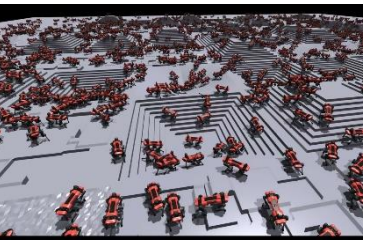
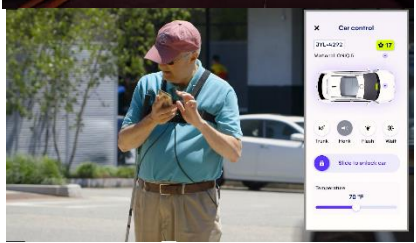
EC500: Robot Learning and Vision for Navigation



Eshed Ohn-Bar

Feb 15, 2023



Poll

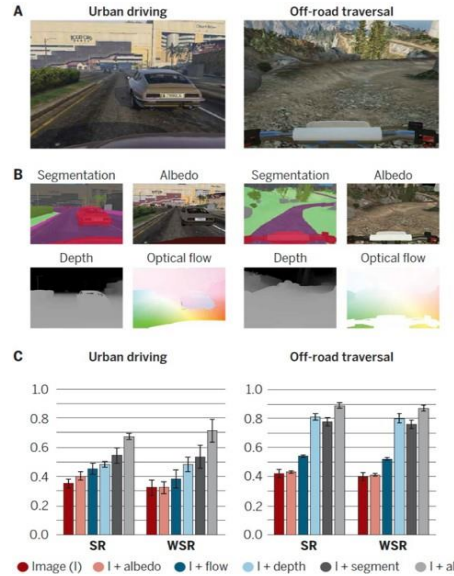
Recap

- History and technological revolutions
- Paradigms for sensorimotor learning
- Deep imitation learning
- Behavior cloning and catastrophic failure
- Dagger and data augmentation methods
- Direct perception
- **Today**: Starting Modular Pipelines

Does Computer Vision Matter for Action?

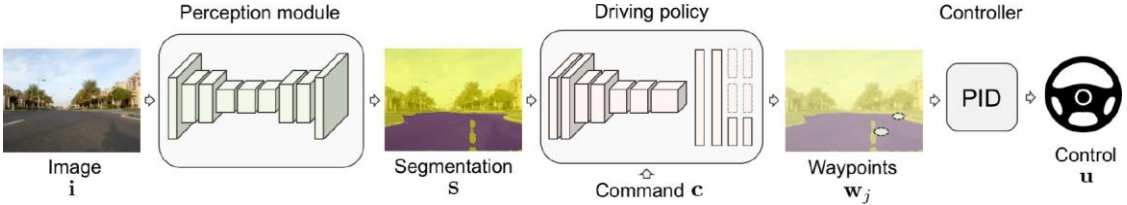
Does Computer Vision Matter for Action?

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



Zhou, Kra¨henbu¨hl and Koltun: Does computer vision matter for action? Science Robotics, 2019.

Driving Policy Transfer: Overview



Problem:

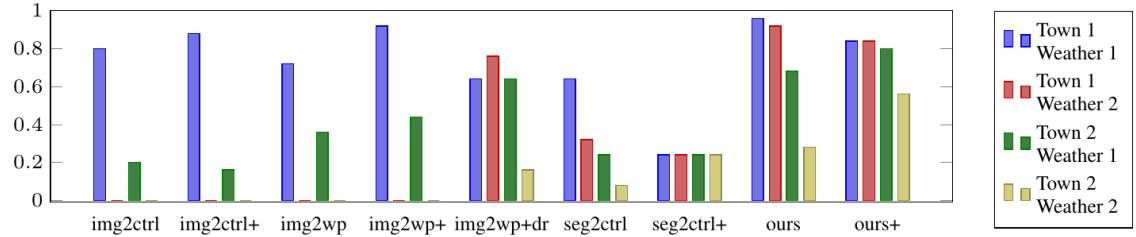
- Driving policies learned in simulation often do not transfer well to the real world

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

Müller, Dosovitskiy, Ghanem and Koltun: Driving Policy Transfer via Modularity and Abstraction. CoRL, 2018.

Driving Policy Transfer: Results

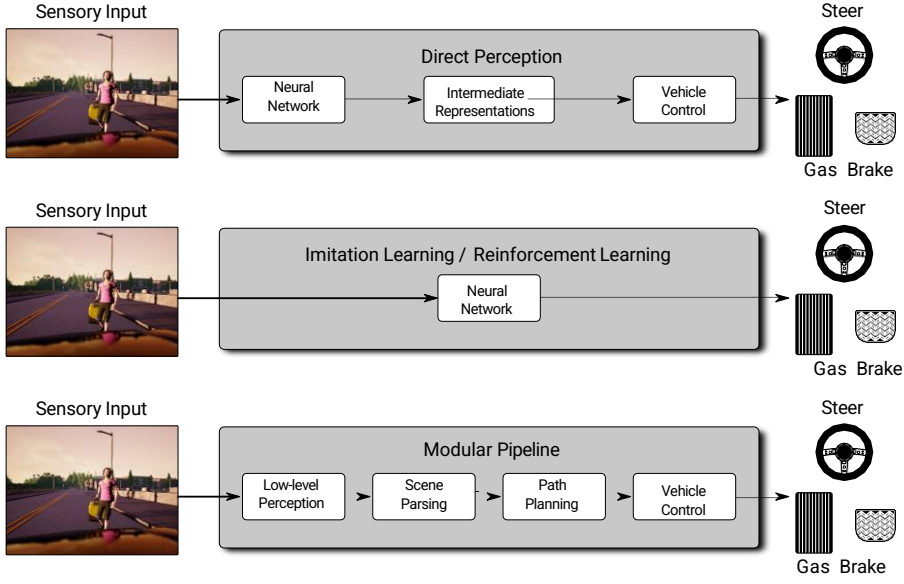


Success Rate over 25 Navigation Trials

- Driving policy: Conditional Imitation Learning (branched)
- Training: Expert agent, random initialization, noise injection
- Control: PID for lateral and longitudinal control
- Results: Full method generalizes best ("+" = with data augmentation)

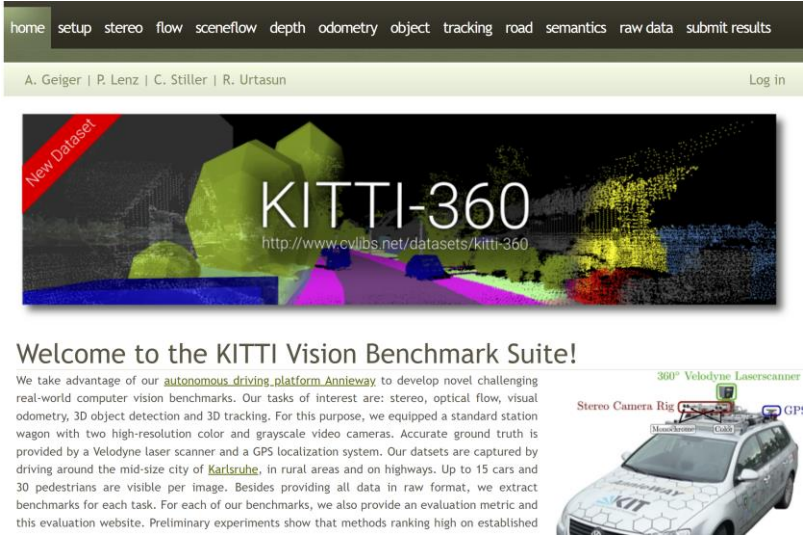
Müller, Dosovitskiy, Ghanem and Koltun: Driving Policy Transfer via Modularity and Abstraction. CoRL, 2018.

Approaches to Self-Driving



KITTI Dataset

https://www.cvlibs.net/datasets/kitti/



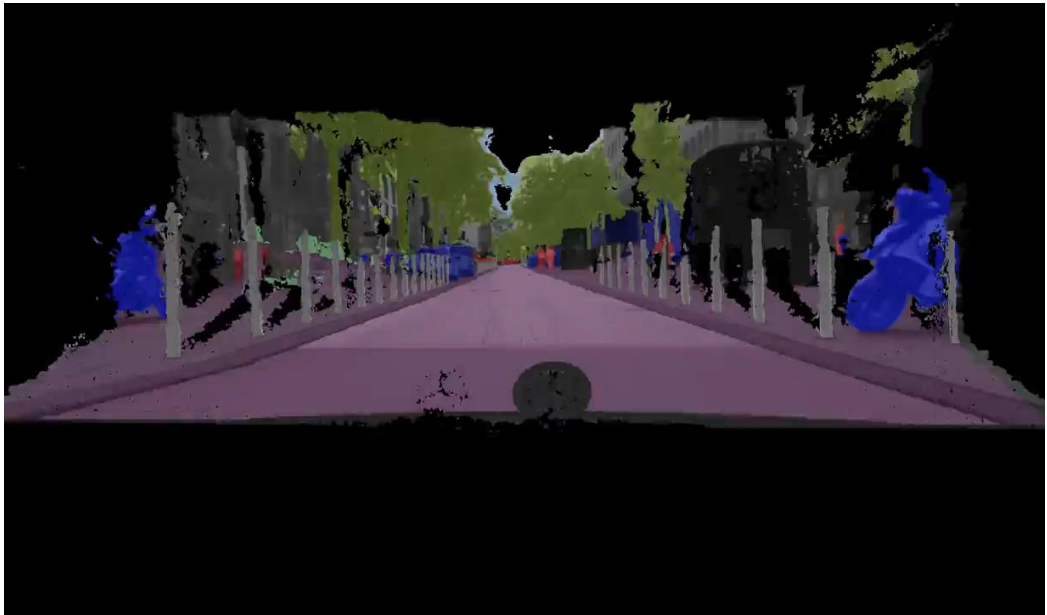
https://www.cvlibs.net/datasets/kitti/

Object Detection and Orientation Estimation Evaluation

Cars

	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment	Compare
1	VoCo			**97.11 %**	98.25 %	**94.46 %**	0.1 s	1 core @ 2.5 Ghz (Python + C/C++)	☐
2	VirConv-S			96.46 %	96.99 %	93.74 %	0.09 s	1 core @ 2.5 Ghz (C/C++)	☐
3	GraR-Vol		code	96.29 %	96.81 %	91.06 %	0.07 s	1 core @ 2.5 Ghz (Python + C/C++)	☐
	H. Yang, Z. Liu, X. Wu, W. Wang, W. Qian, X. He and D. Cai: Graph R-CNN: Towards Accurate 3D Object Detection with Semantic-Decorated Local Graph. ECCV 2022.								
4	GraR-Po		code	96.09 %	96.83 %	90.99 %	0.06 s	1 core @ 2.5 Ghz (Python + C/C++)	☐
	H. Yang, Z. Liu, X. Wu, W. Wang, W. Qian, X. He and D. Cai: Graph R-CNN: Towards Accurate 3D Object Detection with Semantic-Decorated Local Graph. ECCV 2022.								
5	NIV-SSD			96.06 %	96.89 %	88.63 %	0.03 s	1 core @ 2.5 Ghz (C/C++)	☐
6	SFD		code	96.05 %	**98.95 %**	90.96 %	0.1 s	1 core @ 2.5 Ghz (Python + C/C++)	☐
	X. Wu, L. Peng, H. Yang, L. Xie, C. Huang, C. Deng, H. Liu and D. Cai: Sparse Fuse Dense: Towards High Quality 3D Detection with Depth Completion. CVPR 2022.								
7	VPFNet		code	96.04 %	96.63 %	90.99 %	0.06 s	2 cores @ 2.5 Ghz (Python)	☐
	H. Zhu, J. Deng, Y. Zhang, J. Ji, Q. Mao, H. Li and Y. Zhang: VPFNet: Improving 3D Object Detection with Virtual Point based LiDAR and Stereo Data Fusion. IEEE Transactions on Multimedia 2022.								
8	VirConv-T			96.01 %	98.64 %	93.12 %	0.09 s	1 core @ 2.5 Ghz (C/C++)	☐
9	CASDC			95.97 %	98.64 %	92.99 %	0.1 s	1 core @ 2.5 Ghz (Python)	☐
10	TED		code	95.96 %	96.63 %	93.24 %	0.1 s	1 core @ 2.5 Ghz (C/C++)	☐
	H. Wu, C. Wen, W. Li, R. Yang and C. Wang: Transformation-Equivariant 3D Object Detection for Autonomous Driving. AAAI 2023.								
11	RDIoU		code	95.95 %	98.77 %	90.90 %	0.03 s	1 core @ 2.5 Ghz (Python + C/C++)	☐
	H. Sheng, S. Cai, N. Zhao, B. Deng, J. Huang, X. Hua, M. Zhao and G. Lee: Rethinking IoU-based Optimization for Single- stage 3D Object Detection. ECCV 2022.								
12	ACF-Net			95.95 %	96.64 %	93.17 %	n/a s	1 core @ 2.5 Ghz (C/C++)	☐
13	CLOCs		code	95.93 %	96.77 %	90.93 %	0.1 s	1 core @ 2.5 Ghz (Python)	☐
	S. Pang, D. Morris and H. Radha: CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection . 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2020.								

Cityscapes Dataset



NuScenes Dataset

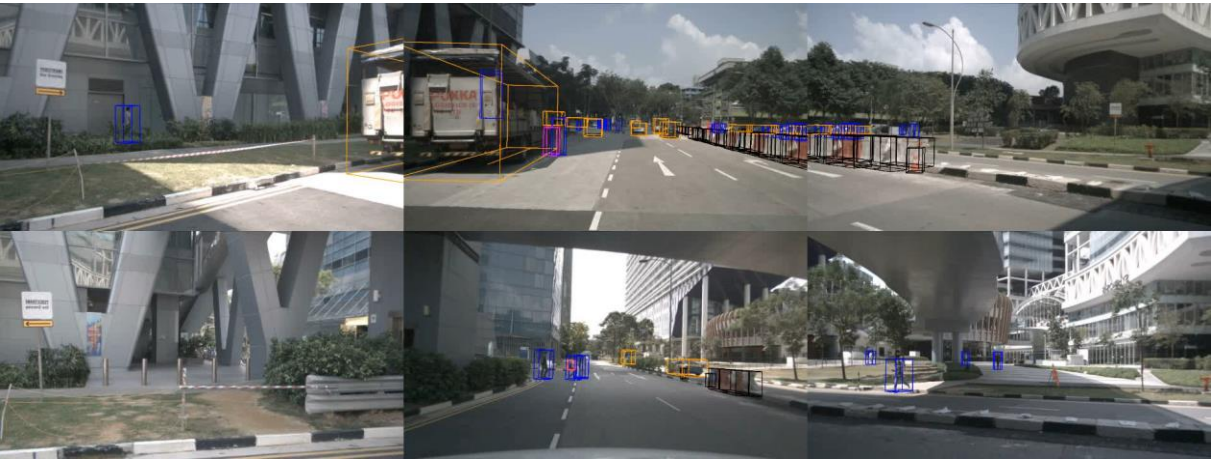
nuScenes:
A multimodal dataset for autonomous driving

H. Caesar, V. Bankit, A. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom

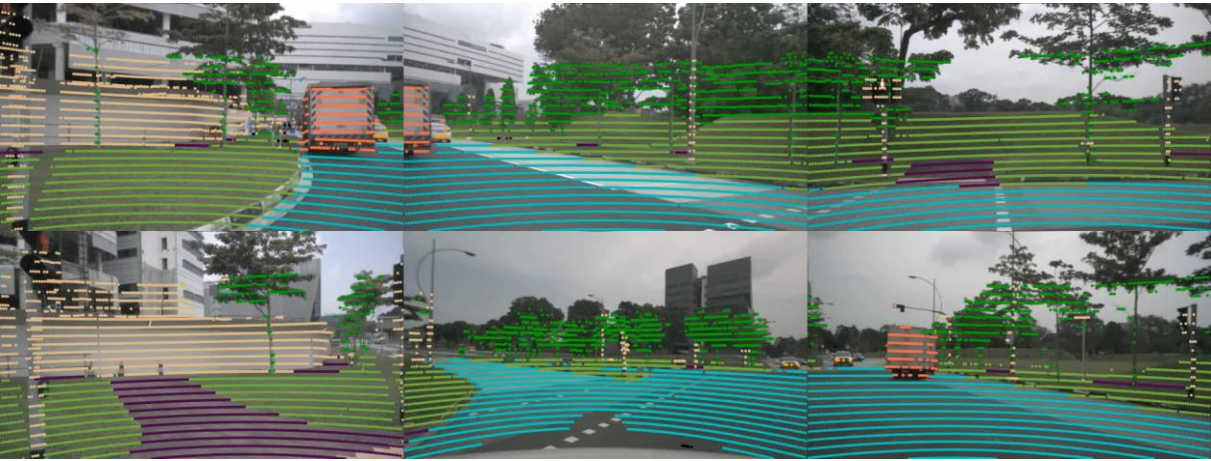
APTIV

NuScenes Dataset

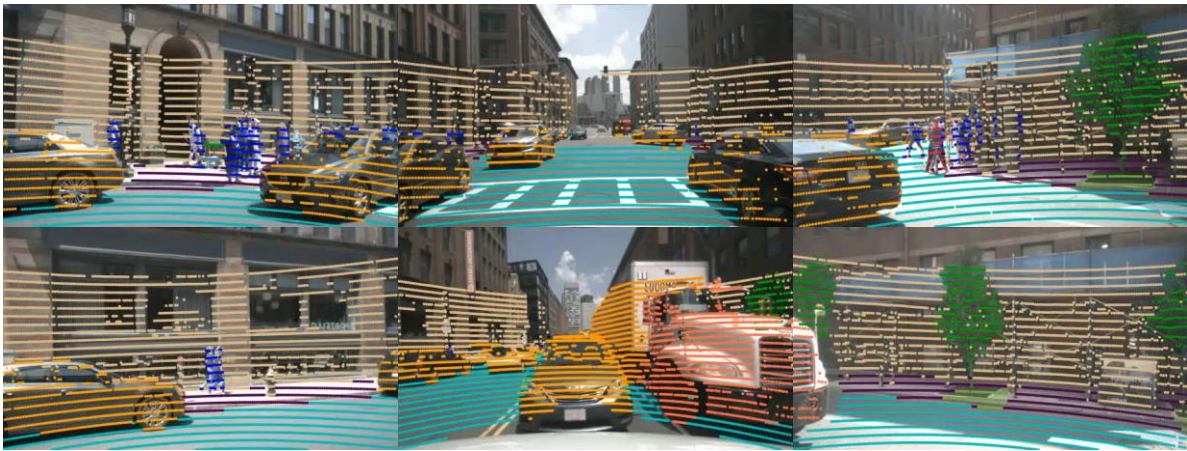
NuScenes Dataset



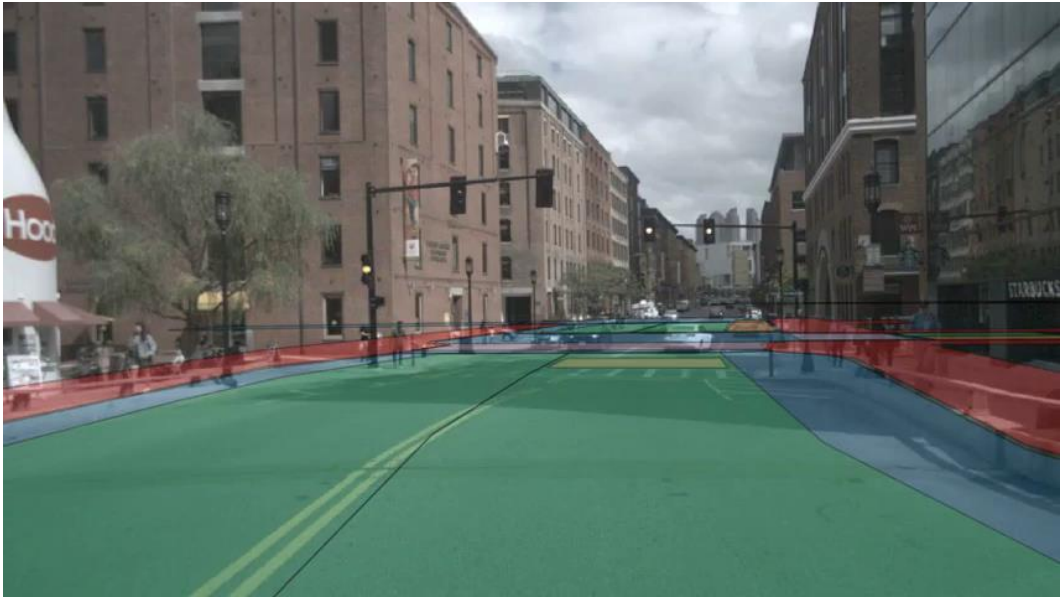
NuScenes Dataset



NuScenes Dataset



NuScenes Dataset



Road and Lane Detection

(some slides adopted from Andres Geiger)

- State-of-the-art commercial systems
- Representations
- Ambiguities
- Lane Marking Detection
- Lane Detection
- Road Segmentation
- Driving Corridor Prediction
- Freespace Estimation

Road and Lane Detection: State-of-the-Art

State-of-the-Art Commercial Systems:

- **Lane departure warning**
  Driver is warned when beginning to leave the current lane

- **Lane keeping support**
  Driver is assisted with minimal steering interventions

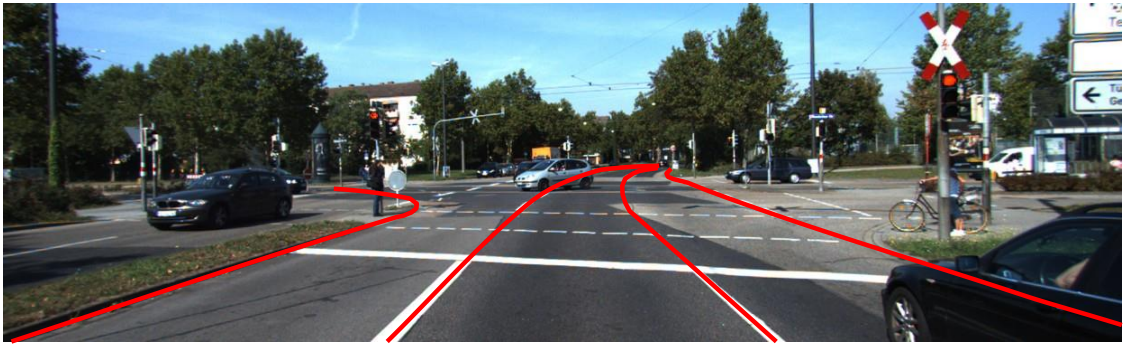
- **Lane keeping/lane departure protection**
  Car keeps the present lane autonomously

Road and Lane Detection: Representations

Road and Lane Detection:

- ► Navigate without detailed global map by sensing “drivable areas” in vicinity of car
- ► Multiple cues available (geometric, semantic, and man-made cues)
- ► Input: image / Lidar/ radar. Multiple possible **output representations**

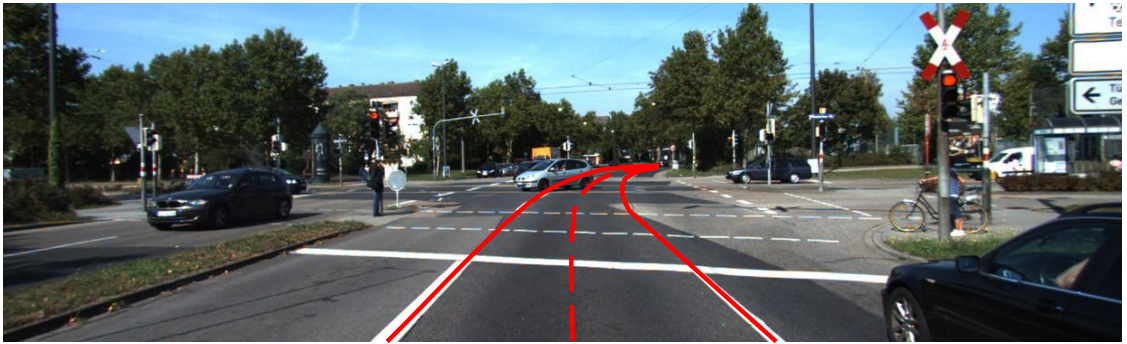
Road and Lane Detection: Representations



Lane Marking Detection:

- Roads are often subdivided into lanes (not everywhere) using lane markings
- Detect lane markings and road curbs; fit parametric model to them
- Steer vehicle based on distance to centerline / lane boundaries

Road and Lane Detection: Representations



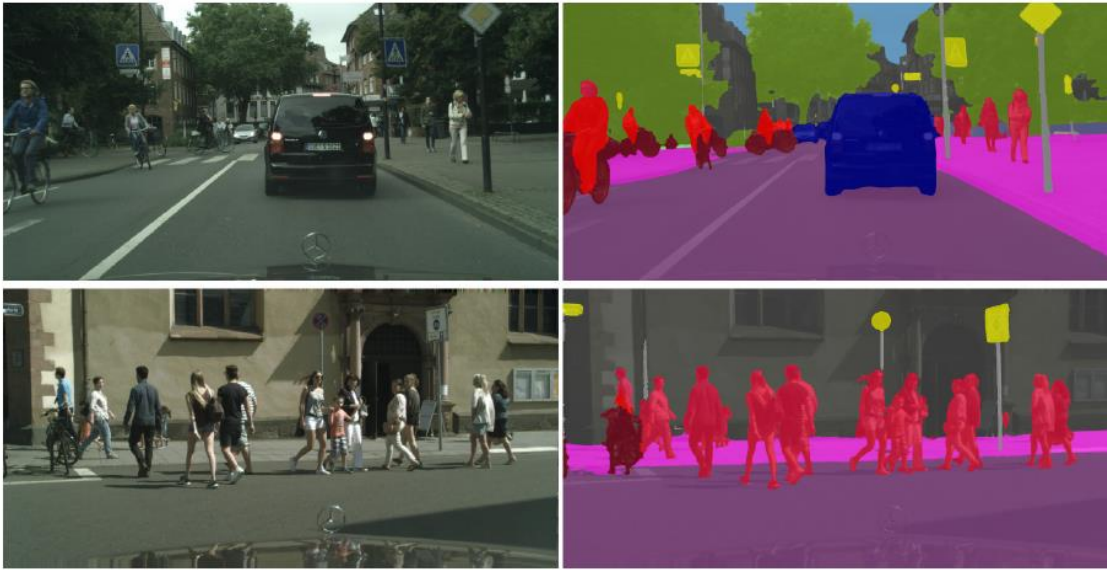
Lane Detection:

- Lane detection groups lane markings into lane
- Yields information about lane width and centerline (dashed)
- Navigation relevant lane depends on high-level plan

Road and Lane Detection: Representations

Road Segmentation:

- Classify each pixel in the image as either road or non-road
- Purely semantic cue ("where is a vehicle allowed to drive?")
- Doesn't consider "reachability" / geometry, no information about lanes



Road and Lane Detection: Representations

Driving Corridor Prediction:

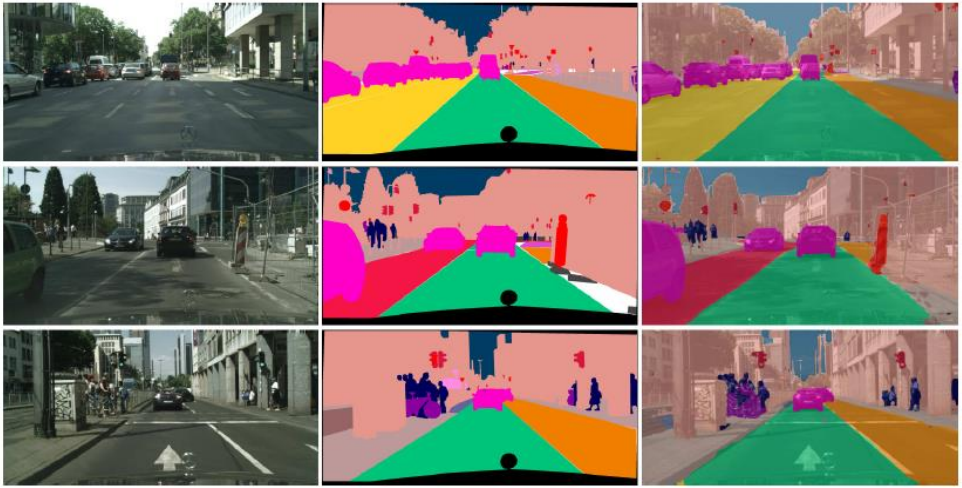
- Estimate the corridor ahead within which the vehicle is supposed to drive
- May or may not consider obstacles (green) within the driving corridor
- Multiple driving corridors, the relevant one depends on the high-level plan

Road and Lane Detection: Representations

Driving Corridor Prediction:

- Estimate the corridor ahead within which the vehicle is supposed to drive
- May or may not consider obstacles (green) within the driving corridor
- Multiple driving corridors, the relevant one depends on the high-level plan

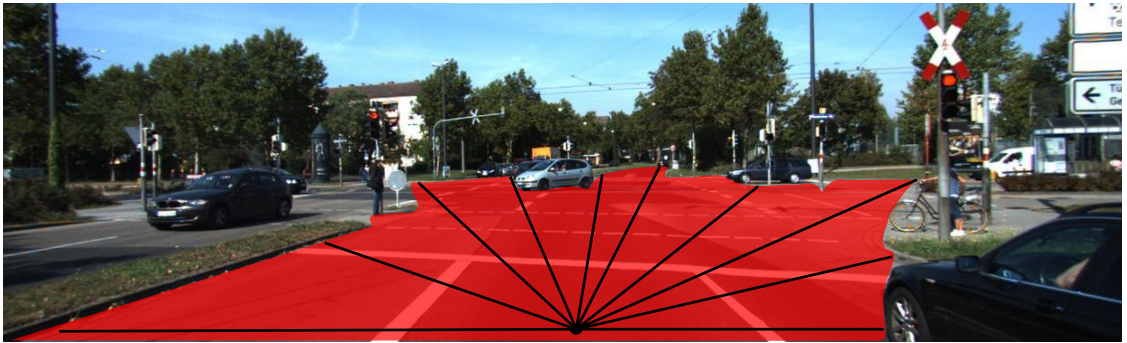
Driving Corridor Prediction



- Fine-grained semantic segmentation (ego/parallel/opposite lanes)

Meyer, Salscheider, Orzechowski and Christoph Stiller: Deep Semantic Lane Segmentation. IROS, 2018.

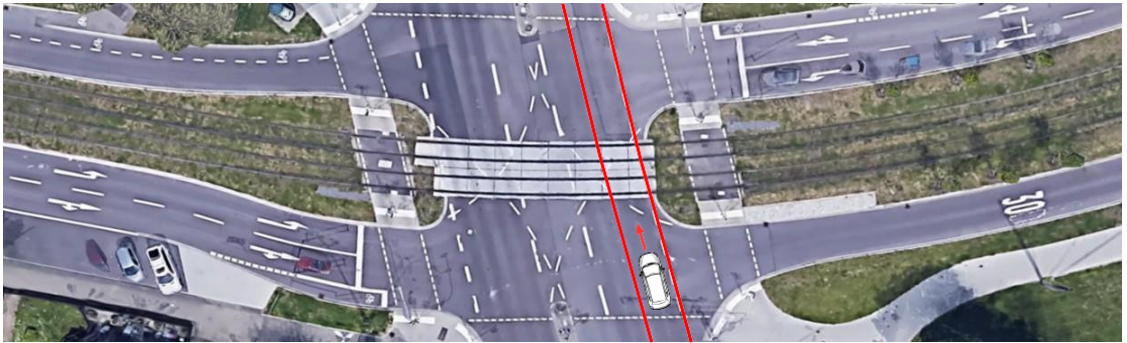
Road and Lane Detection: Representations



Freespace Estimation:

- Estimate places that can be reached without collision
- Purely geometric cue (“which places can be directly reached?”)
- Doesn’t require semantic information or information about lanes ($\Rightarrow$ robust)

Road and Lane Detection: Representations



2D vs. 3D Representation:

- Estimating quantities in the 2D image domain is not directly useful
- Needs to be mapped into 3D (or bird's eye view) where vehicle is controlled
- Given an estimate of the road surface (e.g., 3D plane), this is possible

Road and Lane Detection: Ambiguities

Observations can be ambiguous:

- This is an easier case (similar to highway driving)
- Road segmentation, lane boundaries and driving corridor clearly defined
- Freespace Estimation: is the grass field on the right traversable or not?

Road and Lane Detection: Ambiguities

Observations can be ambiguous:

- Temporary lane markings during road construction, often old ones not erased
- Which lane marking counts? Which ones are relevant (here: bicycle lane?)
- Lane markings often hard to detect (visibility), demaged or missing

Road and Lane Detection: Ambiguities

Observations can be ambiguous:

- Road Segmentation: which parts are considered road? (vs. sidewalk etc.)
- Freespace Estimation: is the sidewalk drivable? (height threshold)
- Lane Boundary Detection: neither lane markings nor curbs in this image

Road and Lane Detection: Ambiguities

Observations can be ambiguous:

- Navigation in parking lots not possible based on lane markings alone
- Accessibility of entryway (left) depends on height of curb (subtle cue)
- Curbs can be of arbitrary height, how to detect them robustly?

Road and Lane Detection: Ambiguities

Observations can be ambiguous:

- Rural streets often unmarked, lane detection very difficult
- Detection of road signs relevant to determine allowed driving directions
- Sometimes difficult to detect if car is parked or not (affecting driving corridor)

Road and Lane Detection: Ambiguities

Observations can be ambiguous:

- Often very unstructured environments which are hard to capture parametrically
- Scene elements sometimes resemble lane markings, but are not
- Some areas not designated for driving, yet not clearly marked (here: poles)

Road and Lane Detection: Ambiguities

Observations can be ambiguous:

- Lane markers often missing at intersections (otherwise too confusing)
- Tram railroad tracks can be confused with lane markings
- Structured but at the same time very unstructured, holistic and reasoning required

Road and Lane Detection: Ambiguities

Observations can be ambiguous:

- Often navigation based solely on lane markings not a good idea
- In particular, construction sites are often badly structured
- Construction signage changes street layout, lanes partially obstructed

Road and Lane Detection: Ambiguities

Observations can be ambiguous:

- Here no lane markings but implicitly two lanes
- However, both lanes narrow down into one based on very subtle cues
- Requires combination of structured planning and reactive control

Road and Lane Detection: Ambiguities

Observations can be ambiguous:

- Sometimes lane markings can have very different semantics
- Allowed driving directions not always easy to infer
- Heavy occlusions might obstruct the sensor field of view

Road and Lane Detection: Summary

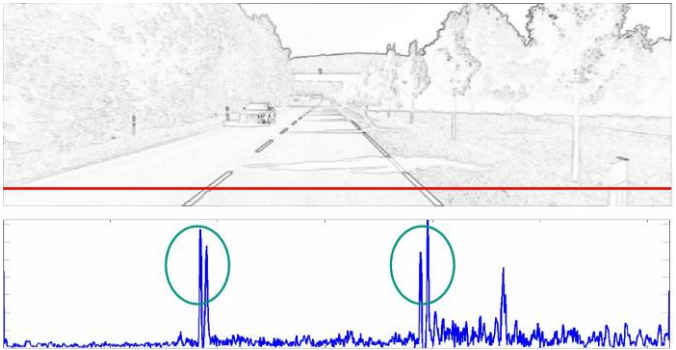
	Pros	**Cons**
Lane Marking	• Relatively easy inference task • Compact parametric models	• Markings missing/deteriorated • Only in structured env. (highway)
Lane	• Directly yields path / centerline • Compact parametric models	• Ambiguous/difficult to estimate • Only in structured env. (highway)
Road Segment.	• Works in unstructred env. • Doesn't rely on accurate geometry	• Semantic ambiguities • Doesn't output path information
Driving Corridor	• Directly relevant to control • Also comprises obstacle inform.	• Depends on control signal • Ambiguous/difficult to estimate
Freespace Est.	• Works in unstructred env. • Doesn't require semantics	• Relies on accurate geometry • Doesn't output path information

Lane Marking Detection

Detection of Lane Markings

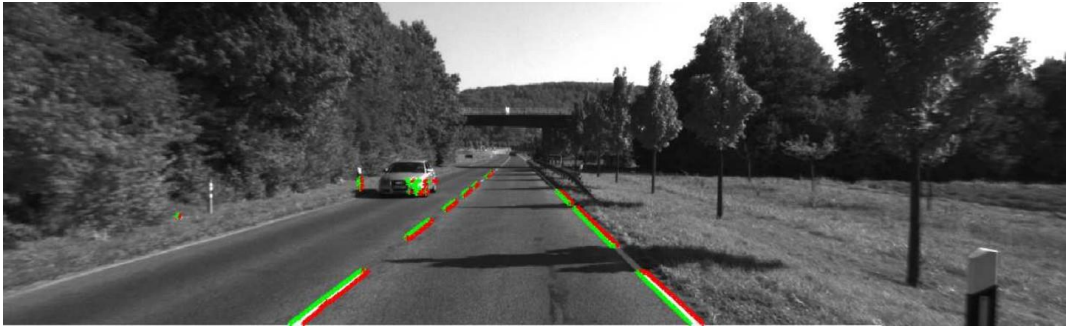
- Gradient Image (convolution with horizontal kernel $k = [-1\ 0\ +1]$)
- Alternatives: steerable filters, templates, learned features

Detection of Lane Markings



- Search for large gradients along each image row
- If depth available: filter points not on ground plane (e.g., eg by plane fitting)
- Remove isolated points or points for which no opposite gradient exists in vicinity

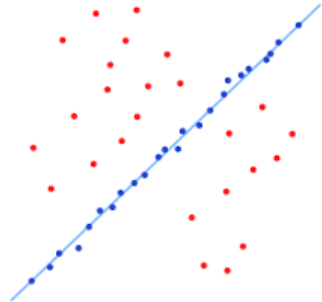
Detection of Lane Markings



- Green: detected left boundary of lane marking
- Red: detected right boundary of lane marking
- False positives can be rejected using heuristics or robust fitting (e.g., random sample consensus - RANSAC)
- Note: not all lane markings detected!

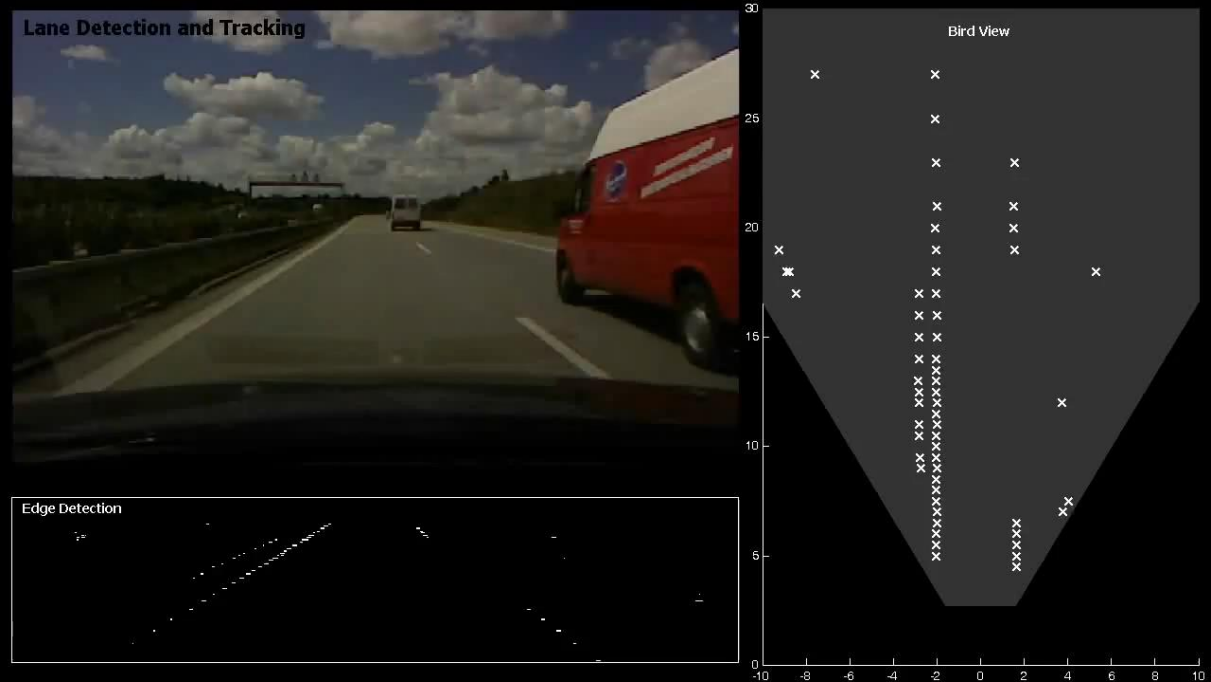
Algorithm 1 RANSAC

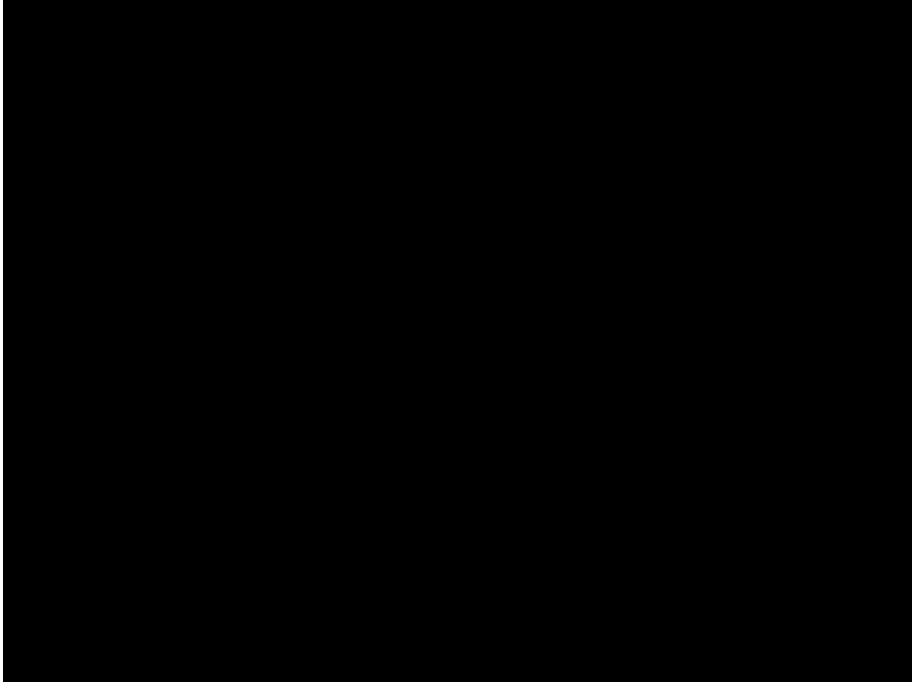
1: Select randomly the minimum number of points required to determine the model parameters.
2: Solve for the parameters of the model.
3: Determine how many points from the set of all points fit with a predefined tolerance ϵ.
4: If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold τ, re-estimate the model parameters using all the identified inliers and terminate.
5: Otherwise, repeat steps 1 through 4 (maximum of N times).



Inverse Perspective Mapping (IPM)

Main takeaway: estimate 3D position by assuming world is flat


Lane Detection and Tracking
Edge Detection
Bird View
30
25
20
15
10
5
0
-10
-8
-6
-4
-2
0
2
4
6
8
10



Panoramic Bird-eye-view/top-down from a monocular camera (numbers are estimated speeds)

Inverse Perspective Mapping: Example

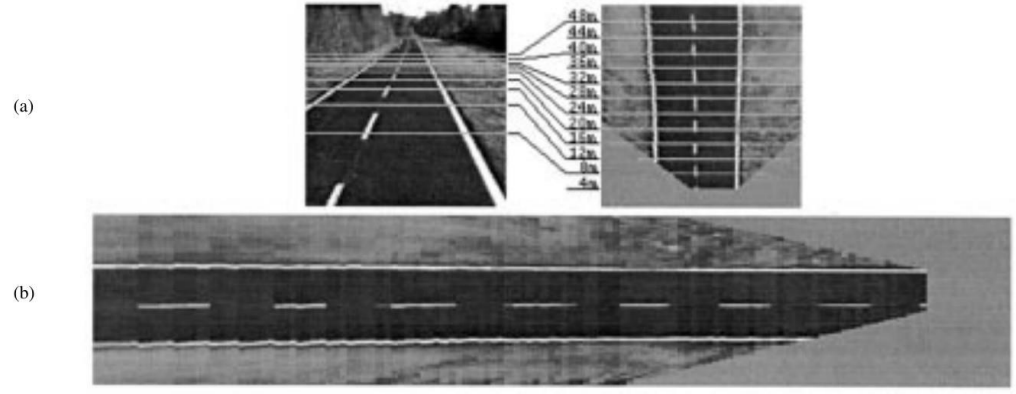
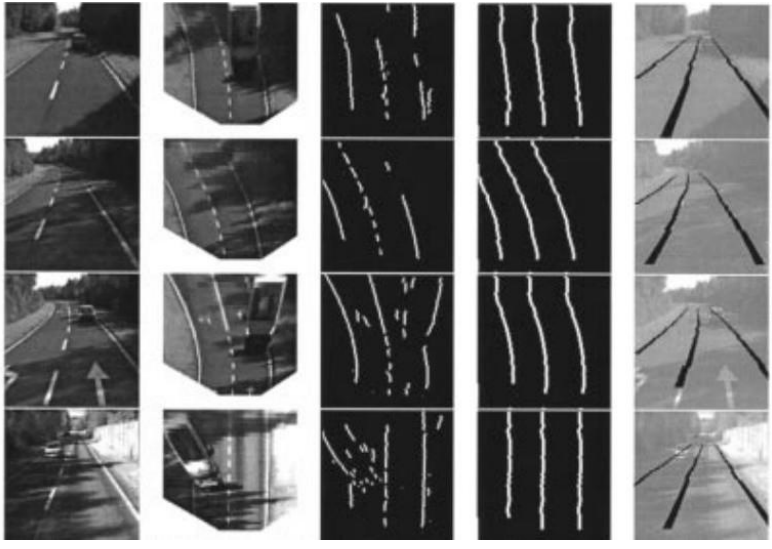


Fig. 8. (a) Horizontal calibration of the MOB-LAB vision system. (b) Rotated version of the remapped image considering an aspect ratio of 1 : 1.

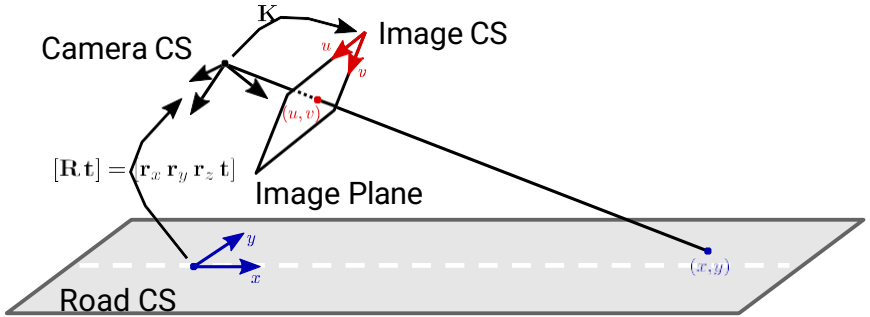
Bertozzi and Broggi: GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection. TIP, 1998.

Inverse Perspective Mapping: Example



Bertozzi and Broggi: GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection. TIP, 1998.

Inverse Perspective Mapping (IPM)

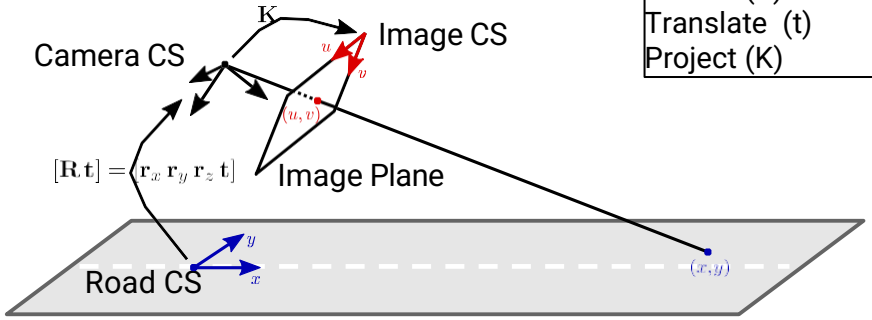


- IPM maps each pixel on the image plane to the respective ground plane point
- IPM is only possible if either per pixel depth or ground model (e.g., plane) known
- How to obtain ground plane (i.e., extrinsic parameters [**R t**])?

Bertozzi and Broggi: GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection. TIP, 1998.

Inverse Perspective Mapping (IPM)

3D Operations:
Rotate (R)
Translate (t)
Project (K)

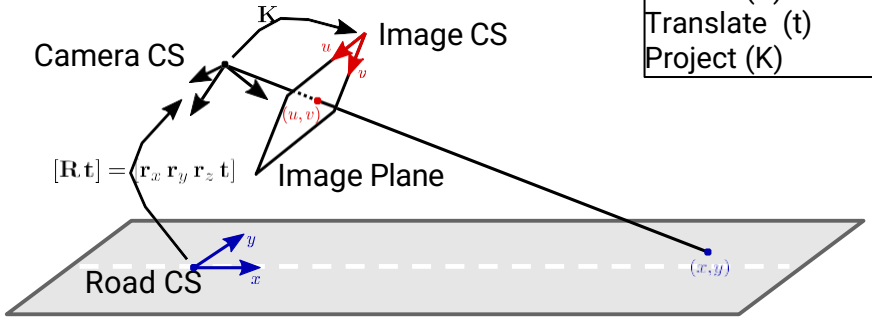


$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \overset{z=0}{\Rightarrow} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \mathbf{K} \begin{bmatrix} \mathbf{r}_x & \mathbf{r}_y & \mathbf{t} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \propto \begin{bmatrix} \mathbf{r}_x & \mathbf{r}_y & \mathbf{t} \end{bmatrix}^{-1} \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Bertozzi and Broggi: GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection. TIP, 1998.

Inverse Perspective Mapping (IPM)

3D Operations:
Rotate (R)
Translate (t)
Project (K)



$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \overset{z=0}{\Rightarrow} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \mathbf{K} \begin{bmatrix} \mathbf{r}_x & \mathbf{r}_y & \mathbf{t} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \propto \begin{bmatrix} \mathbf{r}_x & \mathbf{r}_y & \mathbf{t} \end{bmatrix}^{-1} \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Bertozzi and Broggi: GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection. TIP, 1998.

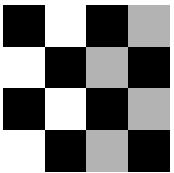
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Homography H (planar projective transformation)

3x3 Homography matrix

Punchline: For planar surfaces, 3D to 2D perspective projection reduces to a 2D to 2D transformation.

Punchline2: This transformation is INVERTIBLE!



Whiteboard example

Parametric Lane Marking Estimation

- Lane marking detection alone is not sufficient (why?)
- Only returns a set of pixels in image or road plane space
- In order to be useful for navigation, this needs to be transformed into a more ***semantically meaningful parametric model***
- Often low-dimensional parametric models (lines, polynomials, bezier curves, splines) are used
  - In the following: splines
- Note 1: This is a multi-model fitting problem!
- Note 2: Outliers must be handled (model must be robust)!

Parametric Lane Marking Estimation

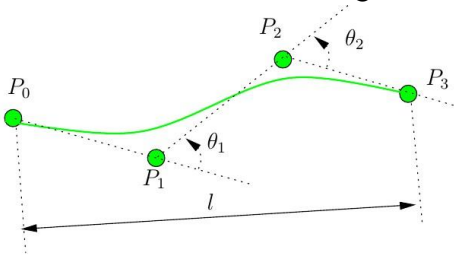


Fig. 7. Spline score computation.

Algorithm 1 RANSAC Spline Fitting

```
for i = 1 to numIterations do
    points=getRandomSample()
    spline=fitSpline(points)
    score=computeSplineScore(spline)
    if score > bestScore then
        bestSpline = spline
    end if
end for
```

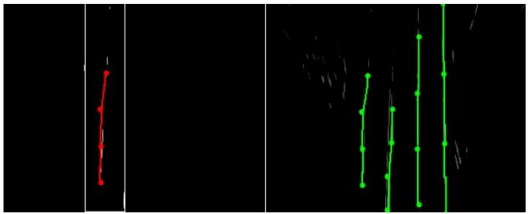


Fig. 8. RANSAC Spline fitting. Left: one of four windows of interest (white) obtained from previous step with detected spline (red). Right: the resulting splines (green) from this step

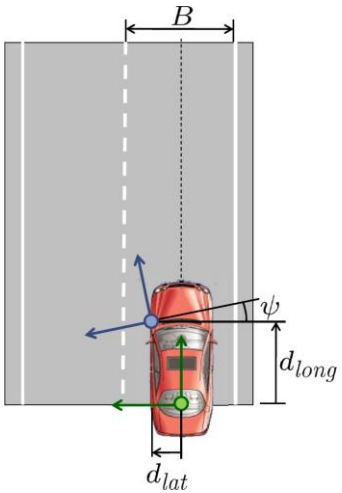
Fig. 10. Post-processing splines. Left: splines before post-processing in blue. Right: splines after post-processing in green. They appear longer and localized on the lanes.

Parametric Lane Marking Estimation

Aly: Real time detection of lane markers in urban streets. IV, 2008.

Task:
Given model for the lanes,
Fit it to observed data points

Model for Straight Lanes



Geometric model: (straight=highway scenario)

- Can we fit a single model for the entire lane?
- Lane width $B \in \mathbb{R}^+$
- Lateral vehicle offset wrt. centerline $d_{lat} \in \mathbb{R}$
- Longitudinal position $d_{long} \in \mathbb{R}$
- Yaw angle $\psi \in [-\pi, \pi]$

State:

$$\mathbf{s} = \begin{bmatrix} B \\ d_{long} \\ d_{lat} \\ \psi \end{bmatrix}$$

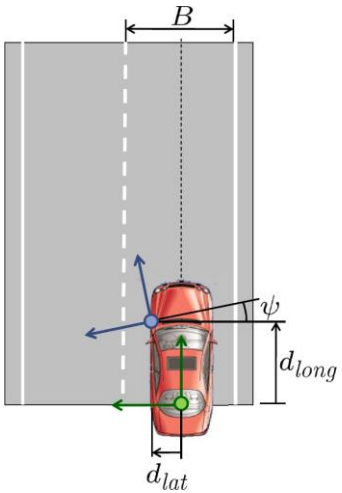
Warning Slide: Math Ahead

Summary:

- ▶ Detect lane markings in camera image
- ▶ Transform position of markings into vehicle coordinates
- ▶ Estimate pose parameters and lane width

Key Takeaway: Modeling straight lanes can be done with a ***closed form regression solution***

Model for Straight Lanes



Geometric model: (straight=highway scenario)

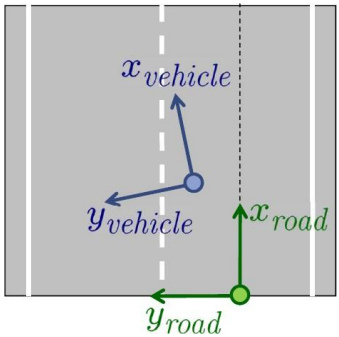
- Can we fit a single model for the entire lane?
- Lane width $B \in \mathbb{R}^+$
- Lateral vehicle offset wrt. centerline $d_{lat} \in \mathbb{R}$
- Longitudinal position $d_{long} \in \mathbb{R}$
- Yaw angle $\psi \in [-\pi, \pi]$

State:

$$\mathbf{s} = \begin{bmatrix} B \\ d_{long} \\ d_{lat} \\ \psi \end{bmatrix}$$

Model for Straight Lanes

Think rotation and translation



Transformation vehicle → road coordinates:

$$\begin{bmatrix} x_{road} \\ y_{road} \end{bmatrix} = \begin{bmatrix} d_{long} \\ d_{lat} \end{bmatrix} + \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} x_{vehicle} \\ y_{vehicle} \end{bmatrix}$$

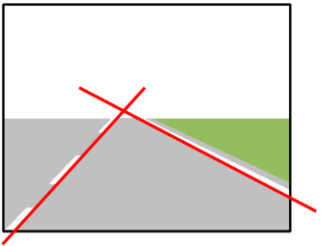
Transformation road → vehicle coordinates:

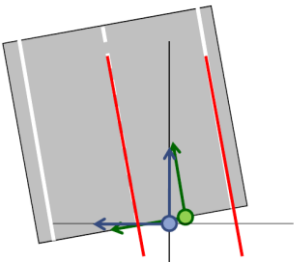
$$\begin{bmatrix} x_{vehicle} \\ y_{vehicle} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \left(\begin{bmatrix} x_{road} \\ y_{road} \end{bmatrix} - \begin{bmatrix} d_{long} \\ d_{lat} \end{bmatrix} \right)$$

Model for Straight Lanes

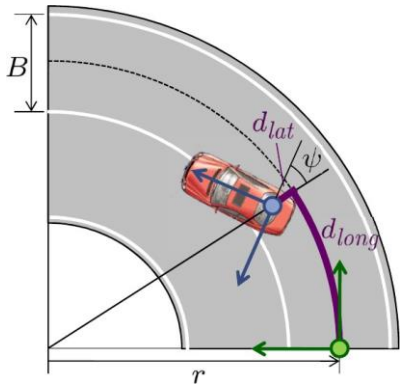
Summary:

- Detect lane markings in camera image
- Transform position of markings into vehicle coordinates
- Estimate pose parameters and lane width





Model for Circular Lanes



Geometric model:

- Lane width $B \in \mathbb{R}^+$
- Signed radius $r \in \mathbb{R}$ or curvature $\kappa = \frac{1}{r}$
- Lateral vehicle offset $d_{lat} \in \mathbb{R}$
- Longitudinal position $d_{long} \in \mathbb{R}$
- Yaw angle $\psi \in [-\pi, \pi]$

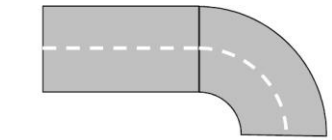
State:

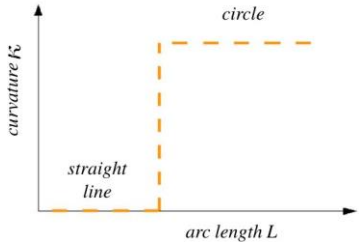
$$\mathbf{s} = \begin{bmatrix} B & r & d_{long} & d_{lat} & \psi \end{bmatrix}^T$$

Circular trajectories are not enough

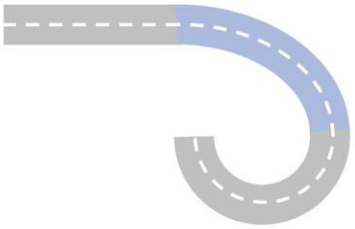
$$y(\ell) \approx \frac{1}{2}\kappa_0\ell^2 + \frac{1}{6}\kappa_1\ell^3$$

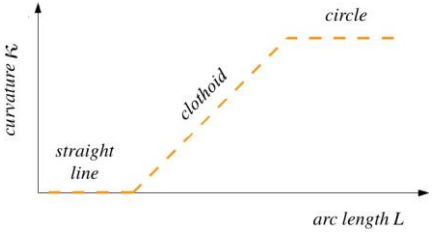
Circle-Straight Combination





Clothoid





Deep Convolutional Image Segmentation

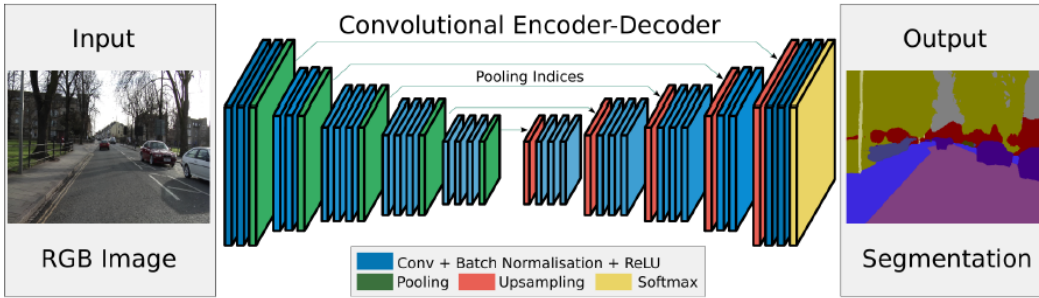
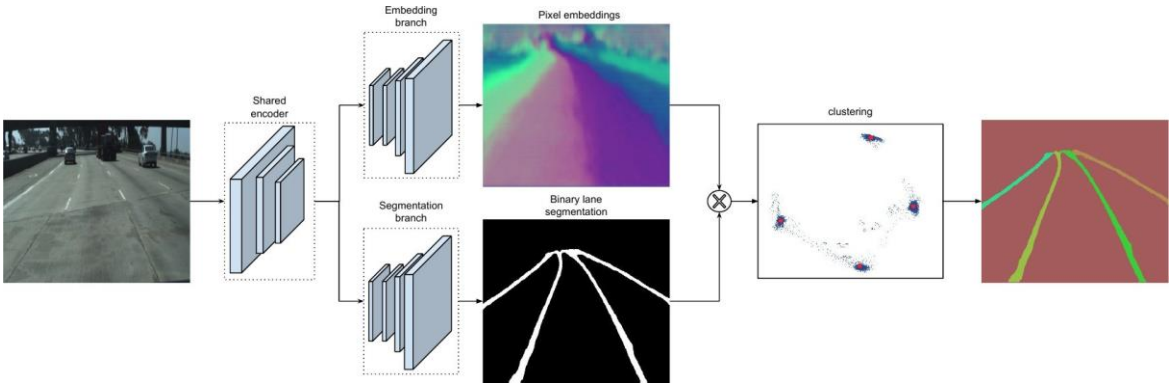


Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

Badrinarayanan, Kendall and Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. PAMI, 2017.

End-to-End Lane Detection



Neven et al.: Towards End-to-End Lane Detection: an Instance Segmentation Approach. IV, 2018.

End-to-End Lane Detection

EmbeddingLoss Function:

$$L_{var} = \frac{1}{C} \sum_{c=1}^{C} \frac{1}{N_c} \sum_{i=1}^{N_c} \left[\|\mu_c - x_i\| - \delta_{\mathrm{v}} \right]_+^2$$

$$L_{dist} = \frac{1}{C(C-1)} \sum_{c_A=1}^{C} \sum_{c_B=1, c_A \neq c_B}^{C} \left[\delta_{\mathrm{d}} - \|\mu_{c_A} - \mu_{c_B}\| \right]_+^2$$

- Clustering loss, μ is cluster mean
- Learns an **embedding** where pixels
  - belong to the same lane are pulled together
  - belong to different lanes are pushed apart

Neven et al.: Towards End-to-End Lane Detection: an Instance Segmentation Approach. IV, 2018.

End-to-End Lane Detection: Results

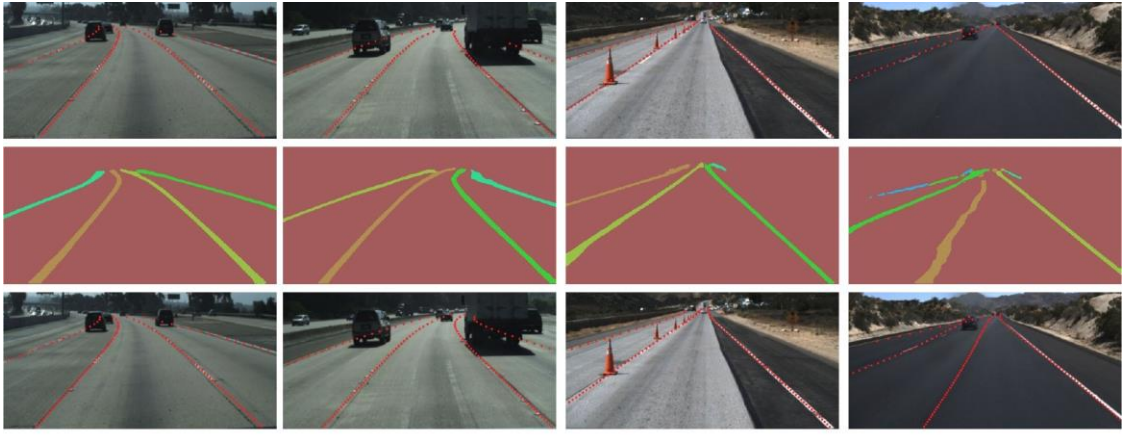


Fig. 5. Visual results. *Top row*: ground-truth lane points. *Middle row*: LaneNet output. *Bottom row*: final lane predicts after lane fitting.

Neven et al.: Towards End-to-End Lane Detection: an Instance Segmentation Approach. IV, 2018.

Further Readings

I Aly: Real time detection of lane markers in urban streets. IV, 2008.

I Badrinarayanan, Kendall and Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. PAMI, 2017.

I Badino, Franke and Mester: Free Space Computation Using Stochastic Occupancy Grids and Dynamic Programming. ICCV Workshops, 2007.