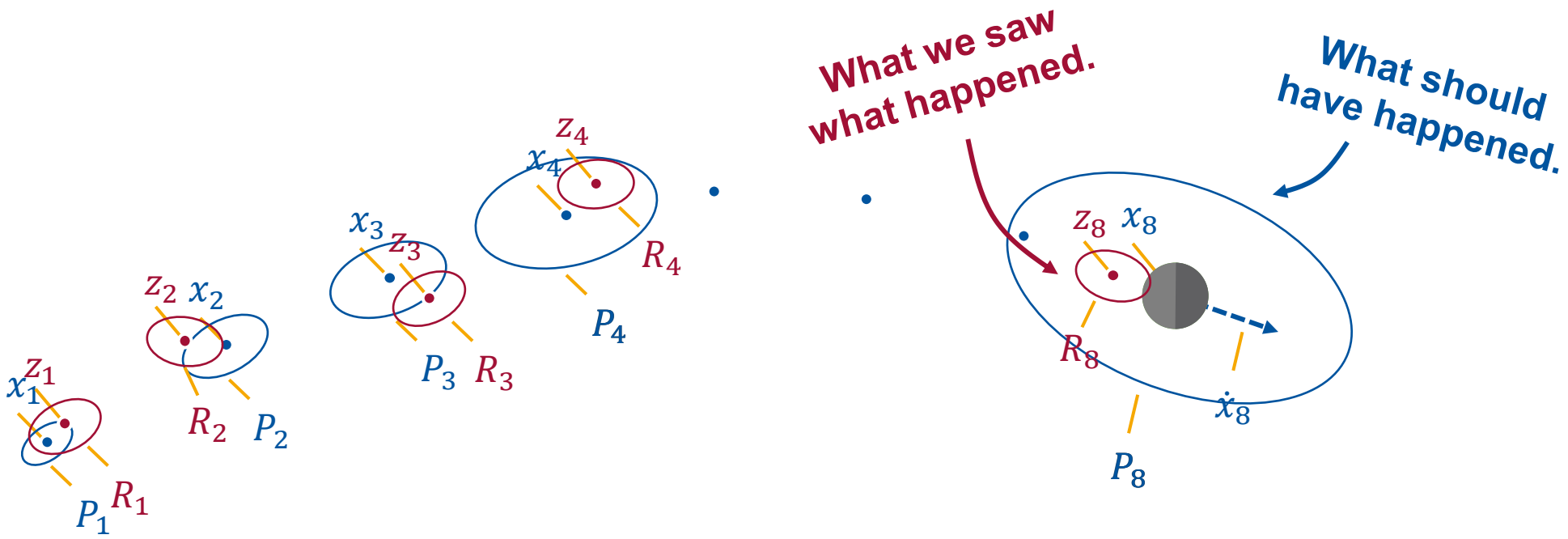
Model uncertanties

$$\hat{x}_{k+1} = A_k x_k + B_k u_k$$
$$\hat{P}_{k+1} = A_k P_k A_k^T + Q_k$$



Model uncertanties

$$\hat{x}_{k+1} = A_k x_k + B_k u_k$$
$$\hat{P}_{k+1} = A_k P_k A_k^T + Q_k$$



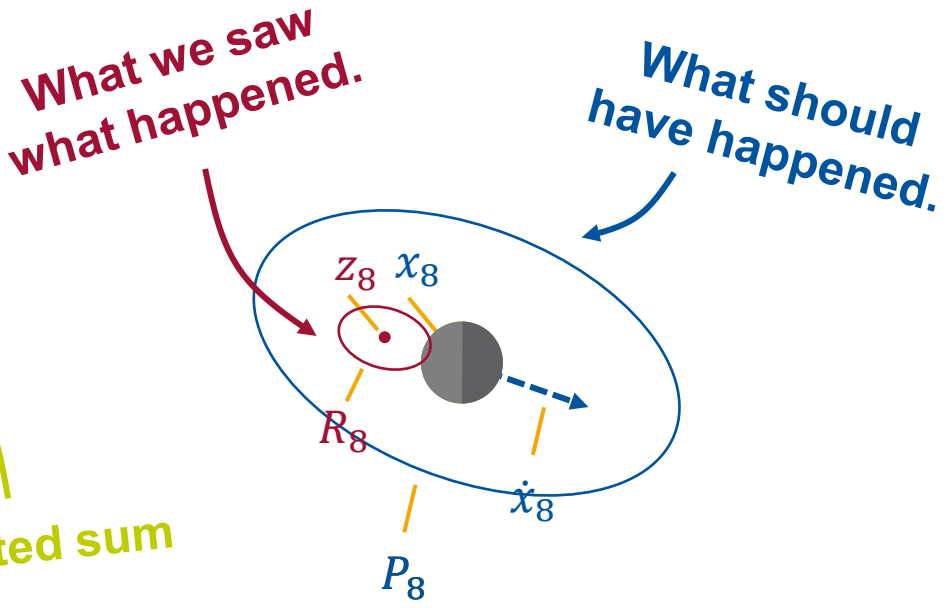
Model uncertanties

$$\hat{x}_{k+1} = A_k x_k + B_k u_k$$
$$\hat{P}_{k+1} = A_k P_k A_k^T + Q_k$$

Weight based on covariances

$$K = \hat{P}_{k+1} H^T \left(H \hat{P}_{k+1} H^T + R_k \right)^{-1}$$
$$x_{k+1} = \hat{x}_{k+1} + K(z_k - H\hat{x}_{k+1})$$
$$P_{k+1} = (I - KH)\hat{P}_{k+1}$$

Weighted sum



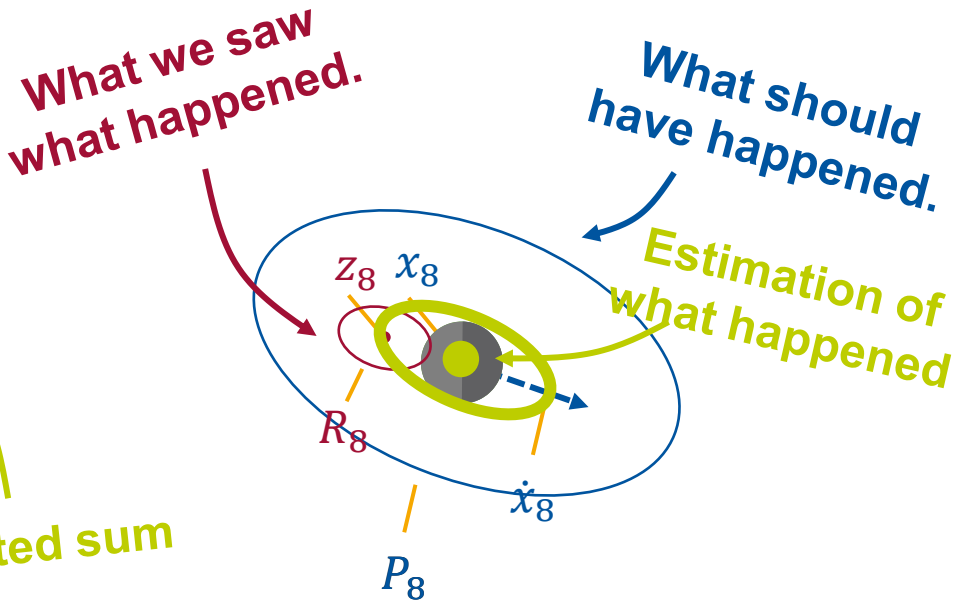
Model uncertanties

$$\hat{x}_{k+1} = A_k x_k + B_k u_k$$
$$\hat{P}_{k+1} = A_k P_k A_k^T + Q_k$$

Weight based on covariances

$$K = \hat{P}_{k+1} H^T \left(H \hat{P}_{k+1} H^T + R_k \right)^{-1}$$
$$x_{k+1} = \hat{x}_{k+1} + K(z_k - H\hat{x}_{k+1})$$
$$P_{k+1} = (I - KH)\hat{P}_{k+1}$$

Weighted sum



Kalman Filter

- State Estimation
- Parameter Estimation
- Prediction
- Sensor Data Fusion
- Estimation of non-measurable Quantities

Kalman Filter

A Kalman filter is an “optimal” recursive estimation algorithm used to estimate states of a system from indirect and uncertain measurements.

Predict

$$\hat{x}_{k+1} = A_k x_k + B_k u_k$$
$$\hat{P}_{k+1} = A_k P_k A_k^T + Q_k$$

What sould have happened.

Update

$$K = \hat{P}_{k+1} H^T \left(H \hat{P}_{k+1} H^T + R_k \right)^{-1}$$
$$x_{k+1} = \hat{x}_{k+1} + K(z_k - H\hat{x}_{k+1})$$
$$P_{k+1} = (I - KH)\hat{P}_{k+1}$$

What we saw what happened.

Kalman Filter

- **State Estimation**
- Parameter Estimation
- **Prediction**
- Sensor Data Fusion
- **Estimation of non-measurable Quantities**

Kalman Filter

A Kalman filter is an “optimal” recursive estimation algorithm used to estimate states of a system from indirect and uncertain measurements.

Predict

$$\hat{x}_{k+1} = A_k x_k + B_k u_k$$
$$\hat{P}_{k+1} = A_k P_k A_k^T + Q_k$$

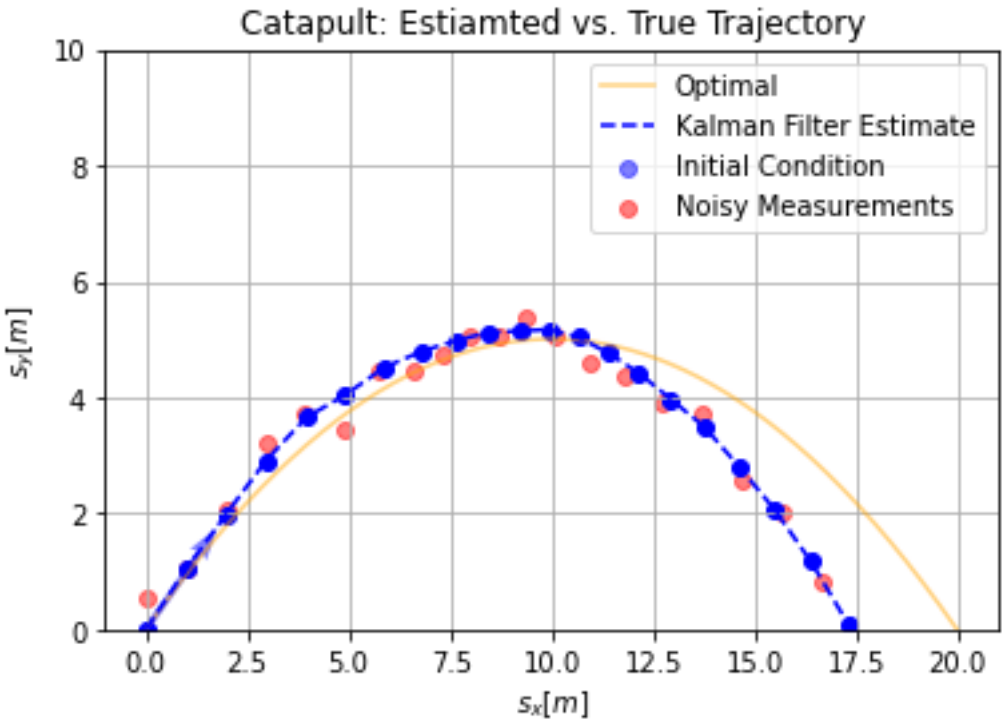
What sould have happened.

Update

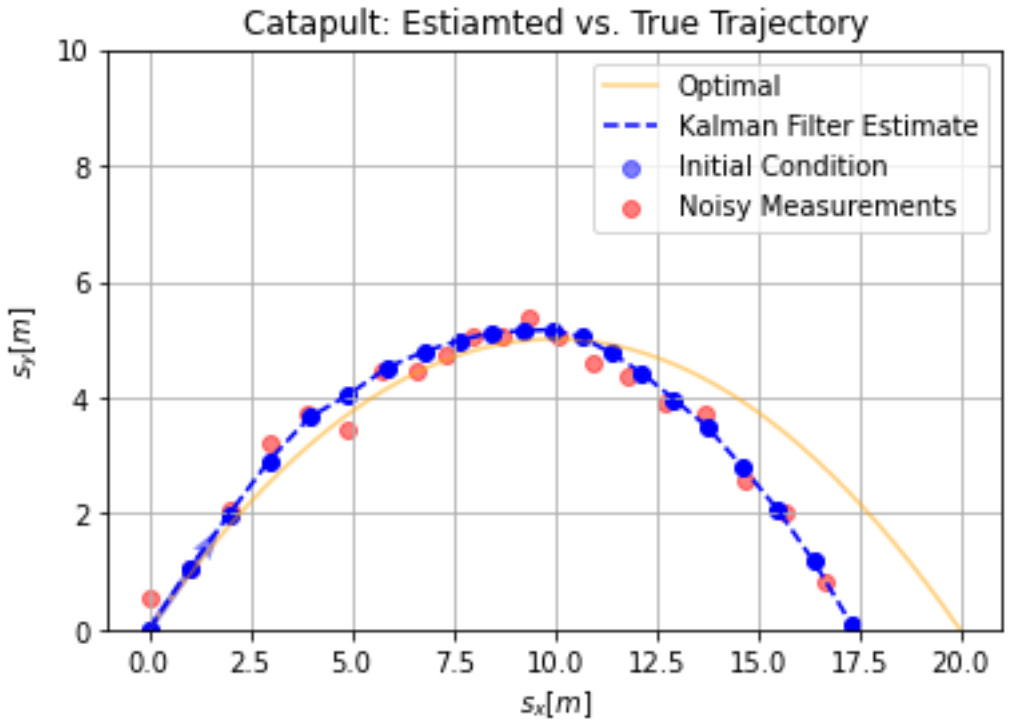
$$K = \hat{P}_{k+1} H^T \left(H \hat{P}_{k+1} H^T + R_k \right)^{-1}$$
$$x_{k+1} = \hat{x}_{k+1} + K(z_k - H\hat{x}_{k+1})$$
$$P_{k+1} = (I - KH)\hat{P}_{k+1}$$

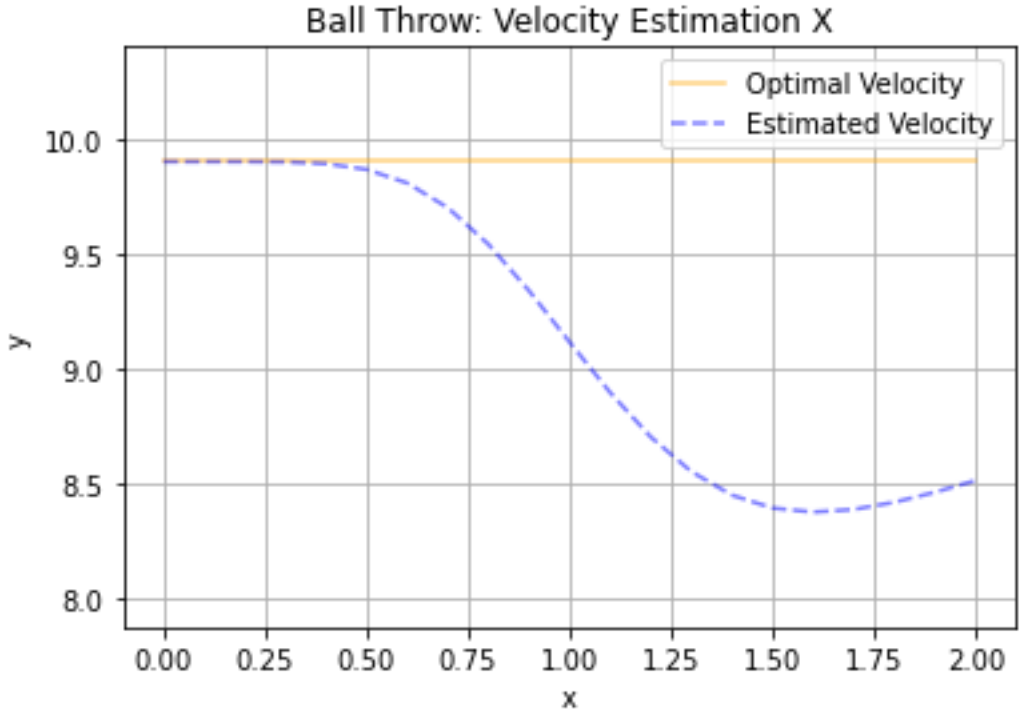
What we saw what happened.

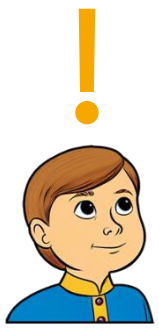
Hands-on Example



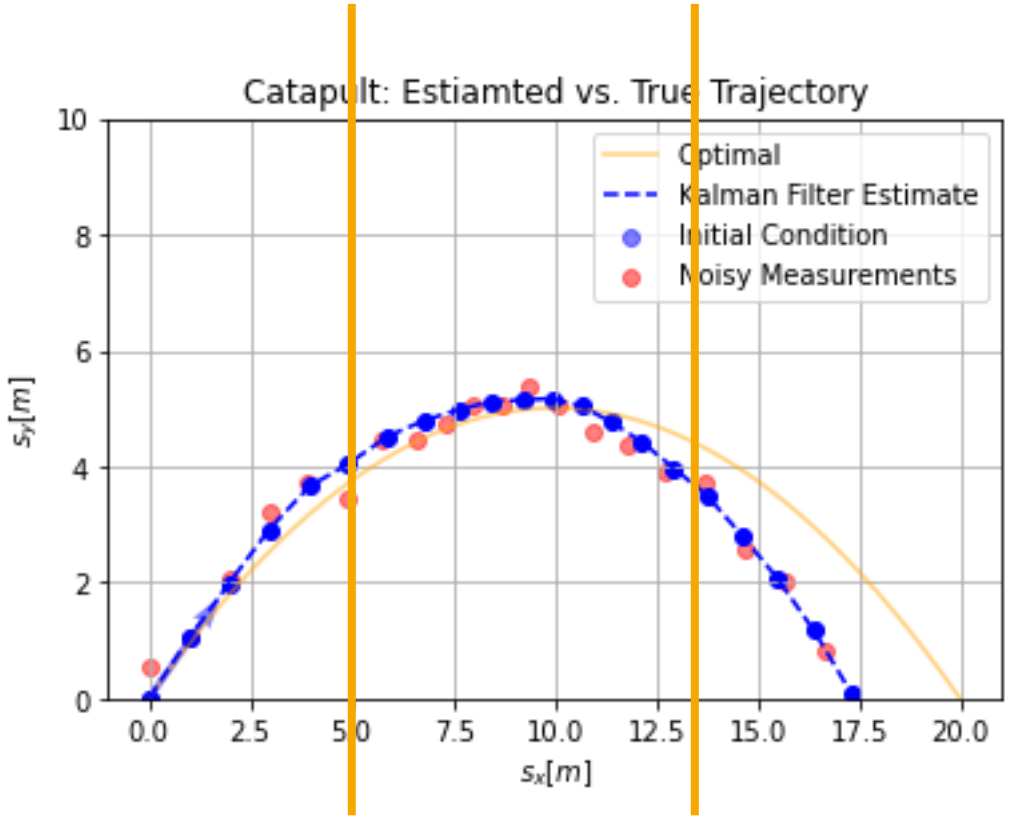
Hands-on Example

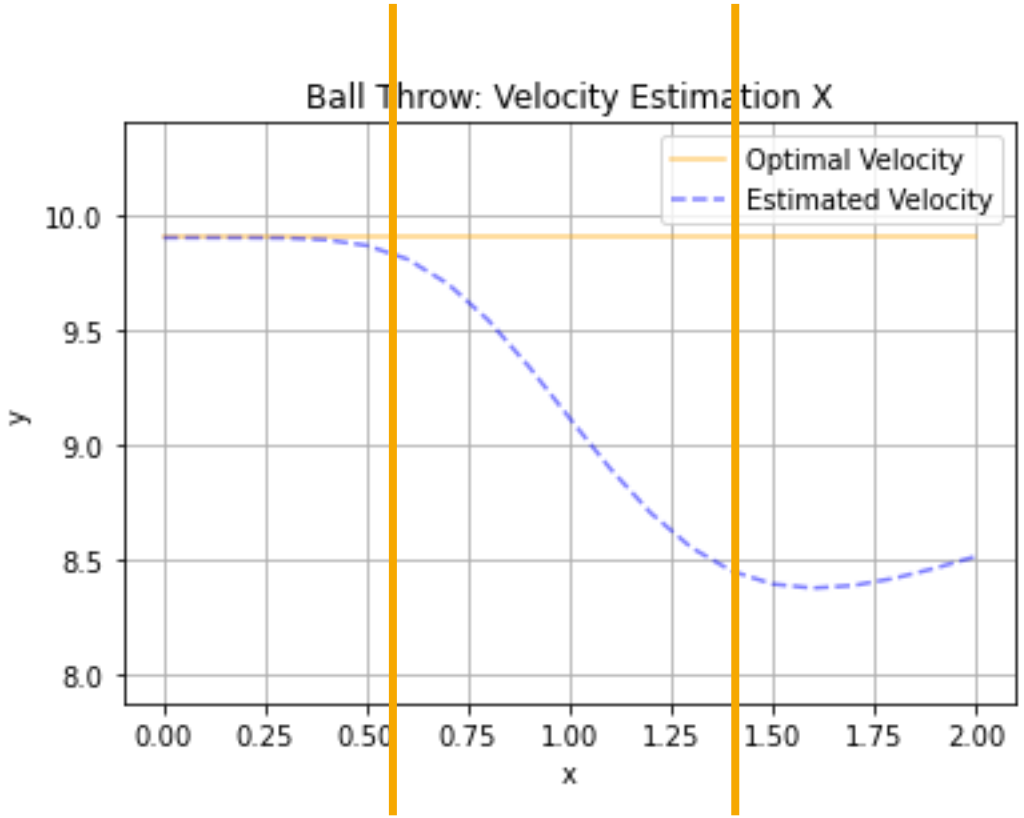


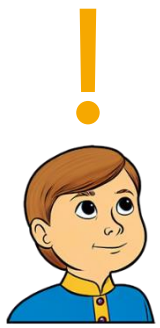




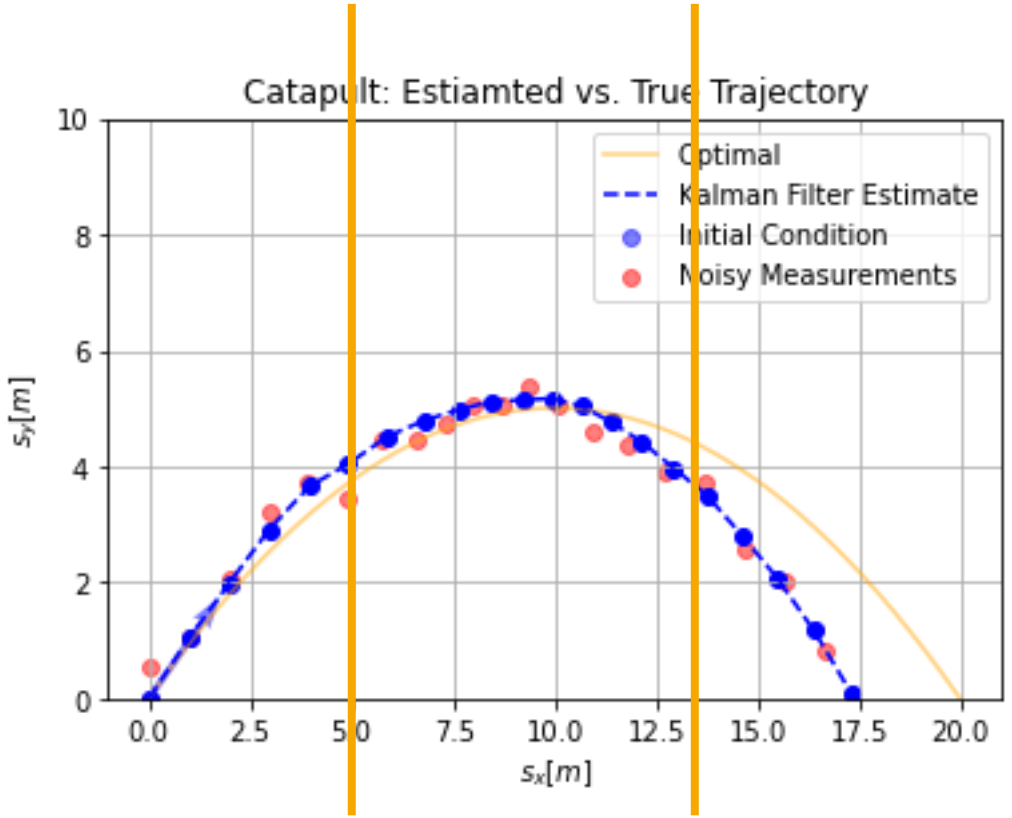
Hands-on Example

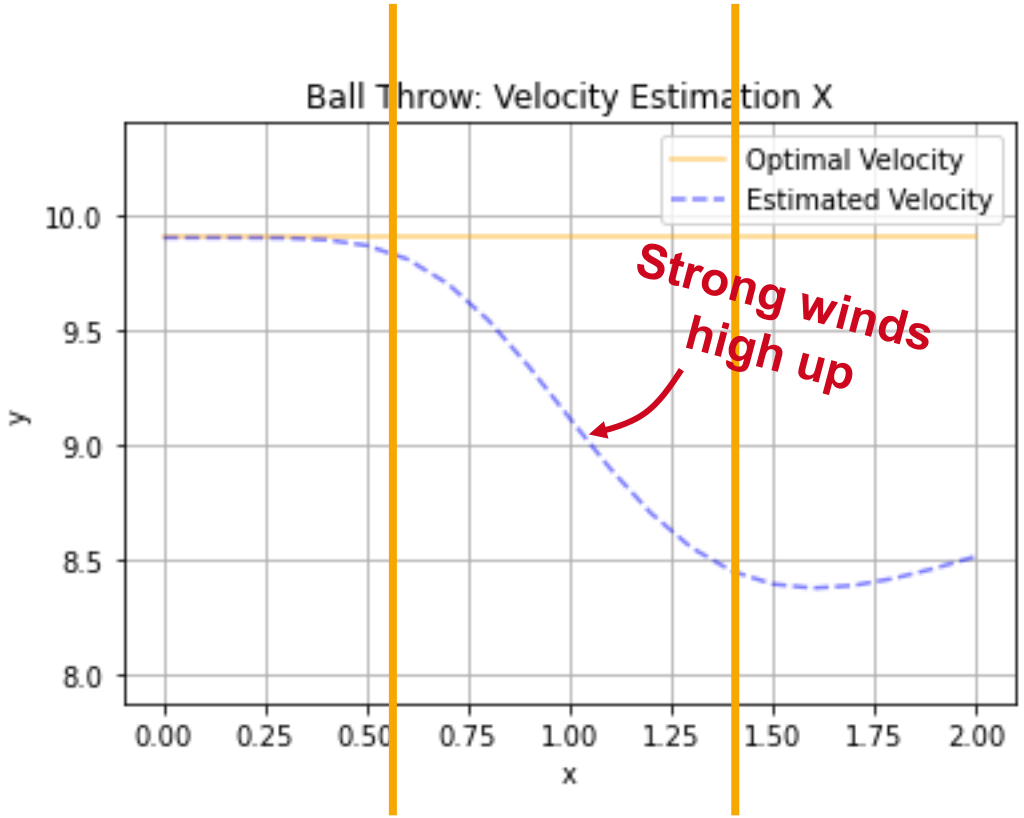


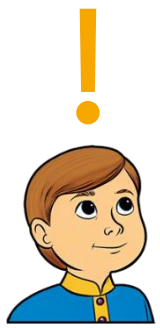




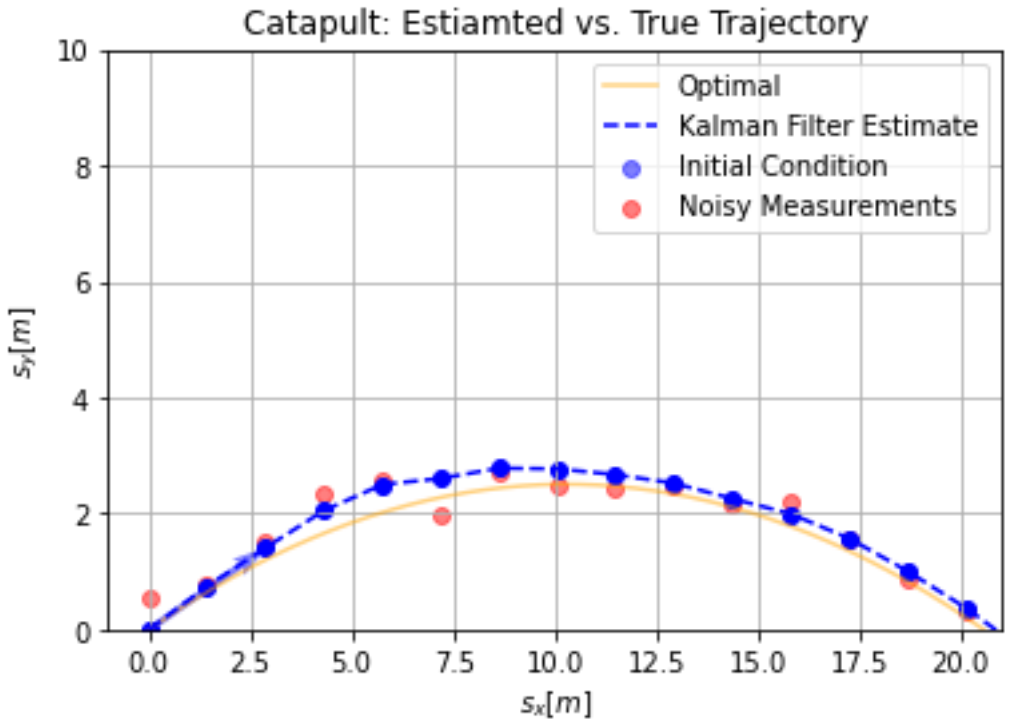
Hands-on Example

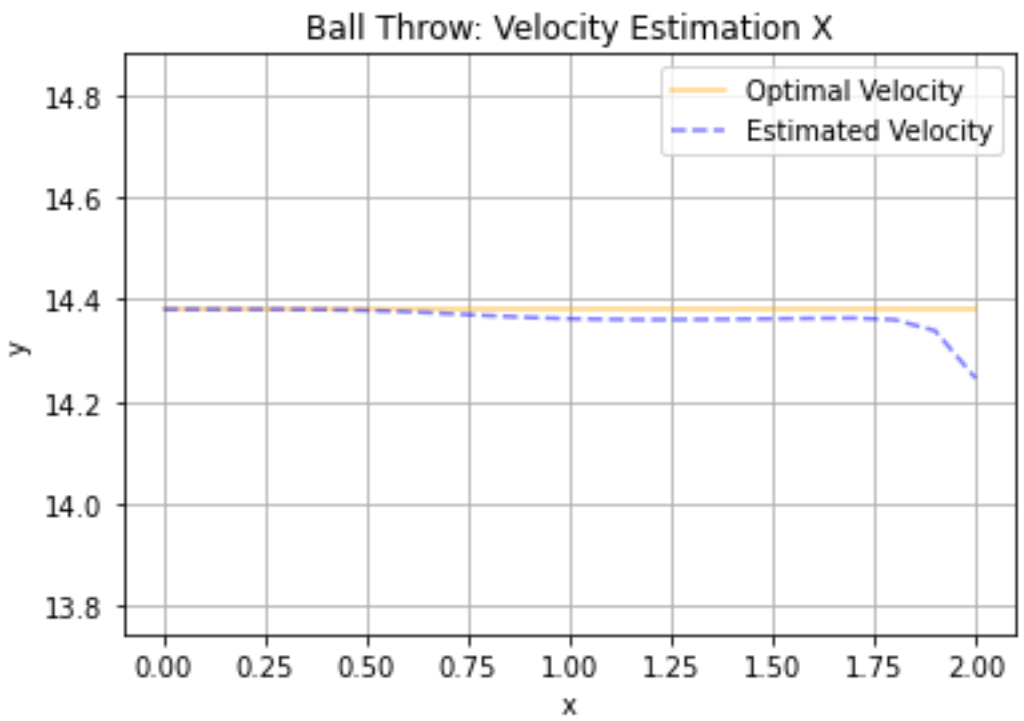


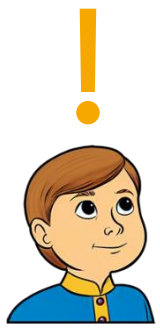




Hands-on Example

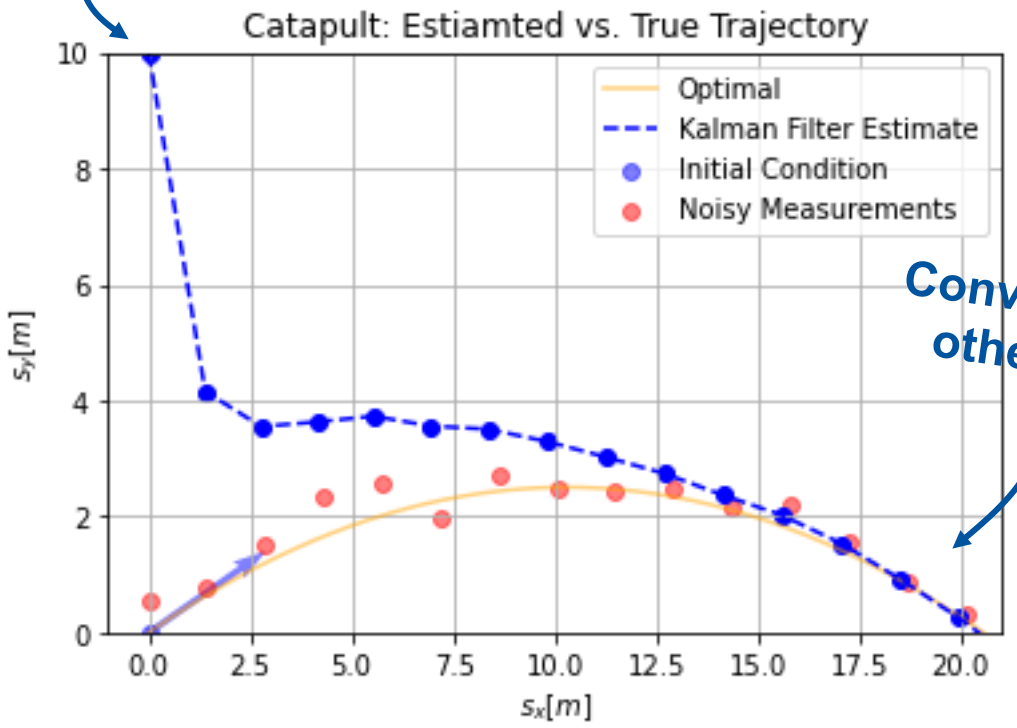




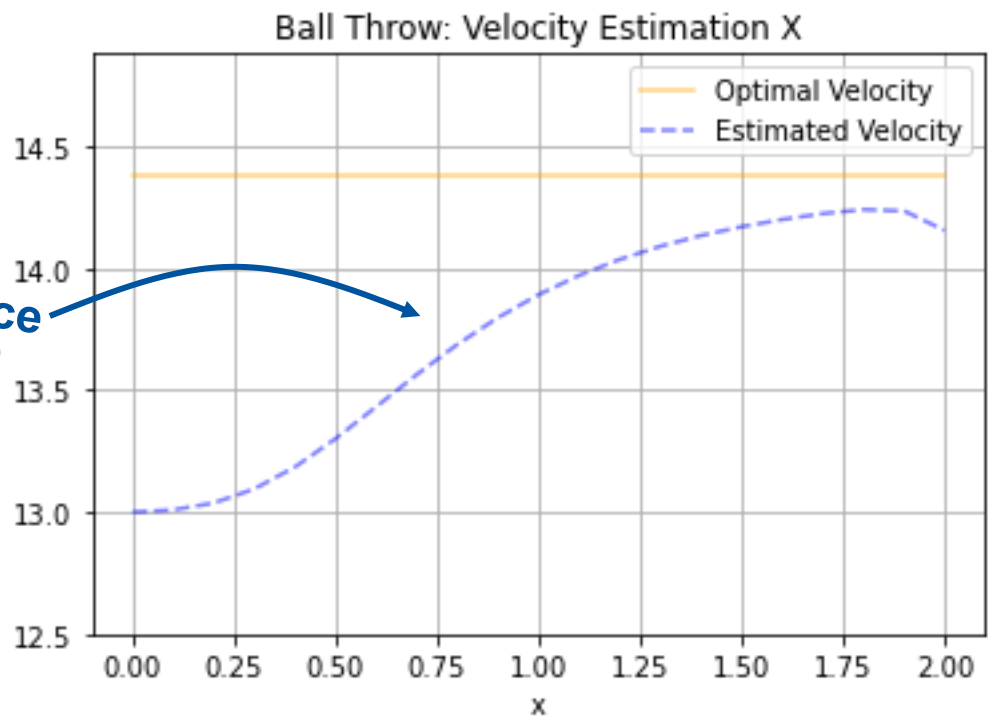


Hands-on Example

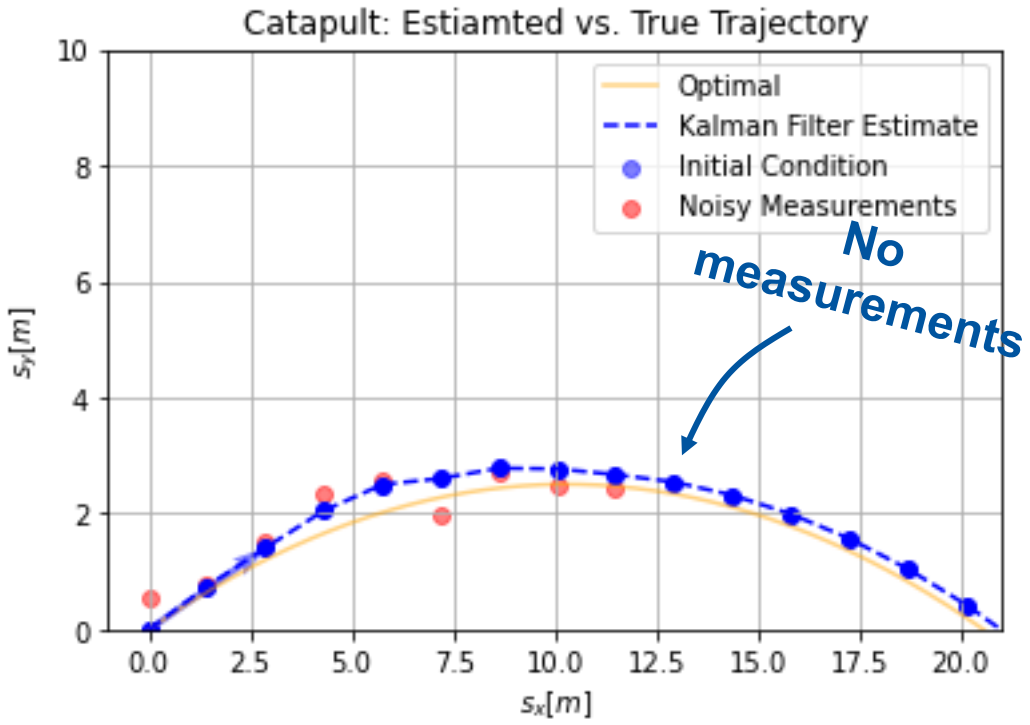
Wrong initial value

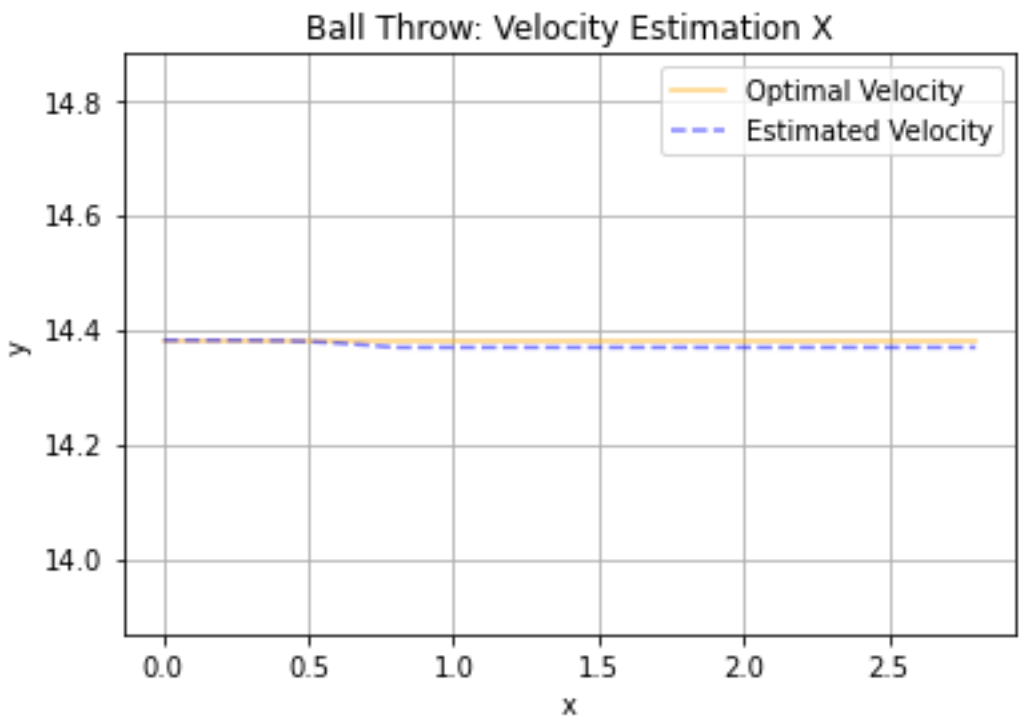


Convergence other time



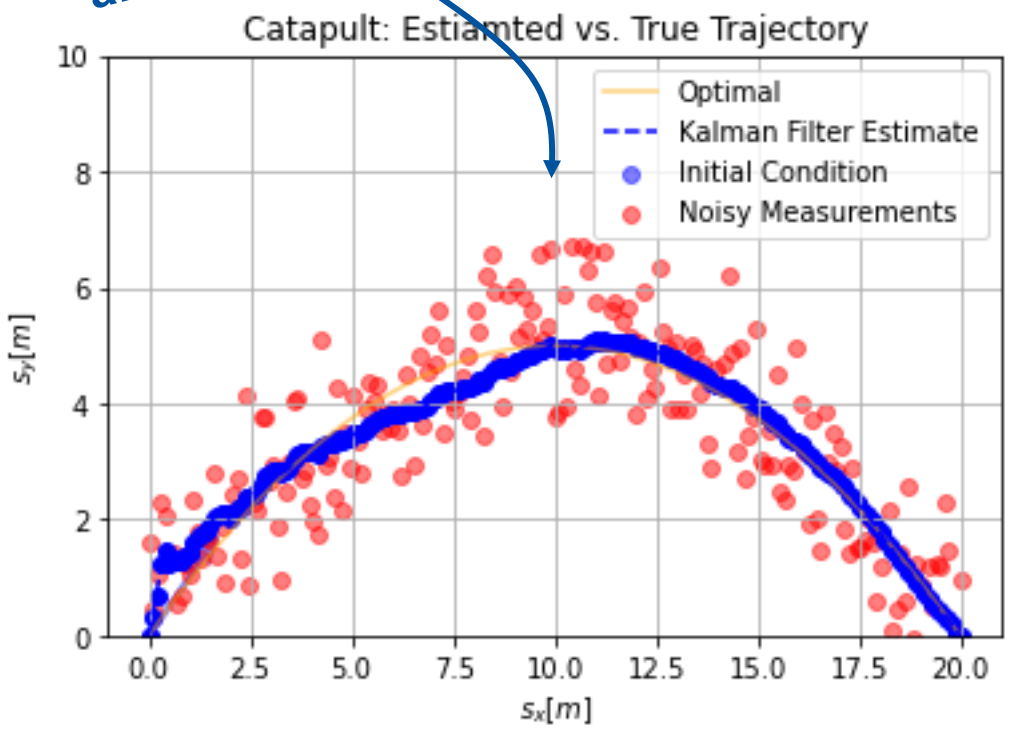
Hands-on Example

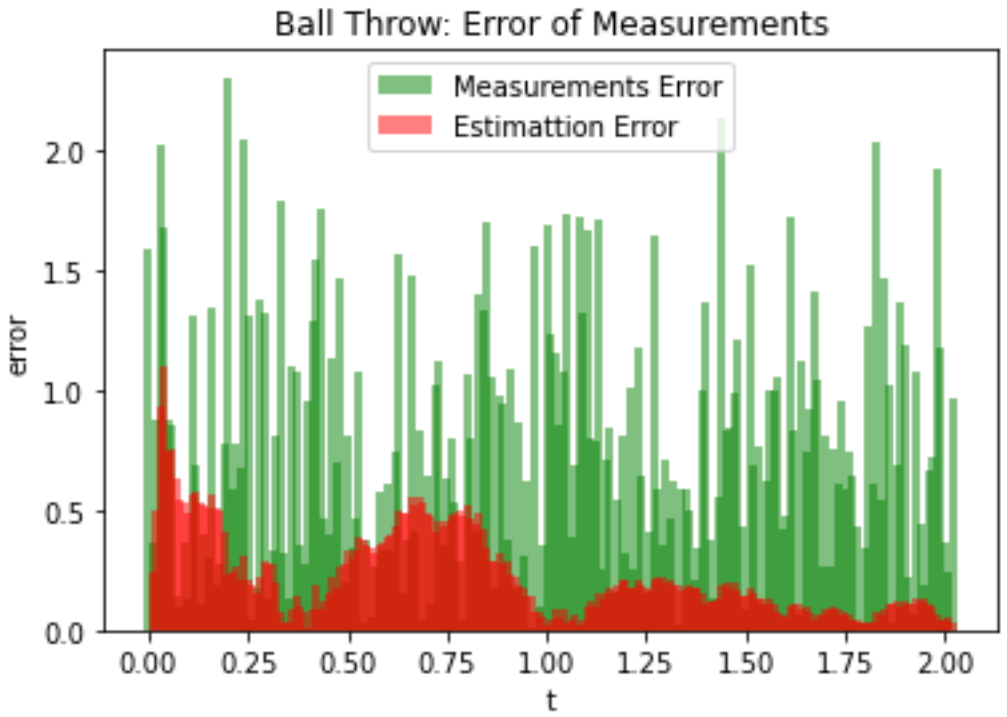




Hands-on Example

High frequency and heavy noise

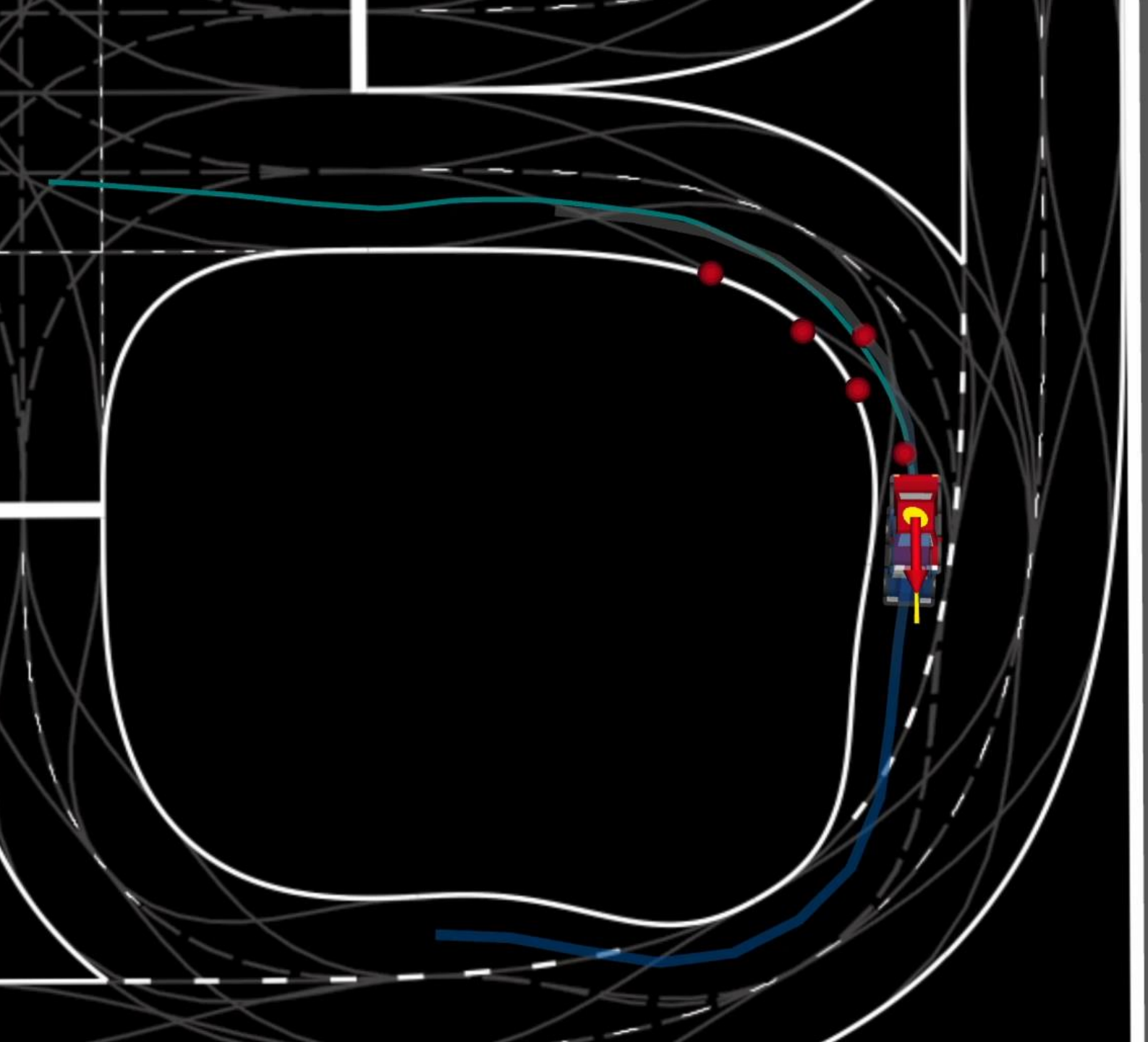




Applications in our research

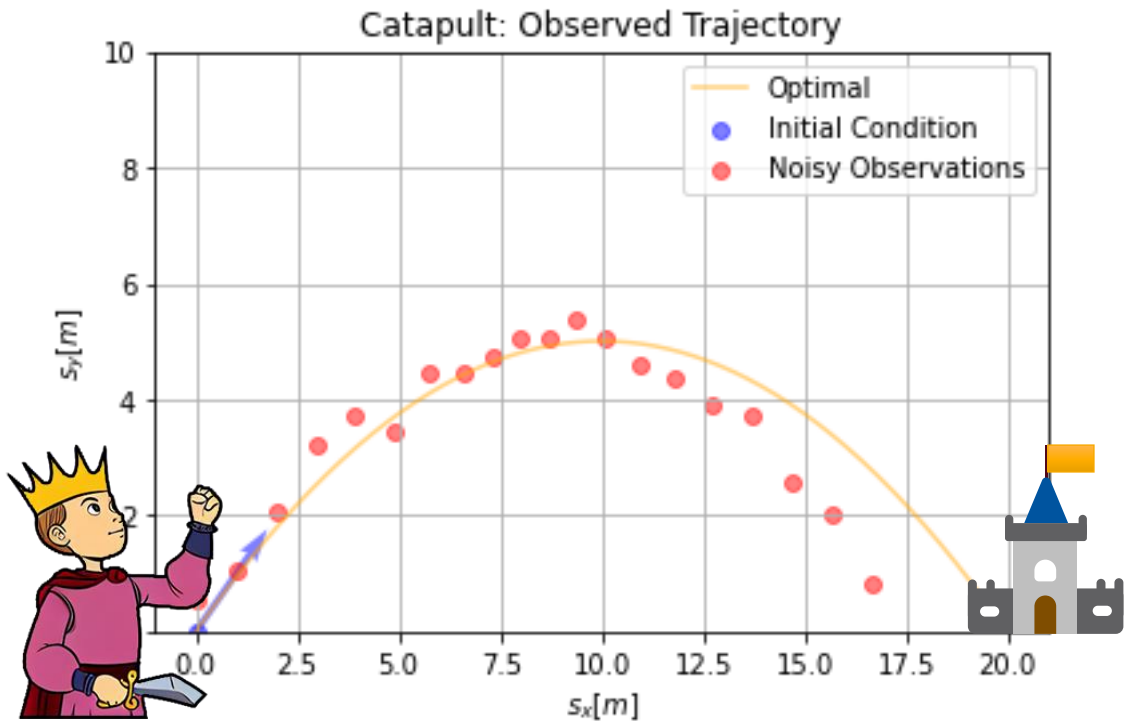
Informatik 11
Embedded Software

RWTH AACHEN UNIVERSITY



Homework

- Aid Karl the Little in attacking his vicious opponent
- **Jupyter notebook on Moodle**
- Exercise to understand Kalman Filters and their parameters
- No Python code required in the exam



Next Part

Software architectures and testing concepts

- With Patrick Scheffe