

Self-Driving Cars

Lecture 3 – Direct Perception

Prof. Dr.-Ing. Andreas Geiger

Autonomous Vision Group

University of Tübingen / MPI-IS

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN



e l l i s
European Laboratory for Learning and Intelligent Systems

Agenda

3.1 Direct Perception

3.2 Conditional Affordance Learning

3.3 Visual Abstractions

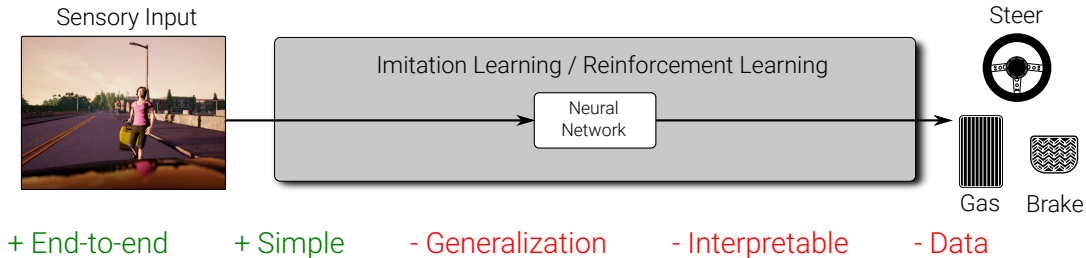
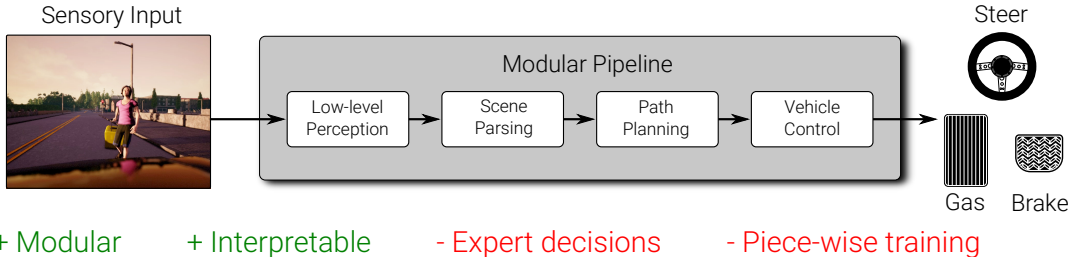
3.4 Driving Policy Transfer

3.5 Online vs. Offline Evaluation

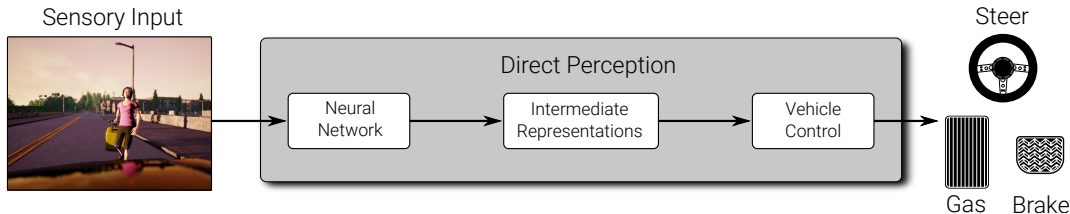
3.1

Direct Perception

Approaches to Self-Driving



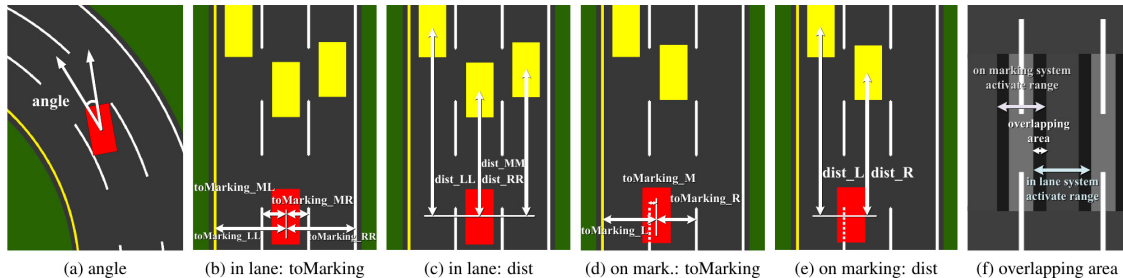
Direct Perception



Idea of Direct Perception:

- ▶ **Hybrid model** between imitation learning and modular pipelines
- ▶ Learn to predict interpretable **low-dimensional intermediate representation**
- ▶ **Decouple** perception from planning and control
- ▶ Allows to exploit **classical controllers** or **learned controllers** (or hybrids)

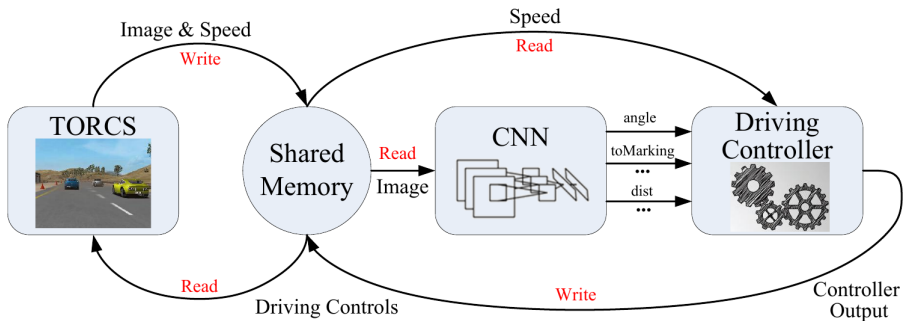
Direct Perception for Autonomous Driving



Affordances:

- **Attributes** of the environment which **limit space of actions** [Gibson, 1966]
- In this case: 13 affordances

Overview



- **TORCS Simulator:** Open source car racing game simulator
- Network: **AlexNet** (5 conv layers, 4 fully conn. layers), 13 output neurons
- Training: Affordance indicators trained with ℓ_2 loss

Affordance Indicators and State Machine

always:

1) angle: angle between the car's heading and the tangent of the road

“in lane system”, when driving in the lane:

2) toMarking_LL: distance to the left lane marking of the left lane

3) toMarking_ML: distance to the left lane marking of the current lane

4) toMarking_MR: distance to the right lane marking of the current lane

5) toMarking_RR: distance to the right lane marking of the right lane

6) dist_LL: distance to the preceding car in the left lane

7) dist_MM: distance to the preceding car in the current lane

8) dist_RR: distance to the preceding car in the right lane

“on marking system”, when driving on the lane marking:

9) toMarking_L: distance to the left lane marking

10) toMarking_M: distance to the central lane marking

11) toMarking_R: distance to the right lane marking

12) dist_L: distance to the preceding car in the left lane

13) dist_R: distance to the preceding car in the right lane

Figure 4: **Complete list of affordance indicators in our direct perception representation.**

while (in autonomous driving mode)

ConvNet outputs affordance indicators

check availability of both the left and right lanes

if (approaching the preceding car in the same lane)

if (left lane exists **and** available **and** lane changing allowable)

left lane changing decision made

else if (right lane exists **and** available **and** lane changing allowable)

right lane changing decision made

else

slow down decision made

if (normal driving)

center_line = center line of current lane

else if (left/right lane changing)

center_line = center line of objective lane

compute steering command

compute *desired_speed*

compute acceleration/brake command based on *desired_speed*

Figure 5: **Controller logic.**

Controller

Steering controller:

$$s = \theta_1(\alpha - d_c/w)$$

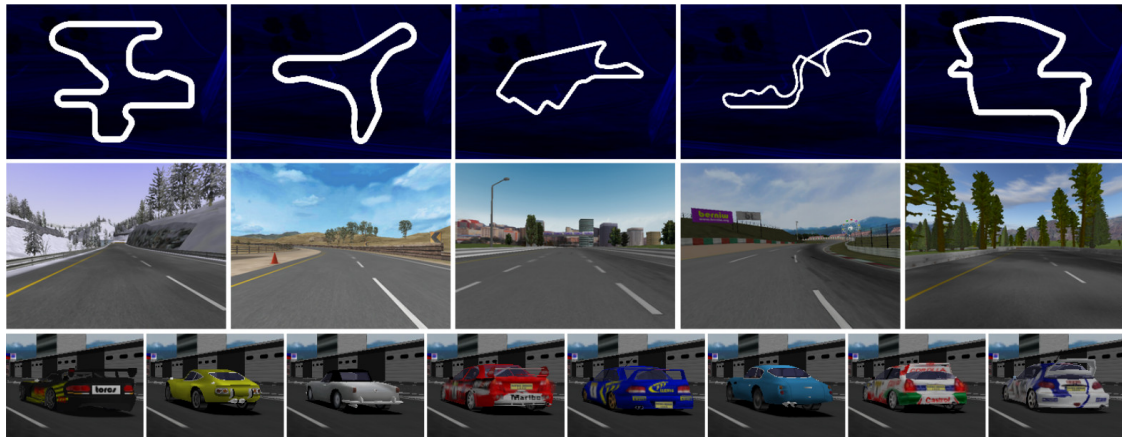
- ▶ s : steering command θ_1 : parameter
- ▶ α : relative orientation d_c : distance to centerline w : road width

Speed controller: (“optimal velocity car following model”)

$$v = v_{max} (1 - \exp(-\theta_2 d_p - \theta_3))$$

- ▶ v : target velocity v_{max} maximal velocity
- ▶ d_p : distance to preceding car $\theta_{2,3}$: parameters

TORCS Simulator



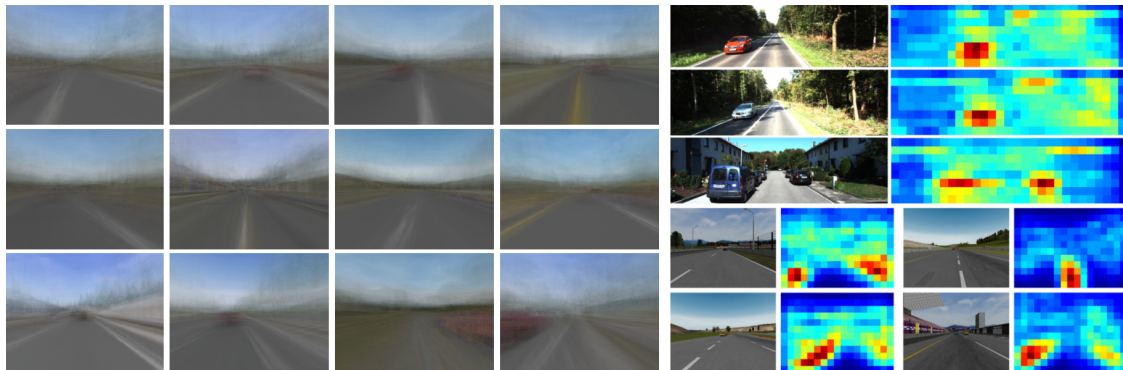
► TORCS: Open source car racing game

<http://torcs.sourceforge.net/>

Results



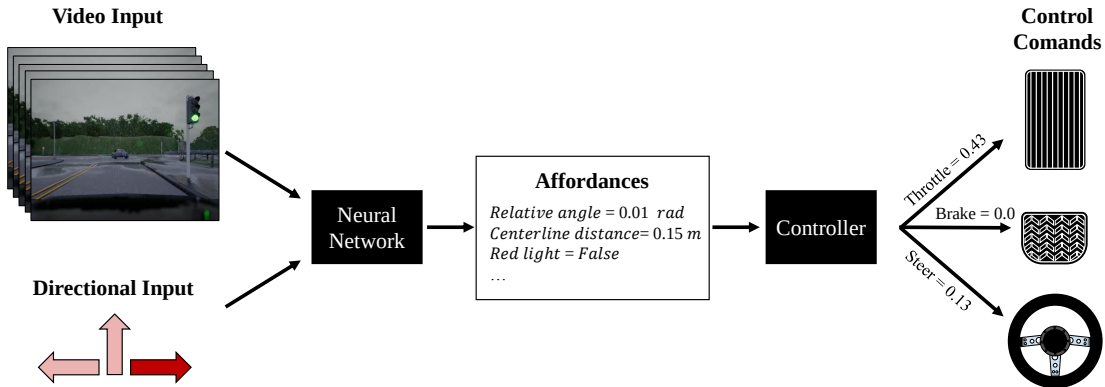
Network Visualization



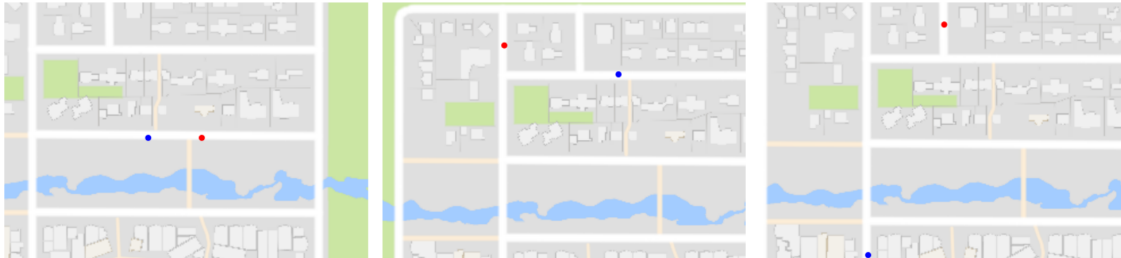
- ▶ Left: Averaged top 100 images activating a neuron in first fully connected layer
- ▶ Right: Maximal response of 4th conv. layer (note: focus on cars and markings)

How can we transfer this idea to cities?

Conditional Affordance Learning

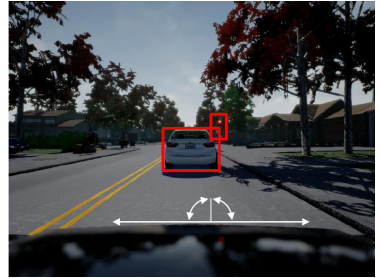
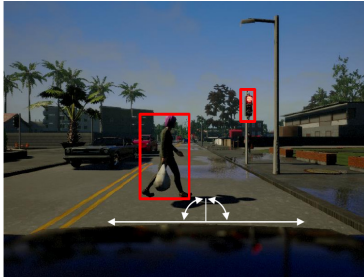


CARLA Simulator



- ▶ Goal: drive from A to B as fast, safely and comfortably as possible
- ▶ Infractions:
 - ▶ Driving on wrong lane
 - ▶ Driving on sidewalk
 - ▶ Running a red light
 - ▶ Violating speed limit
 - ▶ Colliding with vehicles
 - ▶ Hitting other objects

Affordances



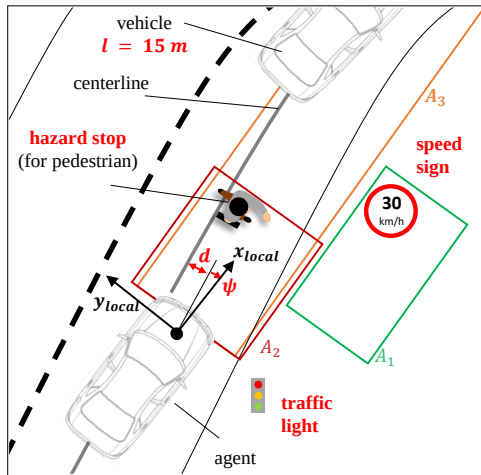
Affordances:

- ▶ Distance to centerline
- ▶ Relative angle to road
- ▶ Distance to lead vehicle
- ▶ Speed signs
- ▶ Traffic lights
- ▶ Hazard stop

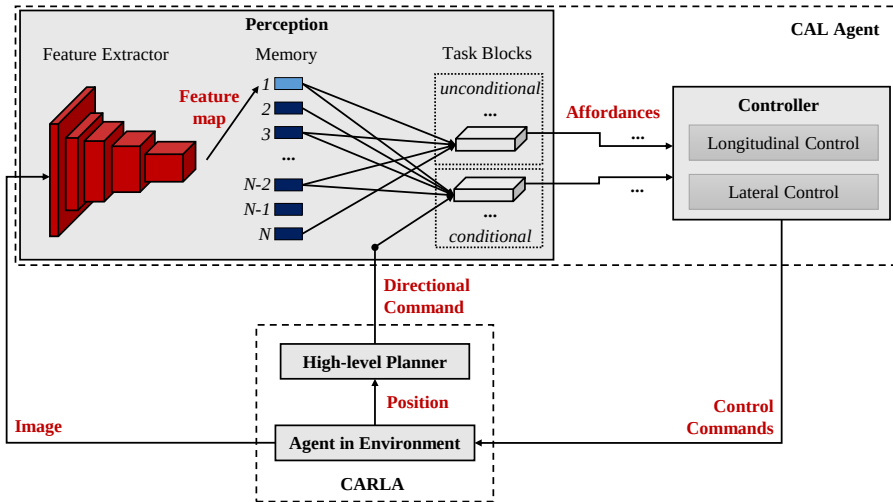
Affordances

Affordances:

- ▶ Distance to centerline
- ▶ Relative angle to road
- ▶ Distance to lead vehicle
- ▶ Speed signs
- ▶ Traffic lights
- ▶ Hazard stop

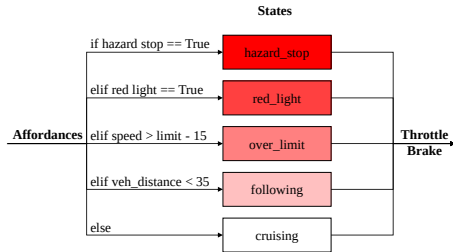


Overview



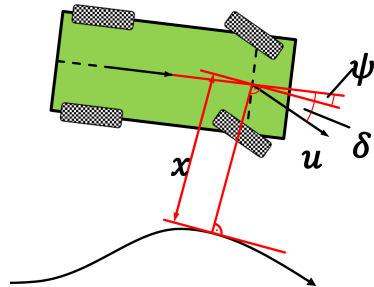
Controller

Longitudinal Control



- Finite-state machine
- PID controller for cruising
- Car following model

Lateral Control



- Stanley controller
- $\delta(t) = \psi(t) + \arctan\left(\frac{kx(t)}{u(t)}\right)$
- Damping term

Parameter Learning

Perception Stack:

- ▶ Multi-task learning: single forward pass $\Rightarrow$ fast learning and inference
- ▶ Dataset: random driving using controller operating on GT affordances
 $\Rightarrow$ 240k images with GT affordances
- ▶ Loss functions:
 - ▶ Discrete affordances: Class-weighted cross-entropy (CWCE)
 - ▶ Continuous affordances: Mean absolute error (MAE)
- ▶ Optimized with ADAM (batch size 32)

Controller:

- ▶ Ziegler-Nichols

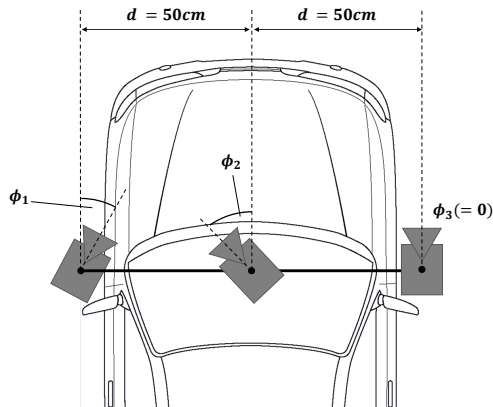
Data Collection

Data Collection:

- ▶ Navigation based on true affordances & random inputs

Data Augmentation:

- ▶ No image flipping
- ▶ Color, contrast, brightness
- ▶ Gaussian blur & noise
- ▶ Provoke rear-end collisions
- ▶ Camera pose randomization



Results

Task	Training conditions				New weather				New town				New town and new weather			
	MP	CIL	RL	CAL	MP	CIL	RL	CAL	MP	CIL	RL	CAL	MP	CIL	RL	CAL
Straight	98	95	89	100	100	98	86	100	92	97	74	93	50	80	68	94
One turn	82	89	34	97	95	90	16	96	61	59	12	82	50	48	20	72
Navigation	80	86	14	92	94	84	2	90	24	40	3	70	47	44	6	68
Nav. dynamic	77	83	7	83	89	82	2	82	24	38	2	64	44	42	4	64

Baselines:

- ▶ MP = Modular Pipeline [Dosovitskiy et al., CoRL 2017]
- ▶ CIL = Conditional Imitation Learning [Codevilla et al., ICRA 2018]
- ▶ RL = Reinforcement Learning A3C [Mnih et al., ICML 2016]

Results



Conditional Navigation

Results



Speed Signs

Results



Car Following

Results



Hazard Stop

Attention



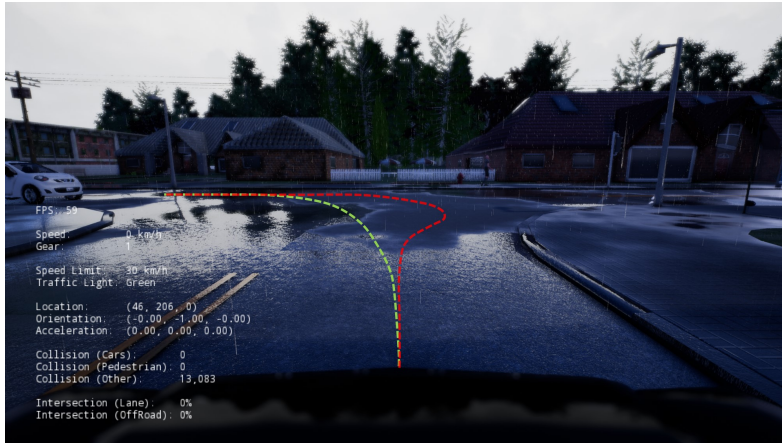
Attention to Hazard Stop

Attention



Attention to Red Light

Path Planning



Optimal Path (green) vs. Traveled Path (red)

Failure Cases



Hazard Stop: False Positive

Failure Cases



Hazard Stop: False Negative

Failure Cases



Red Light: False Positive

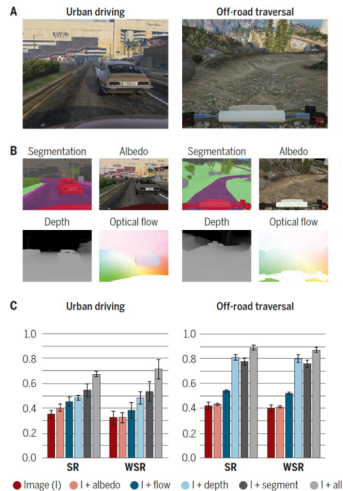
3.3

Visual Abstractions

Does Computer Vision Matter for Action?

Does Computer Vision Matter for Action?

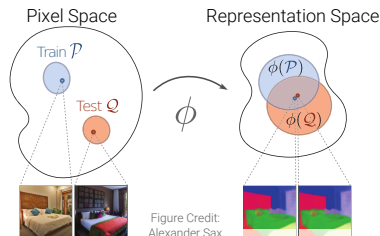
- Analyze various intermediate representations: segmentation, depth, normals, flow, albedo
- Intermediate representations improve results
- Consistent gains across simulations / tasks
- Depth and semantic provide largest gains
- Better generalization performance



Visual Abstractions

What is a good visual abstraction?

- ▶ Invariant (hide irrelevant variations from policy)
- ▶ Universal (applicable to wide range of scenarios)
- ▶ Data efficient (in terms of memory/computation)
- ▶ Label efficient (require little manual effort)



Semantic segmentation:

- ▶ Encodes task-relevant knowledge (e.g. road is drivable) and priors (e.g., grouping)
- ▶ Can be processed with standard 2D convolutional policy networks

Disadvantage:

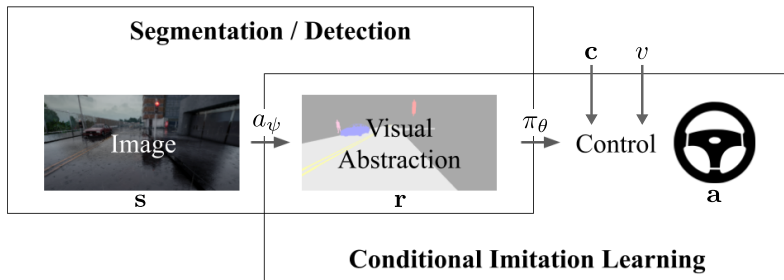
- ▶ Labelling time: ~ 90 min for 1 Cityscapes image

Label Efficient Visual Abstractions

Questions:

- ▶ What is the trade-off between annotation time and driving performance?
- ▶ Can selecting specific semantic classes ease policy learning?
- ▶ Are visual abstractions trained with few images competitive?
- ▶ Is fine-grained annotation important?
- ▶ Are visual abstractions able to reduce training variance?

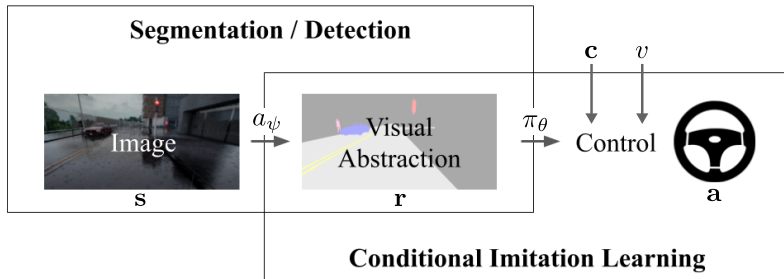
Label Efficient Visual Abstractions



Model:

- ▶ Visual abstraction network $a_\psi : \mathbf{s} \mapsto \mathbf{r}$ (state $\mapsto$ abstraction)
- ▶ Control policy $\pi_\theta : \mathbf{r}, \mathbf{c}, v \mapsto \mathbf{a}$ (abstraction, command, velocity $\mapsto$ action)
- ▶ Composing both yields $\mathbf{a} = \pi_\theta(a_\psi(\mathbf{s}))$ (state $\mapsto$ action)

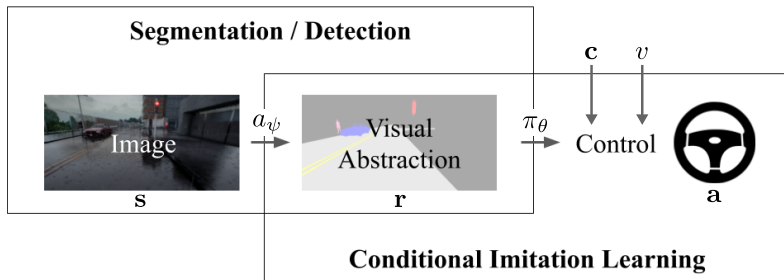
Label Efficient Visual Abstractions



Datasets:

- ▶ n_r images annotated with semantic labels $\mathcal{R} = \{\mathbf{s}^i, \mathbf{r}^i\}_{i=1}^{n_s}$
- ▶ n_a images annotated with expert actions $\mathcal{A} = \{\mathbf{s}^i, \mathbf{a}^i\}_{i=1}^{n_a}$
- ▶ We assume $n_r \ll n_a$

Label Efficient Visual Abstractions



Training:

- ▶ Train visual abstraction network $a_\phi(\cdot)$ using semantic dataset $\mathcal{R}$
- ▶ Apply this network to obtain control dataset $\mathcal{C}_\phi = \{a_\psi(\mathbf{s}^i), \mathbf{a}^i\}_{i=1}^{n_a}$
- ▶ Train control policy $\pi_\theta(\cdot)$ using control dataset $\mathcal{C}_\phi$

Control Policy

Model:

- CILRS [Codevilla et al., ICCV 2019]

Input:

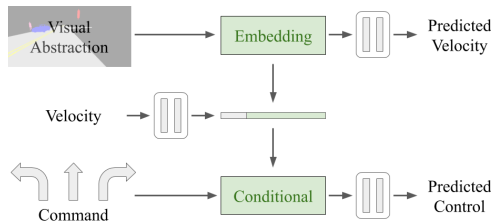
- Visual abstraction $\mathbf{r}$
- Navigational command $\mathbf{c}$
- Vehicle velocity v

Output:

- Action/control $\hat{\mathbf{a}}$ and velocity $\hat{v}$

Loss:

- $\mathcal{L} = \|\mathbf{a} - \hat{\mathbf{a}}\|_1 + \lambda \|v - \hat{v}\|_1$



Visual Abstractions



Privileged Segmentation (14 classes):

- ▶ Ground-truth semantic labels for 14 classes
- ▶ Upper bound for analysis

Visual Abstractions



Privileged Segmentation (6 classes):

- ▶ Ground-truth semantic labels for 2 stuff and 4 object classes
- ▶ Upper bound for analysis

Visual Abstractions



Inferred Segmentation (14 classes):

- ▶ Segmentation model trained on 14 classes
- ▶ ResNet and Feature Pyramid Network (FPN) with segmentation head

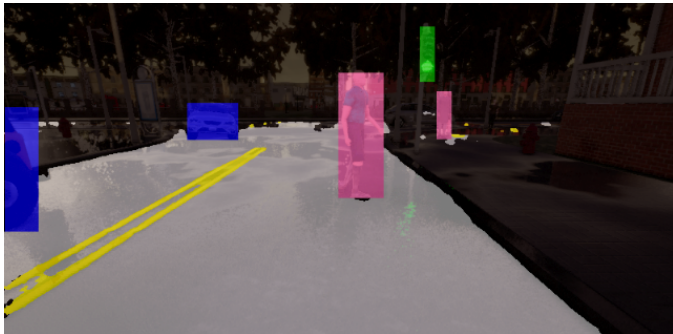
Visual Abstractions



Inferred Segmentation (6 classes):

- ▶ Segmentation model trained on 2 stuff and 4 object classes
- ▶ ResNet and Feature Pyramid Network (FPN) with segmentation head

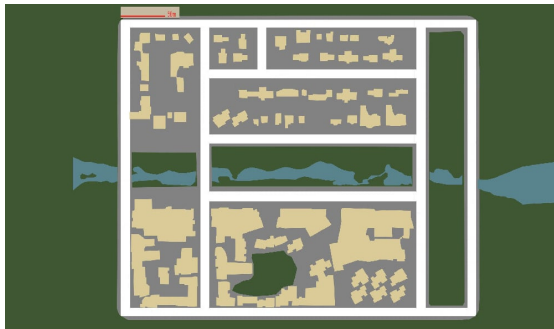
Visual Abstractions



Hybrid Detection and Segmentation (6 classes):

- ▶ Segmentation model trained on 2 stuff classes: road, lane marking
- ▶ Object detection trained on 4 object classes: vehicle, pedestrian, traffic light (r/g)

Evaluation



Training Town



Test Town

- ▶ CARLA 0.8.4 NoCrash benchmark
- ▶ Random start and end location
- ▶ Metric: Percentage of successfully completed episodes (success rate)

Traffic Density



Empty



Regular



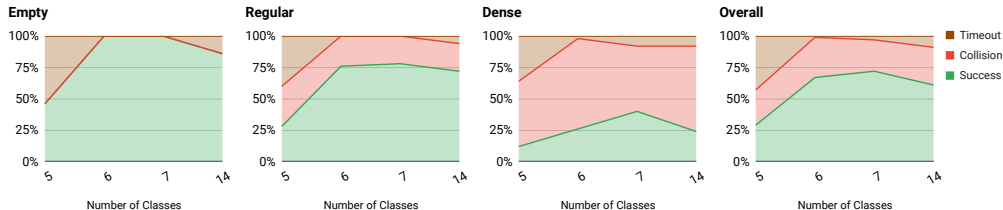
Dense

- Difficulty varies with number of dynamic agents in the scene
- Empty: 0 Agents Regular: 65 Agents Dense: 220 Agents

Identifying Most Relevant Classes (Privileged)

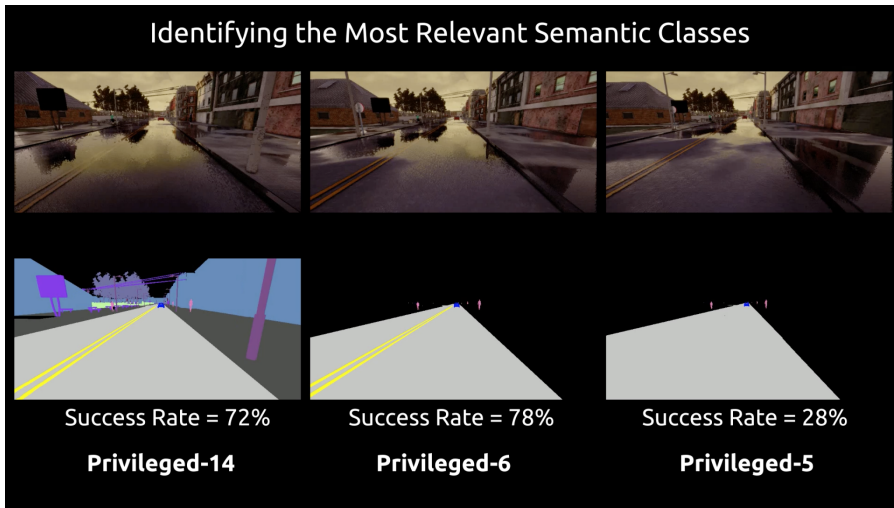
- ▶ 14 classes: road, lane marking, vehicle, pedestrian, green light, red light, sidewalk, building, fence, pole, vegetation, wall, traffic sign, other
- ▶ 7 classes: road, lane marking, vehicle, pedestrian, green light, red light, sidewalk, ~~building, fence, pole, vegetation, wall, traffic sign, other~~
- ▶ 6 classes: road, lane marking, vehicle, pedestrian, green light , red light, ~~sidewalk~~
- ▶ 5 classes: road, ~~lane marking~~, vehicle, pedestrian, green light, red light

Identifying Most Relevant Classes (Privileged)

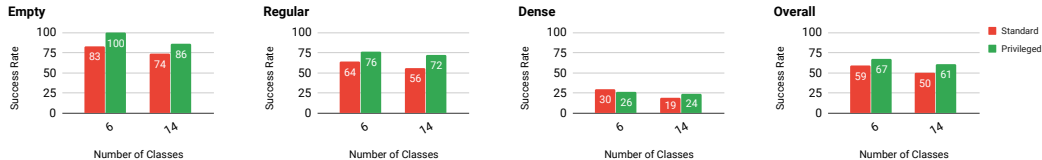


- Moving from 14 to 6 classes does not hurt driving performance (on contrary)
- Drastic performance drop when lane markings are removed

Identifying Most Relevant Classes (Privileged)

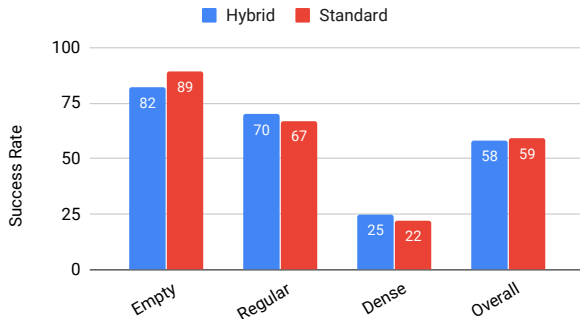


Identifying Most Relevant Classes (Inferred)



- ▶ Small performance drop when using inferred segmentations
- ▶ 6-class representation consistently improves upon 14-class representation
- ▶ We use the 6-class representation for all following experiments

Hybrid Representation

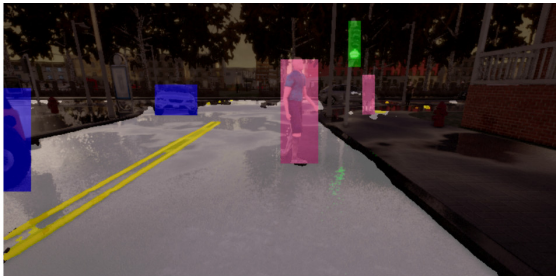


- Performance of hybrid representation matches standard segmentation
- Annotation time (segmentation): ~ 300 seconds per image and per class
- Annotation time (hybrid): ~ 20 seconds per image and per class

Summary



Trained with 6400 finely annotated images and 14 classes
Annotation time $\approx$ 7500 hours, policy success rate = 50%

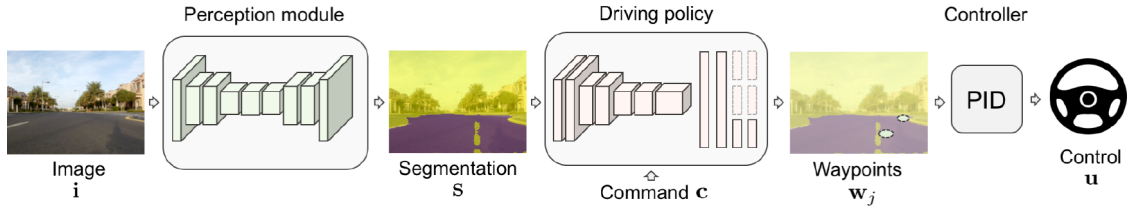


Trained with 1600 coarsely annotated images and 6 classes
Annotation time $\approx$ 50 hours, policy success rate = 58%

3.4

Driving Policy Transfer

Driving Policy Transfer



Problem:

- Driving policies learned in simulation often do not transfer well to the real world

Idea:

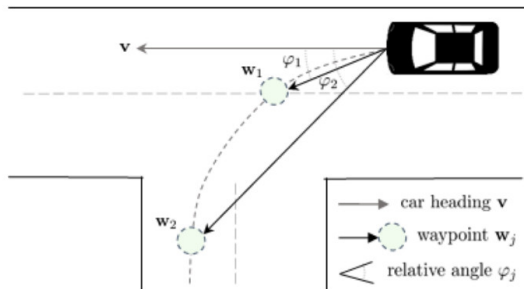
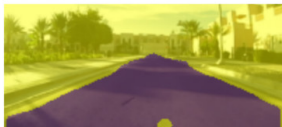
- Encapsulate driving policy such that it is not directly exposed to raw perceptual input or low-level control (input: semantic segmentation, output: waypoints)
- Allows for transferring driving policy without retraining or finetuning

Waypoint Representation

Simulation



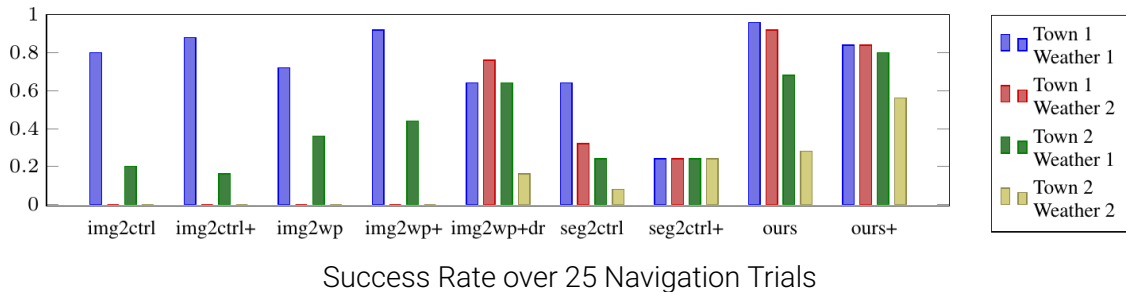
Real world



Representation:

- ▶ Input: Semantic segmentation (per pixel “road” vs. “non-road”)
- ▶ Output: 2 waypoints (distance to vehicle, relative angle wrt. vehicle heading)
 - ▶ One sufficient for steering, second one for braking before turns

Results



- Driving Policy: Conditional Imitation Learning (branched)
- Control: PID controller for lateral and longitudinal control
- Results: Full method generalizes best (“+” = with data augmentation)

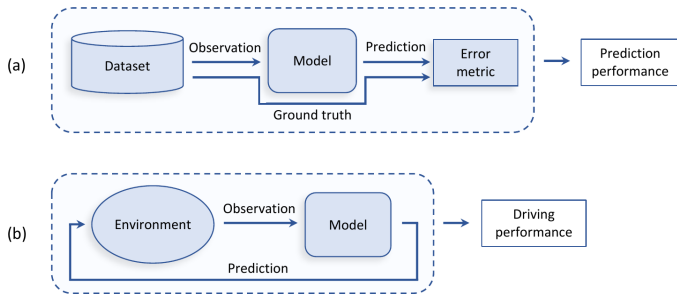
Results



3.5

Online vs. Offline Evaluation

Online vs. Offline Evaluation



- Online evaluation (e.g., CARLA/real vehicle) can be expensive or dangerous
- Offline evaluation on a pre-recorded validation dataset is cheap and easy
- Question: How predictive is offline evaluation (a) for the online task (b)?
- Empirical study using CIL on CARLA trained with MSE loss on steering angle

Online Metrics

- ▶ Success Rate: Percentage of routes successfully completed
- ▶ Average Completion: Average fraction of distance to goal covered
- ▶ Km per Infraction: Average driven distance between 2 infractions

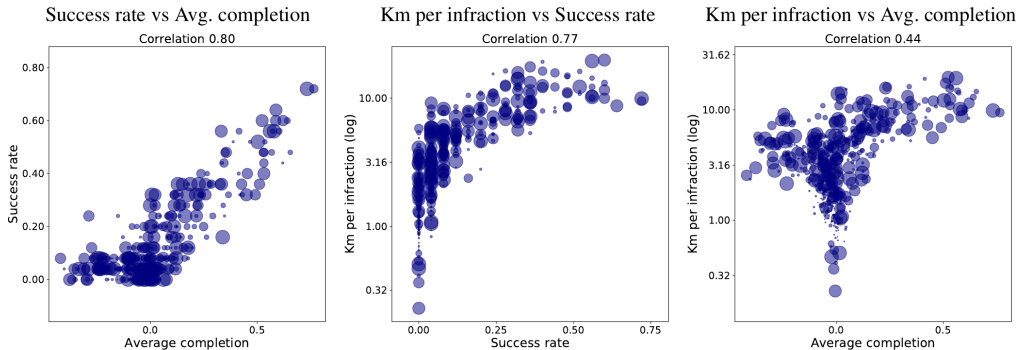
Remark: The current CARLA metrics Infraction Score and Driving Score are not considered in this work from 2018, but would likely lead to similar conclusions.

Offline Metrics

Metric name	Parameters	Metric definition
Squared error	–	$\frac{1}{ V } \sum_{i \in V} \ \mathbf{a}_i - \hat{\mathbf{a}}_i\ ^2$
Absolute error	–	$\frac{1}{ V } \sum_{i \in V} \ \mathbf{a}_i - \hat{\mathbf{a}}_i\ _1$
Speed-weighted absolute error	–	$\frac{1}{ V } \sum_{i \in V} \ \mathbf{a}_i - \hat{\mathbf{a}}_i\ _1 v_i$
Cumulative speed-weighted absolute error	T	$\frac{1}{ V } \sum_{i \in V} \left\ \sum_{t=0}^T (\mathbf{a}_{i+t} - \hat{\mathbf{a}}_{i+t}) v_{i+t} \right\ _1$
Quantized classification error	σ	$\frac{1}{ V } \sum_{i \in V} (1 - \delta(Q(\mathbf{a}_i, \sigma), Q(\hat{\mathbf{a}}_i, \sigma)))$
Thresholded relative error	α	$\frac{1}{ V } \sum_{i \in V} \theta(\ \hat{\mathbf{a}}_i - \mathbf{a}_i\ - \alpha \ \mathbf{a}_i\)$

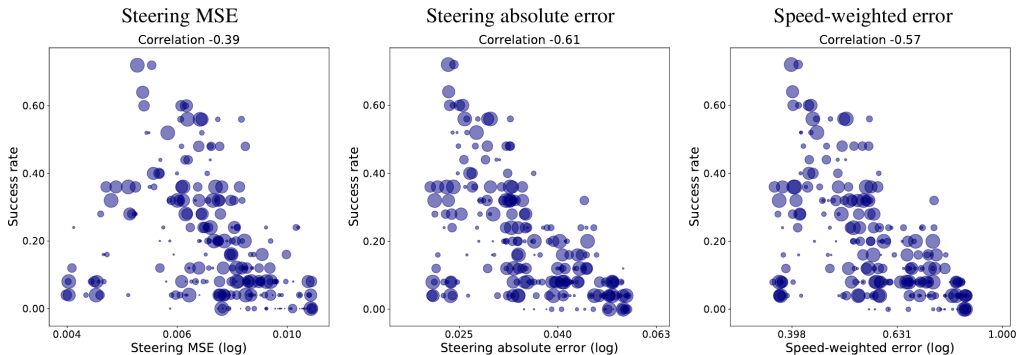
- $\mathbf{a}/\hat{\mathbf{a}}$: true/predicted steering angle $|V|$: #samples in validation set
- v : speed $\delta(\cdot)$: Kronecker delta function θ : Heaviside step function
- $Q \in \{-1, 0, 1\}$: Quantization $x < -\sigma$ $-\sigma \leq x < \sigma$ $x \geq \sigma$

Results: Online vs. Online



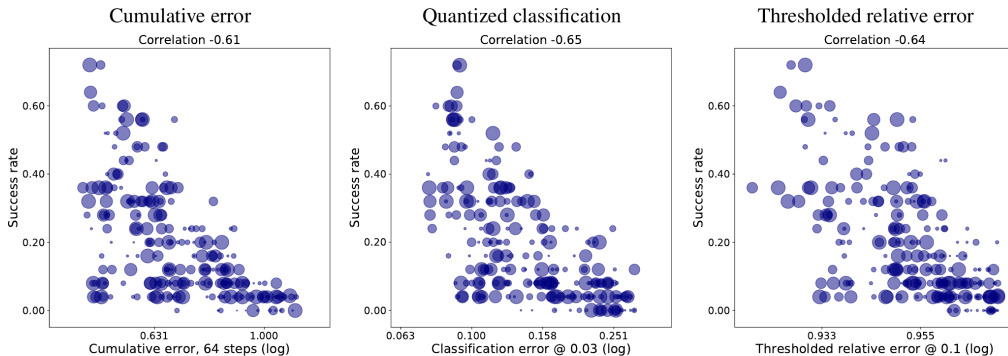
- Generalization performance (town 2, new weather), radius = training iteration
- 45 different models varying dataset size, augmentation, architecture, etc.
- Success rate correlates well with average completion and km per infraction

Results: Online vs. Offline



- All metrics not well correlated, Mean Square Error (MSE) performs worst
- Absolute steering error improves, speed weighting is not important

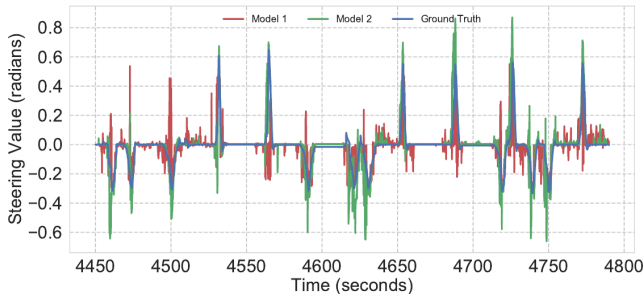
Results: Online vs. Offline



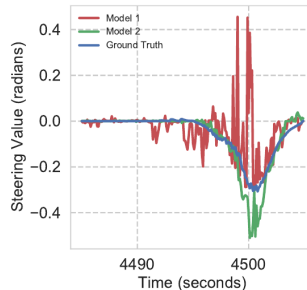
- Cumulating the error over time does not improve the correlation
- Quantized classification and thresholded relative error perform best

Case Study

Steering angle prediction vs Time

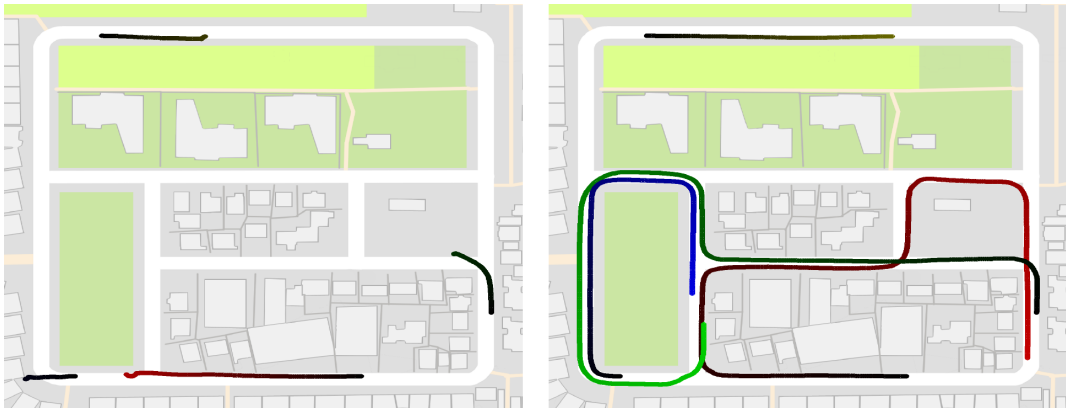


Zoom-in of one turn



- Model 1: Trained with single camera & ℓ_2 loss (=bad model)
- Model 2: Trained with three cameras & ℓ_1 loss (=good model)
- Predictions of both models noisy, but Model 1 predicts occasionally very large errors leading to crashes, however the average prediction error is similar

Case Study



- ▶ Model 1 crashes in every trial but model 2 can drive successfully
- ▶ Illustrates the difficulty of using offline metrics for predicting online behavior

Summary

- ▶ Direct perception predicts intermediate representations
- ▶ Low-dimensional affordances or classic computer vision representations (e.g., semantic segmentation, depth) can be used as intermediate representations
- ▶ Decouples perception from planning and control
- ▶ Hybrid model between imitation learning and modular pipelines
- ▶ Direct methods are more interpretable as the representation can be inspected
- ▶ Effective visual abstractions can be learned using limited supervision
- ▶ Planning can also be decoupled from control for better transfer
- ▶ Offline metrics are not necessarily indicative of online driving performance