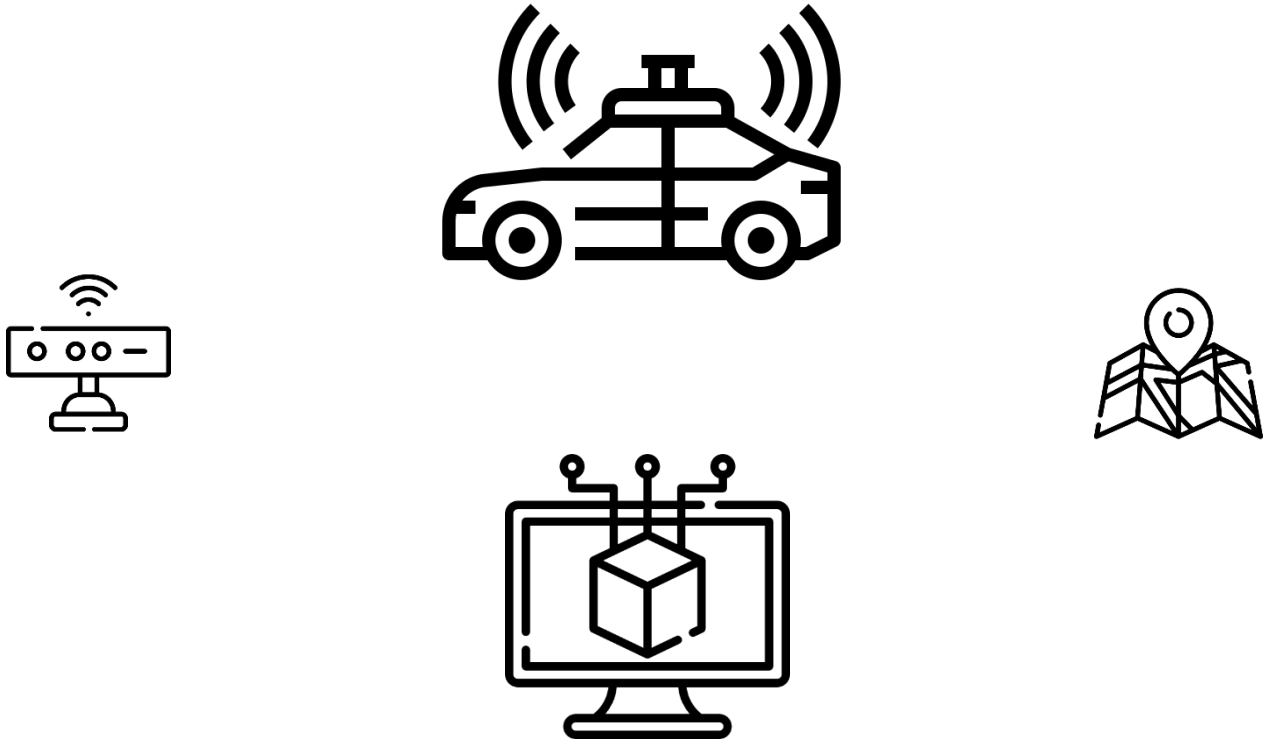
Institute of Automotive Technology
Faculty for Mechanical Engineering
Technical University of Munich

Autonomous Driving Software Engineering

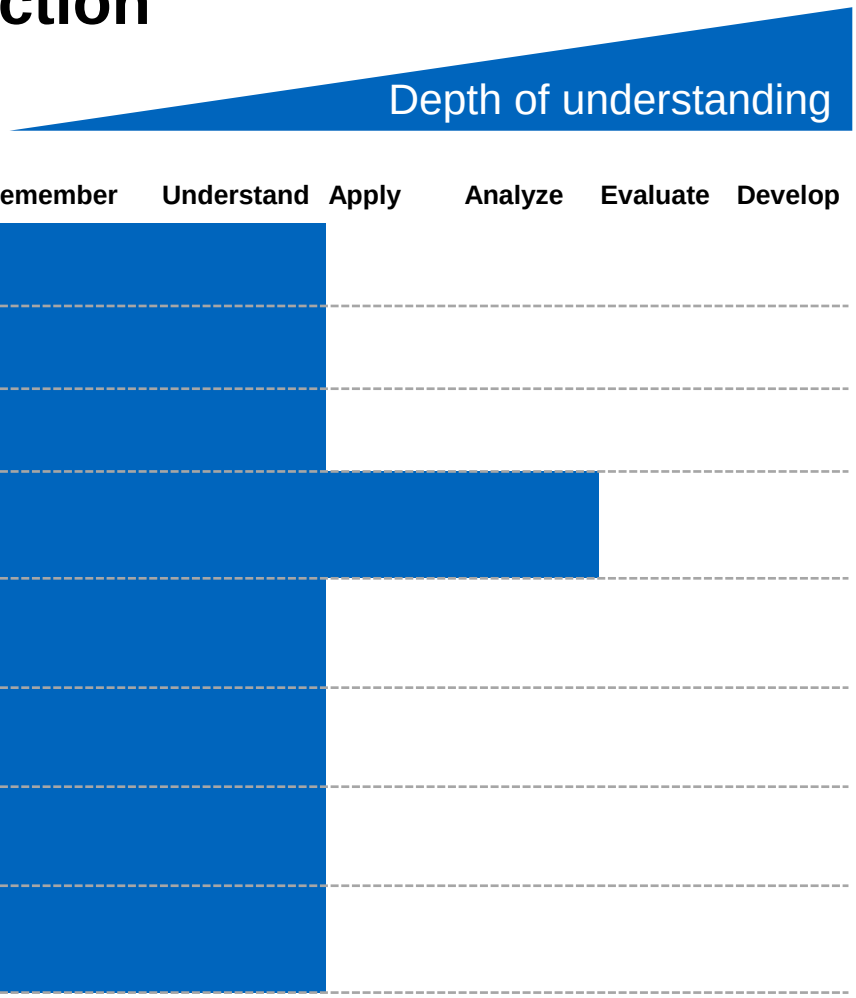
Prof. Dr.-Ing. Markus Lienkamp

Phillip Karle, M. Sc.



Objectives for Lecture 4: Detection

After the lecture you are able to…



… explain the tasks of the detection module and the challenges faced by the detection module

… identify the differences between image classification, object localization, and object detection

… classify the camera detection pipelines (classical CV pipeline vs. deep learning)

... explain the background of the four groups of deep learning camera 3D object detection (mono3DOD) and analyze algorithms belonging to those groups

… outline the challenges of LiDAR detection and the two approaches to solve them

… illustrate the basics of Radar detection

… discuss the necessity and concepts of sensor fusion (early/late fusion)

… explain the challenges and importance of high-quality datasets, the differences between real-world and synthetic datasets, and ways to extent datasets (fleet sourcing)

Introduction

Source: https://www.pexels.com/video/person-driving-in-the-city-4483542/

Additional Slides

The scene in the video shows a (to us humans) relatively easy driving situation in an urban environment. However, the car has to detect all objects, such as cars, cyclists, pedestrians, lanes and traffic lights/signs in all shapes/variations/colors/light situations with a few sensors and computational power. This lecture aims to give a brief introduction on the algorithms that are used to process the sensor data and detect all relevant objects and obstacles.

Detection
Prof. Dr. Markus Lienkamp

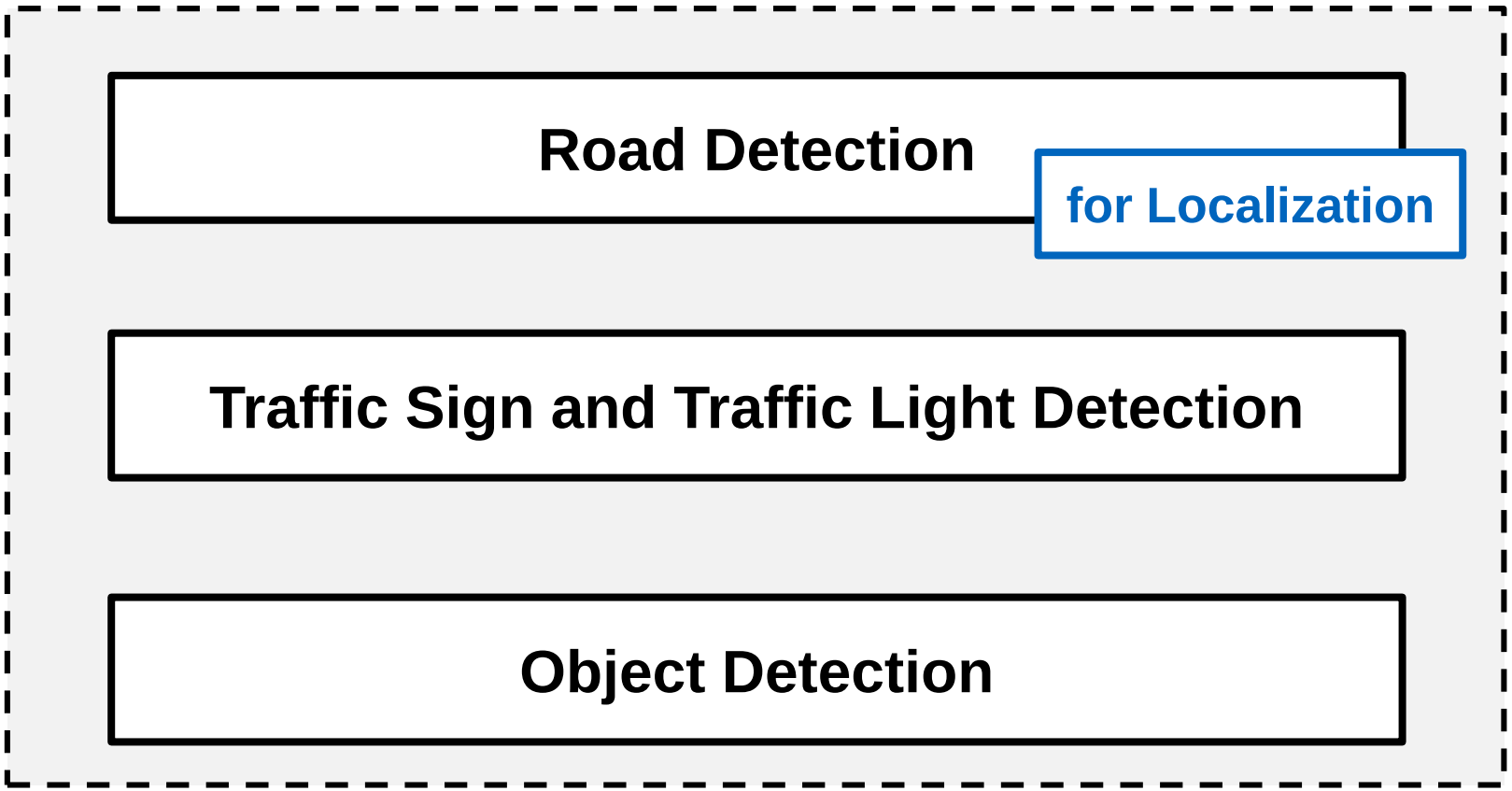
Sebastian Huch, M. Sc.

Agenda

1. **General Definitions**
2. Camera Detection
3. LiDAR Detection
4. Radar Detection
5. Sensor Fusion
6. Datasets
7. Summary

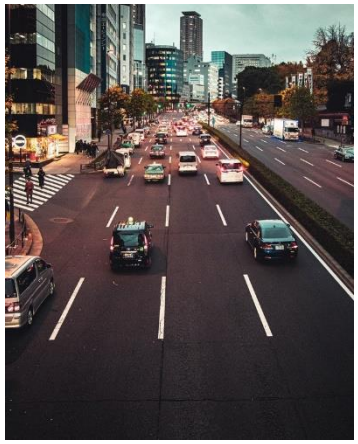
General Definitions

3 Detection Tasks



General Definitions
Detection Tasks

What do we need to detect?



[2][3]

Lanes and different lane markings

Road Detection

General Definitions
Detection Tasks

What do we need to detect?

Traffic lights

Traffic signs – different shapes/countries

[4][5][6][7]

Traffic Sign and Traffic Light Detection

General Definitions
Detection Tasks

What do we need to detect?



Cars – different shapes/sizes/colors, static/dynamic



Bicycles/Motorcycles

Pedestrians

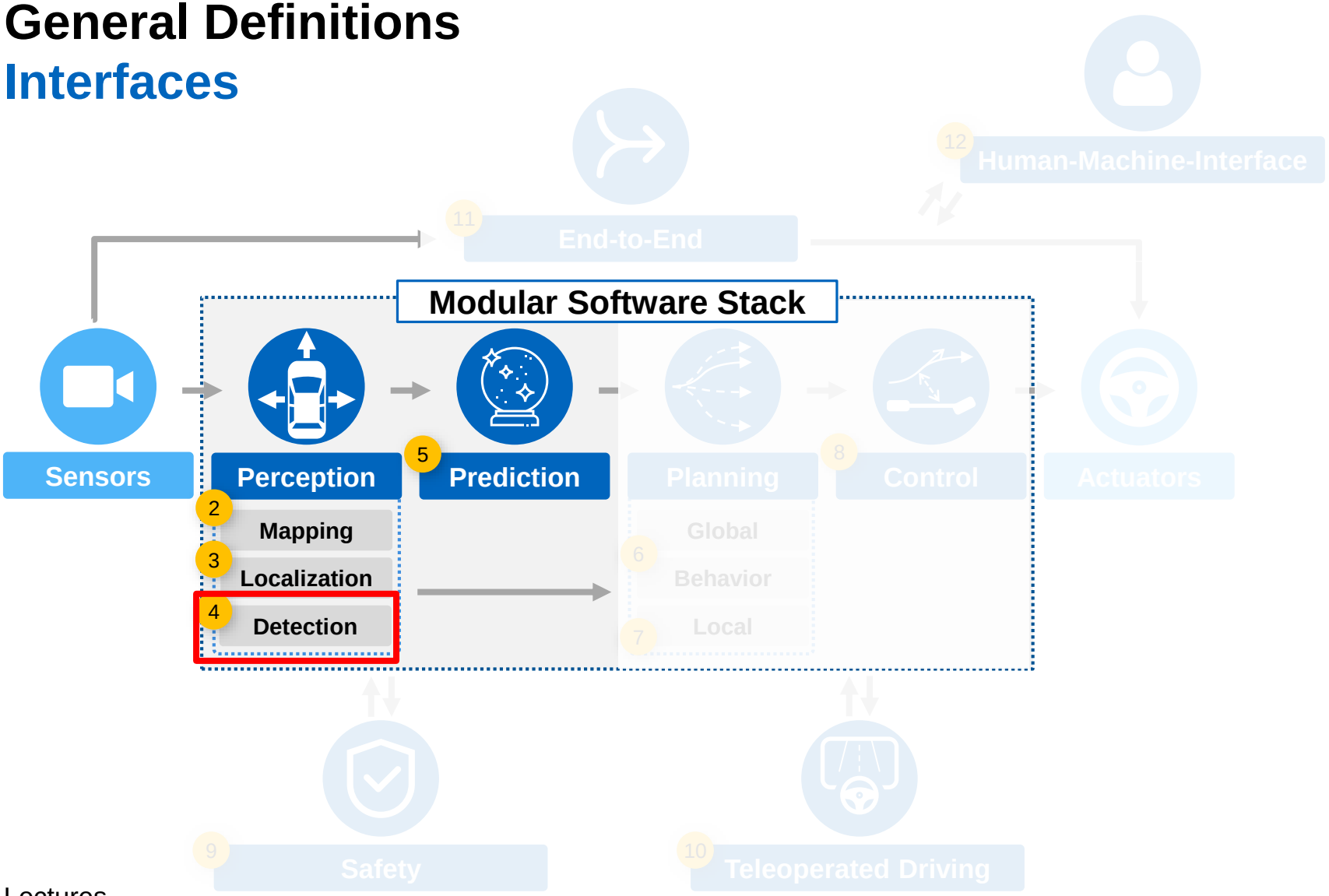
Trucks

Road users

[8][9][10][11][12][13]

Object Detection

General Definitions
Interfaces



X = Lectures

Additional Slides

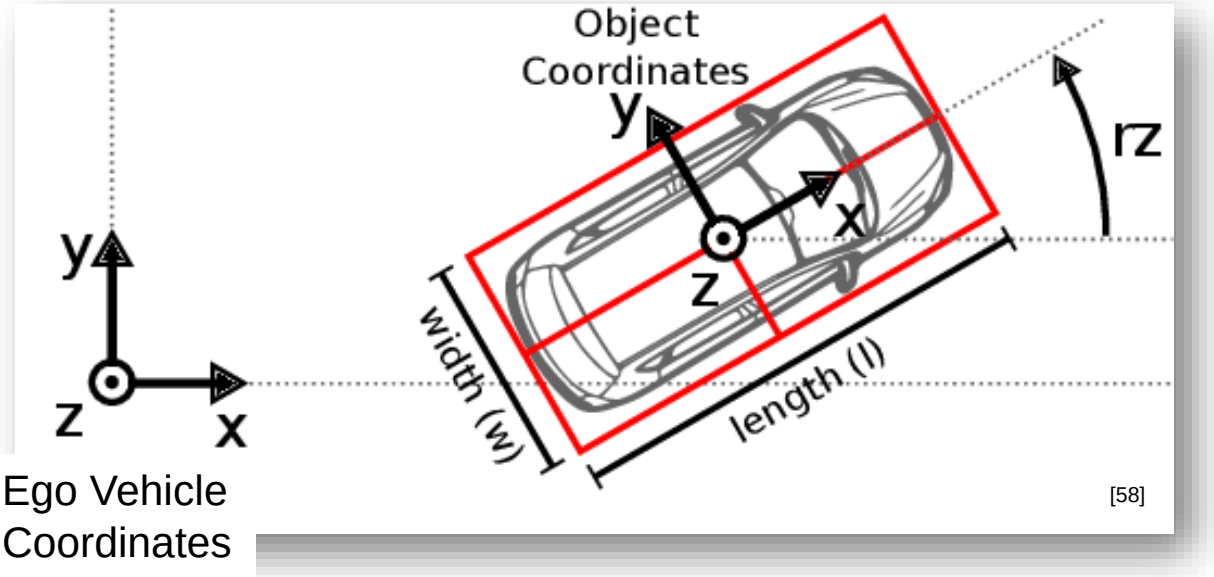
Together with localization and mapping, the detection module is part of the perception in the autonomous driving stack. The interfaces of the detection are with the sensors on the input and to the following modules (prediction and planning) on the output.

General Definitions
Interfaces

Sensor data from multiple sensors

Object Detection

Representation of all **objects and obstacles** in the ego vehicle's proximity that are **relevant for the driving task**



Additional Slides

Output of object detection:

For every detected object, an array of the object‘s charakteristics is calculated/measured.

This array can contain the relative position (x,y,z), the yaw angle, the measured speed v, the dimensions (height h, width w, length l) and the type (e.g. Car, Pedestrian, Cyclist, etc.).

$$\text{Object\_1} = \begin{bmatrix} x \\ y \\ z \\ yaw \\ v \\ h \\ w \\ l \\ type \end{bmatrix}$$

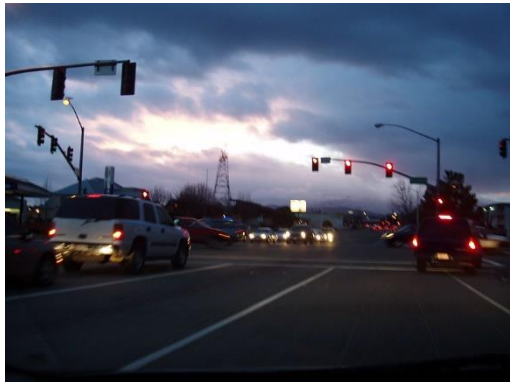
General Definitions

Challenges for Detection

- Weather (rain, snow, fog, …)
- Daytime (night, sunset, …)
- Occlusion
- Compute time (at higher driving speeds)
- Different environments

Snow on road and cars, occlusion [14][15][16][17]

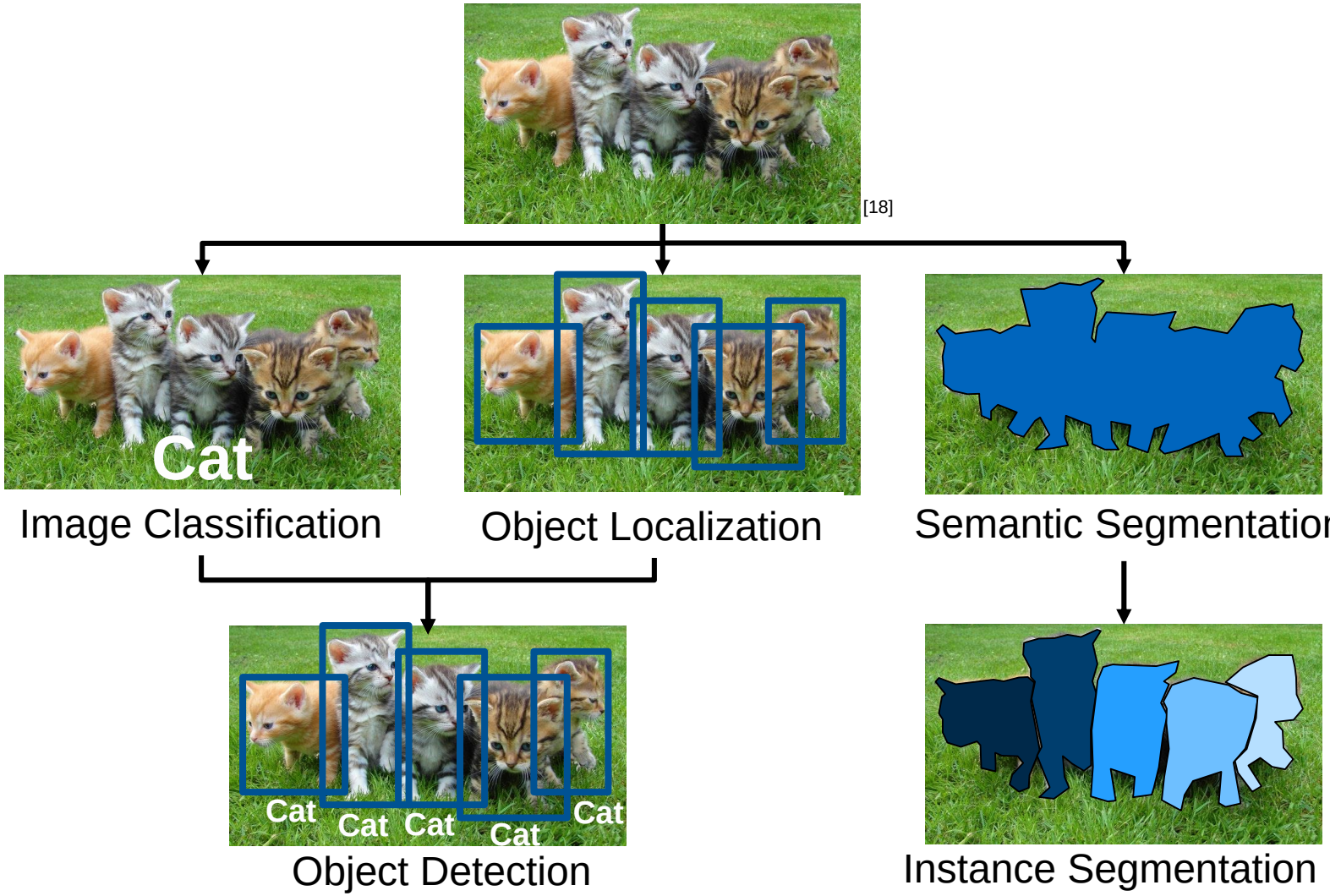
Rain, reflections, wet road



Driving at night

Low sun, high brightness, glare

General Definitions – Classification/Localization/Detection



[18]

Image Classification

Object Localization

Semantic Segmentation

Object Detection

Instance Segmentation

Sources

[1] https://www.pexels.com/photo/aerial-photography-of-cars-on-road-intersection-186537/
[2] https://www.pexels.com/photo/busy-city-road-in-evening-3444690/
[3] https://www.pexels.com/photo/gray-asphalt-road-under-blue-sky-1021683/
[4] https://www.pexels.com/photo/traffic-light-1766849
[5] https://www.pexels.com/photo/stop-sign-1806900/
[6] https://www.pexels.com/photo/speed-limit-sign-on-road-in-countryside-5144489/
[7] https://flic.kr/p/H4AKUm
[8] https://www.pexels.com/photo/city-road-traffic-street-772582/
[9] https://www.pexels.com/photo/people-brasil-guys-avpaulista-109919/
[10] https://www.pexels.com/photo/man-riding-bicycle-on-city-street-310983/
[11] https://www.pexels.com/photo/policeman-standing-near-modern-car-on-road-5662832/
[12] https://www.pexels.com/photo/road-car-fast-speed-20411/
[13] https://www.pexels.com/photo/modern-truck-in-underground-parking-lot-5182299/
[14] https://flic.kr/p/CDM9mN
[15] https://flic.kr/p/Qgqen5
[16] https://flic.kr/p/JYawr
[17] https://flic.kr/p/G75enM
[18] https://www.pexels.com/photo/kitten-cat-rush-lucky-cat-45170/
[58] https://journals.sagepub.com/doi/10.1177/0278364913491297

Detection
Prof. Dr. Markus Lienkamp

Sebastian Huch, M. Sc.

Agenda

1. General Definitions
2. **Camera Detection**
3. LiDAR Detection
4. Radar Detection
5. Sensor Fusion
6. Datasets
7. Summary

Camera Detection
Motivation

- (Mono-) Camera is the **cheapest sensor** for detection
- Camera is **similar to human eyes**
- Camera imagery **includes almost every information** about the environment
- Camera images can be processed by classical **Computer Vision** (CV)-algorithms and **Deep Learning** (DL)-algorithms

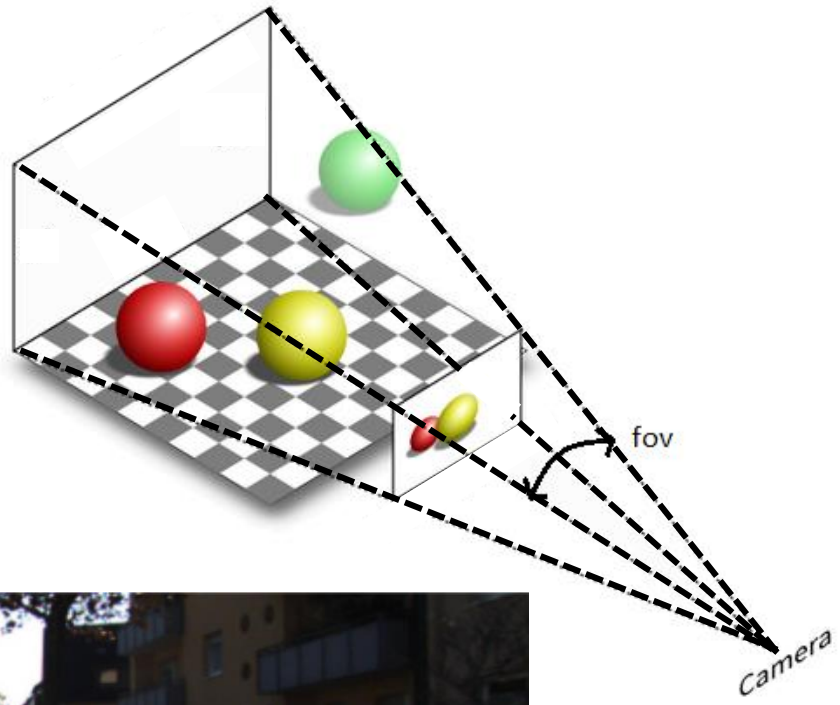
Mono-camera

Stereo-camera

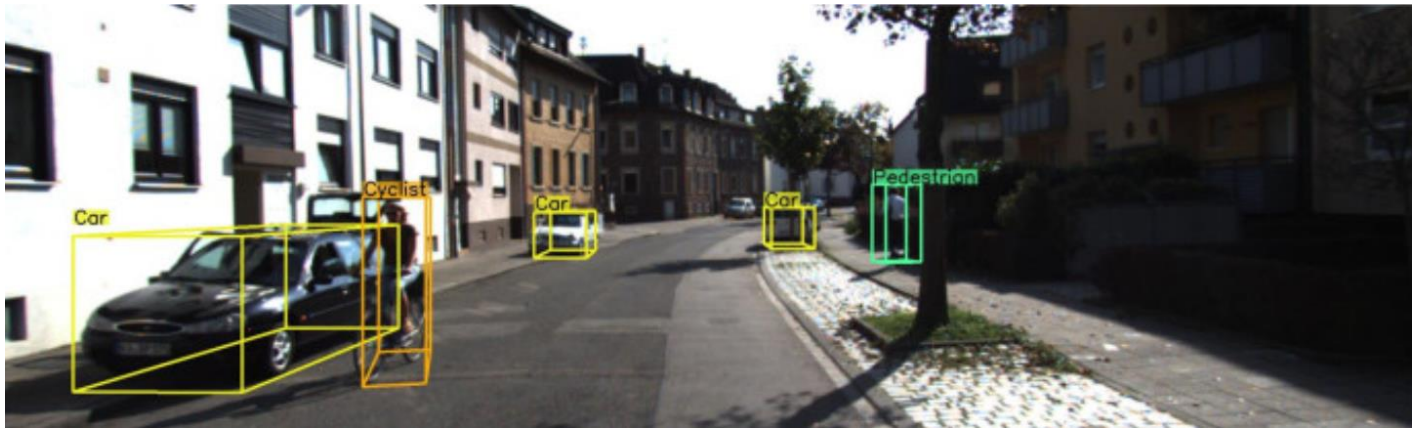
[19][20]

Camera Detection
Challenges

- Loss of depth information – 3D world projected onto 2D images
- Shadow
- Noise

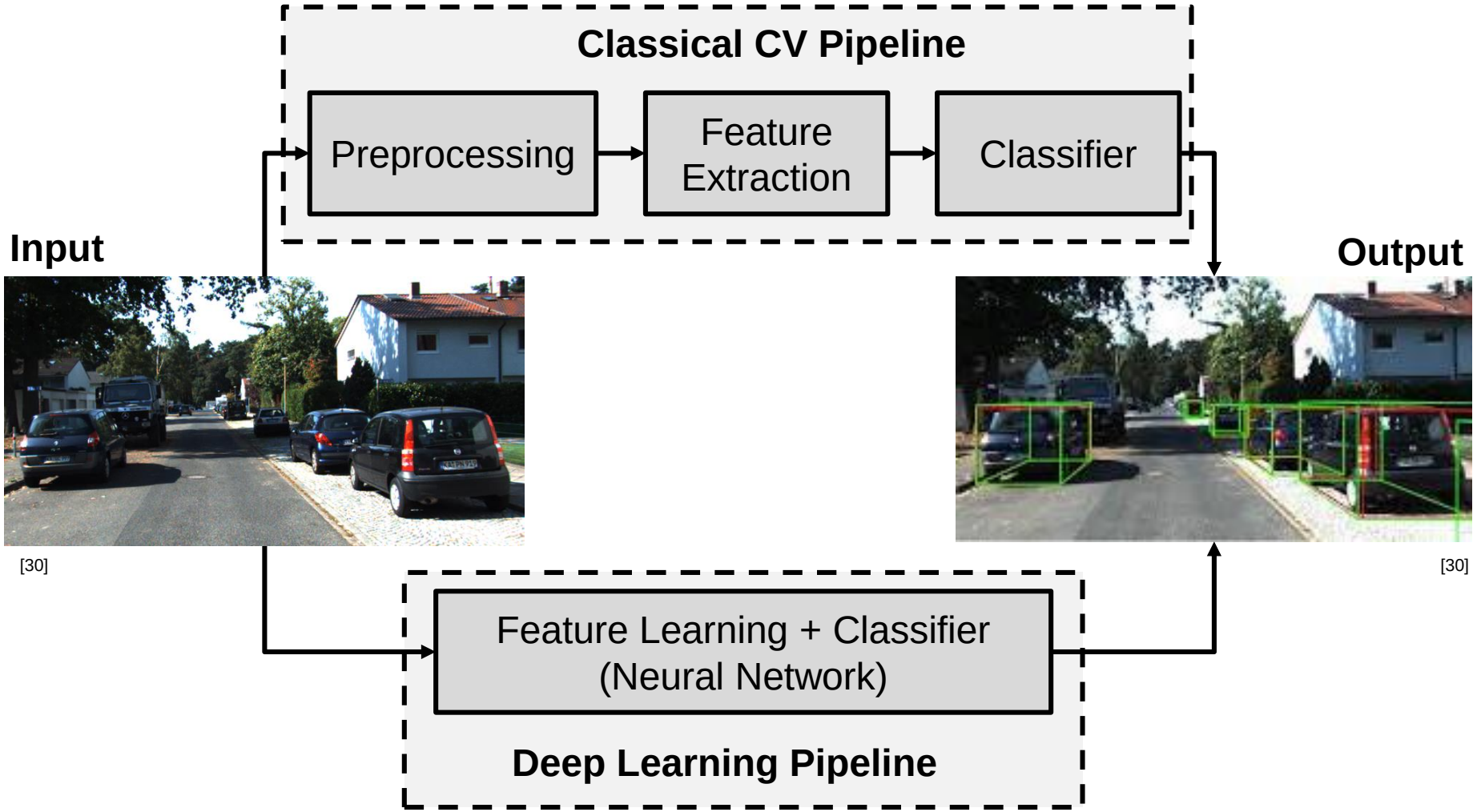


[21]



[22]

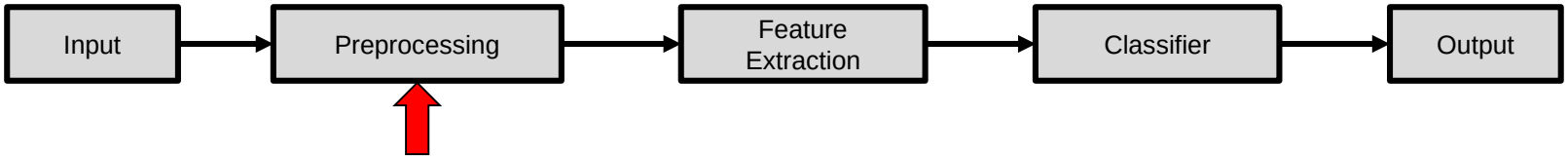
Camera Detection



Additional Slides

More details on the classical CV pipeline can be found in Lecture 2 of Artificial Intelligence in Automotive Technology and on the following additional slides.

Classical CV Pipeline

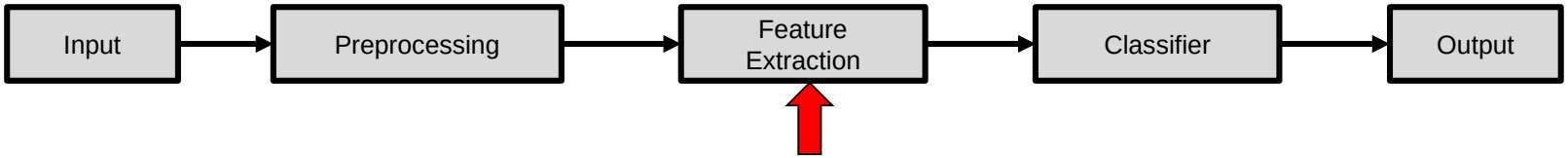


In the **preprocessing** step multiple operations are performed to convert the raw sensor readings to enhanced images.

Preprocessing includes multiple tasks, such as:

- Camera calibration and distortion removal (using extrinsic and intrinsic camera parameters)
- Exposure and gain adjustment
- Contrast enhancements
- Affine transformations (preserves points, straight lines and planes)
- Compression/re-sampling (upsampling or downsampling by changing image width/height)

Classical CV Pipeline

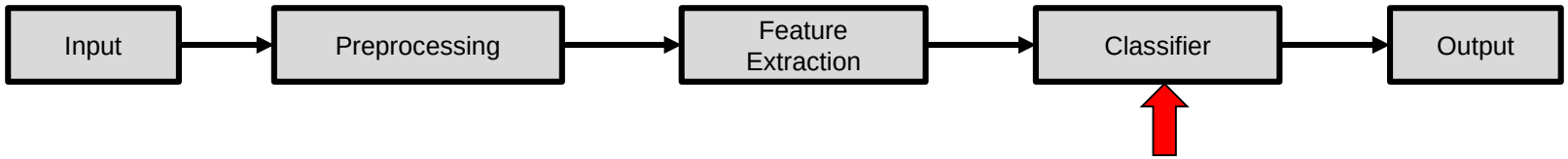


A **feature** is a piece of information which is relevant for solving the computational task related to a certain application.

Features:

- Edges (→ Extraction via Canny edge detection)
- Lines (→ Extraction using Hough transform)
- Corners
- Depth (using disparity maps from stereo cameras)
- Histograms of oriented gradients (HOG)

Classical CV Pipeline



A **classifier** predicts to which of a set of categories the extracted features belong.

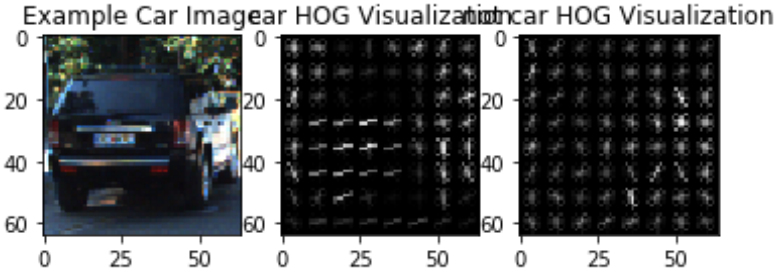
Classification algorithms:

- Support Vector Machine (SVM)
- Decision trees
- AdaBoost

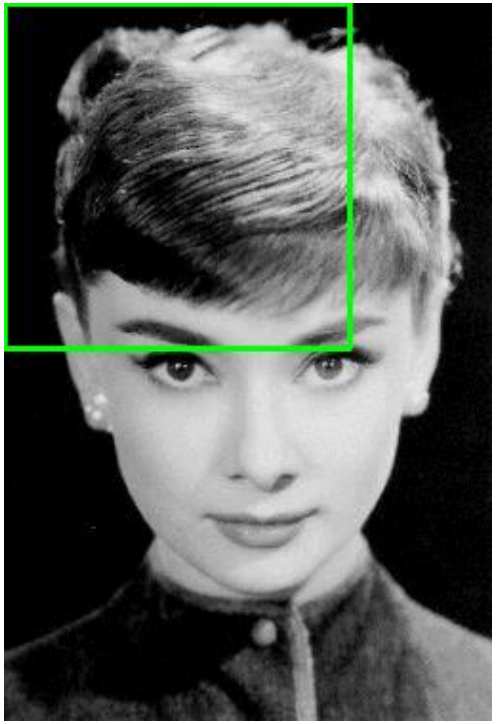
Classical CV Pipeline

Example Pipeline: **Sliding window approach**

- Rectangular region (window) with fixed width/height slides across the image (use multiple image scales since objects can have different sizes!)
- Histogram of oriented gradients (HOG) is computed for each window
- Run linear Support Vector Machine (SVM) classifier on all locations



Source: https://github.com/TusharChugh/Vehicle-Detection-HOG



Source: https://www.pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/

Limitations of the Classical CV Pipeline

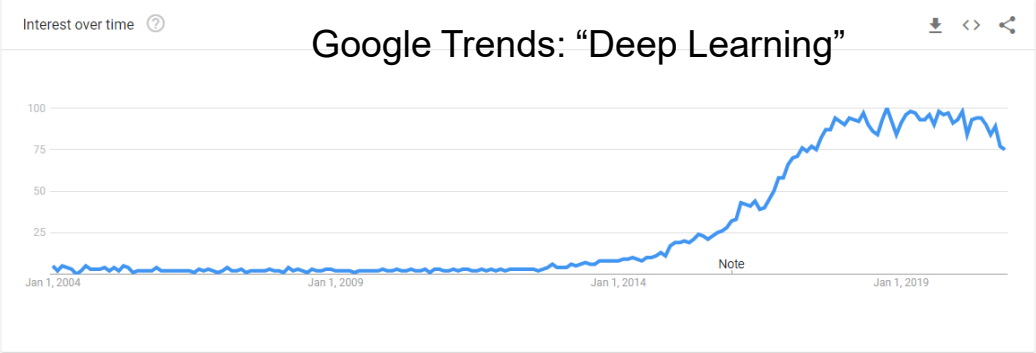
- Requires hand-crafted features that are difficult to design and limited in their representation capabilities
- Performance depends on how accurately the features are identified and extracted
- Feature extraction/selection is time consuming and depend on expert domain knowledge
- No end-to-end learning possible

Camera Detection
Deep Learning Algorithms

Eliminating the limitations of the classical CV algorithms by using Deep Learning algorithms

→ No hand-crafted features

→ Better performance

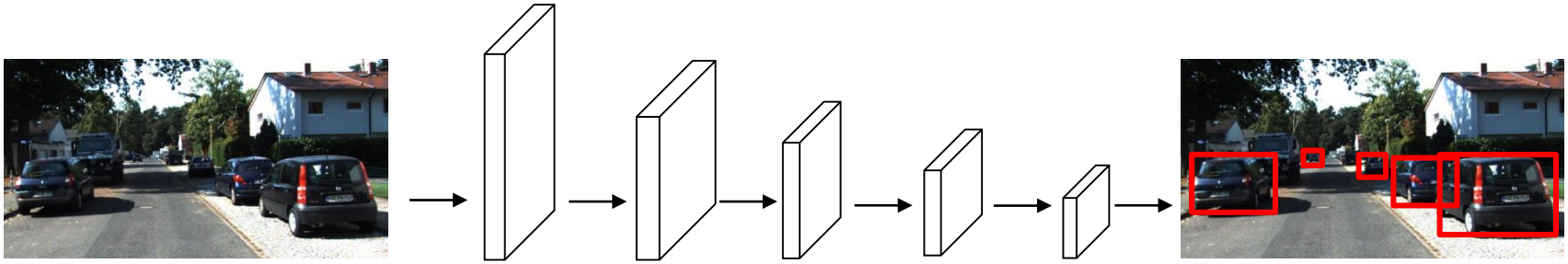


Why now?

- GPU
- Dataset availability
- Efficient algorithms (CNNs)

Camera Detection
2D Object Detection: History

AlexNet (2012)
R-CNN (2013)
Fast R-CNN (2014)
Faster R-CNN (2015)
YOLO (2015)
…

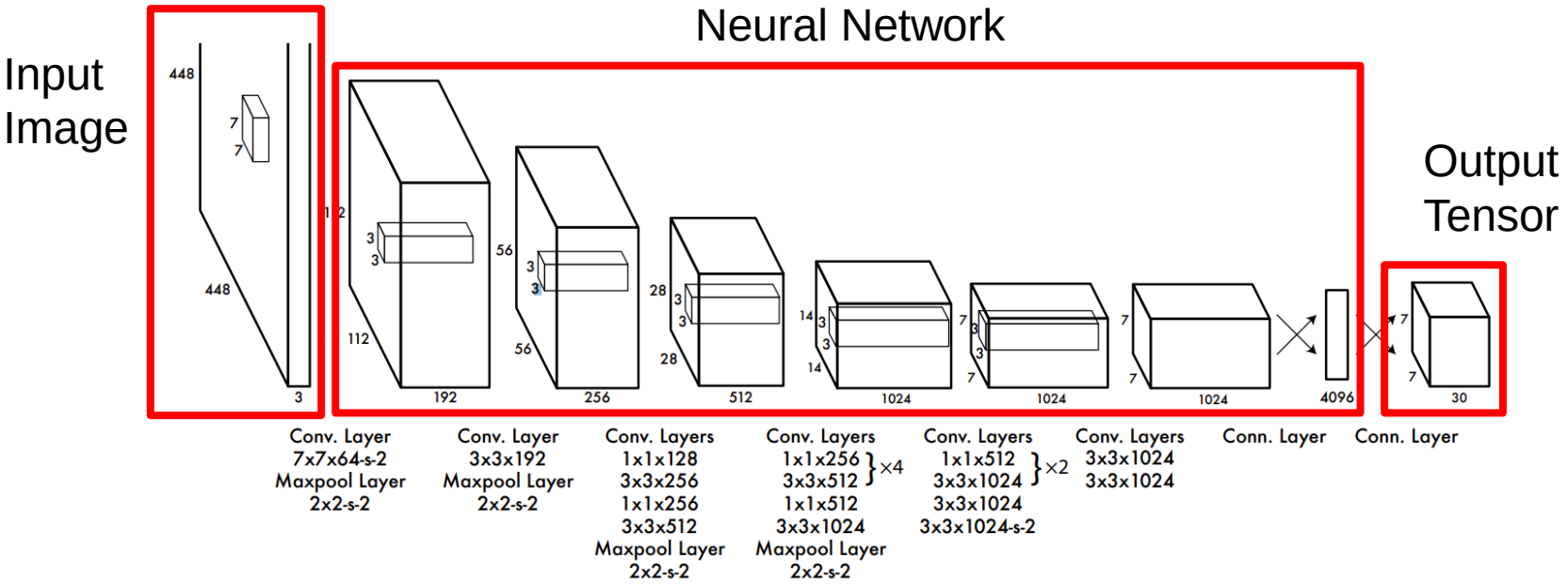


Convolutional Neural Network

[30]

Camera Detection

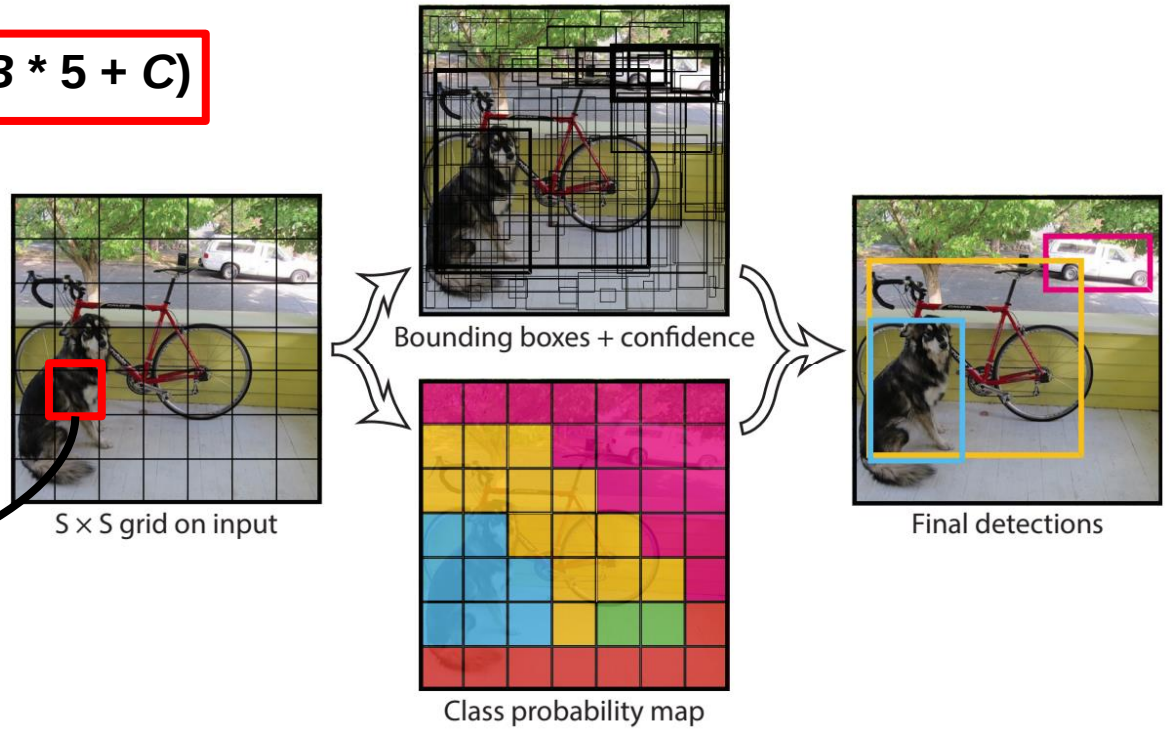
2D Object Detection: YOLO



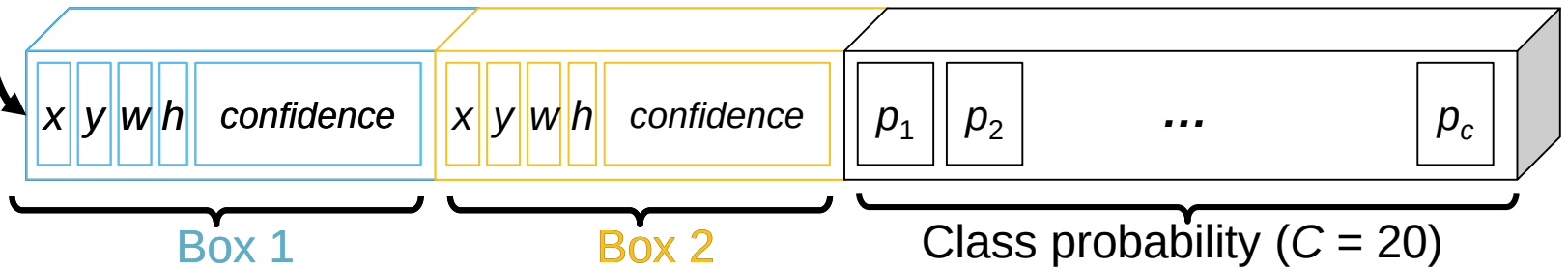
[60]

Camera Detection
2D Object Detection: YOLO

Output Tensor $S * S * (B * 5 + C)$



2 Boxes *B* per cell,
5 parameters each



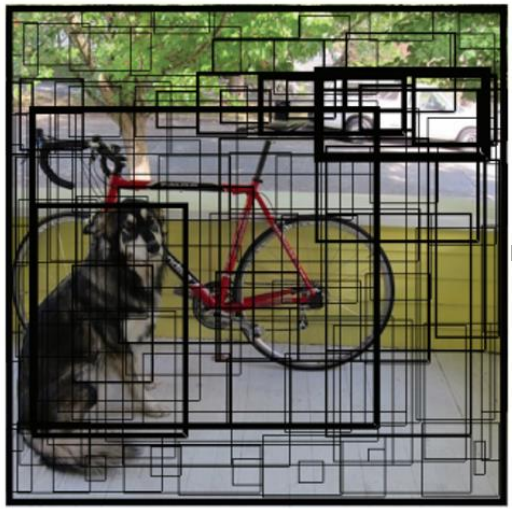
Additional Slides

The input image is downsampled with a convolutional neural network CNN (multiple convolutional layers in row). The output of this CNN is a tensor of size **S * S * (B * 5 + C)**.

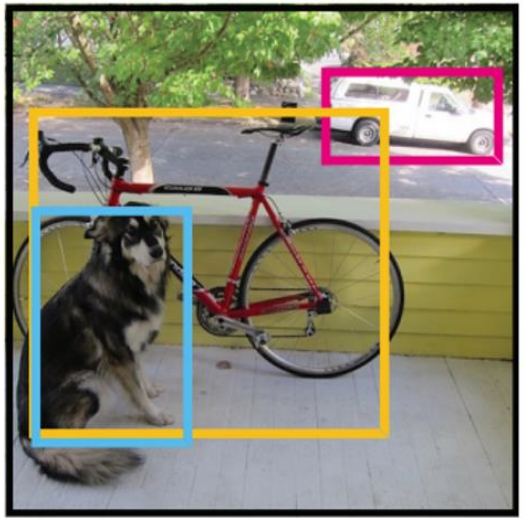
- S is the final grid size in each dimension, in the case of YOLOv1, S = 7. The input image is therefore downsampled to a 7*7 grid, hence it contains 49 cells in total. Each cell contains a feature vector with the length (B * 5 + C).
- B is the number of boxes that are predicted for each cell (YOLOv1 predicts 2 boxes per cell, B=2). Each predicted box B has 5 parameters:
  - X = x distance of the bounding box center from the cell‘s upper left corner (relativ to the image width)
  - Y = y distance of the bounding box center from the cell‘s upper left corner (relativ to the image height)
  - W = width of the predicted bounding box (relativ to the image width)
  - H = height of the predicted bounding box (relativ to the image height)
  - Confidence = confidence of the network (between 0 and 1) that an object is inside the predicted box
- C is the number of classes for the classification. For each cell, the network predicts the probability of this cell belonging to each class. YOLOv1 predicted 20 classes, therefore C=20.

Camera Detection
2D Object Detection: YOLO

7*7 grid, 2 boxes per cell = 98 boxes per image



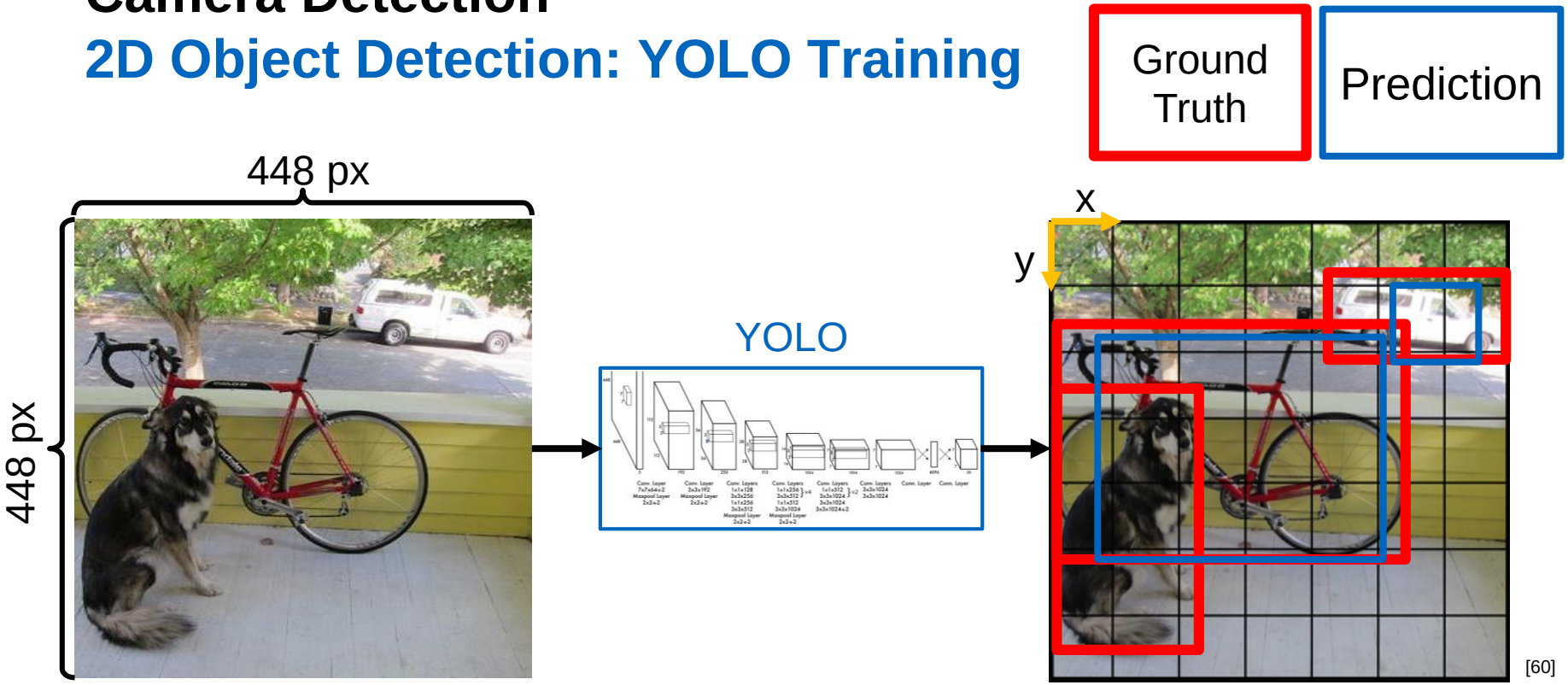
Only boxes with high confidence are selected!



[60]

Camera Detection

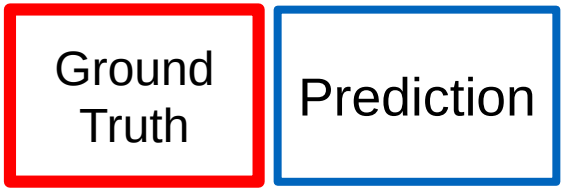
2D Object Detection: YOLO Training



Prediction
(*x, y, w, h, conf, class*):
[194, 223, 279, 219, 0.87, bicyle]
[381, 101, 84, 77, 0.56, car]

Ground Truth
(*x, y, w, h, conf, class*):
[78, 294, 139, 254, 1.0, dog]
[186, 223, 352, 231, 1.0, bicyle]
[361, 95, 177, 82, 1.0, car]

Camera Detection
2D Object Detection: YOLO Training



Prediction
(*x, y, w, h, conf, class*):
[194, 223, 279, 219, 0.87, bicyle]
[381, 101, 84, 77, 0.56, car]

Ground Truth
(*x, y, w, h, conf, class*):
[78, 294, 139, 254, 1.0, dog]
[186, 223, 352, 231, 1.0, bicyle]
[361, 95, 177, 82, 1.0, car]

$$Loss = \sum_{\text{Cell}} \sum_{\text{Box}} \left[(x_{\text{pred}} - x_{\text{GT}})^2 + (y_{\text{pred}} - y_{\text{GT}})^2 \right] + \ \ldots$$

Additional Slides

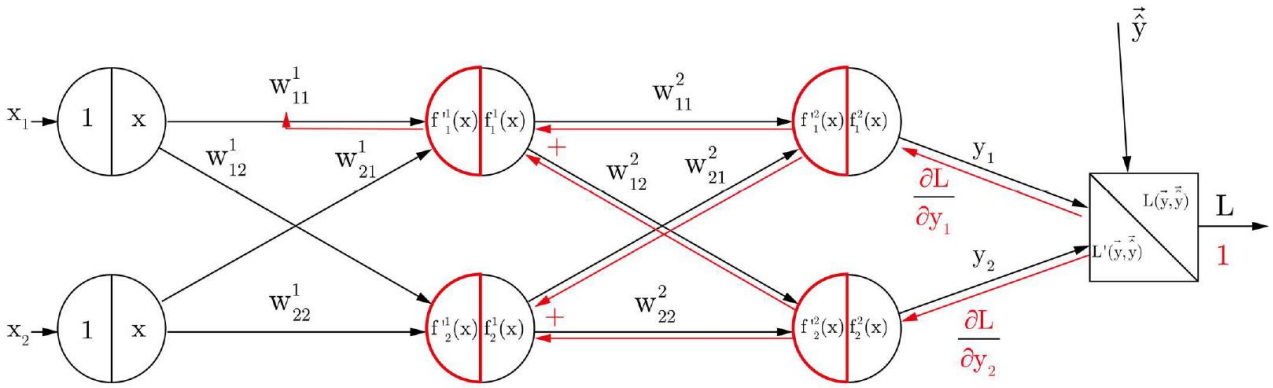
Actual loss function of YOLOv1

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
$$

Camera Detection
2D Object Detection: YOLO Training

$$Loss = \sum_{\text{Cell}} \sum_{\text{Box}} \left[(x_{\text{pred}} - x_{\text{GT}})^2 + (y_{\text{pred}} - y_{\text{GT}})^2 \right] + \ \ldots$$

Feed loss backwards using backpropagation!
Weights of YOLO are updated



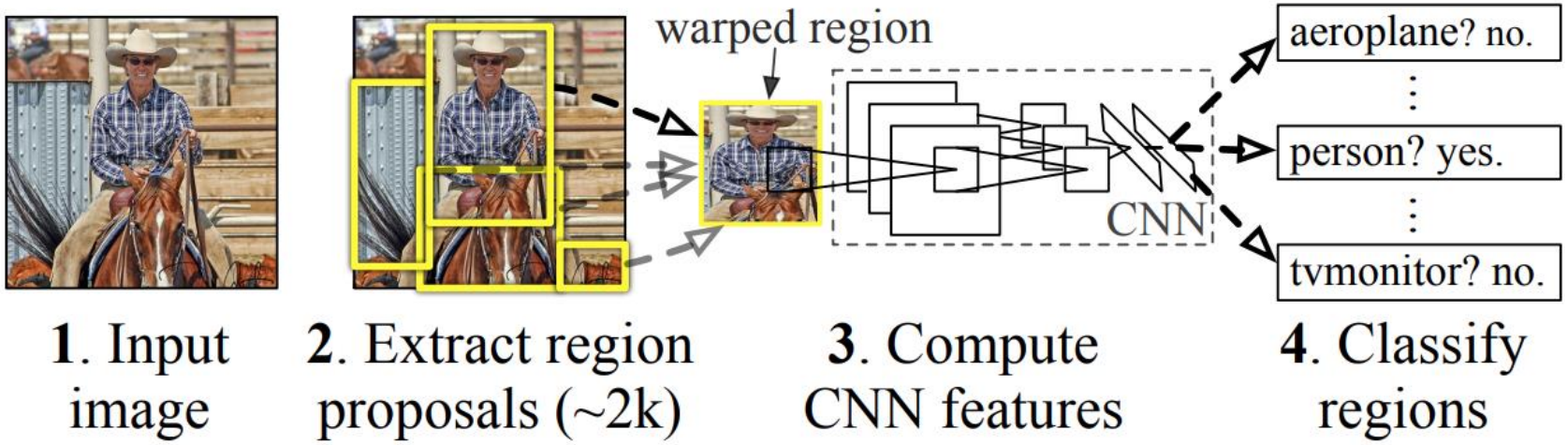
Additional Slides

Backpropagation is explained in detail in the course „Artificial Intelligence in Autonomotive Technology“, Lecture 8: Deep Neural Networks

Additional Slides

R-CNN (Region-based Convolutional Network):

- Region of Interest (ROI) extraction via selective search (~2000)
- CNN to extract features from all ROIs
- Linear SVMs for classification
- Not end-to-end trainable
- Two-stage detector: slow (~47 s per image!)

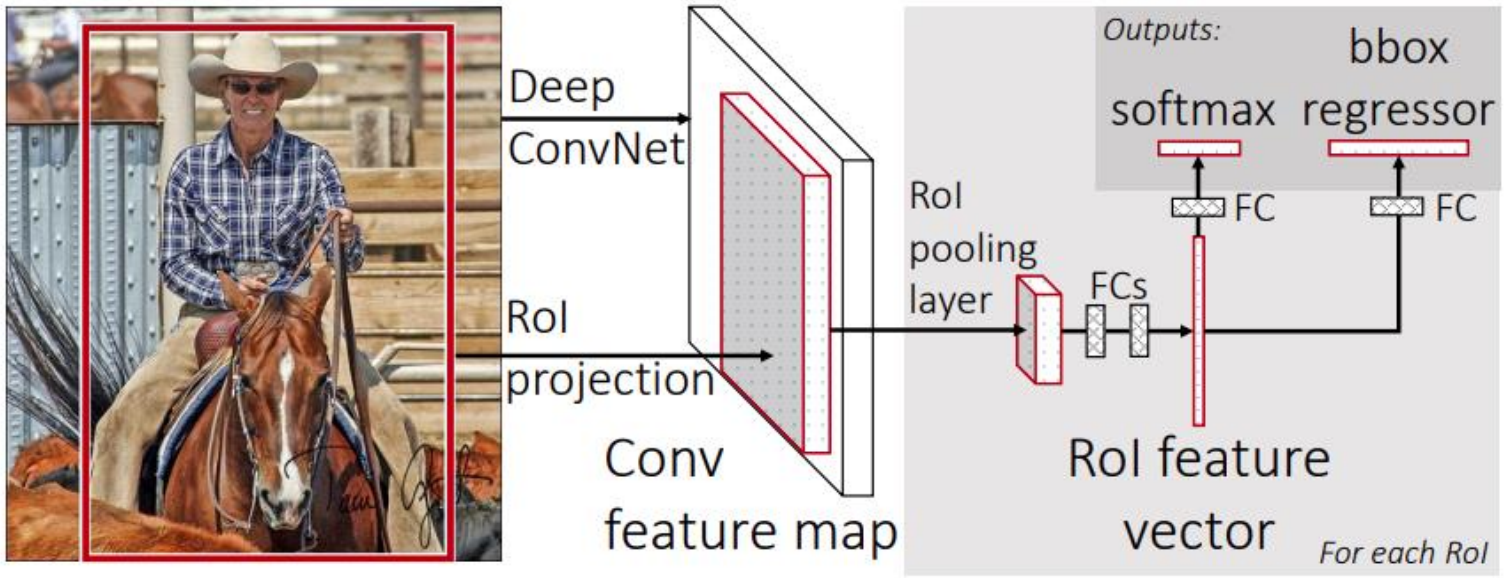


Source: Girshick et al. (2014) – Region-Based Convolutional Networks for Accurate Object Detection and Segmentation

Additional Slides

Fast R-CNN:

- Expensive CNN step is only computed once instead of ~2000x, and Conv feature map is shared by all ROIs
- ROI pooling layer to downsample feature maps with different sizes into a fixed-size vector
- Two-stage detector: slow (~2.3 s per image!)
- End-to-end training possible

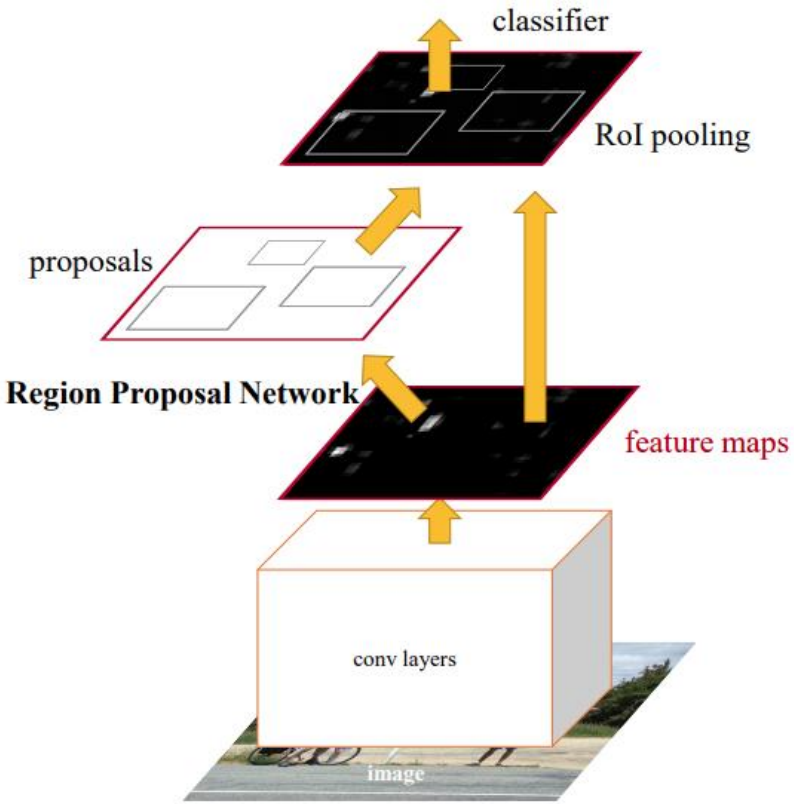


Source: Girshick (2014) – Fast R-CNN

Additional Slides

Faster R-CNN:

- Selective search region proposal is replaced by Region Proposal Network (RPN)
- Classification head is identical to Fast R-CNN
- Two-stage detector: ~0.2 s per image

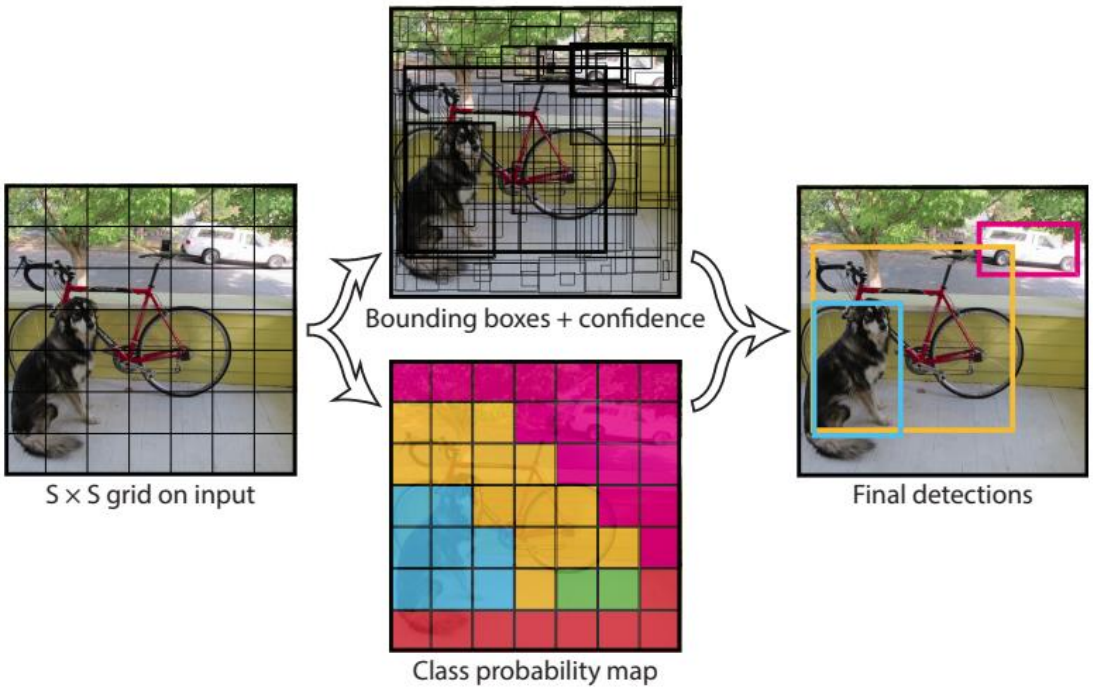


Source: Ren et al. (2015) – Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Additional Slides

YOLO (You Only Look Once):

- Combination of region proposal and region classification in the same CNN
- Input split into an SxS grid, each cell directly regresses bounding box + confidence score
- One-stage detector: fast (~22 ms per image)



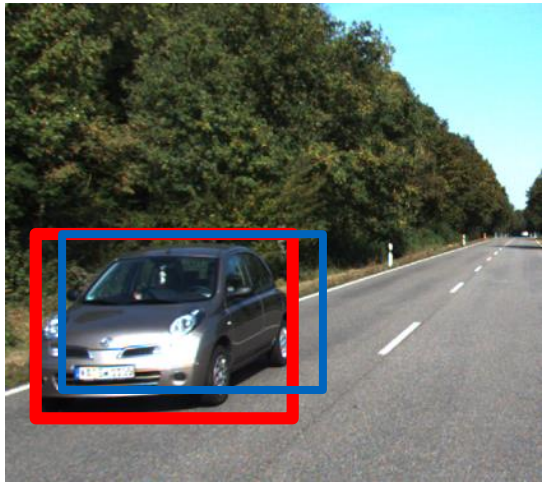
Source: Redmon et al. (2015) – You Only Look Once: Unified, Real-Time Object Detection

Camera Detection
Object Detection: Evaluation

Ground Truth

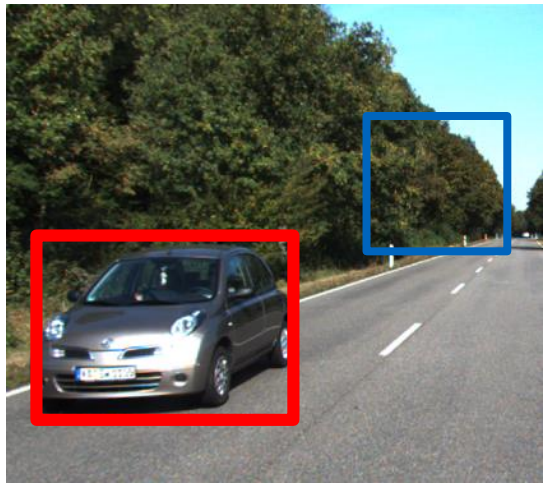
Prediction

True Positive



True Negative

False Positive



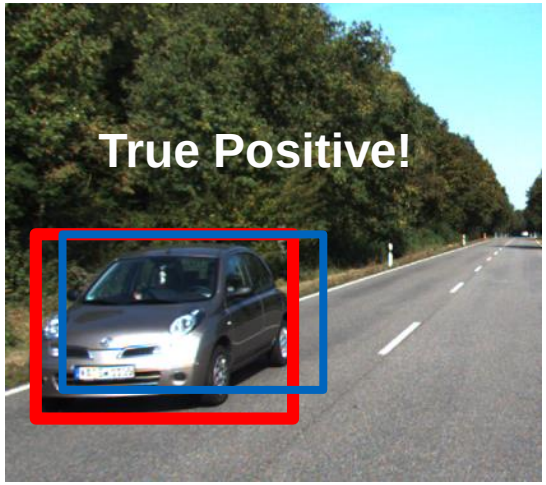
False Negative

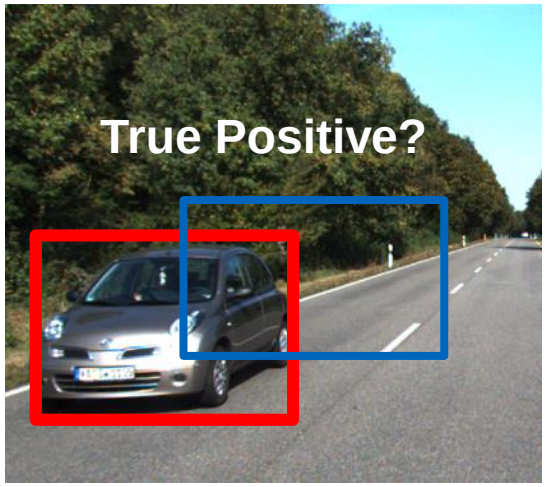
[68]

Camera Detection
Object Detection: Evaluation

Ground Truth

Prediction





[68]

Intersection over Union IoU

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

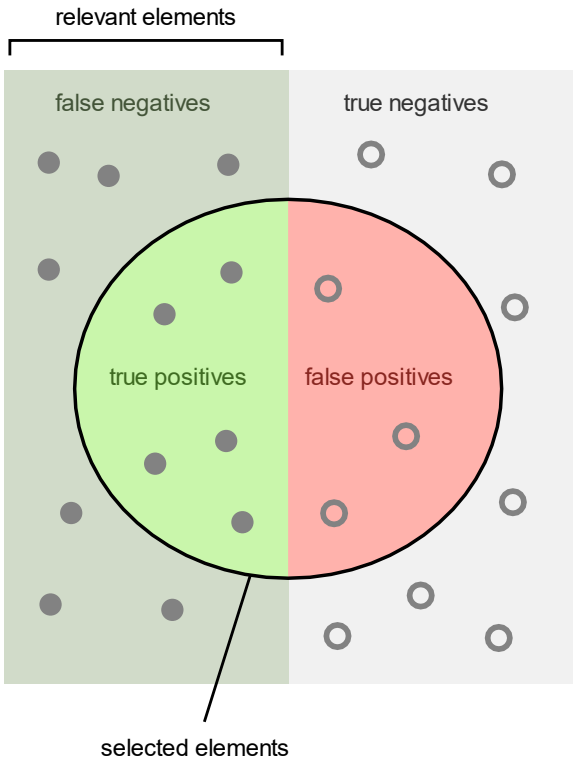
[67]

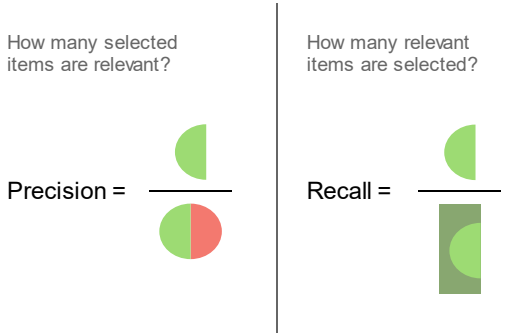
TP only if IoU > threshold (e.g. 0.5)

Camera Detection
Object Detection: Evaluation

$$\textbf{Precision} = \frac{TP}{TP + FP} = \frac{\text{True positives}}{\text{no. of predicted positive}}$$

$$\textbf{Recall} = \frac{TP}{TP + FN} = \frac{\text{True positives}}{\text{no. of actual positive}}$$



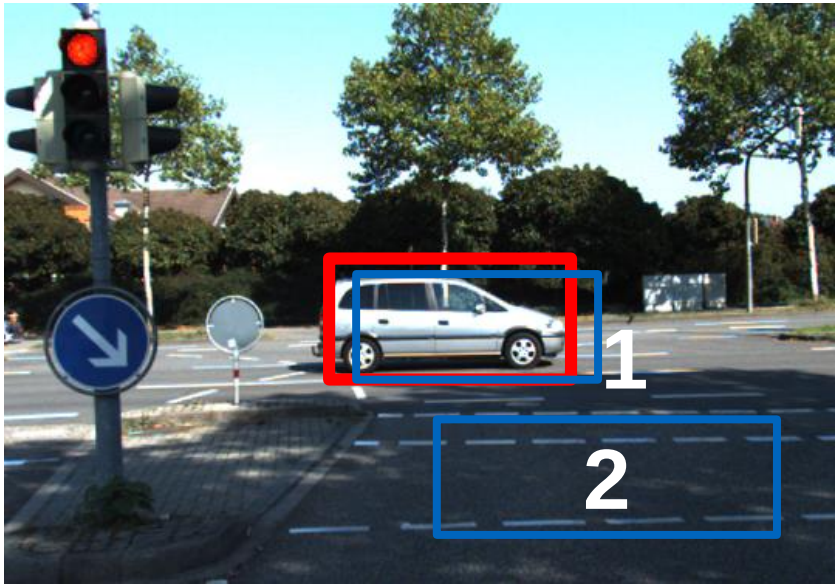


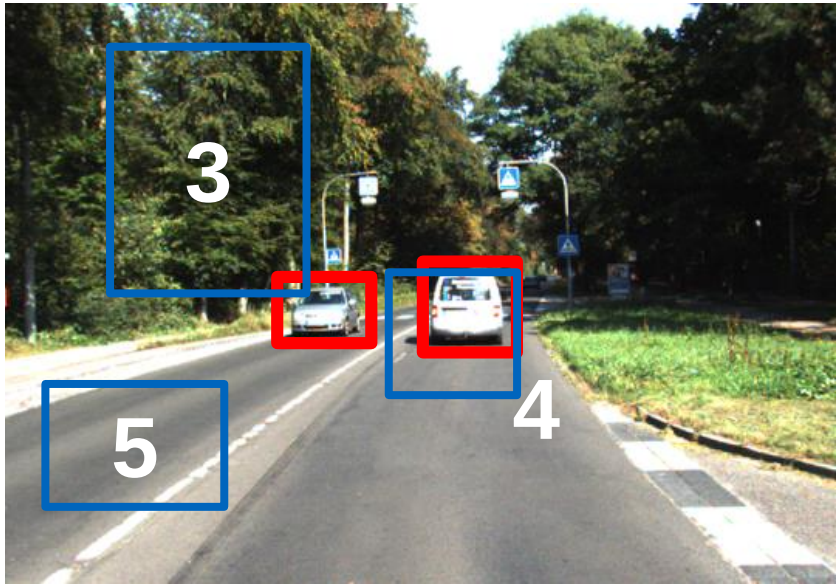
[69]

Camera Detection
Object Detection: Evaluation

Ground Truth

Prediction





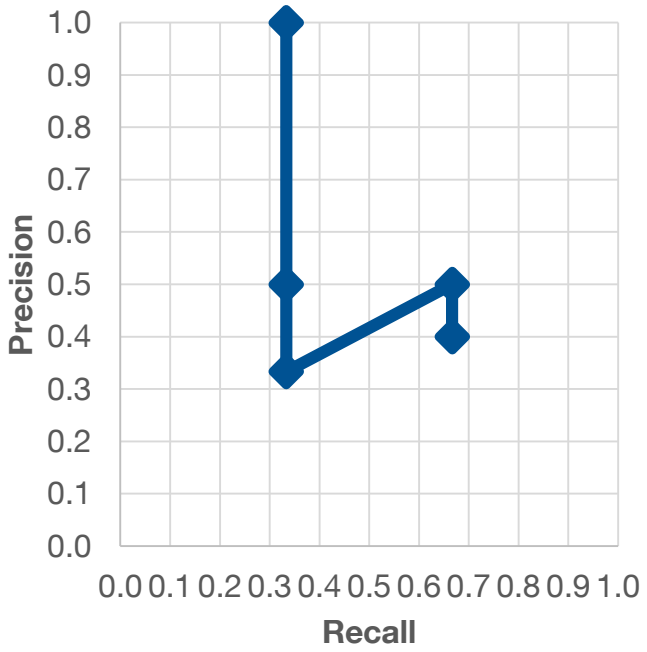
[68]

Box	1	2	3	4	5
TP or FP?	TP	FP	FP	TP	FP
Precision	1/1	1/2	1/3	2/4	2/5
Recall	1/3	1/3	1/3	2/3	2/3

Camera Detection

Object Detection: Evaluation

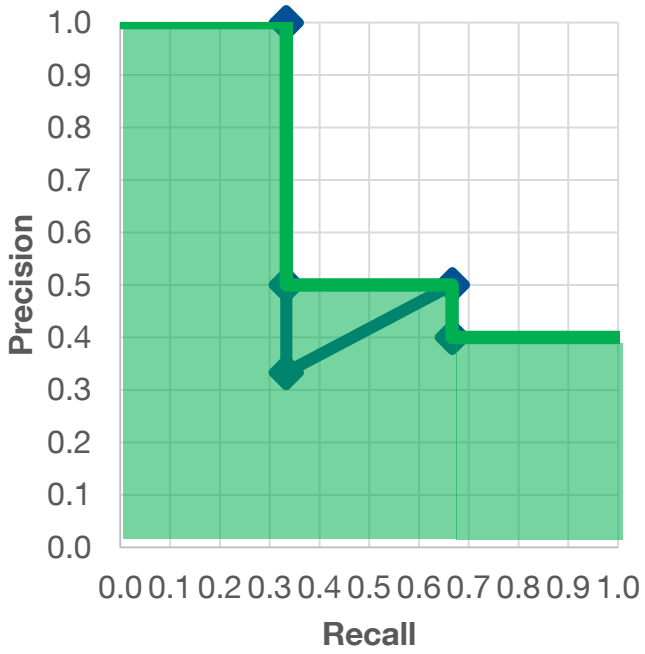
Box	1	2	3	4	5
TP or FP?	TP	FP	FP	TP	FP
Precision	1/1	1/2	1/3	2/4	2/5
Recall	1/3	1/3	1/3	2/3	2/3



Precision-Recall-Curve

Camera Detection
Object Detection: Evaluation

Box	1	2	3	4	5
TP or FP?	TP	FP	FP	TP	FP
Precision	1/1	1/2	1/3	2/4	2/5
Recall	1/3	1/3	1/3	2/3	2/3



Average Precision *AP* = Area under curve

AP = 1/3 * 1.0 + 1/3 * 0.5 + 1/3 * 0.4

AP = 0.633 = 63.3 %

Additional Slides

Object Detection Metrics

Different Object Detection algorithms are usually evaluated on the same dataset, e.g. KITTI for autonomous driving. By calculating the Average Precision AP on a standardized dataset, the algorithm’s performance can be compared with a single number. The average precision can be calculated for 2D detections (by comparing the 2D overlap IoU of predicted with ground truth boxes) or for 3D detections (by comparing the 3D overlap IoU of predicted with ground truth boxes). The evaluation is usually separated for different classes (car, pedestrian, cyclist, etc.). The mean average precision mAP is the mean AP for all classes.

Camera Detection

Monocular 3D object detection (mono3DOD)

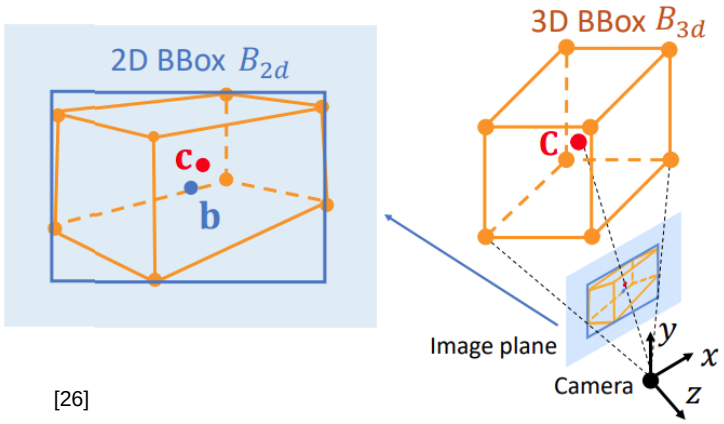
How to lift 2D detections to 3D?

All previously introduced networks predict the bounding box (BBox) of detected objects only in 2D space on the image plane. Autonomous vehicles operate in 3D world and therefore need to localize the surrounding objects in 3D space.

- 2D: 4 DOF – 2D BBox center and 2D size (*x, y, w, h*)
- 3D: 7 DOF – 3D BBox center, 3D size and yaw (*x, y, z, w, h, l,* Ψ (=yaw))

DOF = degrees of freedom

→ To lift 2D to 3D, we can use **strong prior information**, such as the well-known geometry and size of objects (vehicles/pedestrians/ bicycles)



[26]

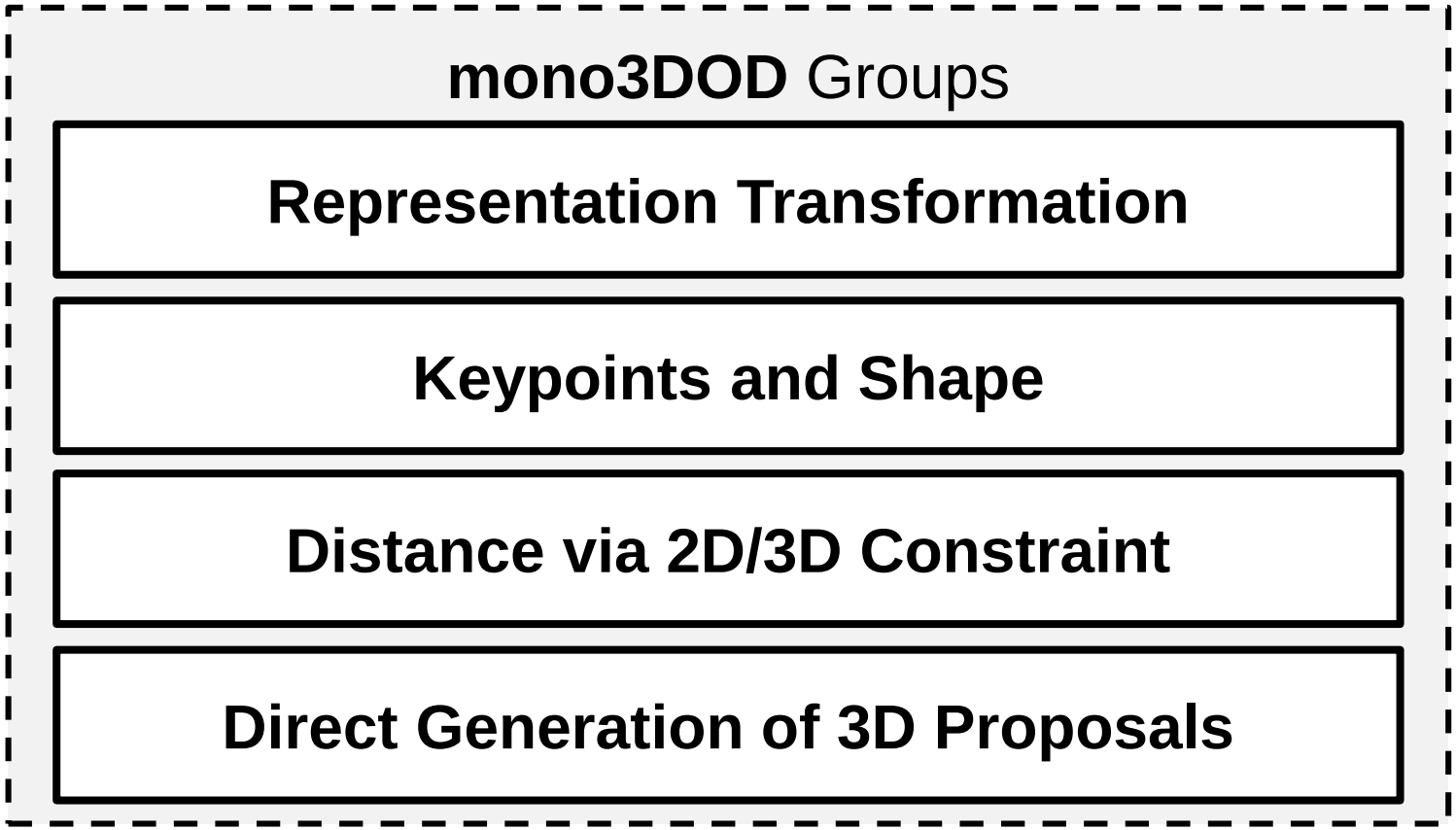
Additional Slides

Degrees of freedom

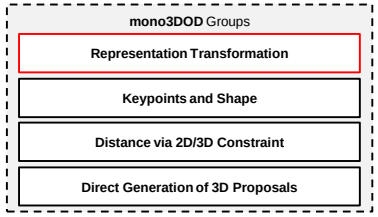
In 3D space, each detected object has 9 degrees of freedom, namely the center of the bounding box (x, y, z), the object‘s dimension (h, w, l) and the three angles (Ψ, θ, Φ) (yaw, pitch, roll). The yaw angle is the rotation of the object around its vertical axis. The pitch and roll angles are usually close to 0 degrees (exception: motorcycles) and hence neglected. This results in 7 DOF per object in the 3D space.

Camera Detection

Monocular 3D object detection (mono3DOD)



[27]



Camera Detection

mono3DOD – Representation Transformation

Motivation:

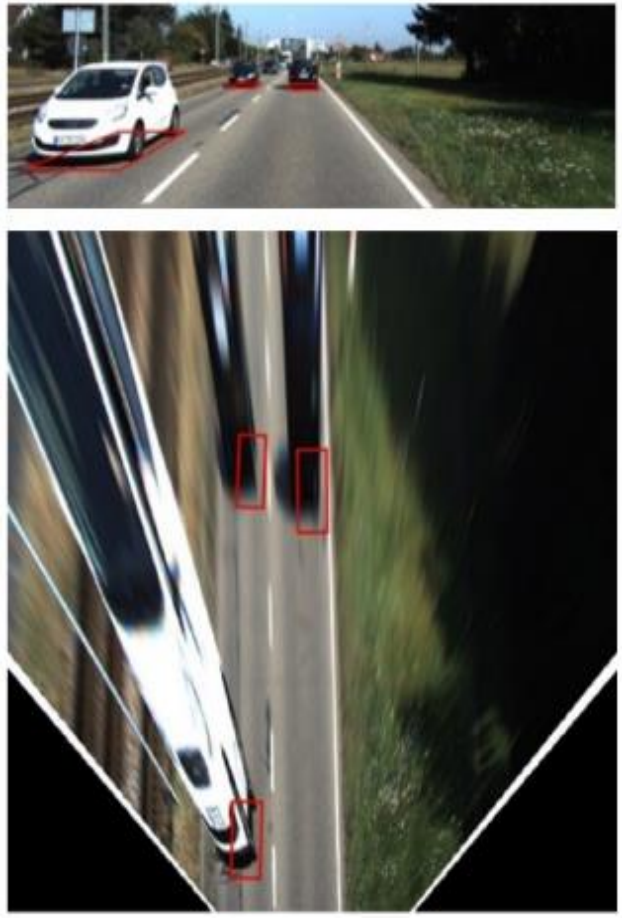
- Perspective view poses two challenges:
  1. Scale variation due to distance
  2. Occlusion

Approach:

- Lift perspective images to **birds-eye-view (BEV)** or transform to **point clouds**

→ Cars have the same size invariant to distance to ego vehicle

→ Different cars do not overlap (if they are not on top of each other…)



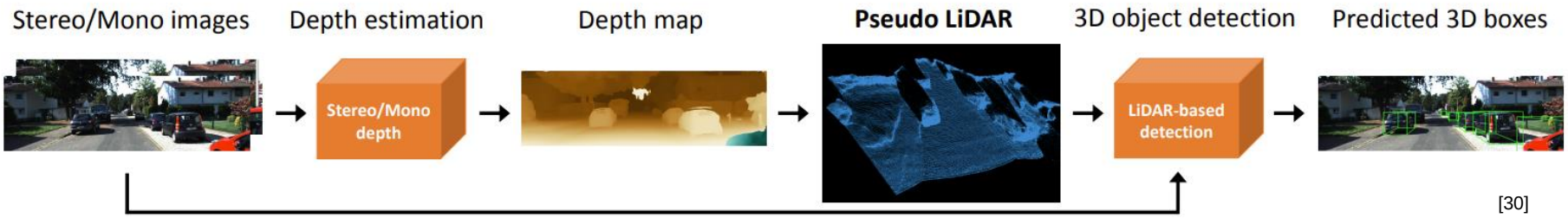
[28]

Additional Slides

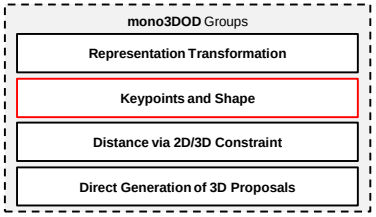
mono3DOD – Representation Transformation – Example

Pseudo-LiDAR (2018)

- Generation of pseudo-LiDAR point clouds from RGB images
- Derivation of the 3D location of each pixel in depth map
- Apply state-of-the-art LiDAR-based 3D object detectors (any could work that uses 3D point clouds!)



[30]



Camera Detection

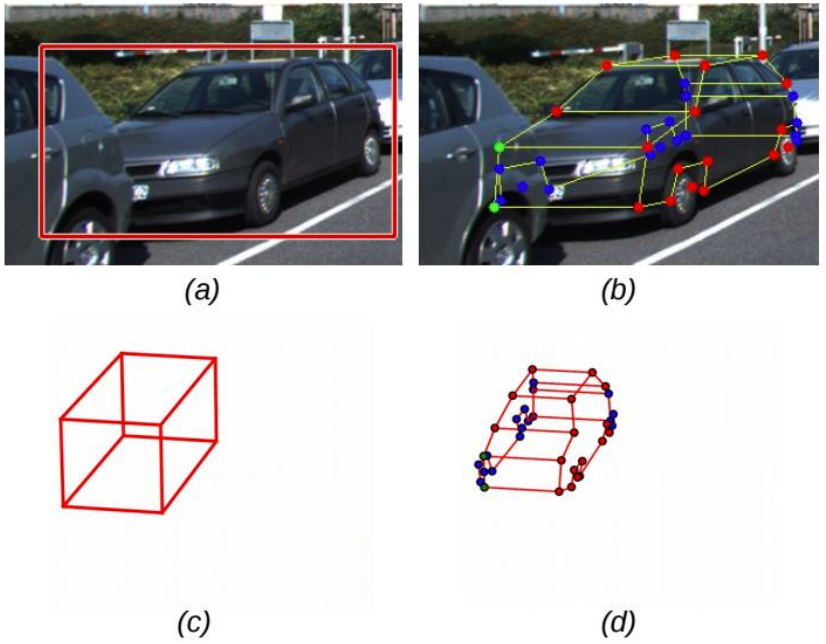
mono3DOD – Keypoints and Shape

Motivation:

- Vehicles are rigid bodies with distinctive common parts (keypoints)
- Dimension of objects is largely known (overall and inter-keypoint size)

Approach:

- Extension of 2D object detection frameworks (Faster R-CNN, YOLO, …) to **predict keypoints**

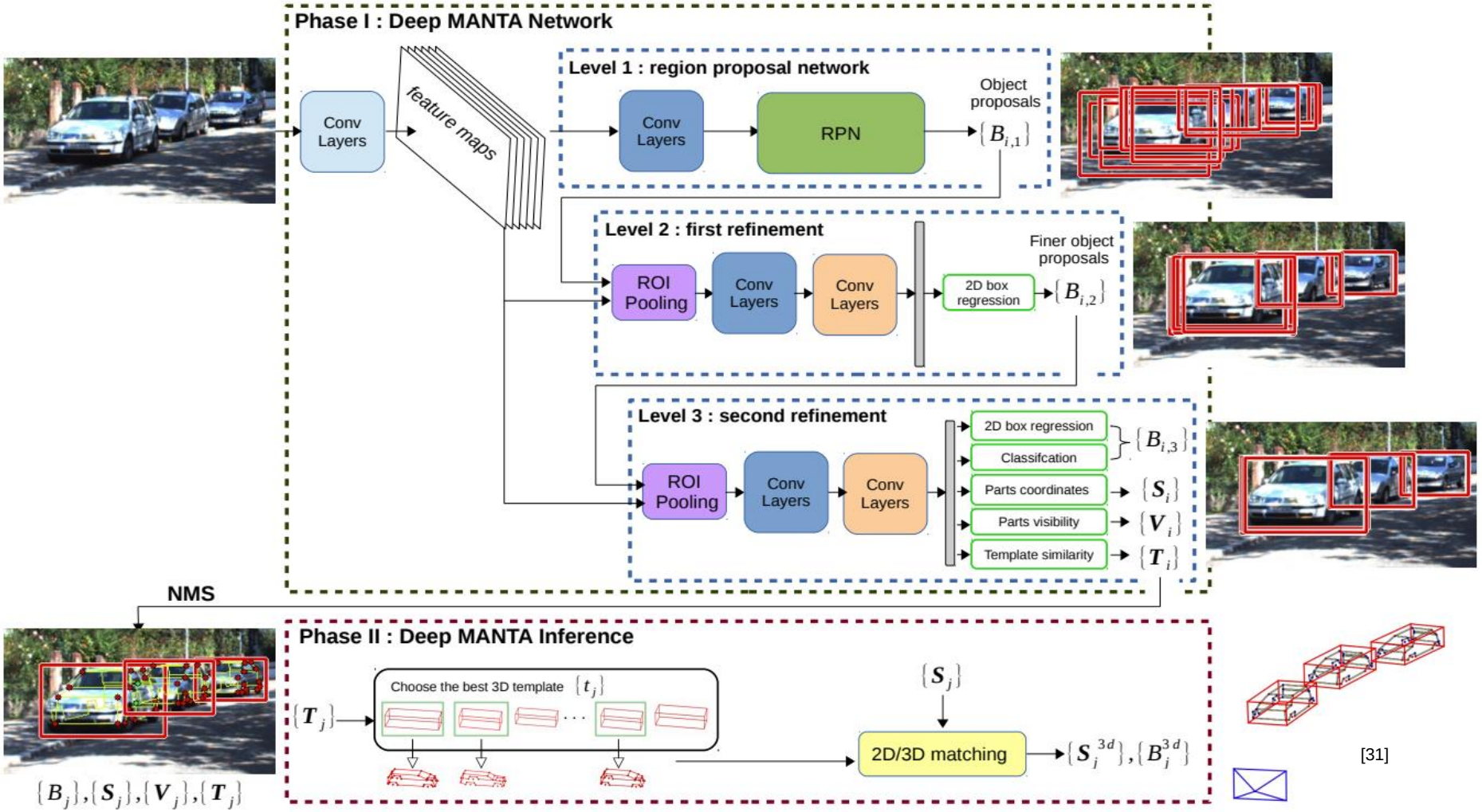


[31]

Additional Slides

mono3DOD – Keypoints and Shape – Example

Deep MANTA



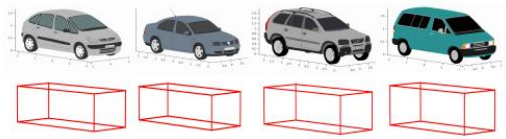
[31]

Additional Slides

Phase I: Cascaded Faster R-CNN framework to regress 2D BBox, classification, 2D keypoints, keypoint visibility, and template similarity

Phase II: Best matching 3D CAD template *c* is selected and 2D/3D matching is performed to recover 3D position and orientation

3D template dataset (103 CAD models, 36 keypoints per model)



$\rightarrow \{\bar{t}_m^{3d}\}_{m\in\{1,\dots,M\}}$ with $\bar{t}_m^{3d} = (w_m, h_m, l_m)$

Phase I returns **template similarity** vector $\mathbf{T}_i$

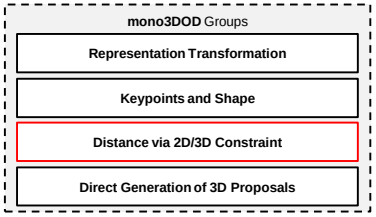
$\mathbf{T}_i = \{r_m\}_{m\in\{1,\dots,M\}}$ with $r_m = (r_x, r_y, r_z)$ (3 scaling factors for each 3D CAD model in dataset)

→ Scaling factors are applied on 3D CAD template to fit real 3D template, resulting templates are $\{t_m^{3d}\}_{m\in\{1,\dots,M\}}$

→ $\mathbf{T}_i$ is similarity between detected vehicle and all 3D templates of the dataset

$$\textbf{Best template } \boldsymbol{c} = \underset{\boldsymbol{m}\in\{\boldsymbol{1},\dots,\boldsymbol{M}\}}{\textbf{argmin}}\, \boldsymbol{d}(\bar{\boldsymbol{t}}_{\boldsymbol{m}}^{\boldsymbol{3d}}, \boldsymbol{t}_{\boldsymbol{m}}^{\boldsymbol{3d}})$$

[31]



Camera Detection

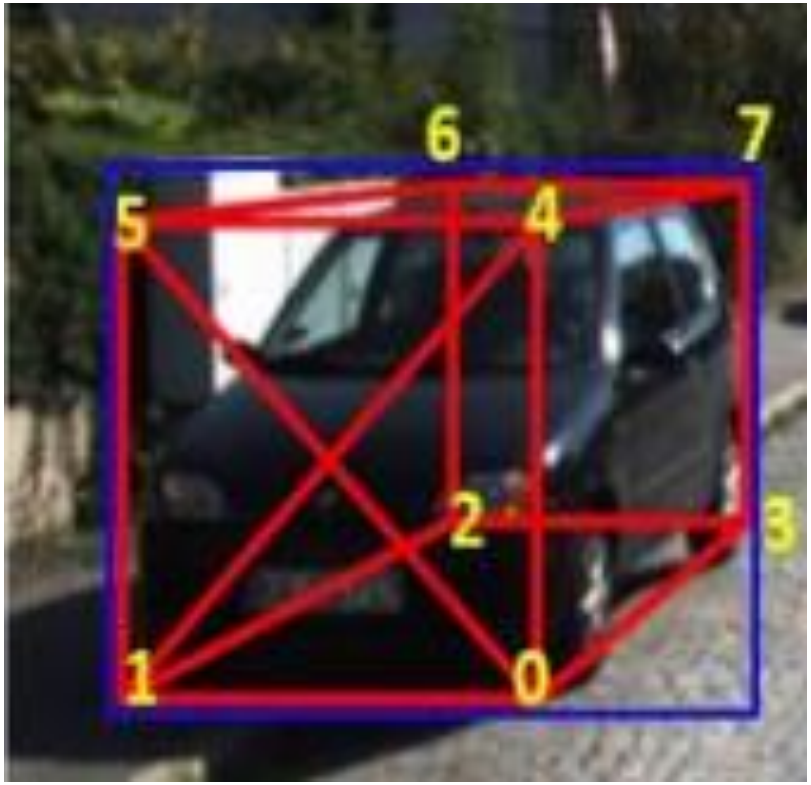
mono3DOD – Distance via 2D/3D Constraint

Motivation:

- When lifting 2D detections to 3D, they need to maintain 2D/3D consistency (e.g. projection of 3D BBox fits tightly into 2D detection window)

Approach:

- Extension of 2D object detection frameworks (Faster R-CNN, YOLO, …) to **regress orientation and size of the object's 3D BBox**



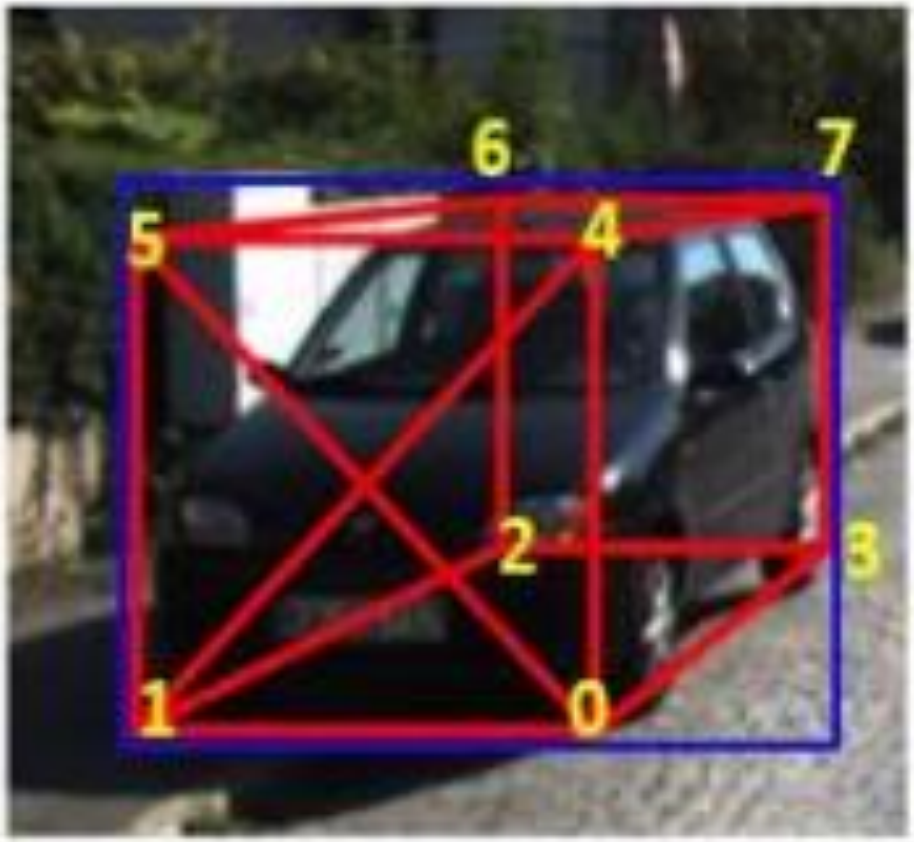
[32]

Additional Slides

mono3DOD – Distance via 2D/3D Constraint – Example

Deep3DBox (2016):

Perspective projection of 3D BBox (red 3D box with 8 cuboid vertexes, numbered 0 to 7) should fit tightly with its 2D detection window (blue 2D box). At least one cuboid vertex should project onto each of the four sides of the 2D Bbox.



[32]

Additional Slides

mono3DOD – Distance via 2D/3D Constraint – Example

Deep3DBox (2016):

- 2D/3D BBox constraints to lift 2D to 3D
- 4 equations for the top-left and bottom-right corners of the 2D BBox (x_{min}, y_{min}, x_{max}, y_{max}) to solve 3 unknown translations (T_x, T_y, T_z) of the object
  → over-determined problem

$$x_{min} = \left(K_{3\times3}[R_{3\times3}|T_{3\times1}] \begin{bmatrix} X^{(1)}_{3\times1} \\ 1 \end{bmatrix} \right)_x$$

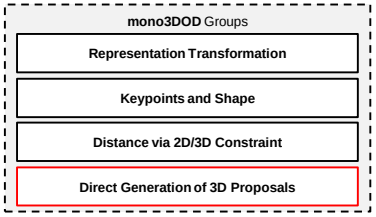
$$x_{max} = \left(K_{3\times3}[R_{3\times3}|T_{3\times1}] \begin{bmatrix} X^{(2)}_{3\times1} \\ 1 \end{bmatrix} \right)_x$$

$$y_{min} = \left(K_{3\times3}[R_{3\times3}|T_{3\times1}] \begin{bmatrix} X^{(3)}_{3\times1} \\ 1 \end{bmatrix} \right)_y$$

$$y_{max} = \left(K_{3\times3}[R_{3\times3}|T_{3\times1}] \begin{bmatrix} X^{(4)}_{3\times1} \\ 1 \end{bmatrix} \right)_y$$

[34]

K … camera intrinsic matrix
R … camera extrinsic rotation matrix
T … (unknown) object translation
$X^{(i)}$ … selected vertices of 3D BBox whose projection lies on the boundaries of the 2D BBox (4 vertices for 4 boundaries)



Camera Detection

mono3DOD – Direct Generation of 3D Proposals

Motivation:

- Solve mono3DOD without any transformations/keypoints/constraints

Approach:

- Directly regress 3D BBoxes from an input image

CNN

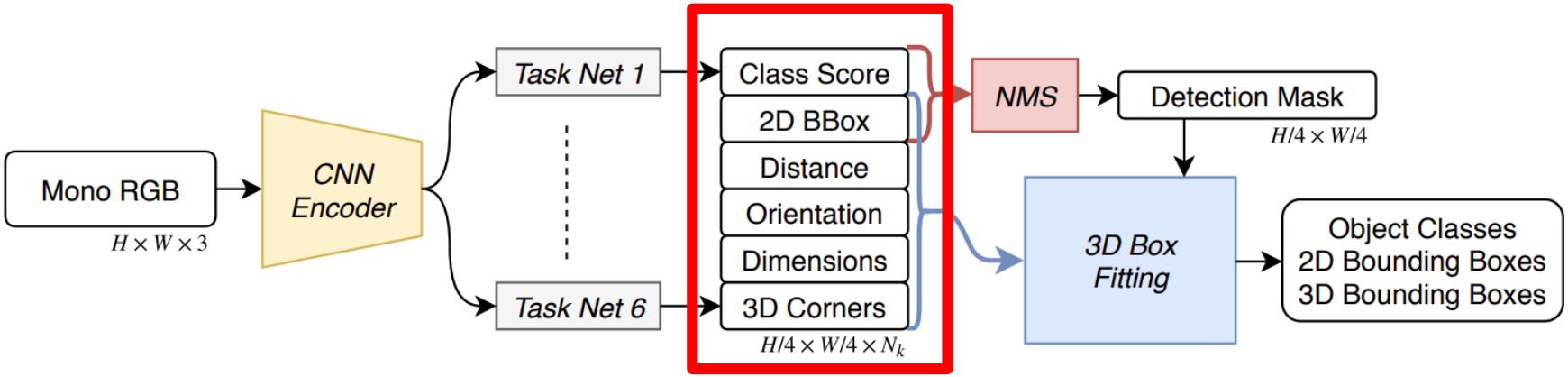
[30]

Additional Slides

mono3DOD – Distance via 2D/3D Constraint – Example

SS3D (2019):

- Direct regression object's class, 2D BBoxes and 3D BBoxes, parameterized as 26 outputs
- 26 outputs have different weights which are learned during training



[36]

Camera Detection
mono3DOD – Performance

KITTI Benchmark 3D Object Detection, Class Car, Moderate (09.12.2020)

Method	AP in %	Sensor
SS3D	7.68	Camera
Pseudo-LiDAR	7.50	Camera

Camera Detection

Traffic Light and Sign Detection

Motivation

- Robust detection and classification of traffic lights and traffic signs is required to obey traffic rules
- Traffic lights and traffic signs are designed for human perception, which is primarily based on vision

→ Camera is the only sensor that can **detect *and* classify traffic lights and traffic signs**

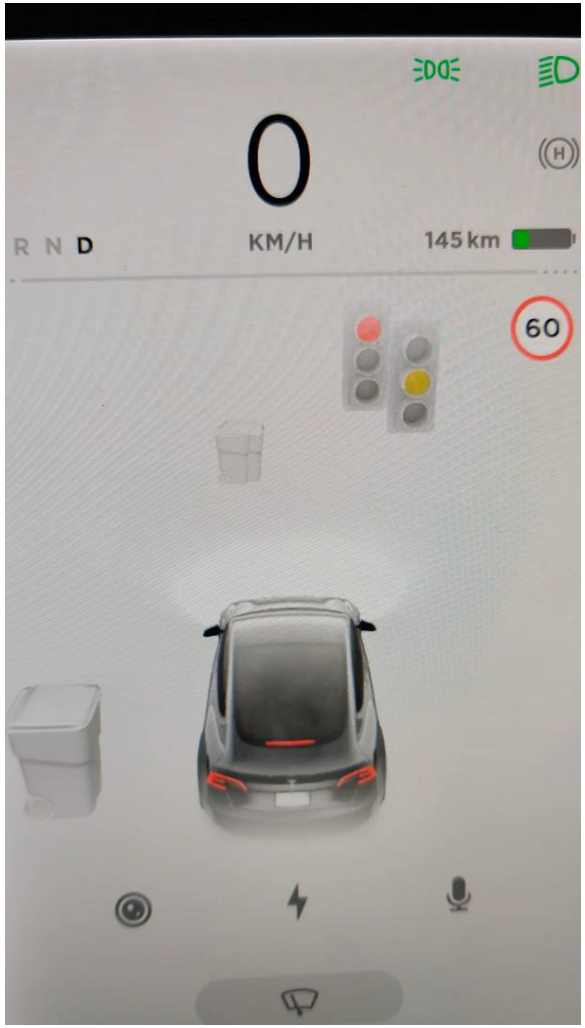
Camera Detection

Traffic Light and Sign Detection

Challenges

[37]

[38]



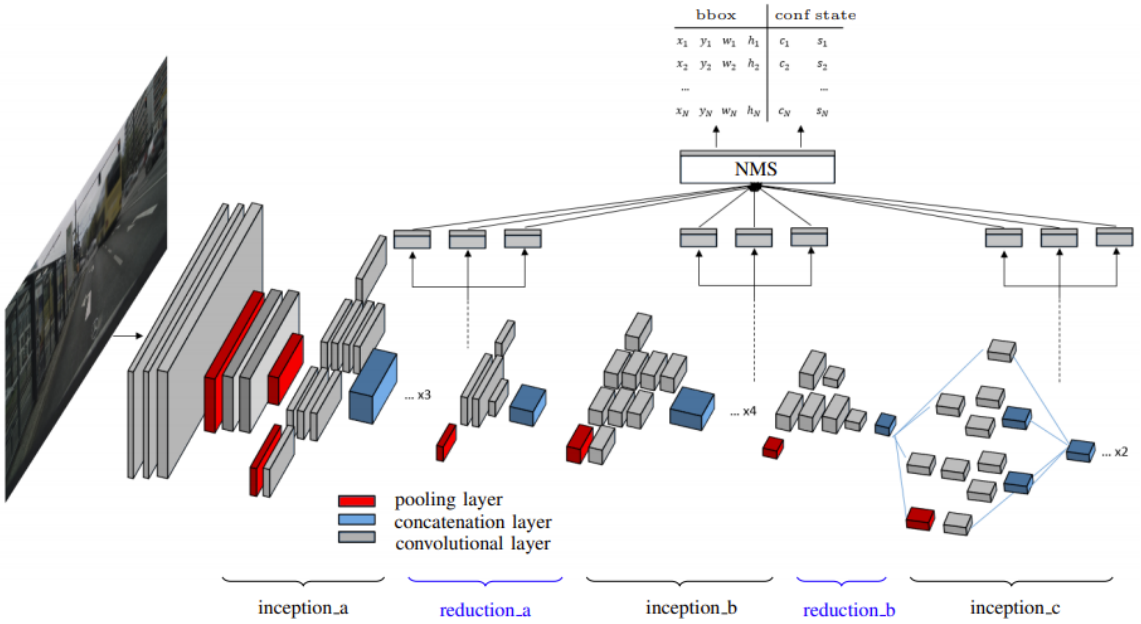
[39]

Additional Slides

Example for Traffic Light Detection

TL-SSD: Single shot detector for traffic light detection:

- CNN for feature extraction
- Conv layers are set up on existing feature maps and predict fixed number of bounding boxes + confidence score, based on predefined prior boxes
- Predefined prior boxes are distributed over a feature map (grid)



Source: Mueller and Dietmayer (2018) – Detecting Traffic Lights by Single Shot Detection

Sources

[19] https://autonomoustuff.com/product/flir-blackfly/
[20] https://autonomoustuff.com/product/nerian-karmin2/
[21] https://xem.github.io/articles/webgl-guide.html
[22] http://cvlab.cse.msu.edu/project-m3d-rpn.html
[23] Wang et. al (2018) – Deep learning for smart manufacturing: Methods and applications
[26] Qin et al. (2018) – MonoGRNet: A Geometric Reasoning Network for Monocular 3D Object Localization
[27] https://towardsdatascience.com/monocular-3d-object-detection-in-autonomous-driving-2476a3c7f57e
[28] Kim and Kum (2019) – Deep Learning based Vehicle Position and Orientation Estimation via Inverse Perspective Mapping Image
[29] Srivastava et al. (2019) – Learning 2D to 3D Lifting for Object Detection in 3D for Autonomous Vehicles
[30] Wang et al. (2018) – Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving
[31] Chabot et al. (2017) – Deep MANTA: A Coarse-to-fine Many-Task Network for joint 2D and 3D vehicle analysis from monocular image
[32] Mousavian et al. (2016) – 3D Bounding Box Estimation Using Deep Learning and Geometry
[33] Liu et al. (2019) – Deep Fitting Degree Scoring Network for Monocular 3D Object Detection
[34] https://towardsdatascience.com/geometric-reasoning-based-cuboid-generation-in-monocular-3d-object-detection-5ee2996270d1
[35] Brazil and Liu (2019) – M3D-RPN: Monocular 3D Region Proposal Network for Object Detection
[36] Joergensen et al. (2019) – Monocular 3D Object Detection and Box Fitting Trained End-to-End Using Intersection-over-Union Loss
[37] https://en.m.wikipedia.org/wiki/Traffic_Light_Tree#
[38] Yang et al. (2018) – Deep detection network for real-life traffic sign in vehicular networks
[39] https://www.reddit.com/r/teslamotors/comments/jfya32/interesting_stoplights/
[60] Redmon et al. (2015) – You Only Look Once: Unified, Real-Time Object Detection
[61] Krizhevsky et al. (2012) – ImageNet Classification with Deep Convolutional Neural Networks
[67] https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/
[68] KITTI Dataset: http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d
[69] https://en.wikipedia.org/wiki/Precision_and_recall

Detection
Prof. Dr. Markus Lienkamp

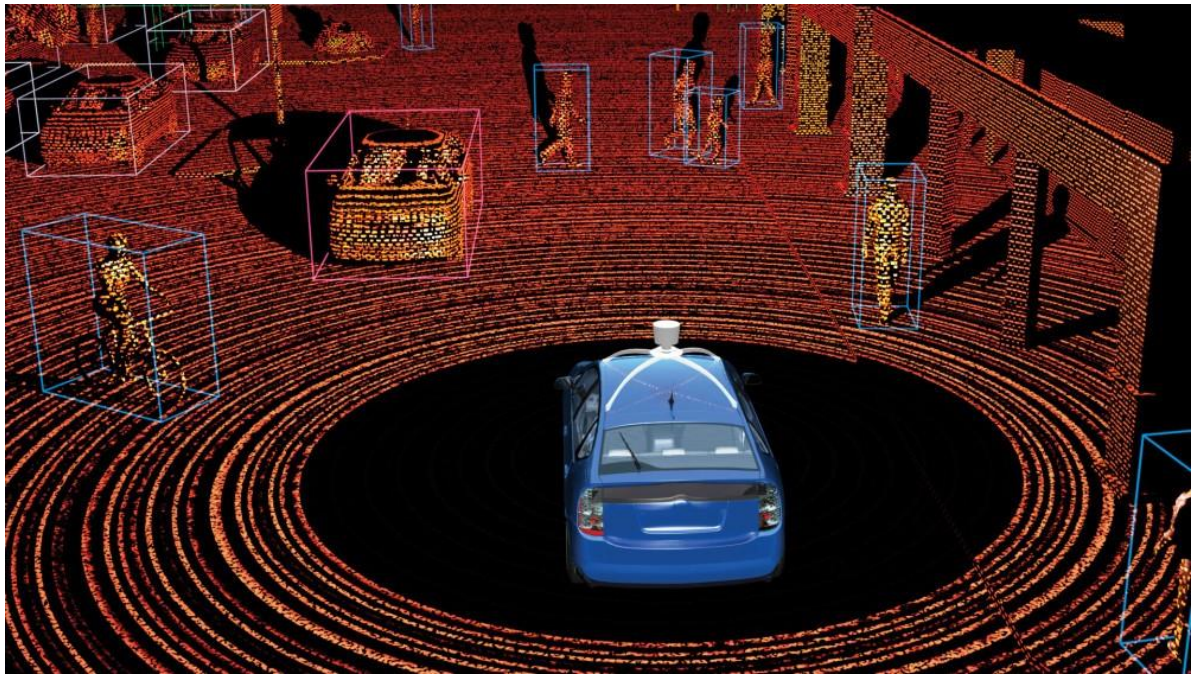
Sebastian Huch, M. Sc.

Agenda

1. General Definitions
2. Camera Detection
3. **LiDAR Detection**
4. Radar Detection
5. Sensor Fusion
6. Datasets
7. Summary

LiDAR Detection
Motivation

- Continuous 360 degrees of visibility with single sensor possible
- Accurate depth information
- Point clouds have higher resolution than Radar measurements



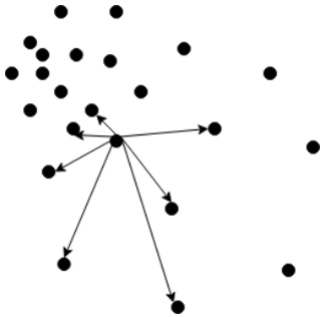
[40]

LiDAR Detection
Challenges

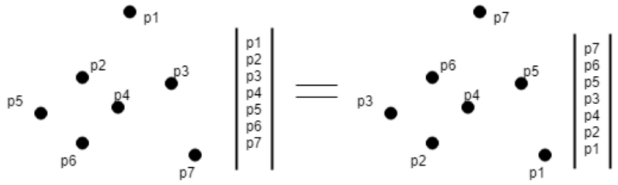
LiDAR sensor outputs are point clouds, which have the following characteristics:

- **Irregularity**: points are not even sampled around the sensor; some regions have dense points while others have sparse points
- **Unstructured**: points are not on a fixed grid, distance between neighboring points varies (compared to equidistant pixels in 2D images)
- **Unordered**: the order of the points in a list has no influence on the 3D point cloud

Irregularity



Unstructured



Unordered

[41]

LiDAR Detection
Challenges

Due to the characteristics of point clouds, camera deep learning methods, especially CNNs, can’t be directly applied to point clouds. CNNs are based on the convolutional operation, which is performed on ordered and regular data on a structured grid.

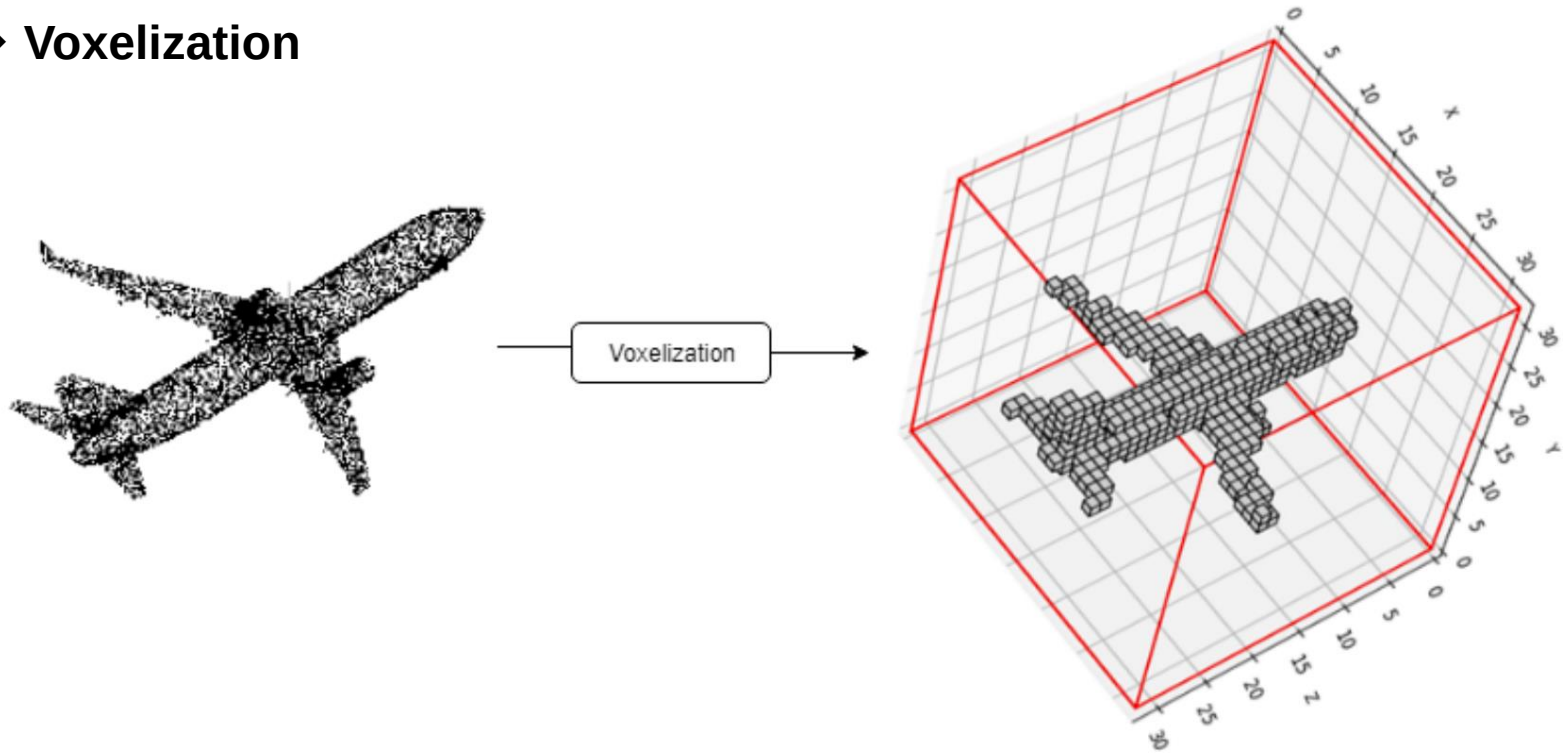
Two ways to tackle these challenges:

- **Structured grid-based learning**
- **Deep learning directly on raw point cloud**

LiDAR Detection
Structured grid-based learning

Conversion of the point cloud data into a structured form before applying deep learning methods to detect objects

→ **Voxelization**



[41]

Additional Slides

- Fixed-size voxels of size v_x, v_y, v_z in the 3D space X, Y, Z
- Resulting voxel grid of size $N_x = X / v_x$, $N_y = Y / v_y$, $N_z = Z / v_z$ (assumption: X, Y, Z are a multiple of v_x, v_y, v_z)
- Voxel representation: Binary (contains points or empty) or features (e.g. x_i, y_i, z_i, r_i (reflectance) for all points within voxel)
- Usually 3D convolutional layers (3D kernels) are used to extract features

LiDAR Detection
Structured grid-based learning

Example Pipeline (Vote3D Network):

(a) Input point cloud

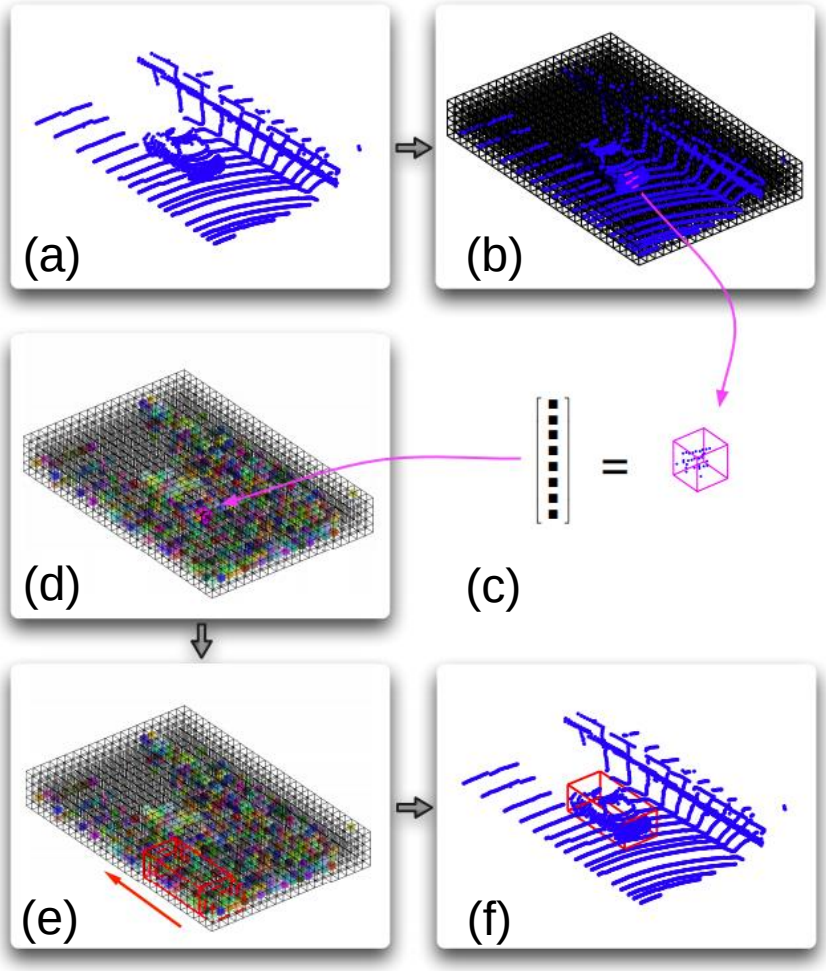
(b) Discretization into 3D grid

(c) Feature vector for each occupied cell

(d) Feature grid

(e) 3D kernel slides through feature grid in all 3 dimensions

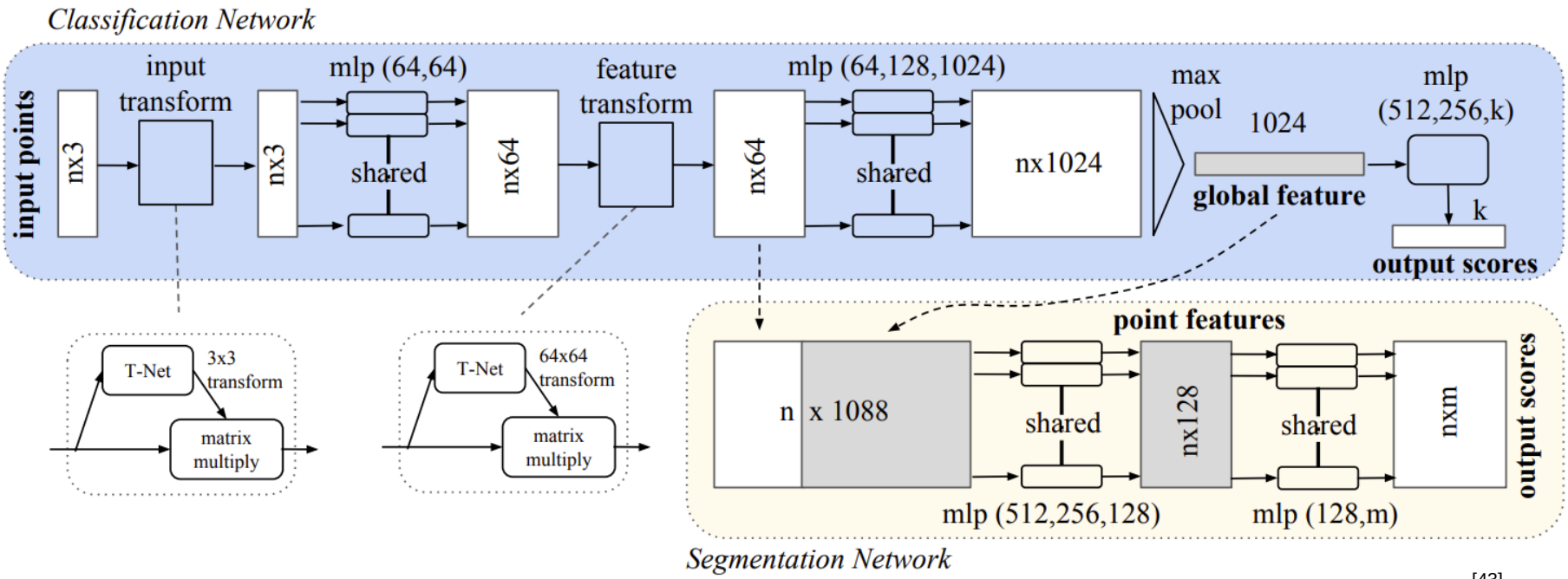
(f) Point cloud with detected object



[42]

LiDAR Detection
DL directly on raw point cloud

PointNet:



[43]

Additional Slides

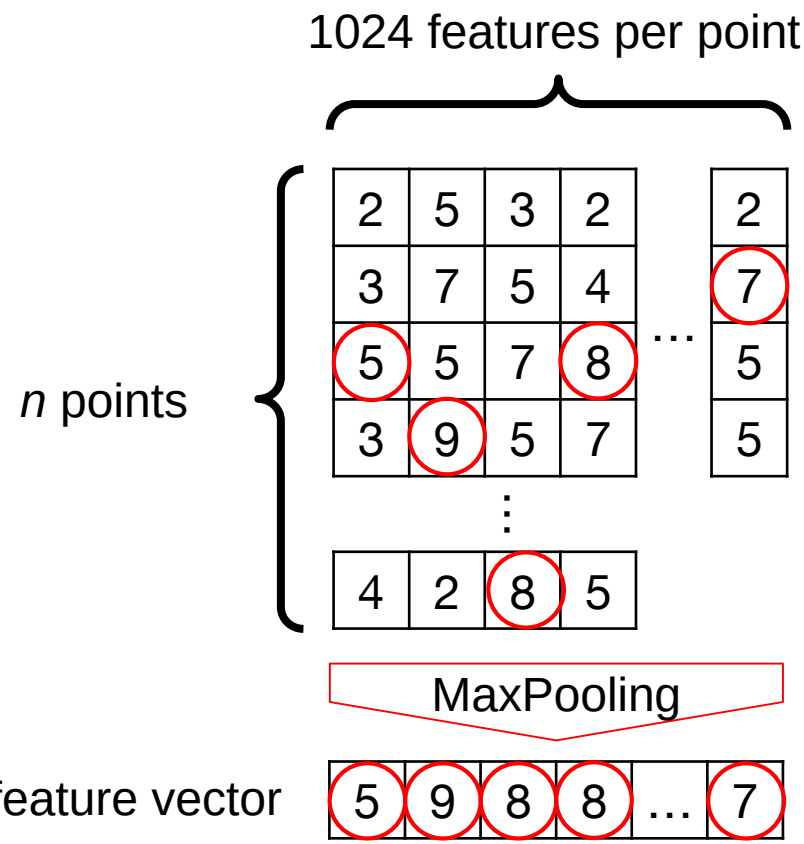
Most of the methods in this area rely on the seminal work of **PointNet:**

- Input is raw point cloud, *n* points with 3 dimensions (XYZ)
- Multilayer perceptrons (MLP) to transform feature dimension of each point from 3 to 1024
- Maxpooling (symmetric function, output is the same irrespective of input order) to generate a global 1024-dimensional feature vector
- The global feature vector can be used as input for classification, segmentation, and object detection

LiDAR Detection

DL directly on raw point cloud

PointNet
MaxPooling



Additional Slides

The main goal of PointNet is the assignment of an individual feature vector for each point in the point cloud, independent of the number of points. It can be used for object detection, but also for point cloud segmentation, classification, or many other tasks with point clouds.

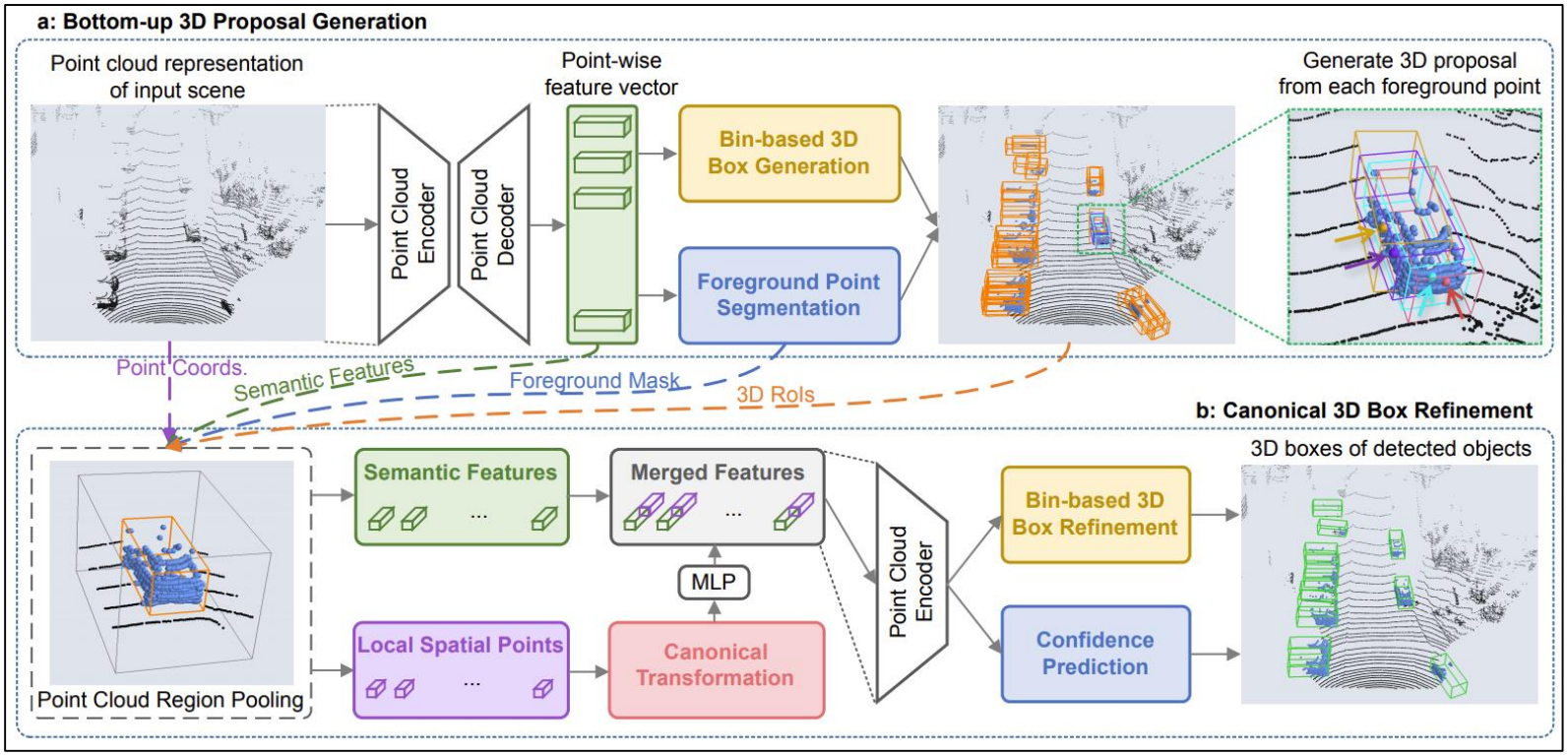
A 3D object detection network that makes use of PointNet (in particular it’s using PointNet++) is PointRCNN, which is build with 2 stages and is explained on the following slides.

LiDAR Detection

DL directly on raw point cloud

PointRCNN (2018) two-stages:

- 1. Stage: semantic segmentation + 3D BBox proposal
- 2. Stage: BBox refinement

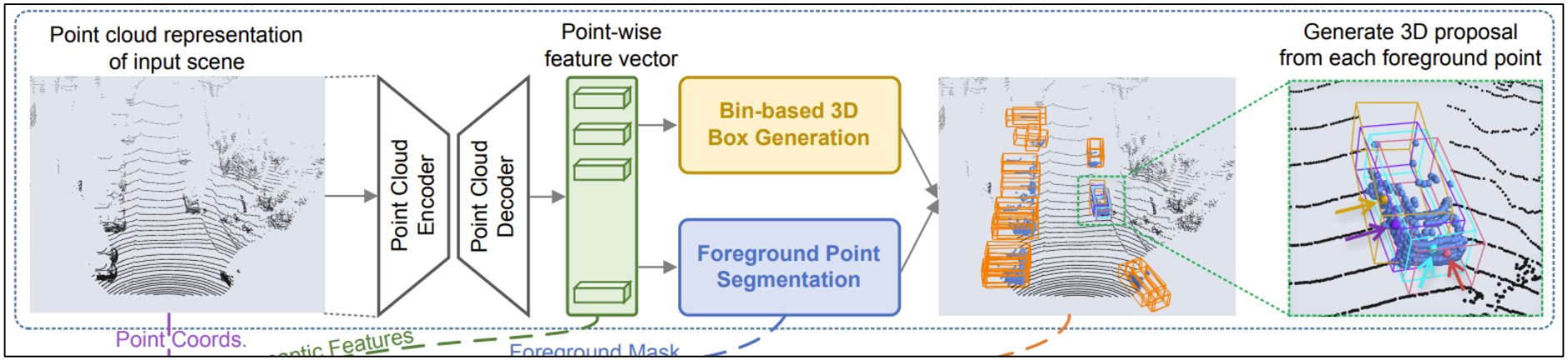


[44]

Additional Slides

PointRCNN **1. Stage**:

- PointNet++ is used to learn point-wise features
- Point-wise features vectors are used for semantic segmentation (separating foreground from background points)
- A 3D BBox with 7 DOF (x,y,z,h,w,l,θ) is proposed for each foreground point

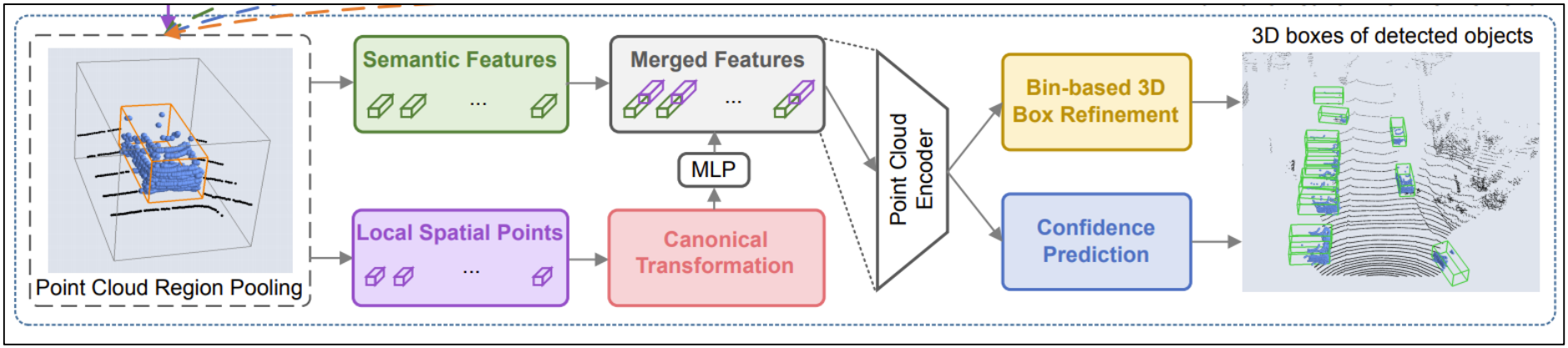


[44]

Additional Slides

PointRCNN **2. Stage**:

- Every proposed 3D BBox from stage 1 is slightly enlarged to encode additional information from its context (surrounding points)
- Each enlarged 3D BBoxes is transformed to its canonical coordinate system (BBox is axis aligned and BBox center is at (0,0,0))
- Merged features are concatenated for each enlarged BBox, including all inside points *xyz*, reflectance *r*, mask *m*, distance *d*, and point-wise feature vectors *f*
- Merged features are fed into another PointNet++ network, output is used for final BBox refinement and confidence classification



[44]

LiDAR Detection

Comparison LiDAR/Camera 3D Object Detection

KITTI Benchmark 3D Object Detection, Class Car, Moderate (09.12.2020)

Method	AP in %
PV-RCNN	81.43
CLOCs	80.67
SA-SSD	79.79
PointRCNN	75.64

LiDAR

SS3D	7.68
Pseudo-LiDAR	7.50

Camera

Additional Slides

Although the 3D projections of camera detection methods look good, their performance, especially at larger distances (>30 m), is inferior to Lidar detection methods. This is reflected in the KITTI benchmark: the best camera methods score an AP of ~10 %, whereas the best Lidar methods are ranked at ~80 % AP for class car (difficulty “moderate”, which means that the 2D box height of all ground truth boxes is at least 25px).

Sources

[40] https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cff
[41] Bello et al. (2020) – Review: Deep Learning on 3D Point Clouds
[42] Wang and Posner (2015) – Voting for Voting in Online Point Cloud Object Detection
[43] Qi et al. (2016) – PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation
[44] Shi et al. (2018) – PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud

Detection
Prof. Dr. Markus Lienkamp

Sebastian Huch, M. Sc.

Agenda

1. General Definitions
2. Camera Detection
3. LiDAR Detection
4. **Radar Detection**
5. Sensor Fusion
6. Datasets
7. Summary

Radar Detection
Motivation

- Only sensor that directly measures the (radial) velocity of detected objects using Doppler effect
- Robust to weather and lighting compared to camera/LiDAR
- Point clouds can be processed in a similar way to LiDAR point clouds, but are sparser
- Widely used for applications such as adaptive cruise control (ACC)

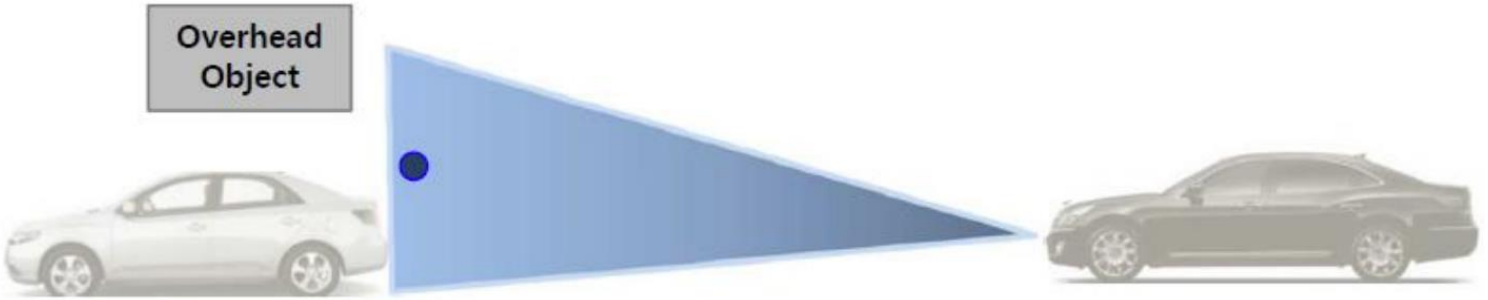
[45]

Radar Detection
Challenges

Low vertical resolution

→ Radar is good for detection of moving objects, but static objects (or objects moving perpendicular to the radar beams) and objects with low speeds are usually filtered out to avoid false positives (e.g. overpass)

[62]



[62]

Radar Detection

Challenges

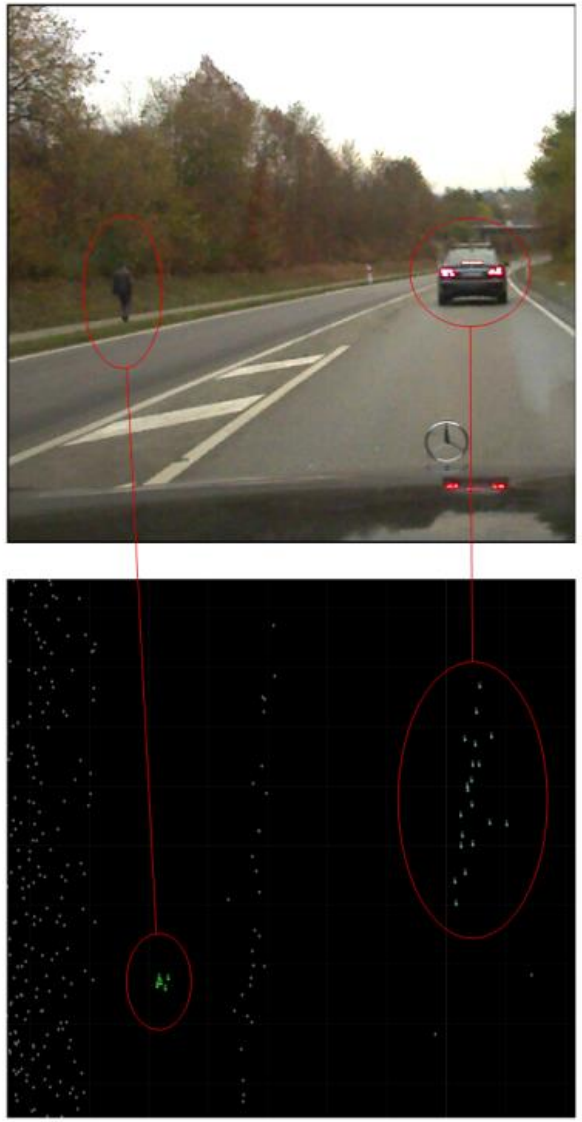
[46]

Nobody was injured!

Radar Detection
Methods

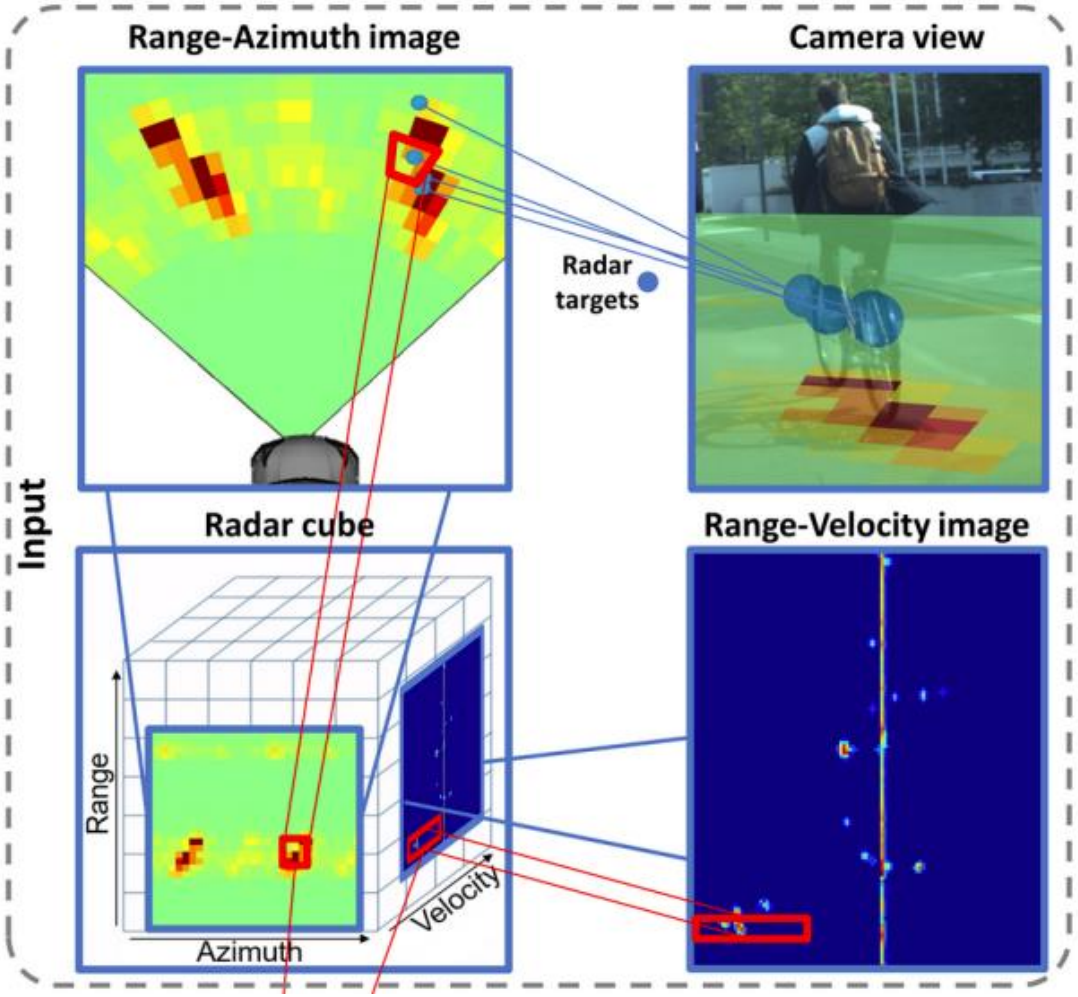
Radar outputs are usually point clouds of reflections (**radar targets**), that include the range r, the azimuth α, the radar cross section RCS (reflectivity) and the object's radial speed v_r.

1. Clustering of radar targets
2. Feature extraction for each cluster (e.g. statistical features such as min/max/mean/std-dev for all measurements r, α, RCS, v_r)
3. Classification using the extracted features



[47]

Radar Detection
Methods using Deep Learning



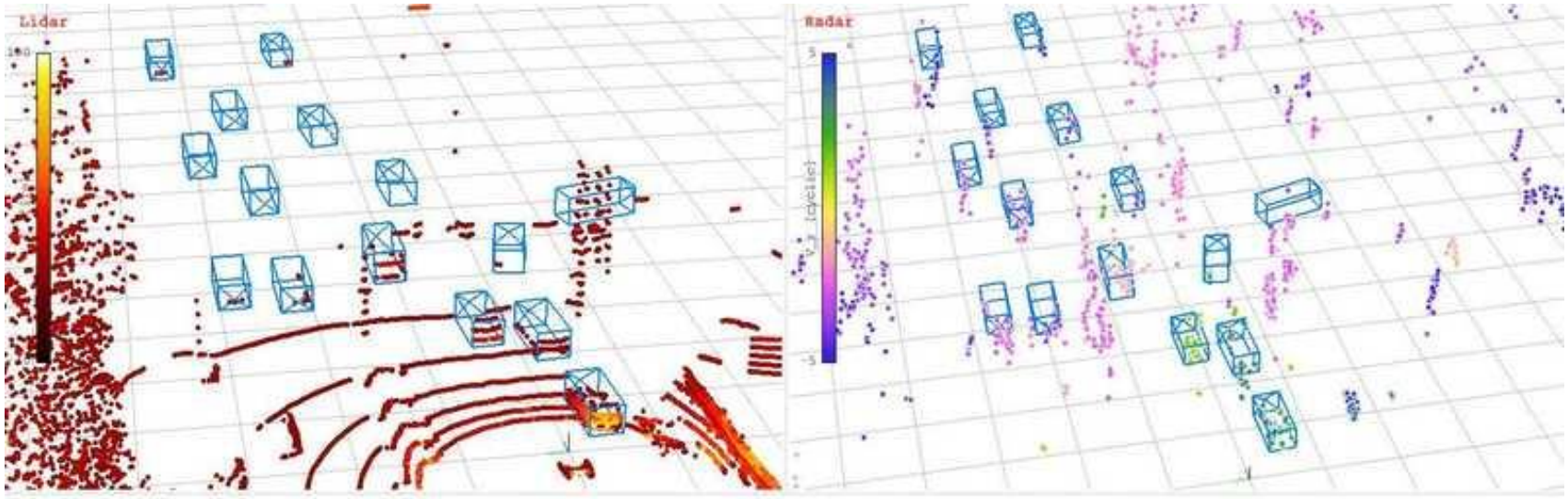
[48]

Radar Cube – 3D data matrix with range, azimuth and velocity axis

→ Radar Cube enables CNNs for feature extraction

Radar Detection
Outlook

High-resolution radar (e.g. company ASTYX) are currently developed. These overcome the biggest weakness of current radar sensors and can bridge the resolution gap to LiDAR sensors, while at the same time keeping the robust characteristics of the radar sensor.



[58]

Sources

[45] https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/automatic-emergency-braking/front-radar-sensor/

[46] https://www.youtube.com/watch?v=cz2O6d3HVyg&ab_channel=gm2839

[47] Scheiner et al. (2019) – Radar-based Feature Design and Multiclass Classification for Road User Recognition

[48] Palffy et al. (2020) – CNN based Road User Detection using the 3D Radar Cube

[58] https://www.astyx.net/entwicklung/astyx-hires2019-datensatz.html

[62] https://in.bgu.ac.il/en/engn/ece/radar/Site%20Assets/Pages/Presentations/Radar2015_Bilik.pdf

Detection
Prof. Dr. Markus Lienkamp

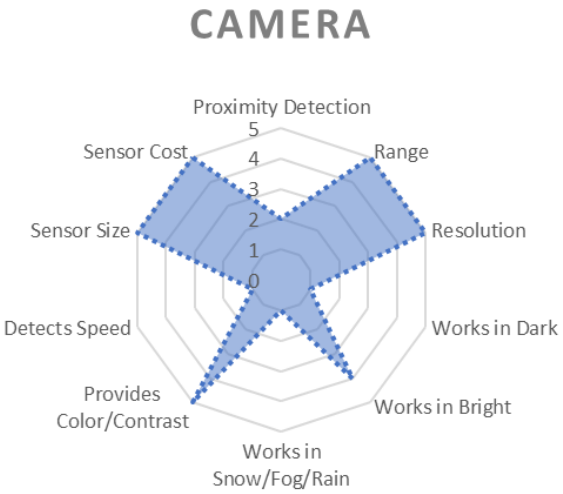
Sebastian Huch, M. Sc.

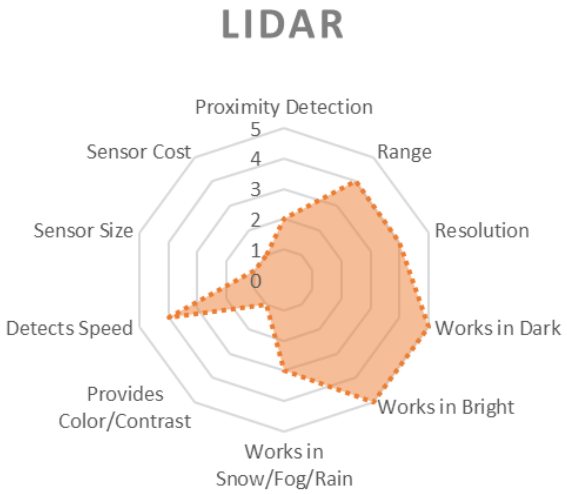
Agenda

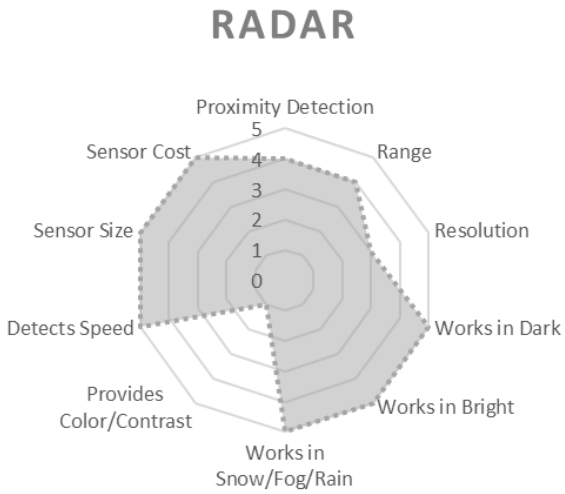
1. General Definitions
2. Camera Detection
3. LiDAR Detection
4. Radar Detection
5. **Sensor Fusion**
6. Datasets
7. Summary

Sensor Fusion
Motivation

- Redundancy
- Reduction of uncertainty
- Sensors are complementing each other:



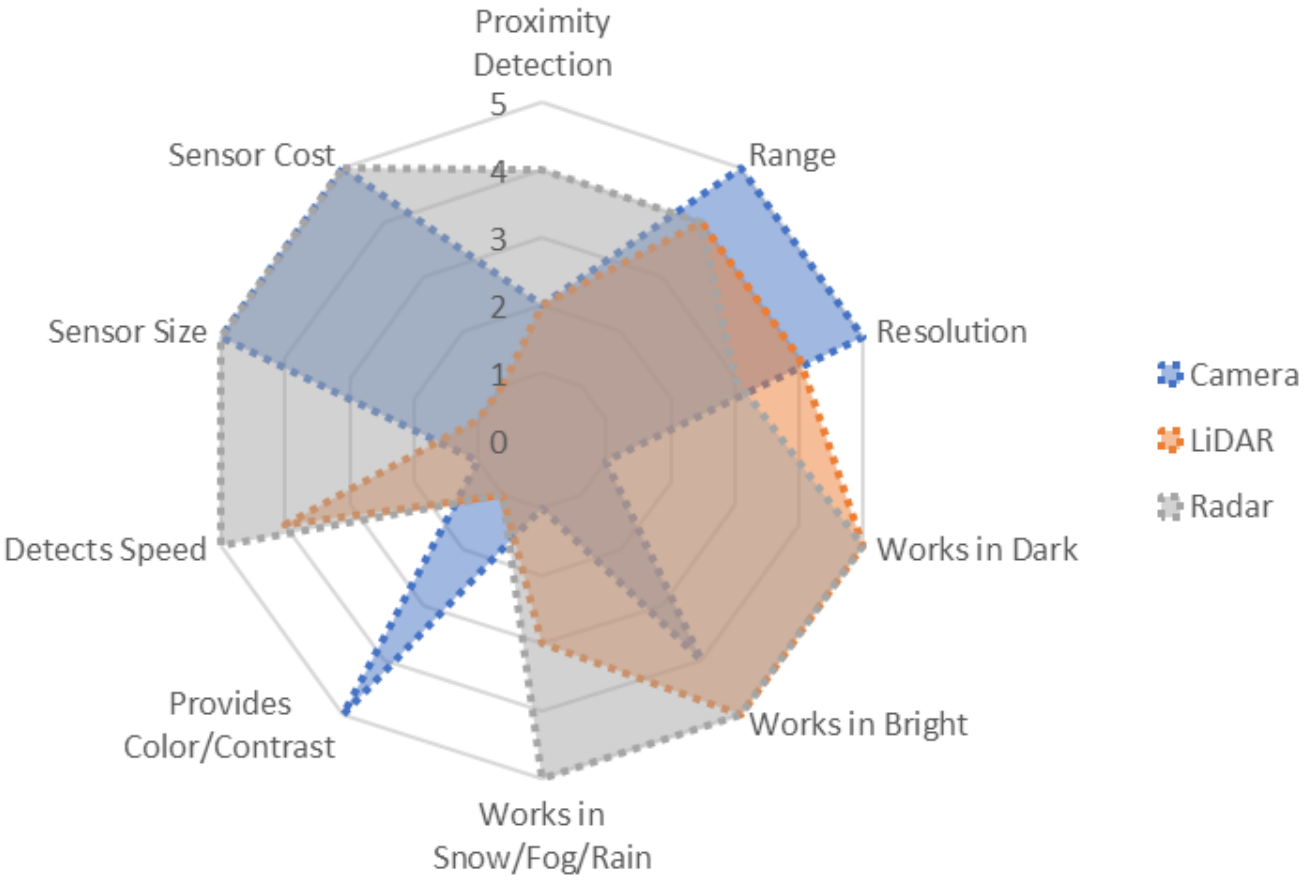




[49]

Radar filters static targets!

Sensor Fusion
Motivation

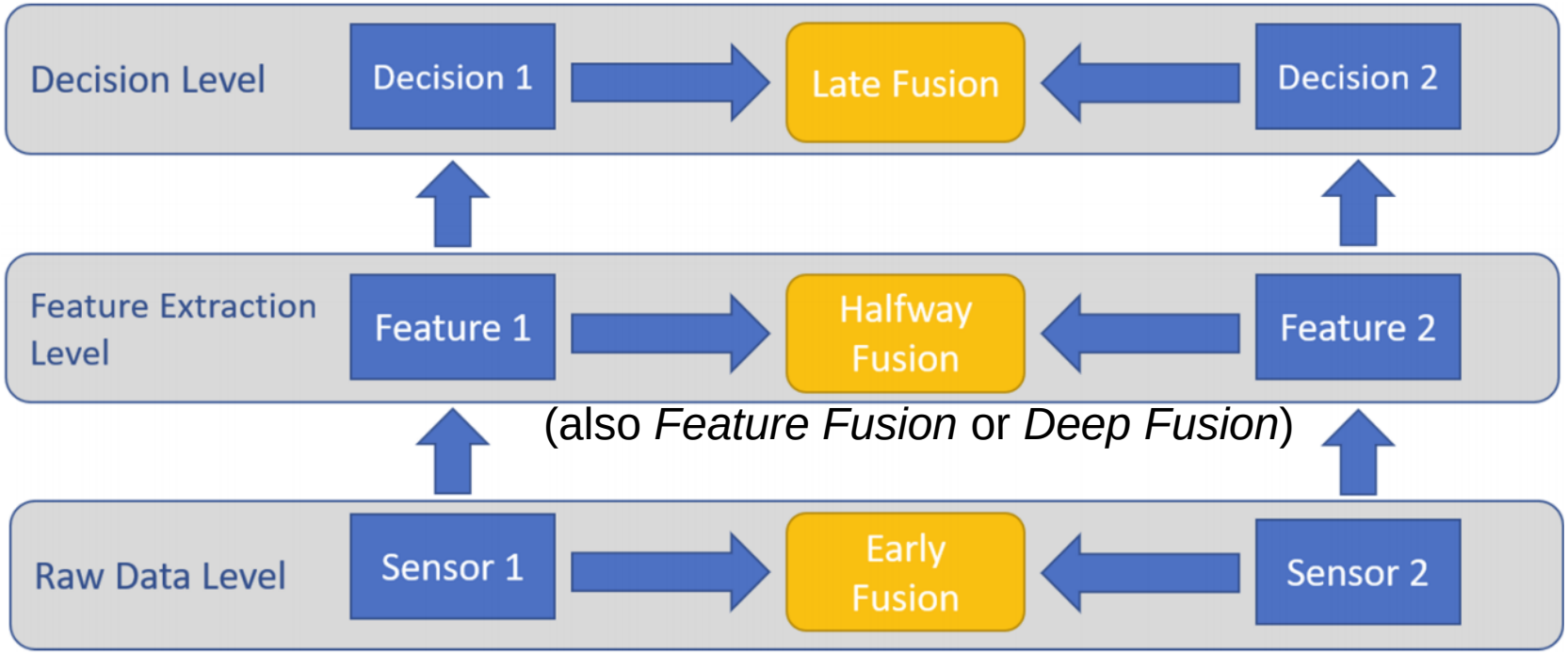


[49]

Additional Slides

Sensor fusion combines the strengths of the different sensor modalities and eliminates their individual weaknesses. Although it looks as the radar sensor has the best overall performance, it lacks in detection of static objects (and also objects moving in rectangular direction to the emitted radar beams).

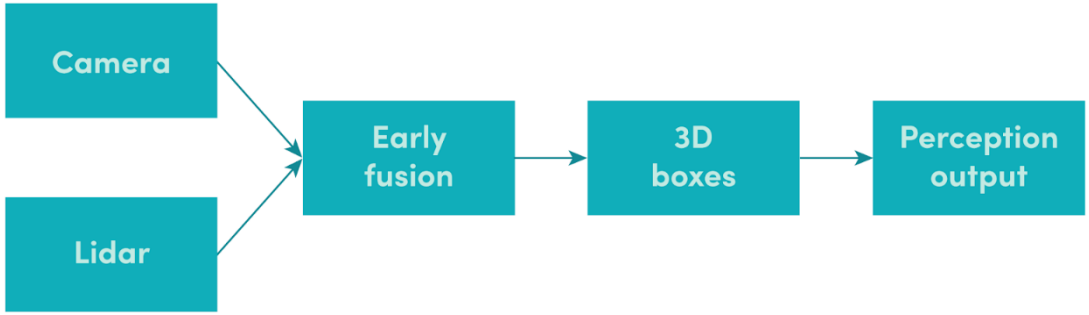
Sensor Fusion
Concepts



[50]

Sensor Fusion
Early Fusion

- Fusion of raw sensor data by joint processing
- E.g. projection of LiDAR point clouds onto 2D images



[51]

Pros:

- Learn and infer from multiple sensor modalities at the same time (directly learn strengths and weaknesses of the modalities)

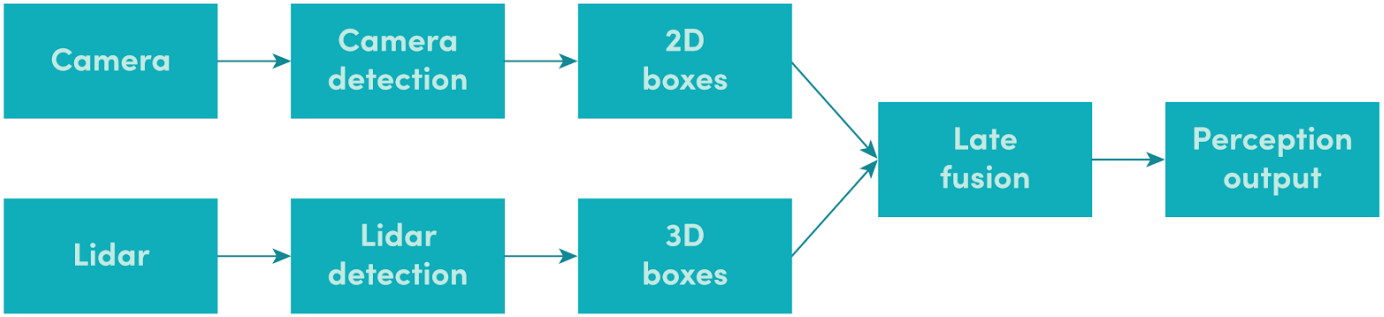
Cons:

- More complex than late fusion
- Dependent on all modalities

Sensor Fusion

Late Fusion

- Independent sensor processing
- Fusion occurs at last stage



[51]

Pros:

- Does not require all modalities be available, can rely on predictions of single modality
- Usage of specialized modules (camera detection, LiDAR detection, …) possible
- Late fusion of bounding boxes using well-known methods like Kalman Filters allows probabilistic reasoning (estimation of uncertainties)

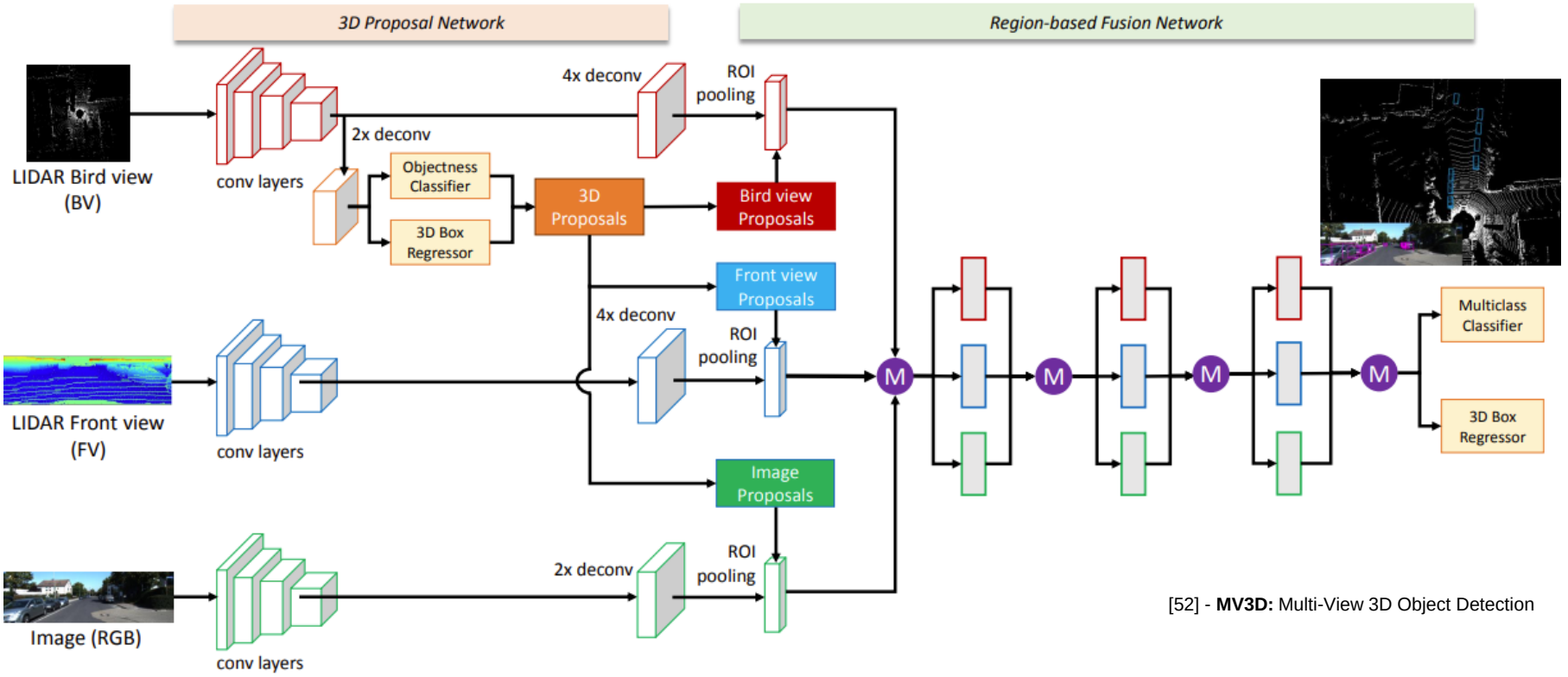
Cons:

- Which modality to trust when their outputs are contradictory, e.g. because of sensor malfunction?

Additional Slides

Feature Fusion with Deep Learning

- Mix of modalities hierarchically in neural network layers
- Features from different modalities can interact over layers



[52] - **MV3D:** Multi-View 3D Object Detection

Sources

[49] https://deeplearning.mit.edu/
[50] Fayyad et al. (2020) – Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review
[51] https://medium.com/lyftself-driving/leveraging-early-sensor-fusion-for-safer-autonomous-vehicles-36c9f58ddd75
[52] Chen et al. (2016) – Multi-View 3D Object Detection Network for Autonomous Driving

Detection
Prof. Dr. Markus Lienkamp

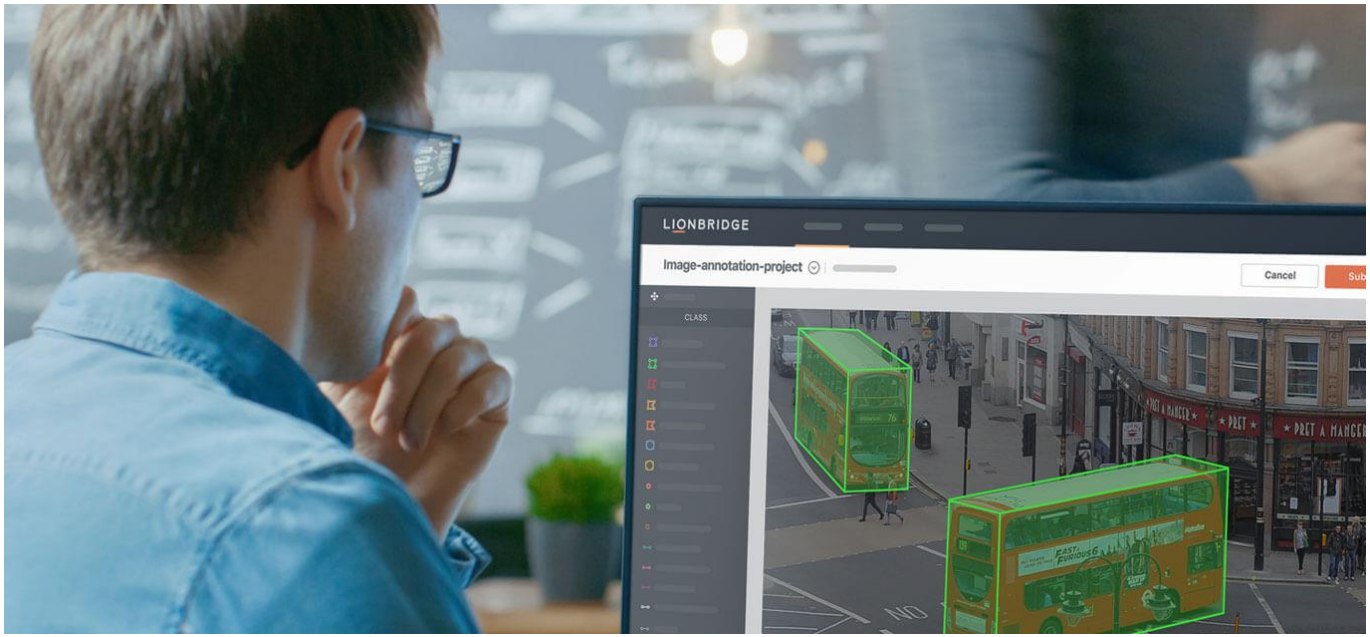
Sebastian Huch, M. Sc.

Agenda

1. General Definitions
2. Camera Detection
3. LiDAR Detection
4. Radar Detection
5. Sensor Fusion
6. **Datasets**
7. Summary

Datasets
Motivation

- Supervised learning
- Labelled datasets for training and testing
- Datasets for benchmarking
- Quantity *and* quality



[65]

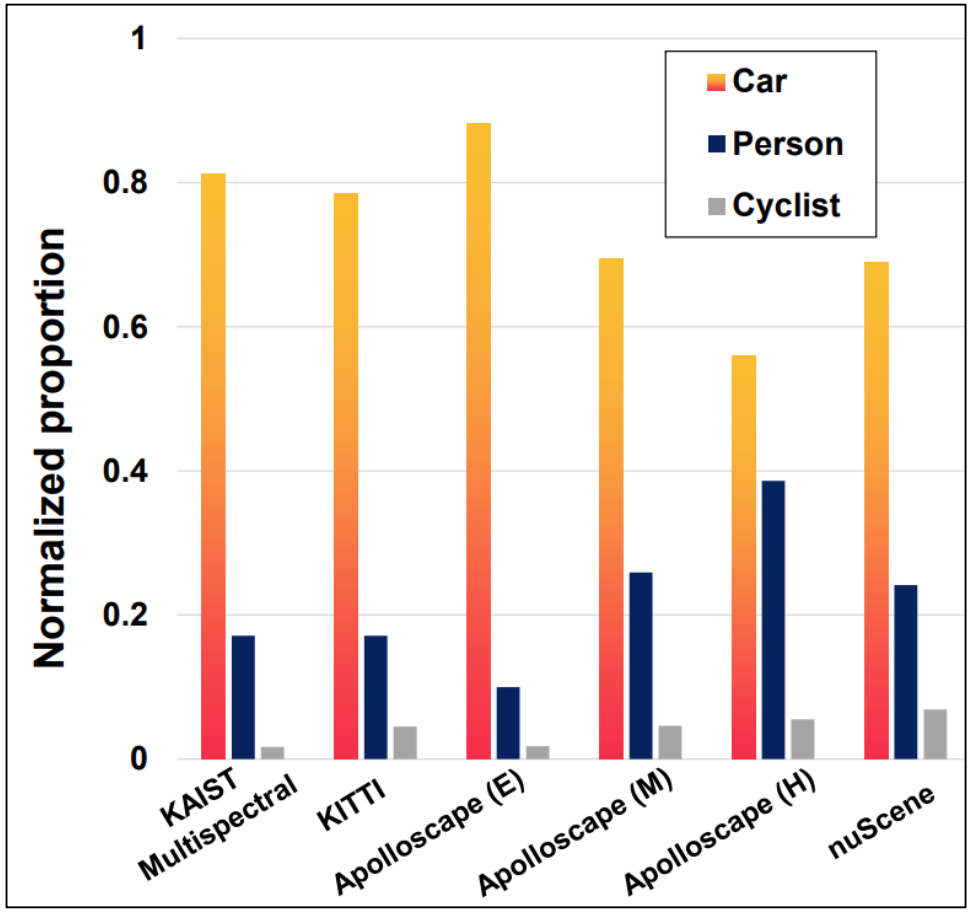
Additional Slides

Most deep learning detection approaches are based on supervised learning. For testing and training of these neural networks, large datasets with labelled ground truth are required. Furthermore, datasets can be used for quantitative evaluations of different approaches and provide insight about their capabilities and limitations, and also their performance compared to other approaches (benchmark). Not only number of annotated frames important, but also their quality (e.g. diversity of scenes to reduce bias) plays an important role for the generalization ability of neural networks.

Datasets
Challenges

Class imbalance:

- Class distributions are highly imbalanced
- Class “Car” is overrepresented in AD datasets



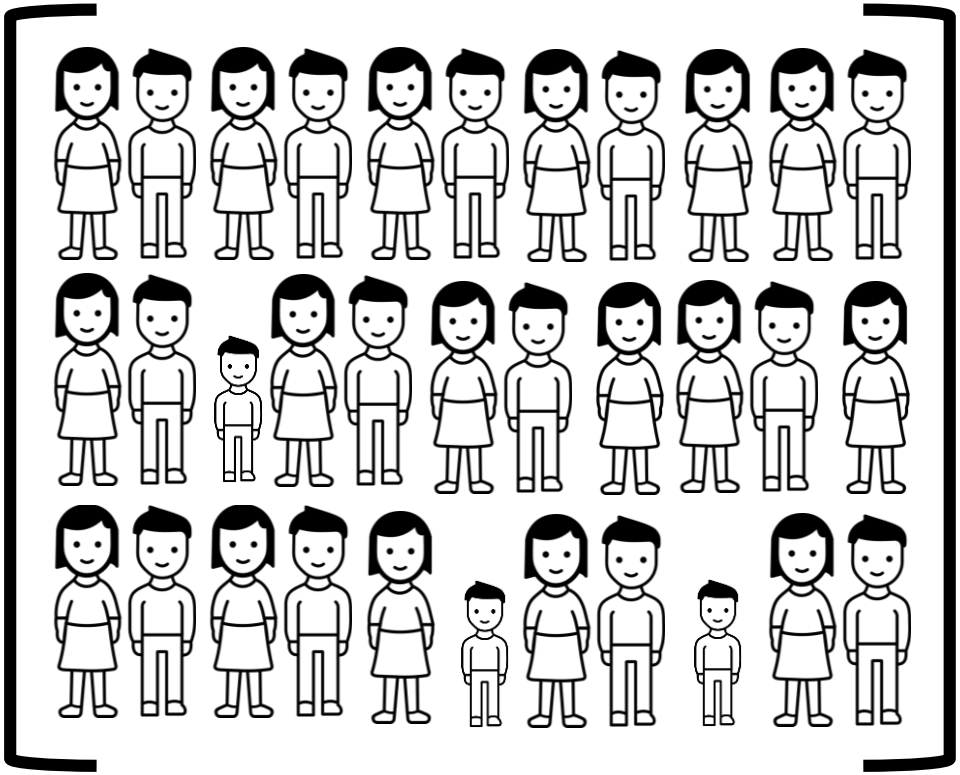
[54]

Datasets
Challenges

Selection bias:

- Low intra-class diversity/high intra-class imbalance
- E.g. class “Person” shows mainly adults and only few children

Class **Pedestrian**



Datasets
Challenges

Capture bias:

- Representation of objects: Objects often shown in the same representation
- E.g. cars are heading in the same direction
- Experiment: Google Image search “car” → most of the cars are shown in an angular view from the front, barely no images showing the vehicle rear

[66]

Datasets

Synthetic Datasets

- Real-world datasets are costly to obtain and accurate labelling is a time-consuming process
- Simulations can be used as an alternative to generate synthetic datasets

Pros:

- “Unlimited” data generation
- Different sensor configurations available
- Automatically annotated (up to pixel-level)
- Varying conditions (weather, lighting, vehicle, traffic)
- Diversity of traffic actors (shapes, sizes, colors, behaviors)
- Rapid scenario construction

Cons:

- Creation of photorealistic virtual worlds is time-consuming and expensive
- Real-world generalization of models trained with synthetic data?

Datasets

Dataset	Realism	Diversity	Autonomous Driving	Evaluation Server	Stereo	Reconstruction	Optical Flow	Object Detection	Traffic Sign Detection	Semantic Segmentation	Road Detection	Lane Detection	Tracking
Middlebury [581]	+	--		✔	XS	XS	XS						
EPFL Multi-View [627]	++	+		✔		XS							
DTU MVS [319]	+	-				S							
ETH3D [591]	++	+		✔	S	S							
Tanks and Temples [351]	++	+		✔		S							
SlowFlow [316]	++	++					S						
HCI Benchmark [357]	++	+	✔	✔			M						
MPI Sintel [92]	O	+		✔	M		M						
Flying Chairs [174]	--	--					L						
Flying Things [450]	-	O	(✔)		L		L						
ImageNet [160]	++	++						XL		XL			
PASCAL VOC [194]	++	++						XL		XL			
Microsoft Coco [420]	++	++						XL		XL			
Cityscapes [133]	++	+	✔	✔				L		L			
EuroCity Persons Dataset [68]	++	++	✔	✔				L					
Mapillary [487]	++	++	✔							L			
ApolloScape [307]	++	+	✔					L		XL		XL	XL
NuScenes [93]	++	+	✔					XL		XL			
Berkeley DeepDrive [755]	++	+	✔					XL		XL	XL	XL	
German Traffic Sign Recognition Benchmark [623]	++	+	✔	✔				XL	L	XL	XL	XL	
German Traffic Sign Detection Benchmark [299]	++	+	✔	✔				XL	M	XL	XL	XL	
Tsinghua-Tencent 100K [793]	++	+	✔					XL	XL	XL	XL	XL	
SYNTHIA [558]	O	+	✔							XL			
Playing for Data [551]	+	+	✔							L			
Playing for Benchmarks [550]	+	+	✔	✔			XL	XL		XL			XL
Caltech Lanes Dataset [8]	++	+	✔									M	
VPGNet Dataset [394]	++	+	✔									L	
MOTChallenge [389]	++	+		✔									M
Caltech Pedestrian Detection [172]	++	+	✔										XL
KITTI [238]	++	+	✔	✔	S	S	S	M		S	S	S	M
VirtualKITTI [221]	O	+	✔	✔	S	S	L	L		L			L

[56]

Additional Slides

Some popular multimodal datasets with 3D annotations for autonomous driving:

Real-world:

KITTI (*2012/2015*, 7481 training and 7518 test images + LiDAR point clouds, labelled 2D/3D BBoxes)

Argoverse (*2019*, 13121 training and 5015 test images + LiDAR point clouds, labelled 3D BBoxes)

nuScenes (*2019*, 1.4M frames for camera+radar, 390k frames for LiDAR, labelled 3D BBoxes)

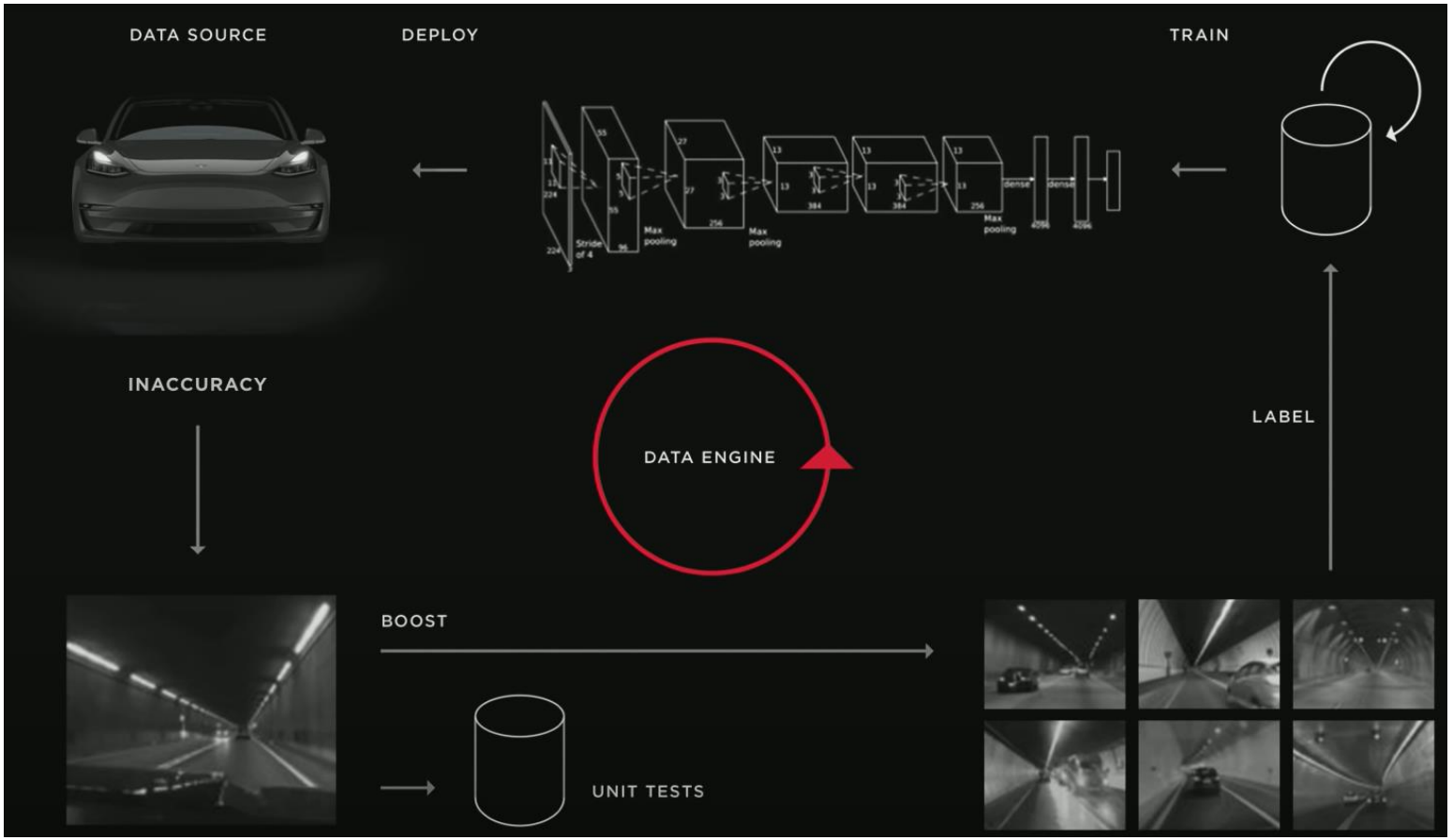
BLVD (*2019*, 123k frames for camera+LiDAR, labelled 3D BBoxes)

Synthetic:

VirtualKITTI/VirtualKITTI2 (*2016/2020*, based on original KITTI, using Unity game engine)

Datasets

Data sourcing from a fleet

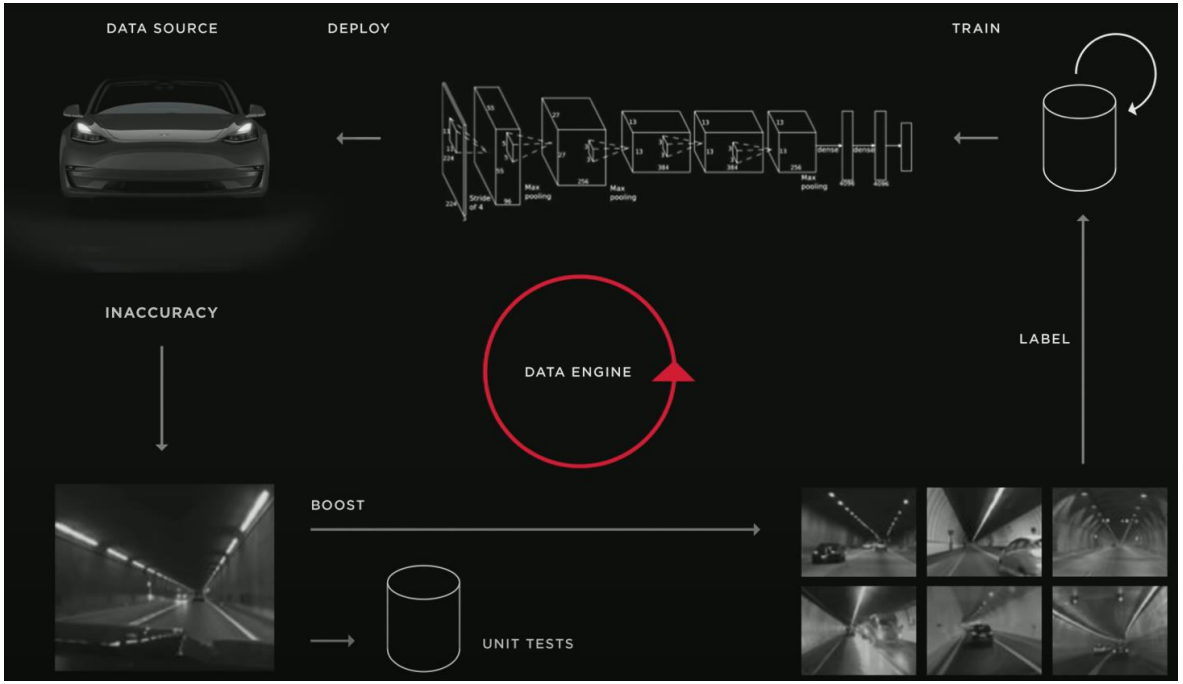


[57]

Additional Slides

Datasets don’t cover all rare edge cases. Some cases are so rare, we can’t even image that they exist in real world. Therefore, additional data sourced from a fleet in real-world is necessary to improve the performance on those edge cases.

1. Deploy initially trained network in fleet (could run in the background)
2. Include mechanisms that notice inaccuracies of the network (false positives or false negatives)
3. Use the frames with misbehaviour in future unit tests
4. Source similar frames from the fleet
5. Label sourced frames and include them in dataset
6. Train network again
7. Repeat process



[57]

Sources

[54] Feng et al. (2019) – Deep Multi-modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges

[56] Janai et al. (2019) – Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art

[57] https://www.youtube.com/watch?v=Ucp0TTmvqOE&ab_channel=Tesla

[65] https://lionbridge.ai/articles/5-approaches-to-data-labeling-for-machine-learning-projects/

[66] https://www.wsj.com/articles/one-driver-can-prevent-a-traffic-jam-1476204858

Detection
Prof. Dr. Markus Lienkamp

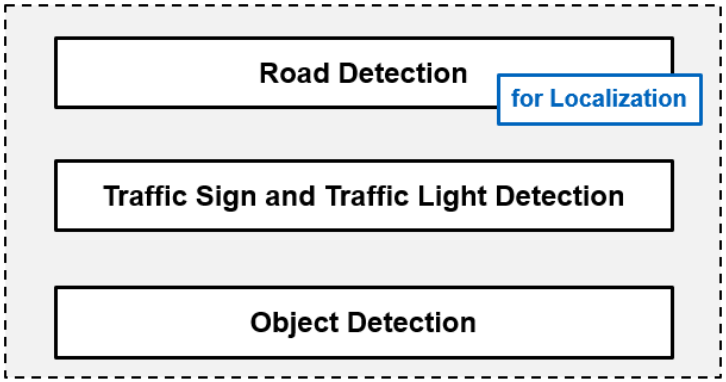
Sebastian Huch, M. Sc.

Agenda

1. General Definitions
2. Camera Detection
3. LiDAR Detection
4. Radar Detection
5. Sensor Fusion
6. Datasets
7. **Summary**

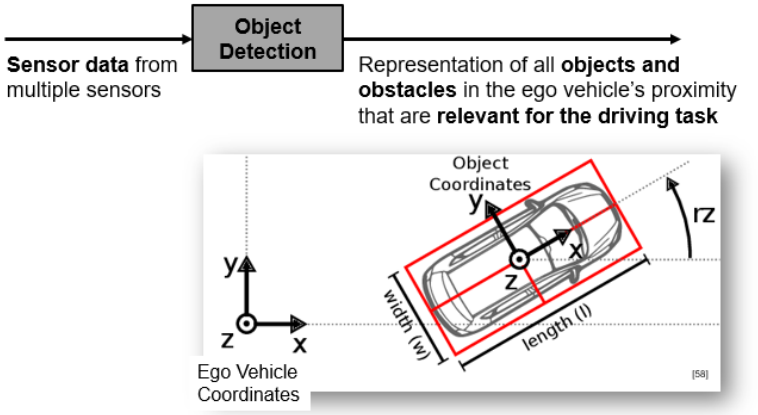
Summary

General Definitions
3 Detection Tasks



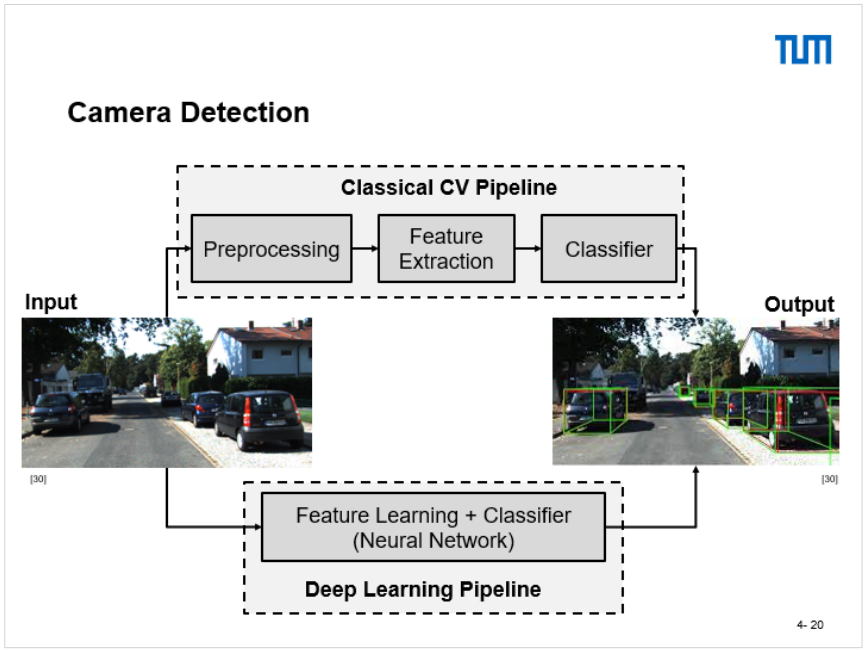
4- 6

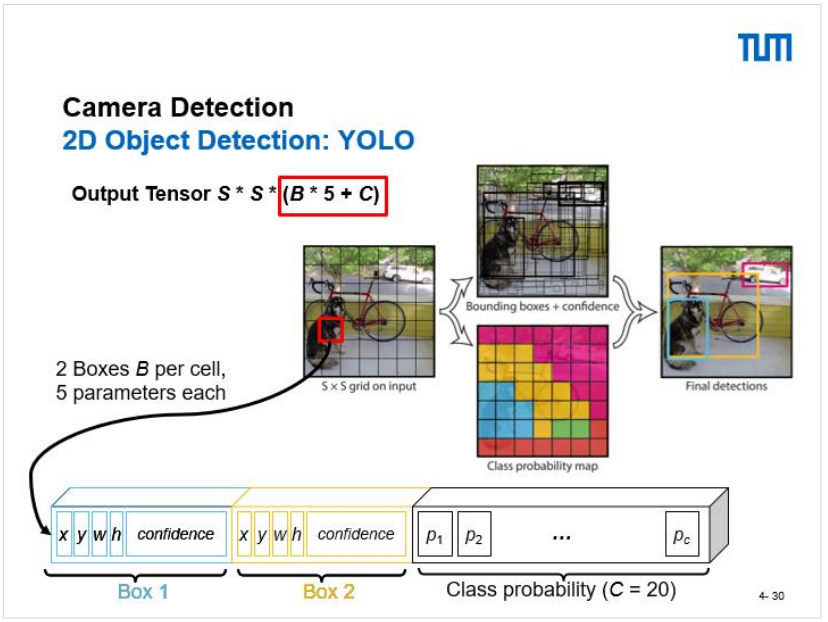
General Definitions
Interfaces



4- 12

Summary


Camera Detection
Classical CV Pipeline
Preprocessing
Feature Extraction
Classifier
Input
Output
[30]
[30]
Feature Learning + Classifier (Neural Network)
Deep Learning Pipeline
4- 20


Camera Detection
2D Object Detection: YOLO
Output Tensor S * S * (B * 5 + C)
Bounding boxes + confidence
S x S grid on input
Final detections
2 Boxes B per cell,
5 parameters each
Class probability map
x y w h confidence
x y w h confidence
p1 p2 ... pc
Box 1
Box 2
Class probability (C = 20)
4- 30

Summary

Camera Detection

Monocular 3D object detection (mono3DOD)

mono3DOD Groups

- Representation Transformation
- Keypoints and Shape
- Distance via 2D/3D Constraint
- Direct Generation of 3D Proposals

[27]

4- 51

Camera Detection

Traffic Light and Sign Detection

Challenges

[37]

[38]

[39]

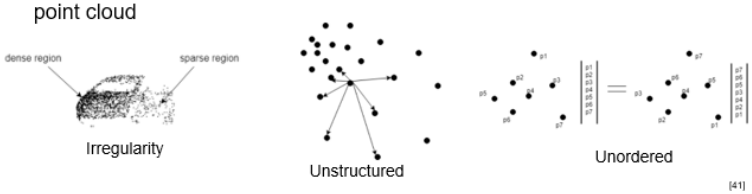
4- 65

Summary

LiDAR Detection
Challenges

LiDAR sensor outputs are point clouds, which have the following characteristics:

- **Irregularity**: points are not even sampled around the sensor; some regions have dense points while others have sparse points
- **Unstructured**: points are not on a fixed grid, distance between neighboring points varies (compared to equidistant pixels in 2D images)
- **Unordered**: the order of the points in a list has no influence on the 3D point cloud



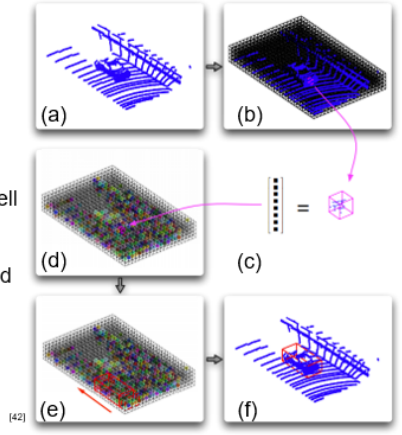
[41]

4- 70

LiDAR Detection
Structured grid-based learning

Example Pipeline (Vote3D Network):

(a) Input point cloud
(b) Discretization into 3D grid
(c) Feature vector for each occupied cell
(d) Feature grid
(e) 3D kernel slides through feature grid in all 3 dimensions
(f) Point cloud with detected object



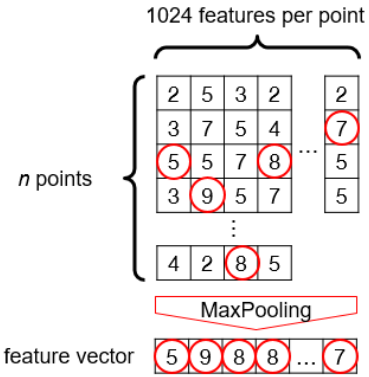
[42]

4- 74

Summary

LiDAR Detection
DL directly on raw point cloud

PointNet
MaxPooling



4- 77

LiDAR Detection
Comparison LiDAR/Camera 3D Object Detection

KITTI Benchmark 3D Object Detection, Class Car, Moderate (09.12.2020)

Method	AP in %	
PV-RCNN	81.43	LiDAR
CLOCs	80.67	
SA-SSD	79.79	
PointRCNN	75.64	
SS3D	7.68	Camera
Pseudo-LiDAR	7.50	

4- 82

Summary

Radar Detection
Challenges

Low vertical resolution

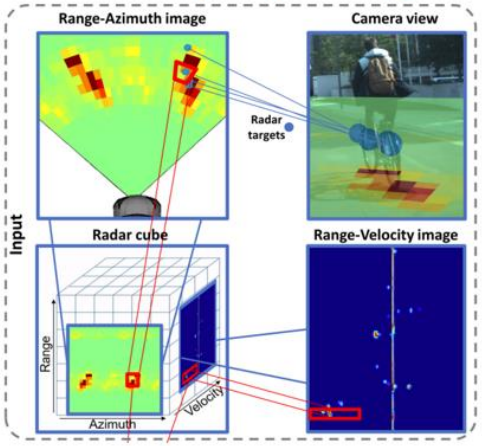
→ Radar is good for detection of moving objects, but static objects (or objects moving perpendicular to the radar beams) and objects with low speeds are usually filtered out to avoid false positives (e.g. overpass)

[62]

[62]

4- 87

Radar Detection
Methods using Deep Learning



[48]

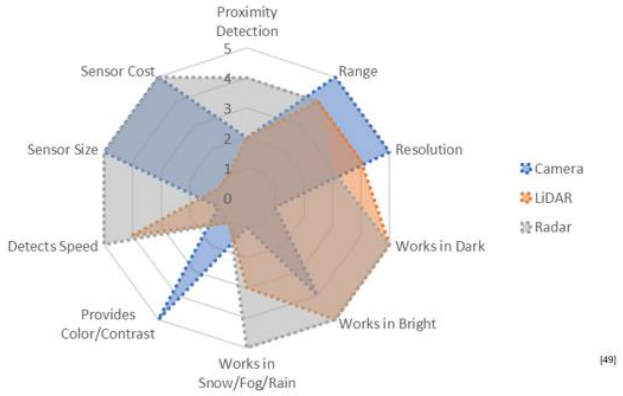
Radar Cube – 3D data matrix with range, azimuth and velocity axis

→ Radar Cube enables CNNs for feature extraction

4- 90

Summary

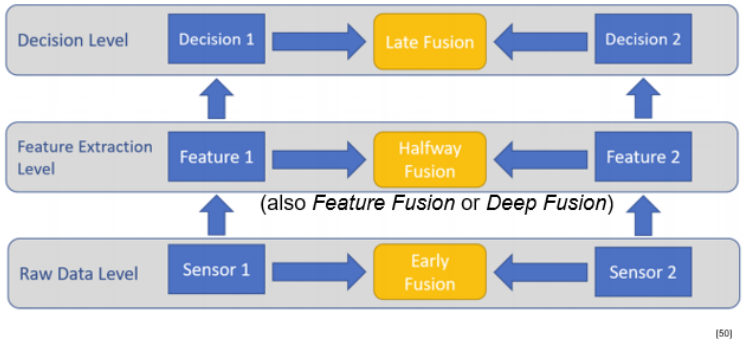
Sensor Fusion
Motivation



[49]

4- 95

Sensor Fusion
Concepts



[50]

4- 97

Summary

Datasets
Challenges

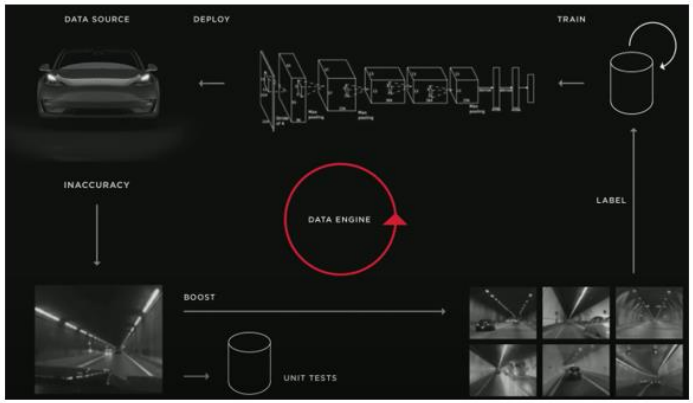
Capture bias:

- Representation of objects: Objects often shown in the same representation
- E.g. cars are heading in the same direction
- Experiment: Google Image search “car” → most of the cars are shown in an angular view from the front, barely no images showing the vehicle rear

[66]

4- 107

Datasets
Data sourcing from a fleet



[57]

4- 111

Additional Slides

Summary – What did we learn today

- We know that detection is a crucial task in autonomous driving.
- We learned that the detection module is part of the **perception** and uses **sensor data to perceive the AV's environment**.
- We learned the 3 main tasks of detection: **object detection**, **road detection**, and **traffic light/sign detection**.
- We know the multiple challenges faced by detection.
- We learned the differences between **classification**, **localization**, **detection**, and **segmentation**.
- We learned the two different approaches of camera detection: **classical CV pipelines** or **Deep Learning pipelines**.
- We understand the **motivation of using Deep Learning approaches** for camera detection.
- We understand the problem to lift 2D detection to 3D space and learned the 4 different groups of monocular 3D object detection (**Representation Transformation, Keypoints and Shape, Distance via 2D/3D Constraint, Direct Generation of 3D Proposals**).
- We have seen approaches to solve the other detection tasks, such as **traffic light/sign detection**.
- We understand the motivation but also challenges of using **LiDAR** for detection and know how to solve approach these challenges (**Structured grid-based learning or DL directly on raw point cloud**).
- We understand the concepts of **voxelization** and **PointNet**.
- We learned the basics of **Radar** detection.
- We understand the strengths/weaknesses of each sensor modality and the motivation for **sensor fusion and its concepts**.
- We understand the necessity and challenges of high-quality **datasets** for deep learning methods and the different characteristics of datasets.