
HE VD
IG

# Intelligence Artificielle pour les systèmes autonomes (IAA)

## End-to-end learning and direct perception

Prof. Yann Thoma - Prof. Marina Zapater

*Février 2024*

*Basé sur le cours du Prof. A. Geiger*

# Outline

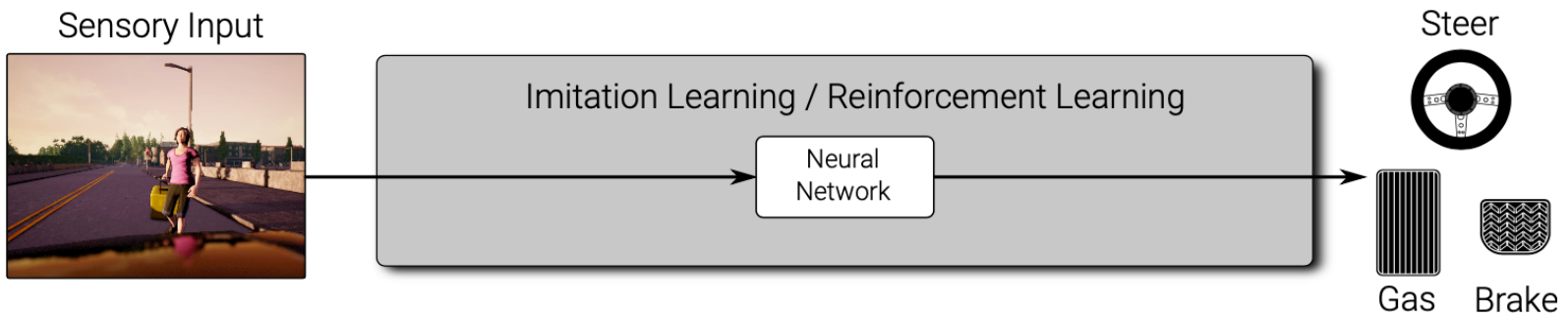
## Today's lesson

→ **Imitation Learning**
- PilotNet and Dronet
- Conditional Imitation Learning

→ Direct perception
- Affordance Learning
- Conditional Affordance Learning
- Visual abstractions

# End-to-end learning

**The “Second approach” we saw on our introduction lecture**




→ End-to-end requires cheap(er) annotations

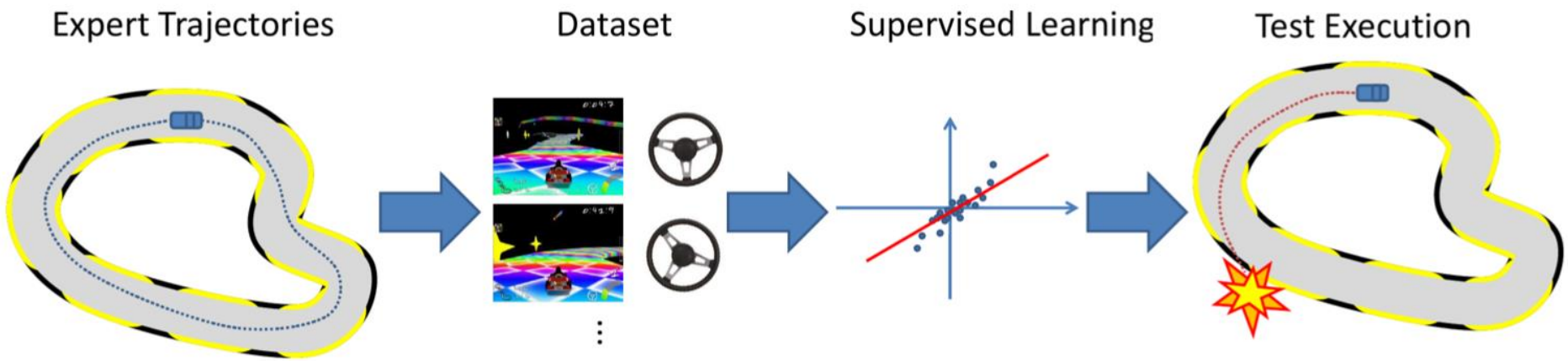
→ Issues with training/generalization

→ Lack of interpretability…

→ Certification issue…

# Imitation learning basics

## Learning from “experts”




→ Hard-coding policies is difficult : better use a data-driven approach

→ Given demonstrations (expert trajectories) train a policy to mimic decisions

https://smartlabai.medium.com/a-brief-overview-of-imitation-learning-8a8a75c44a9c

# Imitation Learning

## Car Racing: Imitating the human driver's trajectories


Trainer
(Human Driver)


Trainee
(Neural Network)

# A “more formal” definition of imitation learning

## For general imitation learning

→ We have an environment (Markov Decision process) comprised of:

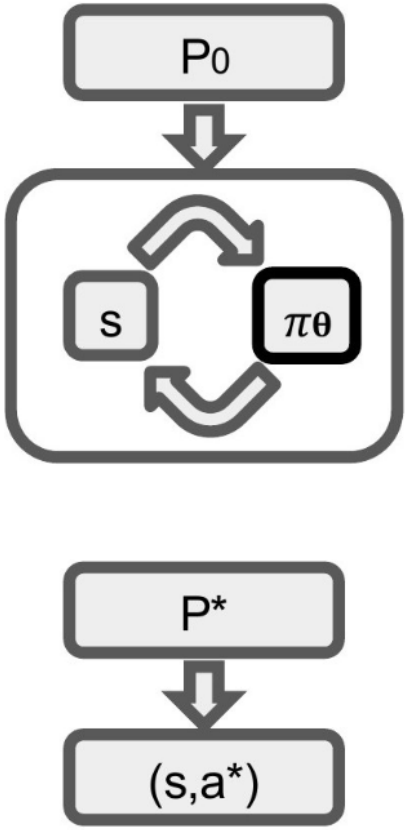
- States (s) : what can be observed (for example the game screen)
- Actions (a) : the actuations we can perform (e.g.: speed, turning angle)

→ The goal is to learn a policy $\pi_\theta$

- The complexity relies on the fact that each time the expert takes an action, that brings you to a different state…
- So you would need to ask the expert each and every time, or said differently the state distribution $P(s|\pi_\theta)$ depends on the roll-out of the policy

→ To solve this problem we use “behavioral cloning”

- The state distribution is reduced $P^*$ to the possibilities seen by the expert
- Now this is a supervised learning problem



# The simplest “flavour” of imitation learning

## Behavioural cloning (when it works and when it does not)

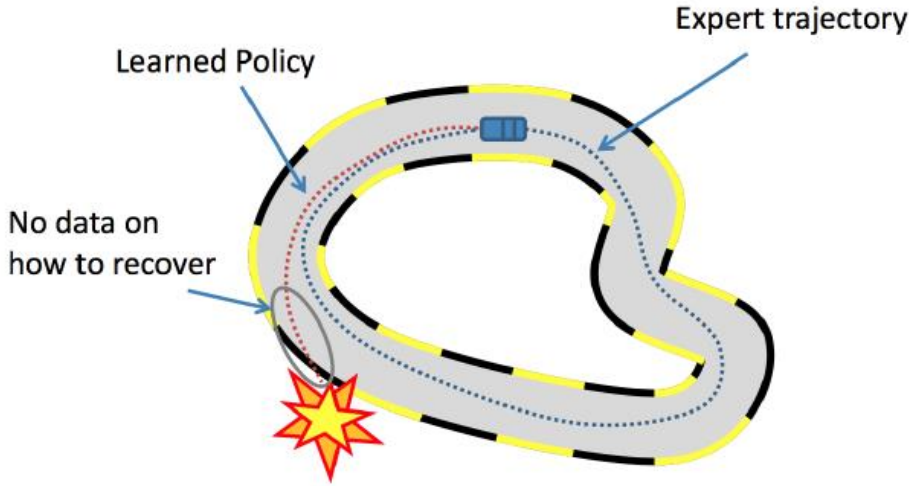
→ We try to find a policy that given a certain state provides the best action $(s_0, a_0)$, $(s_1, a_1)$...

→ At each decision interval, we minimize the loss function $\mathrm{L}(a^*, \pi_\theta(s))$

→ What if $\pi_\theta$ makes a mistake?

- We’ll enter into a state we never saw before
- We have no data from the expert on how to recover
- We cannot recover (we crash!)

→ So… what do we do?



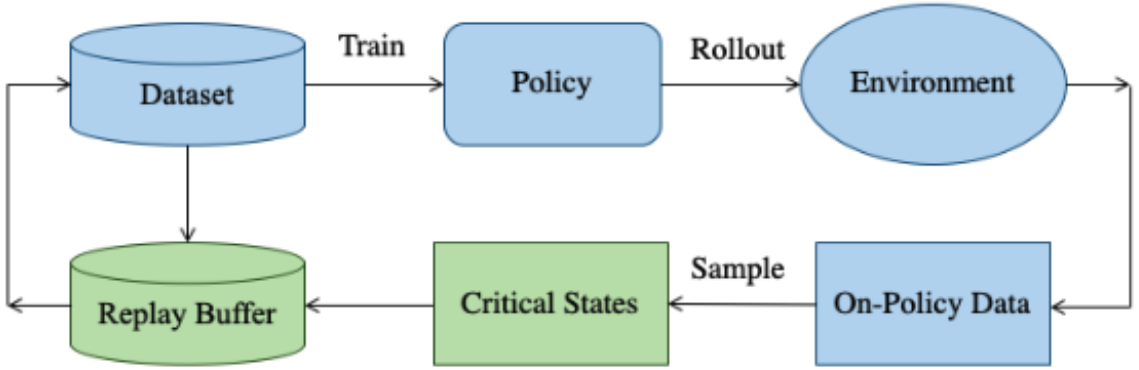
# Training with critical states

## Having access to an interactive demonstrator: data and policy aggregation

→ Sample critical states from the collected policy data based on the utility they provided to the learned policy (in terms of driving behavior)

→ Incorporate a replay buffer which focuses on the high uncertainty regions

→ Or… you choose another algorithm



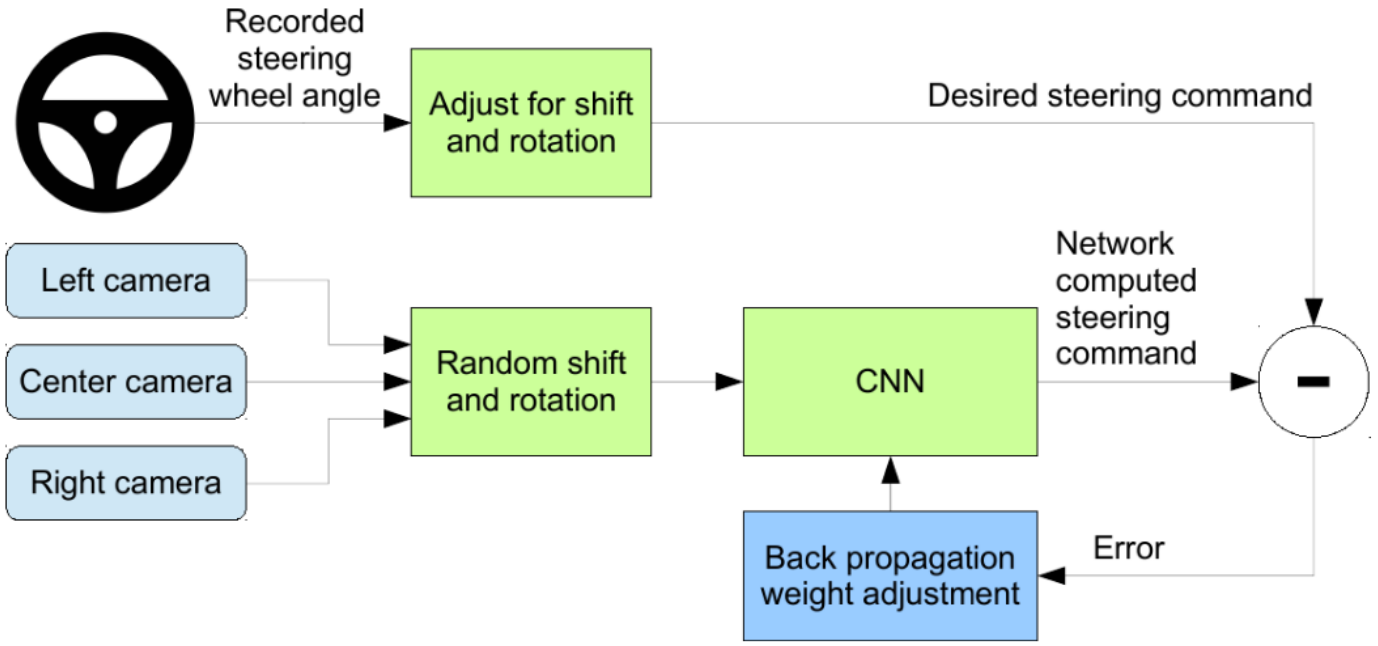
# PilotNet: End-to-end learning for self-driving cars

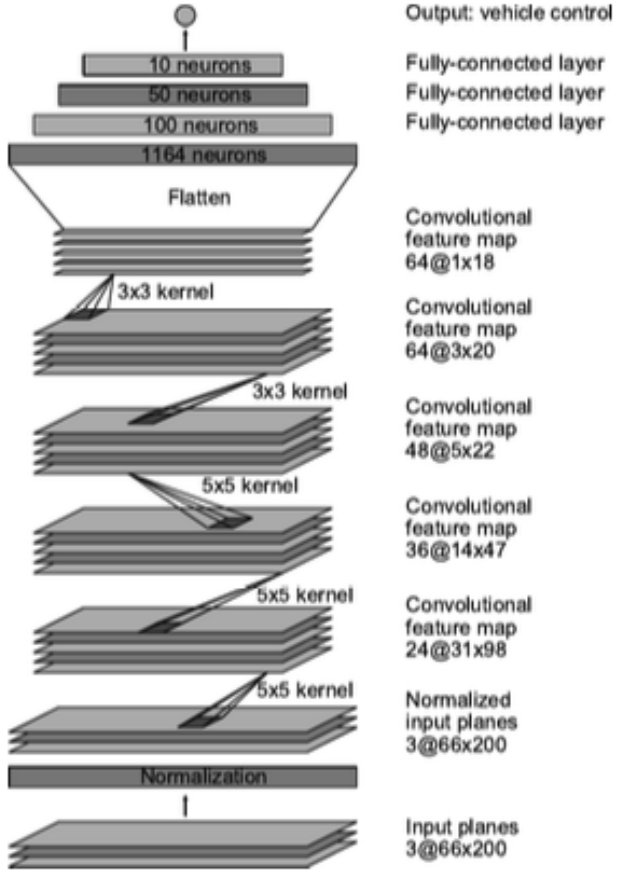
## System overview

→ 3 cameras provide virtually shifted/rotated images

→ Convolutional network (250k params)

→ YUV image representation



# PilotNet

## NVIDIA BB8


https://www.youtube.com/watch?v=-96BEoXJMs0

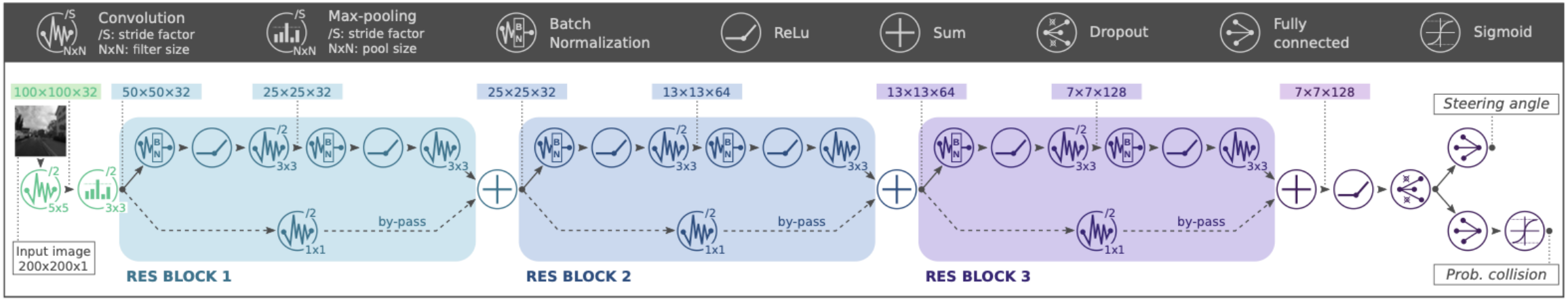
# Dronet

## And simplifications over Dronet to make Crazyflie fly

→ Using input images from the drone camera (grayscale)

→ Feeding those to ResNets

→ To obtain the probability of collision and the steering angle the drone should take




https://cyberlearn.hes-so.ch/pluginfile.php/2170089/mod_resource/content/0/IoT19_Palossi.pdf

# Dronet Training and Testing

## And performance

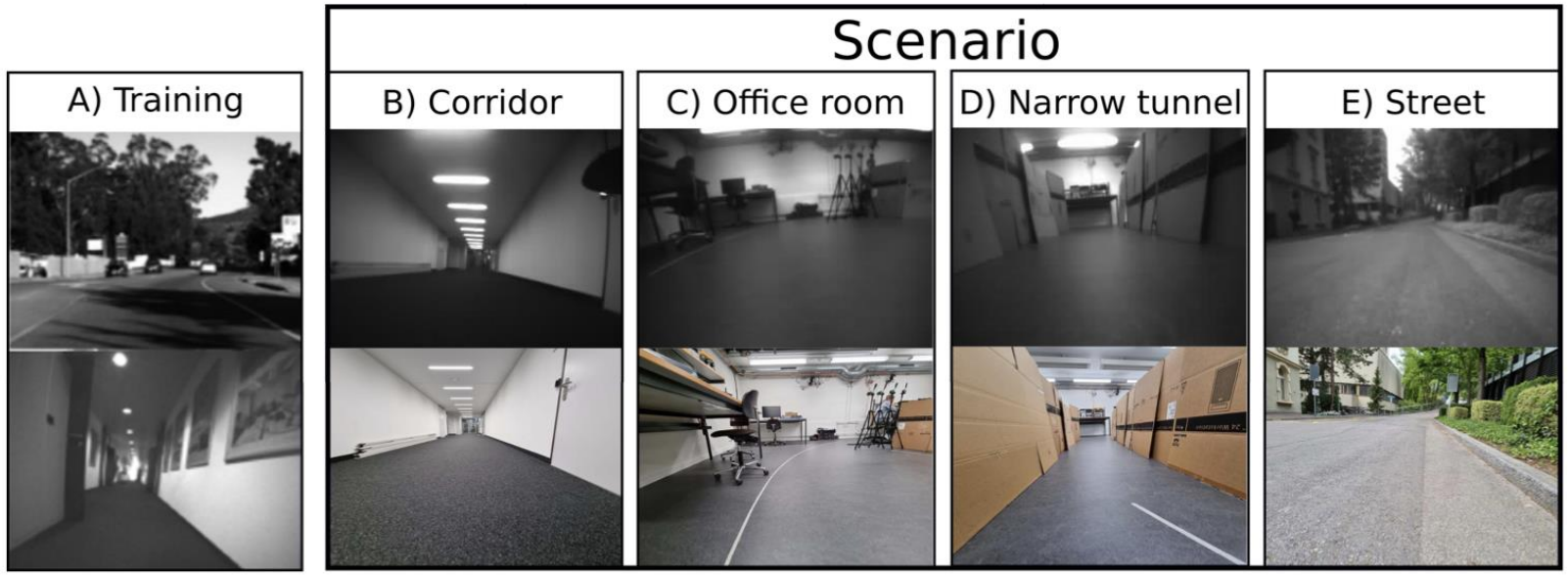



Fig. 8. Samples of: A) the training images (both Udacity and Himax dataset); B) working-place corridor; C) office room with furniture; D) narrow pipeline-like tunnel; E) public street.

https://cyberlearn.hes-so.ch/pluginfile.php/2170086/mod_resource/content/0/Improving_Autonomous_Nano-Drones_Performance_via_Automated_End-to-End_Optimization_and_Deployment_of_DNNs-3.pdf

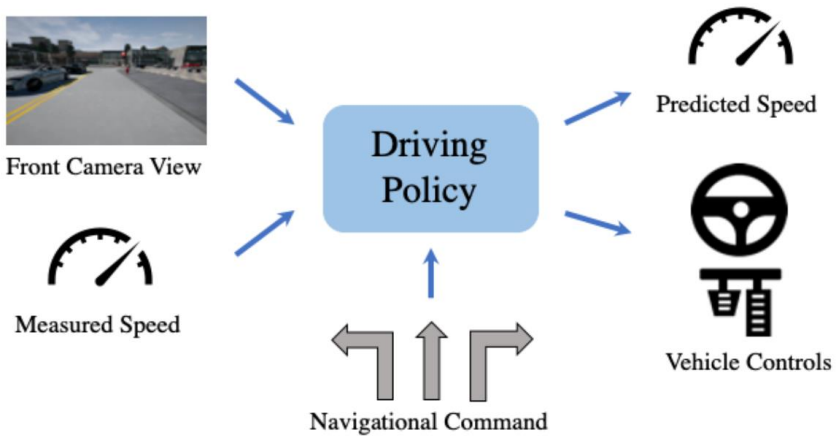
# Conditional imitation learning

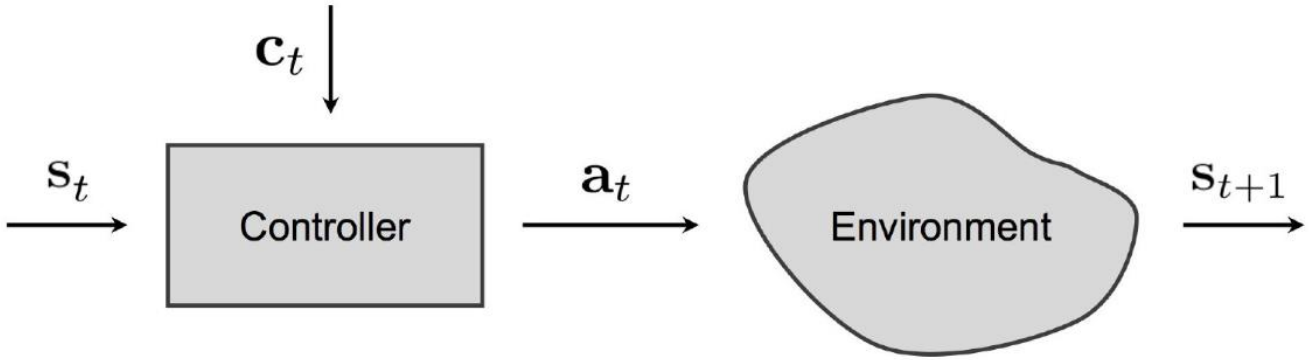
## Taking decisions when there are several possibilities

→ We add a condition controller on the navigation command (left, right, straight)

→ High-level navigation command provided by GPS

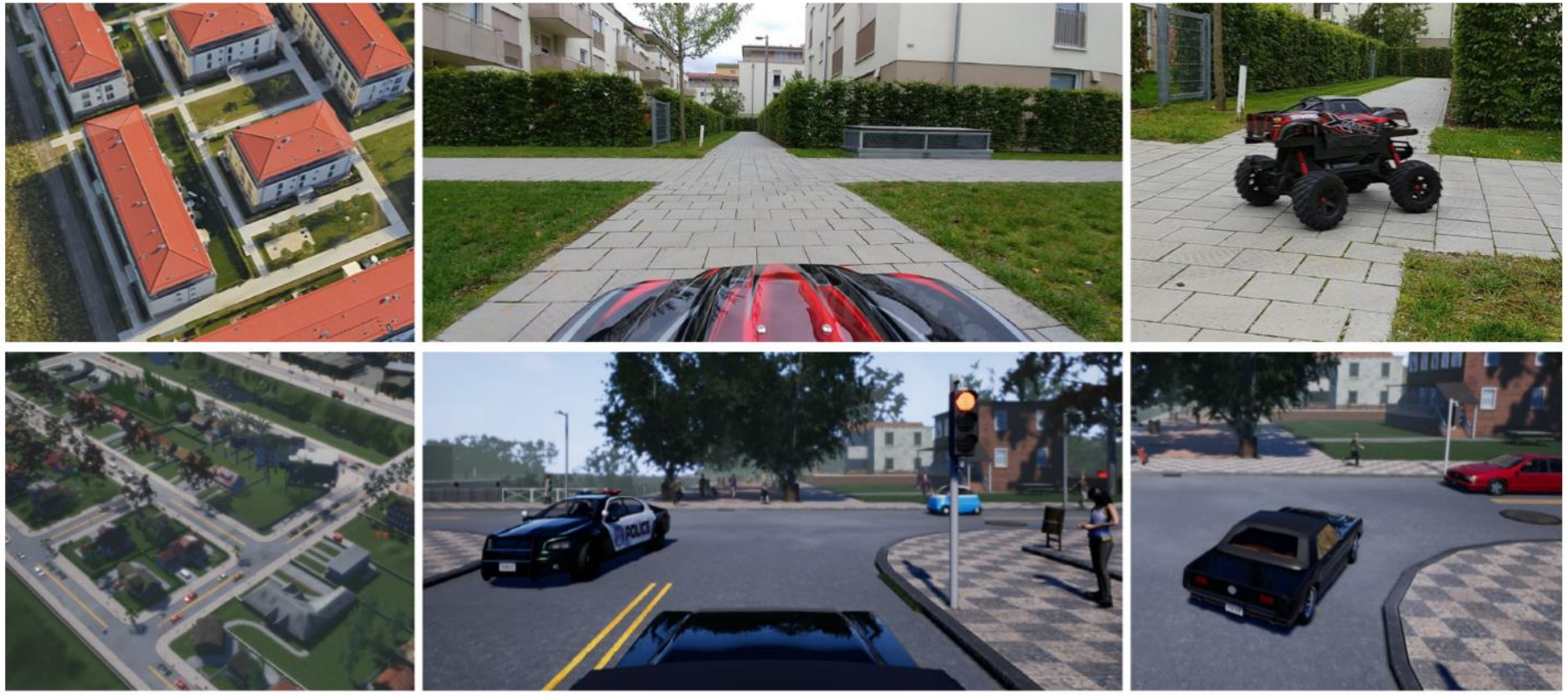
- Telling the vehicle where to go on next turn (left, right, straight)
- Reducing ambiguity induced by the environment



# Conditional imitation learning

## End-to-end driving via Conditional Imitation Learning

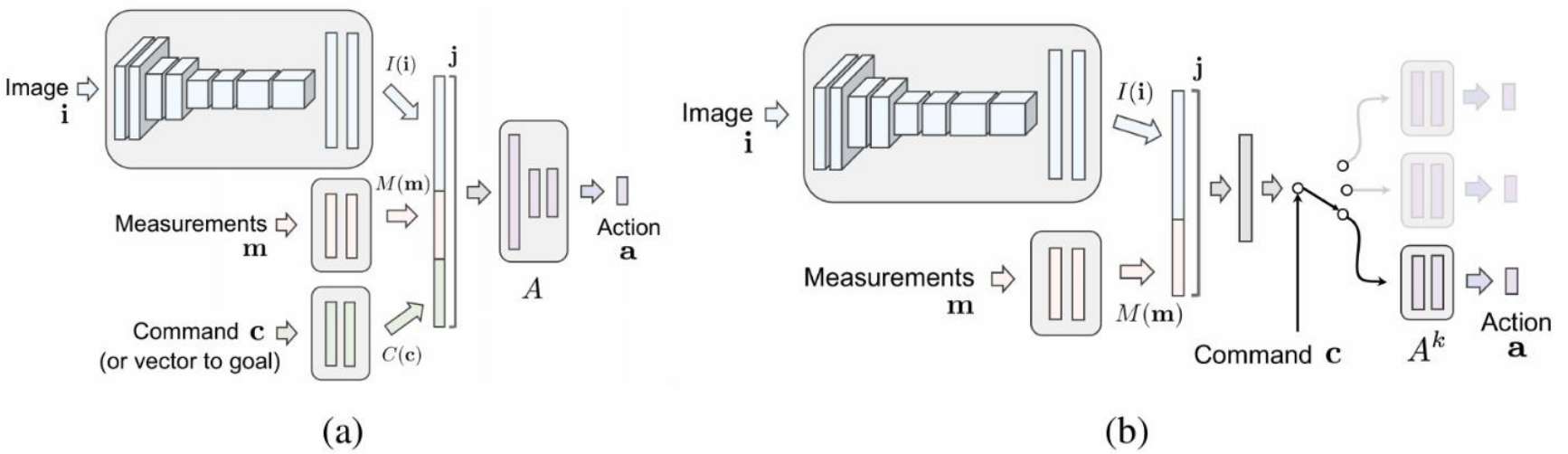


(a) Aerial view of test environment (b) Vision-based driving, view from onboard camera (c) Side view of vehicle

https://www.youtube.com/watch?v=cFtnflNe5fM

# Different proposals for conditional imitation learning

## Two example architectures




Measurements “m” capture additional information (speed of vehicle)

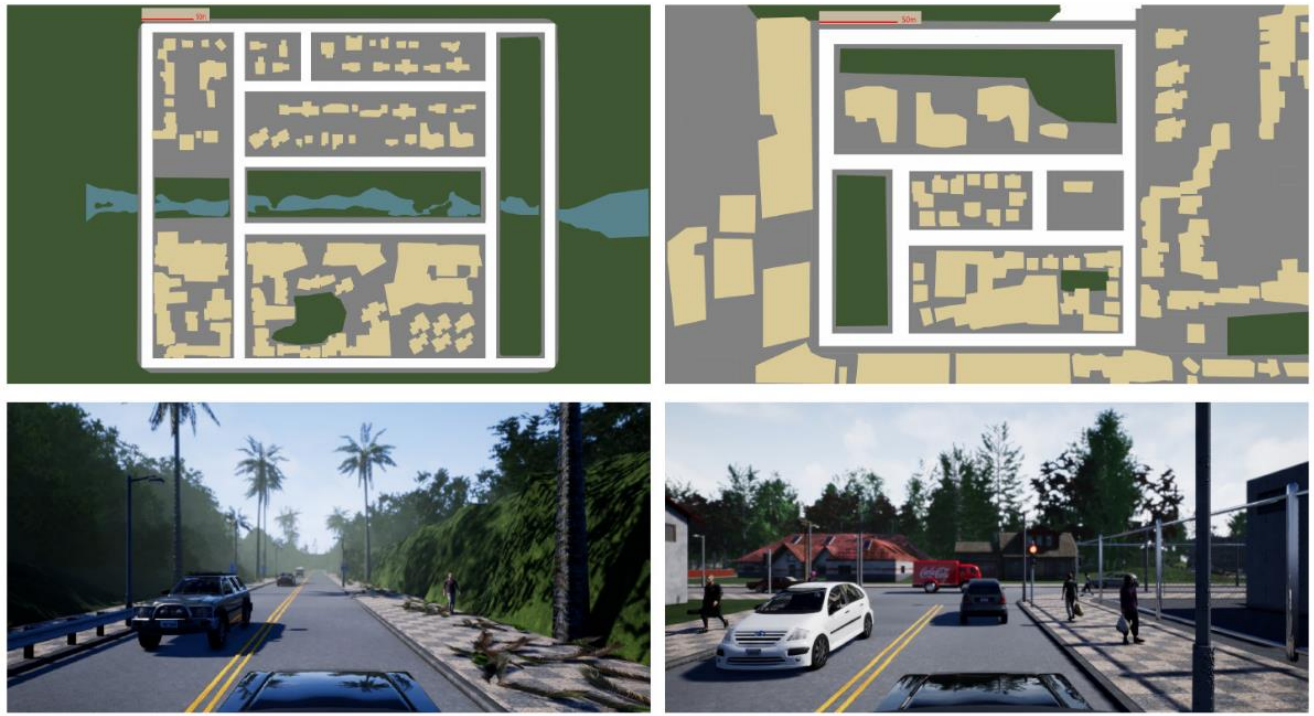
a) Extract features and concatenate with image features C(c)

b) Command “c” acting as a switch between specialized submodules

http://vladlen.info/papers/conditional-imitation.pdf

# Code available for testing on Carla simulator

## Training in one town, testing on another



Town 1 (training)  Town 2 (testing)

https://github.com/carla-simulator/imitation-learning

# Outline

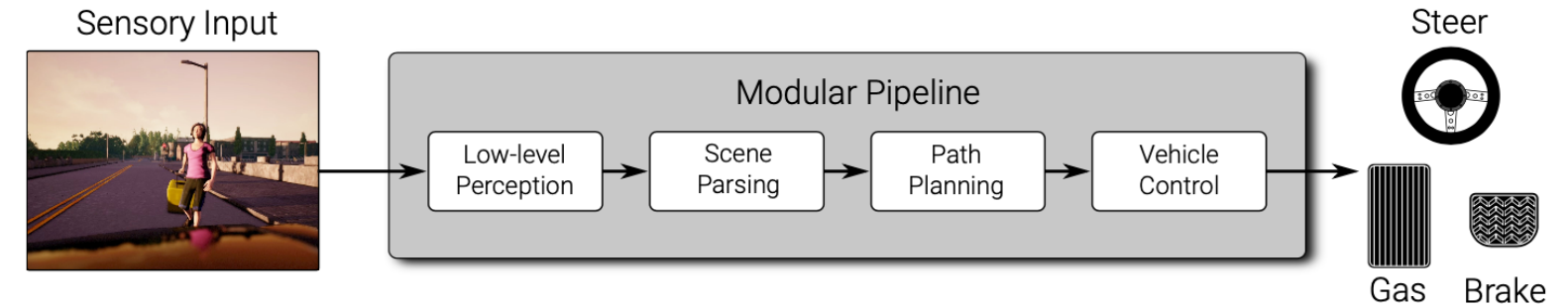
## Today's lesson

→ Imitation Learning
- PilotNet and Dronet
- Conditional Imitation Learning

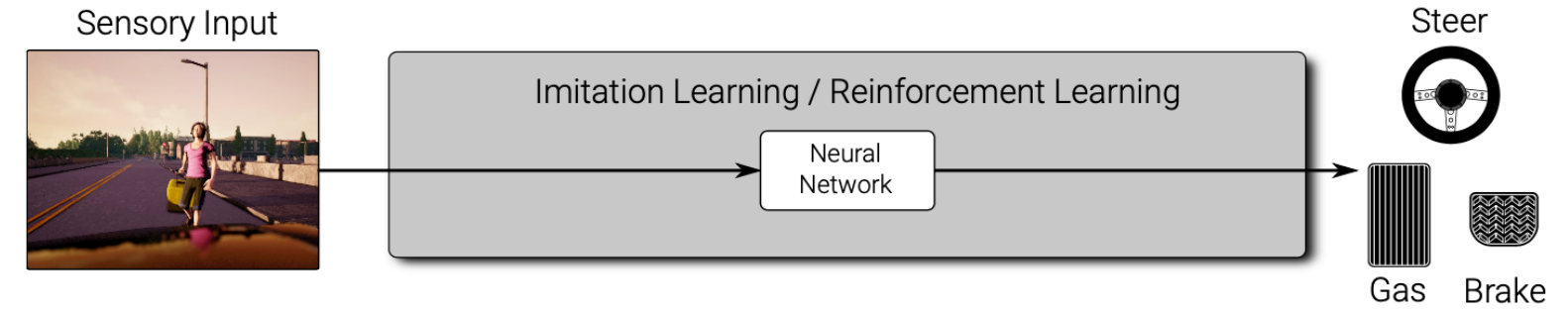
→ **Direct perception**
- Affordance Learning
- Conditional Affordance Learning
- Visual abstractions

# Direct perception

## Advantages and drawbacks wrt other methods



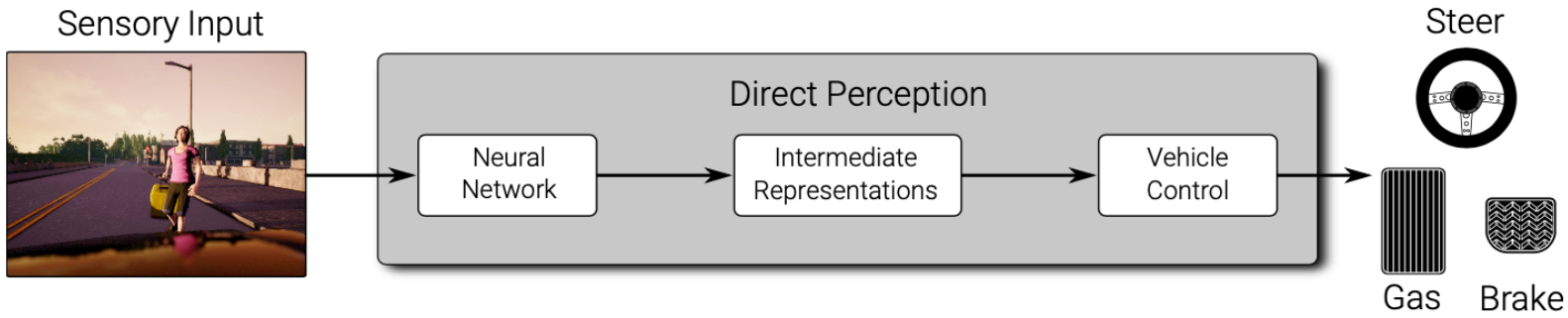

+ Modular + Interpretable - Expert decisions - Piece-wise training




+ End-to-end + Simple - Generalization - Interpretable - Data

# Direct perception

## A middle ground between end-to-end and modular pipeline




→ Hybrid model between end-to-end and modular pipelines

→ Learning to predict low-dimensional intermediate representations

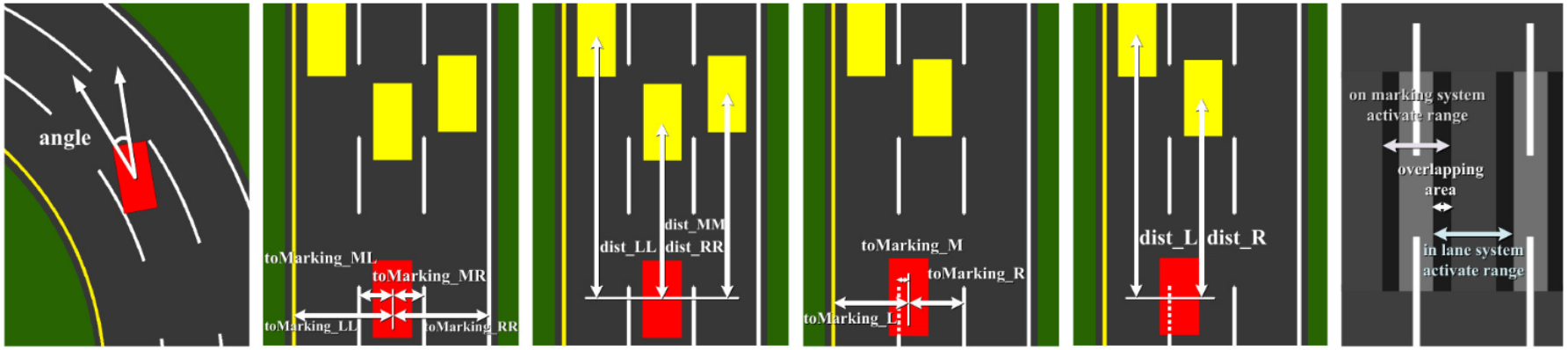
→ Decoupling perception from planning and control

→ Allows to exploit classical controllers or learned controllers (or a mix)

# Direct perception for autonomous driving

## Using “affordances”

→ Affordance: attribute of the environment which limits the space of actions




(a) angle (b) in lane: toMarking (c) in lane: dist (d) on mark.: toMarking (e) on marking: dist (f) overlapping area

→ In this example, 13 affordances in total

https://arxiv.org/pdf/1505.00256.pdf

# Learning affordances via a CNN

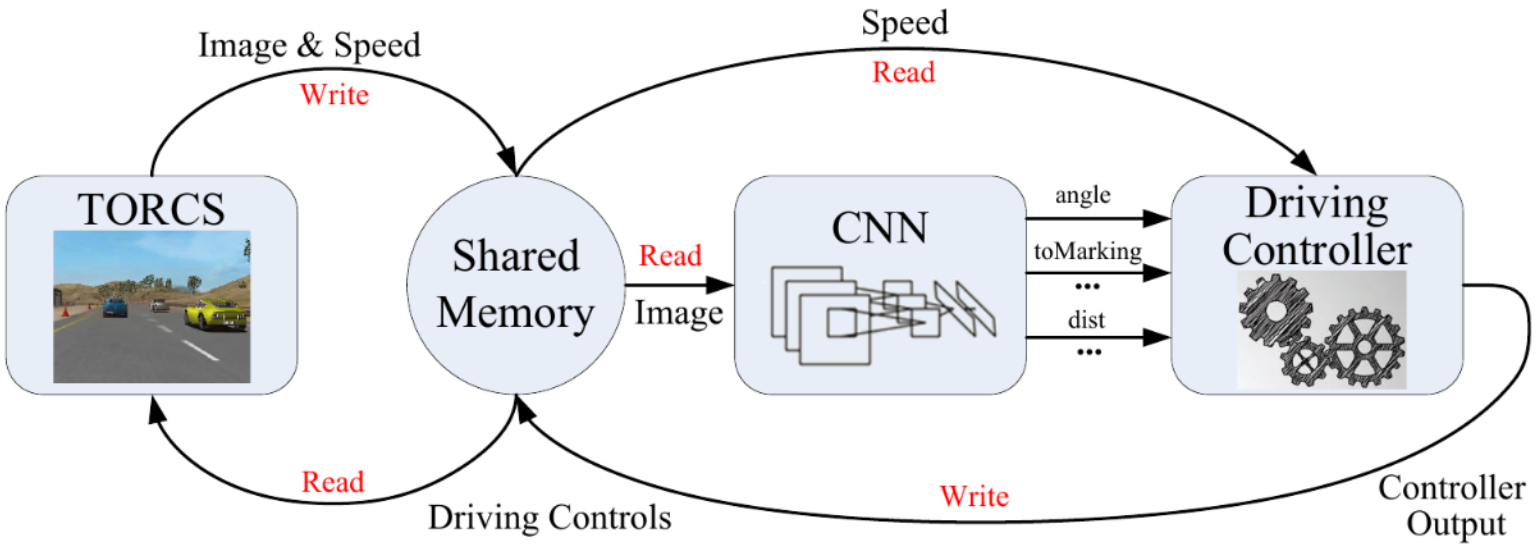
## And feeding them to a driving controller

→ Using the TORCS simulator to gather data and simulate

→ Proposal of a simple CNN (AlexNet)

→ Training the affordance indicators

→ Using a state machine to drive the car



# Controller logic

## A state machine for driving the car (using the affordances)

**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 $desired\_speed$

**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

# Affordance Learning

## Results

PRINCETON UNIVERSITY

**Learning Affordance for Direct Perception in Autonomous Driving**

Chenyi Chen Ari Seff Alain Kornhauser Jianxiong Xiao

Princeton University

https://www.youtube.com/watch?v=5hFvoXV9gII&t=2s

# Conditional Affordance Learning (CAL)

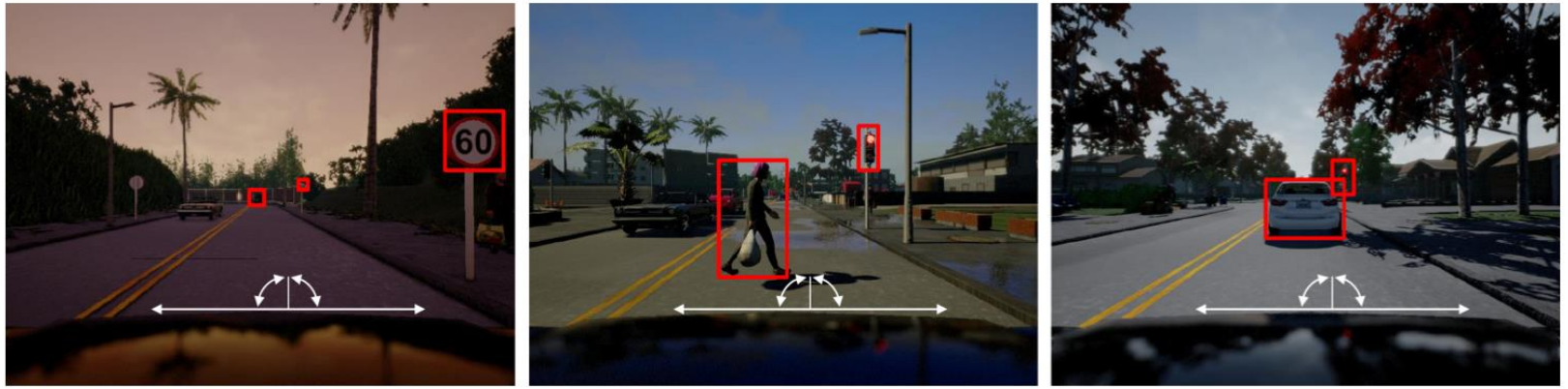
**Same idea, ported to urban environments and adding a condition**

→ Using Carla simulator

→ Goal: driving from A to B fast, safely and comfortably

→ Infractions:

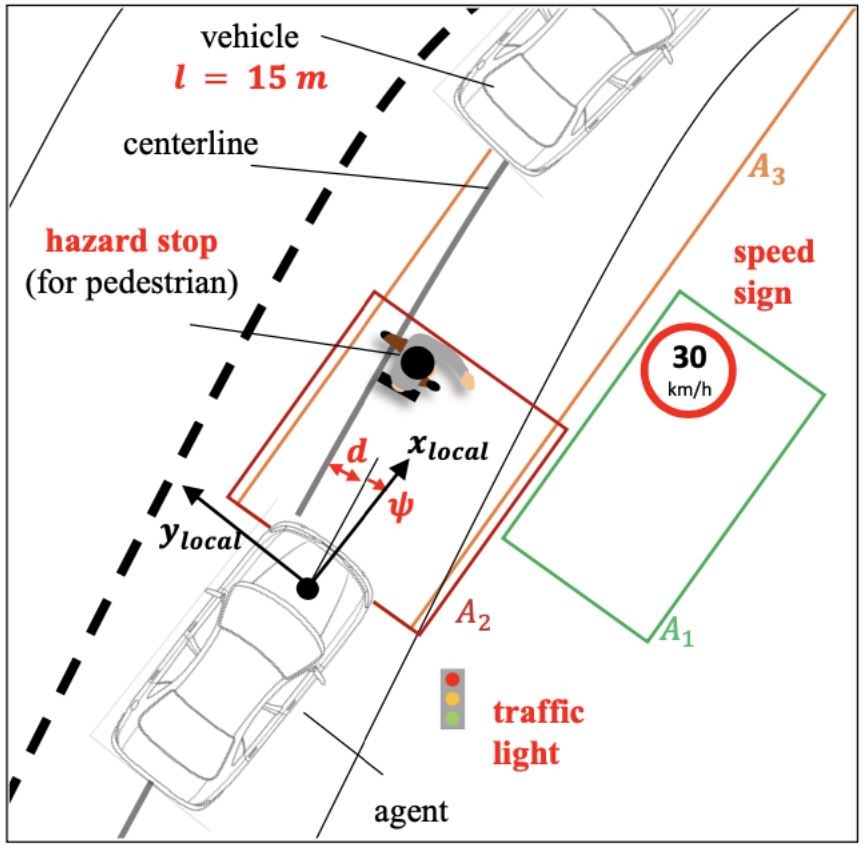
- driving on wrong lane, driving on sidewalk, running on a red light, violating speed limits, colliding with vehicles, hitting objects/pedestrians



# CAL in urban environments

## Affordances

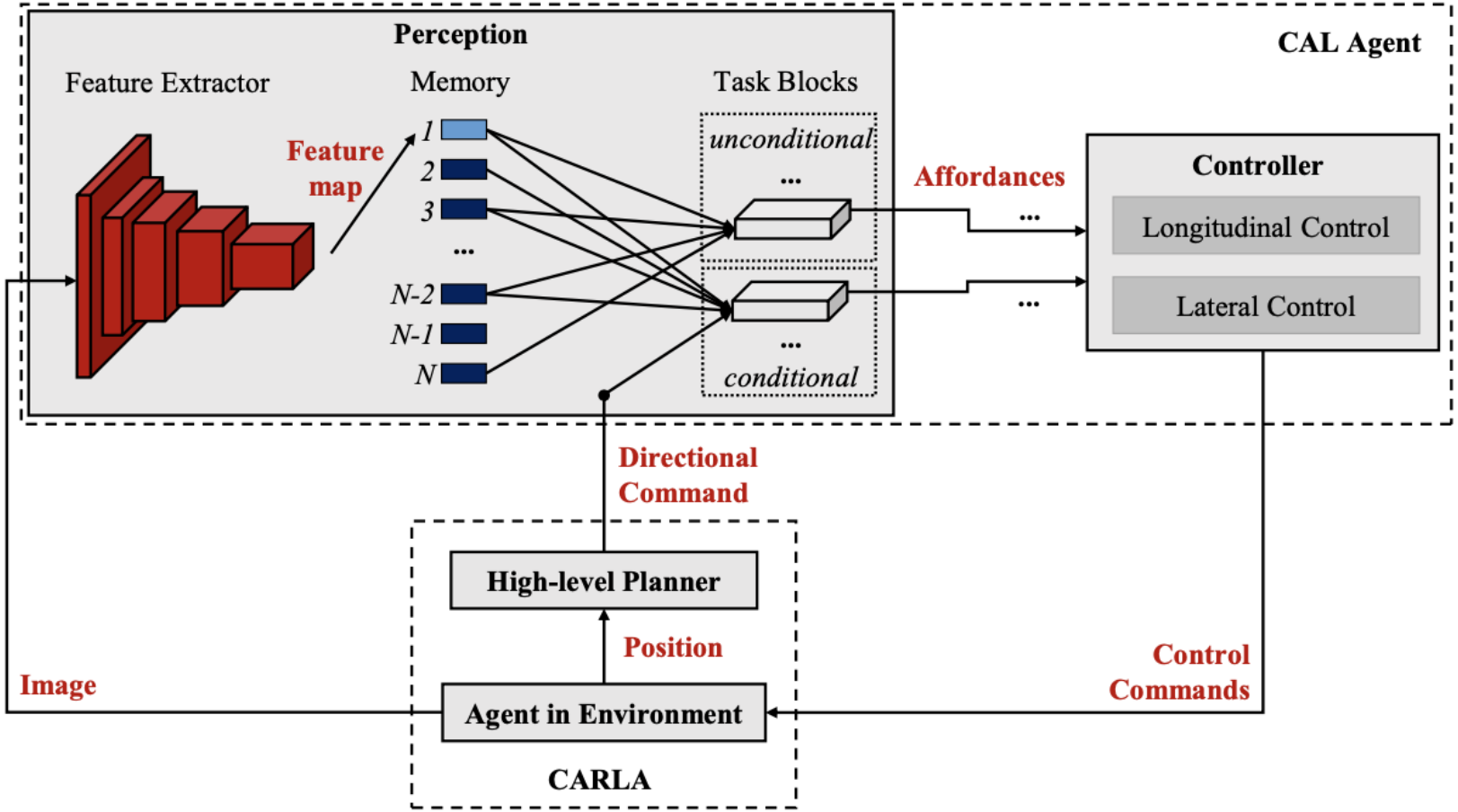
→ Distance to centerline
→ Relative angle to the road
→ Distance to lead vehicle
→ Speed signs
→ Traffic lights
→ Hazard stop (pedestrian)



# CAL in urban environments

## Overview of the solution

Images from car camera(s)



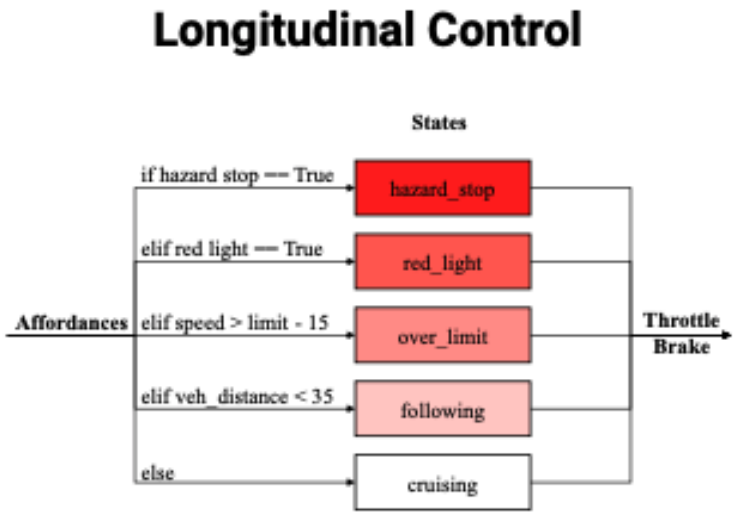
# How is the controller block implemented?

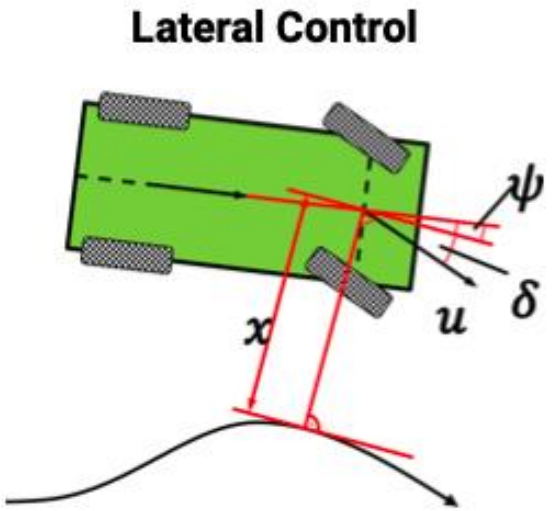
## Longitudinal and lateral control

→ Can be implemented in “the traditional way” (vehicle control and dynamics lecture!)

→ Longitudinal control
- Finite State Machine
- PID controller for cruising
- Car following model

→ Lateral control
- Stanley controller



# Conditional Affordance Learning

## Results


https://github.com/xl-sr/CAL

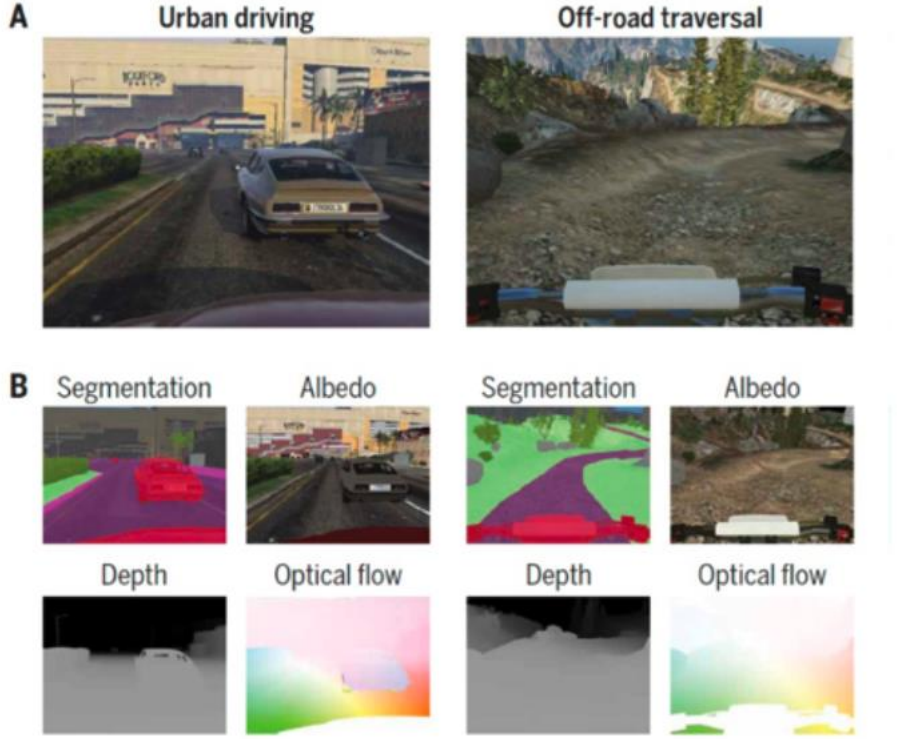
# Visual abstractions

## Better “intermediate representations” improve results

→ Does computer vision matter for action?
- Yes, it does. Better intermediate representations should improve the results.

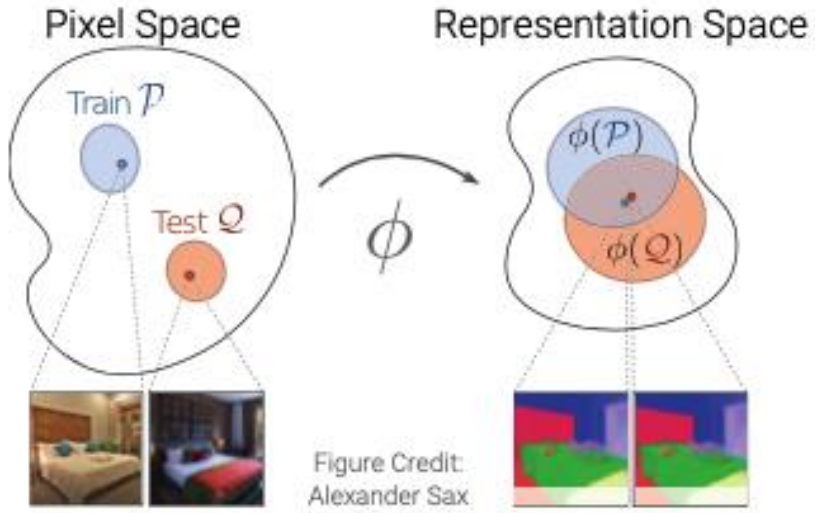
→ Several different intermediate representations:
- Image segmentation, depth, optical flow…

→ Depth and semantic segmentation often provide better results
- That is, they generalize better



# What is a good visual abstraction?

## What does better generalization mean?




→ A good visual abstraction is:

- Invariant : hides irrelevant variations
- Universal : applicable to a wide range of scenarios
- Data efficient in memory/computation
- Label efficient: requires little manual labelling effort

→ Semantic segmentation:

- Encodes task-relevant knowledge and groupings
- Can be processed with standard CNNs
- But… labelling time can become huge: ~90min for 1 cityscapes image

→ Trade-off when labelling

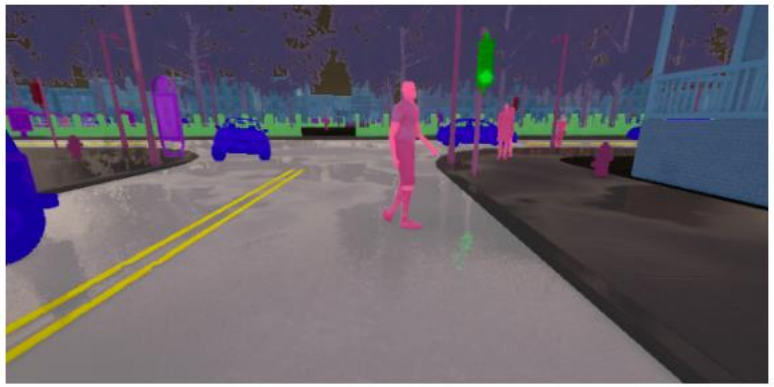
https://arxiv.org/pdf/1905.12887.pdf

# Identifying the most relevant classes

## Different number of classes for object detection and segmentation

→ Visual abstraction is an abstraction:

- Algorithm performance not necessarily improves with number of classes



14 classes

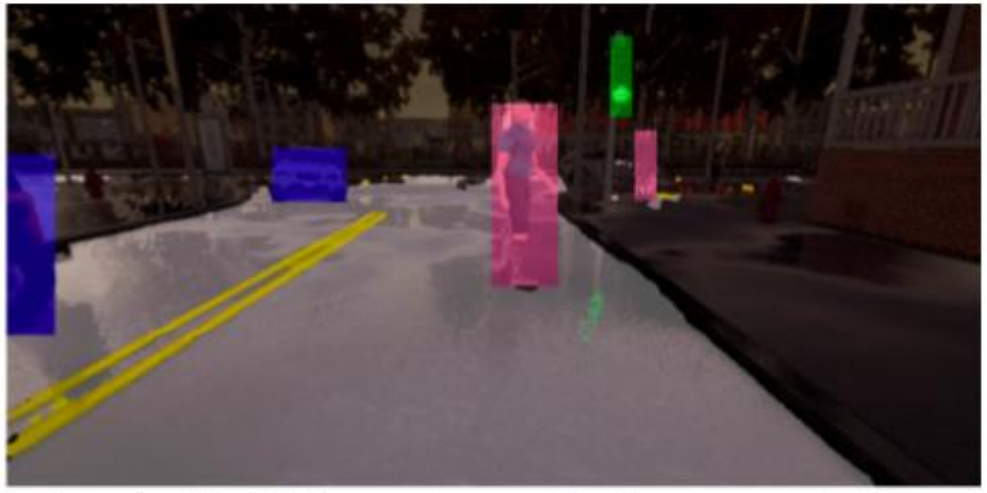
6 classes

14 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, greenlight, redlight.

# Training time and annotation time

**Similar (and even better) success rate for simplified classification**

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



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

https://arxiv.org/pdf/2005.10091.pdf

# TODOs for today

## Exercises

1. Analysis of the Dronet running on the Crazyflie (papers provided on Cyberlearn)

2. Conditional Imitation Learning

3. Conditional Affordance Learning


Hes·so
Haute Ecole Spécialisée
de Suisse occidentale