DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
UNIVERSIDAD DE CHILE

# EL7021: Deep Drive
# Computer Vision Applications for Autonomous Cars


Departamento de Ingeniería Eléctrica
Advanced Mining Technology Center (AMTC)
Universidad de Chile

# Computer Vision in Autonomous Cars


# Computer Vision in Autonomous Cars

Three main areas of research:

- Scene understanding.
- State estimation.
- Autonomous Driving.

# Scene Understanding

**Objective:** Provide environment related information to produce correct driving behavior.

**Tasks:**

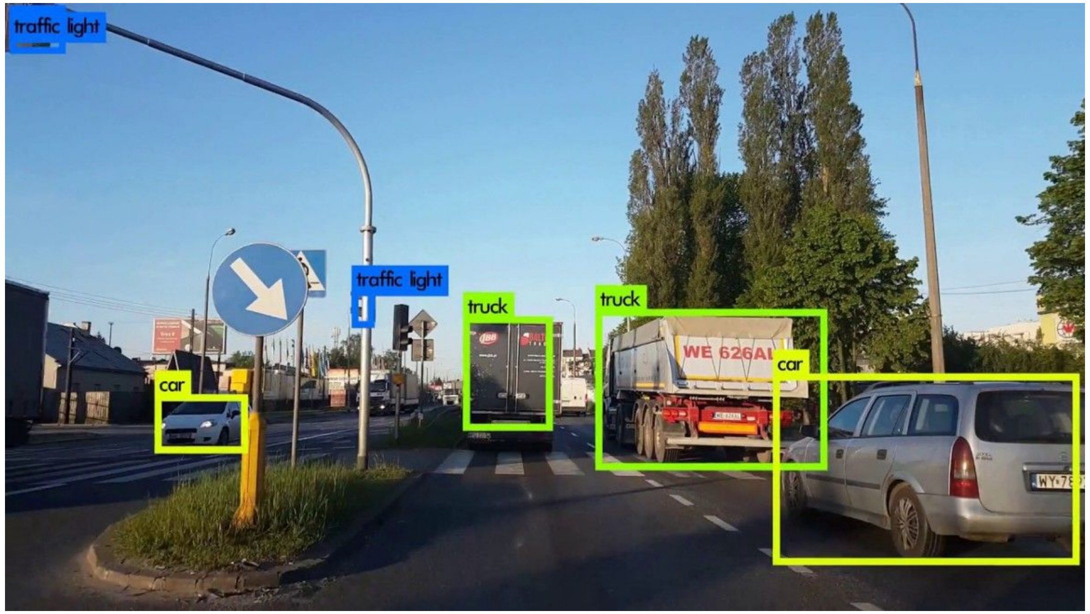
- Object Recognition
- Semantic Segmentation

Static nature allows the use of image databases (test performance holds as long as the test data distributes similar)

# YOLO: Real-Time Object Detection

Object recognition was previously splitted between **object detection** and **object classification.**

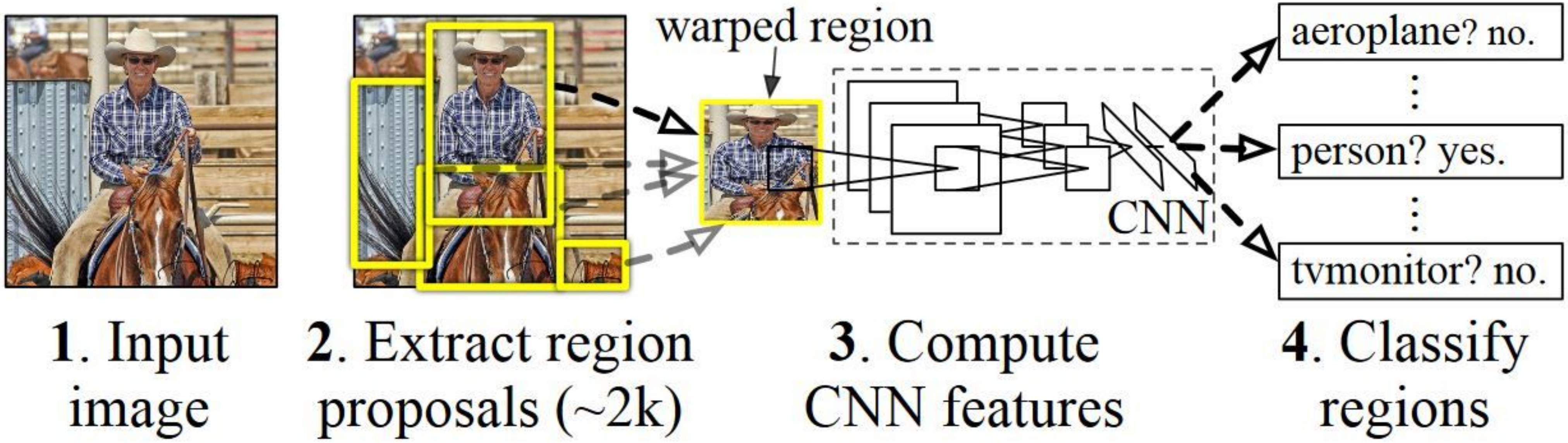
Brute force sliding windows cannot be used in real time, since performance and execution times depend strongly on the window stride.



# YOLO: Real-Time Object Detection

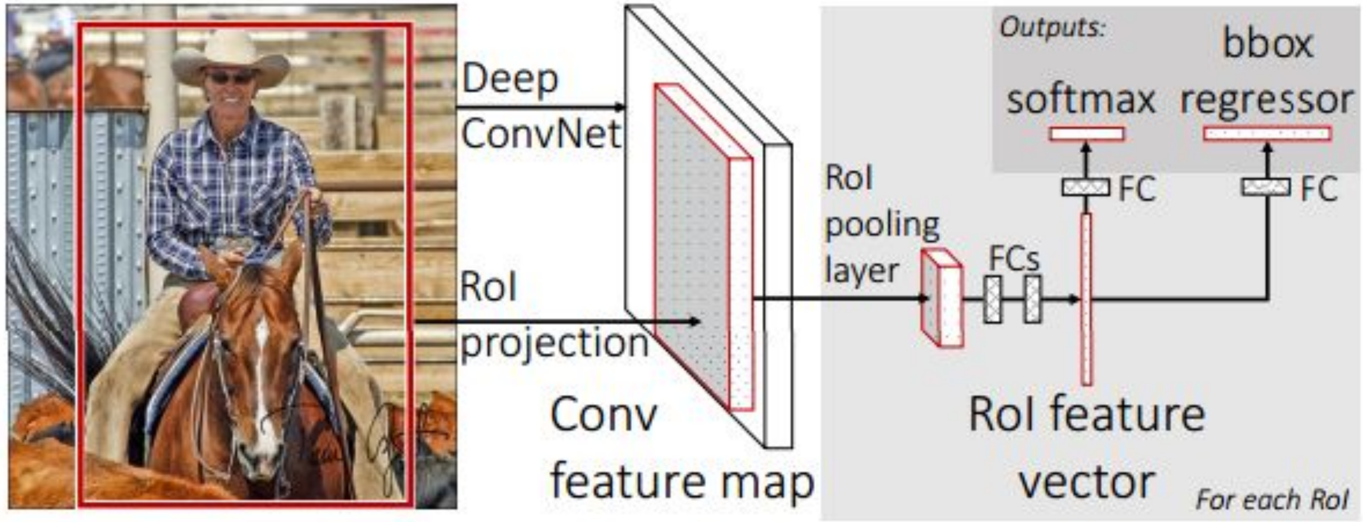
| | R-CNN | Fast R-CNN | Faster R-CNN | YOLO |
|---|---|---|---|---|
| Embedded ROI Generation | ✗ | ✗ | ✓ | ✓ |
| # Inferences / Image | 2000 | 1 | 1 | 1 |
| End-to-end | ✗ | ✗ | ✓ | ✓ |
| Global Information | ✗ | ✗ | ✗ | ✓ |

# R-CNN

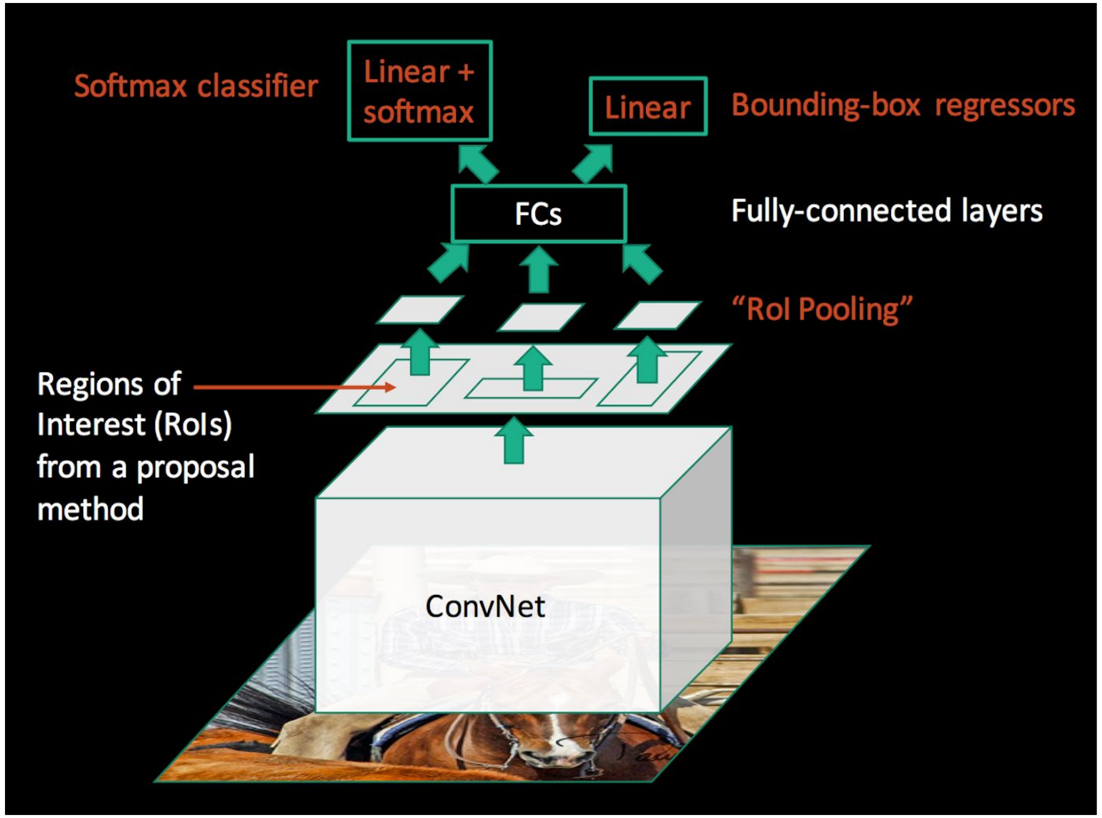

warped region
aeroplane? no.
person? yes.
tvmonitor? no.
CNN
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

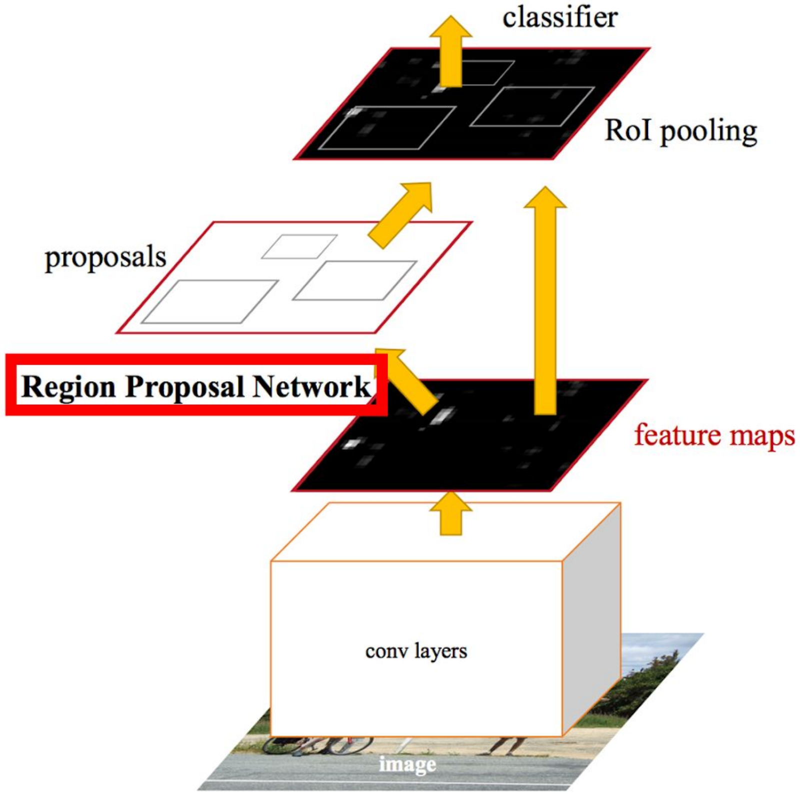
# R-CNN*

# Fast R-CNN

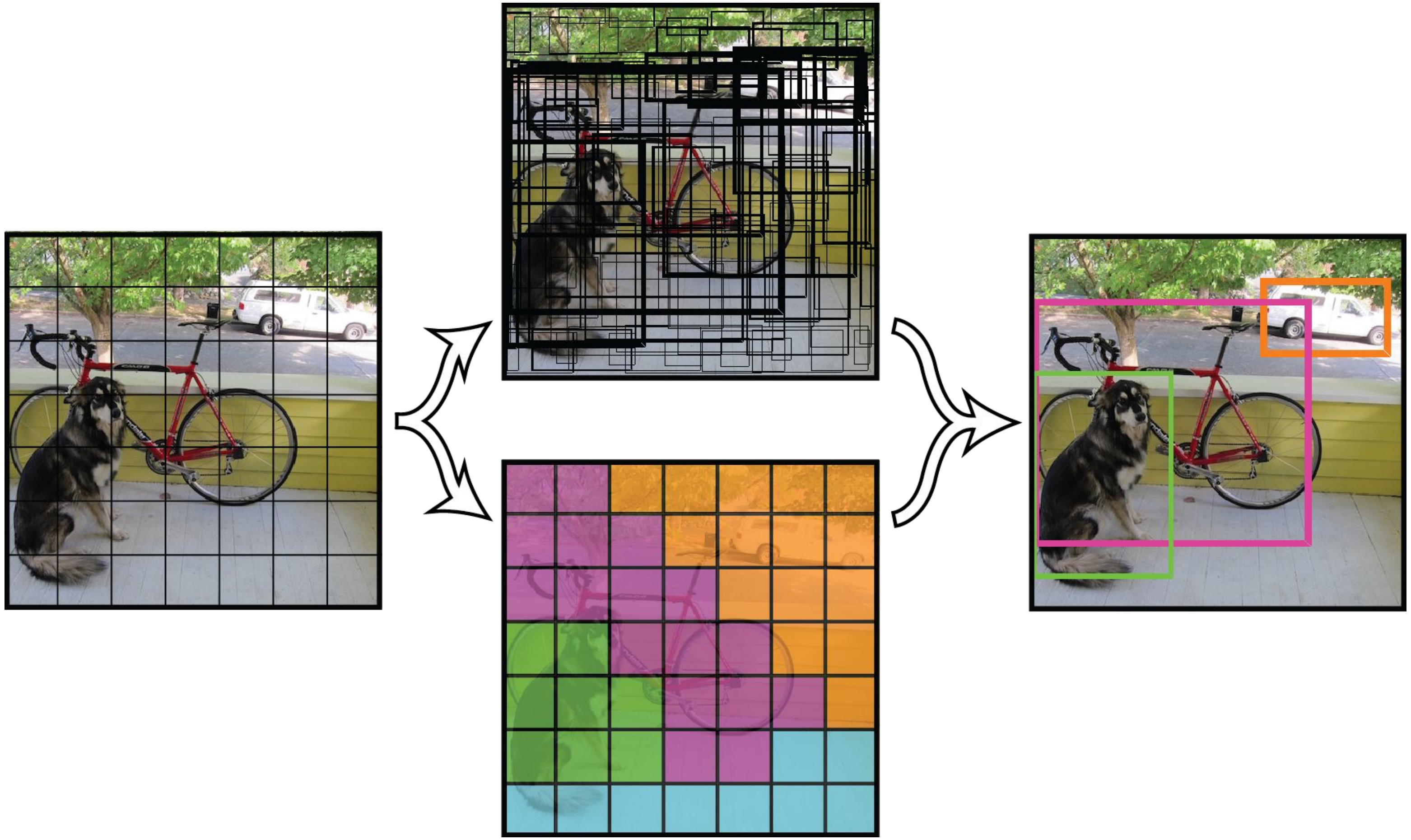

Deep
ConvNet
RoI
projection
Conv
feature map
RoI
pooling
layer
FCs
Outputs:
softmax
bbox
regressor
FC
FC
RoI feature
vector
For each RoI

# Faster R-CNN

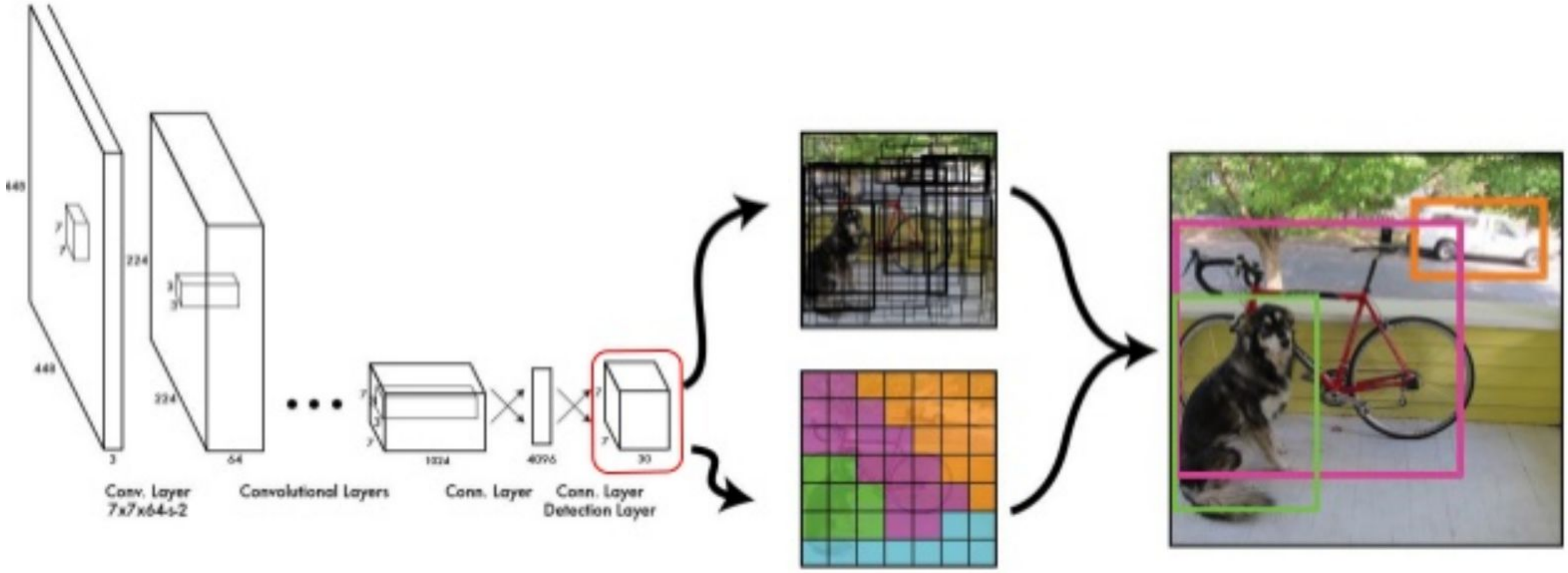

Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling"
Regions of Interest (RoIs) from a proposal method
ConvNet


classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image

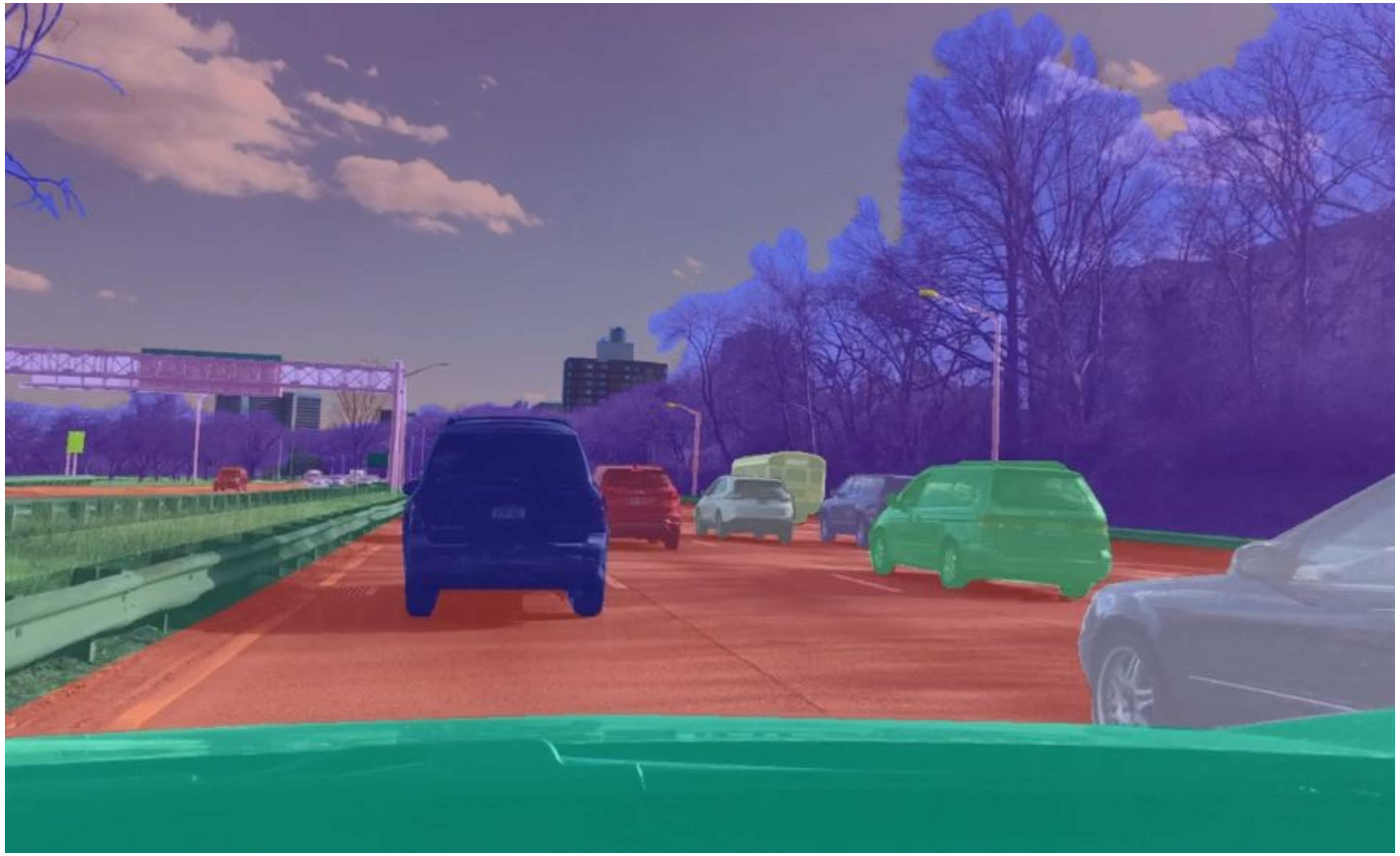
# YOLO v1: Real-Time Object Detection



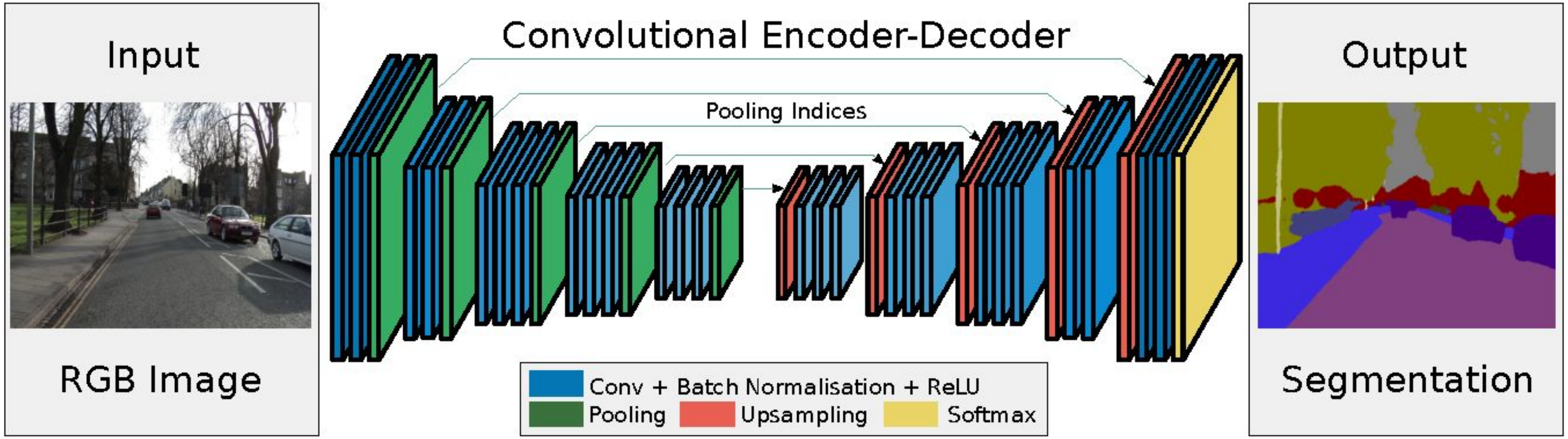
# YOLO v1: Real-Time Object Detection



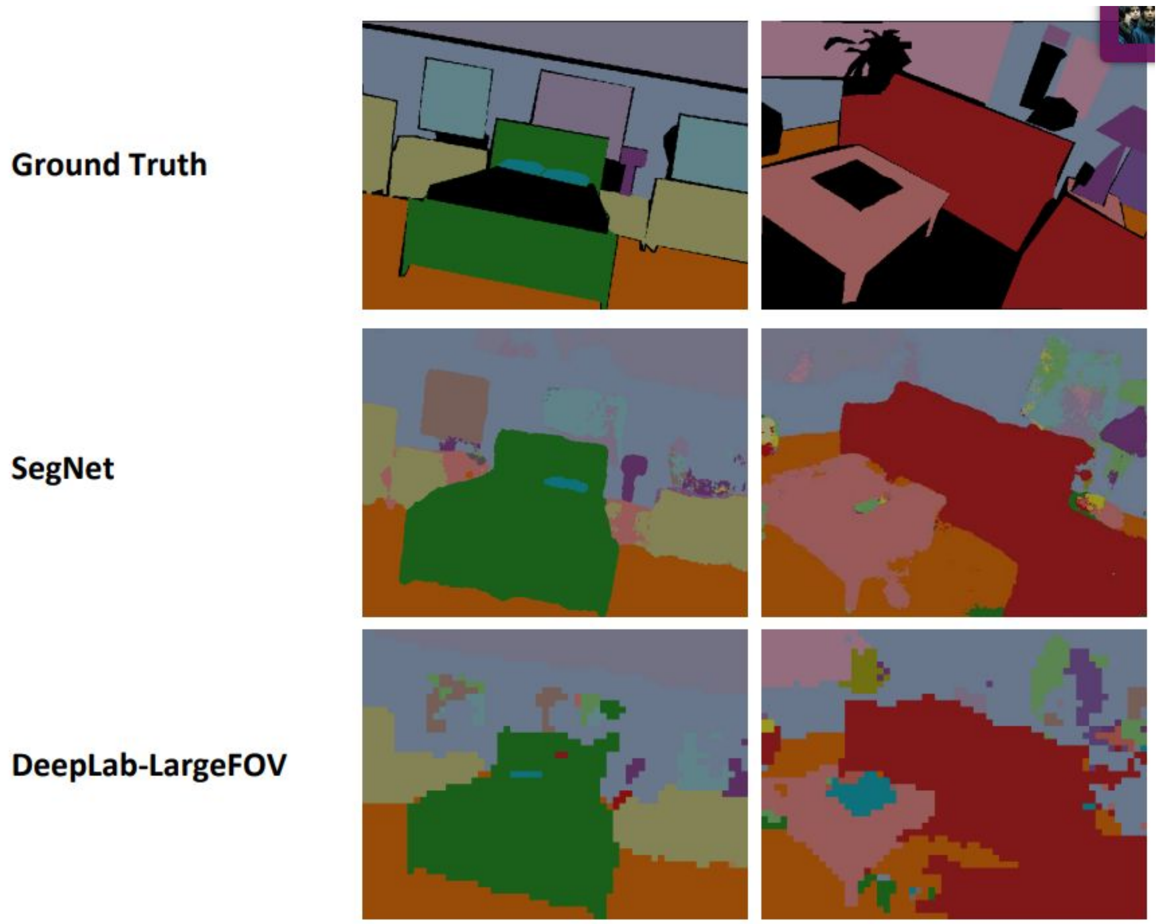
# SegNet: A Deep Encoder-Decoder Architecture for Image Segmentation



# SegNet: A Deep Encoder-Decoder Architecture for Image Segmentation



# SegNet: A Deep Encoder-Decoder Architecture for Image Segmentation



# State Estimation

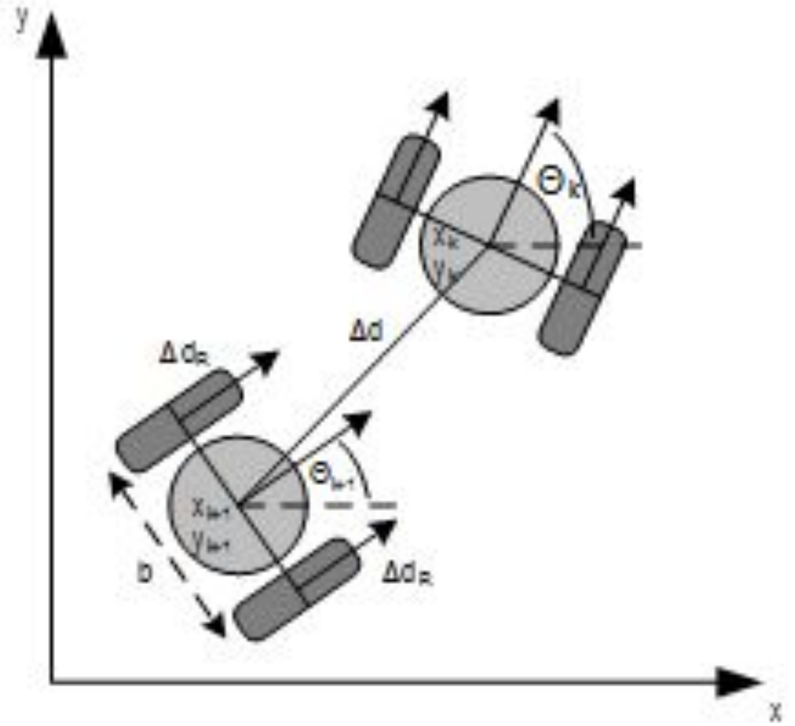
**Objective:** Integrate visual information to produce state related information.

**Tasks:**

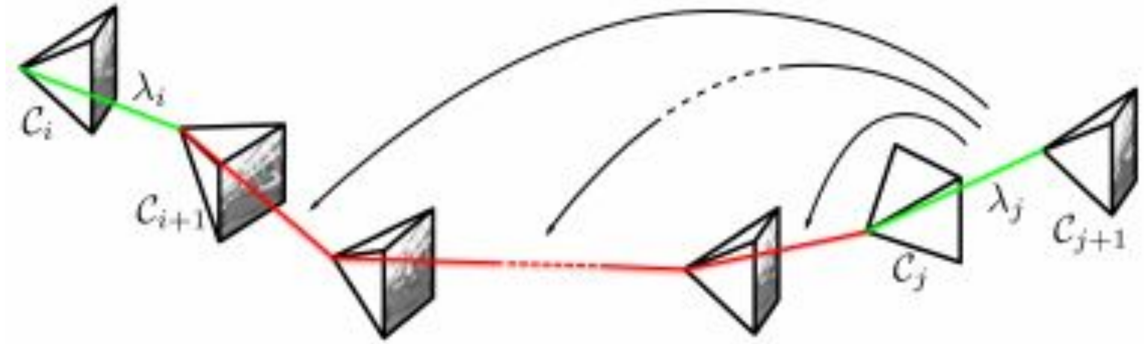
- Visual Odometry
- Visual SLAM

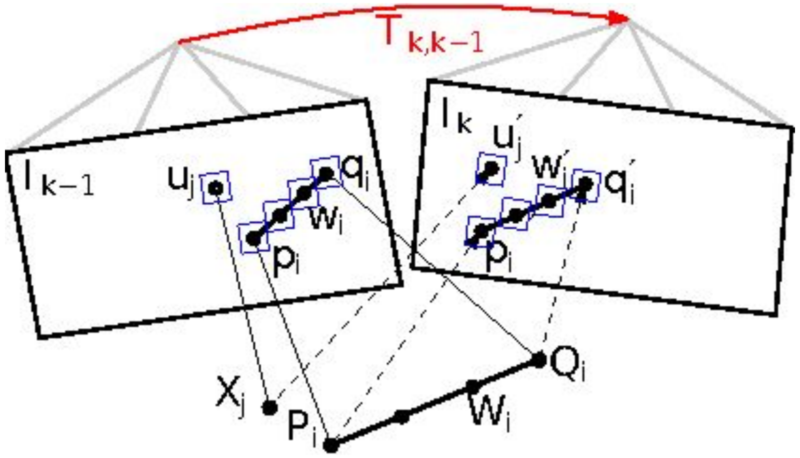
Assumes markov processes. However, since actions are independent from the system, video databases can be used.

# Visual Odometry

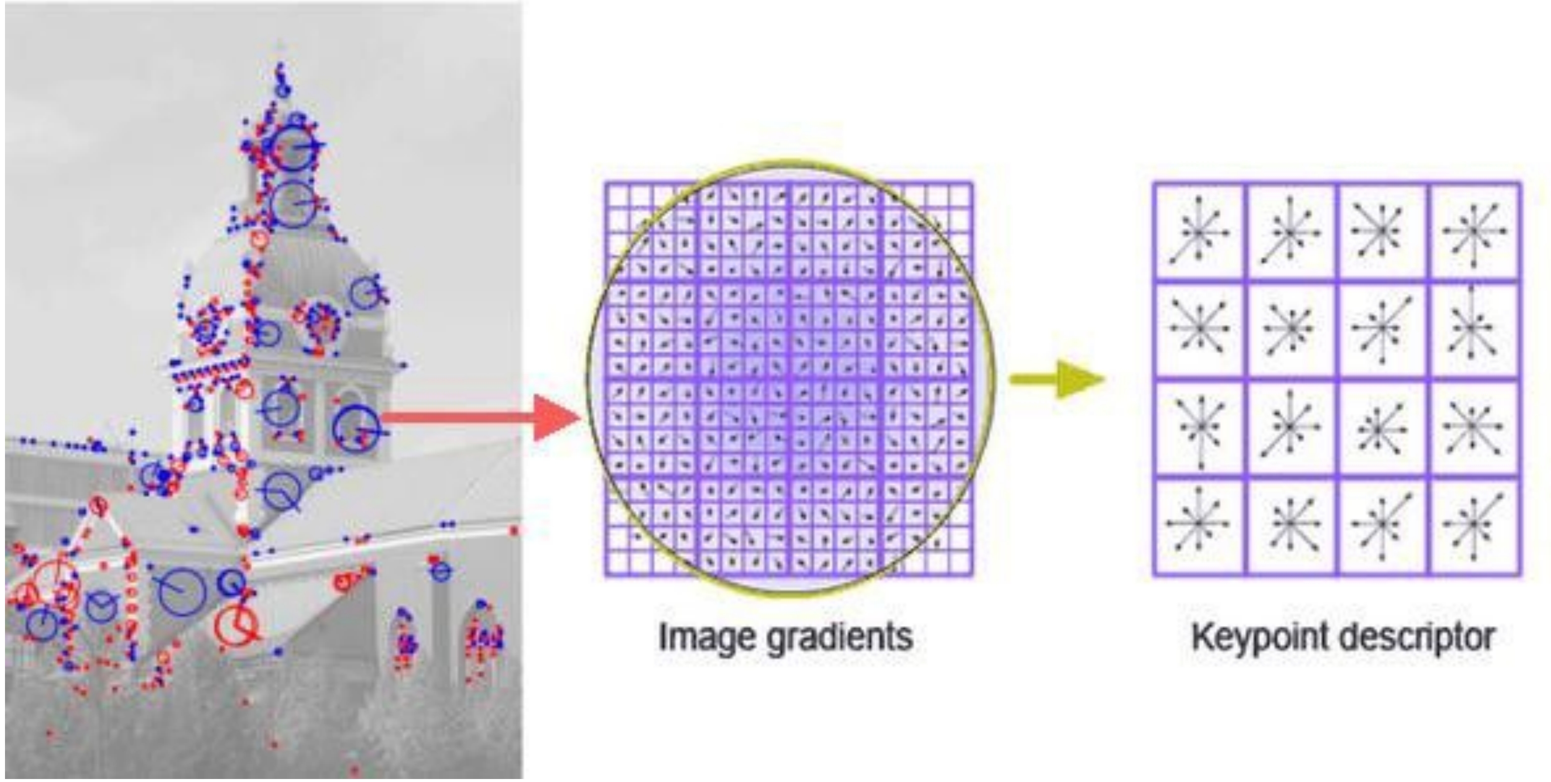


# Visual Odometry

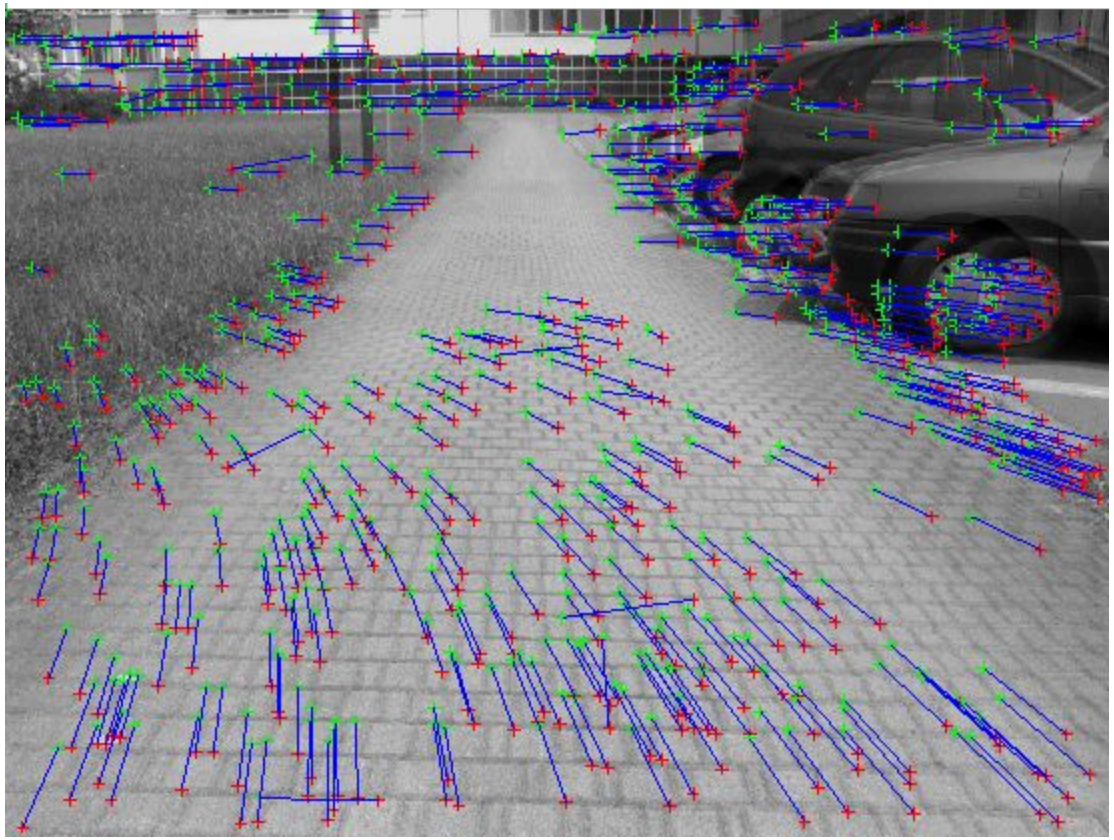


# Visual Odometry



# Visual Odometry



# Visual SLAM

# Autonomous Driving

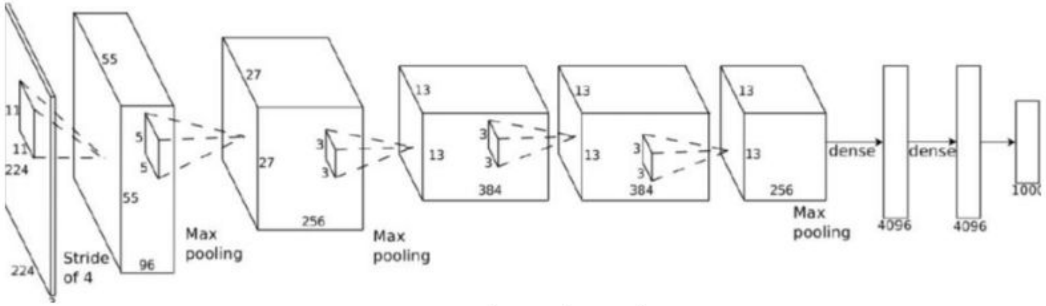
**Objective:** Produce steering actions for autonomous driving.

The most integrated and complex task for autonomous car.
Several approaches exist.

Assumes markov processes. The system's actions affect the chain (data distribution). The use of databases is NOT representative of test performance.

# Visual Navigation (Imitation Learning)

$\mathbf{o}_t$




$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

$\mathbf{u}_t$

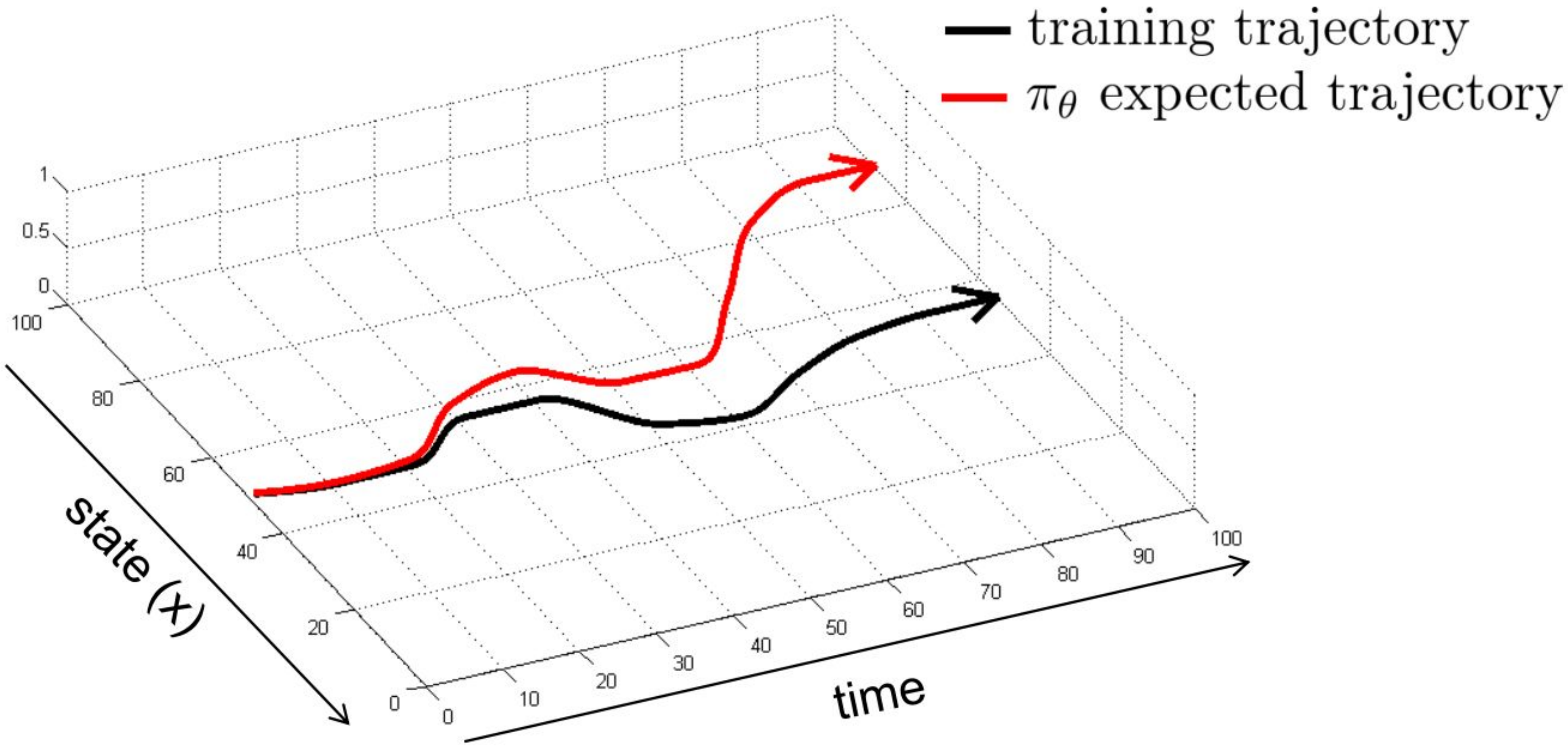
# Visual Navigation (Imitation Learning)

$\mathbf{o}_t$
$\mathbf{u}_t$

→ training data → supervised learning

$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

# Visual Navigation (Imitation Learning)



# Visual Navigation (Imitation Learning)

## Why does it not work ???

Policy $\pi(u|o,\theta)$ was trained to copy $\pi_{train}(u|o)$ under $p_{train}(x)$, more specifically $x_t \sim p(x_{t-1}, \mathbf{u_{t-1}})$.

Since during test $u \sim \pi(u|o,\theta)$, the state distributions are expected to differ. This implies that test performance is usually really low (especially, in high dimensional problems).

# Visual Navigation (Imitation Learning)

## DAgger: Dataset Aggregation

goal: collect training data from $p_{\pi_\theta}(\mathbf{o}_t)$ instead of $p_{\text{data}}(\mathbf{o}_t)$

how? just run $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

but need labels $\mathbf{u}_t$!

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

# Deep Learning Frameworks

**Tensorflow**

C++ written. Provides an API for several languages.

Mature library. Active community.

Focus on deployment

**PyTorch**

C++ written. Python interface.

Relatively new. Growing community.

Similar to numpy, easy to use.

**Darknet**

C written. C, C++, and python interfaces.

Lightweight and the easiest to deploy.

Single contributor.