



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

IROS19

11th International workshop on

Planning, Perception and Navigation for Intelligent Vehicles

Full Day Workshop

November 4th, 2019, Macau, China

<https://project.inria.fr/ppniv19/>

Organizers

Pr Marcelo Ang (NUS, Singapore)
Pr Christian Laugier (INRIA, France),
Pr Philippe Martinet (INRIA, France),
Pr M.A. Sotelo (University of Alcalà, Spain)
Pr Christoph Stiller (KIT, Germany)

Contact

Director of Research Philippe Martinet
Inria - CHORALE team
2004 route des Lucioles, 06902 Sophia-Antipolis, FRANCE
Phone: +33 240 376 975, Sec : +33 240 376 934, Fax : +33 240 376 6930
Email: Philippe.Martinet@inria.fr
Home page: <http://www-sop.inria.fr/members/Philippe.Martinet/>



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Foreword

The purpose of this workshop is to discuss topics related to the challenging problems of autonomous navigation and of driving assistance in open and dynamic environments. Technologies related to application fields such as unmanned outdoor vehicles or intelligent road vehicles will be considered from both the theoretical and technological point of views. Several research questions located on the cutting edge of the state of the art will be addressed. Among the many application areas that robotics is addressing, transportation of people and goods seem to be a domain that will dramatically benefit from intelligent automation. Fully automatic driving is emerging as the approach to dramatically improve efficiency while at the same time leading to the goal of zero fatalities. This workshop will address robotics technologies, which are at the very core of this major shift in the automobile paradigm. Technologies related to this area, such as autonomous outdoor vehicles, achievements, challenges and open questions would be presented. Main topics include: Road scene understanding, Lane detection and lane keeping, Pedestrian and vehicle detection, Detection, tracking and classification, Feature extraction and feature selection, Cooperative techniques, Collision prediction and avoidance, Advanced driver assistance systems, Environment perception, vehicle localization and autonomous navigation, Real-time perception and sensor fusion, SLAM in dynamic environments, Mapping and maps for navigation, Real-time motion planning in dynamic environments, Human-Robot Interaction, Behavior modeling and learning, Robust sensor-based 3D reconstruction, Modeling and Control of mobile robot, Deep learning applied in autonomous driving, Deep reinforcement learning applied in intelligent vehicles.

Previously, several workshops were organized in the near same field. The 1st edition [PPNIV'07](#) of this workshop was held in Roma during ICRA'07 (around 60 attendees), the second [PPNIV'08](#) was in Nice during IROS'08 (more than 90 registered people), the third [PPNIV'09](#) was in Saint-Louis (around 70 attendees) during IROS'09, the fourth edition [PPNIV'12](#) was in Vilamoura (over 95 attendees) during IROS'12, the fifth edition [PPNIV'13](#) was in Vilamoura (over 135 attendees) during IROS'13, the sixth edition [PPNIV'14](#) was in Chicago (over 100 attendees) during IROS14, the seventh edition [PPNIV'15](#) was in Hamburg (over 150 attendees) during IROS15, the eighth edition [PPNIV'16](#) was in Rio de Janeiro (over 100 attendees) during ITSC16, the ninth edition [PPNIV'17](#) was in Vancouver during IROS17 (over 170 attendees), the 10th edition [PPNIV'18](#) was in Madrid during IROS18 (over 350 attendees), and this 11th edition [PPNIV'19](#) has gathered over 300 attendees.

In parallel, we have also organized [SNODE'07](#) in San Diego during IROS'07 (around 80 attendees), MEPPC08 in Nice during IROS'08 (more than 60 registered people), [SNODE'09](#) in Kobe during ICRA'09 (around 70 attendees), [RITS'10](#) in Anchorage during ICRA'10 (around 35 attendees), [PNAVHE11](#) in San Francisco during the last IROS11 (around 50 attendees), and the last one [WMEPC14](#) in Hong Kong during the last ICRA14 (around 65 attendees),

This workshop is composed with 4 invited talks and 6 selected papers. One round table has gathered specialist in ***Human vehicle interaction***. Four sessions have been organized:

Session I: Machine & Deep Learning

Session II: Perception & Situation awareness

Session III: Planning & Navigation

Session IV: Human vehicle interaction



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Intended Audience concerns researchers and PhD students interested in mobile robotics, motion and action planning, robust perception, sensor fusion, SLAM, autonomous vehicles, human-robot interaction, and intelligent transportation systems. Some peoples from the mobile robot industry and car industry are also welcome.

This workshop is made in relation with IEEE RAS: RAS Technical Committee on “Autonomous Ground Vehicles and Intelligent Transportation Systems” (<http://tab.ieee-ras.org/>).

Christian Laugier, Philippe Martinet, Marcelo Ang, Miguel Angel Sotelo and Christoph Stiller



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session I

Machine & Deep Learning

- **Keynote speaker: Roland Meertens (AID, Munich, Germany)**
Title: The road towards perception for autonomous driving: methods, challenges, and the data required
- **Title: Transformation-adversarial network for road detection in LIDAR rings, and model-free evidential road grid mapping**
Authors: E. Cappelier, F. Davoine, V. Cherfaoui, Y. Li
- **Title: End-to-End Deep Neural Network Design for Short-term Path Planning**
Authors: M.Q. Dao, D. Lanza, V. Frémont



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session I

Keynote speaker: **Roland Meertens**
(AID, Munich, Germany)

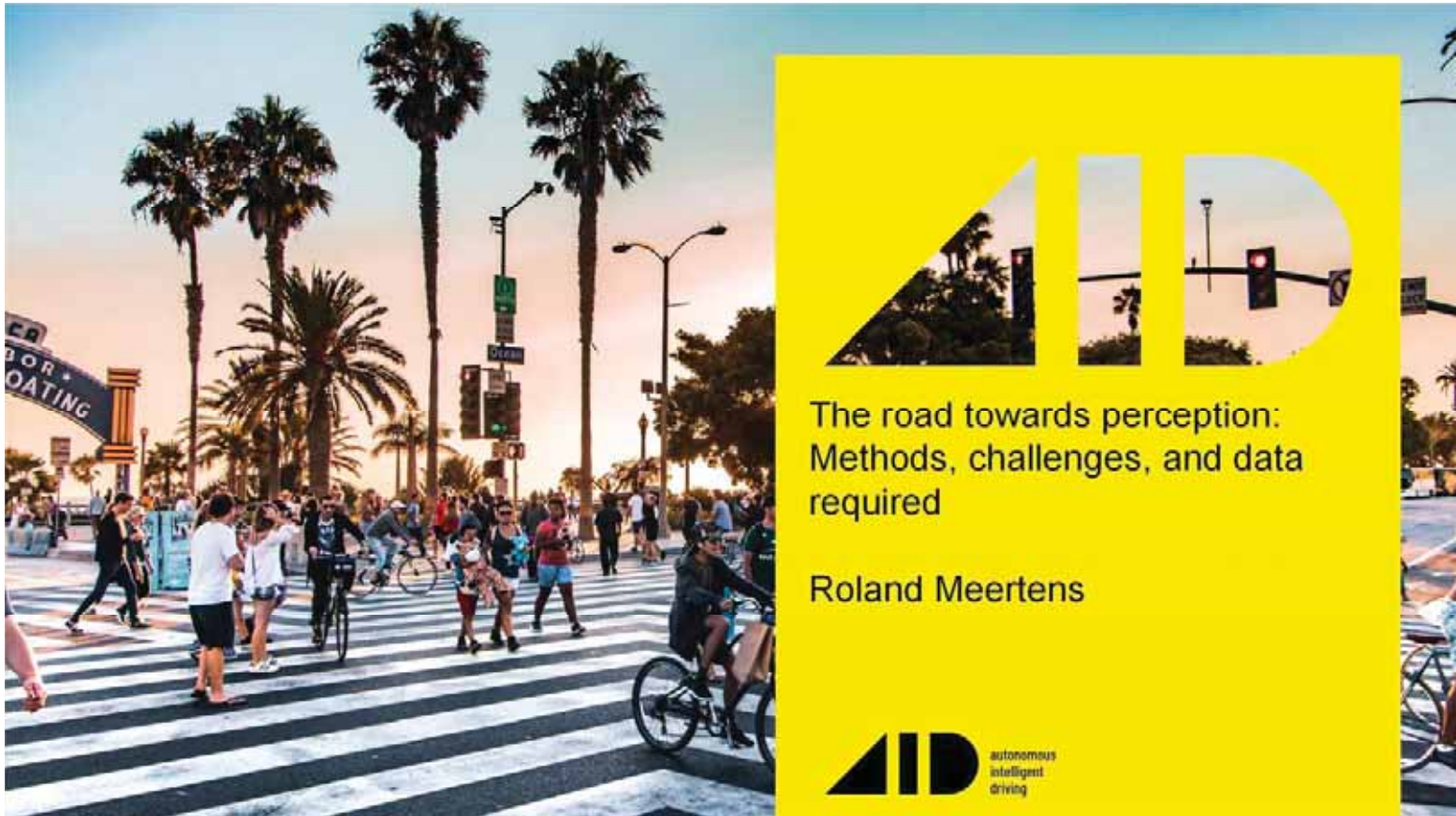
The road towards perception for autonomous driving: methods, challenges, and the data required

Abstract: Self-driving cars are expected to make a big impact on our daily lives within a couple of years. However, first we should solve the most interesting Artificial Intelligence (AI) problem of this century: perception. We will look at the problem of perception for autonomous vehicles, the sensors which are used to solve this problem, and the methods which are currently state of the art. We will also take a look at the available data: a crucial thing we need to teach machines about the world.

Biography: Roland is developing machine learning solutions for the perception problem of autonomous vehicles at AID- Autonomous Intelligent Driving in Munich. He works with lidar and camera data to solve challenges such as 3D bounding box detection, semantic segmentation, and localisation. Previously he worked on deep learning approaches for natural language processing (NLP) problems at Infor, computer vision for drones as a researcher at the TU Delft, and social robotics at SpirOps.



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems



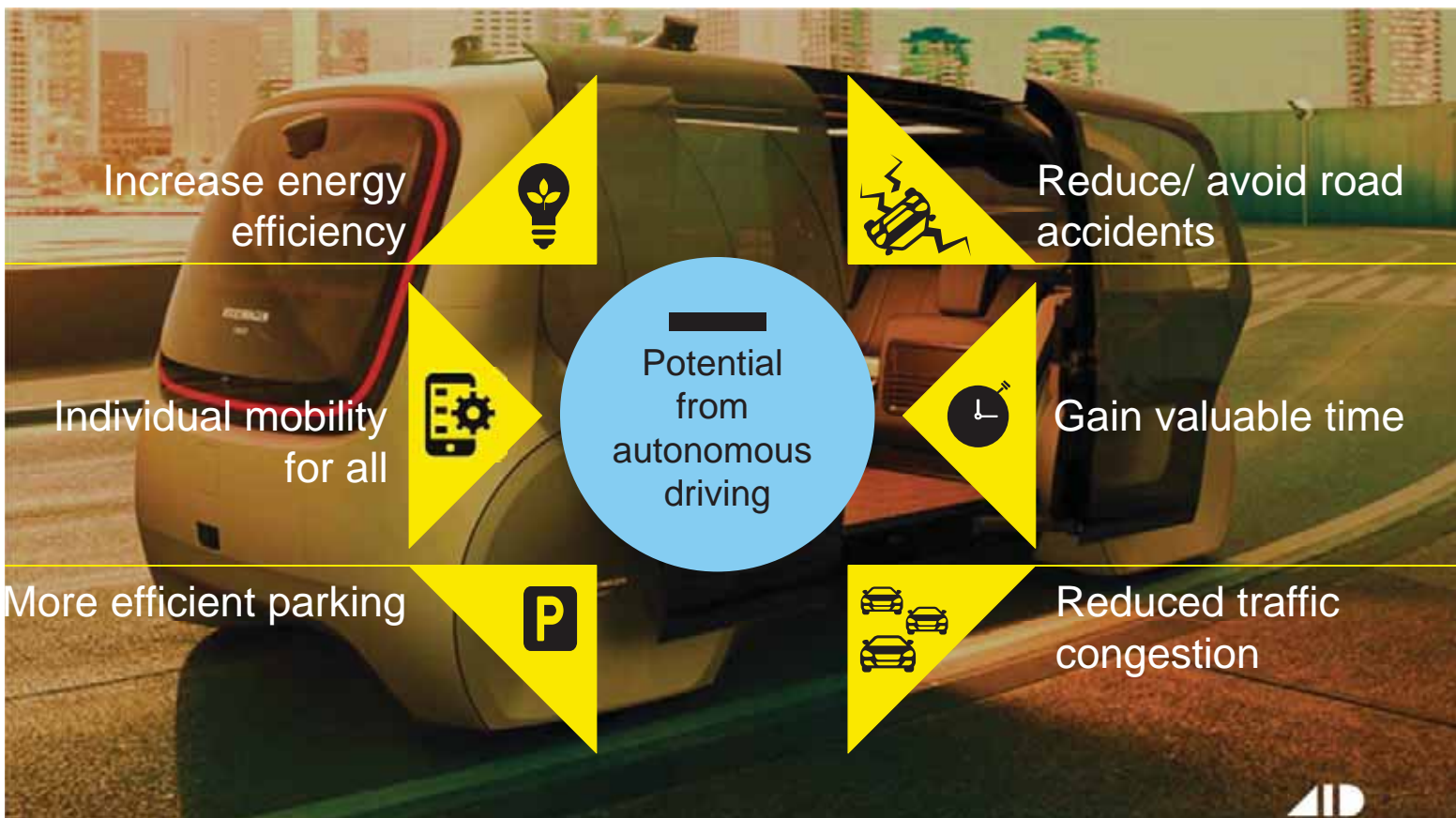
Topics of today

- ▲ Why do self-driving cars matter to humans and industries? What is AID doing with self-driving cars?
- ▲ Generic introduction into sensors and methods
- ▲ 3D object detection – progress, methods, challenges I think are interesting
- ▲ The importance of having the right data for autonomous vehicles – neural networks are data-hungry!

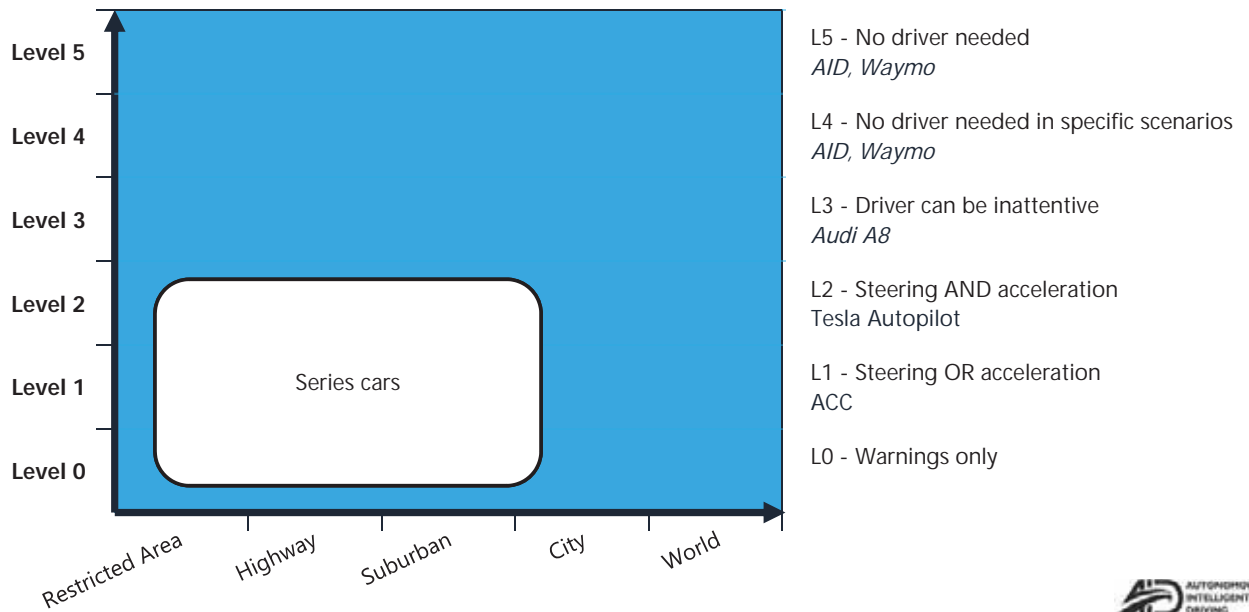
Who is presenting?

Roland Meertens

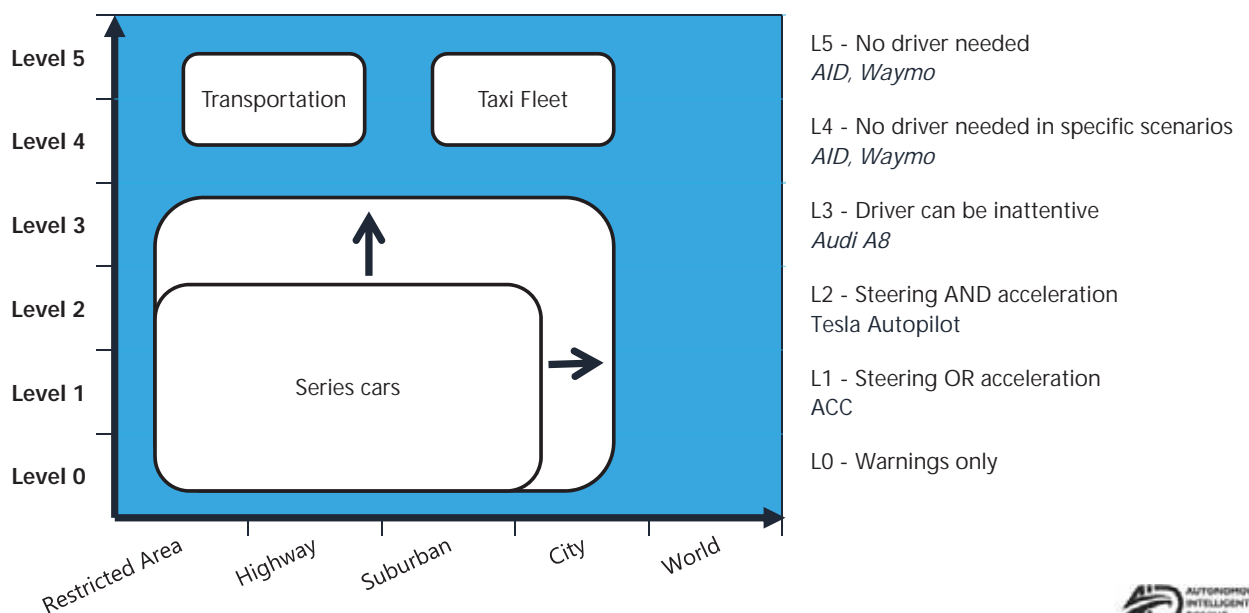
- ▲ Studied artificial intelligence in the Netherlands
- ▲ Worked on autonomous drones at the TU Delft
- ▲ Working in the perception area on ML algorithms
- ▲



ROADMAP FOR AUTONOMOUS DRIVING



ROADMAP FOR AUTONOMOUS DRIVING



Different approaches towards autonomy

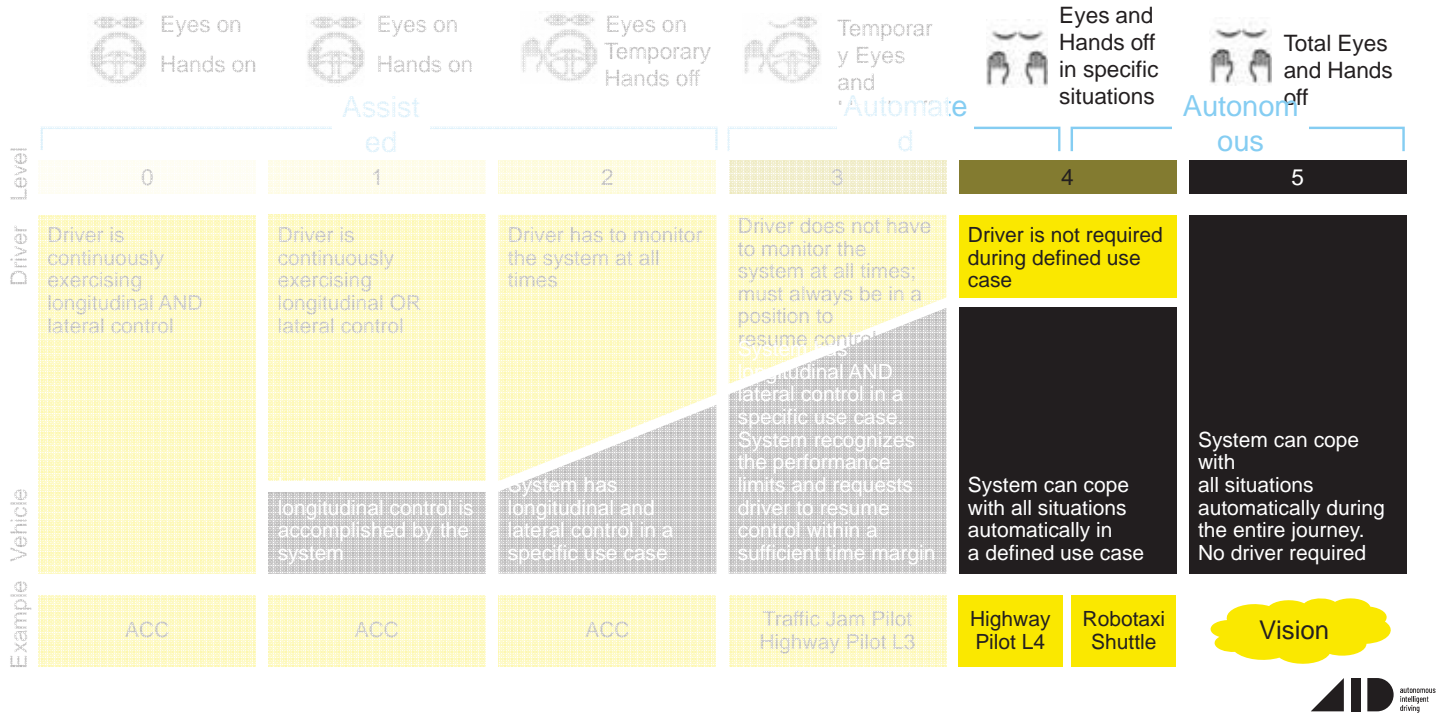


5 levels of autonomy for self-driving cars

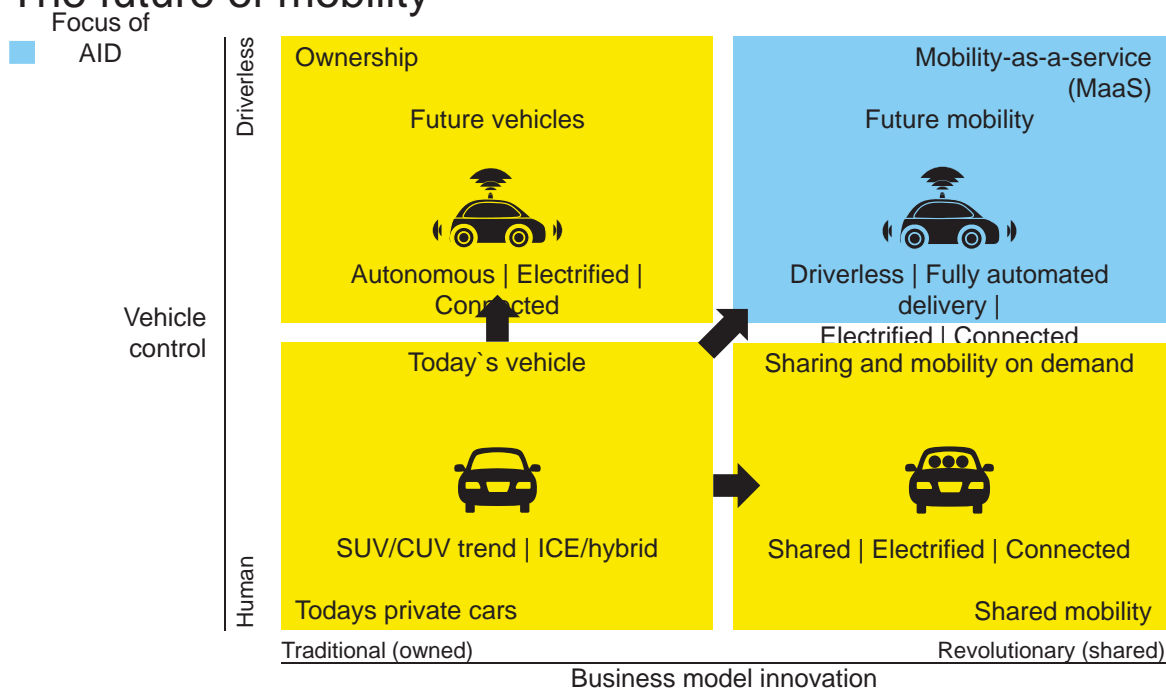
	Assisted		Automated		Autonomous	
	0	1	2	3	4	5
Driver	Eyes on Hands on	Eyes on Hands on	Eyes on Temporary Hands off	Temporary Eyes and Hands off	Eyes and Hands off in specific situations	Total Eyes and Hands off
Vehicle	Driver is continuously exercising longitudinal AND lateral control	Driver is continuously exercising longitudinal OR lateral control	Driver has to monitor the system at all times	Driver does not have to monitor the system at all times; must always be in a position to resume control	Driver is not required during defined use case	System can cope with all situations automatically during the entire journey. No driver required
Example	ACC	ACC	ACC	Traffic Jam Pilot Highway Pilot L3	Highway Pilot L4 Robotaxi Shuttle	Vision



AID's focus is level 4 & 5 in urban environments only



The future of mobility



Source: together 2025



AID Focuses on fast-tracking the development of autonomous vehicles in urban areas



- ▲ 01 AID was launched in March 2017 as 100% subsidiary of AUDI AG. We are located in the center of Munich.
- ▲ 02 Our mission is to build the universal autonomous driving system that improves the lives of millions of people.
- ▲ 03 Our vision is to create a future where Autonomous Driving is embraced by all and not just the few.



11/6/2019 AID PPT Master 2019

12

AID – a true startup atmosphere in the heart of Munich

- ▲ At AID, we believe that Autonomous Driving needs to work for everyone, and we want to ensure that it benefits all people, not just the few.
- ▲ We are aware of the enormous challenge and responsibility in creating the standard system for Autonomous Driving.
- ▲ That is why we are leading a more human autonomous driving company.

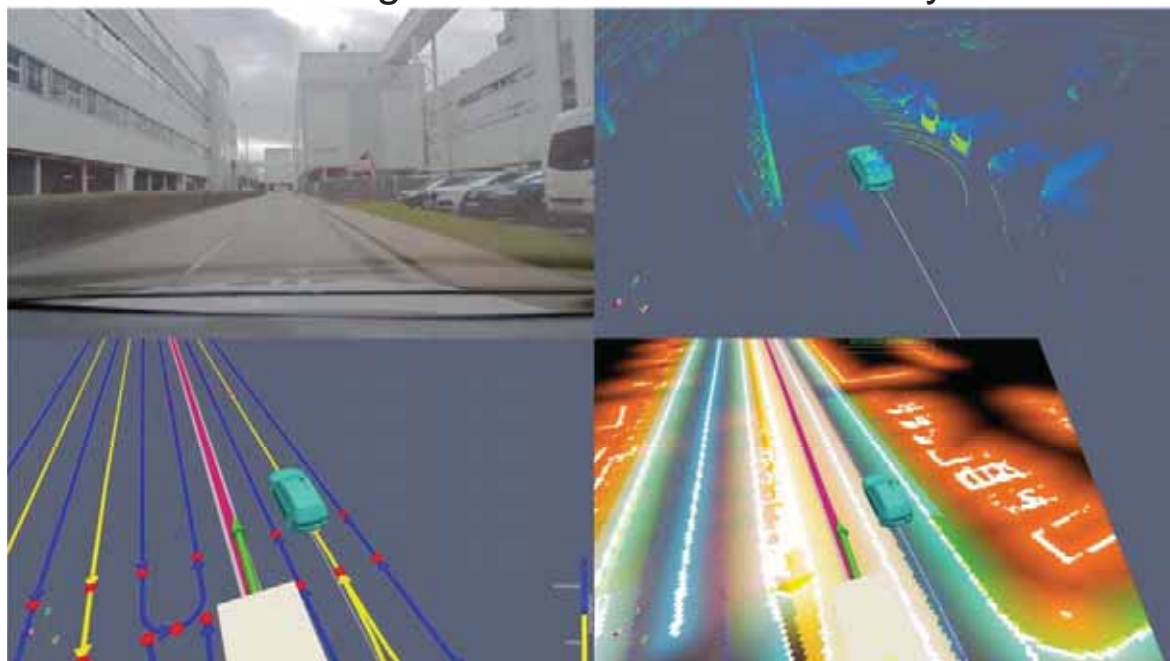
> 220 people > 35 nationalities > 36 years average age
English speaking company

Our car fleet

- ▶ Our fleet of VW eGolf and Audi e-tron drive daily in the city of Munich and in specific areas within one of Audi's plants.
- ▶ We have the unique advantage of belonging to the world's biggest OEM (VW Group), which gives us access to technical knowledge and scalability, but having at the same time the agility of a start-up.



Video of self-driving behavior in an Audi factory



The sensors of autonomous driving



Sensor abilities & failure modes

	Flat texture	Darkness	Low reflectivity	Fog	Dust, Dirt	Velocity	Resolution
 Camera	✓	✗	✓	✗	✗	✗	✓
 Lidar	✗	✓	✗	✗ ¹	✗	✗ ²	✗
 Radar	✗	✓	✓	✓	✓	✓	✗

Sensor redundancy is crucial to overcome individual failure modes. Radar is most robust but application is limited due to low resolution.

Source: 1 - 905nm lidar 2 - non-FMCW lidar

LiDAR



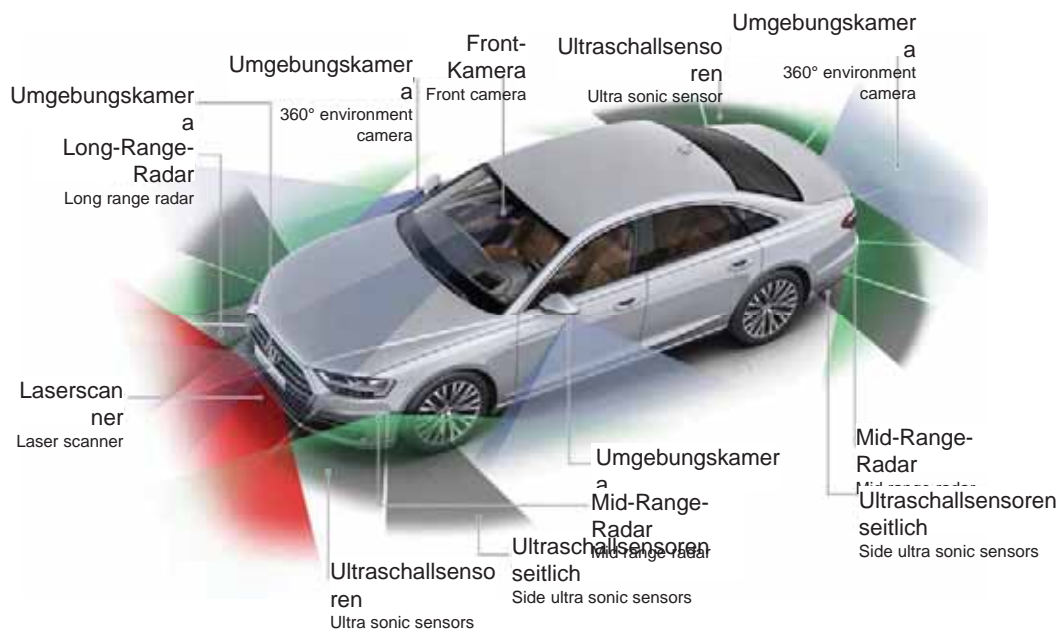
Measures distances with the speed of light

- ▲ Sends out a light pulse and measures the time till it gets back
- ▲ Spins around to measure distances all around the vehicle



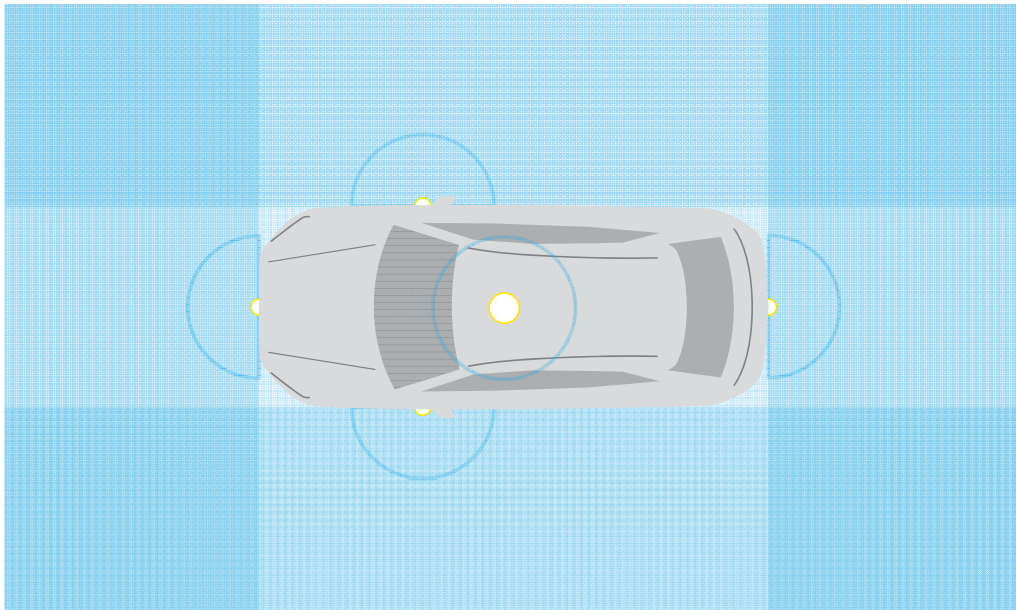
Audi A8: ready for L3 autonomy

Sensor areas for environment observation



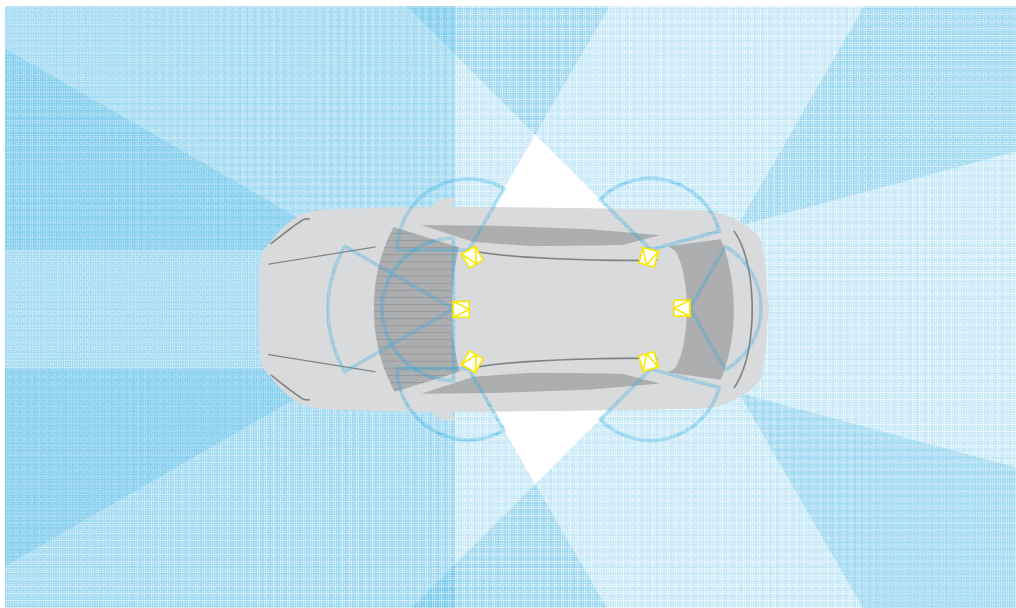
AD Sensor Set: LIDAR

EXEMPLARY, 180° AND 360° FOV



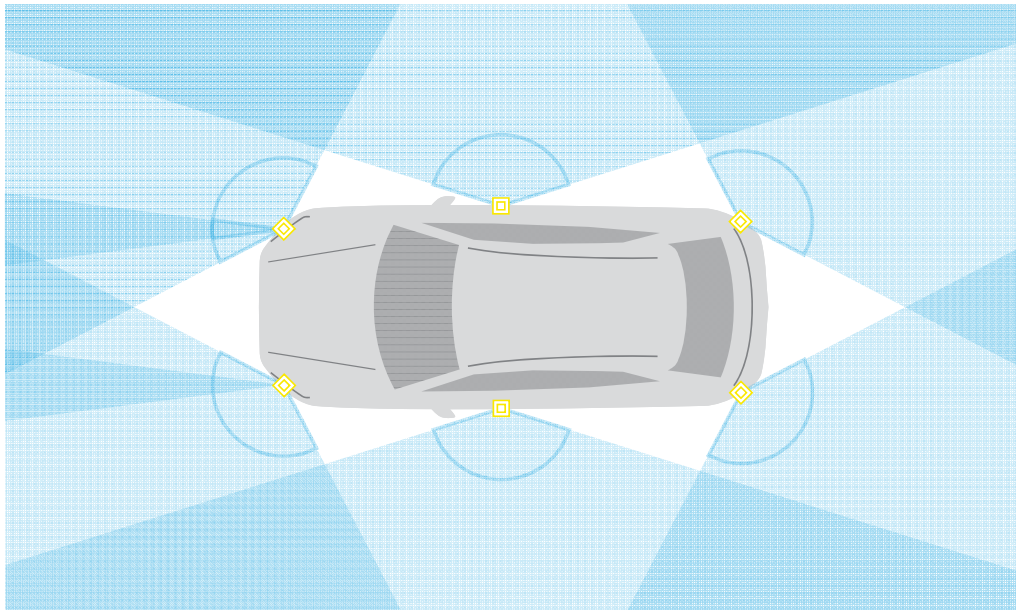
AD Sensor Set: CAMERA

EXEMPLARY 60°, 120° AND 180° FOV



AD Sensor Set: RAdar

EXEMPLARY, 145° AND 15° FOV



Components of autonomous driving

Core Technologies for Self-Driving Vehicles

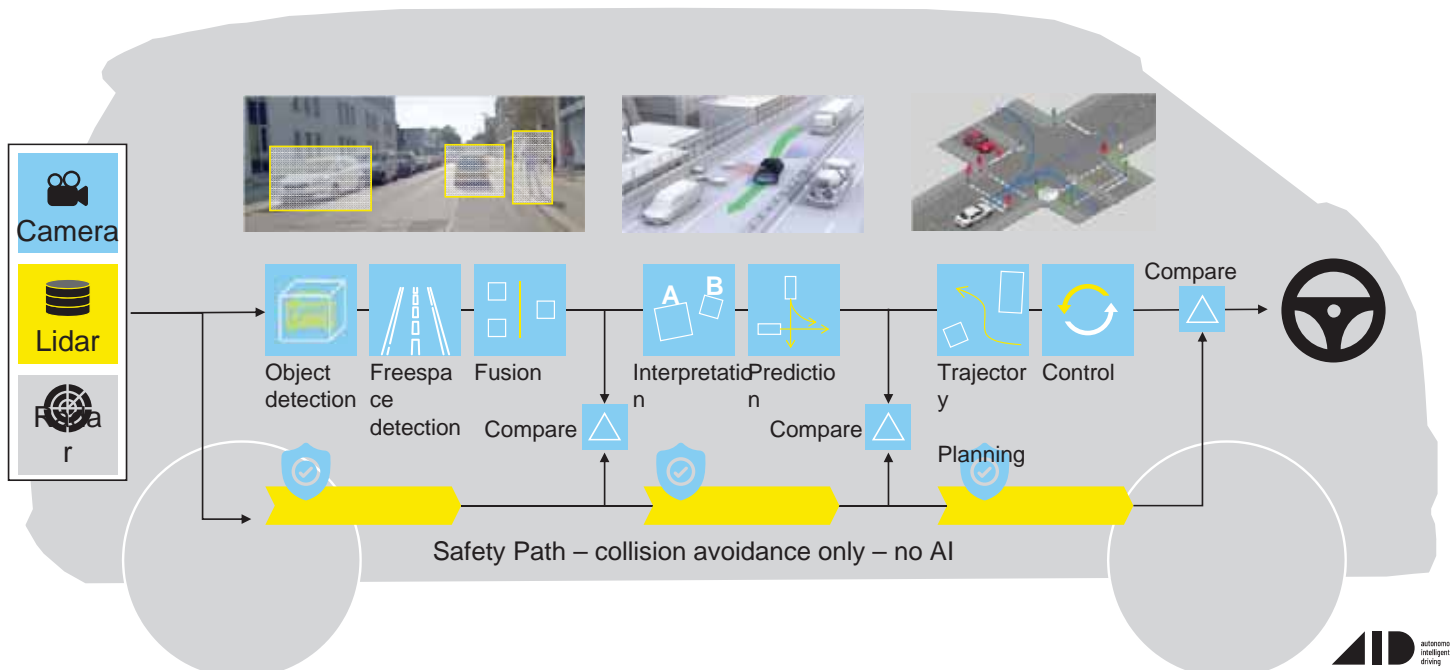


11/6/2019 Testing United 2019, Vienna

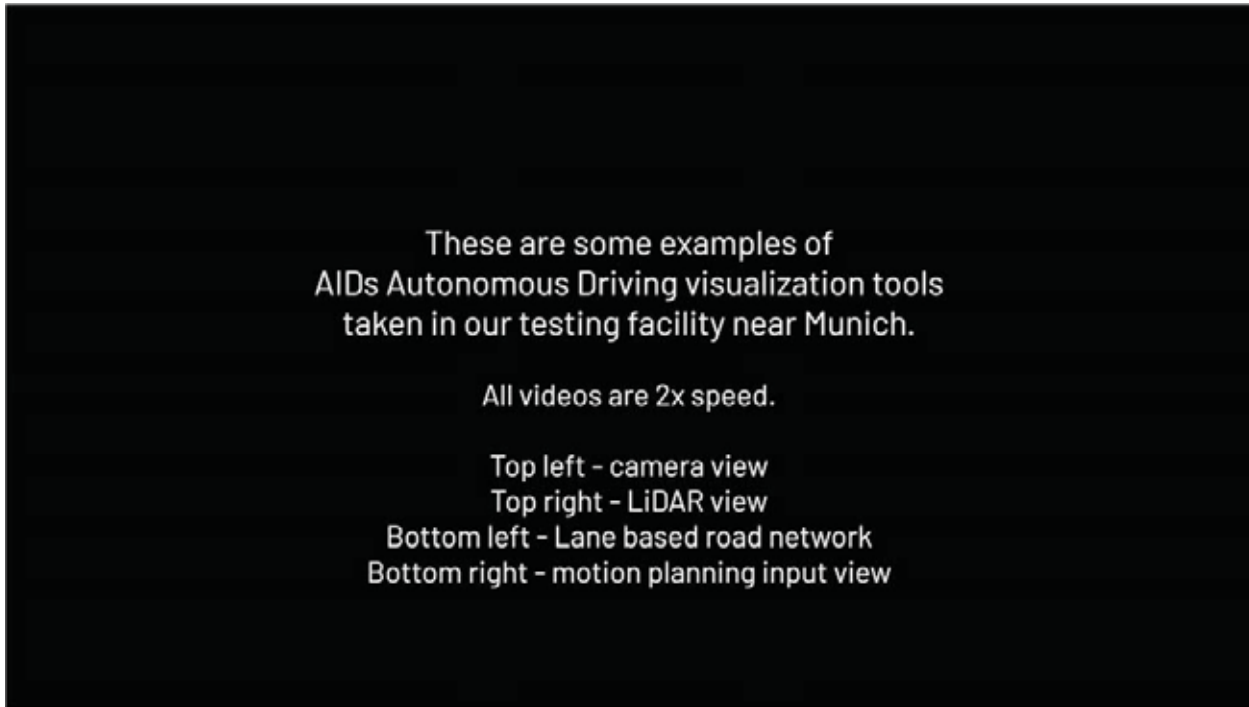
24

In the car: taking a modular approach

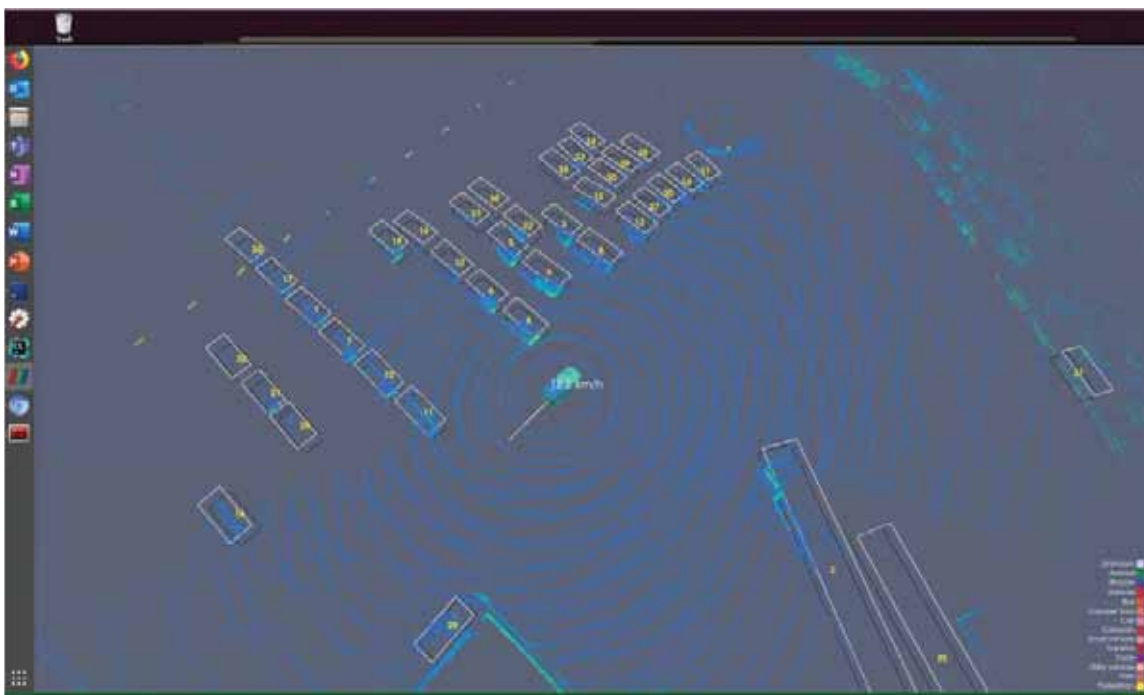
(note: we don't do end-to-end learning, might prove useful in the future)



Example of subcomponents



3D Object detection



Object detection – progress in 2019

Car

	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment	Compare
1	ATL			79.79 %	88.75 %	74.16 %	0.04 s	1 core @ 2.5 Ghz (Python)	
2	STD			79.71 %	87.95 %	75.09 %	0.08 s	GPU @ 2.5 Ghz (Python + C/C++)	
2. Yang, Y., Sun, S., Liu, X., Shen and J. Jia: <i>STD: Simple to Detect 3D Object Detector for Point Cloud</i> . ICCV 2019.									
3	MMLab-PartA ²			78.49 %	87.81 %	73.51 %	0.08 s	GPU @ 2.5 Ghz (Python + C/C++)	
3. Shi, Z., Wang, X., Wang and H. Li: <i>PartA²: Hier. 3D Part-Aware and Attentional Neural Network for Object Detection from Point Cloud</i> . arXiv preprint arXiv:1907.01670 2019.									
4	Patches - EMP			78.41 %	89.84 %	73.15 %	0.5 s	GPU @ 2.5 Ghz (Python)	
4. Lefner, A., Mitterecker, T., Adler, M., Hofmarcher, B., Hessler and S. Hochreiter: <i>Patch Refinement: Localized 3D Object Detection</i> . arXiv preprint arXiv:1910.04093 2019.									
5	ELE			78.35 %	86.95 %	73.33 %	0.1 s	GPU @ 2.5 Ghz (Python + C/C++)	
6	H3D-FusionRCNN			78.29 %	88.46 %	70.75 %	0.1 s	1 core @ 2.5 Ghz (C/C++)	
7	CP			78.11 %	86.40 %	71.63 %	0.1 s	1 core @ 2.5 Ghz (C/C++)	
8	GNN3D			77.70 %	86.42 %	72.71 %	1 s	GPU @ 2.5 Ghz (Python)	
9	MLF			77.62 %	86.21 %	67.68 %	0.05 s	GPU @ 2.0 Ghz (Python)	
10	liberATG-MMF			77.43 %	88.40 %	70.22 %	0.08 s	GPU @ 2.5 Ghz (Python)	
M. Liang*, B. Yang*, Y. Chen, H. Hu and B. Liptson: <i>Multi-Task Multi-Sensor Fusion for 3D Object Detection</i> . CVPR 2019.									
11	Fast Point R-CNN v1			77.40 %	85.29 %	70.24 %	0.06 s	GPU @ 2.5 Ghz (Python + C/C++)	
Y. Chen, S. Liu, X. Shen and J. Jia: <i>Fast Point R-CNN</i> . Proceedings of the IEEE international conference on computer vision (ICCV) 2019.									
12	Patches			77.20 %	88.67 %	71.82 %	0.15 s	GPU @ 2.0 Ghz	
J. Lefner, A. Mitterecker, T. Adler, M. Hofmarcher, B. Hessler and S. Hochreiter: <i>Patch Refinement: Localized 3D Object Detection</i> . arXiv preprint arXiv:1910.04093 2019.									
13	H3D-VoxelFPM			76.70 %	85.64 %	69.44 %	0.02 s	GPU @ 2.5 Ghz (Python + C/C++)	
B. Wang, J. An and J. Cao: <i>Voxel-FPM: multi-scale voxel feature aggregation in 3D object detection from point clouds</i> . arXiv preprint arXiv:1907.05286v2 2019.									
14	SDF			76.61 %	86.63 %	71.28 %	0.05 s	GPU @ 2.5 Ghz (Python + C/C++)	
15	3D InL Loss			76.50 %	86.16 %	71.39 %	0.08 s	GPU @ 2.5 Ghz (Python + C/C++)	
D. Zhou: <i>InL Loss for 3D Object Detection</i> . International Conference on 3D Vision (3DV) 2019.									



Object detection – progress 2019

Many improvements in 3D object detection methods

Car

	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment	Compare
1	ATL			79.79 %	88.75 %	74.16 %	0.04 s	1 core @ 2.5 Ghz (Python)	
2	STD			79.71 %	87.95 %	75.09 %	0.08 s	GPU @ 2.5 Ghz (Python + C/C++)	
2. Yang, Y., Sun, S., Liu, X., Shen and J. Jia: <i>STD: Simple to Detect 3D Object Detector for Point Cloud</i> . ICCV 2019.									
3	MMLab-PartA ²			78.49 %	87.81 %	73.51 %	0.08 s	GPU @ 2.5 Ghz (Python + C/C++)	
3. Shi, Z., Wang, X., Wang and H. Li: <i>PartA²: Hier. 3D Part-Aware and Attentional Neural Network for Object Detection from Point Cloud</i> . arXiv preprint arXiv:1907.01670 2019.									
4	Patches - EMP			78.41 %	89.84 %	73.15 %	0.5 s	GPU @ 2.5 Ghz (Python)	
4. Lefner, A., Mitterecker, T., Adler, M., Hofmarcher, B., Hessler and S. Hochreiter: <i>Patch Refinement: Localized 3D Object Detection</i> . arXiv preprint arXiv:1910.04093 2019.									
5	ELE			78.35 %	86.95 %	73.33 %	0.1 s	GPU @ 2.5 Ghz (Python + C/C++)	
6	H3D-FusionRCNN			78.29 %	88.46 %	70.75 %	0.1 s	1 core @ 2.5 Ghz (C/C++)	
7	CP			78.11 %	86.40 %	71.63 %	0.1 s	1 core @ 2.5 Ghz (C/C++)	
8	GNN3D			77.70 %	86.42 %	72.71 %	1 s	GPU @ 2.5 Ghz (Python)	
9	MLF			77.62 %	86.21 %	67.68 %	0.05 s	GPU @ 2.0 Ghz (Python)	
10	liberATG-MMF			77.43 %	88.40 %	70.22 %	0.08 s	GPU @ 2.5 Ghz (Python)	
M. Liang*, B. Yang*, Y. Chen, H. Hu and B. Liptson: <i>Multi-Task Multi-Sensor Fusion for 3D Object Detection</i> . CVPR 2019.									
11	Fast Point R-CNN v1			77.40 %	85.29 %	70.24 %	0.06 s	GPU @ 2.5 Ghz (Python + C/C++)	
Y. Chen, S. Liu, X. Shen and J. Jia: <i>Fast Point R-CNN</i> . Proceedings of the IEEE international conference on computer vision (ICCV) 2019.									
12	Patches			77.20 %	88.67 %	71.82 %	0.15 s	GPU @ 2.0 Ghz	
J. Lefner, A. Mitterecker, T. Adler, M. Hofmarcher, B. Hessler and S. Hochreiter: <i>Patch Refinement: Localized 3D Object Detection</i> . arXiv preprint arXiv:1910.04093 2019.									
13	H3D-VoxelFPM			76.70 %	85.64 %	69.44 %	0.02 s	GPU @ 2.5 Ghz (Python + C/C++)	
B. Wang, J. An and J. Cao: <i>Voxel-FPM: multi-scale voxel feature aggregation in 3D object detection from point clouds</i> . arXiv preprint arXiv:1907.05286v2 2019.									
14	SDF			76.61 %	86.63 %	71.28 %	0.05 s	GPU @ 2.5 Ghz (Python + C/C++)	
15	3D InL Loss			76.50 %	86.16 %	71.39 %	0.08 s	GPU @ 2.5 Ghz (Python + C/C++)	
D. Zhou: <i>InL Loss for 3D Object Detection</i> . International Conference on 3D Vision (3DV) 2019.									
16	E-Grader			76.39 %	87.36 %	68.69 %	0.37 s	GPU @ 2.5 Ghz (Python + C/C++)	
B. Wang and X. An: <i>Grader: Gradient-based Feature Grading for 3D Object Detection</i> . AAAI 2019.									
17	AttNet3D-Net			76.29 %	84.71 %	68.18 %	0.09 s	1 core @ 3.3 Ghz (C/C++)	
18	SDFNet			76.24 %	87.34 %	71.16 %	0.39 s	GPU @ 2.5 Ghz (Python)	
19	SelfSup3DNet			76.13 %	86.04 %	70.76 %	0.08 s	1 core @ 2.5 Ghz (Python)	
20	SDFNet2			76.09 %	84.88 %	70.57 %	0.05 s	1 core @ 2.5 Ghz (Python + C/C++)	
21	ES		code	76.04 %	84.31 %	70.77 %	0.05 s	1 core @ 2.5 Ghz (C/C++)	
22	ES-RCNN			76.00 %	86.34 %	70.87 %	0.1 s	1 core @ 2.5 Ghz (Python)	
23	SECOND-V1.3		code	75.96 %	84.65 %	68.71 %	0.04 s	GPU @ 2.0 Ghz (Python + C/C++)	
24	ESNet			75.94 %	84.39 %	68.92 %	0.039s	GPU @ 2.5 Ghz (Python + C/C++)	
25	MMF			75.91 %	84.46 %	68.69 %	0.3 s	GPU @ 2.5 Ghz (C/C++)	
26	PartRCNN-revised			75.81 %	86.23 %	68.89 %	0.1 s	GPU @ 2.5 Ghz (Python + C/C++)	
27	ContextNet			75.76 %	85.40 %	70.29 %	0.05 s	GPU @ 2.5 Ghz (Python)	
28	MMLab-PartA ² CH		code	75.44 %	86.50 %	70.70 %	0.1 s	GPU @ 2.5 Ghz (Python + C/C++)	
S. Shi, Z. Wang and H. Li: <i>PartA²CH: Hierarchical Attentional Aggregation and Attentional Feature-wise Grouping</i> . Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2019.									
29	AS3DNet		code	75.43 %	86.10 %	68.88 %	0.0047s	1 core @ 2.5 Ghz (Python)	
A. Wang and K. Zhou: <i>A Simple but Effective 3D Object Detector</i> . NeurIPS 2019.									
30	ES-net2			75.30 %	85.72 %	69.90 %	0.04 s	GPU @ 3.5 Ghz (Python)	

Object detection – progress 2019

Many improvements in 3D object detection methods
 But still a long way to go for pedestrians.

Pedestrian

Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment	Compare
1	IP3D		44.37 %	55.07 %	40.05 %	0.2 s	GPU @ 2.5 Ghz (Python + C/C++)	📄
Z. Yang, Y. Sun, S. Liu, X. Shen and J. Jia: IP3D: Interative Point-based Object Detector for Point Cloud . CoRR 2018.								
2	TANet		44.34 %	53.72 %	40.49 %	0.035s	GPU @ 2.5 Ghz (Python + C/C++)	📄
3	A-VoxelNet		44.30 %	53.66 %	40.43 %	0.029 s	GPU @ 2.5 Ghz (Python)	📄
4	F-ConvNet		43.38 %	52.16 %	38.80 %	0.47 s	GPU @ 2.5 Ghz (Python + C/C++)	📄
Z. Wang and K. Jia: Custom Geonets: Sparse Features to Aggregate Local Point-Wise Features for Amodal 3D Object Detection . IROS 2019.								
5	VMVS		43.27 %	53.44 %	39.51 %	0.25 s	GPU @ 2.5 Ghz (Python)	📄
J. Xu, A. Pan, S. Walsh and S. Waslander: Improving 3D object detection for pedestrians with virtual multi-view synthesis orientation estimation . IROS 2019.								
6	STD		42.47 %	53.29 %	38.35 %	0.08 s	GPU @ 2.5 Ghz (Python + C/C++)	📄
Z. Yang, Y. Sun, S. Liu, X. Shen and J. Jia: STD: Sparse-to-Dense 3D Object Detector for Point Cloud . ICCV 2019.								
7	AVOID-EPH	code	42.27 %	50.46 %	39.04 %	0.1 s	Titan X (Pascal)	📄
J. Xu, M. Muzaffar, J. Lee, A. Harakeh and S. Waslander: Joint 3D Proposal Generation and Object Detection from View Awareness . IROS 2018.								
8	F-PointNet	code	42.15 %	50.53 %	38.08 %	0.17 s	GPU @ 3.0 Ghz (Python)	📄
C. Qi, W. Liu, C. Wu, H. Su and L. Guibas: Frustum PointNets for 3D Object Detection from RGB-D Data . arXiv preprint arXiv:1711.08488 2017.								
9	PointPillars	code	41.92 %	51.45 %	38.89 %	16 ms	1080Ti GPU and Intel i7 CPU	📄
A. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang and O. Beijbom: PointPillars: Fast Encoders for Object Detection from Point Clouds . CVPR 2019.								
10	spBBM	code	41.52 %	49.17 %	39.08 %	0.10 s	1 core @ 2.5 Ghz (C/C++)	📄
K. Shin: Improving a Quality of 3D Object Detection by Spatial Transformation Mechanism . arXiv preprint arXiv:1910.04853 2019.								

Car

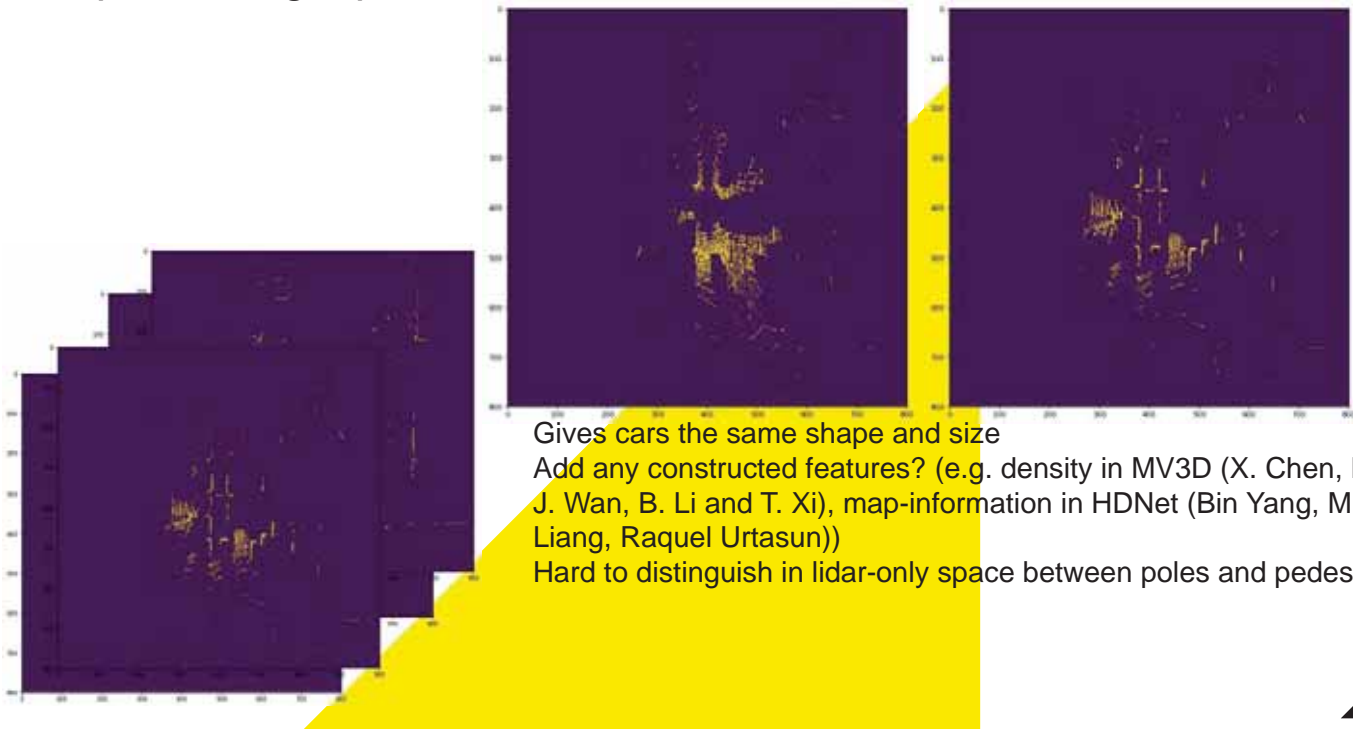
Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment	Compare
1	ASL		79.79 %	88.75 %	74.16 %	0.04 s	1 core @ 2.5 Ghz (Python)	📄
2	3D		79.79 %	87.95 %	75.09 %	0.08 s	GPU @ 2.5 Ghz (Python + C/C++)	📄
3	3DMatch	code	78.49 %	87.81 %	73.53 %	0.08 s	GPU @ 2.5 Ghz (Python + C/C++)	📄
Z. Yang, Y. Sun, S. Liu, X. Shen and J. Jia: 3DMatch: A Self-Supervised 3D Object Detection on Point Cloud . arXiv preprint arXiv:1907.04898 2019.								
4	FastNet-3D	code	78.41 %	89.84 %	73.15 %	0.5 s	GPU @ 2.5 Ghz (Python)	📄
A. Lichtenberg, A. Mouskouri, T. Adel, M. Himmelsbach, S. Roth and S. Roth: FastNet: Fast 3D Object Detection on Point Clouds . arXiv preprint arXiv:1910.04853 2019.								
5	3D		78.35 %	86.95 %	73.33 %	0.1 s	GPU @ 2.5 Ghz (Python + C/C++)	📄
6	3DMatch		78.29 %	88.46 %	76.75 %	0.1 s	1 core @ 2.5 Ghz (C/C++)	📄
7	3D		78.15 %	86.46 %	71.63 %	0.1 s	1 core @ 2.5 Ghz (C/C++)	📄
8	3DMatch	code	77.70 %	86.40 %	72.71 %	1 s	GPU @ 2.5 Ghz (Python)	📄
9	3DMatch		77.64 %	86.35 %	73.48 %	0.09 s	GPU @ 2.5 Ghz (Python)	📄
10	3DMatch		75.30 %	85.72 %	83.80 %	0.04 s	GPU @ 3.5 Ghz (Python)	📄

Representing input 3D bounding box detection

- Lidar measurements can be seen as
 - Unordered set of measurements (e.g. pointnet)
 - Birds—eye view (e.g. mv3d / pixor)
 - In terms of lidar sensor intrinsics (lidar front view)

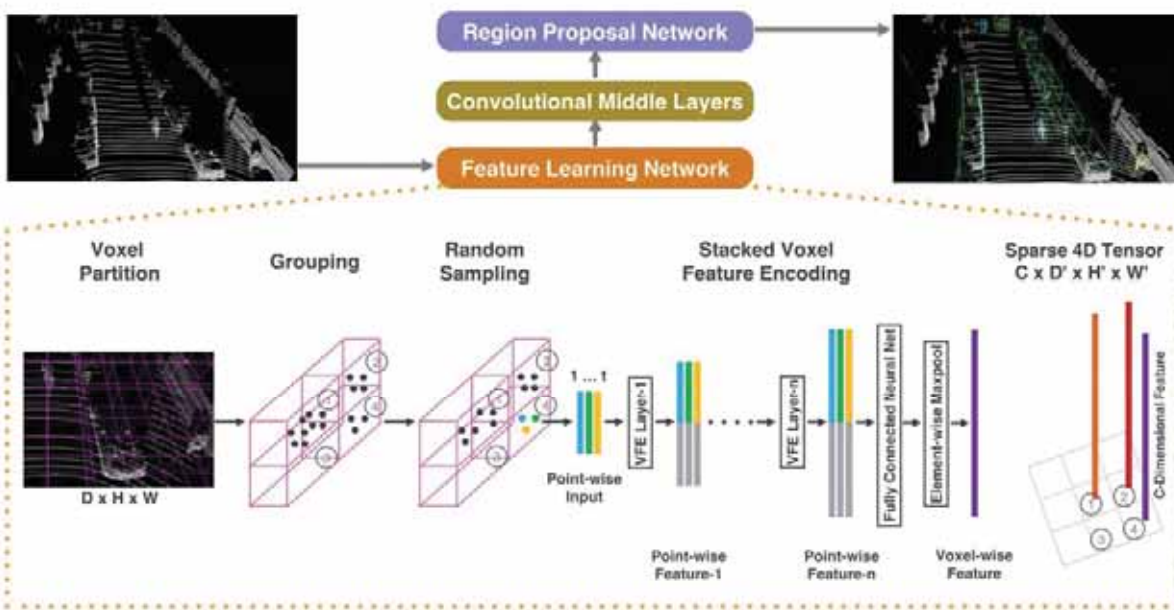


Representing input 3D bounding box detection – BEV projection



Representing input 3D bounding box detection – Voxels / Pillars

VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. Yin Zhou, Oncel Tuzel



Representing input 3D bounding box detection – Frustums

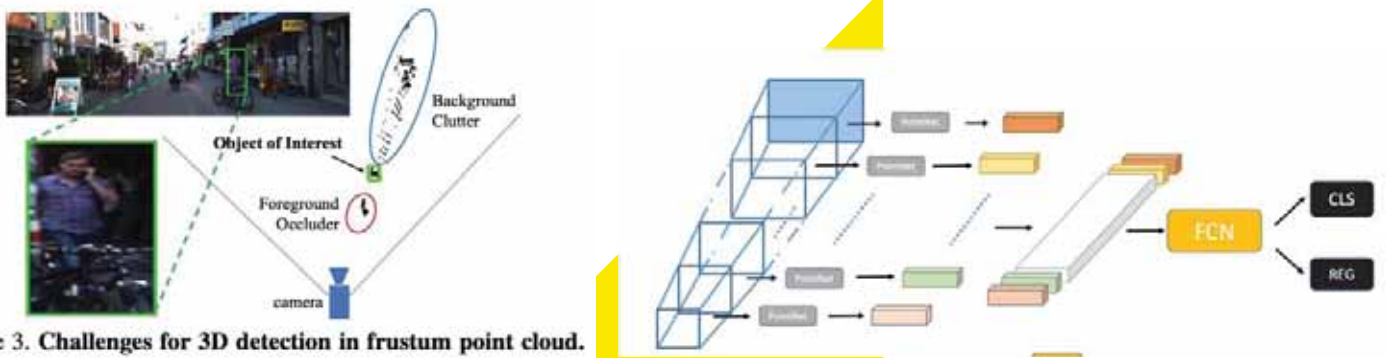
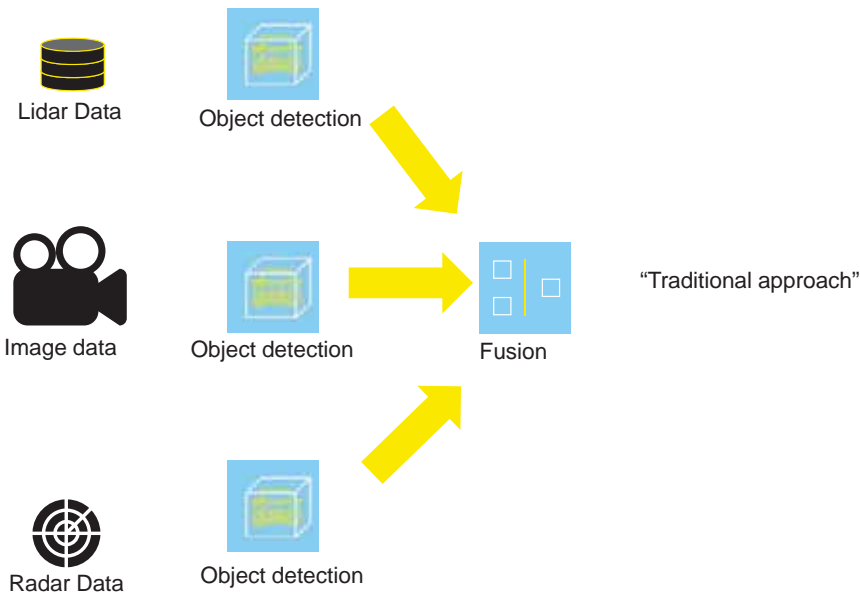


Figure 3. Challenges for 3D detection in frustum point cloud.

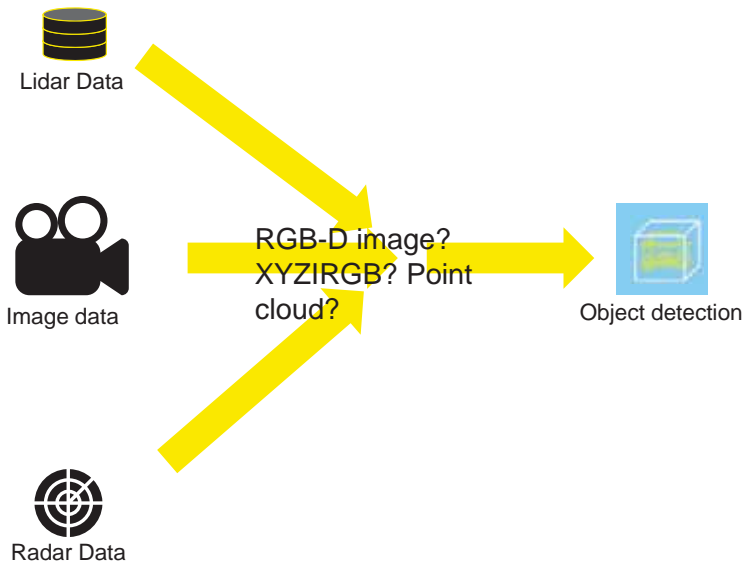
e.g. Frustum PointNets for 3D Object Detection from RGB-D Data (Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, Leonidas J. Guibas)
 Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection (Zhixin Wang, Kui Jia) -> presented at IROS 2019 ;)



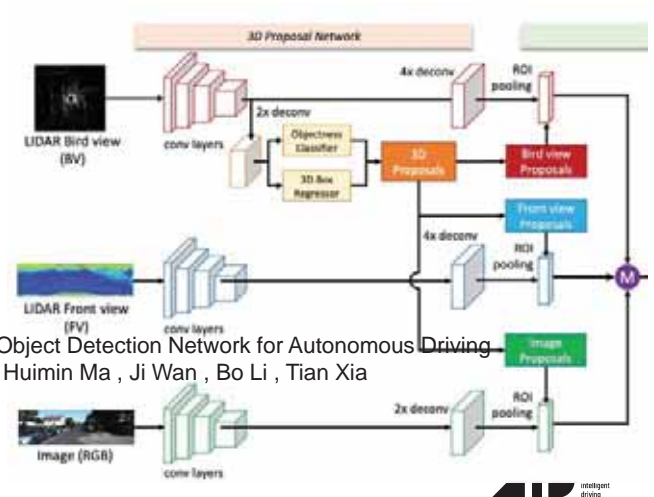
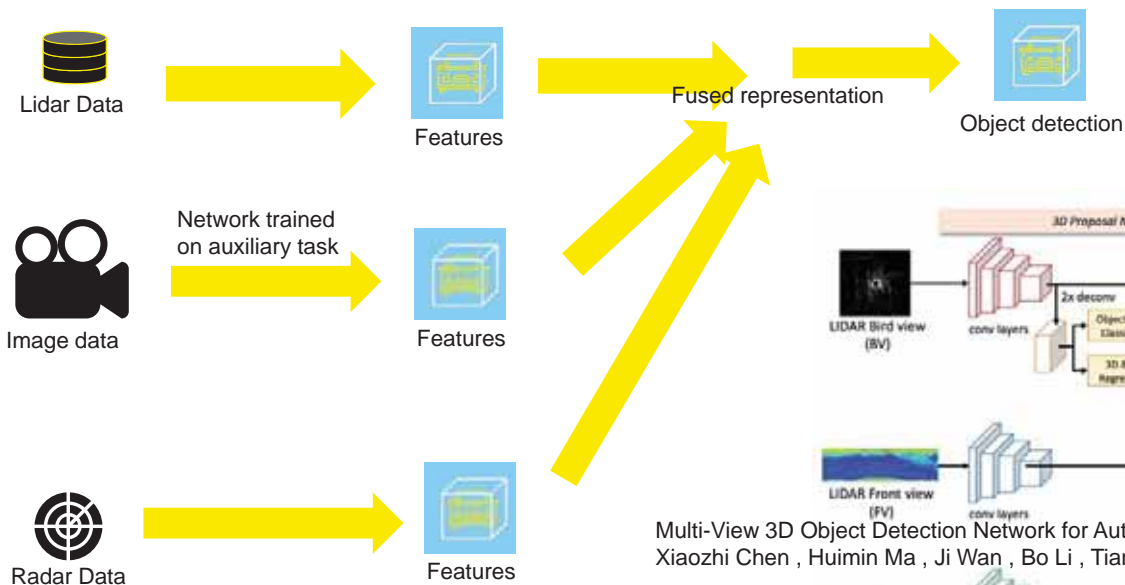
Where to fuse data for autonomous driving



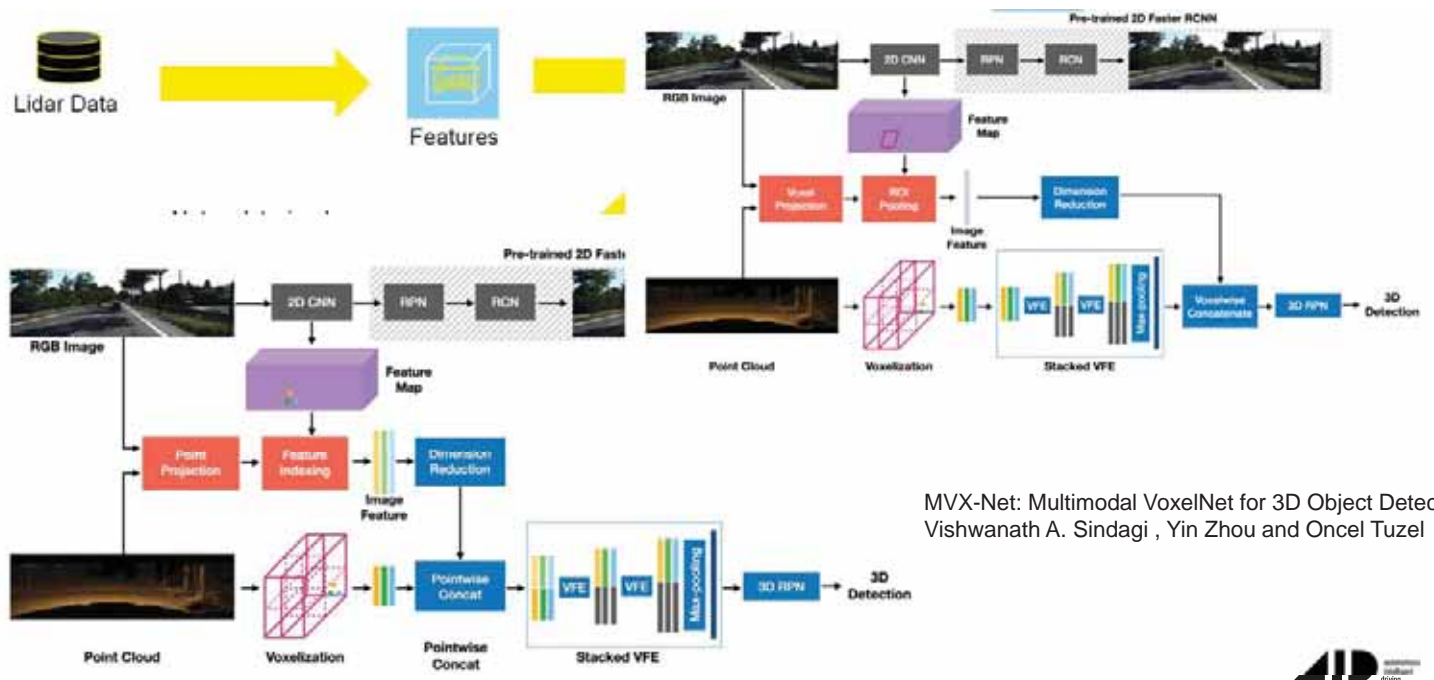
Where to fuse data for autonomous driving



Where to fuse data for autonomous driving



Where to fuse data for autonomous driving



MVX-Net: Multimodal VoxelNet for 3D Object Detection
 Vishwanath A. Sindagi, Yin Zhou and Oncel Tuzel



Auxiliary tasks and domain adaptation / simulation

Improving your performance with cheap(er) data

- ▲ Neural networks are still data-hungry
- ▲ Domain adaptation is a hot and difficult topic. For neural networks it's not known what features should be in a simulation environment to transfer well to the real world.
- ▲ Multi-Task Multi-Sensor Fusion for 3D Object Detection Ming Liang, Bin Yang, Yun Chen, Rui Hu, Raquel Urtasun

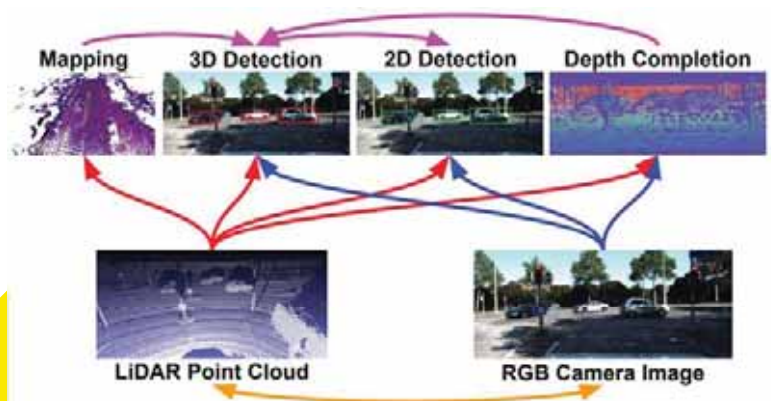
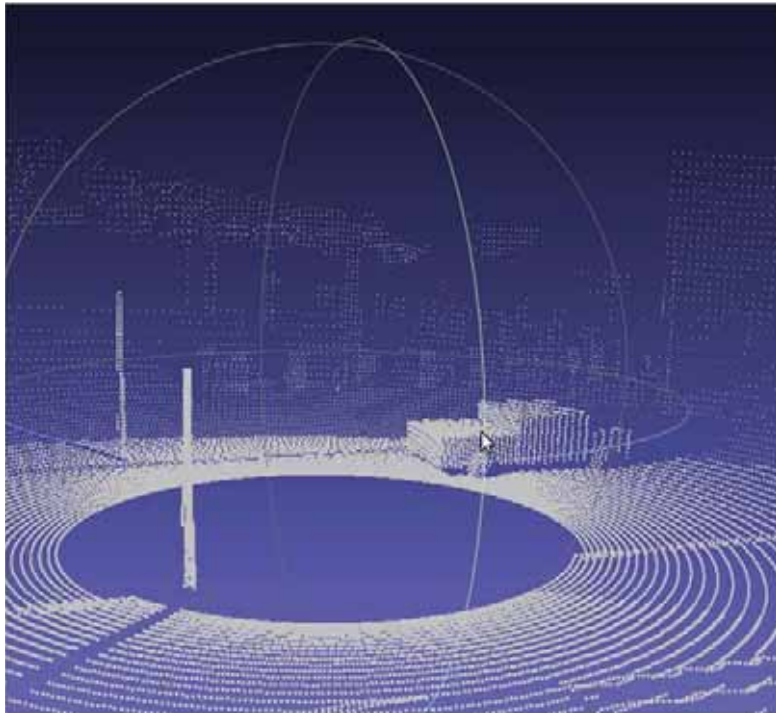


Figure 1. Different sensors (bottom) and tasks (top) are complementary to each other. We propose a joint model that reasons on two sensors and four tasks, and show that the target task - 3D object detection can benefit from multi-task learning and multi-sensor fusion.



Closing the simulation gap

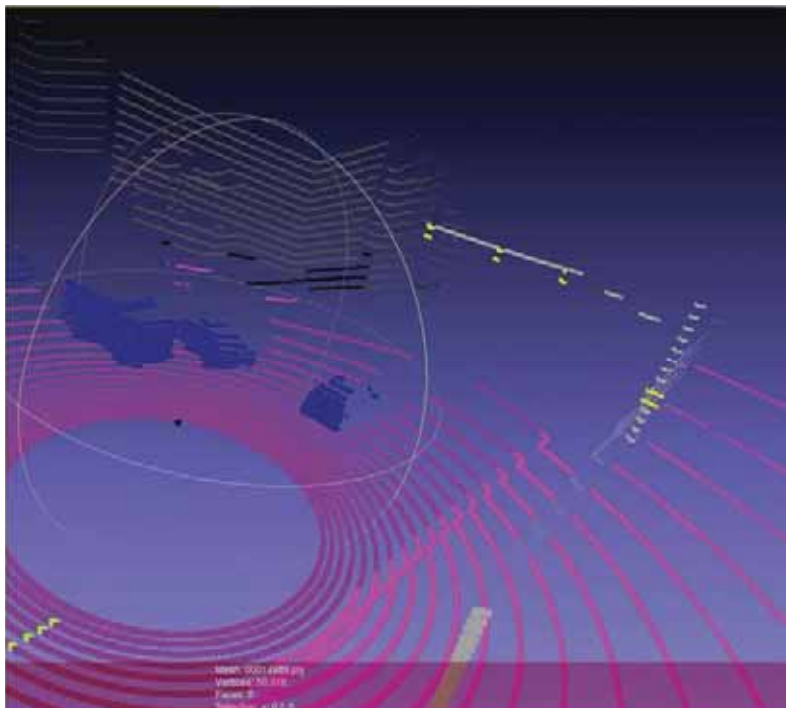


A 3D point cloud as seen through the eyes of Carla

- ▲ Very useful for finding the limit of your algorithm in a fixed environment
- ▲ Varying results in academic literature on how to make simulated data useful.



Domain adaptation is a hard task

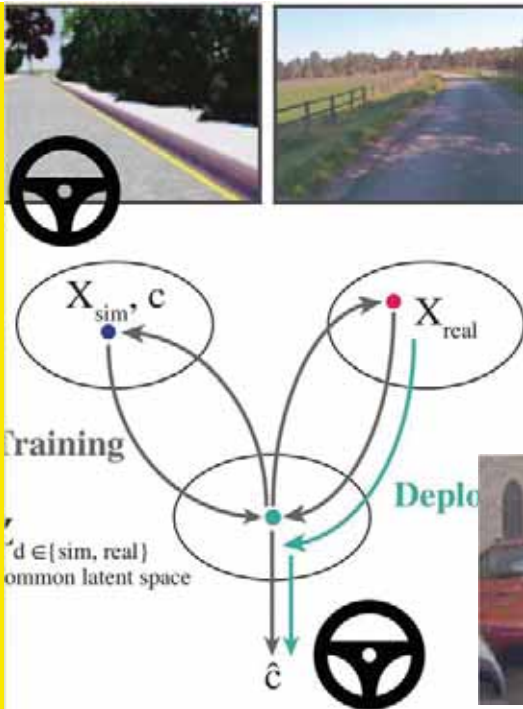


What features in the world are important?

- ▲ Is an accurate geometry of cars important?
- ▲ Do we have to model intensity of lidar?
- ▲ And should car-windows return lidar points?
- ▲ And should windows reflect objects and place them inside buildings?
- ▲ Should trees in simulation move/sway to solve problems with trees becoming dynamic objects in the real world?
- ▲ How about different seasons?
- ▲ GANs can be useful to create more hard-to-obtain data, but it's hard to generate data which broadens the perception capabilities of networks in a reasonable way.



Domain adaptation

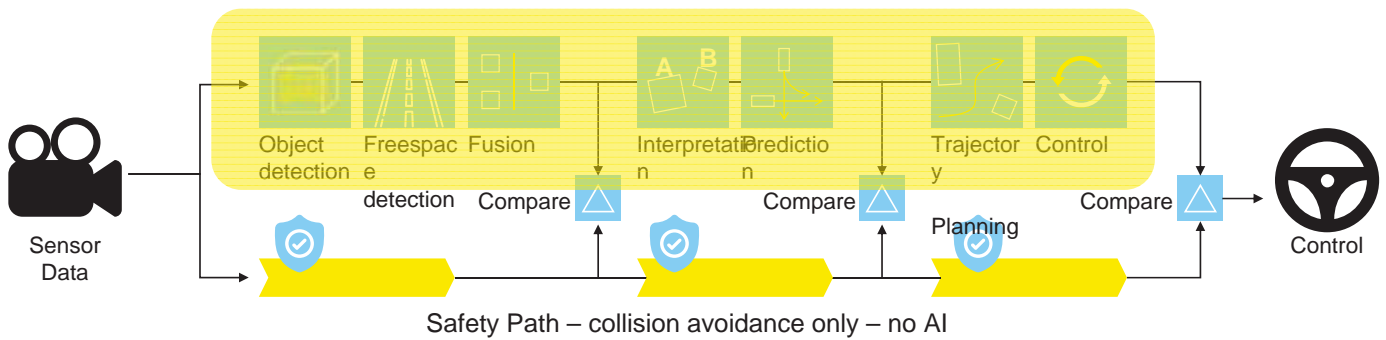


Creating a common latent space

- Learning to Drive from Simulation without Real World Labels Alex Bewley, Jessica Rigley, Yuxuan Liu, Jeffrey Hawke, Richard Shen, Vinh-Dieu Lam, Alex Kendall
- SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, Kurt Keutzer



Estimating, evaluating, and passing along uncertainty across components is still an unsolved problem



Sampling-free Epistemic Uncertainty Estimation Using Approximated Variance Propagation

Janis Postels^{1,2}

Francesco Ferroni²

Huseyin Coskun¹

Nassir Navab¹

Federico Tombari^{1,3}

¹Technical University Munich



²Autonomous Intelligent Driving GmbH



³Google



Types of Uncertainty

Aleatoric

- Inherent to the data
- Unavoidable

vs

Epistemic

- Model uncertainty
- Can be explained away by data

→ Mixture Density Networks [1] ?

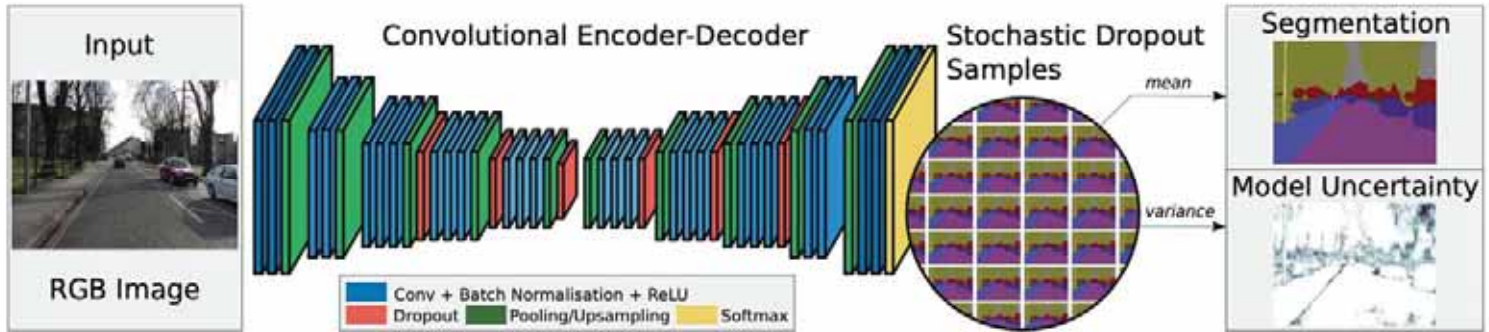
[1] C. M. Bishop. "Mixture Density Networks." NCRG/94/004, 1994.



Epistemic Uncertainty for Large Neural Networks

Stochastic regularization ~ Bayesian Neural Network [1][2][3]

Approach: Train Neural Network with stochastic regularization and keep stochastic process at inference.



[1] Y. Gal. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." ICML, 2016.

[2] D. P. Kingma. "Variational dropout and the local reparameterization trick." NIPS, 2015.

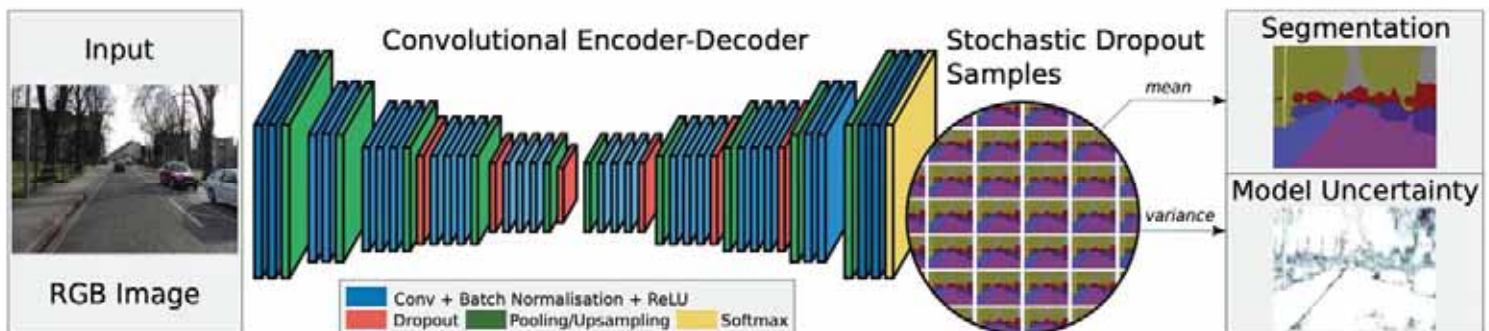
[3] S. Wang. "Fast dropout training." ICML, 2013.



Epistemic Uncertainty for Large Neural Networks

Stochastic regularization ~ Bayesian Neural Network [1][2][3]

Approach: Train Neural Network with stochastic regularization and keep stochastic process at inference.



→ **Requires sampling during inference time**

[1] Y. Gal. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." ICML, 2016.

[2] D. P. Kingma. "Variational dropout and the local reparameterization trick." NIPS, 2015.

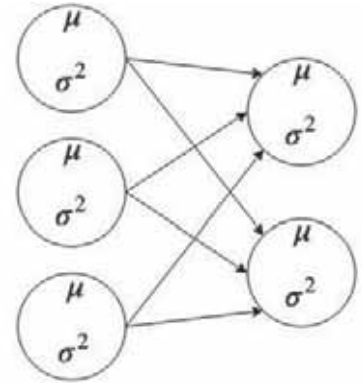
[3] S. Wang. "Fast dropout training." ICML, 2013.



Solution: Propagate Uncertainties Using Error Propagation

Noise layer (e.g. dropout) places **distribution over activations**

- Mean propagated by normal forward propagation
- **Covariance** propagated using **error propagation**



$$\Sigma_{l+1} = J^T \Sigma_l J$$



Results: Semantic Segmentation with Bayesian Segnet [1]

Original Image Ground Truth



MC Dropout [2]



Our

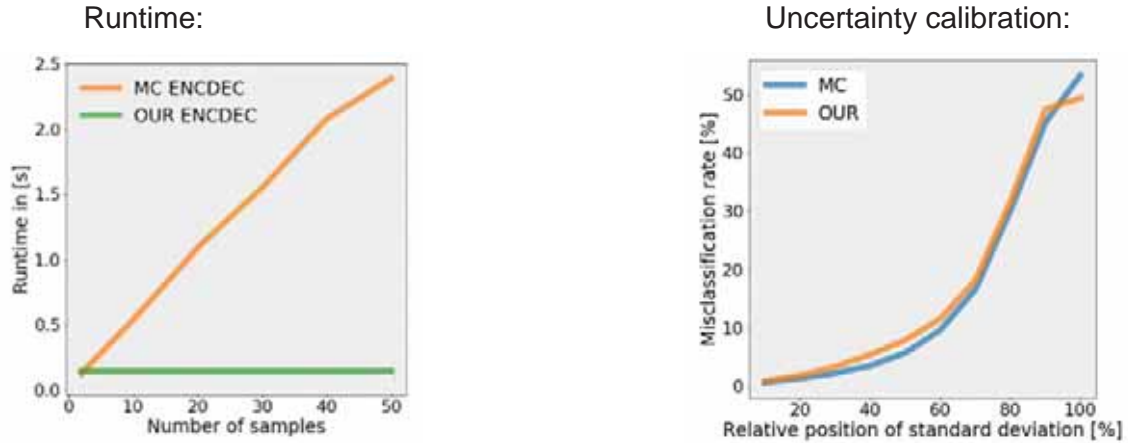


[1] A. Kendall. "Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding." BMVC, 2017.

[2] Y. Gal. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." ICML, 2016.



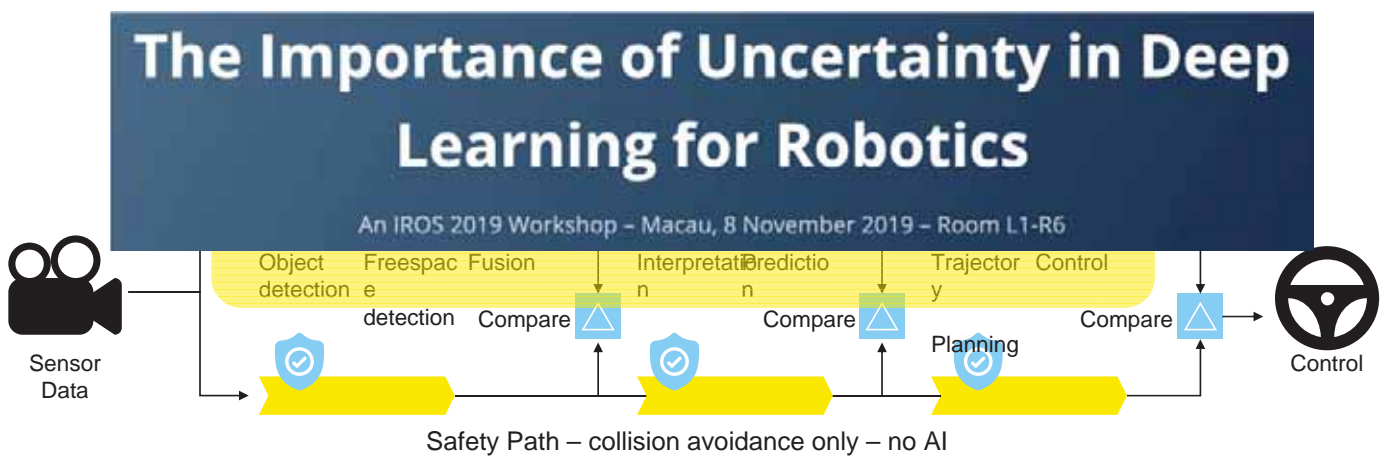
Results: Semantic Segmentation with Bayesian Segnet [1]



[1] A. Kendall. "Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding." BMVC, 2017.



Estimating, evaluating, and passing along uncertainty across components is still an unsolved problem



The importance of collecting the right data: the long-tail distribution of situations



The importance of collecting the right data: the long-tail distribution of situations



Public datasets

BEFORE JANUARY 2018

- Oxford robotcar
- Cityscapes
- Kitti
- Daimler pedestrian benchmark
- Mapillary vistas
- Udacity



Public datasets

AFTER JANUARY 2018:

- Berkeley DeepDrive
- Apolloscape
- Comma 2k19
- Lyft
- Nuscenes
- Semantic KITTI
- Unsupervised LLamas



Level 5

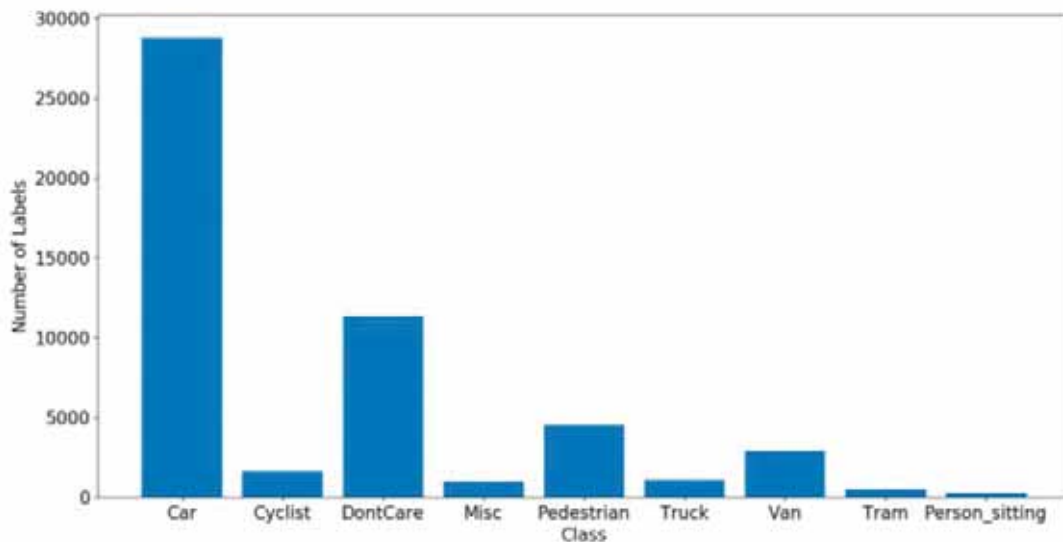


It's not about the amount of data! Address the long-tail distribution of situations



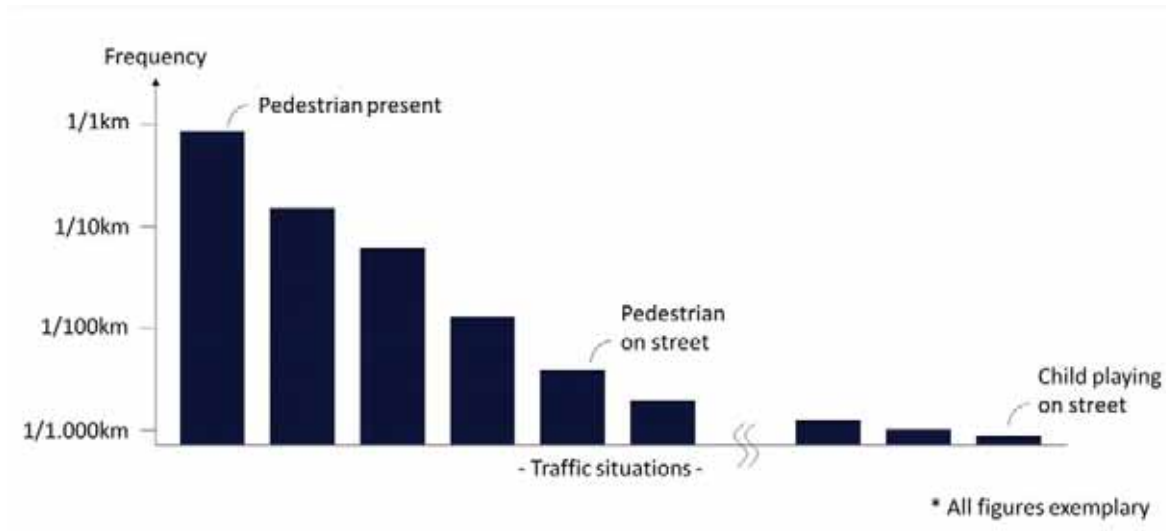
Addressing the long-tail distribution of situations

- ▲ The famous KITTI dataset has many cars, but barely any cyclists

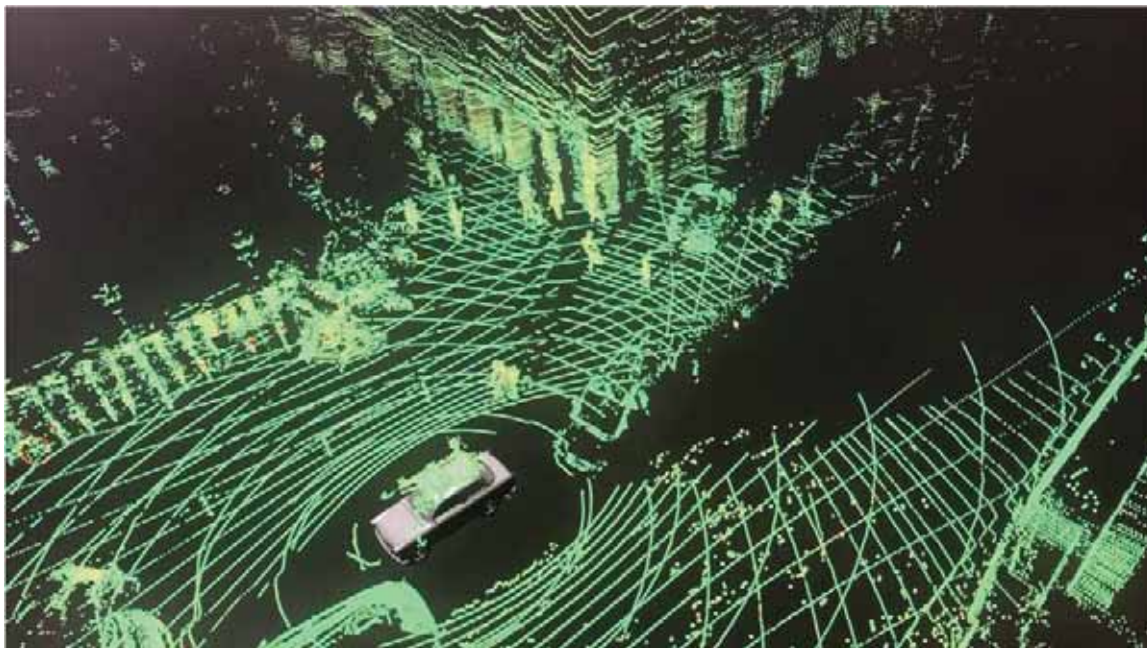


The long-tail distribution of traffic situations

Not only presence is important: the context in which we see the class is also important



Example of rare data



Understanding situation specific model performance



06.11.2019 AI for Autonomous Driving

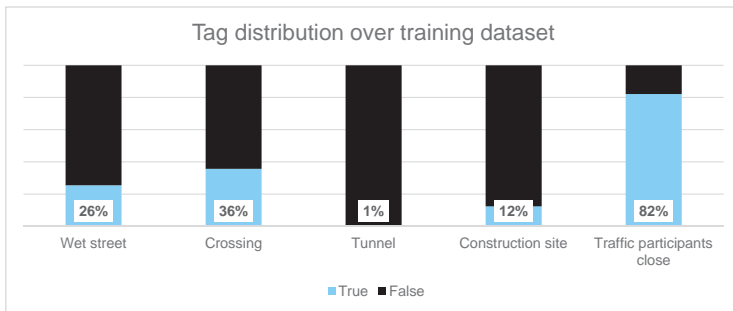
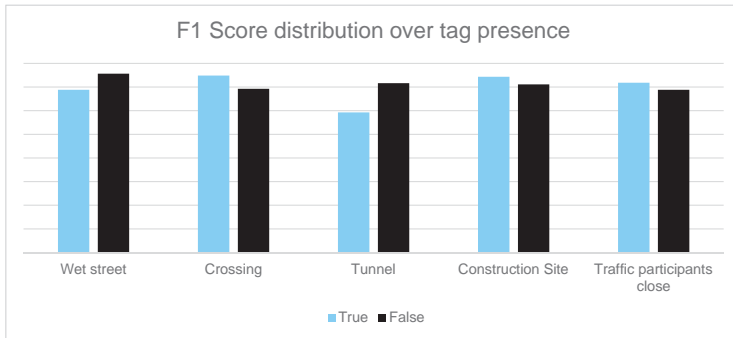
60

Popular Machine Learning KPIs aren't representing real-world performance well

	Classification vs Safety	Semantic Segmentation vs. Distance	Object Detection: Relevance in traffic context																																
Common KPIs	F1 Score	Intersection over Union (IoU)	mAP, F1 Score																																
	<table border="1"> <thead> <tr> <th colspan="2"></th> <th colspan="4">Ground Truth</th> </tr> <tr> <th colspan="2"></th> <th>Car</th> <th>Truck</th> <th>Human</th> <th>Road</th> </tr> </thead> <tbody> <tr> <th rowspan="4">Prediction</th> <th>Car</th> <td>Green</td> <td>Light Green</td> <td>Orange</td> <td>Yellow</td> </tr> <tr> <th>Truck</th> <td>Light Green</td> <td>Green</td> <td>Orange</td> <td>Yellow</td> </tr> <tr> <th>Human</th> <td>Yellow</td> <td>Yellow</td> <td>Green</td> <td>Yellow</td> </tr> <tr> <th>Road</th> <td>Red</td> <td>Red</td> <td>Red</td> <td>Green</td> </tr> </tbody> </table> <p>Safety criticality →</p>			Ground Truth						Car	Truck	Human	Road	Prediction	Car	Green	Light Green	Orange	Yellow	Truck	Light Green	Green	Orange	Yellow	Human	Yellow	Yellow	Green	Yellow	Road	Red	Red	Red	Green	<p>Are missed close pixels really as bad as distant ones?</p>
		Ground Truth																																	
		Car	Truck	Human	Road																														
Prediction	Car	Green	Light Green	Orange	Yellow																														
	Truck	Light Green	Green	Orange	Yellow																														
	Human	Yellow	Yellow	Green	Yellow																														
	Road	Red	Red	Red	Green																														
Better	Consider safety impact	Bin over distance	Consider Time-To-Collision (TTC) and Closest In-Path Vehicle (CIPV)																																



Model performance depends on environment conditions



Scenarios that are underrepresented in the training data cause low model performance.

Model performance can be mapped to ODD attributes to detect these scenarios.

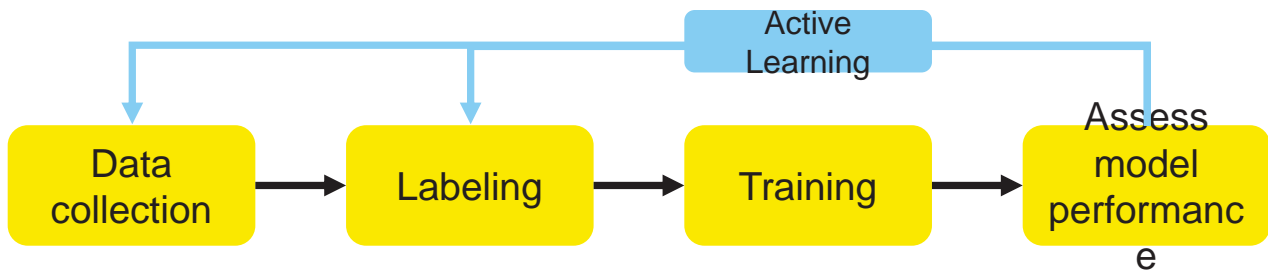
Targeted data collection provide mitigation.



Using active learning to identify valuable data

Active learning: knowing where you are weak while selecting data

- Active Learning: Mechanism that controls training dataset extension according to model performance



- For scenarios where model performance is low: Collect data in the field or simulate corresponding environments



Active learning via trained Auto-Tagger

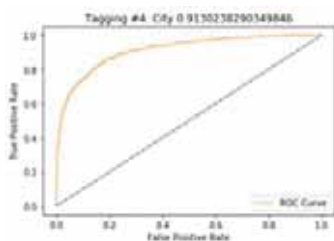
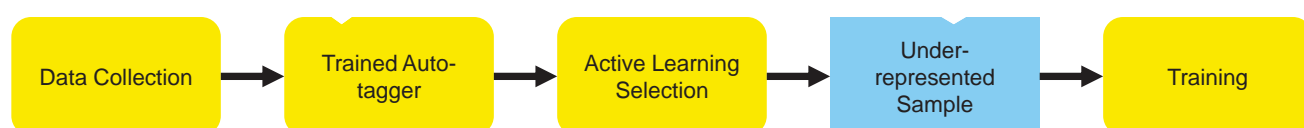


Image properties
 Roadworks: True
 Cloudy: True
 Traffic participants close: False
 Traffic participants far: True
 City: False
 Rainy: False
 Day: True



The speed at which your AI can adapt will be an important factor for autonomous driving



11/6/2019 AID PPT Master 2019

66

The human brain can quickly adapt to new situations

Germany legalises e-scooters but bans them from the pavement

COMMENTS

by Alice Tidey · 25/05/2019



German lawmakers voted on Friday to allow e-scooters to take to the streets making the UK the last major European economy to still ban them.

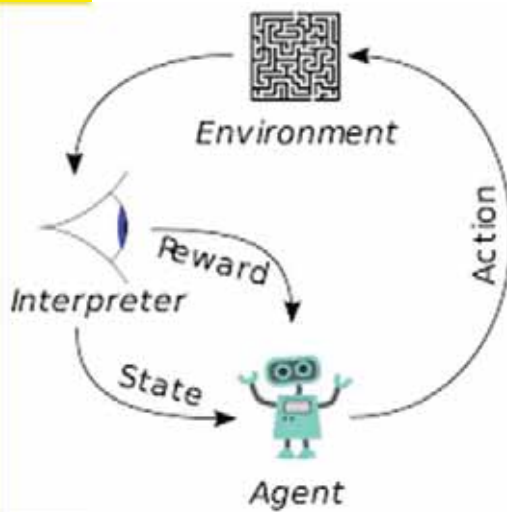
The text approved by the Bundesrat on Friday enables battery-powered scooters to circulate on roads and cycle paths but forbids them from being used on the pavements. Users must be 14 or over and must respect a 20 kilometres per hours speed limit.

AI should do so too

- ▲ Traffic changes over time, and is even seasonal
- ▲ A self-driving car should be able to adapt to new objects appearing over time
- ▲ A self-driving car should be able to adapt to unknown objects



AI needs data to adapt to new situations



The loop from observation to deployment should be as short as possible

- ▲ Perceive weird object
- ▲ Label this object
- ▲ Train a new model with new data
- ▲ Verify that this model is abiding everything you expect
- ▲ Deploy a new model

Sony Trains ResNet-50 on ImageNet in 224 Seconds



NOV 26 2018 • 7 MIN READ

Researchers from Sony announced that they trained a machine learning ResNet 50 architecture on ImageNet, an image database organized according to the WordNet hierarchy, in only 224 seconds. The resulting network has a top-1 accuracy of 75% on the validation set of ImageNet. The researchers achieved this record by using 2,100 Tesla V100 Tensor Core GPUs from NVIDIA. Besides this new timing record, they also got a 90% GPU scaling efficiency using 1,088 Tesla V100 Tensor Core GPUs.



Conclusion

- ▲ Self-driving cars will improve the lives of humans (and allow dogs to transport themselves to the park)
- ▲ Lidar, camera and radar are the main sensors for self-driving cars, and each of them is important.
- ▲ Many advances in methods for 3D bounding box detection, but many interesting challenges are remaining.
- ▲ Data is important for autonomous vehicles, and adaptability of your AI is an important factor.



THANK YOU



AID - Autonomous
Intelligent Driving
GmbH
Ungererstr. 69
D-80805 München
aid-driving.eu



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session I

Machine & Deep Learning

- **Title: Transformation-adversarial network for road detection in LIDAR rings, and model-free evidential road grid mapping**
Authors: E. Cappelier, F. Davoine, V. Cherfaoui, Y. Li
- **Title: End-to-End Deep Neural Network Design for Short-term Path Planning**
Authors: M.Q. Dao, D. Lanza, V. Frémont



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Transformation-adversarial network for road detection in LIDAR rings, and model-free evidential road grid mapping

Edouard CAPELLIER^{1,2}, Franck DAVOINE², Veronique CHERFAOUI², You LI¹

Abstract— We propose a deep learning approach to perform road-detection in LIDAR scans, at the point level. Instead of processing a full LIDAR point-cloud, LIDAR rings can be processed individually. To account for the geometrical diversity among LIDAR rings, an homothety rescaling factor can be predicted during the classification, to realign all the LIDAR rings and facilitate the training. This scale factor is learnt in a semi-supervised fashion. A performant classification can then be achieved with a relatively simple system. Furthermore, evidential mass values can be generated for each point from an observation of the conflict at the output of the network, which enables the classification results to be fused in evidential grids. Experiments are done on real-life LIDAR scans that were labelled from a lane-level centimetric map, to evaluate the classification performances.

I. INTRODUCTION

LIDAR sensors are traditionally used within occupancy grid mapping frameworks, to detect obstacles and infer the traversability of the environment. Evidential occupancy grid mapping frameworks usually assume that the ground is fully traversable, and evaluate the occupancy of cells from strong geometrical assumptions [1]–[3].

Yet, the applicability of such systems, in the context of autonomous driving, can be limited. First of all, they might fail to generate appropriate results, when the geometrical model they are based on is not satisfied anymore, which is likely to occur in complex urban areas. For example, the flat world assumption is not satisfied anymore at a speed bump. Then, areas that are traversable by an urban autonomous vehicle usually belong to the road: modelling the ground is thus not sufficient in most driving situations. Road detection in LIDAR scans is thus crucial, when aiming to implement evidential occupancy grid mapping algorithms in autonomous systems, that are intended to drive in urban areas. The use of machine learning could leverage the need for strong geometrical assumptions, as the system could be able to learn how to behave on edge-cases (speed-bumps, for instance), instead of relying on strong geometrical assumptions.

Inspired by the recent PointNet architecture [4] and novel advances in evidential classification [5], we propose to rely on a neural network that processes LIDAR rings individually, and can be used to output evidential mass values for each LIDAR point. Being able to represent the output of the neural

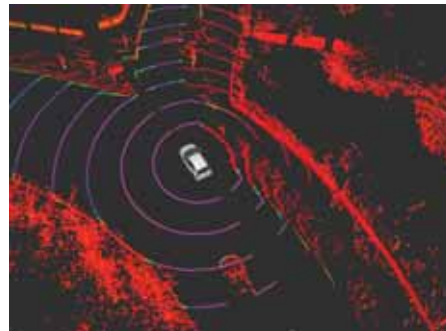


Fig. 1: Example of classification result. The grey ego-vehicle drives towards the road exit. The purpler a point is, the more likely it is to be on the road.

network as evidential mass values is particularly valuable when trying to understand what was learnt, since the total amount of knowledge available at each position can be quantified. Moreover, the evidential outputs of the network can directly be used in a model-free evidential grid mapping framework.

The paper is organized as follows: in Section II, we propose a short literature review ; Section III presents how evidential mass values can be obtained from a neural network that was trained on coarse labels ; Section IV presents the ring-level neural network that we propose to perform road detection ; Section V presents the data collection and evaluation procedures used to train and evaluate the classifier and finally, Section VI presents a simple model-free evidential grid mapping system relying on the proposed classifier.

II. LITERATURE REVIEW

A. Evidential grid mapping from LIDAR scans

Yu et al. [1] originally proposed an evidential sensor model to build polar occupancy grids from LIDAR scans. Based on the angular resolution and the beam divergence of the sensor, a polar missed detection rate was estimated, and a false alarm rate was empirically defined. From a ground-detection step relying on a flat-world assumption, the belief in the occupancy of each grid cell was then evaluated over time, according to an evidential framework. Such evidential polar grids however have to be interpolated, and mapped into a Cartesian coordinate system to perform fusion over time, at the cost of a loss in the correctness of the model. We ourselves proposed in [2] to evaluate a cartesian missed detection rate, to tackle this limitation while relying again on a ground detection algorithm and a flat world assumption. We

*This work is supported by a CIFRE fellowship from Renault S.A.S

¹Renault S.A.S, 1 av. du Golf, 78288 Guyancourt, France. Contact: name.surname@renault.com

²Sorbonne Universités, Université de technologie de Compiègne, CNRS, HeuDiaSyc, Centre de recherche Royallieu, CS 60319, 60 203 Compiègne cedex, France. Contact: name.surname@hds.utc.fr

however observed that such strong geometrical assumptions lack of flexibility, and are not always satisfied in practice. Simple ground detection also often fails to properly capture the actual drivable area. A road detection step, alongside a more flexible model, are thus needed to generate evidential grids from LIDAR scans in a more robust fashion.

B. Road detection from LIDAR scans

State-of-the-art approaches for road detection in LIDAR scans rely on image processing techniques. Fernandes et al. [6] proposed to project LIDAR points into a 2D image plane, to upsample them, and to detect the road in this image plane via an histogram similarity measure. Caltagirone et al. [7] proposed to project LIDAR points into a 2D sparse feature grid corresponding to a bird's eye view, and to train a convolutional neural network to predict a dense road region from this sparse representation. Lyu et al. [8] proposed to train a neural network on range images generated from the spherical projection of LIDAR scans, and to fit a polygon representing a dense drivable area on the predicted road points. Although those approaches are currently the best performing LIDAR-only road-detection approaches on the KITTI dataset, they all aim at predicting a dense road area from a sparse LIDAR scan, and thus rely on upsampling. All those approaches then predict the presence of road on locations where no actual LIDAR measurements were actually available, which is an undesirable behavior for a LIDAR-only road detection algorithm. Indeed, gaps or small obstacles could be present but remain unobserved due to the sparsity of a LIDAR sensor, in areas where those algorithms would predict the presence of road. Moreover, due to the limitations of the KITTI dataset, in which the road is only labelled in a front camera view, those systems do not detect the road on complete LIDAR scans. Point-level road detection should be performed in complete LIDAR scans, so as to only represent information in areas that are actually observed.

C. PointNet: Machine Learning on raw point clouds

The recent PointNet architecture, introduced by Qi et al. [4], processes vectors of raw point-clouds, in which the point coordinates are directly stored. PointNet applies a multi-layer perceptron to each individual point, and produces a feature vector describing the whole point-cloud by applying a global max operator on the features extracted from each point. Although simple, this solution has proven to approach, or overpass, state-of-the-art performances on several perception tasks relying on point-clouds. It was extended in [9], by extracting local features in a point-cloud at several contextual scales, based on the metric distances between points. The resulting system outperforms the original PointNet architecture, at the cost of an increased complexity and inference time. However, PointNet architectures suffer from several drawbacks. First of all, they require a fixed number of input points. Secondly, PointNets usually expect normalized, relatively dense and constrained inputs. This makes the architecture improper when aiming to process large-scale LIDAR scans [10], and often requires to

split large point-clouds into individually processed voxels. Processing LIDAR points at the ring level could however leverage these limitations, as LIDAR rings are dense. Yet, a proper grid mapping framework relying on such a point-level classification is still to be defined. Especially, a proper way to represent the outputs of such a classifier into an evidential framework is still to be defined.

III. EVIDENTIAL REINTERPRETATION OF BINARY GLR CLASSIFIERS

T. Denoeux, in [5], proposed to reinterpret generalized logistic regression (GLR) classifiers as performing a fusion of evidential mass functions. With such a view, it is possible to construct evidential mass values, from the weights at the output of a neural network. Thanks to this technique, it becomes trivial to generate and accumulate evidential road detection results into an evidential 2D grid from a classifier, without relying on any explicit geometrical model. This is what we call *model-free evidential road grid mapping*.

Let a binary classification problem with $X = (x_1, \dots, x_d)$, a d -dimensional input vector, and $Y \in \Theta = \{\theta, -\theta\}$ a class variable. Let $p_1(x)$ be the probability that $Y = \theta$ according to the fact that $X = x$. Let w be the output of a binary logistic regression classifier, trained to solve the aforementioned classification problem ; $p_1(x)$ is such that:

$$p_1(x) = S(w) = S\left(\sum_{j=1}^d \beta_j \phi_j(x_c) + \beta_0\right) \quad (1)$$

with S being the sigmoid function, and the β values being usually learnt alongside those of the potentially non-linear ϕ_i mappings. In Eq. 1, w exactly corresponds to the output of a deep neural network trained as a binary GLR classifier, with x_c being its input. There exist α_j values such that:

$$\sum_{j=1}^d \alpha_j = \beta_0 \quad (2)$$

$$w = \sum_{j=1}^d w_j = \sum_{j=1}^d (\beta_j \phi_j(x_c) + \alpha_j) \quad (3)$$

Each w_j can then be seen as a piece of evidence towards θ or $-\theta$, depending on its sign. Let w_j^+ be the positive part of w_j , and let w_j^- be its negative part. Let $w^+ = \sum w_j^+$, $w^- = \sum w_j^-$. An evidential mass function m_{LR} can be generated as follows:

$$m_{LR} = \{\theta\}^{w^+} \oplus \{-\theta\}^{w^-} \quad (4)$$

This means that any binary GLR classifier can be seen as a fusion of simple mass functions, that can be derived from the parameters of the final linear layer of the classifier. However, the α_j values in Eq. 2 have to be estimated. Let $\alpha = (\alpha_1, \dots, \alpha_d)$. T. Denoeux proposed to select the α vector that maximize the sum of the $m_{LR}(\Theta)$ mass values over the training set, so as to get the most uncertain and cautious solution. This leads to the following minimization problem:

$$\min f(\alpha) = \sum_{i=1}^n \sum_{j=1}^d (\beta_j \phi_j(x_i) + \alpha_j)^2 \quad (5)$$

with $\{(x_i, y_i)\}_{i=1}^n$ being the training dataset.

An exact solution to this minimization problem exists [5], but it requires to perform an additional post-processing step after the training, and relies on the assumption that the parameters obtained after the training are reliable. When working with unperfect or coarse labels, an approximate solution is thus needed. We observed in [11] that an approximate solution to the minimization problem in Eq. 5 could be obtained directly during the training, by considering the α vector as the bias values of an Instance-Normalization layer present at the output of the network. Let $v(x_c) = (v_1(x_c), \dots, v_d(x_c))$ be the mapping modelled by all the consecutive layers of the classifier but the last one ; let \bar{v}_j be the mean value of the v_j function on the training set, and $\sigma(v_j)^2$ its corresponding variance. Then, if it is assumed that Instance-Normalization is used as the final layer of the network, Eq. 5 becomes:

$$\min f(\alpha) = \sum_{i=1}^n \sum_{j=1}^d \left(\beta_j \frac{v_j(x_c) - \bar{v}_j}{\sqrt{\sigma(v_j)^2 + \epsilon}} + \alpha_j \right)^2 \quad (6)$$

After development, the following expression is obtained:

$$\min f(\alpha) = n \sum_{j=1}^d \beta_j^2 + n \sum_{j=1}^d \alpha_j^2 \quad (7)$$

By simply applying L2-regularization on the linear parameters of the final layer, this expression will be minimized during the training. The network can then be trained to generate relevant evidential mass values, even when the network is optimized on coarse labels.

IV. TRANSFORMATION-ADVERSARIAL NETWORK FOR POINT-LEVEL ROAD DETECTION IN LIDAR RINGS

A. Ring-level PointNet

Typically, dense LIDAR sensors rely on stacked lasers that individually sweep the scene. A LIDAR ring represents a set of points that is obtained after the sweep of the environment by a single laser of a LIDAR. To detect the road in LIDAR scans, without having to transform the raw points into another representation, a classifier inspired by PointNet can be used. To leverage the limitations of PointNet that were exposed in Sec. II, the processing is done at the ring level. Indeed, the maximum number of points that a LIDAR ring can include can be computed from the angular resolution of the LIDAR. Then, contrary to what was done in [4] and [10], no sampling of the point-cloud is needed. Moreover, LIDAR rings are often dense, especially at short range, since each laser sweeps the whole scene, which would facilitate the reasoning of a PointNet-like network. And in the event of missing points, the input vector can typically be padded with an already present point, since the point-cloud wise max-pooling operation used in PointNet can filter duplicate point features. Finally, the maximum number of points in each sweep is relatively small, which means that the LIDAR rings will be easily processed in parallel.

However, LIDAR rings vary significantly among each others: a ring acquired by a top laser and a bottom laser will include points that were acquired at very different distances. A training scheme, inspired by the recent successes of generative-adversarial networks (GAN) in the image domain [12], was proposed to cope with this issue.

B. Transformation-adversarial network for LIDAR rings

GANs rely on the conjunction of two alternatively trained systems. The first one, called the generator, is optimized to generate artificial samples that are as realistic as possible. The second one, called the discriminator, is trained to discriminate real and artificial samples. The two systems are competing against each other: the generator aims at fooling the discriminator, and the discriminator aims at detecting samples generated by the generator. Similarly, we propose a Transformation-adversarial network, or TAdNet, composed of a Transformation network, and a Classification/Discrimination network. In the original PointNet, T-Nets predict affine transformation matrices applied to the whole input cloud, and to intermediate features extracted by point-level MLPs. Those T-Nets are optimized during the training, alongside the other parameters of the network.

The Transformation network that we propose, which also applies a transformation predicted by a T-Net to the input, is optimized separately from the rest of the system. To cope with the variability among LIDAR rings, the Transformation network also includes an H-Net. This H-Net, or homothety network, processes the transformed point-cloud obtained from the transformation predicted by the T-Net, and predicts an explicit rescaling factor, that is applied to the coordinates of all the points. The input points are represented by their Cartesian coordinates (x, y, z) , spherical coordinates (ρ, ϕ, θ) , and their intensity. To account for the risk of redundancy among the point features, the ϕ and θ angles are the uncorrected azimuth and zenith at which the point was acquired, while the Cartesian coordinates are obtained after correction. Let h be the scale predicted by the H-Net. Then, the coordinates of the input points are rescaled as follows: $x_* = hx$, $y_* = hy$, $z_* = hz$, $\rho_* = h\rho$. All the other features are left unchanged.

The Transformation network can then learn to remap all the LIDAR rings into a constrained range, that is suitable for the road classification task. We assumed that it should be difficult to predict the ring ID of properly remapped and constrained LIDAR rings. The Transformation network is thus trained alongside a Classification/Discrimination network, and aims at generating similar LIDAR rings. This Classification/Discrimination network is in fact a multi-task PointNet, without any initial T-Net. It has to both perform road detection among the LIDAR points, and predict the ID of the LIDAR ring that it processes. This ring ID is predicted from the output of a small Pointnet-like subnetwork that is fed with the vector of concatenated point-level features and cloud-level features, that can be obtained after the max-pooling operation that every PointNet-like network uses. Following the results in Eq. 7, Instance-Normalization is

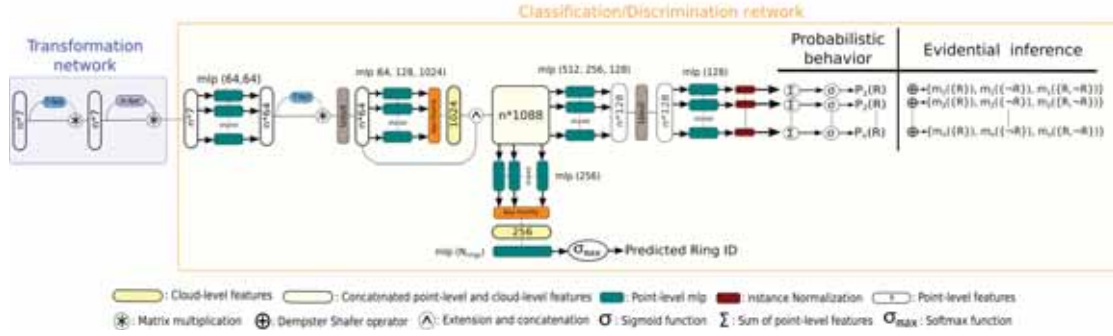


Fig. 2: Transformation-adversarial network for road-detection in LIDAR scans

used on the outputs used for road detection. The whole system is depicted in Fig. 2.

C. Training procedure

A PointNet-like system is typically trained with a multi-task loss. In the context of this study, the problem is point-level road detection in LIDAR rings. The loss chosen for this task, noted L_{ce} was the classical cross-entropy loss. The second component of the loss used for the training was a geometrical regularization loss. Let A be the transformation matrix predicted by the T-Net inside the Classification/Transformation network. This 64 by 64 matrix is more difficult to optimize than the simple transformation matrix predicted by the first T-Net, but should be as orthogonal as possible. Then, the loss on A to minimize, noted L_{geo} , is:

$$L_{geo}(A) = \|I - AA^T\|^2 \quad (8)$$

Finally, the ring ID prediction error is again evaluated from the cross-entropy loss, calculated from the actual ring ID. We note this loss L_{ring} . Let L_{Tr} , the loss used to optimize the Transformation network, and L_{CD} , the loss used to optimize the Classification/Discrimination network. For each ring, let P_{road} , Y_{road} , P_{Ring} and Y_{Ring} be, respectively, the point-wise predicted probability that each point belongs to the road, the corresponding road labels, the predicted ring ID and the corresponding ring label. Then:

$$\begin{aligned} L_{CD} &= \lambda_{road} L_{ce}(Y_{road}, P_{road}) \\ &+ \lambda_{ring} L_{ce}(Y_{Ring}, P_{Ring}) \\ &+ \lambda_{geo} L_{geo}(A) \\ L_{Tr} &= \lambda_{road} L_{ce}(Y_{road}, P_{road}) \\ &- \lambda_{ring} L_{ce}(Y_{Ring}, P_{Ring}) \\ &+ \lambda_{geo} L_{geo}(A) \end{aligned}$$

The whole system is trained thanks to the algorithm 1. To facilitate the training, UOut [13] was used. Originally, UOut was proposed because it was observed that Dropout shifts the mean and standard deviations of the features, which is not desirable when using Batch-Normalization, or Instance-Normalization. Uout, on the other hand, marginally affects those statistics. As Instance-Normalization is used on the output features of the network, due to the results of Eq. 7, UOut is a reasonable choice to regularize the model.

Algorithm 1 Training of the proposed system

```

Transformation network: T ;
Classification/Discrimination network: CD ;
N training rings are available ;
for e epochs do
  for N/n iterations do
    Sample n batches ( $b_0, \dots, b_t$ ) from the training set
    for i in range(n) do
       $b_i^* = T(b_i)$ 
      RoadClassif, RingID = CD( $b_i^*$ )
      Update CD from  $L_{CD}$ 
    end for
    for i in range(n) do
       $b_i^* = T(b_i)$ 
      RoadClassif, RingID = CD( $b_i^*$ )
      Update T from  $L_{Tr}$ 
    end for
  end for
end for

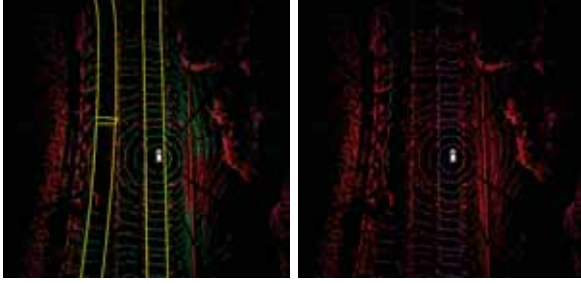
```

V. EXPERIMENTS AND EVALUATION OF THE CLASSIFICATION PERFORMANCES

A. Automatic labelling of a LIDAR dataset from a lane-level map

To properly evaluate the system, a dedicated LIDAR dataset was needed. No open-source LIDAR dataset including 360° point-level road labels was available when conducting this study. An autonomous perception platform equipped with a Velodyne VLP-32C running at 10 Hz was thus used to collect raw LIDAR scans in Guyancourt, France, in order to create a dataset with point-level road labels in LIDAR scans. Each LIDAR ring was composed of a maximum of 1800 points. The labelling was done automatically thanks to a pre-existing lane-level centimetric map, as shown in Fig 3. The data collection vehicle also included a Trimble BX940 inertial positioning system coupled with an RTK Satinfo modem, for localization.

A ground detection algorithm [14] was used to label obvious obstacles with a probability of being road-points equal to 0. The detected ground-points were projected into the map plane, for labelling. Following [15], the localization error was assumed to follow a zero-mean Gaussian model. Covariance matrices corresponding to the estimated position were provided by the localization system. The variance of



(a) Raw point-cloud, and the corresponding map available at the points are labelled as obstacles ; recording position. Green points the purpler a point is, the most belong to the pre-detected ground. likely of being a road point it is

Fig. 3: Automatic labelling procedure of a LIDAR point-cloud from a lane-level centimetric map.

the localization error was assumed to be the maximum variance on the easting/northing coordinates, noted σ_{xy}^2 . This pessimistic assumption facilitates the computations, and accounts for possibly undetected timing or calibration errors. Let a detected ground-point x_i , with d_i the distance between its projection on the map plane and the closest mapped road-edge. The labelled probability of x_i being a road point y_i can be computed from the cumulative distribution function of the normal distribution. If x_i was projected into a mapped road:

$$y_i = \int_{-\infty}^{d_i} \frac{1}{\sigma_{xy}\sqrt{2\pi}} \exp^{-\frac{1}{2}\left(\frac{x}{\sigma_{xy}}\right)^2} dx \quad (10)$$

Otherwise:

$$y_i = 1 - \int_{-\infty}^{d_i} \frac{1}{\sigma_{xy}\sqrt{2\pi}} \exp^{-\frac{1}{2}\left(\frac{x}{\sigma_{xy}}\right)^2} dx \quad (11)$$

To prevent the presence of redundant data, the labelling procedure was only launched every ten scans. It was also disabled when the vehicle was stopped. The final dataset was finally generated from 2334 labelled LIDAR scans acquired in Guyancourt, France. In practice, when d_i was larger than $10 * \sigma_{xy}$, y_i was set to either 0 (the point is not projected into a road) or 1 (the point is projected into a road). 0-1 labels represent more than 96,5% of the labels. A 70/30 split was used to create a training and a validation set from this data. To ensure that the train and validation dataset are significantly different, the scans were first ordered according to their recording date. Then, the validation set was created from the earliest and latest fifteen percents of the dataset. With such a dataset of automatically and softly labelled LIDAR scans, being able to generate evidential mass values while training on coarse labels, as allowed by the use of Instance Normalization and L2-regularization, is valuable.

B. Evaluation procedure and results

We report the classification results in Table I. Three systems were evaluated: the proposed Transformation-Adversarial Network (TAdNet), a ring-level PointNet, and a scan-level PointNet, to quantify the interest of the refinements introduced with TAdNet. The point-level MLPs were following

Model	All labels		0-1 labels	
	F1-score	Accuracy	F1-score	Accuracy
PointNet [4] - <i>ring</i>	0.868	0.973	0.907	0.983
PointNet [4] - <i>scan</i>	0.899	0.980	0.933	0.988
TAdNet - <i>ours</i>	0.933	0.987	0.959	0.993

TABLE I: Classification results for PointNet on LIDAR scans and tings, and for the proposed TAdNet, on the validation set

the original architecture proposed in [4], with a ReLU activation function and systematic use of Batch Normalisation. The three systems were implemented in PyTorch. The two PointNets consisted in exactly the same layers as TAdNet, except for the H-Net and the ring-ID prediction subnetwork that were removed. Instance Normalization and UOut were still used, as the resulting systems were all intended to be used for model-free evidential road-grid mapping. The Adam optimizer was used for the three networks, with a learning rate of 0.0001. Following the recommendations from the original authors of Uout, the random numbers generated by the Uout layers were sampled from a $[-0.1, 0.1]$ range. Empirical observations showed that, instead of only applying L2-regularization to the final layer of the networks, applying it to all the parameters led to better numerical stability. Then, a weight-decay of 0.0001 was applied to all the parameters of the three networks, except for the parameters of the Transformation-network in TAdNet, on which a weight-decay of 0.00001 was applied. All the T-Nets and the H-Net were initialized with identity transformations. Following [4], all the parameters of the multi-task losses were set to 1 for the regular PointNets, and for TAdNet, λ_{ring} was set to 0.8, λ_{road} was set to 1.2 and λ_{geo} was set to 1. TAdNet and the ring-level PointNet were trained on mini-batches including 64 rings, and the scan-level PointNet was trained on mini-batches of 2 scans, as each scan was composed of 32 rings. We report F1-scores and accuracies on the full validation dataset, and on only the 96.5% of 0-1 labels. In the case of non-binary labels, a point was considered to be labelled as a road-point if its labelled probability was higher than 0.5. And a point was considered to be classified as road if the predicted probability was higher than 0.5. Table I reports the respective results of those approaches in the validation set. All approaches have satisfactory results, even if TAdNet outperforms all the approaches in all the indicators. The interest of the rescaling performed by TAdNet is obvious, as the ring-level PointNet is by far the worst performing approach, while TAdNet outperforms the scan-level PointNet, even though it only processes rings.

VI. MODEL-FREE EVIDENTIAL ROAD GRID MAPPING FROM THE CLASSIFICATION RESULTS

Evidential road grids can easily be generated from TAdNet, and the expression in Equation 4. For each point, three evidential mass values can be extracted: $m(\{R\})$, for the road class ; $m(\{-R\})$, for the obstacle class ; and $m(\{R, -R\})$, for the unknown class. Then, a grid can be obtained by projecting all the LIDAR points into the xy-plane. The

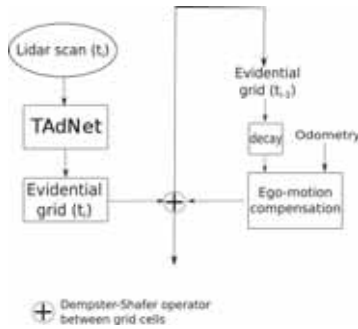


Fig. 4: Simple model-free evidential road grid mapping algorithm

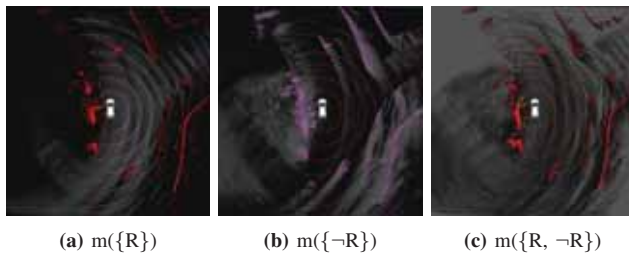


Fig. 5: Model-free evidential road grid mapping. The accumulated evidential grid is overlaid with the point-level evidential mass values generated from the LIDAR sensor

Dempster-Shafer operator can then be used to fuse the mass values of all the points that are projected into a given cell. Finally, the evidential grids can be fused over time thanks to the algorithm in Fig 4, which follows the approach in [1] but applies it to evidential mass values generated from TAdNet, instead of using a geometrical model. Figure 5 presents an example of model-free evidential road grid map generated from this algorithm, and TAdNet. A $(45\text{m} \times 45\text{m})$ area around the vehicle was covered by a road grid having a cell size of $(0.1\text{m} \times 0.1\text{m})$. A decay rate of 0.98 was used, and the odometry was coming from the IMU present in the localization system previously used for the collection of the labelled LIDAR scans. Only the 20 lowest LIDAR rings were used. A video of a grid accumulation in a roundabout is available¹.

VII. CONCLUSION

We presented TAdNet, a Transformation-adversarial network inspired by PointNet that performs road detection in LIDAR rings. The classification results can be used to generate evidential road grid maps without needing an explicit geometrical model, as showed by some experiments done on real-life data, and a TAdNet trained on coarse LIDAR labels obtained from a map. The next step will consist in evaluating other approaches for model-free evidential road grid mapping, in a more reliable fashion. To do so, a dataset of 368 LIDAR scans was already finely labelled by hand, and will be used for validation purposes in the future.

ACKNOWLEDGMENT

This work was realized within the SIVALab joint laboratory between Renault S.A.S, the CNRS and HeuDiaSyc.

REFERENCES

- [1] C. Yu, V. Cherfaoui, and P. Bonnifait, "An evidential sensor model for velodyne scan grids," in *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*, IEEE, 2014, pp. 583–588.
- [2] E. Capellier, F. Davoine, V. Frémont, J. Ibañez-Guzman, and Y. Li, "Evidential grid mapping, from asynchronous LIDAR scans and RGB images, for autonomous driving," in *21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 2595–2602.
- [3] S. Wirges, C. Stiller, and F. Hartenbach, "Evidential occupancy grid map augmentation using deep learning," in *IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 668–673.
- [4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR)*, IEEE, vol. 1, no. 2, p. 4, 2017.
- [5] T. Denoeux, "Logistic regression, neural networks and dempster-shafer theory: A new perspective," *ArXiv preprint arXiv:1807.01846*, 2018.
- [6] R. Fernandes, C. Premebida, P. Peixoto, D. Wolf, and U. Nunes, "Road detection using high resolution lidar," in *2014 IEEE Vehicle Power and Propulsion Conference (VPPC)*, IEEE, 2014, pp. 1–6.
- [7] L. Caltagirone, S. Scheidegger, L. Svensson, and M. Wahde, "Fast lidar-based road detection using fully convolutional neural networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2017, pp. 1019–1024.
- [8] Y. Lyu, L. Bai, and X. Huang, "Chipnet: Real-time lidar processing for drivable region segmentation on an fpga," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2018.
- [9] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108.
- [10] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3D semantic segmentation of point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 716–724.
- [11] E. Capellier, F. Davoine, V. Cherfaoui, and Y. Li, "Evidential deep-learning for arbitrary lidar-object classification in the context of autonomous driving," *Intelligent Vehicles Symposium (IV)*, 2019.
- [12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [13] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," *ArXiv preprint arXiv:1801.05134*, 2018.
- [14] P. Chu, S. Cho, S. Sim, K. Kwak, and K. Cho, "A fast ground segmentation method for 3D point cloud," *Journal of information processing systems*, vol. 13, no. 3, pp. 491–499, 2017.
- [15] K. A. B. Ahmad, M. Sahnoudi, and C. Macabiau, "Characterization of GNSS receiver position errors for user integrity monitoring in urban environments," in *ENC-GNSS 2014, European Navigation Conference*, 2014.

¹<https://drive.google.com/file/d/1R7WuZaIvUqPHVRbplDLglea5b46zugE5/view?usp=sharing>

End-to-End Deep Neural Network Design for Short-term Path Planning

Minh Quan Dao¹, Davide Lanza¹ and Vincent Frémont¹

Abstract—Early attempts on imitating human driving behavior with deep learning have been implemented in an reactive navigation scheme which is to directly map the sensory measurement to control signal. Although this approach has successfully delivered the first half of the driving task - predicting steering angles, learning vehicle speed in an end-to-end setting requires significantly large and complex networks as well as the accompanying dataset. Motivated by the rich literature in trajectory planning which timestamps a geometrical path under some dynamic constraints to provide the corresponding velocity profile, we propose an end-to-end architecture for generating a non-parametric path given an image of the environment in front of a vehicle. The level of accuracy of the resulting path is 70%. The first and foremost benefit of our approach is the ability of incorporating deep learning into the navigation pipeline. This is desirable because the neural network can ease the hardness of developing the see-think-act scheme, while the trajectory planning at the end adds a level of safety to the final output by ensuring it obeys static and dynamic constraint.

I. INTRODUCTION

Motion planning methods for autonomous vehicles are classically developed to sequentially perform path planning, obstacles avoidance, and trajectory optimization [1], [2], [3]. Path planning module, which can be implemented by RRT [4], A* [5], Lattices and motion primitives [6], or function optimization [7], takes into account the geometry characteristic of the environment to produce a collision-free (with regard to static obstacle) non-parametric path. On the other hand, the trajectory optimization module, which subsumes the obstacles avoidance, aims to optimize both way points and the corresponding velocity profile so that they respect vehicle’s kinematics and dynamic obstacles [8], [7].

Deep learning can help simplify the path planning and trajectory optimization pipeline above with just a deep neural network which learns human driving behavior in a supervised manner. The first successful example traces back to ALVINN [9], where a shallow network was used to calculate the steering angle directly from images. This work is revised in the deep learning era in [10], the authors went beyond a mere pattern recognition, learning the entire steering angle prediction pipeline for autonomous cars by building a mapping from images obtained by a forward camera to steering angle with a deep CNN. Taking the similar approach but with different input, [11] used visual information obtained by an event-based camera to train their steering model. A more global approach to motion planning using deep learning is to

learn a spatial traversability maps [12], [13]. In these works, rather than just predict a single steering angle, the model is designed to learn a cost function which can be later used for path planning. The common shortage of these works is their inability of addressing vehicle speed.

To solve this problem by the end-to-end learning, [14] uses recurrent layers together with CNN network to learn a complete driving policy (steering angle and vehicle speed). Though this work has showed its capability and robustness, it comes with the cost of an extremely large dataset and fairly complex network as well as the training process. Taking a different approach, [15] proposed an integrated solution where a CNN is trained on monocular image data (as in [10]) to output a path, this path is then used as the initial guess for a Particle Swarm Optimization algorithm [16] which in turn transforms the path to a complete trajectory.

Another method to infer vehicle speed is to calculate it proportional to the collision probability [17]. The network designed in this work is made of three consecutive ResNet blocks following by two parallel fully-connected layers, which respectively output the steering angle and a collision probability. The steering prediction is learned through a regression problem, while the collision prediction is addressed as a binary classification problem.

In this paper, we propose and evaluate a deep neural network architecture inspired by DroNet for short-term path planning which is to predict a sequence of steering angles directly from an image obtained by forward camera, hence an end-to-end model. The main difference between [17] and our work is the statement of steering angle prediction problem. In fact, as mentioned in [18], it can be transformed from a regression problem of continuous values to a classification problem where the steering angle range is tessellated into discrete spans with width of 0.01 radians. Such choice of span width is justified by the jitter of steering angle applied by a human driver in straight road. Moreover, the calculated steering sequence is mapped into a non-parameterized path and, once the path is output, any motion planning algorithm can be implemented to timestamp the path.

The rest of this paper is organized as follows. Sec.II adapts the DroNet architecture for learning a single steering angle through a classification problem. In Sec.III, we further modify the resulted architecture and the used dataset to enable the network to learn a geometrical path in an end-to-end setting. The performance of the path planning model derived in the Sec.III is evaluated in Sec.IV. Then Sec.V describes the short-term path generation using the steering angles sequence and car-like vehicles motion constraints.

¹Authors are with Centrale Nantes, LS2N - UMR 6004 Nantes, France
email: vincent.fremont@ec-nantes.fr

Concluding remarks are made in Sec. in VI.

II. LEARNING STEERING ANGLES

Taking a different approach compared to the majority of researches in end-to-end learning for autonomous driving which formulates the prediction of steering angles as a regression problem [9], [10], [17], our model is designed to be a classifier. The reason of our choice stems from the fact that for a regression-based network, there is a continuous range of angles to infer, but only a finite number of samples with which to train against. By tessellating the range of steering angles into discrete bins, the requirement of infinite training samples to fully cover such continuous range is no longer effective. Moreover, Sec.IV shows that with sufficiently small bins, our classifier can outperform the DroNet - a regression-based model.

A. Network Architecture

[17] has showed that their the architecture responds strongly to "line-like" features in the forward images which has a strong relation with the resulted steering angles. Motivate by this work, we design our model out of their ResNet-made body and put a classifier on top of it. This classifier is made of 2 dense layers. The first has 800 neurons activated by ReLU, while the second has 227 neurons (equal to the number of classes) and is activated by Softmax function. The conceptual architecture is shown in Fig.1. Each ResNet block in this figure is comprised of 3 convolutional layers: 2 on the main path and 1 on the shortcut (see Fig. 2).

The hyperparameters of each ResNet block are shown in Tab. I.

Stage	Layer	Number of kernels	Kernel size	Stride Stride	Padding Padding
1	Conv2D.a	32	3	2	same
1	Conv2D.b	32	3	1	same
1	Conv2D.c	32	1	2	same
2	Conv2D.a	64	3	2	same
2	Conv2D.b	64	3	1	same
2	Conv2D.c	64	1	2	same
3	Conv2D.a	128	3	2	same
3	Conv2D.b	128	3	1	same
3	Conv2D.c	128	1	2	same

TABLE I
MODIFIED RESNET CNN BODY PARAMETERS

The intuition behind our model is that the ResNet-made body should learn better how to output useful feature maps which probably contains roads shape and drivable area, while the classifier on its top should learn how to output the steering angle given the provided feature map.

B. Data Preparation

The dataset used to train our model is Udacity dataset challenge 2¹. This dataset contains several hours of driving on suburban road in good weather and lightning condition.

¹<https://github.com/udacity/self-driving-car/tree/master/datasets/CH2>

After processing, the dataset is organized as a time-order list. An element of this list contains an image captured by front-facing camera, the associated steering angle, GPS coordinate of the vehicle at this time step and other information.

To decide the class of any steering angle, a histogram of all steering angles in the dataset ranging from -2.051 to 1.903 radian is built (see Fig.4). The width of every bin of this histogram is 1 degree. The class of an angle is the index of the bin it belongs to. As can be seen in Fig.4, the distribution of collected steering angles among these classes is imbalance. To ensure the network does not overlook the less frequent classes during the training process, class i is assigned a weight $w(i)$ based on the median frequency balancing method in [19].

$$w(i) = \frac{\text{median frequency of the dataset}}{\text{frequency of this class}} \quad (1)$$

Here, the numerator is the ratio between the number of samples in class i and the total number of samples in the dataset. The denominator is the median of the all frequencies. The resulted weight $w(i)$ is later used to scale the contribution of every sample in class i to the total loss function.

C. Model Training

The model's weights are initialized randomly and the network is trained by minimizing the cross entropy loss function using Adam optimizer with default parameters. After training for 100 epochs with 1200 batch size, our model's accuracy on validation set peaks at 66% before dropping due to over fitting. With this level of accuracy, its qualitative performance measured by Root Mean Square Error (RMSE) and Explained Variance score (EVA) are respectively 0.1083 and 0.8338. These values are competitive, compared to recent development in end-to-end learning model: DroNet [17] and [11]. The details comparison is carried out in Sec.IV.

III. LEARNING A GEOMETRICAL PATH

As shown in the previous section, a single steering angle can be learned through a classification problem. Nevertheless, knowing the steering angle is just half of the autonomous driving task. The other control signal to be provided, is the vehicle's speed. There are no means of inferring a vehicle speed given a single steering angle at the same time instance. However, the rich trajectory planning literature suggests that a geometrical path can be timestamped to generate a velocity profile [20] such that it can satisfy some dynamic constraint. Therefore, in this section, we modify the resulted architecture from the previous section and the used dataset to enable the network to learn a geometrical path.

A. Network Architecture

Inspired by [15], a path can be encoded as a sequence of steering angles, each of which is applied to a predefined traveling distance. Based on this insight, a network can learn a path by learning a sequence of steering angles. This leads to the replacement of a single classifier on top of the

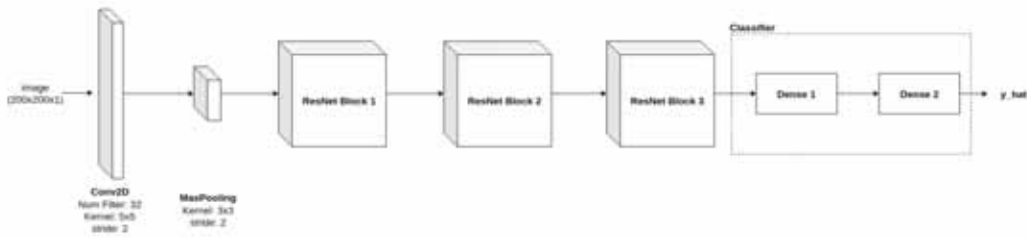


Fig. 1. Modified DroNet architecture [17]

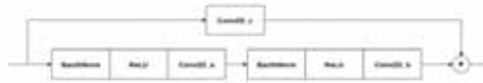


Fig. 2. Components of a ResNet block.

architecture in Sec.II by an array of 5 classifiers having the same number of neurons and activation function. Putting the ResNet-made body and the array of classifiers together, the complete architecture is shown in Fig. 3. Here, a block **Head i** is a 2-dense-layer classifier.

B. Dataset Preparation

The new model's output is interpreted as a sequence of 5 steering angles. Each angle is applied to 2 meters of traveling distance. As a result, a training sample is prepared as following:

- X : an image of the environment in front of the vehicle
- y : a list of one-hot vectors. This first vector denotes the class of the steering angle associates with the frame represented by X . The i -th vector represents the steering angle of the frame $i \times 2$ meters away from X .

Such definition of y suggests that the data set used to train path planning model needs to explicitly contain the distance information between two adjacent labels. This distance can be retrieved from the GPS coordinate of each frame provided in the original Udacity dataset. Upon completely being generated, the whole dataset is divided into training set and validation set with respectively 19000 and 2350 samples.

C. Model Training

Since the ResNet-made body is the same the model which predicts a single steering angle in Sec.II, the weight of the ResNet-made body of the path planning model in this section is trained from the best weight obtained in Sec.II. Conversely, all classifiers' weight are initialized randomly. The loss function is chosen to be the cross-entropy and is minimized by the Adam optimizer. The choice of hyper-parameters is the same as in Sec. II.

After training for 50 epochs, each of 5 classifiers in our model achieves at least 70% of accuracy.

IV. MODEL PERFORMANCE

A. Quantitative Performance

Using the same approach of [17], [11], the quantitative performance is measured by Root Mean Square Error (RMSE) and the Explained Variance score (EVA). The performance over these metrics of 5 classifiers in our path planning model compared to a constant estimator, which always predicts 0 as steering angle, a random one, the DroNet [17], and the model taking input from an event-based camera [11] is shown in Tab. II. All classifiers in our model outperform the DroNet in both RMSE and EVA, while fall behind event-based model with a small margin in RMSE. This shows by tessellating the range of steering angle into sufficiently small intervals, a classification-based model can deliver a better result, compared to a regression-based one.

Model	RMSE	EVA
Constant baseline	0.2129	0
Random baseline	0.3 ± 0.001	-1.0 ± 0.022
DroNet	0.1090	0.7370
Event-based Model	0.0716	0.8260
Head.0	0.0869	0.8933
Head.1	0.0920	0.8781
Head.2	0.1052	0.8382
Head.3	0.0820	0.9012
Head.4	0.0851	0.8943

TABLE II

PATH PLANNING MODEL QUANTITATIVE PERFORMANCE

B. Qualitative Performance

The comparison between the histograms of predicted angle classes and their ground truth on validation set are shown in Fig.4. This figure indicates a relative match between the predicted distribution and the true distribution.

In addition, the normalized confusion matrix of the first classifier is shown in Fig.5. This matrix features a clear, large magnitude main diagonal. This means the majority of predicted angle classes is actually the true class. Nevertheless, there are a few strong cells in the bottom of Fig.5 implying

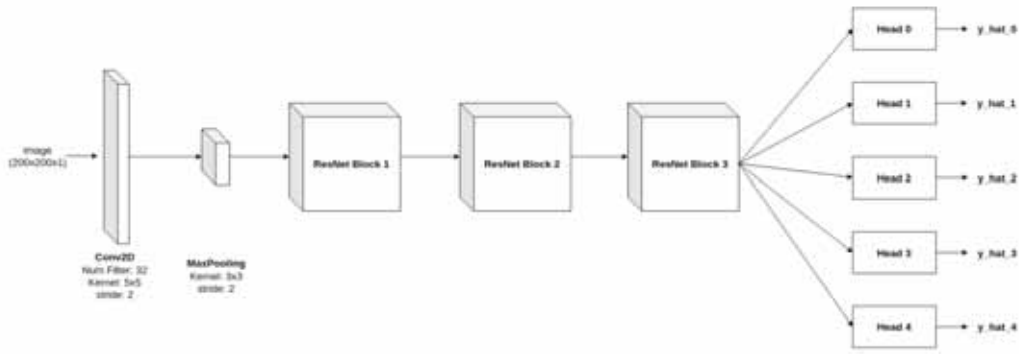


Fig. 3. Path planning architecture (CNN part)

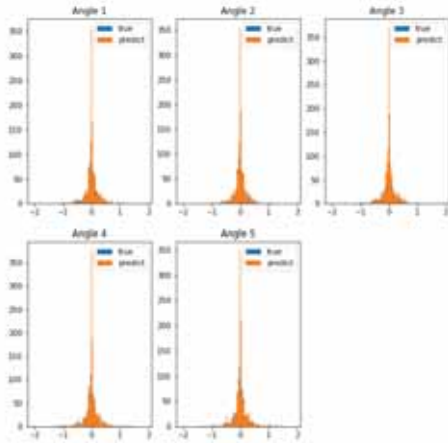


Fig. 4. Predicted angle classes distribution of each classifier compared to their ground truth

that the classifier fails to predict the class of extreme right angles.

C. Layer Activation Visualization

In an attempt to understand how our model produces its prediction, the outputs of each ResNet block in the path planning model are displayed in Fig.6. This figure shows that the first block recognizes lane mark and vehicles, while the second block segments the drivable area. The last block learns a down-sampled mapping. Together, these three ResNet blocks learn useful feature maps which contains roads shape and drivable area, while the classifiers on the top learn to calculate the steering angle given those feature maps.

V. INTERPRETING A STEERING SEQUENCE AS A PATH

Since the motion of car-like vehicles is constrained to be circular around its Instantaneous Center of Rotation (ICR)

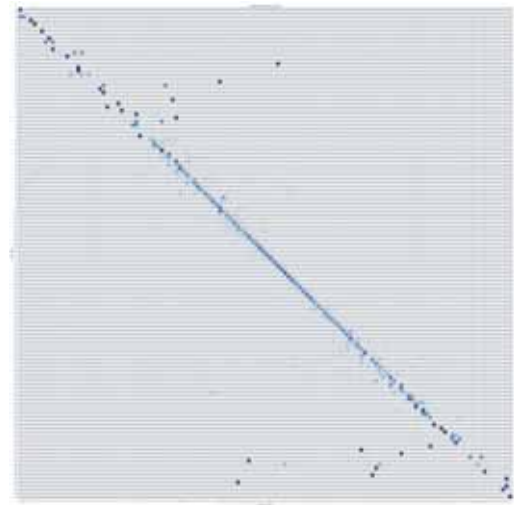


Fig. 5. Normalized confusion matrix of the first classifier.

(see Fig. 7), a sequence of steering angles can be interpreted as a geometrical path (i.e. sequence of way points) by applying each angle in the sequence to a predefined traveling distance of s meters.

In Fig.7, L is the distance between the front and rear axle. $\delta_i (i = 0 \dots N)$ is a steering angle ($N + 1$ is the length of steering angles sequence). At each time instance, the pose of the vehicle is represented by the pose of the local frame attached to the center of its rear axle - $O_i x_i y_i z_i (i = 0 \dots N)$. x_i goes from the rear axle to the front axle, and is perpendicular to these axles. z_i is orthogonal to the plane of motion, and pointing outward. y_i is defined such that $O_i x_i y_i z_i$ is right-handed. The target is to calculate the position of the center of the front axle relative to the local body frame at the presence - $O_0 x_0 y_0 z_0$. Assuming that the vehicle's motion is planar, the transformation from frame $O_i x_i y_i z_i$ to frame $O_{i+1} x_{i+1} y_{i+1} z_{i+1}$ is described by:

$${}^i T_{i+1} = \begin{bmatrix} Rot_{z, \phi_i} & {}^i t_{i+1} \\ 0_{1 \times 2} & 1 \end{bmatrix} \quad (2)$$

Here, ${}^i t_{i+1}$ is the coordinate of O_{i+1} in frame i , and

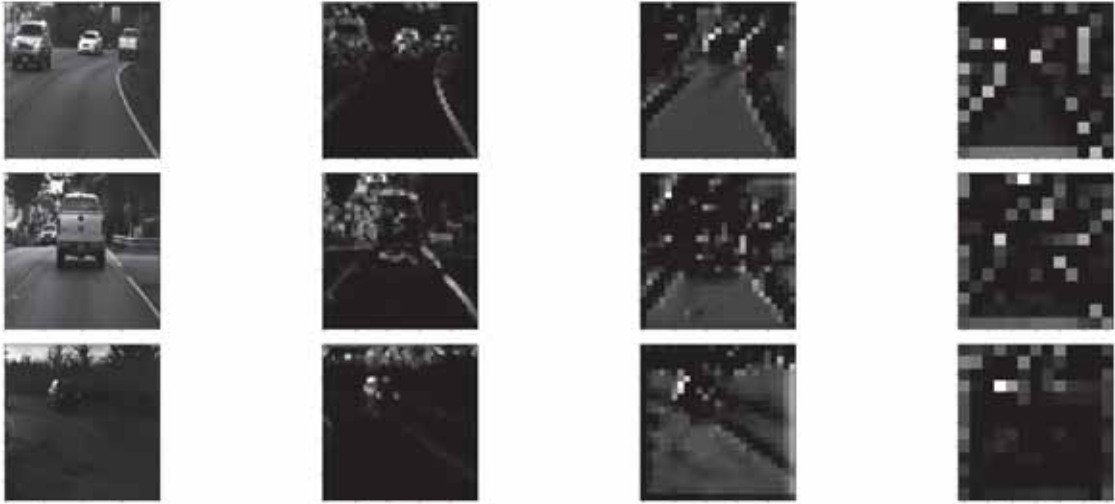


Fig. 6. Output of each ResNet block with respect to different input images. From left to right, the images are respectively the input image, output of the first, second, and third block.

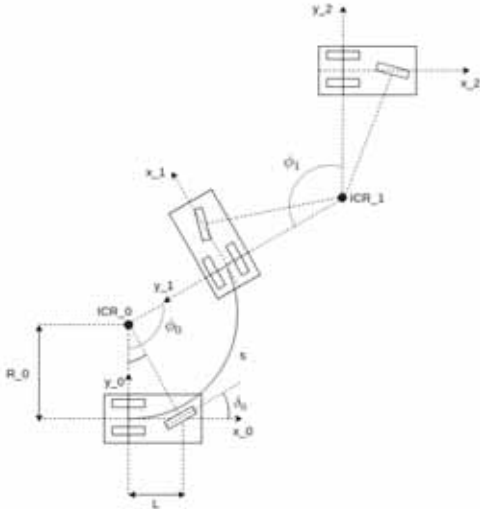


Fig. 7. Car-like vehicle's motion diagram with two different value of steering angles. Each of these angles is applied to an arc distance of s meters.

Rot_{z, ϕ_i} is the matrix represents the rotation around the z -axis by an angle ϕ_i . The radius of the circular motion around ICR_i is calculated as

$$R_i = \frac{L}{\tan \delta_i} \quad (3)$$

Eq.3 implies that when the steering angle is close to zero (i.e. the steering wheel is kept at the neutral position), the radius of motion approach infinity, hence a straight motion. Given R_i , the coordinate of O_{i+1} in frame i is

$${}^i t_{i+1} = 2R_i \sin\left(\frac{\phi_i}{2}\right) \begin{bmatrix} \cos\left(\frac{\phi_i}{2}\right) \\ \sin\left(\frac{\phi_i}{2}\right) \end{bmatrix} \quad (4)$$

ϕ_i is the angle between x_i and x_{i+1} . As shown in Fig.7, this angle can be calculated by:

$$\phi_i = \frac{s}{R_i} = \frac{s \tan \delta_i}{L} \quad (5)$$

With Eq. 5 and Eq. 4. The transformation from frame i to frame $i+1$ in Eq. 2 is now fully defined. The local position of the center of the front axle in homogeneous form is:

$${}^i L_i = [L, 0, 1]^T \quad (6)$$

This position is transformed into the local frame at the present time by the following equation

$${}^0 L_i = {}^0 T_i {}^i L_i = \prod_{j=1}^i {}^{j-1} T_j {}^j L_i \quad (7)$$

To test the quality of the predicted path and its ground truth, the trained path planning model is used to infer path from forward images taken from Udacity dataset. The inference process is implemented on a laptop equipped with an NVIDIA GeForce MX130, an Intel Core i7-8650U (1.90GHz), and 16GB of RAM. The inference time for 1 sample (i.e. 1 image) is 1 millisecond. Examples of path generated by both true and predicted sequence of steering are shown in Fig.8, while the extended video of path planning model's output compared to ground truth can be found in this link: <https://youtu.be/X2fi2xVr2jE>. Fig.8 as well as the video shows a good match between the predicted path and its ground truth, which in turn proves the quality of the prediction of our model.

VI. CONCLUSIONS

In this paper, we explored an end-to-end learning approach to path planning for autonomous vehicles. In details, a neural network made of three ResNet blocks and an array of

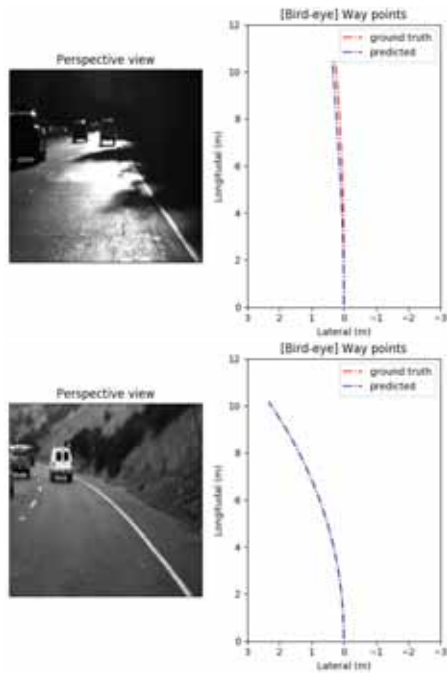


Fig. 8. Display of the predicted path. Left: Images captured by front-facing camera. Right: Path in cartesian coordinates

classifiers is trained to output a sequence of steering angles which is later interpreted into a geometrical path.

The advantage of this approach is that it paves the way for the integration of deep neural network into the motion planning framework. Specifically, the geometrical path learned by the network is then timestamped using trajectory optimization technique to finally produce a parameterized path (a path with a velocity profile). Besides providing two crucial control signals of the driving task (steering angle and velocity), this integrated approach enhances the reliability of the predicted trajectory while ensuring it is natural and consistent with vehicles' kinematic.

For the future work, the accuracy of the array of classifiers needs to be improved. Furthermore, the network generalization should be evaluated on other autonomous vehicles datasets with different camera parameters. Since a sequence of steering angles implies a time order, it might be helpful if the network can learn a temporal relation among the steering angles. This can be done by exploring the application of recurrent layers such as LSTM cells to enable the model learning such time relation.

The code used in this paper is hosted on GitHub in Minh-Quan Dao's ECN-E2E repository: <https://github.com/quan-dao/ECN-E2E>.

ACKNOWLEDGMENT

This research has been conducted as part of the HI-ANIC (Human Inspired Autonomous Navigation In Crowds) project, funded by the French Ministry of Education and Research and the French National Research Agency (ANR-17-CE22-0010).

REFERENCES

- [1] C. Katrakazas, M. A. Quddus, W. Hua Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," 2015.
- [2] T. Gu, J. Snider, J. M. Dolan, and J. Lee, "Focused trajectory planning for autonomous on-road driving," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, June 2013, pp. 547–552.
- [3] Wenda Xu, Junqing Wei, J. M. Dolan, Huijing Zhao, and Hongbin Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 2061–2067.
- [4] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using rrt," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 1681–1686.
- [5] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al., "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.
- [6] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 1879–1884.
- [7] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha: a local, continuous method," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, June 2014, pp. 450–457.
- [8] C. Liu, W. Zhan, and M. Tomizuka, "Speed profile planning in dynamic environments via temporal optimization," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 154–159.
- [9] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.
- [10] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: <http://arxiv.org/abs/1604.07316>
- [11] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5419–5427, 2018.
- [12] M. Wulfmeier, D. Z. Wang, and I. Posner, "Watch this: Scalable cost-function learning for path planning in urban environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2089–2095.
- [13] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *I. J. Robotics Res.*, vol. 36, pp. 1073–1087, 2017.
- [14] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3530–3538, 2016.
- [15] C. Hubschneider, A. Bauer, J. Doll, M. Weber, S. Klemm, F. Kuhnt, and J. M. Zillner, "Integrating end-to-end learned steering into probabilistic autonomous driving," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 1–7.
- [16] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5, Oct 1997, pp. 4104–4108 vol.5.
- [17] A. Loquercio, A. Maqueda, C. del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.
- [18] P. Penkov and V. S. J. Ye, "Applying techniques in supervised deep learning to steering angle prediction in autonomous vehicles," 2016.
- [19] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," *eprint arXiv:1411.4734*, 2014.
- [20] C. Katrakazas, M. Quddus, W. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416 – 442, 2015. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0968090X15003447>



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session II

Perception & Situation awareness

- **Keynote speaker: Ruigang Yang (Baidu, China)**
Title: Sim to Real: Using Simulation for 3D Perception and Navigation
- **Title: Feature Generator Layer for Semantic Segmentation Under Different Weather Conditions for Autonomous Vehicles**
Authors: O. Erkent, C. Laugier
- **Title: An Edge-Cloud Computing Model for Autonomous Vehicles**
Authors: Y. Sasaki, T. Sato, H. Chishiro, T. Ishigooka, S. Otsuka, K. Yoshimura, S. Kato



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session II

Keynote speaker: **Ruigang Yang**
(Baidu, China)

Sim to Real: Using Simulation for 3D Perception and Navigation

Abstract: The importance for simulations, in both robotics and more recently autonomous driving, has been more and more recognized. In this talk, I will talk the fairly extensive line of simulation research at Baidu's Robotics and Autonomous Driving Lab (RAL), from low-level sensor simulation, such as LIDAR, to high-level behavior simulation, such as drivers/pedestrians. These different simulations tools are designed to either produce an abundant amount of annotated data to train deep neural network, or directly provide an end-to-end environment to test all aspects of robots/autonomous vehicles movement capabilities.

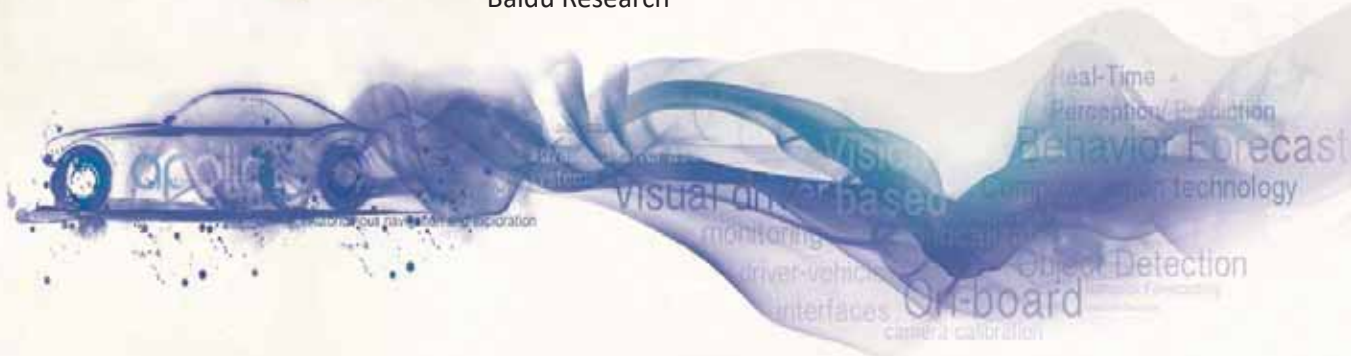
Biography: Ruigang Yang is currently Chief Scientist for 3D Vision at Baidu Research. He leads the Robotics and Autonomous Driving Lab (RAL). Before joining Baidu, he was a full professor of Computer Science at the University of Kentucky. He obtained his PhD degree from University of North Carolina at Chapel Hill and his MS degree from Columbia University. His research interests span over computer graphics and computer vision, in particular in 3D reconstruction and 3D data analysis. He has published over 100 papers, which, according to Google Scholar, has received over 10000 citations with an h-index of 50 (as of 2018). He has received a number of awards, including US NSF Career award in 2004, best demonstration award in CVPR 2006, and University of Kentucky's Dean's Research Award in 2013. He is currently an associate editor of IEEE TPAMI and a senior member of IEEE.



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Sim-to-Real: Using Simulation for 3D Perception and Navigation

Ruigang Yang
Baidu Research



Quick Facts about Baidu

- # 1 Search Engine in China: www.baidu.com
- Founded in 2000
- # of Employees: 42000
- Alexa Ranking: 4

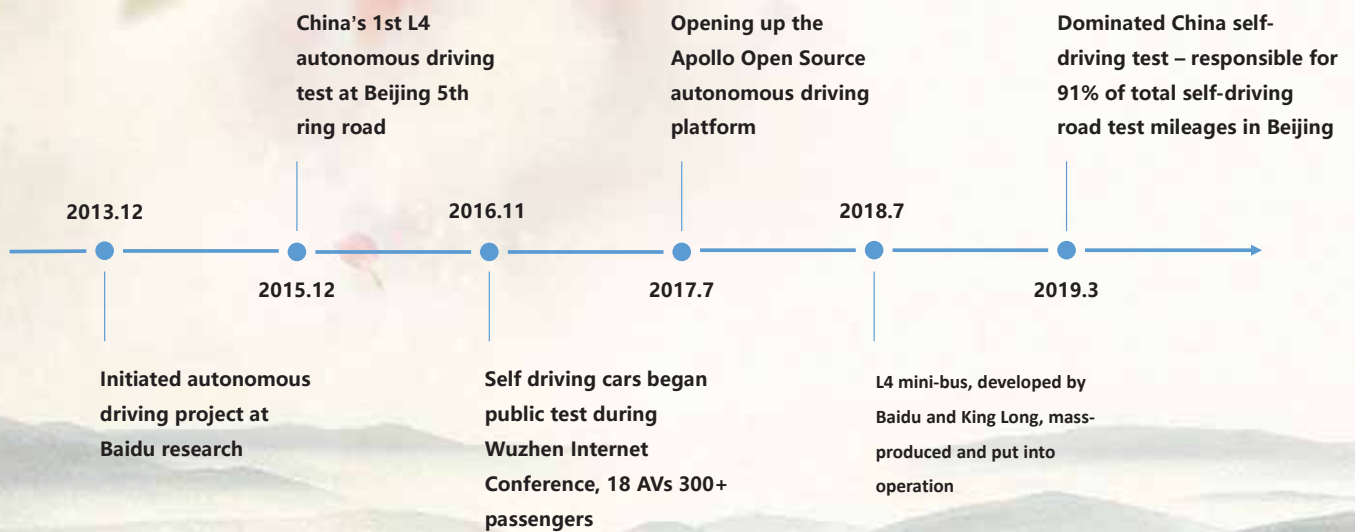


Apollo Project

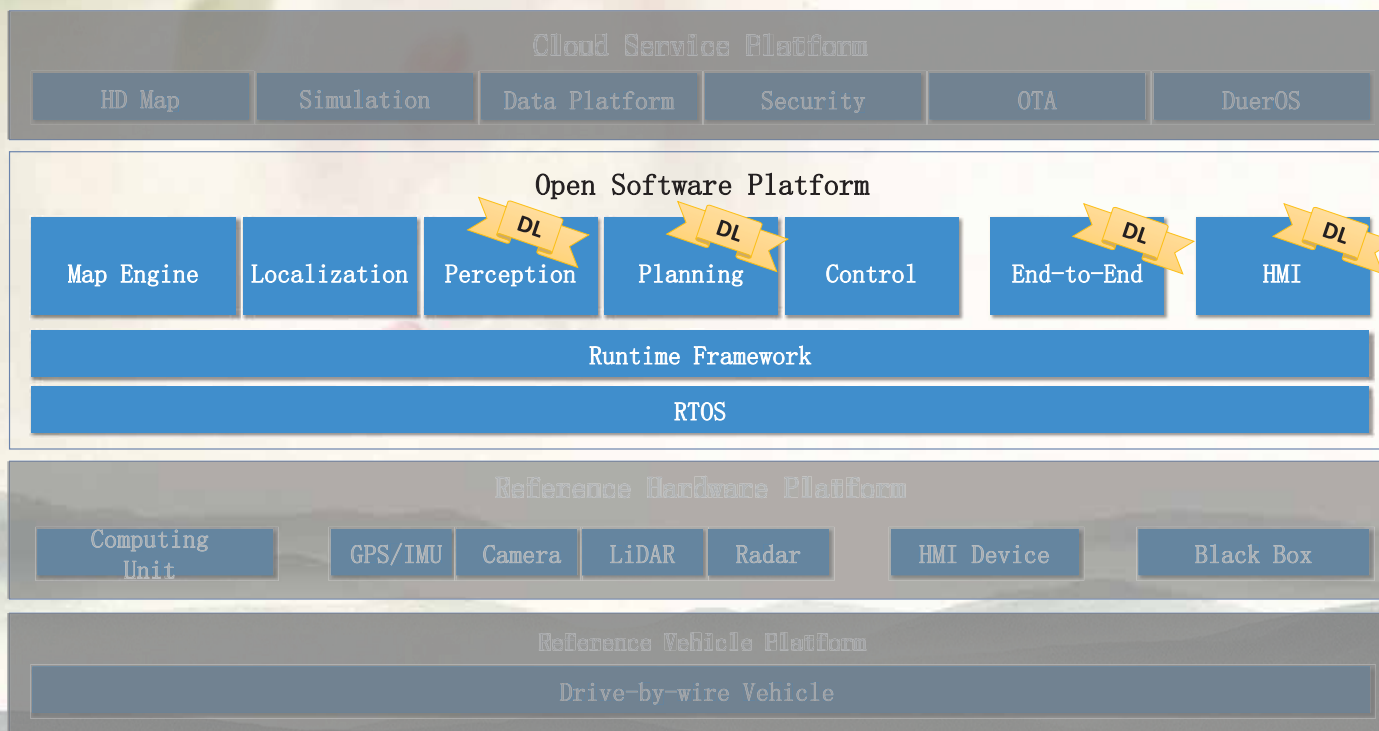
Open-Source Platform for Autonomous Driving



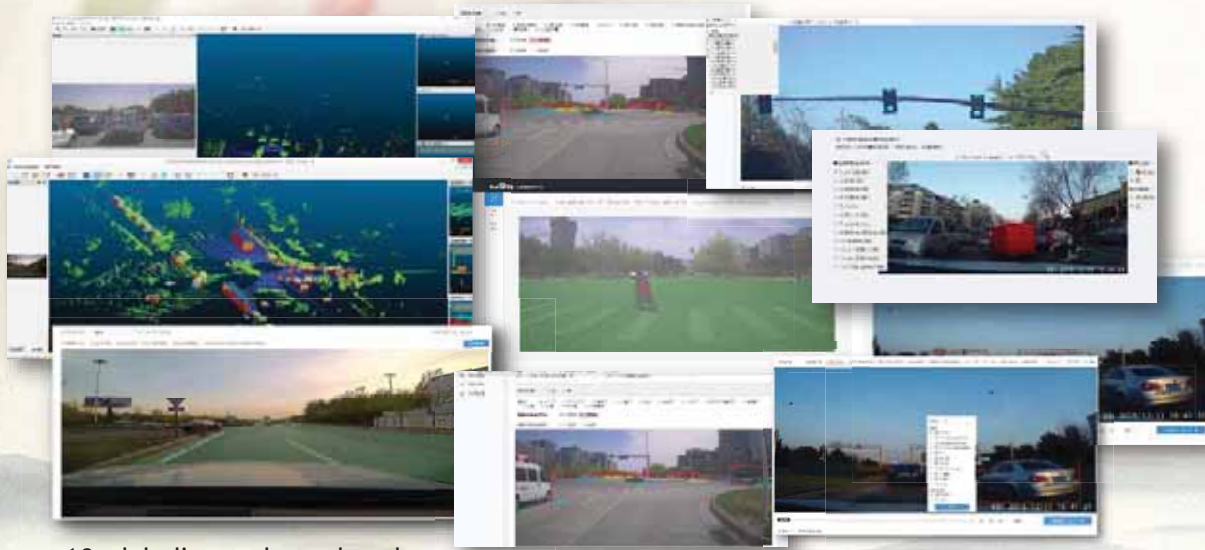
Baidu's Autonomous Driving Milestones



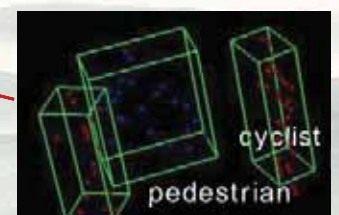
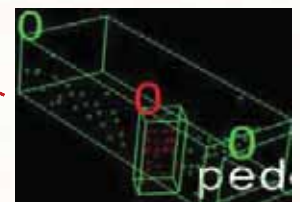
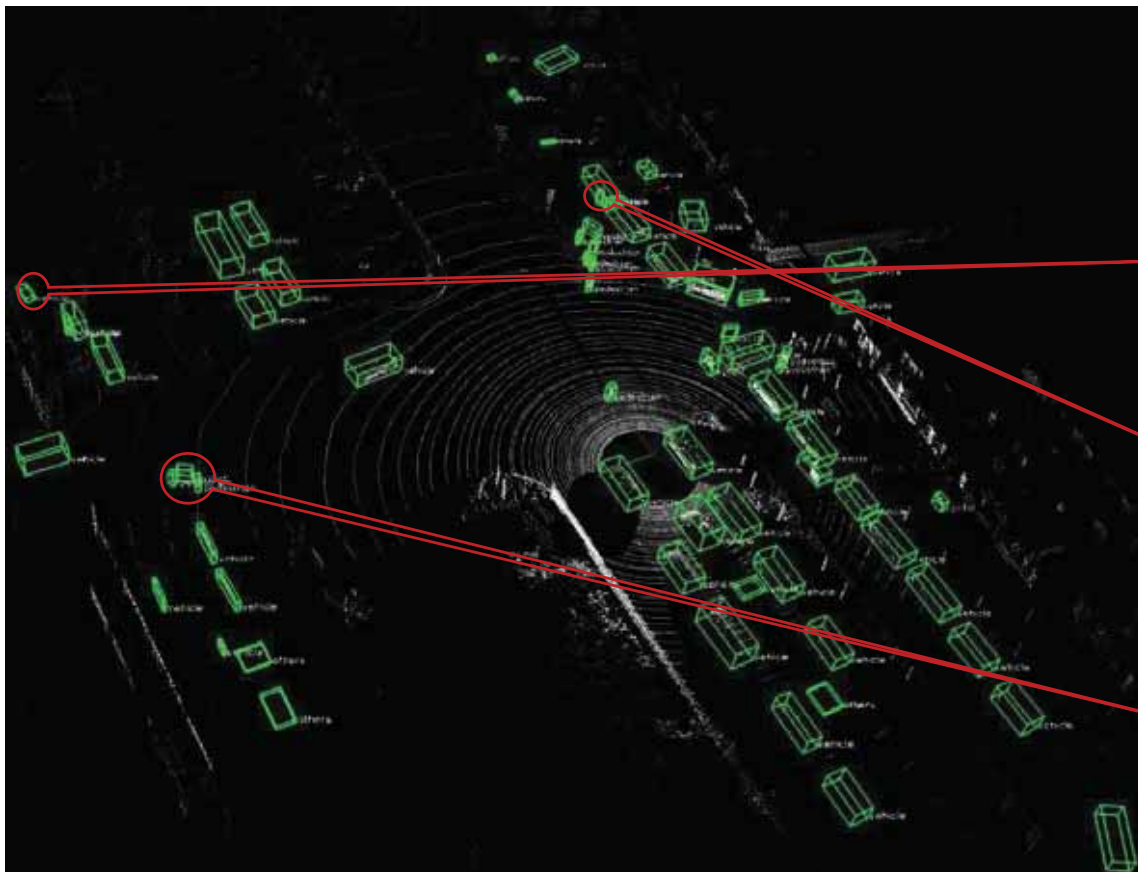
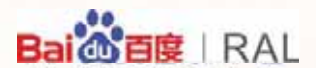
Apollo Architecture



Ground Truth Data from Manual Labeling

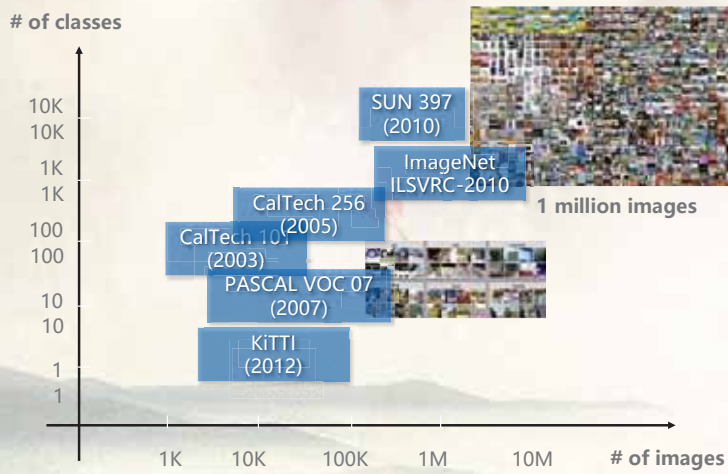


- 10+ labeling tasks and tools
- Hundreds of in house professional labelers
- Millions of image/point cloud data with ground truth labels



Too Much Data: Daily Road-Test

Academic Research Data Sets

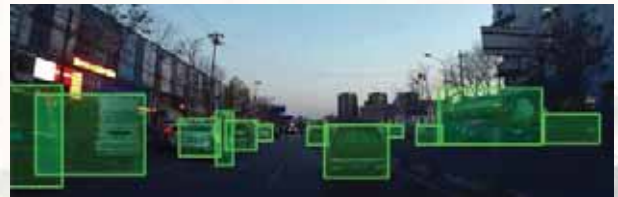


The largest academic research dataset has 1M images

Baidu Self-driving Car Fleet



1M images per-camera, per-car, per-day



ApolloScape : Open Tools and Datasets for Autonomous Driving Research

自动驾驶公开数据集
Open Autonomous Driving Datasets

下一代仿真技术
Next-Gen Simulation

导航和控制
Navigation

ApolloScape Roadmap



Datasets

Scene Parsing Dataset

- a state-of-the-art mobile scanner
- 200,000 frames
- per-pixel semantic labeling



adding new sensors

- Thermal Images
- Rare Events with Stereoview

Rare Event

- Crowd Sourced



- Continuous Expansion in Coverage and Size



Simulation

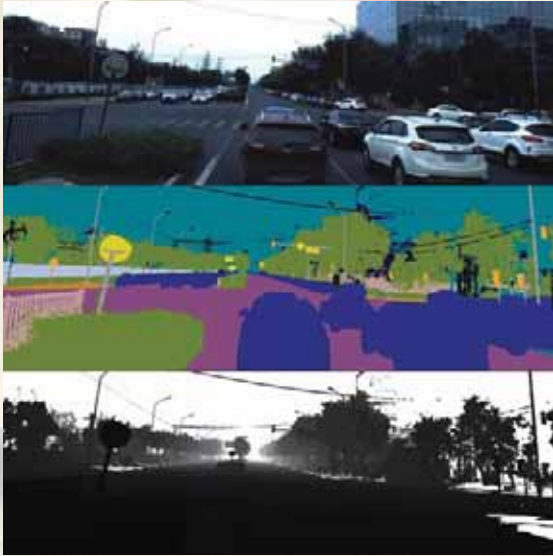


- Moving Object Trajectories

- Driving Behavior Modeling
- Real-scene Modeling

- Integrated Perception and Navigation

Data Sets

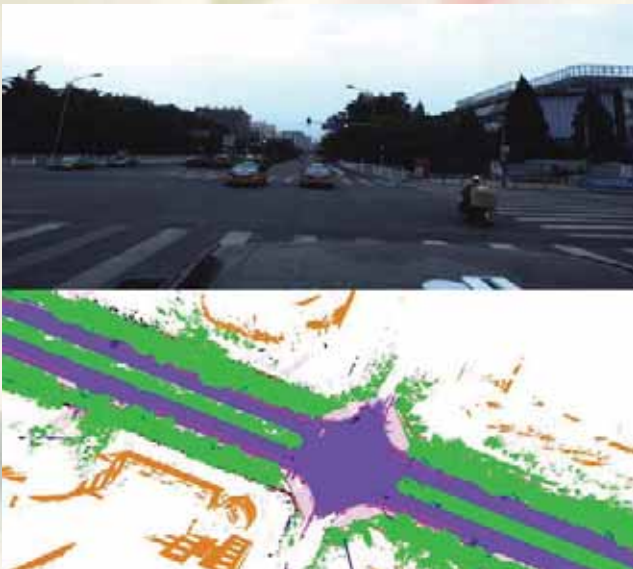


Scene Parsing

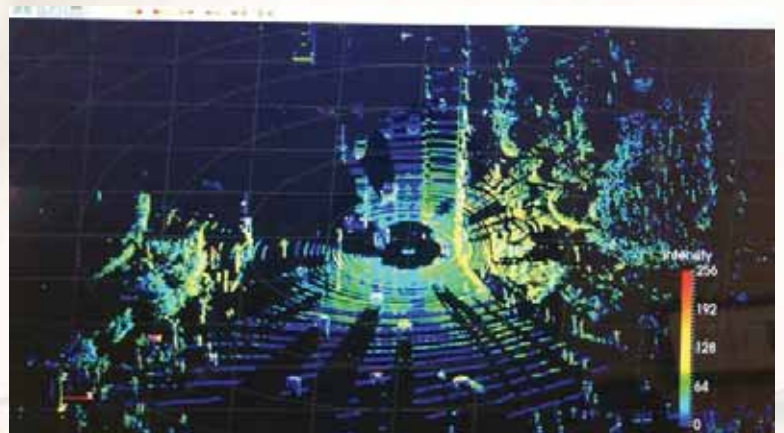


Fine-grain Lane-markings

Data Sets



3D Trajectories



Dense Trajectory

Data Sets: 3D Car Model Fitting

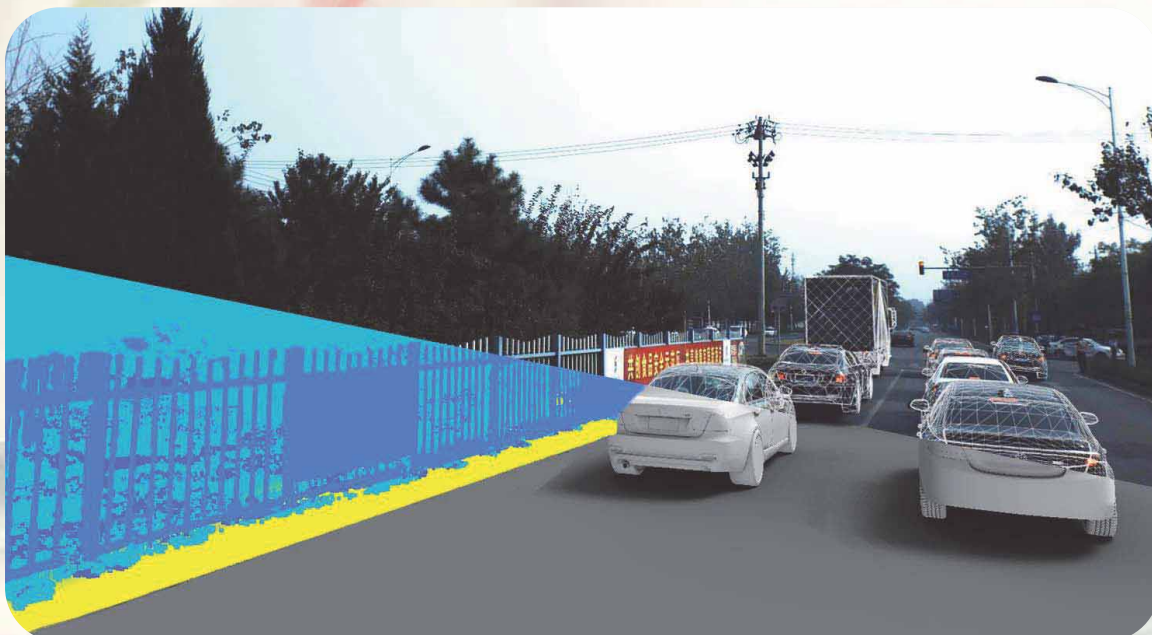


<https://www.kaggle.com/c/pku-autonomous-driving>

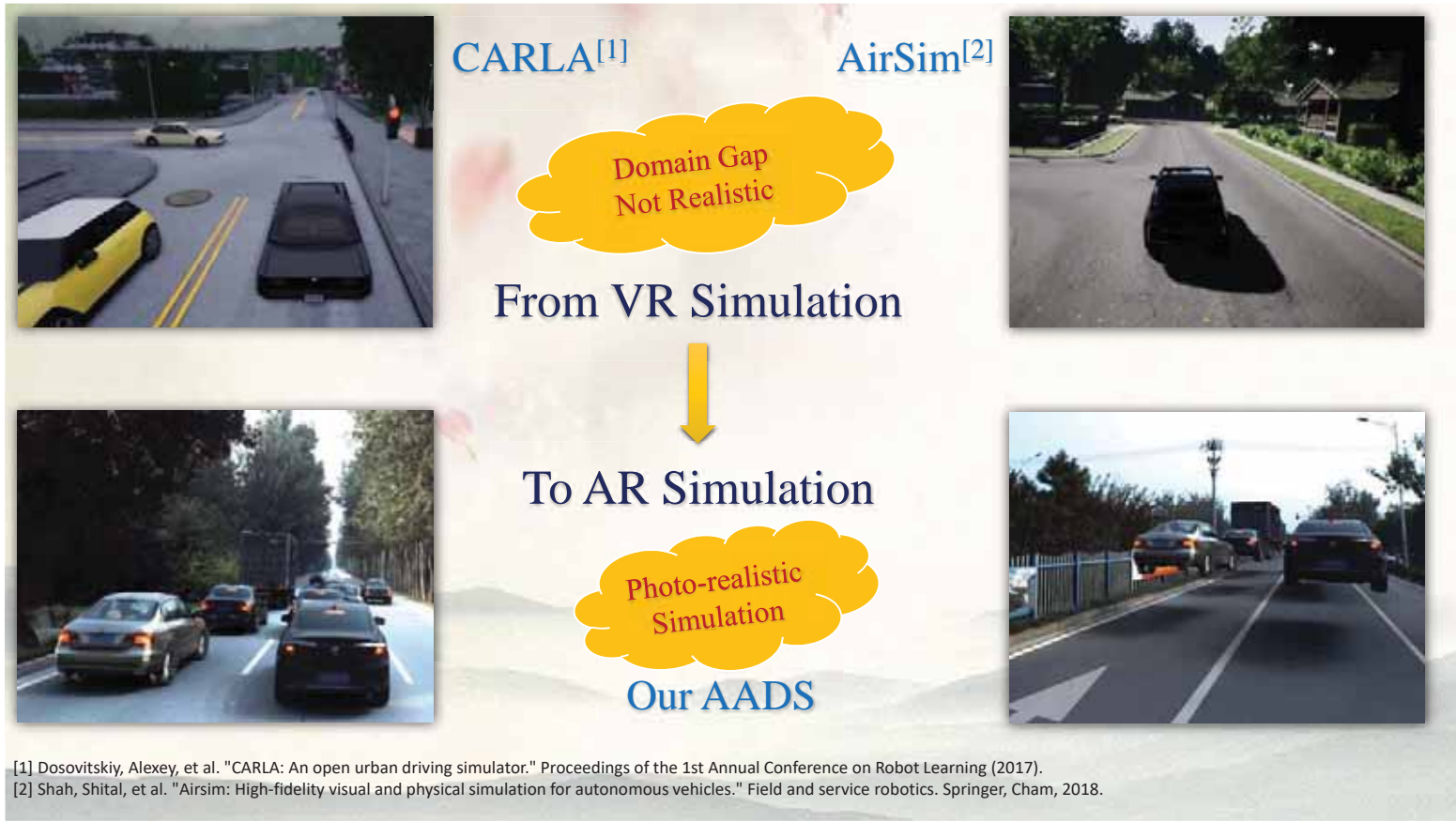
Baidu/Peking University
3D Pose Kaggle Challenge:

- Start: Oct, 2019
- Case Prize: \$25,000

AADS: Augmented autonomous driving simulation using data-driven algorithms^[1]



[1] W. Li, C. W. Pan, R. Zhang, J. P. Ren, Y. X. Ma, J. Fang, F. L. Yan, Q. C. Geng, X. Y. Huang, H. J. Gong, W. W. Xu, G. P. Wang, D. Manocha, R. G. Yang. Science Robotics. 4, eaaw0863 (2019).



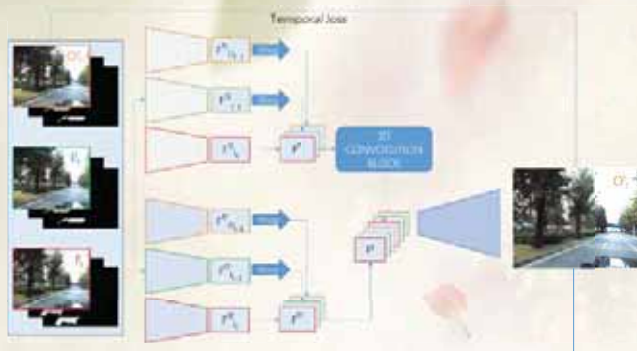
AADS Pipeline



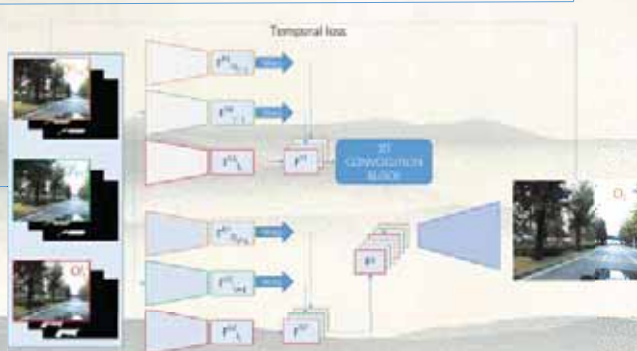
Data Preprocess



Clean background via deep video inpainting



Input video



Inpainted video



High Quality Automatic Object Removal for Autonomous Driving Videos

Paper ID: 1950

Novel View Synthesis

stitch mask | final result
 simple fusion



Large-baseline NVS :

$$\arg \min_{\{x_i\}} \sum_i \lambda_1 E_1(x_i) + \lambda_2 E_2(x_i) + \sum_{(i,j) \in \mathcal{N}} \lambda_3 E_3(x_i, x_j) + \lambda_4 E_4(x_i, x_j) + \lambda_5 E_5(x_i, x_j)$$

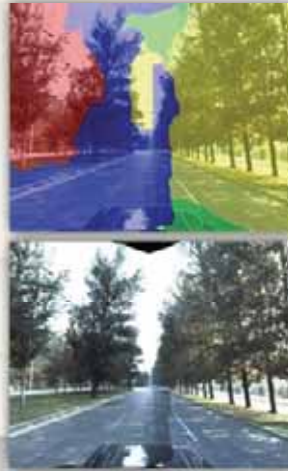
$$E_1(x_i) = E_{angle}(x_i) W_{label}(x_i)$$

$$E_2(x_i) = \begin{cases} \infty, & M(i) = 1 \\ 0, & \text{otherwise} \end{cases}$$

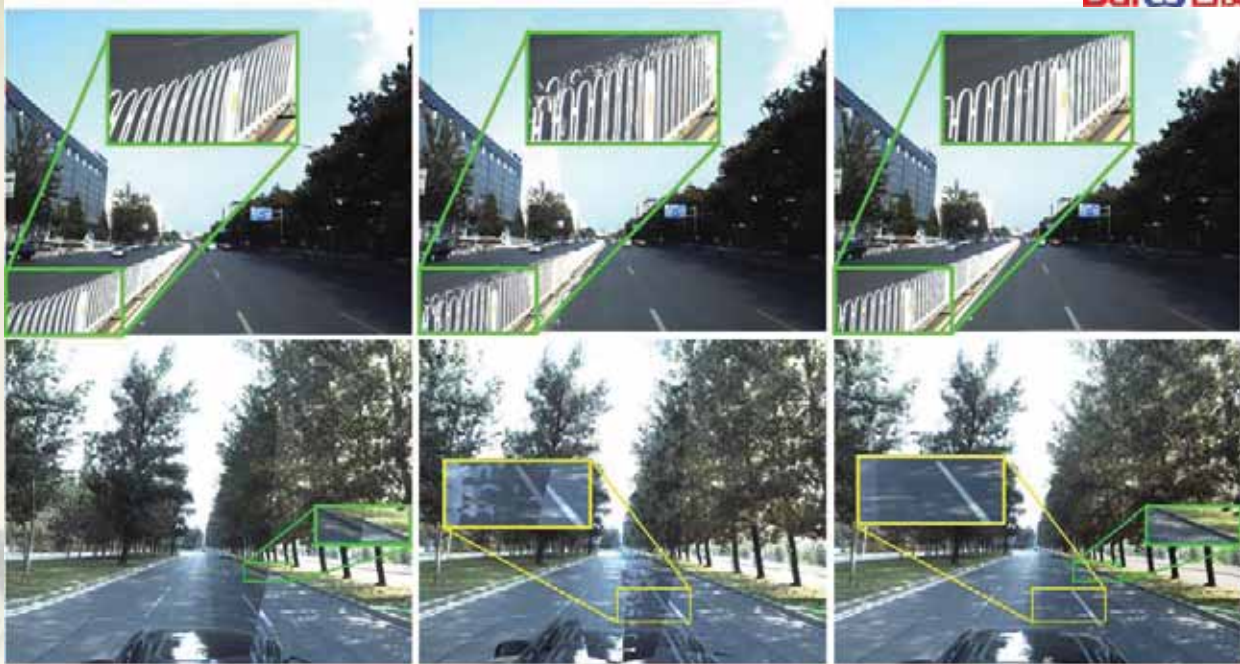
$$E_3(x_i, x_j) = \min(\|c_i^{x_i} - c_i^{x_j}\|^2, \tau_c) + \min(\|c_j^{x_i} - c_j^{x_j}\|^2, \tau_c)$$

$$E_4(x_i, x_j) = |g_i^x - g_j^x| + |g_i^y - g_j^y|$$

$$E_5(x_i, x_j) = \min(|d_i^{x_i} - d_i^{x_j}|, \tau_d) + \min(|d_j^{x_i} - d_j^{x_j}|, \tau_d)$$



Novel view synthesis comparison



Liu et al.^[1]

Chaurasia et al.^[2]

Our method

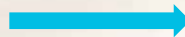
[1] F. Liu, M. Gleicher, H. Jin, A. Agarwala, Content-preserving warps for 3D video stabilization. ACM Trans. Graph. 28, 44 (2009).

[2] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, G. Drettakis, Depth synthesis and local warps for plausible image-based navigation. ACM Trans. Graph. 32, 30 (2013).

View Interpolation and Extrapolation

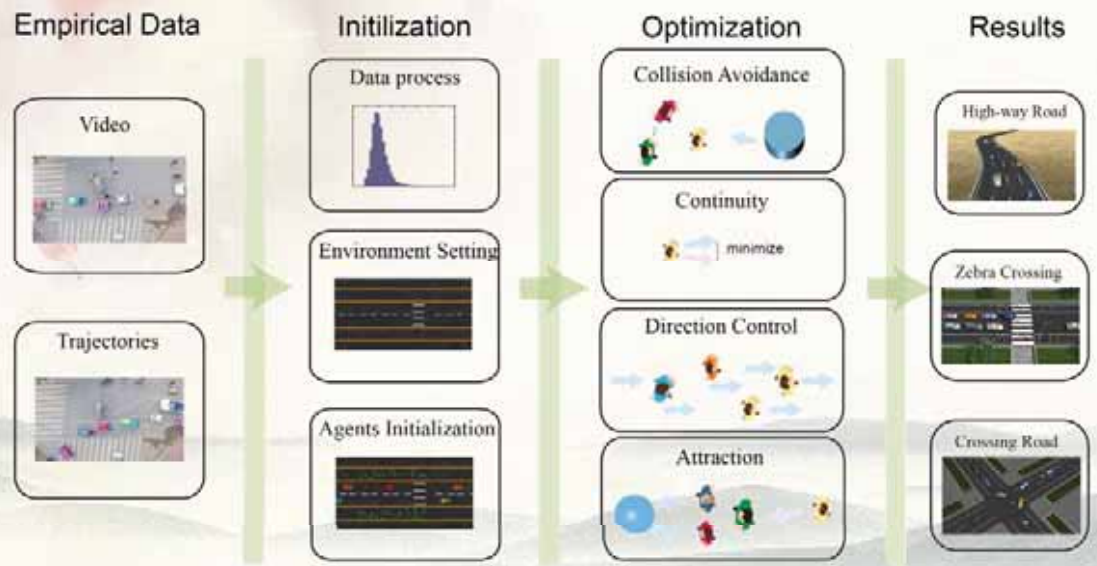


AADS-Flow Sim



Flow Simulation Framework

- Data set
Trajectories of agents
- Initialization
Environment setting、
Agents Initialization
- Optimization Algorithm
Combine data-driven method
and dynamics-based method
- Simulation results
We can simulate traffic with
different roads and agents



Sidewalk

Characters: cars and pedestrians

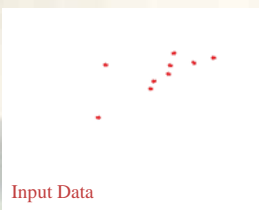
Agent Number: 80 and 50

Time Performance: 0.0319 s/f

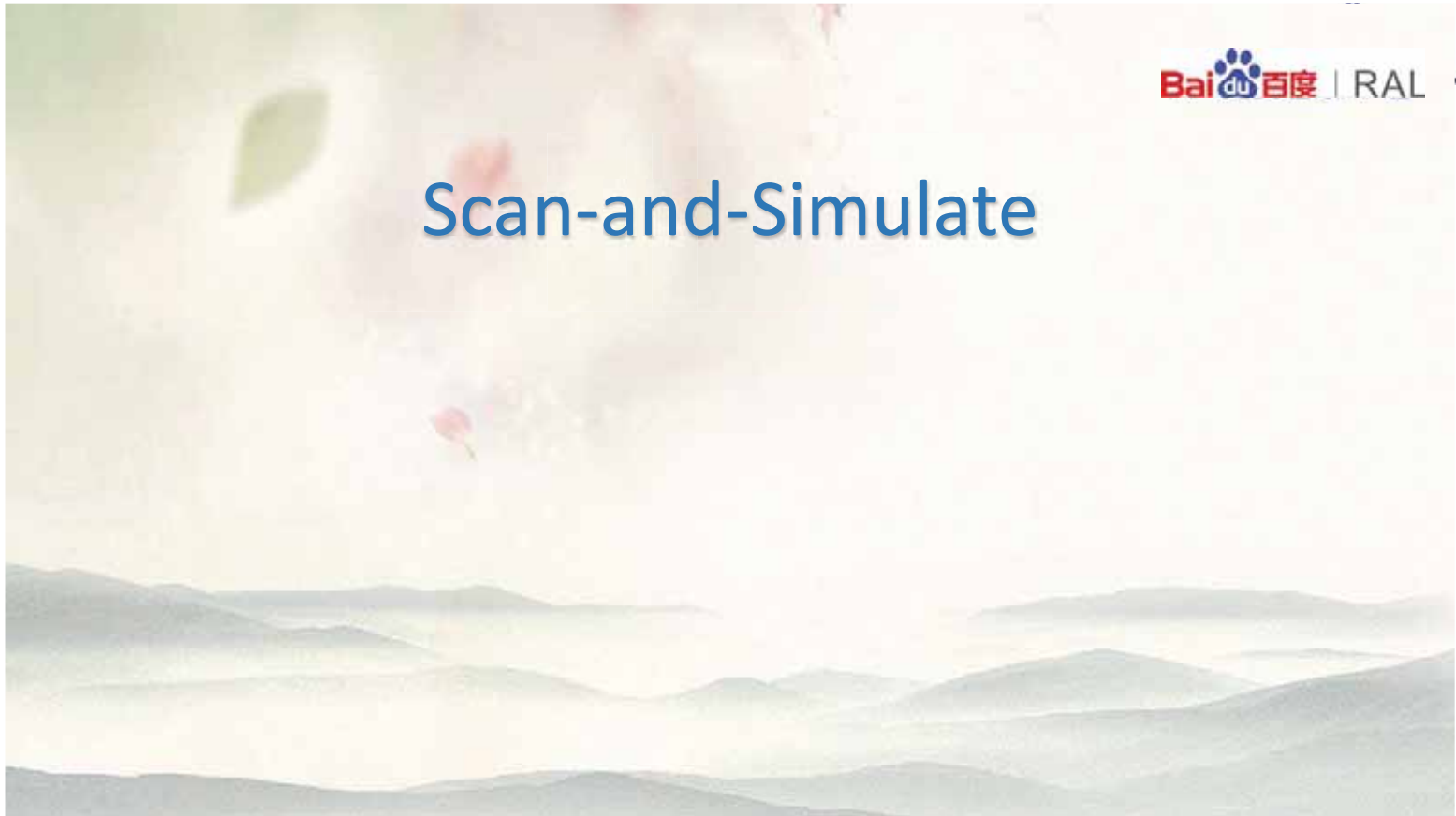
Dataset: [NGS 2013]
(trajectories)
[Zhang et al. 2012]
(trajectories)



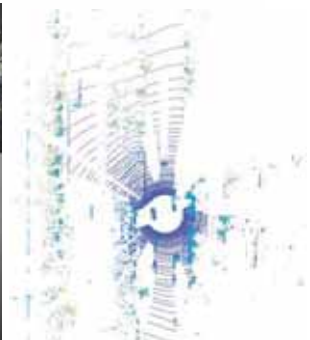
Our method can also combine different datasets into a single scenario.



Scan-and-Simulate



End-to-End Simulation



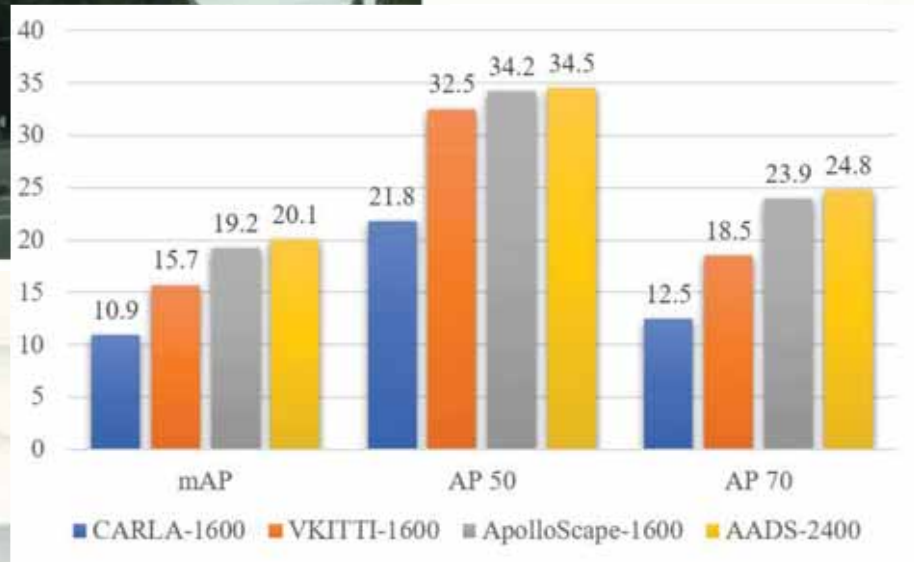
38



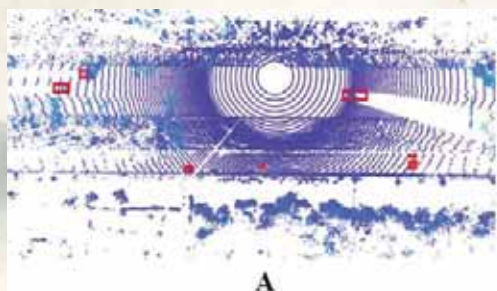
Baidu | RAL

Evaluation of RGB simulation

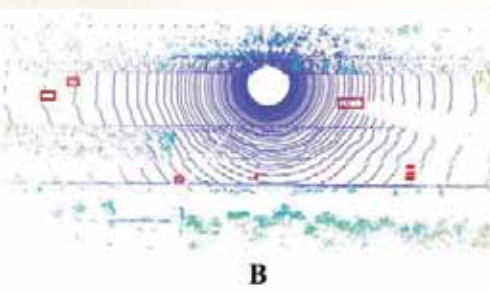
CARLA	Virtual-KITTI
Our AADS	Cityscape (for testing)



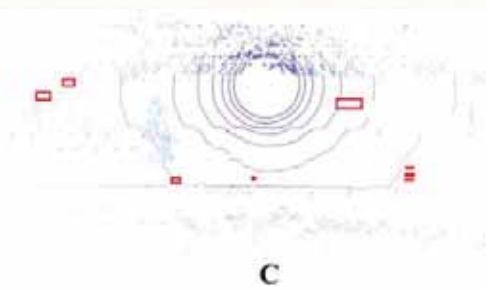
LiDAR simulation



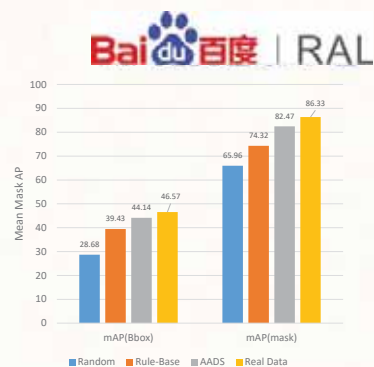
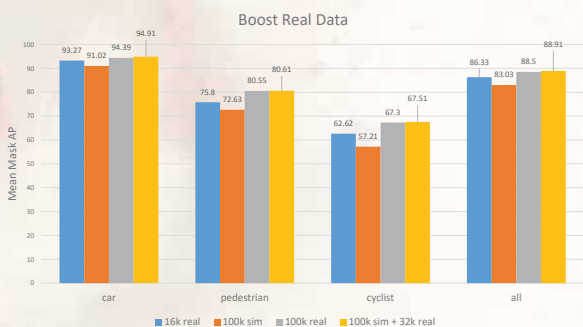
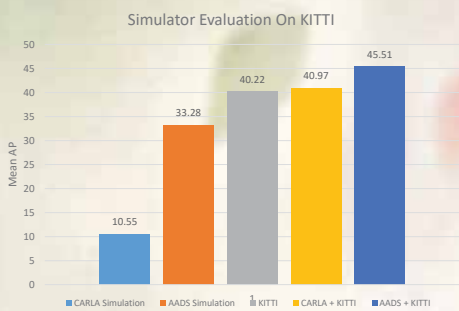
Velodyne VLS-128



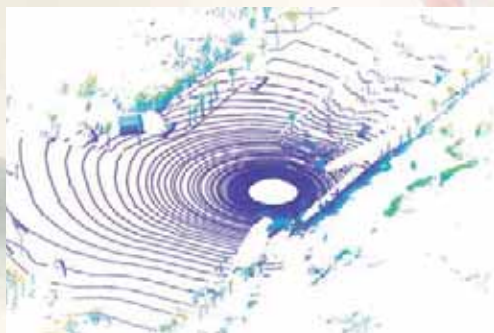
Velodyne HDL-64E



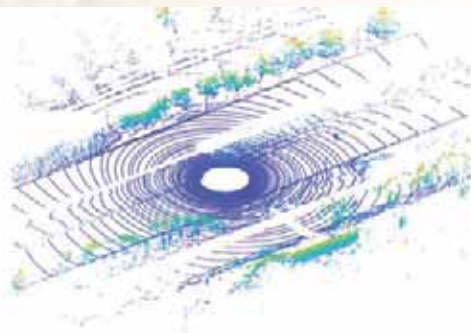
Velodyne VLP-16



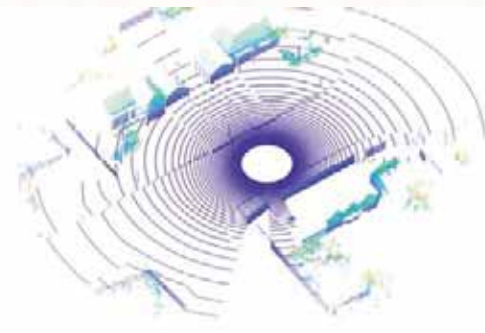
物体检测和跟踪



AADS Simulation



Captured Data



CARLA Simulation

APOLLOSCOPE

Navigation Research

Baidu 百度 | RAL

How do robot move now?



Fixed in a cage



Empty Space



Slow and Stop

Motivation

- Challenging Traffic Conditions: China



46

Motivation

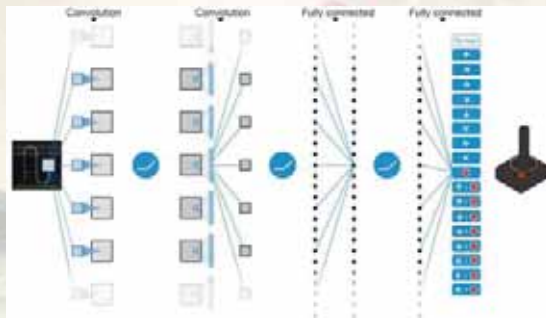
- Most Challenging Traffic Conditions: India



47

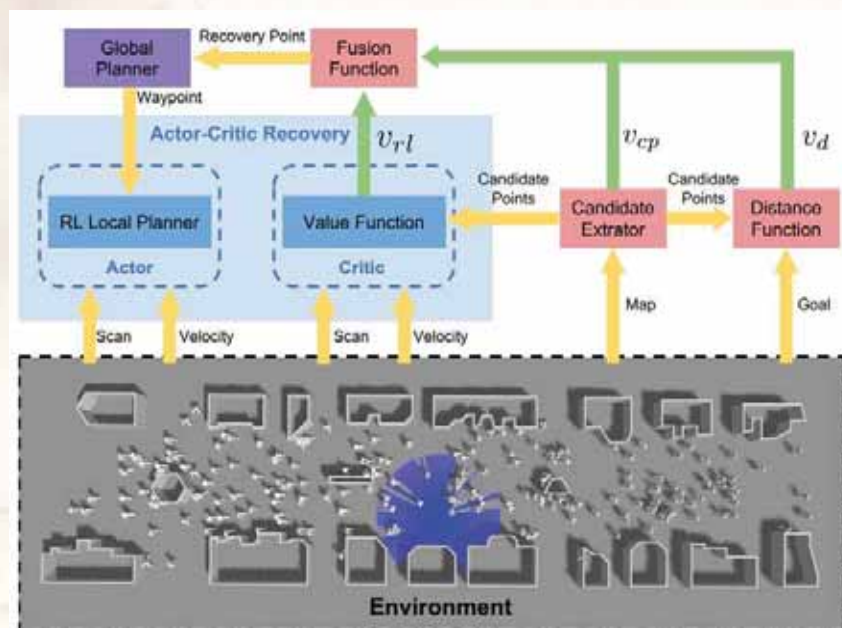
Deep Reinforcement Learning

- Deep learning can build an end-to-end map from raw sensor to command without any hand-crafted rules
- Reinforcement Learning have the potential ability of surpass human-level intelligence

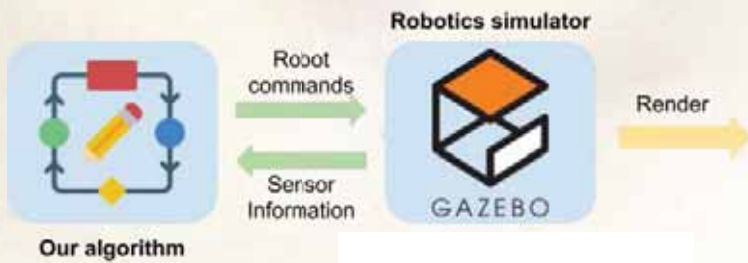


Reinforcement learning formulation

- Controller
 - Inputs: three frames of 2D LiDAR data and goal
 - Output: steering velocity
- Minimize objectives about policies of entire crowd
 - Collision cost
 - Time to reach goal
- Solve using PPO (Proximal Policy Optimization), adapted for multi-agents

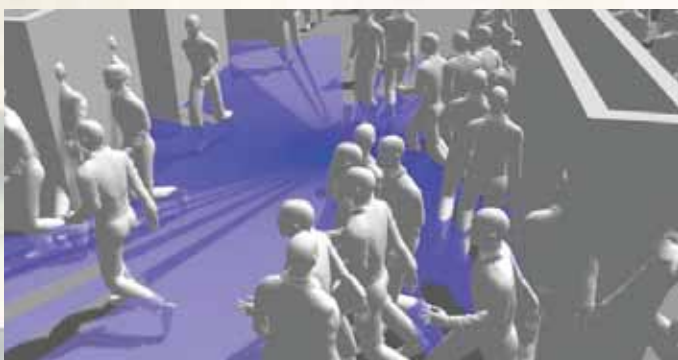


System Architecture



Sim-to-Real

Learning collision avoidance policy

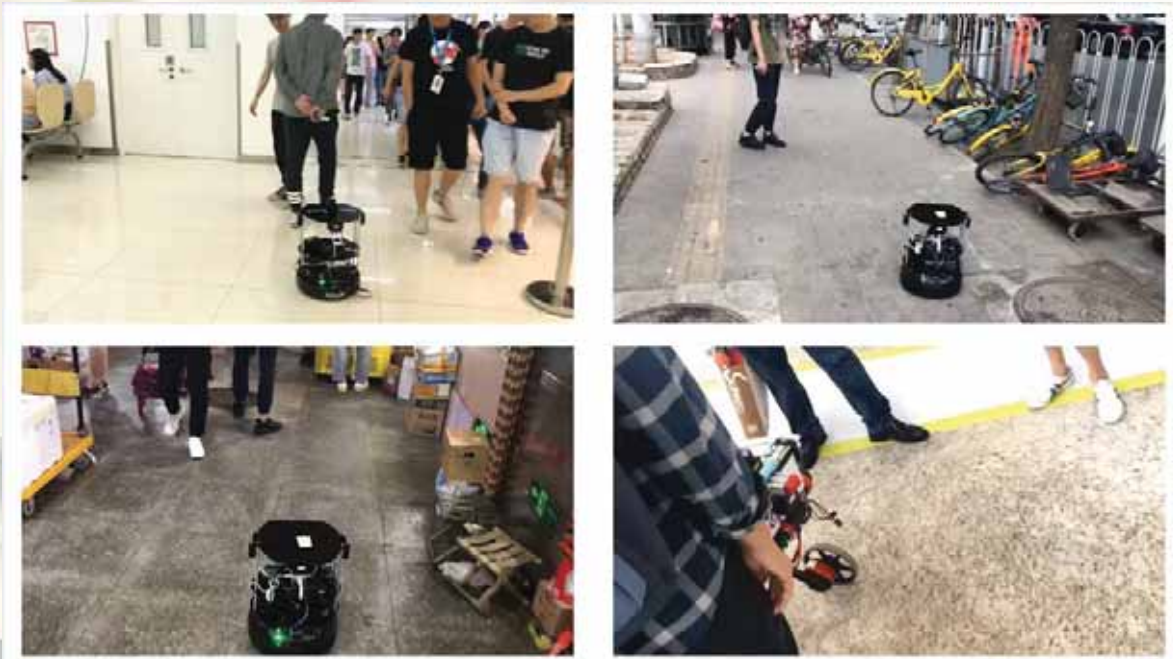


Real world

Navigate through Crowd



Follow me

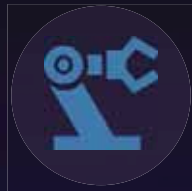


Baidu Autonomous Excavator

Apollo Extension



Mobile Platform



Actuator





Yang, et al, Tuesday, CT14 (LG-R14) Robotics in Construction



Summary

The cost of collecting, annotating, and testing with real data is too high
→ Simulation

- 3D Vision and 3D Graphics are complementary
- Domain gap in simulation vs real : Transfer learning??

Acknowledgment

- Prof. Guoping Wang @Peking University
- Prof. Xiaogang Jin @ Zhejiang University
- Prof. Wenping Wang, Jia Pan @Hong Kong University
- Prof. Dinesh Monacha @University of Maryland.

apollo

<http://apollo.auto/index.html>

APOLLOSCAPE

<http://apolloscape.auto>

Baidu 百度 | RAL

<http://research.Baidu.com/RAL>



Challenge for 3D Pose from a Single Image
Start Oct, 2019
@ kaggle.com

yangruigang@Baidu.com



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session II

Perception & Situation awareness

- **Title: Feature Generator Layer for Semantic Segmentation Under Different Weather Conditions for Autonomous Vehicles**
Authors: O. Erkent, C. Laugier
- **Title: An Edge-Cloud Computing Model for Autonomous Vehicles**
Authors: Y. Sasaki, T. Sato, H. Chishiro, T. Ishigooka, S. Otsuka, K. Yoshimura, S. Kato



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Feature Generator Layer for Semantic Segmentation Under Different Weather Conditions for Autonomous Vehicles

Özgür Erkent¹, Christian Laugier¹

Abstract—Adaptation to new environments such as semantic segmentation in different weather conditions is still a challenging problem. We propose a new approach to adapt the segmentation method to diverse weather conditions without requiring the semantic labels of the known or the new weather conditions. We achieve this by inserting a feature generator layer (FGL) into the deep neural network (DNN) which is previously trained in the known weather conditions. We only update the parameters of FGL. The parameters of the FGL are optimized by using two losses. One of the losses minimizes the difference between the input and output of FGL for the known weather domain to ensure the similarity between generated and non-generated known weather domain features; whereas, the other loss minimizes the difference between the distribution of the known weather condition features and the new weather condition features. We test our method on SYNTHIA dataset which has several different weather conditions with a well-known semantic segmentation network architecture. The results show that adding an FGL improves the accuracy of semantic segmentation for the new weather condition and does not reduce the accuracy of the semantic segmentation of the known weather condition.

I. INTRODUCTION

Semantic information of the environment provide valuable data for the navigation and planning in intelligent vehicles. Recent developments in deep learning methods make them dominant in semantic segmentation such as DeepLabV3 [1] and SegNet [2]. Furthermore, they can be integrated with spatial 2D maps of the environment [3], [4] to produce 2D spatial semantic maps of the surroundings [5]. However, adaptation to new environments such as varying weather conditions is still a challenging problem.

Deep learning methods need a tremendous amount of supervised data to train. Although datasets with labeled images exist for semantic segmentation of RGB images taken from autonomous vehicles at pixel level for classes such as *road*, *car*, *pedestrian*, etc. (e.g. [6], [7]), providing a dataset for each separate weather condition is exhaustive. Thus, we focus on the problem of unsupervised adaptation to new weather conditions and use the same deep neural network (DNN) model to semantically segment successfully both known weather condition and the new weather condition images.

In unsupervised domain adaptation, the known weather condition for which we have the labels is called as the *source domain data* and the new weather condition without labels is called as the *target domain data*. We assume that we already have an initial DNN model to estimate the semantic

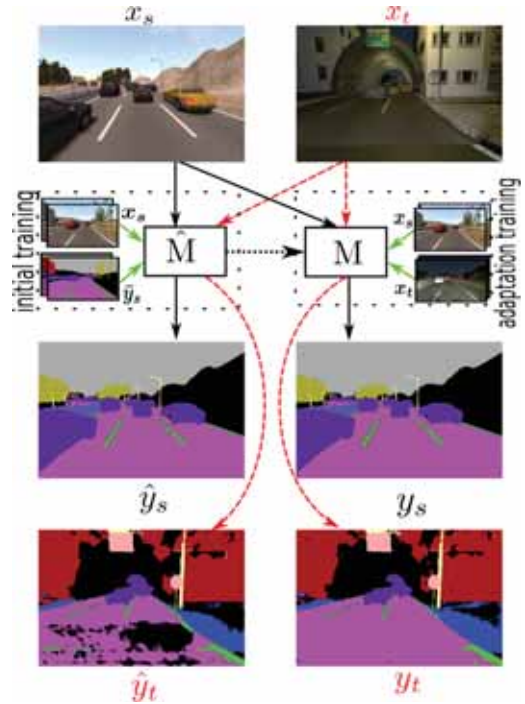


Fig. 1: A general overview. The initial model \hat{M} is trained with source domain images x_s and labels y_s . Then, it adapts to the new weather condition by only using the source domain images x_s and target domain images x_t . No labels are used during adaptation training. Green lines represent the flow of data used for training, black straight lines represent the flow for the segmentation of source domain and red dashed lines represent the flow for the segmentation of target domain. Note that the holes on the road segmentation at night diminish after adaptation while the segmentation of the daylight condition remains intact.

segmentation of the RGB images trained on *source domain data* as shown in Fig. 1. We propose to obtain a new DNN model by inserting a feature generator layer (FGL) such that the accuracy of the network improves with respect to the original DNN model for target domain segmentation and its accuracy does not degrade for the source domain segmentation. This implies that the same model can be used in different weather conditions for autonomous vehicles, which would be an advantage since the weather conditions can change during driving, e.g. it can start and stop raining during the same ride.

We insert the FGL after the initial feature extraction layers of the DNN, which corresponds to the “encoder” part of

¹ INRIA, Chroma Team, Rhône-Alpes, France. Correspondence: ozgur.erkent@inria.fr

the network. We only update the parameters of FGL for adaptation. We define two losses to optimize FGL. One loss is the difference between the distribution of the source and target domain data. We investigate three alternatives for this; Wasserstein distance [8], Jensen-Shannon divergence [9] and maximum mean discrepancy (MMD) [10]. The other loss measures the difference between the output and input of the FGL for source domain. We require the output of FGL to be similar to its input for the source domain.

The contributions of this paper can be listed as follows:

- To use only the source domain data without labels, target domain data without labels and pre-trained DNN model;
- To update only the parameters of the inserted FGL which results in faster adaptation training;
- To obtain an adapted DNN model with an improved accuracy for the target domain and a similar accuracy to the pre-trained model for source domain data;
- To report the accuracy of the adapted model for both domains.

We evaluate our approach by adapting the SegNet [2] for varying weather conditions since it is fast and does not require much memory which are important criteria for autonomous vehicles. We test it in Winter, Spring and Night conditions on SYNTHIA [7].

The rest of the paper is organized as follows. First, in Section. II, we discuss briefly the literature related to the domain adaptation. Then, In Section. III, we explain our solution by inserting a FGL with three alternative distance measures for data distribution. In Section. IV, we evaluate our method with SYNTHIA under two different weather conditions. Finally, we conclude the paper with a brief summary and possible future directions of research.

II. RELATED WORK

One of the approaches to unsupervised domain adaptation is to use images with incremental changes in the target domain. Dai *et al.* [11] train a network with the images obtained at different times of the twilight with incremental darkness to adapt to darkness. The labels for the first twilight time zone are estimated with the original network which was trained with daylight images are accepted as “*target domain ground truth*” labels. The network is re-trained with a mixture of source and “*target domain ground truth*” labels. Wulfmeier *et al.* [12] use an other incremental approach. They train a separate encoder for target domain data with generative adversarial network (GAN) [9] to make the target domain features similar to source domain features. Although the performance is improved for the target domain, one of the concerns is the requirement of incremental data collection which will not be feasible for various weather/illumination conditions for autonomous vehicles.

Another approach is to transform the input images from the target domain to the source domain so that they are similar in appearance. Porav *et al.* [13] obtain a dataset with a special stereo camera to capture the image of the scene both with rain and without rain and train a network

to convert the rainy images into de-rainy ones. The rainy condition is maintained with a glass with water droplets. The requirement of the exactly same scenes under different weather conditions is difficult to obtain for each weather condition. Cycada method [14] uses generators at different parts of the network. The usage of multiple generators ensure a cyclic transformation. Requirement of multiple generators makes this approach computationally expensive.

Another group of studies focus on minimizing the distribution in the output of the network. Vu *et al.* [15] implement adversarial entropy minimization for the outputs of the source and target domains. Wu *et al.* [16] find the geodesic loss [17] on the output of the network and back-propagate it for optimization. Zhang *et al.* [18] implement a super-pixel and label generator to transform the target domain output into the source domain output. The output is converted into a higher-dimensional feature space to be used with adversarial training, which may not always guarantee to represent the difference of the distribution between source and target domains. Furthermore, another problem with this group of approaches is that the necessary information to transform target domain into source domain data may have already been lost in the initial layers of the network.

Finally, we consider studies that compare the distributions of the features in both domains. The features generally have a high-dimension; therefore, adversarial training can be directly used. Tzeng *et al.* [19] proposed one of the early unsupervised domain adaptation in DNNs. They adapt a DNN by inserting a generator into the network and train it by using the MMD [20] to measure the discrepancy between the source and target domain features. Long *et al.* [21] also propose a similar architecture, but with multiple MMD kernels instead of single, and without additional adaptation layer. Additionally, both use the labels of the source domain to optimize the parameters of the network. Ganin *et al.* [22] used GANs [9] to adapt. In some other variants, GANs [9] are used for adaptation in semantic segmentation by deploying a discriminator at different layers of the network to measure the difference in distribution between the source and domain data features and the output (e.g. [23], [24], [25]). Peng *et al.* [26] and Hong *et al.* [27] insert a generator into the network to transform the distribution of target feature distribution into source domain feature distribution. Peng *et al.* [26] train based on W-GANs [8] with penalty gradient [28] while Hong *et al.* [27] use a GAN [9] based approach.

We covered the domain adaptation approaches that can be applied to DNNs by considering the computational complexity and the memory requirements for autonomous vehicles. It should be noted that some of the mentioned methods are used for adaption from the simulator domain to the real world domain; however, we will focus only on the new weather condition adaptation. We propose an approach which is different from the mentioned methods in three aspects. Firstly, we develop a method which does not require the source domain labels for adaptation. Secondly, it is usually a common practice to report the accuracy of the target domain with the new trained model [10]; since we propose to use

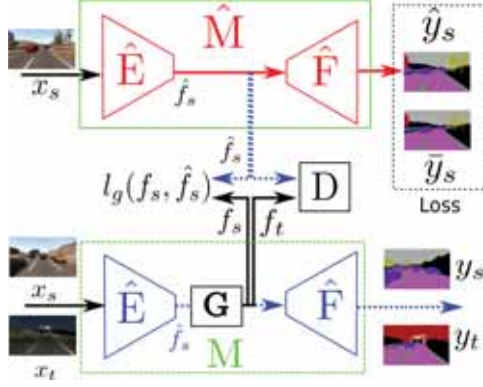


Fig. 2: First, the initial model \hat{M} is trained with source domain ground truth labels \hat{y}_s . The training of the initial model is shown with red straight lines. Then, FGL layer G is inserted into this model and the parameters of G is optimized by using the source domain and target domain features f_s and f_t . The training of adaptation is shown with black straight lines. The blue dotted lines are not used for back-propagation.

the same model in different weather conditions, we provide the performance of the same network in both target and source domains. Our final difference is that we don't update the parameters of the pre-trained model. We only update the parameters of the generator which significantly reduces memory and time requirements for training.

III. DOMAIN ADAPTATION

In classical unsupervised domain adaptation, source domain data with its labels $\mathcal{S} = \{(\mathbf{x}_s^i, y_s^i)\}_{i=1}^{n_s}$ and target domain data $\mathcal{X}_t = \{(\mathbf{x}_t^j)\}_{j=1}^{n_t}$ are provided where n_s and n_t are the number of source and target domain samples respectively. In our problem, we consider that only source data $\mathcal{X}_s = \{(\mathbf{x}_s^i)\}_{i=1}^{n_s}$, target data \mathcal{X}_t and a previously learned deep neural network model $\hat{M}_{\hat{\gamma}}$ with parameters $\hat{\gamma}$ to predict the labels y_s^i which can minimize the source cost $c_s(\hat{M}_{\hat{\gamma}}) = \Pr_{(\mathbf{x}, y) \sim p} [\hat{M}_{\hat{\gamma}}(\mathbf{x}) \neq y]$ are available. The source and target domains have probability distributions of \Pr_p and \Pr_q respectively. Our objective is to obtain a model such that it can minimize the target cost $c_t(M_{\gamma}) = \Pr_{(\mathbf{x}, y) \sim q} [M_{\gamma}(\mathbf{x}) \neq y]$ and source cost $c_s(M_{\gamma}) = \Pr_{(\mathbf{x}, y) \sim p} [M_{\gamma}(\mathbf{x}) \neq y]$ simultaneously using available source data, target data and the previously learned model.

We divide the neural network $\hat{M}_{\hat{\gamma}}$ into two: encoder $\hat{E}_{\hat{\theta}}$ and the decoder $\hat{F}_{\hat{\alpha}}$ such that $\hat{M}_{\hat{\gamma}}(\mathbf{x}) = \hat{F}_{\hat{\alpha}}(\hat{E}_{\hat{\theta}}(\mathbf{x}))$ as shown in Fig. 2. We propose a method where we insert a feature generator layer G_{θ} between $\hat{F}_{\hat{\alpha}}$ and $\hat{E}_{\hat{\theta}}$. All the parameters are kept constant except G_{θ} . The output of the new adapted layer becomes $\mathbf{f} = G_{\theta}(\hat{E}_{\hat{\theta}}(\mathbf{x}))$. The inputs and outputs of the layer are same in size. We require the generator to produce the target domain features \mathbf{f}_t similar to the source domain features \mathbf{f}_s in distribution and in addition, we need the source domain features to be generated with minimal change, which can be formulated as the following optimization:

$$\inf_{G(\cdot) \in \mathcal{G}} D(\Pr_{(\hat{\mathbf{f}}) \sim p}, \Pr_{(G(\hat{\mathbf{f}})) \sim q}) + \lambda l_g(\hat{\mathbf{f}}_s, G(\hat{\mathbf{f}}_s)) \quad (1)$$

where \mathcal{G} is the set of all possible generators, D measures the distance between the target and source domain feature distributions, $\lambda > 0$ is a regularization hyperparameter and l_g measures the cost between the source domain features $\hat{\mathbf{f}}_s^i = \hat{E}_{\hat{\theta}}(\mathbf{x}_s^i)$ and generated source domain features $G(\hat{\mathbf{f}}_s^i)$. We use the mean squared error (MSE) $l_g(\hat{\mathbf{f}}_s, G(\hat{\mathbf{f}}_s)) = \|\hat{\mathbf{f}}_s - G(\hat{\mathbf{f}}_s)\|_2^2$. We hold on the assumption that features $\{(\hat{\mathbf{f}}_t^j)\}_{j=1}^{n_t}$ sampled from the target distribution $\Pr_{(G(\hat{\mathbf{f}})) \sim q}$ contain necessary and sufficient information to classify the target domain data similar to the source data.

We propose three different distribution distance measures for $D(\Pr_{(\hat{\mathbf{f}}) \sim p}, \Pr_{(G(\hat{\mathbf{f}})) \sim q})$:

W-GAN proposal: First, we start with Wasserstein GANs [8]. m -th order Wasserstein distance between two distributions can be defined as follows:

$$W_m(\Pr_{(\hat{\mathbf{f}}) \sim p}, \Pr_{(G(\hat{\mathbf{f}})) \sim q}) = \inf_{\Pr_{\hat{\mathbf{f}}_s \sim p, G(\hat{\mathbf{f}}_t) \sim q}} \mathbb{E} [\|\hat{\mathbf{f}}_s - G(\hat{\mathbf{f}}_t)\|^m] \quad (2)$$

where the minimization is over all joint distributions. Then, Eq. 1 can be rewritten as follows:

$$\inf_{G(\cdot) \in \mathcal{G}} \inf_{\Pr_{\hat{\mathbf{f}}_s \sim p, G(\hat{\mathbf{f}}_t) \sim q}} \mathbb{E} [\|\hat{\mathbf{f}}_s - G(\hat{\mathbf{f}}_t)\|^m] + \lambda l_g(\hat{\mathbf{f}}_s, G(\hat{\mathbf{f}}_s)) \quad (3)$$

the second part of the optimization deals with the loss in the source domain. Hence its optimal value is not affected by the target domain, we solve it as a minimization problem in the source domain only. However, the first part needs to consider the joint distribution of both source and target domains for optimization. It is an optimal transport problem and a linear programming problem, therefore it always has a dual formulation (Kantorovich-Rubinstein) [29]. Arjovsky *et al.* [8] proposed an adversarial network for $m = 1$ and its variant with gradient penalty has been introduced by Gulrajani *et al.* [28]. We describe the implementation of the generative method with W-GANs in Algorithm.1.

GAN proposal: Secondly, we use the GANs based on Jensen-Shannon (JS) distance implemented by Goodfellow *et al.* [9] JS distance between $\Pr_{(\hat{\mathbf{f}}) \sim p}$ and $\Pr_{(G(\hat{\mathbf{f}})) \sim q}$ can be defined as follows:

$$JS = KL(\Pr_{\hat{\mathbf{f}}_s \sim p} \| \Pr_{\hat{\mathbf{f}} \sim r}) + KL(\Pr_{G(\hat{\mathbf{f}}_t) \sim q} \| \Pr_{\hat{\mathbf{f}} \sim r}) \quad (4)$$

where KL is the Kullback-Leibler divergence and $\Pr_{\hat{\mathbf{f}} \sim r} = (\Pr_{\hat{\mathbf{f}}_s \sim p} + \Pr_{G(\hat{\mathbf{f}}_t) \sim q})/2$ is the mixture. We describe the implementation of the generative method with GANs in Algorithm.1.

MMD proposal: Finally, we use maximum mean discrepancies which was used in domain adaptation problems with DNNs (e.g. Long *et al.* [21]). It maximizes the two-sample test power and minimizes the Type II error. If \mathcal{H}_k is the reproducing Hilbert space (RKHS) with a characteristic positive-definite reproducing kernel k and the mean embedding of distribution p in \mathcal{H}_k is a unique element $\mu_k(p)$ such that $\mathbb{E}_{\mathbf{x} \sim p} f(\mathbf{x}) = \langle f(\mathbf{x}), \mu_k(p) \rangle_{\mathcal{H}_k}, \forall f \in \mathcal{H}_k$; then, we define $D(\Pr_{(\hat{\mathbf{f}}) \sim p}, \Pr_{(G(\hat{\mathbf{f}})) \sim q})$ to be the MMD between distributions p and q which is the RKHS distance between the mean

Algorithm 1: Feature Generator Domain Adaptation

Require: $\mathcal{X}_s, \mathcal{X}_t, G_\vartheta, D_\beta, \lambda > 0, \hat{E}, n$: Mini-batch size,
 $A \in \{\text{WGAN}, \text{GAN}, \text{MMD}\}$, $n_{\text{crit}}, \lambda_p > 0$: Penalty gradient
if ($A = \text{WGAN}$) **OR** ($A = \text{GAN}$) **then**
 initialize the parameters of D_β and G_ϑ

else

provide a characteristic positive-definite kernel k and
 initialize the parameters of G_ϑ

while (β, ϑ) not converged **do**

Sample $\{(\mathbf{x}_s^i)\}_{i=1}^n \in \mathcal{X}_s, \{(\mathbf{x}_t^i)\}_{i=1}^n \in \mathcal{X}_t$

Compute $\hat{\mathbf{f}}_s^i = \hat{E}(\mathbf{x}_s^i), \mathbf{f}_t^i = G(\hat{E}(\mathbf{x}_t^i)), \mathbf{f}_s^i = G(\hat{E}(\mathbf{x}_s^i));$

if $A = \text{WGAN}$ **then**

for $k = 1, \dots, n_{\text{crit}}$ **do**

Sample a random number $\epsilon \sim U[0, 1];$

$\tilde{\mathbf{f}}^i = \epsilon \hat{\mathbf{f}}_s^i + (1 - \epsilon) \mathbf{f}_t^i;$

Update D by ascending

$$\frac{1}{n} \sum_{i=1}^n D(\hat{\mathbf{f}}_s^i) - D(\mathbf{f}_t^i) - \lambda_p (\|\nabla_{\tilde{\mathbf{f}}} D(\tilde{\mathbf{f}}^i)\|_2 - 1)^2$$

Update G by descending

$$\frac{1}{n} \sum_{i=1}^n D(\mathbf{f}_t^i) + \lambda_g(\mathbf{f}_s^i, \hat{\mathbf{f}}_s^i)$$

else if $A = \text{GAN}$ **then**

for $k = 1, \dots, n_{\text{crit}}$ **do**

Update D by ascending

$$\frac{1}{n} \sum_{i=1}^n \log D(\hat{\mathbf{f}}_s^i) + \log(1 - D(\mathbf{f}_t^i))$$

Update G by descending

$$\frac{1}{n} \sum_{i=1}^n \lambda_g(G(\hat{\mathbf{f}}_s^i) - \log(D(\mathbf{f}_t^i)))$$

else if $A = \text{MMD}$ **then**

Update G by descending

$$\begin{aligned} & \frac{1}{n} \sum_{i=1}^n \lambda_g(\mathbf{f}_s^i, \hat{\mathbf{f}}_s^i) + \frac{1}{n(n-1)} \sum_{l \neq j} k(\mathbf{f}_t^l, \mathbf{f}_t^j) \\ & + \frac{1}{n(n-1)} \sum_{l \neq j} k(\hat{\mathbf{f}}_s^l, \hat{\mathbf{f}}_s^j) - \frac{2}{n^2} \sum_{l,j} k(\mathbf{f}_t^l, \hat{\mathbf{f}}_s^j) \end{aligned}$$

embeddings of p and q . Using the Kernel trick, MMD can be computed with the expectation of kernel functions as follows:

$$\begin{aligned} \text{MMD}_k(\text{Pr}_p, \text{Pr}_q) &= \mathbb{E}_{\hat{\mathbf{f}}_s, \hat{\mathbf{f}}_{s'}} k(\hat{\mathbf{f}}_s, \hat{\mathbf{f}}_{s'}) + \mathbb{E}_{\mathbf{f}_t, \mathbf{f}_{t'}} k(\mathbf{f}_t, \mathbf{f}_{t'}) \\ &\quad - 2\mathbb{E}_{\hat{\mathbf{f}}_s, \mathbf{f}_t} k(\hat{\mathbf{f}}_s, \mathbf{f}_t) \end{aligned} \quad (5)$$

where $\mathbf{f}_s, \mathbf{f}_{s'} \sim p, \mathbf{f}_t, \mathbf{f}_{t'} \sim q$ which means that $\mathbf{f}_{s'}$ and $\mathbf{f}_{t'}$ are sampled from source and target domains respectively and they are not same samples with \mathbf{f}_s and \mathbf{f}_t . We explain the implementation of the method in Algorithm. 1. Since MMD has an unbiased U-statistic estimator, it can be used with a stochastic gradient descent (SGD) method to optimize the parameters of a deep neural network [21]. It is important to remind that MMDs require a large number of samples and a high computation time to optimize the parameters of the network.

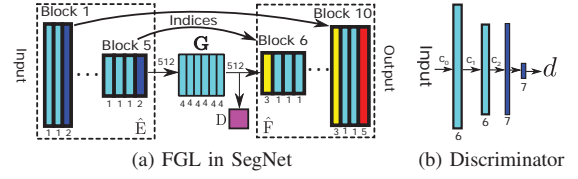


Fig. 3: (a) FGL is inserted after Encoder \hat{E} and before decoder \hat{F} . D measures the distance between the distributions of generated source and target domain features and used only during adaptation training. The labels for the layers are as follows: 1: 2D Convolution and Batch Normalization (BN) and ReLU, 2: Max-Pooling, 3: Upsampling, 4: 2D Convolution and ReLU, 5: Softmax. The generator has 512 input channels and 512 output channels. (b) Input has c_0 channels. After second convolutional layer, it is reduced to c_2 channels. Layer labels: 6: 2D Convolution and Instance Normalization, 7: Fully Connected Layer. The final output d is a scalar value.

IV. EXPERIMENTS

The number of datasets for the semantic segmentation with specific weather conditions is limited. Although datasets may contain examples from different weather conditions, the label of the weather condition for each image is not available.

SYNTHIA is a synthetic dataset obtained for semantic and instance segmentation of the surroundings of a vehicle [7]. We use only the frontal left RGB images and its semantically annotated label images for the ground truth. We use three different weather conditions in two different subway scenarios: Spring, Winter and Night from Sequence 01 and 06. We train our initial SegNet model with the labels and RGB images of Spring-01 which contains 1188 images.

The encoder layers of SegNet use convolution layers similar to VGG-16 [30]. The initial parameters of the network are taken from a pre-trained version of VGG-16 network for classification of ILSVRC/ImageNet [31].

We use Stochastic Gradient Descent for optimization of initial SegNet model with Spring as source domain and a learning rate of 1×10^{-4} and a momentum of 0.9. The mini-batch size is 6. Approximately 198 epochs are necessary to cover all the images in the training set. We train until the loss converges or the maximum number of iterations is 2000. Once the network converges, the parameters are kept constant and the FGL is inserted into the network after the encoder layer as shown in Fig. 3a. FGL has six 2D convolution layers with same size of input and outputs and a kernel size of 3×3 . Each convolution is followed by a rectified-linear non-linearity (ReLU) unit. We use a network for the discriminator when we use W-GAN [8] or GAN [9] to measure the differences in the distributions of the features. The Discriminator network is shown in Fig. 3b. It has two 2D-convolutional layers followed by the instance normalization layer [32]. Two fully connected (fc) layers follow these convolutional layers where the first fc is followed by a ReLU. The parameters

TABLE I: Adaptation to Winter for SYNTHIA with SegNet

%	Init	FGL+W1	FGL+JS	FGL+MMD
Spring-06	59.8(0)	59.4(-0.7)	59.5(-0.4)	59.7(-0.2)
Winter-01	55.9(0)	61.7(10.4)	60.6(8.4)	59.6(6.6)
Winter-06	47.8(0)	48.5(1.4)	48.5(1.4)	48.5(1.5)

TABLE II: Adaptation to Night for SYNTHIA with SegNet

%	Init	FGL+W1	FGL+JS	FGL+MMD
Spring-06	59.8(0)	59.7(-0.1)	59.9(0.2)	60.0(0.4)
Night-01	53.7(0)	58.6(9.1)	58.2(8.4)	57.5(7.0)
Night-06	35.6(0)	38.8(8.9)	38.3(7.6)	38.4(7.8)

are $c_0 = 512$, $c_1 = c_2 = 64$. For GAN, additional ReLU and softmax layer is added to the end of the discriminator. The initial parameters are chosen randomly from a normal distribution. To optimize the parameters of the Generator and the Discriminator network, we use Adam for optimization with hyperparameters $\alpha = 1 \times 10^{-4}$, $\beta_1 = 0.7$ and $\beta_2 = 0.9$. The hyperparameters for GAN and W-GAN are $\lambda_p = 0.1$, $\lambda = 10$, $n_{\text{crit}} = 1$ and a mini batch size of 128 is used. For the characteristic kernel k in MMD, we used an inverse multiquadratics kernel $k(\mathbf{f}_t, \mathbf{f}_s) = C / (C + \|\mathbf{f}_t - \mathbf{f}_s\|_2^2)$ where $C = 2d_f\sigma_f$. d_f is the dimensionality of the features. For MMD, the hyperparameters are $\lambda = 100$, $\sigma_f = 1$ and a mini batch size of 64 is used. The number of maximum iterations is 10000 for GAN and W-GAN and 5000 for MMD. The training takes approximately 4.8 h for W-GAN and GAN, and 13.7 h for MMD. The input size of the images are downsampled to 380×500 pixels. For analysis, the output image is later upsampled to the size of the SYNTHIA dataset ground truth label images. The inference time is around 1.3 ms on the GPU, this doesn't include the transfer time of the image to the GPU. All computation time evaluations are made on an Nvidia GTX 1080Ti.

We adapt the network to two different new weather conditions: winter and night. For adaptation, we use the images from sequence Winter-01 for winter with 1027 images and from sequence Night-01 for night with 935 images. After we obtain the adapted model, we test the source weather condition with images from Spring-06 to test the amount of accuracy degradation in the source domain after adaptation; with the target images used to adapt the network Winter-01 (or Night-01) and with the images that were not used for adaptation training, Winter-06 (or Night-06) to test the performance of the network with the unobserved target domain images. We believe that these cases are in accordance with a real scenario for an autonomous vehicle. The weather may change during driving and the incoming images will be the ones that were not used during adaption.

The results are shown in Table. I and Table. II. We use the mean of intersection over union (mIoU), which is widely used in semantic segmentation analysis. The first value is the mIoU, while the value in parenthesis is the amount of change with respect to the initial model after the adaptation method is applied. We compare three measure distances after the insertion of FGL and the initial model without FGL (Init).

It can be seen that the accuracy of the initial model

trained with Spring-01 degrades when the weather conditions change. When the road path sequence is different from the initially trained Sequence-01, the degradation is even higher (for the Sequence 06). Furthermore, the accuracy drop in Night-06 condition is higher. After adaptation in the same sequence, the accuracy increases for all the measures under the new weather condition. However, the accuracy increase for Winter-06 is not as high as the original sequence. To understand its reason better, we trained initial SegNet model with Winter-01 and tested Winter-6 with this new model. Even in this case, the mIoU is only 53.9%. Therefore, the capacity of the SegNet may not be sufficient to segment the Winter conditions for SYNTHIA. It should be noted that the accuracy still increases slightly for Sequence-06 after adaptation. Interestingly, the accuracy for Winter-01 is better than the source domain which can be due to the similar structures in the same sequence. For the Night condition, even the accuracy of the Spring-06 increases slightly for some distance measures. Probably this is due to the improvement of the segmentation of the dark regions such as inside the tunnel after adaptation to Night. For Night condition, the accuracy increases for both Sequences. For both Winter and Night conditions, Wasserstein critic for the distance measure gives a better result for the adaptation domains. Therefore, it can be concluded that for the network with a small capacity, W-GAN approach can be used with FGL to adapt to a new weather condition.

The qualitative results are shown in Fig. 4. The segmentation of Spring is not affected after adaptation (Fig. 4a) to the initial model for Winter condition. In Fig. 4b, the results are shown after the model is adapted with images from Winter-01 without any labels from either domains. In GT, the right side of the road is covered with undefined region. However, the initial model falsely segments this region as the sky. After adaptation with W-GAN, the region wrongly labeled as sky is reduced significantly. It should be noted that W-GAN is more effective for this case. In Fig. 4c, the Winter-06 is tested on the same model used in Winter-01, model which is adapted to the Winter condition based on the initial model on Spring-01 and adapted with the images from Winter-01 only. After segmentation with the original model, there is a hallucination of a vehicle segmentation in front of our vehicle (on the 3rd column). This diminishes after adaptation. For Night condition, we adapt the initial model by using the images from Night-01 and it is used for segmenting all the Night images. As it can be seen in Fig. 4d, there are holes on the road after segmentation with the initial model; however, they disappear after adaptation for all adaptation approaches. We use the same adapted model to segment the images of Night-06. As it can be seen in Fig. 4e, the right side of the road is undefined in the GT; however, the initial model segments this region as road, which is dangerous if the vehicle decides to drive towards this area. After adaptation with W-GAN, the wrong segmented area disappears. It reduces in size after adaptation with GAN, but MMD approach is not very effective in this case. Finally, we show a failed case in Fig. 4f. The road is covered with

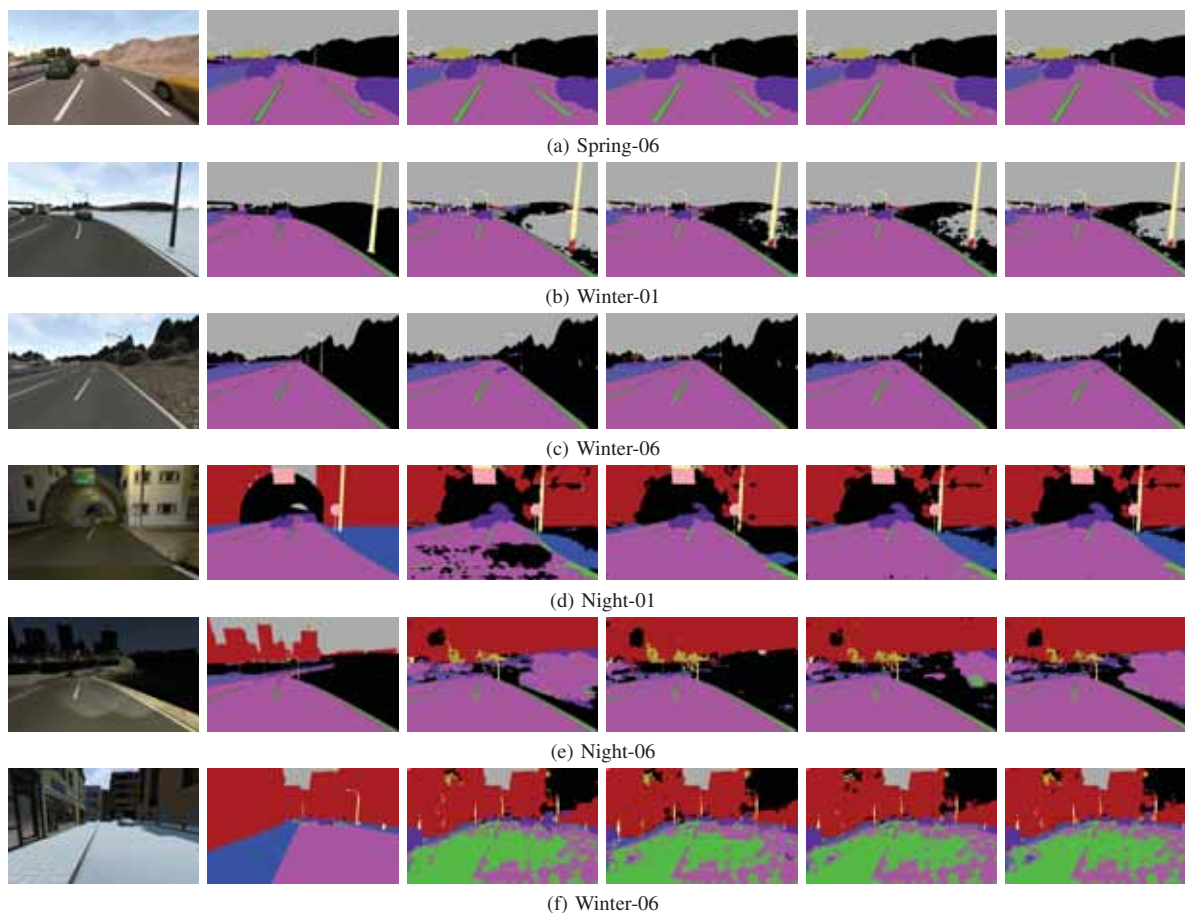


Fig. 4: Visual result samples for SYNTHIA dataset. From left to right: RGB, Ground Truth (GT), Result with the initial model (initial), Adaptation with W-GAN, Adapted with GAN and Adaptation with MMD.

snow and none of the methods segment the road successfully. This is probably due to the unrealistic visualization of the winter conditions where the snow resemble the lanemarks and most of it is recognized as landmark. Also, probably the target features do not contain sufficient information for the road with snow to be classified as road similar to the source domain data.

V. CONCLUSION

We showed that an unsupervised adaptation method can be used to improve the accuracy of semantic segmentation of a DNN for different weather conditions without using the labels of the source domain to modify the parameters of a pre-trained DNN model. We used SegNet [2] due to its speed and accuracy and a synthetic dataset with different weather conditions. As part of the future work, we will evaluate the method with real images in different weather conditions and develop the approach to be used with more complicated network architectures.

ACKNOWLEDGEMENT

This work was supported by Toyota Motor Europe.

REFERENCES

- [1] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," 2017. [Online]. Available: <http://arxiv.org/abs/1706.05587>
- [2] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE PAMI*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [3] L. Rummelhard, A. Nègre, and C. Laugier, "Conditional monte carlo dense occupancy tracker," in *ITSC*. IEEE, 2015, pp. 2485–2490.
- [4] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.
- [5] O. Erkent, C. Wolf, C. Laugier, D. Gonzalez, and V. Cano, "Semantic grid estimation with a hybrid bayesian and deep neural network approach," in *IEEE IROS*, 2018, pp. 888–895.
- [6] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016.
- [7] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez, "The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes," 2016.
- [8] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.
- [9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.
- [10] B. Gong, K. Grauman, and F. Sha, "Connecting the Dots with Landmarks: Discriminatively Learning Domain-Invariant Features for Unsupervised Domain Adaptation," in *ICML*, vol. 28, 2013, pp. 1–9.
- [11] D. Dai and L. Van Gool, "Dark Model Adaptation: Semantic Image

- Segmentation from Daytime to Nighttime,” in *ITSC*, 2018. [Online]. Available: <http://arxiv.org/abs/1810.02575>
- [12] M. Wulfmeier, A. Bewley, and I. Posner, “Incremental Adversarial Domain Adaptation for Continually Changing Environments,” in *ICRA*, 2018.
- [13] H. Porav, T. Bruls, and P. Newman, “I Can See Clearly Now : Image Restoration via De-Raining,” in *ICRA*, 2019. [Online]. Available: <http://arxiv.org/abs/1901.00893>
- [14] J. Hoffman, E. Tzeng, T. Park, J.-y. Zhu, U. C. Berkeley, P. Isola, K. Saenko, A. A. Efros, T. Darrell, and U. C. Berkeley, “CYCADA: Cycle-Consistent Adversarial Domain Adaptation,” in *ICML*, 2018, pp. 1–15.
- [15] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, “ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation,” in *CVPR*, 2019. [Online]. Available: <http://arxiv.org/abs/1811.12833>
- [16] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, “SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud,” in *ICRA*, 2019. [Online]. Available: <http://arxiv.org/abs/1809.08495>
- [17] P. Morerio, J. Cavazza, V. Murino, and P. Analysis, “Minimal-Entropy Correlation Alignment For Unsupervised Deep Domain Adaptation,” in *ICLR*, no. 2011, 2018, pp. 1–14.
- [18] Y. Zhang, P. David, and B. Gong, “Curriculum Domain Adaptation for Semantic Segmentation of Urban Scenes,” in *ICCV*, 2018.
- [19] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” *CoRR*, vol. abs/1412.3474, 2014.
- [20] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schoelkopf, and A. Smola, “A Kernel Method for the Two-Sample Problem,” *JMLR*, vol. 13, pp. 723–773, 2012.
- [21] M. Long, Y. Cao, J. Wang, and M. I. Jordan, “Learning Transferable Features with Deep Adaptation Networks,” in *ICML*, vol. 37, 2015. [Online]. Available: <http://arxiv.org/abs/1502.02791>
- [22] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.
- [23] Y.-H. Tsai, W.-C. Hung, S. Schuster, K. Sohn, M.-H. Yang, and M. Chandraker, “Learning to Adapt Structured Output Space for Semantic Segmentation,” in *CVPR*, 2018, pp. 7472–7481. [Online]. Available: <http://arxiv.org/abs/1802.10349>
- [24] Y. Zhang, Z. Qiu, T. Yao, D. Liu, and T. Mei, “Fully Convolutional Adaptation Networks for Semantic Segmentation,” in *CVPR*, 2018, pp. 6810–6818. [Online]. Available: <http://arxiv.org/abs/1804.08286>
- [25] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa, “Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation,” in *CVPR*, 2018, pp. 3752–3761. [Online]. Available: <http://arxiv.org/abs/1711.06969>
- [26] X. Peng, Y. Li, Y. L. Murphey, X. Wei, and J. Luo, “Domain Adaptation with One-step Transformation,” in *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018*. IEEE, 2018, pp. 539–546.
- [27] W. Hong, Z. Wang, M. Yang, and J. Yuan, “Conditional Generative Adversarial Network for Structured Domain Adaptation,” in *CVPR*, 2018, pp. 1–10.
- [28] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.
- [29] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008, vol. 338.
- [30] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, “Imagenet large scale visual recognition challenge,” *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.
- [32] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, “Instance normalization: The missing ingredient for fast stylization,” *CoRR*, vol. abs/1607.08022, 2016.



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

An Edge-Cloud Computing Model for Autonomous Vehicles

Yu Sasaki¹, Tomoya Sato¹, Hiroyuki Chishiro¹,
Tasuku Ishigooka², Satoshi Otsuka², Kentaro Yoshimura², and Shinpei Kato^{1,3}

Abstract— Edge-cloud computing for autonomous driving has been a challenge due to the lack of fast and reliable networks to handle a large amount of data and the traffic cost. The recent development of 5th Generation (5G) mobile network allows us to consider an edge-cloud computing model for autonomous vehicles. However, previous work did not strongly focus on the edge-cloud computing model for autonomous vehicles in 5G mobile network. In this paper, we present an edge-cloud computing model for autonomous vehicles using a software platform, called Autoware. Using 1 Gbit/s simulated network as an alternative of 5G mobile network, we show that the presented edge-cloud computing model for Autoware-based autonomous vehicles reduces the execution time and deadline miss ratio despite the latencies caused by communications, compared to an edge computing model.

I. INTRODUCTION

In recent years, autonomous driving has captured the attention from all over the world, such as the DARPA Urban Challenge [1], and many manufacturers have been working to realize autonomous vehicles. As the state-of-the-art embedded computers lack in scalability, most products are either used in restricted and controlled space or developed to handle partial functions, such as an emergency brake [2].

SAE Standards J3016 [3] defines *Level of driving*: Level 0 being normal driving and Level 5 being completely automated driving. Level 4 and Level 5 driving analyzes moving objects from a camera and predicts their paths in real time and, therefore, demands a large amount of computation resources.

In such situations, distributed computing is common in many fields. There are two main distributed system models for autonomous driving: edge-cloud computing model and cloud-based vehicular network (CVN) model.

The edge-cloud computing model for autonomous vehicles deploys edge devices in the vehicle to reduce latency and power consumption and cloud servers to provide a large amount of computation resources. There are a few studies of the edge-cloud computing model for autonomous vehicles. Kumar et al. [4] introduced a cloud-assisted system, called Carcel, and experimented with a golf cart and six iRobot Create robots. Although this study suggested the usefulness of edge-cloud computing for autonomous driving, Carcel required sensors on multiple locations and the car was driven at 2m/s.

¹Graduate School of Information Science and Technology, The University of Tokyo, Japan

²Hitachi Ltd., Japan

³Tier IV, Inc., Japan

We thank Akito Ohsato for his help in Autoware modification and providing ROSBAG data to test.

Another distributed model, CVN, is a model where multiple vehicles create a wireless link network, called Vehicular Ad hoc Networks (VANETs) [5], through which vehicles share data or computational resources. Although CVNs have been studied in previous works, VANETs are only useful in platoon vehicles and difficult to use if the vehicle is driven alone.

Edge-cloud computing models for autonomous driving are rarely analyzed due to the unreliable mobile network. For example, mobile networks such as the 4th Generation (4G) and Wi-Fi have an unreliable and narrow network bandwidth. The prior work noted that the mobile network was insufficient and its latency was critical for edge-cloud computing [4].

5th Generation (5G) mobile network has 20 Gbit/s bandwidth [6] in theory and much faster than the 4G network, which has 1 Gbit/s bandwidth at best [7]. 5G mobile network is beneficial to autonomous vehicles in various ways and provides sufficient bandwidth to send a large amount of data and receive the result with little latency. Thanks to 5G mobile network, vehicle vendors can use scalable cloud servers to execute heavy tasks while running light tasks in edge devices. This is both energy-efficient and cost-effective, therefore, energy-efficient offloading in 5G mobile network are studied in other fields such as mobile edge computing [8]. Meanwhile, the practicality of applying edge-cloud computing models to autonomous driving in 5G mobile network is an unsolved matter.

In this paper, we present an edge-cloud computing model for autonomous vehicles in 5G mobile network. We create 1 Gbit/s ethernet network between an NVIDIA DRIVE PX2, an NVIDIA's automotive computer, and a desktop machine to simulate 5G mobile network between an edge device and a cloud server. Since 5G has 20 Gbit/s at best and the target bandwidth is 1.2Gbit/s, 1 Gbit/s ethernet network is enough to simulate the bandwidth of 5G mobile network. We quantitatively evaluate the effectiveness of the presented edge-cloud computing model in Level 4 and Level 5 autonomous driving, called Autoware [9].

In the rest of the paper, the background is introduced in Section II. The overview of the presented edge-cloud computing model is explained in Section III. The evaluations of the presented edge-cloud computing model are introduced and the results are discussed in Section IV. We compare our work with related one in Section V and conclude this paper in Section VI.

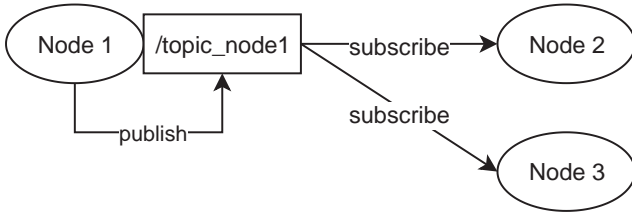


Fig. 1. Publisher/subscriber model in ROS

II. BACKGROUND

A. ROS

The most of automated driving applications use Robot Operating System (ROS) [10] as a main framework. ROS is an open-source meta-operating system for robots and provides extensive tools and libraries designed for processing data from sensors and actuators. An ROS process is called a *node*, and developers can easily specify which node runs on which machine.

Fig. 1 shows a publisher/subscriber model in ROS. A node communicates with another node using a message queue called *topic*. A system using ROS framework creates a directed acyclic graph structure where a vertex represents a node and an edge of two vertexes represents a communication between two nodes. The communication between nodes is sent by TCPROS protocol which uses persistent, stateful TCP/IP socket connections. Hence, even when two nodes are on different machines, one node can seamlessly acquire the *topic* data over the network.

B. Autoware

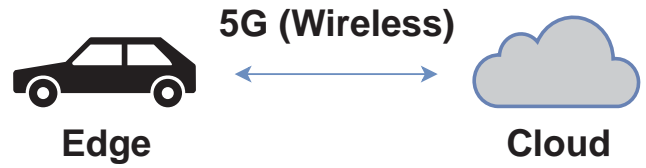
Autoware [9] is an ROS-based open-source platform developed for autonomous vehicles. Autoware collects the signals from various sensors such as camera, IMU, LiDAR, and GNSS, then, estimates the location of the vehicle, plans the path, and controls the vehicle. Autoware is designed for Level 4 and Level 5 autonomous driving and tested in demonstrated experiments in Japan [11], [12]. Now Autoware is maintained by the Autoware Foundation [13], which is a non-profit organization and composed of more than 20 global members. We use Autoware to simulate the real-world autonomous driving situation.

III. PRESENTED EDGE-CLOUD COMPUTING MODEL

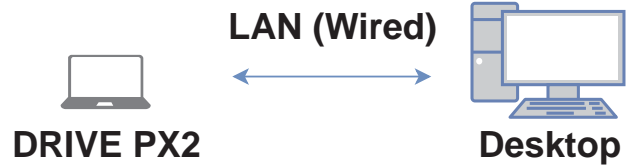
In this section, we explain the presented edge-cloud computing model. Before explaining the model, our simulation model and ROS component model are introduced.

Fig. 2 shows the actual 5G cloud model and simulation model. The actual 5G cloud model uses edge devices in the vehicle and cloud servers connected with 5G mobile network. In contrast, the simulation model uses an NVIDIA DRIVE PX2 and Desktop PC as an edge device and a cloud server, respectively.

We consider the situation where a vehicle runs on actual roads, and focus on two main features in Autoware [9]: the self-localization and path planning. We use Normal Distributions Transform (NDT) matching [14] and A* path planner



(a) Actual 5G cloud model



(b) Simulation model

Fig. 2. Actual 5G cloud model and simulation model

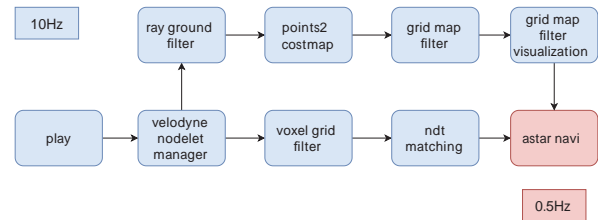


Fig. 3. Overview of our ROS component model

[15] as the self-localization and path planning, respectively, because these features are mainly used for Autoware-based autonomous vehicles.

Fig. 3 shows the overview of our ROS component model. *astar_navi* has the period of 0.5Hz (2,000ms) and other components have the period of 10Hz (100ms). Our ROS component model is composed of the following 9 components.

- 1) *play* reads point clouds from ROSBAG data (recording and playing back data).
- 2) *velodyne_nodelet_manager* creates 360 degree point clouds by Velodyne LiDAR data.
- 3) *voxel_grid_filter* downsamples the point cloud data (Velodyne LiDAR data in this case) by taking a spatial average of the points in the cloud.
- 4) *ndt_matching* handles NDT matching, which is used as the self-localization algorithm. NDT is a laser scan matching algorithm of self-localization that uses two point cloud data. One is made beforehand (reference scan) and the other is acquired from vehicle sensors (input scan).
- 5) *ray_ground_filter* removes ground from point clouds.
- 6) *points2costmap* creates 2D projection from 3D point clouds.
- 7) *grid_map_filter* creates non-entering area from 2D obstacle points.

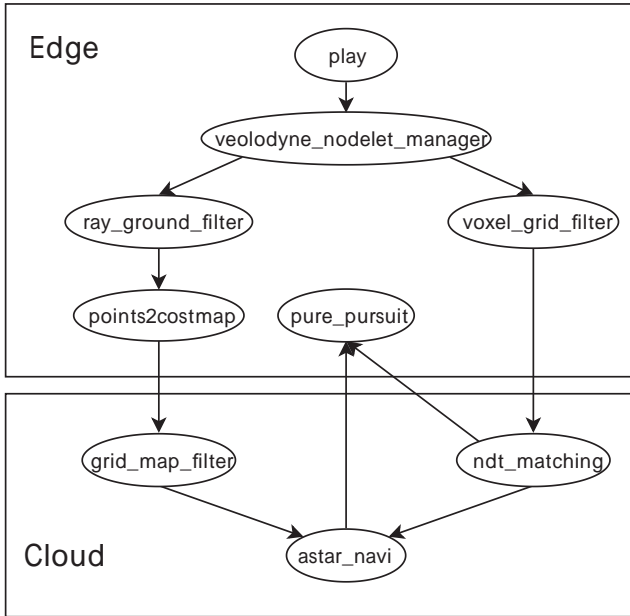


Fig. 4. Presented edge-cloud computing model

- 8) `astar_navi` handles A* planner, which heuristically calculates the shortest path from the start point to the goal point.
- 9) `pure_pursuit` is responsible for controlling vehicle actuators.

Fig. 4 shows the presented edge-cloud computing model. The edge devices execute `play`, `velodyne_nodelet_manager`, `ray_ground_filter`, `points2costmap`, `voxel_grid_filter`, and `pure_pursuit`. In contrast, the cloud servers execute `grid_map_filter`, `ndt_matching`, and `astar_navi`.

Preliminary Evaluations: The reason why the presented edge-cloud computing model is suitable for Autoware-based autonomous vehicles is because we have conducted the preliminary experiments with several edge-cloud computing models. In these edge-cloud computing models, these components are assigned to an edge device or a cloud server, respectively, in order to evaluate the execution time and communication cost. The results of the preliminary evaluations show that `ndt_matching` on DRIVE PX2 has the worst case execution time of 283ms. In addition, `astar_navi` on DRIVE PX2 often takes more than 10,000ms. The results indicate that DRIVE PX2 cannot execute NDT matching or A* planner within respective deadlines: 100ms and 2,000ms. Therefore, we conclude that the main processes of NDT matching (`ndt_matching`) and A* planner (`grid_map_filter` and `astar_navi`) should be run on the cloud server. Under this condition, we have chosen this model because of the lowest traffic (40 to 100MiBs). Other edge-cloud computing models exceeded 120MiB and 1 Gbit/s ethernet network could be the bottleneck. Therefore, these models would not be suitable.

We have made several modifications to Autoware, which

TABLE I
SPECIFICATION OF EDGE DEVICE AND CLOUD SERVER

	Edge	Cloud
CPU	2x Tegra X2	1x Core i7-8700
Architecture	2x NVIDIA Denver + 4x ARM Cortex A57	Intel x86 6 Core (12 Thread)
CPU Frequency	2.0GHz (Denver) + 2.0GHz (Cortex A57)	3.2GHz
Memory	16GB LPDDR4	32GB DDR4
GPU	1x Pascal GPU	1x GTX1080
Total CUDA Core	512	1280
GPU Memory	4GB GDDR5	8GB GDDR5
Linux Kernel	4.9.38-rt25-tegra	4.16.5-041605-generic
Ubuntu Version		16.04
ROS version		kinetic
Autoware version		1.9.1

are a mere fix to run on two machines. We do not modify the actual data processing algorithm that might compromise the execution time. Our main modification is creating the *sync node* to observe the message queue in each experiment and evaluate the execution time. Our implementation is available at <https://github.com/pflab-ut/distributed-autoware>.

IV. EVALUATIONS

A. Setups

The evaluations use the edge device and cloud machine, which are connected to the same network with 1 Gbit/s ethernet to simulate 5G mobile network. Table I shows the specification of the edge device and cloud server.

We use the simulation feature in Autoware. The simulation data used in the evaluations are those obtained by using the Velodyne HDL-32E LiDAR which covers 360 horizontal Fields Of View (FOV). The Velodyne HDL-32E LiDAR has 32 channels and covers +10 to 30 vertical FOV. The scan frequency is 10Hz and scan matchings are executed with 100ms period.

The input scan data used for simulations are available on ROSBAG STORE (<https://rosvag.tier4.jp/index/>), #178. These data are taken around Nagoya University, Japan, and the vehicle in this simulation travels in the same path for 3 minutes.

We denote the experiments, which are executed on the edge machine as E_{edge} , those on the cloud machine as E_{cloud} , and those under the edge-cloud computing model as E_{ec} .

Since our ROS component model has two different periods as shown in Fig. 3, we use A* planner and NDT matching because these features are mainly used for Autoware-based autonomous vehicles.

B. A* Planner

The execution time of A* planner depends on the location of the vehicle and the location of the goal. Therefore, we make four cases in which one autonomous vehicle parks on the campus street in Nagoya University, Japan, and starts to drive. These cases set the same start point and different goal points, and measure the execution time of A* planner to find the path to each goal.

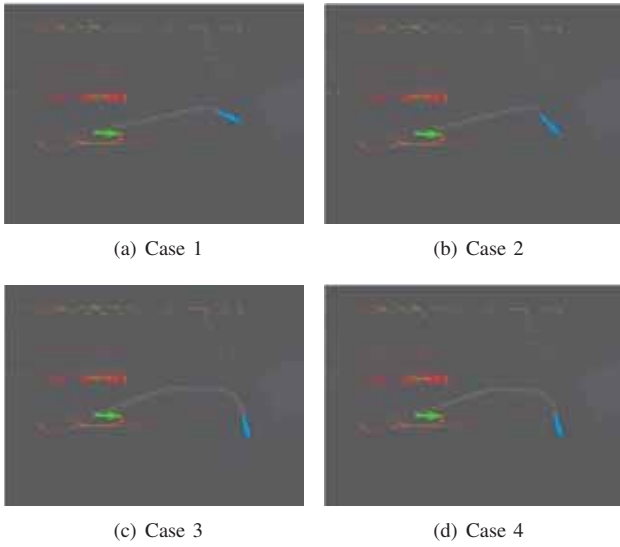


Fig. 5. Start, goal, and results of A* planner of case 1, 2, 3, and 4. The green arrow denotes the start point and the blue arrow denotes the goal point. The white line represents the presented path of A* planner. Red, green, and yellow points represent points acquired from LiDAR sensors.

Fig. 5 shows the start, goal and, results of A* planner of case 1, 2, 3, and 4. The green arrow denotes the start point and the blue arrow denotes the goal point. We execute 100 trials of each case and measure the execution time to find the path to each goal. The white line represents the generated path of A* planner.

Fig. 6 shows the execution time of A* planner of case 1, 2, 3, and 4. For goal 1, the execution time of A* planner is within 2,000ms deadline in every model. E_{edge} misses the deadline in the goal 2 while E_{cloud} and E_{ec} are well within the deadline. For goal 3, E_{edge} is 24,037ms and E_{cloud} is 2,450ms. However E_{ec} is within the deadline, 1,949ms. Finally, for goal 4, although E_{ec} misses the deadline, E_{edge} has about 10.4 times longer execution time compared to E_{ec} . As A* planner is a CPU dependent task, E_{cloud} and E_{ec} have significant advantages against E_{edge} . Furthermore, since the computation is distributed into two machines, the results also show that E_{ec} performs better than E_{cloud} , up to 8%.

These results strongly clarify the contribution of the presented edge-cloud computing model. In addition, they address the importance of the model because even the slight change of the goal point will result their execution time to be varied drastically.

C. NDT Matching

In NDT matching, the specific parameters must be defined with considering the trade-off between execution time and accuracy. Our parameters of NDT matching are as follows. The voxel size is 1 meter. The maximum step size of Newton's method is 100. Convergence threshold is 0.0001. The input point clouds from these LiDARs are downsampled by `voxel.grid.filter`. When downsampled, input point cloud spaces are divided into a cubic voxel. The size of

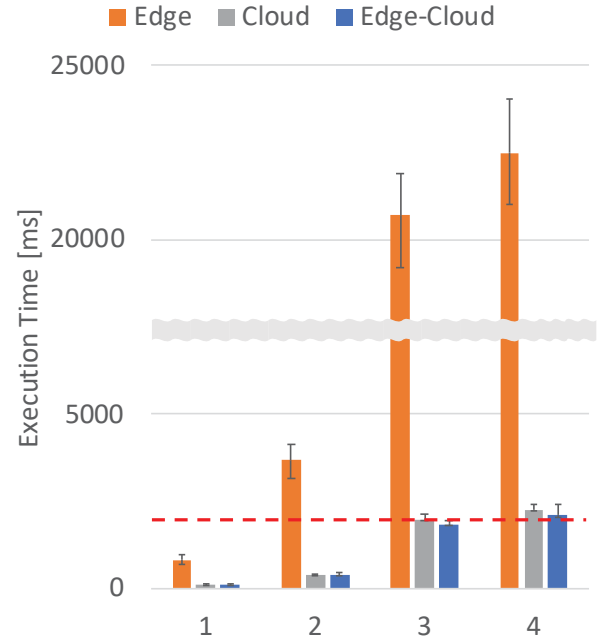


Fig. 6. Execution time of A* planner of case 1, 2, 3, and 4. The red dotted line represents 2000ms, the deadline of A* planner.

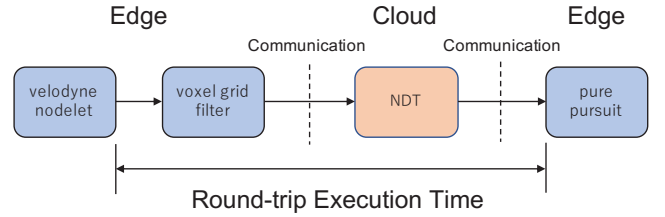


Fig. 7. Round-trip execution model

each cubic voxel, called *leaf size*, is set to 2.0. For parameters of A* planner, the resolution is 0.2, the maximum distance of `grid_map_filter` is 1 meter, and the filter is `points2costmap`. We measure two metrics in NDT matching: execution time and deadline miss ratio.

(i) **Execution Time:** We measure the round-trip execution time that takes to complete NDT matching. Fig. 7 shows the round-trip execution time (edge-cloud-edge) in NDT matching, as shown in Fig. 4. We run 2,000 scans of NDT matching for E_{edge} , E_{cloud} , and E_{ec} .

Fig. 8(a) shows the round-trip execution time of every scan. On average, E_{cloud} takes 13.9ms and E_{edge} takes 55.4ms, while E_{ec} takes 35.1ms, 57% faster than E_{edge} . In Fig. 8(b), the worst case execution time of E_{edge} is 348ms and that of E_{ec} is 103ms. This indicates that the presented edge-cloud computing model reduces the execution time compared to the edge computing model and slightly increases the execution time compared to the cloud computing model.

On E_{ec} , only 2 of 2,000 scans are longer than 100ms. This implies that even with unpredictable traffic cost, the presented edge-cloud computing model has more clustered

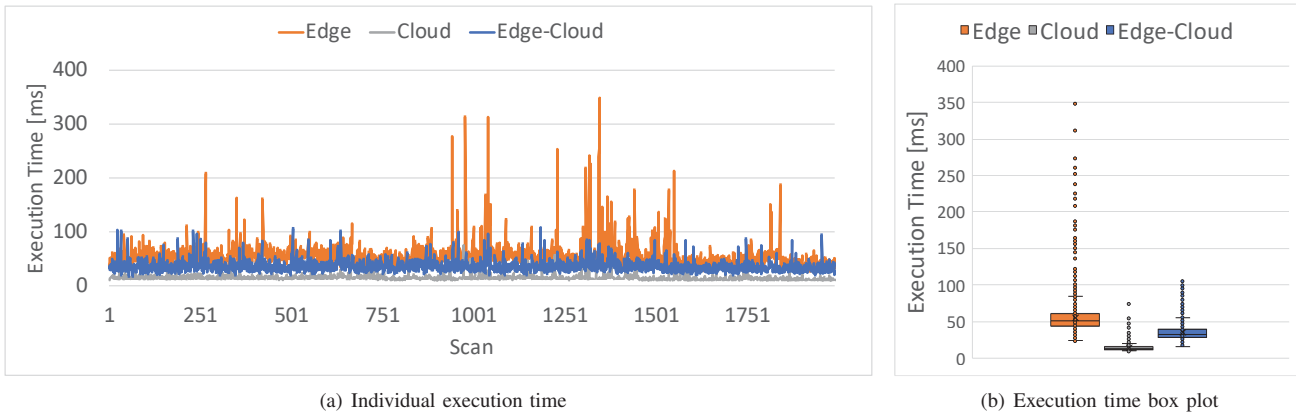


Fig. 8. Round-trip execution time of NDT matching

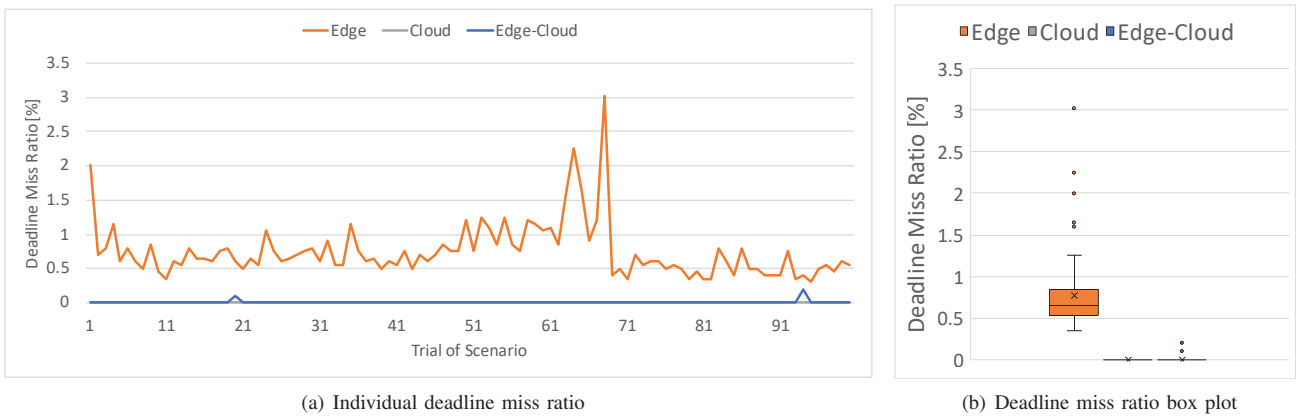


Fig. 9. Deadline miss ratio of NDT matching

execution time than the edge computing model. The existence of outlier is fatal in NDT matching because the accuracy of the algorithm heavily relies on the previous data. When the execution time of a certain scan is longer than 100ms, the next scan cannot use the previous data and make self-localization less accurate and longer to compute, which leads to another overload in NDT matching. This supports the fact that some of the spikes in Fig. 8(a) are clustered.

(ii) **Deadline Miss Ratio:** At the same time, we measure the deadline miss ratio of NDT matching as the following definition.

$$\text{Deadline Miss Ratio} = 1 - \frac{N_{pp}}{N_{vel}}, \quad (1)$$

where N_{pp} is the number of message pure_pursuit received and N_{vel} is the number of message velodyne_nodelet_manager sent.

The deadline miss ratio can be derived from the overview of the message queue. Since the message queue size of the every topic is set to 1, and velodyne_nodelet_manager tries to send new data every 100ms. If the component of ndt_matching misses its deadline (100ms), it does not fetch the latest data from velodyne_nodelet_manager. This causes the overflow

of the queue when velodyne_nodelet_manager tries to send the data in the next scan. Thus, the number of queue overflows equals to the deadline miss ratio. One scenario runs 2,000 scans like Fig. 8(a) as one trial. We run 100 trials of the scenario and measure the deadline miss ratio in each trial. Therefore, the total number of measurements is 200,000 (= 2,000 scans * 100 trials).

Fig. 9(a) shows the deadline miss ratio of each model. Out of 100 trials, E_{ec} achieves 98 trials without any deadline miss, which is almost the same as E_{cloud} . Meanwhile, E_{edge} has the maximum of 3% deadline miss ratio and every set has at least one deadline miss. As Fig. 9(b) shows, the best case in E_{edge} is 0.35%, which is worse than the worst case in the edge-cloud computing model (0.20%). This experiment ensures us the stability in the execution time of the presented edge-cloud computing model. Only 6 of 200,000 measured are longer than 100ms deadline, which puts the success ratio (N_{pp}/N_{vel}) of the presented edge-cloud computing model as 99.997%.

V. RELATED WORK

Edge-cloud computing is an uprising field due to the widespread use of IoT and mobile devices. One of the well-known applications is mobile edge computation offloading

(MECO) [16], which is used in mobile applications. Mobile devices have constraints in their computation power of mobile devices and their battery life. Many applications, depending on their uses, offload some of their computation on to their servers over the internet. There are many optimization methods introduced which parts of the application are offloaded to the cloud. Mao et al. focused on energy consumption and computation power [17], while Huang et al. focused on network latency and computation power [18].

Kumar et al. focused on the edge-cloud computing model for autonomous vehicles [4]. They introduced an ROS-based cloud-assisted system, Carcel, and conducted demonstrated experiments with a golf cart and six iRobot Create robots. Carcel collects data by the UDP connection from the vehicle and sensors placed in multiple locations, analyzes the data, calculates the path using RRT* algorithm [19] at a cloud server, and returns the result to the vehicle. Carcel reduced the average time to detect obstacles such as pedestrians by 4.6 times compared to contemporary systems without access to the cloud. While Carcel requires multiple sensors as roadside infrastructures, our system does not require any additional sensors other than map data and sensors on the vehicle, and hence its use is flexible.

Another distributed model, CVN, is a model where multiple vehicles create a wireless link network to communicate with each other, called VANETs [5]. Through VANETs, a vehicle can share map data and computational resources to improve vehicle infrastructures, or a vehicle can share its positions to other vehicles to improve inter-vehicle communication and safety of passengers. Since VANETs require multiple vehicles, they are difficult to use when non-autonomous vehicles are involved, especially when a heavy vehicle between vehicles leads to frequent disconnection problems called shadowing [20]. Unlike VANETs, this paper focuses on improving standalone autonomous vehicles by utilizing scalable cloud servers to compute heavy tasks.

VI. CONCLUSION

We presented the edge-cloud computing model for autonomous vehicles. The presented edge-cloud computing model assigns ROS components to edge devices or cloud servers, respectively, in order to satisfy the requirement of the execution time and communication cost. We use 1 Gbit/s simulated network as an alternative of 5G mobile network. The simulation results by ROSBAG data demonstrate that the presented edge-cloud computing model reduces the execution time and deadline miss ratio compared to the edge computing model and slightly increases them compared to the cloud computing model.

In future work, we will investigate the behavior of the presented edge-cloud computing model in actual 5G mobile network, especially focusing on the schedulability analysis considering communication delay and losses that would occur under mobile networks are desirable. In addition, we will add the cloudlet (micro datacenters in close to proximity to edge devices) [21] to the presented edge-cloud computing model for more realistic use cases.

REFERENCES

- [1] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, and M. N. C. et al., "Autonomous Driving in Urban Environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.
- [2] B. Fildes, M. Keall, N. Bos, A. Lie, Y. Page, C. Pastor, L. Pennisi, M. Rizzi, P. Thomas, and C. Tingvall, "Effectiveness of low speed autonomous emergency braking in real-world rear-end crashes," *Accident Analysis & Prevention*, vol. 81, pp. 24–29, 2015.
- [3] SAE International, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," SAE International, Warrendale, PA, Standard, 2014.
- [4] S. Kumar, S. Gollakota, and D. Katabi, "A Cloud-Assisted Design for Autonomous Driving," in *Proceedings of the First Edition of the MCC workshop on Mobile Cloud Computing*. ACM, 2012, pp. 41–46.
- [5] H. Hartenstein and K. P. Laberteaux, *VANET: Vehicular Applications and Inter-Networking Technologies*. John Wiley & Sons, 2010.
- [6] International Telecommunications Union, "Minimum requirements related to technical performance for IMT-2020 radio interface(s)," International Telecommunications Union, Geneva, Switzerland, Tech. Rep., 2017.
- [7] —, "Requirements related to technical performance for IMT-Advanced radio interface(s)," International Telecommunications Union, Geneva, Switzerland, Tech. Rep., 2008.
- [8] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.
- [9] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An Open Approach to Autonomous Vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, 2015.
- [10] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *Proceedings of the International Conference on Robotics and Automation Workshop on Open Source Software Workshop on Open Source Software*, vol. 3, 2009, pp. 1–6.
- [11] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monroy, T. Ando, Y. Fujii, and T. Azumi, "Autaware on Board: Enabling Autonomous Vehicles with Embedded Systems," in *Proceedings of the 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems*. IEEE, 2018, pp. 287–296.
- [12] H. Chishiro, K. Suito, T. Ito, S. Maeda, T. Azumi, K. Funaoka, and S. Kato, "Towards Heterogeneous Computing Platforms for Autonomous Driving," in *Proceedings of the 15th IEEE International Conference on Embedded Software and Systems*. IEEE, 2019, pp. 1–8.
- [13] The Autaware Foundation, <https://www.autaware.org/>.
- [14] P. Biber and W. Strasser, "The Normal Distributions Transform: A New Approach to Laser Scan Matching," in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2003, pp. 2743–2748.
- [15] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [16] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.
- [17] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.
- [18] D. Huang, P. Wang, and D. Niyato, "A Dynamic Offloading Algorithm for Mobile Computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.
- [19] S. Karaman and E. Frazzolis, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
- [20] M. Boban, T. T. V. Vinhoza, M. Ferreira, J. Barros, and O. K. Tonguz, "Impact of Vehicles as Obstacles in Vehicular Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 15–28, 2011.
- [21] M. Satyanarayanan, "The Emergence of Edge Computing," *IEEE Computer*, vol. 50, no. 1, pp. 30–39, 2017.



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session III

Planning & Navigation

- **Keynote speaker: Danwei Wang (NTU, Singapore)**
Title: Intelligent Perception, Navigation and Control for Multi-robot Systems
- **Title: miniSAM: A Flexible Factor Graph Non-linear Least Squares Optimization Framework**
Authors: J. Dong, Z. Lv
- **Title: Linear Camera Velocities and Point Feature Depth Estimation Using Unknown Input Observer**
Authors: R. Benyoucef, L. Nehaoua, H. Hadj-Abdelkader, H. Arioui



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session III

Keynote speaker: : **Danwei Wang**
(NTU, Singapore)

Intelligent Perception, Navigation and Control for Multi-robot Systems

Abstract: While tremendous progress have been made in the development of localization and navigation algorithms for single robot, the operation of the multi-robot systems has recently garnered significant attention. This talk aim to report recent advancements in multi-robot systems research, which are developed by Prof Wang Danwei's Group at Nanyang Technological University, Singapore. Emphases are placed on intelligent perception, navigation and control technologies that enable autonomous systems to operate in cluttered and GPS-denied environments. The talk will introduce a systematic multi-robot framework that contains core functions such as multi-sensor data fusion, complex scene understanding, multi-robot localization and mapping, moving object reasoning, and formation control.

Biography: WANG, Danwei received and his Ph.D and M.E.S. degrees from the University of Michigan, Ann Arbor, USA, in 1989 and 1986, respectively, and his B.E. degree from the South China University of Technology, China, in 1982. Currently, he is a professor, School of Electrical and Electronic Engineering, NTU, Director, ST Engineering-NTU Corporate Lab. He also served as Director, EXQUISITUS, Centre for E-City, and Deputy Director of the Robotics Research Centre. From 2005 to 2011, he served as Head, Division of Control and Instrumentation, a member of School Management Committee and a senator in NTU academic council. He is also a facilitated faculty member in Singapore Arm Force - Nanyang Technological University Academy (SNA).


He was awarded the Alexander von Humboldt Fellowship in Germany from 1996-1997. He is an Associate Editor for the International Journal of Humanoid Robotics since 2003, Member, Editorial Board, International Journal of Vehicular and Autonomous Systems and Advisor for



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

ASME Book Series in Robotics Engineering. He has served as guest editors for various journals, including Journal of Field Robotics. He has been invited to deliver keynote speeches in 9 international conferences and workshops and he was awarded the best conference paper award or in the final list of best conference paper award. He has served in the International Program Committees for IEEE IROS 2006 (Technical Co-Chair) and many recent IEEE IROSs and IEEE ICRA as Associate Editor. He is a senior member of IEEE and active in conference organization (as general chair, technical chair, keynote chair, etc).


He is actively involved in teaching and research related to robotics and control. He has developed and taught postgraduate and undergraduate courses on engineering mathematics, control engineering and robotics. As at Oct 2013, he has completed research projects with total fund of 13million SGD (USD/SGD=1.25) and has on-going research project fund of 30million SGD. Dr Wang has been working on robotics/control since 1985. He is a recognized expert on iterative learning/repetitive control theory and applications. He has been working on outdoor mobile robotics since 1995 and was a team leader in a collaboration project with PSA. He led two teams of professors and researchers to pass the qualifying rounds and participate in the Finals of both TechX Challenge 2008 and 2013, respectively, which are outdoor autonomous robot competitions. He was instrumental in the developments of two laboratories for outdoor mobile robots and intelligent robotics in the school of EEE. In recent years, he has successfully developed a framework of fault diagnosis and isolation for complex and hybrid systems. He has also made substantial contributions to the field of satellite formation flying and satellite attitude fault tolerant control. He is the supervisor for 29 Ph.D theses, 38 Master theses and 27 post-doctorates/visiting researchers. He has published 4 books, 7 book chapters, 6 patents (5 published and 1 filed) and over 400 technical papers and articles in international refereed journals and conferences. His research interests include a wide range of topics: robotic manipulators and force control, advanced control design, intelligent systems, learning control, mobile robotics, mobile robot path and trajectory control, satellite formation flying and fault tolerant attitude control, fault diagnosis and prognosis for complex systems, traffic light control. SCI citations to his papers amount to 3614 as of Feb 2017.



**NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE**

**Intelligent Perception, Navigation and
Control for Multi-robot Systems**

Prof Danwei Wang
School of EEE
Nanyang Technological University
Singapore
Email: edwwang@ntu.edu.sg



Outline

- 1. Overview**
- 2. Multi-Modal Environmental Perception**
- 3. Collaborative Uniqueness Reasoning**
- 4. Collaborative Localization and Mapping**
- 5. Conclusions**



The challenge of unstructured environment

- **Communication:** constrained communication and no GPS
- **Physical Environment:** Highly dynamic and unstructured

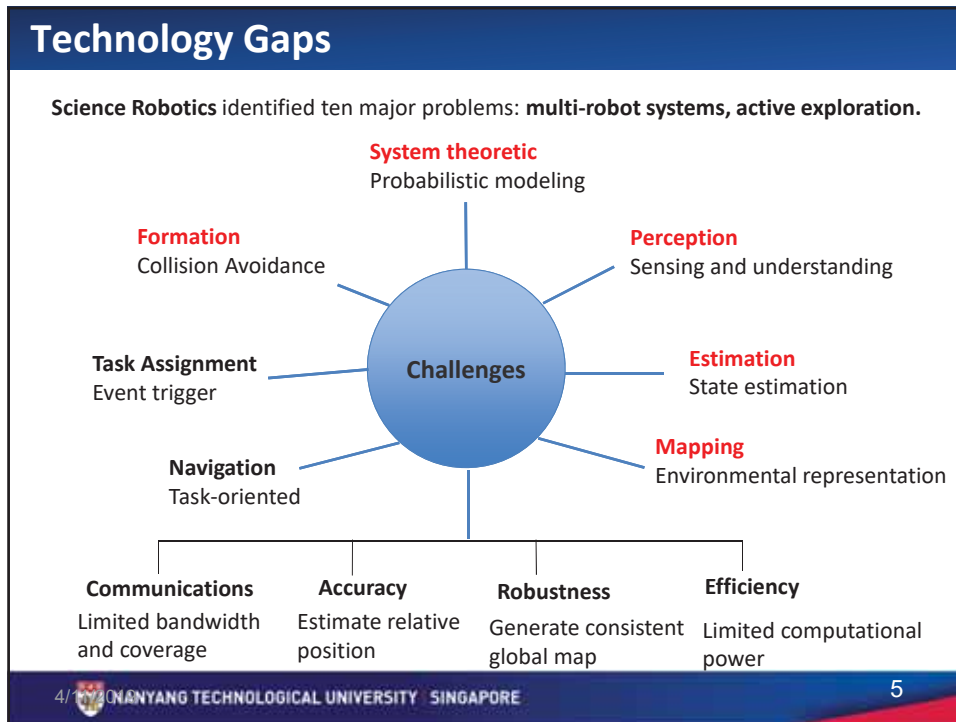
All-terrain, all-weather, all-platform and all-sensor

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
3

Research Project

➤ National Research Foundation Singapore Project of "Multi-Robot Systems in Complex Environments"

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
4

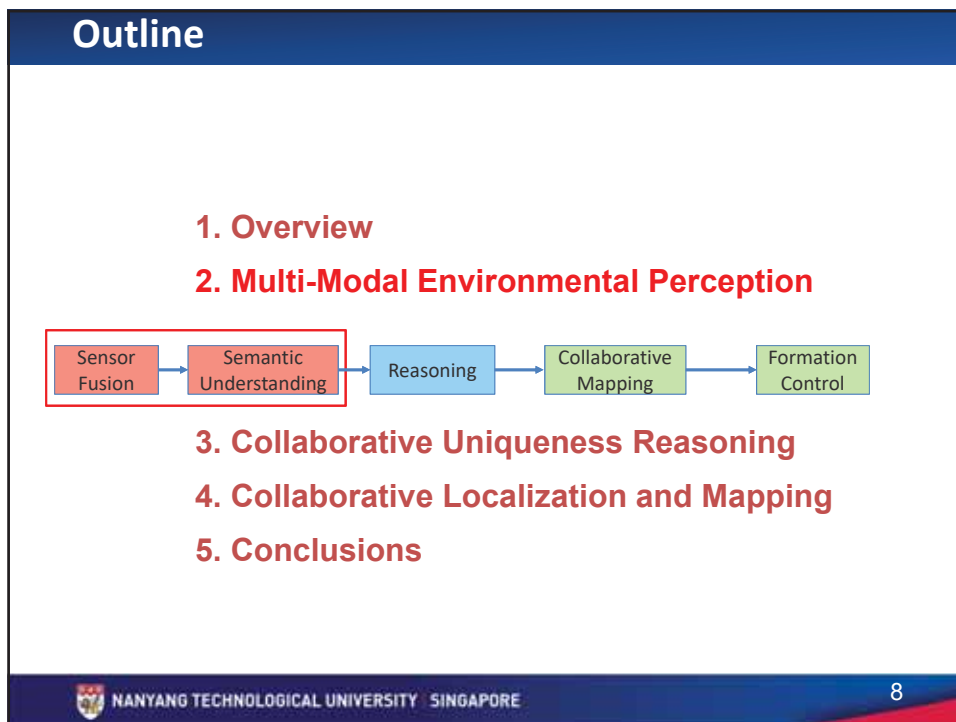
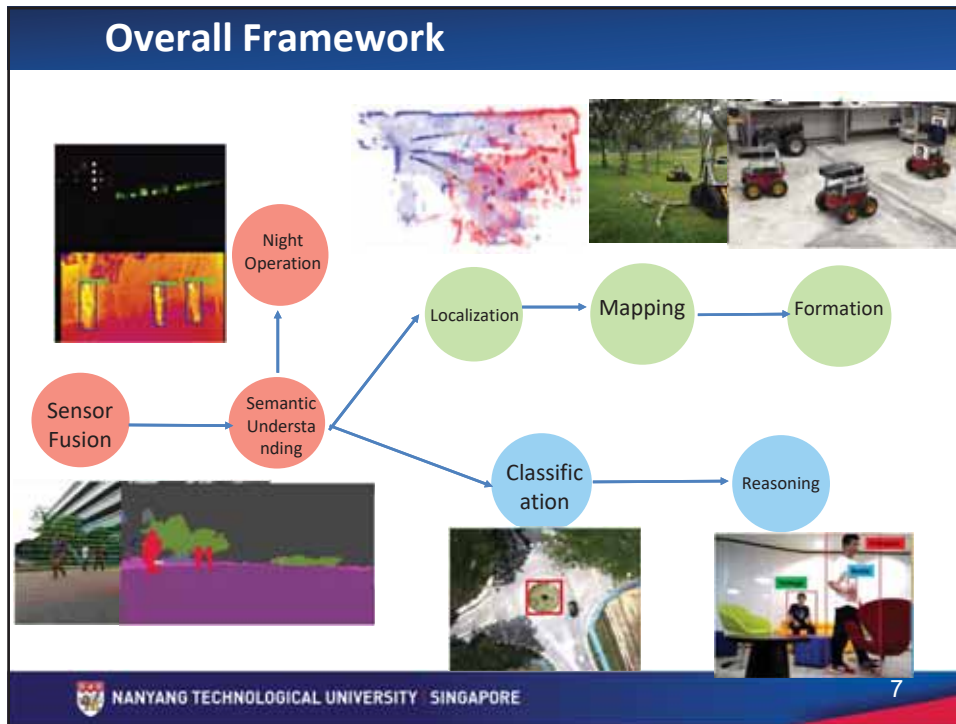


Objective of our research

- **Objective:**
 - Beyond the autonomy of single robot, operate multi-robots in GPS-Denied unstructured and dynamic environment :
 - **Multi-Modal Information Fusion:** provide a comprehensive understanding of the complicated surroundings.
 - **Dynamic Reasoning:** multi-robot systems could collaboratively analyze the dynamics.
 - **Collaborative Mapping and Formation:** multi-robot systems need to achieve precise localization, collaborative mapping and formation.

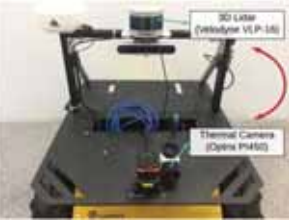


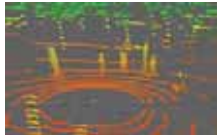



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE 6




Multi-Modal Environmental Perception

- Introduction
 - Each sensor has its limitation and strength

visual camera
thermal camera
3D LiDAR

- Improve the perception capability of mobile robots under poor lighting conditions (heterogeneous sensors fusion)
- Calibration is necessary
 - Build the correspondences between different sensors
 - **No method:** a sparse 3D LiDAR and a thermal camera


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
9

Multi-Modal Environmental Perception

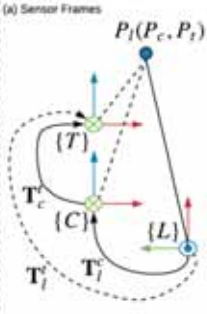
Objective

To obtain $T_l^t = \begin{bmatrix} R_l^t & T_l^t \\ \mathbf{0} & 1 \end{bmatrix}$ between a sparse 3D LiDAR $\{L\}$ and a thermal camera $\{T\}$

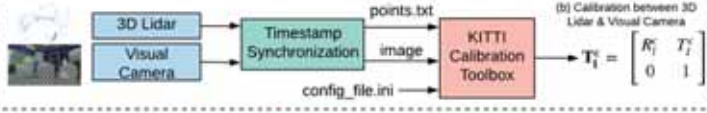
Challenge: extract common features from sparse point cloud and thermal image is **difficult**

- To obtain T_l^t , a two-step method is proposed
 - A visual camera is introduced to assist the process

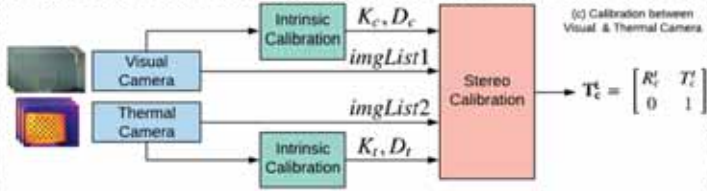
(a) Sensor Frames




(b) Calibration between 3D Lidar & Visual Camera



(c) Calibration between Visual & Thermal Camera



$$T_l^t = T_c^t T_l^c$$


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
10

Multi-Modal Environmental Perception

Real World Experiment

Classification

Semantic

Sensor Fusion

YOLO v3

Calibration

Sensor Fusion

Filtered Static Points

Person 1 Person 2 Person 3

Human Stream

- Enhance the perception to the three-dimensional semantic level, enabling dual-threaded processing of dynamic object and static observation, providing day and night all-weather dynamic scene sensing capabilities.

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

11

Multi-Modal Environmental Perception

Night Environment Perception

Person 1 Person 2 Person 3

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

12

Outline

1. Overview
2. Multi-Modal Environmental Perception
3. Collaborative Uniqueness Reasoning

```

graph LR
    A[Sensor Fusion] --> B[Semantic Understanding]
    B --> C[Reasoning]
    C --> D[Collaborative Mapping]
    D --> E[Formation Control]
    style C stroke:#f00,stroke-width:2px
    
```

4. Collaborative Localization and Mapping
5. Conclusions


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

13


Single Robot Uniqueness Reasoning

Background


- What is **Uniqueness Reasoning**: It is a cognitive and logical process for identifying the unique individual in the environment.
 - ① the target behaves differently from others
 - ② specific target in respective mission
- Why is **Uniqueness Reasoning** important?



Mobile Surveillance



Robot Police



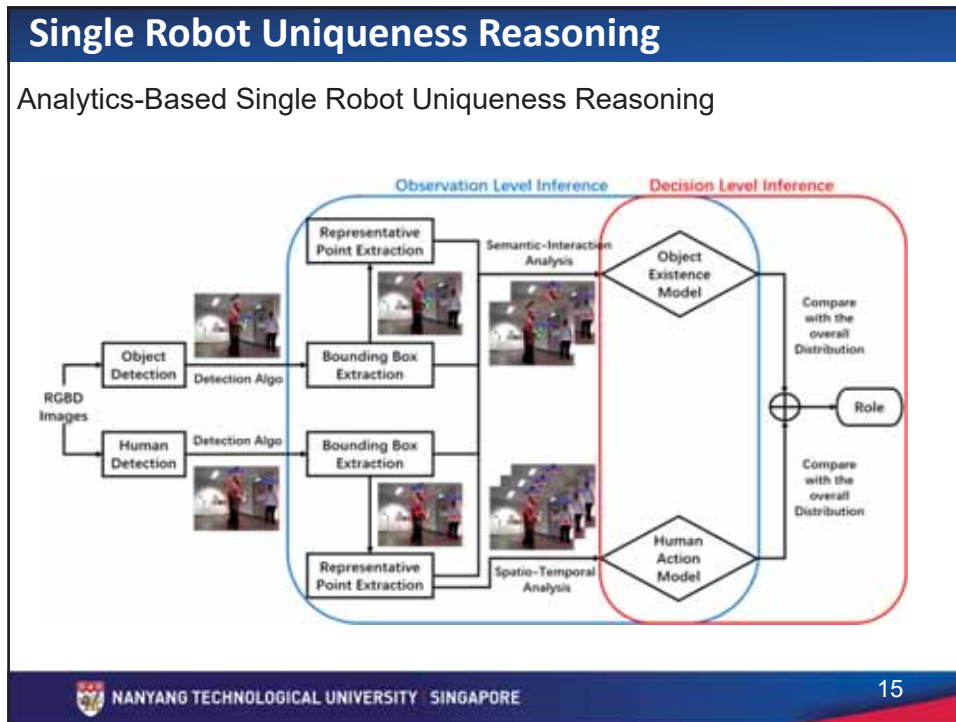
Car Accident

Huge Safety and Security Risks and Demands !

Chule Yang, Yufeng Yue, and Danwei Wang, "Probabilistic Reasoning for Unique Role Recognition Based on the Fusion of Semantic-Interaction and Spatio-Temporal Features", IEEE Transactions On Multimedia, 2019

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

14



Single Robot Uniqueness Reasoning

➤ **Uniqueness Reasoning Formulation**

On the basis of the Bayes' theorem, the uniqueness I of each person j at time t can be decomposed as below:

$$P_j(I_t | \Theta_t^{SI}, \Theta_t^{ST}, g_t^o, g_t^h, g_{t-1}^h) = \underbrace{P_j(I_t | \Theta_t^{SI}, \Theta_t^{ST})}_{\text{Spatial-Semantic Inference}} \cdot \underbrace{P_j(\Theta_t^{SI}, \Theta_t^{ST} | g_t^o, g_t^h, g_{t-1}^h)}_{\text{High-Level Feature Extraction}}$$

- θ^{SI} Semantic-Interaction Feature

Negative

$S_p^o \cap S_p^h = \emptyset, \text{ and } |c^h - c^o| > \epsilon$

Negative

$S_p^o \cap S_p^h = \emptyset$

Positive

$S_p^o \cap S_p^h \neq \emptyset, \text{ and } |c^h - c^o| < \epsilon$
- θ^{ST} Spatiotemporal Feature

Time Step t-1

Time Step t

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE 16

Single Robot Uniqueness Reasoning

➤ Real Environment Experiment

Table 5.5: Evaluation of uniqueness reasoning in three different scenarios (in %).

Decision Method	Input Data	Scenario 1		Scenario 2		Scenario 3		Time ($10^{-4}s$)
		Precision	Recall	Precision	Recall	Precision	Recall	
d_{2D} (OEM)	RGB Image	74.63	27.03	87.22	88.90	97.05	91.86	2.00
STA (HAM)	RGB Image	63.10	53.00	78.96	76.12	27.40	14.91	6.00
MIF (OEM+HAM)	RGB Image	92.17	81.82	88.85	89.14	98.21	99.79	6.80

17

Multi-Robot Uniqueness Reasoning

Analytics-Based Multi-Robot Uniqueness Reasoning

- **Motivation**
 - Single robot can not practically handling occlusion, sensor failure, limited field of view and varying illumination.
 - No existing literature studies the problem of uniqueness reasoning by using multi-robot system.

Chule Yang, Danwei Wang, Yufeng Yue, and Prarinya Siritanawan, "Multimodal Information Fusion for Human-Oriented Context Awareness Using Mobile Robot", Information Fusion, 2019

18

Multi-Robot Uniqueness Reasoning

➤ Real Environment Experiment

Table 6.3: Human uniqueness reasoning results from the single robot and multi-robot system. (The best performance is denoted in bold.)

Experiments	Robot 1 Reasoning		Robot 2 Reasoning		Collaborative Reasoning	
	Precision	Recall	Precision	Recall	Precision	Recall
Daytime Rainforest	85.04	82.76	66.97	53.28	86.24	88.85
Night Forest	89.47	88.54	-	-	89.47	88.54
Open Carpark	59.40	54.86	57.65	52.64	64.88	61.57

19

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Outline

1. Overview
2. Multi-Modal Environmental Perception
3. Collaborative Uniqueness Reasoning
4. Collaborative Localization and Mapping


5. Conclusions

20


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Multi-Robot Localization and Mapping


- **Motivation:**



• Large environment mapping




• Outer space exploration



• Rescue after an earthquake

 - In multi-robot systems, maintaining a **comprehensive understanding** of the environment is a fundamental requirement.
 - Mapping: provide global perception ability
 - Localization: multi-robot localization without GPS
 - Planning: multi-robot path planning in unknown environment

1. Yufeng Yue, P.G.C.N. Senarathne, Chule Yang, Jun Zhang, Mingxing Wen, and Danwei Wang, "Hierarchical Probabilistic Fusion Framework for Matching and Merging of 3D Occupancy Maps", *IEEE Sensors Journal*, 2018.
 2. Yufeng Yue, Chule Yang, P. G. C. N. Senarathne, and Danwei Wang, "A General Multi-level Framework for Distributed Multi-Robot 3-D Map Fusion using Heterogeneous Sensors Under Constrained Environment", *IEEE Systems Journal*, 2019



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

21


Multi-Robot Localization and Mapping

Challenge

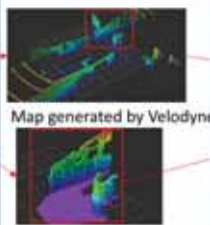
- For a group of robots in **arbitrary** (structured/semi-structured/**unstructured**) **environment**, maps generated with heterogeneous sensors need to be fused.

➡


- Maps generated by heterogeneous sensors have different viewing angles, range and densities, a flexible data association strategy is needed.



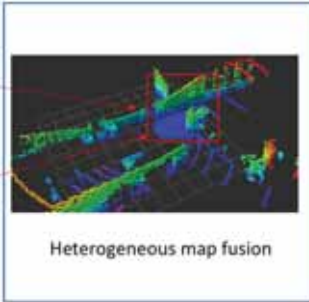
Robot platform




Map generated by Velodyne



Map generated by RGB-D camera



Heterogeneous map fusion



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

22

Multi-Robot Localization and Mapping

1.Feature Extraction
→
2.Single robot SLAM
→
3.Exploration
→
4.Path planning
→
5.Collaborative mapping

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
23

Multi-Robot Localization and Mapping

General probabilistic framework

A multi-level framework allows for **joint local and global** map matching and merging, which should act as the **theory basis** for multi-robot equipped with **heterogeneous sensors**.

Problem Formulation

The problem of collaborative mapping is joint estimation of relative transformation (map matching) and generating the global map given all partial maps (map merging).

$$p(M_r^{\mathcal{R}}, T_{r, \mathcal{R}} | m_r, m_{\mathcal{R}}) \propto \underbrace{p(T_{r, \mathcal{R}} | m_r, m_{\mathcal{R}})}_{\text{map matching}} \cdot \underbrace{p(M_r^{\mathcal{R}} | T_{r, \mathcal{R}}, m_r, m_{\mathcal{R}})}_{\text{map merging}}$$

- Flexibility to heterogeneous sensors
- Uncertainty propagation

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
24

Multi-Robot Localization and Mapping

Overall Process

- A general collaborative mapping probabilistic scheme is proposed that provides flexibility for different types of sensors and SLAM methods.
- It allows for joint estimating relative transformation between all robots and integrating probabilistic global map.

1. Single robot SLAM
2. Distributed map sharing
3. Probabilistic map matching
4. Time sequential map merging

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
25

Multi-Robot Localization and Mapping

Experiment

- At $t=0s$, probabilistic data association strategy successfully matches the maps.


- At the end of the mission, the fused local map and global map.

Perception Level	
Local Map Level	
Global Map Level	


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
26

Multi-Robot Localization and Mapping


Forest Environment



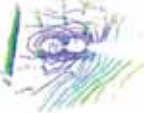
Environment




Robot 1 Local Map




Robot 2 Local Map




Fused Global Map




Environment




Robot 1 Local Map




Robot 2 Local Map




Fused Global Map




Environment



Robot 1 Local Map




Robot 2 Local Map



Fused Global Map

- Fully unstructured environment
- Successfully fused from start position
- Global consistent map




NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE


27

Multi-Robot Localization and Mapping

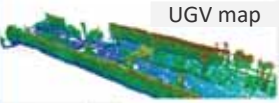
UAV-UGV Collaborative Mapping




UGV




UGV data



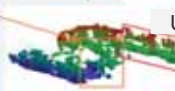
UGV map




UAV



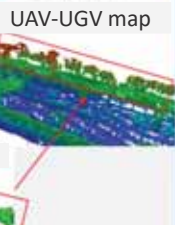
UAV data



UAV map




+



UAV-UGV map

- Different modality of sensor output
- Large difference of field of view

➔ Successfully produces global consistent map



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

28

Multi-Robot Localization and Mapping

Overall Accuracy

- Our algorithm outperforms other algorithms in most of the cases.

Dataset	ICP		P2P ICP		GICP		NDT		First PMM		Full PMM	
	Euler	Meter	Euler	Meter	Euler	Meter	Euler	Meter	Euler	Meter	Euler	Meter
Corridor	31.39	1.63	19.97	1.51	10.58	1.71	13.63	2.56	12.06	0.62	9.37	0.57
WKW	5.77	1.95	6.31	1.78	5.32	2.84	5.51	2.32	6.90	1.47	5.60	1.34
Willow	6.54	3.60	4.97	2.78	5.28	2.72	8.07	5.13	8.88	2.54	4.99	2.00

Tab. Mean error(Degree, meter; best performance in bold.)

Overall Efficiency

- Computation time of our algorithm is compared against the ICP, P2P ICP, GICP and NDT
- High efficiency of our algorithm comes from the use of the two-level structure

Datasets	ICP	P2P ICP	GICP	NDT	Full PMM
Corridor	3.14	2.90	24.08	136.42	2.13
WKW	10.35	8.13	46.66	233.33	5.31
Willow	5.59	4.47	40.27	217.38	3.83

Tab. The efficiency comparison of the algorithms (Seconds)

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
29

Multi-Robot Localization and Mapping

Overall Uncertainty

- The size of the map is decreased compared with directly stitching.
- Combing probabilities of both maps effectively to decrease the uncertainty.

	Directly Stitch		Taking Average		Relative Entropy Filter	
	Size	Entropy	Size	Entropy	Size	Entropy
Corridor Dataset	40763	0.2061	32787	0.2141	32787	0.2002
WKW Dataset	101364	0.2374	82765	0.2371	82765	0.2317
Willow Dataset	95596	0.1583	75442	0.1680	75442	0.1526

Data Transmission

- Sharing 3D map will significantly reduce the size of data to be shared.


(a) Size of Data in Daytime Rainforest.

(b) Size of Data in Night Forest.

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
30

Virtual-Structure Formation Tracking


- **Motivation**
 - Virtual-structure formation tracking can be used in applications such as sensitive area monitoring, unknown environment exploration, etc.
 - Map information is not available
 - Most of the existing approaches assumes that map information is known in advance.




Virtual-Structure Formation Tracking

- **Objectives**
 - Formation tracking
 - Avoidance Control
 - Inter-Robot Collision Avoidance
 - Obstacle Avoidance
 - Connectivity Maintenance
- **Assumptions**
 - With communication
 - No map information

Yuanzhe Wang, Mao Shan, Yufeng Yue, and Danwei Wang, "Virtual-Structure Formation Analysis and Tracking Control of Multiple Nonholonomic Mobile Robots", *IEEE Transactions on Control Systems Technology*, 2019



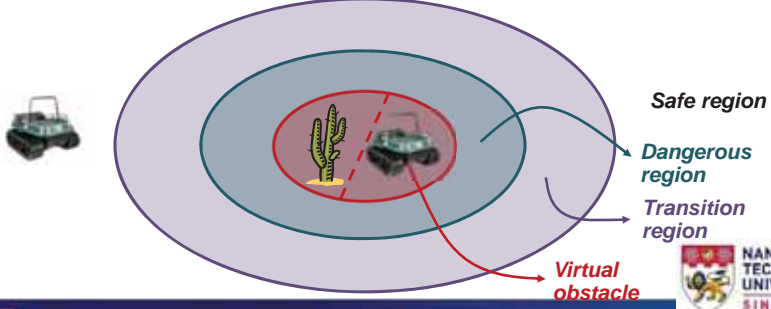
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE




31


Virtual-Structure Formation Tracking

- **Modeling of Obstacle / Inter-Robot Collision Avoidance**
 - Virtual obstacle: A circle centered at the nearest obstacle point / the neighboring robot.
 - Safe region, robot does not care the obstacle.
 - Dangerous region, avoidance control holds the highest priority.
 - Transition region, the control input is the weighted combination of formation tracking and avoidance control.





NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE



32

Virtual-Structure Formation Tracking

Control Design

- Control Scheme

Controller

$$\mathbf{u}_i = \mathbf{u}_i^t + \mathbf{u}_i^a$$

Safe Region

- Bounded formation tracking

Transition Region

- Weighted combination of formation and avoidance.

Dangerous Region

- Focus on collision avoidance and connectivity maintenance.

$$W = W_o W_n W_c$$

Avoidance Control $W = 0$
Weighted Combination
Formation Tracking $W = 1$

Tracking Part

$$\mathbf{u}_i^t = \hat{\mathbf{p}}_i^d + \mathbf{g}(\tilde{\mathbf{p}}_i)$$

Avoidance Part

$$\mathbf{u}_i^a = -\underline{(1-W)}\mathbf{u}_i^t - k_i \frac{\partial \phi_i}{\partial \mathbf{p}_i}$$


4/11/2019

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

33

Virtual-Structure Formation Tracking

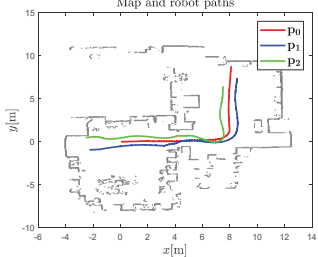
Experiment Results



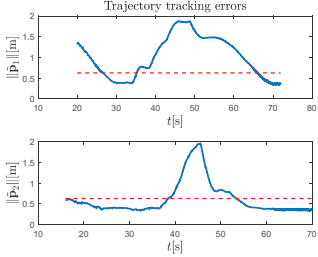
When there are no obstacles around, bounded formation tracking can be achieved.

Collision avoidance and connectivity maintenance can be guaranteed.

Map and robot paths



Trajectory tracking errors




NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

34

Outline

1. Overview
2. Multi-Modal Environmental Perception
3. Collaborative Uniqueness Reasoning
4. Collaborative Localization and Mapping
5. Conclusions


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
35

Conclusions

Conclusions

Situation Awareness and Intelligent Navigation for Multi-robot Systems

1. Multi-modal information fusion and understanding


2. Multi-Level Probabilistic Uniqueness Reasoning

3. General probabilistic collaborative mapping framework

```

graph LR
    A[Sensor Fusion] --> B[Semantic Understanding]
    B --> C[Reasoning]
    C --> D[Collaborative Mapping]
    D --> E[Formation Control]
    
```

Addressed the problem of perception, reasoning, navigation, formation in GPS-Denied and dynamic environments for multi-robot systems.


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

4/11/2019

Thank you!



Email: edwwang@ntu.edu.sg



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

37



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session III

Planning & Navigation

- **Title: miniSAM: A Flexible Factor Graph Non-linear Least Squares Optimization Framework**
Authors: J. Dong, Z. Lv
- **Title: Linear Camera Velocities and Point Feature Depth Estimation Using Unknown Input Observer**
Authors: R. Benyoucef, L. Nehaoua, H. Hadj-Abdelkader, H. Arioui



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

miniSAM: A Flexible Factor Graph Non-linear Least Squares Optimization Framework

Jing Dong¹ and Zhaoyang Lv²

Abstract—Many problems in computer vision and robotics can be phrased as non-linear least squares optimization problems represented by *factor graphs*, for example, simultaneous localization and mapping (SLAM), structure from motion (SfM), motion planning, and control. We have developed an open-source C++/Python framework *miniSAM*, for solving such factor graph based least squares problems. Compared to most existing frameworks for least squares solvers, *miniSAM* has (1) full Python/NumPy API, which enables more agile development and easy binding with existing Python projects, and (2) a wide list of sparse linear solvers, including CUDA enabled sparse linear solvers. Our benchmarking results shows *miniSAM* offers comparable performances on various types of problems, with more flexible and smoother development experience.

I. INTRODUCTION

Solving non-linear least squares is important to many areas in robotics, including SLAM [1], SfM [2], motion planning [3], and control [4], [5]. Furthermore, researchers in these areas often use *factor graphs*, a probabilistic graphical representation to model the non-linear least squares problem. Dellaert and Kaess [1] first connected factor graphs to non-linear least squares, and the graph inference algorithms to sparse linear algebra algorithms.

There are existing libraries for solving non-linear least squares problems. Existing widely used frameworks by SLAM and SfM communities include Ceres [6], g2o [7], and GTSAM [8]. In particular, GTSAM uses factor graph to model the non-linear least square problems, and solves the problems using graphical algorithms rather than sparse linear algebra algorithms. However, for performance reasons all existing frameworks are implemented in C++ and therefore have the disadvantage that they require complex C++ programming, especially when users merely want to define or customize loss functions.

We introduce a flexible, general and lightweight factor graph optimization framework *miniSAM*[†]. Like GTSAM, *miniSAM* uses factor graphs to model non-linear least square problems. The APIs and implementation of *miniSAM* are heavily inspired and influenced by GTSAM, but *miniSAM* is a much more lightweight framework, and that extends the flexibility of GTSAM as follows:

- Full Python/NumPy API, with the ability to define custom cost functions and optimizable manifolds to

^{*}This work was mostly finished when both authors were PhD students at College of Computing, Georgia Institute of Technology, Atlanta, USA. We would like to thank Prof. Frank Dellaert and Dr. Mustafa Mukadam giving suggestions on this work. This work received no financial support.

¹thu.dongjing@gmail.com

²zhaoyang.lv@gatech.edu

[†]<https://github.com/dongjing3309/minisam>

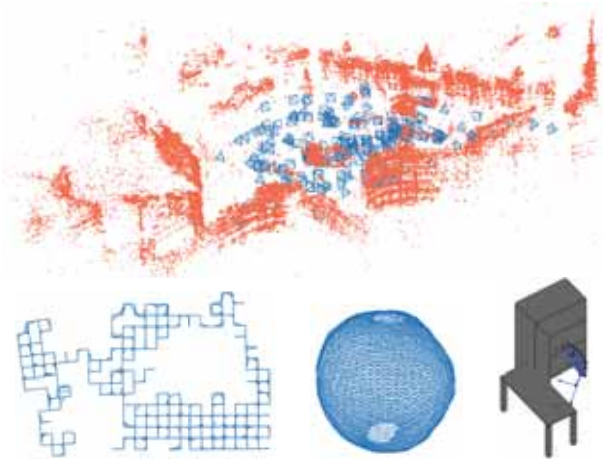


Fig. 1: Example problems solved by *miniSAM*. Top: bundle adjustment problem *Trafalgar* [9], camera poses are shown in blue and landmarks are shown in red. Bottom (from left to right): 2D pose graph problem *M3500* [10], 3D pose graph problem *Sphere* [10], Barrett WAM arm motion planning problem [3].

enable faster and easier prototyping.

- A wide list of sparse linear solver choices, including CUDA supported GPU sparse linear solvers.
- It is lightweight and requires minimal external dependencies, thus making it great for cross-platform compatibility.

In this paper, we first give an introduction to non-linear least squares and the connection between sparse least squares and factor graphs. Then, we introduce the features and basic usage of *miniSAM*, finally we show benchmarking results of *miniSAM* on various SLAM problems.

II. INTRODUCTION TO NON-LINEAR LEAST SQUARES AND FACTOR GRAPHS

A. Non-linear Least Square Optimization

Non-linear least squares optimization is defined by

$$x^* = \operatorname{argmin}_x \sum_i \rho_i(\|f_i(x)\|_{\Sigma_i}^2), \quad (1)$$

where $x \in \mathcal{M}$ is a point on a general n -dimensional manifold, $x^* \in \mathcal{M}$ is the solution, $f_i \in \mathbb{R}^m$ is a m -dimensional vector-valued error function, ρ_i is a robust kernel function, and $\Sigma_i \in \mathbb{R}^{m \times m}$ is a covariance matrix. The Mahalanobis distance is defined by $\|v\|_{\Sigma}^2 \doteq v^T \Sigma^{-1} v$ where $v \in \mathbb{R}^m$ and Σ^{-1} is the information matrix. If we

factorize the information matrix by Cholesky factorization $\Sigma^{-1} = R^T R$, where R is upper triangular, we have

$$\|v\|_{\Sigma}^2 = v^T \Sigma^{-1} v = v^T R^T R v = \|Rv\|^2. \quad (2)$$

If we consider the simplified case where ρ_i is identity and define $h_i(x) \doteq R_i f_i(x)$, then Eq. (1) is equivalent to

$$x^* = \operatorname{argmin}_x \sum_i \|h_i(x)\|^2 \quad (3)$$

as per Eq. (2). If we define a *linearization point* $x_0 \in \mathcal{M}$, and the Jacobian matrix of $h_i(x)$

$$J_i \doteq \left. \frac{\partial h_i(x)}{\partial x} \right|_{x=x_0} \quad (4)$$

then the Taylor expansion is given by

$$h_i(x_0 + \Delta x) = h_i(x_0) + J_i \Delta x + O(\Delta x^2), \quad (5)$$

which we can use to solve the least square problem by searching a *local* region near x_0 , and find the solution by iteratively solving a *linearized* least squares problem

$$\Delta x^* = \operatorname{argmin}_{\Delta x} \sum_i \|J_i \Delta x + h_i(x_0)\|^2, \quad (6)$$

where $\Delta x^* \in \mathbb{R}^n$, and the solution is updated by

$$x^* = x_0 + \Delta x^*. \quad (7)$$

If \mathcal{M} is simply a vector space \mathbb{R}^n then the above procedure is performed iteratively in general by setting x_0 of next iteration from x^* of current iteration, until x^* converges. Trust-region policies like Levenberg-Marquardt can be also applied when looking for Δx^* .

When \mathcal{M} is a general manifold, we need to define a local coordinate chart of \mathcal{M} near x_0 , which is an invertible map between a local region of \mathcal{M} around x_0 and the local Euclidean space, and also an operator \oplus that maps a point in local Euclidean space back to \mathcal{M} . Thus Eq. (7) on general manifolds is

$$x^* = x_0 \oplus \Delta x^*. \quad (8)$$

A simple example of \oplus is for the Euclidean space where it is simply the plus operator.

To solve the linear least squares problem in Eq. (6), we first rewrite Eq. (6) as

$$\Delta x^* = \operatorname{argmax}_{\Delta x} \|J \Delta x + b\|^2, \quad (9)$$

where J is defined by stacking all J_i vertically, similarly b is defined by stacking all $h_i(x_0)$ vertically. Cholesky factorization is commonly used solve Eq. (9). Since the solution of linear least squares problem in Eq. (9) is given by the normal equation

$$J^T J \Delta x^* = J^T b, \quad (10)$$

we apply Cholesky factorization to symmetric $J^T J$, and we have $J^T J = R^T R$ where R is upper triangular. Then solving Eq. (10) is equivalent to solving both

$$R^T y = J^T b \quad (11)$$

$$R \Delta x^* = y \quad (12)$$

in two steps, which can be both solved by back-substitution given that R is triangular. Other than Cholesky factorization, QR and SVD factorizations can be also used to solve Eq. (9), although with significantly slower speeds. Iterative methods like pre-conditioned conjugate gradient (PCG) are also widely used to solve Eq. (10), especially when $J^T J$ is very large.

B. Connection between Factor Graphs and Sparse Least Squares

Dellaert and Kaess [1] have shown factor graphs have a tight connections with non-linear least square problems. A factor graph is a probabilistic graphical model, which represents a joint probability distribution of all factors

$$p(x) \propto \prod_i p_i(x_i), \quad (13)$$

where $x_i \subseteq x$ is a subset of variables involved in factor i , $p(x)$ is the overall distribution of the factor graph, and $p_i(x_i)$ is the distribution of each factor. The maximum a posteriori (MAP) estimate of the graph is

$$x^* = \operatorname{argmax}_x p(x) = \operatorname{argmax}_x \prod_i p_i(x_i). \quad (14)$$

If we consider the case where each factor has Gaussian distribution on $f_i(x_i)$ with covariance Σ_i ,

$$p_i(x_i) \propto \exp\left(-\frac{1}{2} \|f_i(x_i)\|_{\Sigma_i}^2\right), \quad (15)$$

then MAP inference is

$$\begin{aligned} x^* &= \operatorname{argmax}_x \prod_i p_i(x_i) = \operatorname{argmax}_x \log\left(\prod_i p_i(x_i)\right), \quad (16) \\ &= \operatorname{argmin}_x \prod_i -\log(p_i(x_i)) = \operatorname{argmin}_x \sum_i \|f_i(x_i)\|_{\Sigma_i}^2. \quad (17) \end{aligned}$$

The MAP inference problem in Eq. (17) is converted to the same non-linear least squares optimization problem in Eq. 1, which can be solved following the same steps in Section II-A.

There are several advantages of using factor graph to model the non-linear least squares problem in SLAM. Factor graphs encode the probabilistic nature of the problem, and easily visualize the underlying sparsity of most SLAM problems since for most (if not all) factors x_i are very small sets. We give an example in the next section, which clearly visualizes this sparsity in a factor graph.

C. Example: A Pose Graph

Here we give a simple example of using factor graph to solve a small pose graph problem. The problem is shown in Fig. 2a, where a vehicle moves forward on a 2D plane, makes a 270 degrees right turn, and has a relative pose loop closure measurement which is shown in red. If we want to estimate the vehicle's poses at times $t = 1, 2, 3, 4, 5$, we define the system's state variables

$$x = \{x_1, x_2, x_3, x_4, x_5\}, \quad (18)$$

loss functions have been implemented in miniSAM). In the pose graph example two types of factors are used: unary `PriorFactor` and binary `BetweenFactor`.

In the second step we provide the initial variable values as the linearization point. In miniSAM variable values are stored in structure `Variables`, where each variable is indexed by its key. Finally, we call a non-linear least square solver (like Levenberg-Marquardt) to solve the problem. Result variables are returned in a `Variables` structure with status code.

B. Define Factors

Here we discuss how to define a new factor in miniSAM. As mentioned defining a new factor can be done in both C++ and Python in miniSAM, by inheriting from `Factor` base class. The implementation of a factor class includes an error function `error()` that defines $f_i(x_i)$, which returns a `Eigen::VectorXd` in C++, or a NumPy array in Python. And Jacobian matrices function `jacobians()` that defines $\partial f_i(x_i)/\partial x_i$ for each variable in x_i , which return a `std::vector<Eigen::MatrixXd>` in C++, or a list of NumPy matrices in Python. We show an example prior factor on $SE(2)$ in Python in Snippet 2.

Analytic Jacobians $\partial f_i(x_i)/\partial x_i$ is usually quite complex for non-trivial factors, and is the main bottleneck for faster prototyping. miniSAM provides a solution by inheriting from `NumericalFactor` base class, numerical $\partial f_i(x_i)/\partial x_i$ through finite differencing will be evaluated during optimization, thus saving developer’s time deriving analytic Jacobians. We leave automatic differentiation for Jacobian evaluation as future work.

C. Define Optimizable Manifolds

miniSAM already has build-in support for optimizing various commonly used manifold types in C++ and Python, including Eigen vector types in C++, NumPy array in Python, and Lie groups $SO(2)$, $SE(2)$, $SO(3)$, $SE(3)$ and $Sim(3)$ (implementations provided by Sophus library [13]), which are commonly used in SLAM and robotics problems.

We can also customize manifold properties of any C++ or Python class for miniSAM. In Python this is done by defining manifold-related member functions, including `dim()` function returns manifold dimensionality, and `local()` and `retract()` functions defines the local coordinate chart. An example of defining a vector space manifold \mathbb{R}^2 in Python is in Snippet 3. In C++ we use a non-intrusive technique called *traits*, which is a specialization of template `minisam::traits<>` for the type we are adding manifold properties. Using traits to define manifold properties has two advantages: (1) optimizing a class without modifying it, or even without knowing details of implementation (e.g. adding miniSAM optimization support for third-party C/C++ types), (2) making optimizing primitive type (like float/double) possible.

IV. EXPERIMENTS

To test the performance of miniSAM, we run a benchmark on multiple problems of different types and scales, and

TABLE II: Optimization times in second of different frameworks with different sparse linear solvers, grouped by single-thread or multi-thread.

	2D-PG	3D-PG	BA
Ceres + Eigen LDLT	0.090	2.735	54.96
g2o + Eigen LDLT	0.059	2.697	63.66
GTSAM + Multifrontal Cholesky	0.228	2.002	83.67
GTSAM + Sequential Cholesky	0.207	2.836	83.85
miniSAM + Eigen LDLT	0.088	3.341	64.38
Ceres + CHOLMOD	0.080	0.941	28.17
g2o + CHOLMOD	0.064	0.821	35.68
miniSAM + CHOLMOD	0.090	1.107	39.24
miniSAM + cuSOLVER Cholesky	0.458	1.791	49.77

compare with multiple existing frameworks. We choose three SLAM and SfM problem for benchmarking, from small to large.

- 2D pose graph problem `M3500` [10], which contains 3500 2D poses and 5453 energy edges.
- 3D pose graph problem `Torus` [10], which contains 5000 3D poses and 9048 energy edges.
- Bundle adjustment problem `Dubrovnik` [9], which contains 356 camera poses, 226730 landmarks and 1255268 image measurements.

For all problems we use Levenberg-Marquardt algorithm to solve, and fix the number of iterations to 5.

We run the benchmark with the following frameworks and sparse linear solvers

- Ceres [6] with Eigen simplicial LDLT solver, and CHOLMOD [14] Cholesky solver.
- g2o [7] with Eigen simplicial LDLT solver, and CHOLMOD Cholesky solver.
- GTSAM [8] with built-in multi-frontal and sequential graph elimination solvers.
- miniSAM with Eigen simplicial LDLT solver, CHOLMOD Cholesky solver, and CUDA cuSOLVER GPGPU Cholesky solver.

For miniSAM, all factors and manifolds are implemented natively in C++. All frameworks in benchmarking are compiled in single-thread, except CHOLMOD and CUDA cuSOLVER solvers are compiled in multi-thread (using all 12 available CPU threads during benchmarking, and GPU is used with CUDA). The benchmarking is performed on a computer with Intel Core i7-6850K CPU, 128 GB memory, and a NVIDIA TITAN X GPU with 12GB graphic memory. The results are shown in Table. II, and are grouped by single-thread or multi-thread.

We can see in Table. II that when the same sparse linear solver is used, miniSAM has slightly worse runtime compare to Ceres and g2o, but (except for 3D pose graph case) has better runtime compared to GTSAM, which does not use third-party sparse linear solvers. The extra overhead of miniSAM compare to Ceres and g2o are mainly due to two major miniSAM design choices:

- miniSAM avoids using any compile-time array or matrix, and all internal vectors and matrices are dynam-

ically allocated. The use of dynamic size arrays involves extra memory allocation overhead and forbids any compile-time optimization by modern CPU SIMD instructions.

- miniSAM avoids using any raw pointer and manual memory management.

The reason to make above design choices is that to make miniSAM have a Python API consistent with C++ API, and to make Python interface possible to implement, since Python does not have machinery to support template programming or explicit memory management.

We also found CUDA cuSOLVER is not as fast as CHOLMOD CPU solver when using all 12 available CPU threads, and it is particularly slow on small problems. Finally, CUDA cuSOLVER has an one-time launch delay of about 350ms, once per executable launch. Given such circumstances using CUDA cuSOLVER is currently only good for large problems.

V. CONCLUSION

We gave a brief introduction to miniSAM, our non-linear least squares optimization library. We demonstrate the basic usage of miniSAM, show its flexibility in fast prototyping in Python, and its performance in benchmarking of multiple types of problems in SLAM and robotics applications. We recognize miniSAM has a relatively small performance loss compared to other state-of-the-art frameworks, mostly due to miniSAM's design to adapt Python API, so currently miniSAM is not great for performance-critical applications. But hopefully we can solve the problem in the future by porting better sparse linear solvers (like GPU-enabled iterative solver) to mitigate this issue.

REFERENCES

- [1] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, pp. 1181–1203, Dec 2006.
- [2] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment – a modern synthesis," in *Vision Algorithms: Theory and Practice* (W. Triggs, A. Zisserman, and R. Szeliski, eds.), vol. 1883 of *LNCIS*, pp. 298–372, Springer Verlag, 2000.
- [3] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using Gaussian processes and factor graphs," in *Robotics: Science and Systems (RSS)*, 2016.
- [4] D.-N. Ta, M. Kobilarov, and F. Dellaert, "A factor graph approach to estimation and model predictive control on unmanned aerial vehicles," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 181–188, 2014.
- [5] M. Mukadam, C.-A. Cheng, X. Yan, and B. Boots, "Approximately optimal continuous-time motion planning and control via probabilistic inference," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 664–671, 2017.
- [6] S. Agarwal, K. Mierle, and Others, "Ceres solver." <http://ceres-solver.org>.
- [7] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, (Shanghai, China), May 2011.
- [8] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Tech. Rep. GT-RIM-CP&R-2012-002, Georgia Institute of Technology, 2012.
- [9] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle adjustment in the large," in *European Conf. on Computer Vision (ECCV)*, 2010.
- [10] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, "Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 4597–4604, 2015.
- [11] W. Jakob, J. Rhinelander, and D. Moldovan, "pybind11 – seamless operability between C++11 and python," 2017. <https://github.com/pybind/pybind11>.
- [12] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3." <http://eigen.tuxfamily.org>, 2010.
- [13] H. Strasdat, "Sophus: C++ implementation of lie groups using eigen." <https://github.com/strasdat/Sophus>.
- [14] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate," *ACM Transactions on Mathematical Software (TOMS)*, vol. 35, no. 3, p. 22, 2008.

APPENDIX: PYTHON EXAMPLE CODE SNIPPETS

Snippet 1. A pose graph optimization in Python

```

import numpy as np
from minisam import *
from minisam.sophus import *

# build factor graph for least square problem
graph = FactorGraph()
loss = DiagonalLoss.Sigmas(np.array([1.0, 1.0, 0.1])) # loss function of sensor measurement model
graph.add(PriorFactor(key('x', 1), SE2(SO2(0), np.array([0, 0])), loss)) # prior as first pose
graph.add(BetweenFactor(key('x', 1), key('x', 2), SE2(SO2(0), np.array([5, 0])), loss)) # odometry measurements
graph.add(BetweenFactor(key('x', 2), key('x', 3), SE2(SO2(-3.14/2), np.array([5, 0])), loss))
graph.add(BetweenFactor(key('x', 3), key('x', 4), SE2(SO2(-3.14/2), np.array([5, 0])), loss))
graph.add(BetweenFactor(key('x', 4), key('x', 5), SE2(SO2(-3.14/2), np.array([5, 0])), loss))
graph.add(BetweenFactor(key('x', 5), key('x', 2), SE2(SO2(-3.14/2), np.array([5, 0])), loss)) # loop closure

# variables initial guess, with random added-on noise
init_values = Variables()
init_values.add(key('x', 1), SE2(SO2(0.2), np.array([0.2, -0.3])))
init_values.add(key('x', 2), SE2(SO2(-0.1), np.array([5.1, 0.3])))
init_values.add(key('x', 3), SE2(SO2(-3.14/2 - 0.2), np.array([9.9, -0.1])))
init_values.add(key('x', 4), SE2(SO2(-3.14 + 0.1), np.array([10.2, -5.0])))
init_values.add(key('x', 5), SE2(SO2(3.14/2 - 0.1), np.array([5.1, -5.1])))

# solve least square optimization by Levenberg-Marquardt algorithm
opt = LevenbergMarquardtOptimizer()
result_values = Variables() # results
status = opt.optimize(graph, init_values, result_values)
if status != NonlinearOptimizationStatus.SUCCESS:
    print("optimization error :", status)

```

Snippet 2. A minimal Python prior factor example on SE(2)

```

import numpy as np
from minisam import *

# python implementation of prior factor on SE2
class PyPriorFactorSE2(Factor): # or inherit from NumericalFactor
    # constructor
    def __init__(self, key, prior, loss):
        Factor.__init__(self, 3, [key], loss)
        self.prior_ = prior
    # make a deep copy
    def copy(self):
        return PyPriorFactorSE2(self.keys()[0], self.prior_, self.lossFunction())
    # error vector
    def error(self, variables):
        curr_pose = variables.at(self.keys()[0]) # current variable
        return (self.prior_.inverse() * curr_pose).log()
    # jacobians, not needed if inherit from NumericalFactor
    def jacobians(self, variables):
        return [np.eye(3)]

```

Snippet 3. A minimal Python 2D point optimizable manifold

```

import numpy as np

# A 2D point class (x, y)
class PyPoint2D(object):
    # constructor
    def __init__(self, x, y):
        self.x = float(x)
        self.y = float(y)
    # local coordinate dimension
    def dim(self):
        return 2
    # map manifold point other to local coordinate
    def local(self, other):
        return np.array([other.x - self.x, other.y - self.y], dtype=np.float)
    # apply changes in local coordinate to manifold, \oplus operator
    def retract(self, vec):
        return PyPoint2D(self.x + vec[0], self.y + vec[1])

```


Linear Camera Velocities and Point Feature Depth Estimation Using Unknown Input Observer

R. Benyoucef, L. Nehaoua, H. Hadj-Abdelkader and H. Arioui

Abstract—In this paper, we propose a new approach to estimate the missing 3D information of a point feature during the camera motion and reconstruct the linear velocity of the camera. This approach is intended to solve the problem of relative localization and compute the distance between two Unmanned Aerial Vehicles (UAV) within a formation. An Unknown Input Observer is designed for the considered system described by a quasi-linear parameter varying (qLPV) model with unmeasurable variables to achieve kinematic from motion estimation. An observability analysis is performed to ensure the possibility of reconstructing the state variables. Sufficient conditions to design the observer are derived in terms of Linear Matrix Inequalities (LMIs) based on Lyapunov theory. Simulation results are discussed to validate the proposed approach.

Keywords: Nonlinear Observers, qLPV Systems, Feature Depth Estimation, LMI constraints, Lyapunov Theory, Kinematic from Motion.

I. INTRODUCTION

Many works have been conducted to solve the localization problem when a team of robots cooperate with each other to achieve some defined tasks. For example in [17], this problem is solved for two terrestrial robots using exteroceptive sensors. Also In [18], a visual localization modules is proposed to estimate the relative positions of agents within a fleet of Unmanned Aerial Vehicles (UAVs).

The challenging problem of 3D structure estimation using the visual information has attracted more interest recently, we can find in literature various techniques to tackle this problem which can refer to Simultaneous Localization And Mapping (SLAM) in robotics [1] and Structure from Motion (SfM) in computer vision [2] [19].

In earlier work, researchers have addressed this problem using Stereo Vision Algorithms [3], which consists on reconstructing the depth of a feature point from two images of the same scene using triangulation. But later on, the idea of using a single camera lead to multiple other approaches, one can cite [4], where the observation of the point feature depth is achieved using the persistency of excitation lemma that results from the adaptive control theory [5], [6] and [7]. One of the major disadvantage of all these cited works is the fact that their analysis is based on the assumption of neglecting a disturbance term which affects the dynamic behavior of the system. Furthermore solutions based on Extended Kalman Filter (EKF) have been proposed in [8] [9]. However the main drawback of this approach is that they involve a certain

degree of linearization which contradicts most of the studied system dynamics. In the present paper, we achieve the estimation of the 3D information of a feature point together with recovering the linear velocity of the camera with respect to x and y axis assuming a perfect knowledge of the angular velocity of the camera and the linear velocity with respect to its z axis. The system is described with qLPV representation [13][14] and based on the new description of the system, An unknown input observer (UIO) is designed, which allows estimating the state of the system in presence of unknown inputs.

Obtaining an accurate linear velocity have a significant effect on the control of autonomous vehicles and drones. The straight forward method to estimate the velocity is to use data fusion [10] where Global Positioning System (GPS) and Inertial Measurement Unit (IMU) are used for this purpose. But this method fails when it comes to indoor tasks or limited sensor resolution. Moreover, we can find in literature some other approaches. For example in [11], an Extended Kalman filter (EKF) is employed to estimate linear and angular velocity of an object during its free flight, on the other hand in [12], Riccati observer is used to solve this problem. This work focuses on solving the relative localization problem for coordination and control of multiple autonomous aerial agents using the data provided by the on-board cameras of the UAVs to estimate the relative distance from the other agents with respect to its camera reference frame and reconstruct the relative velocity of the camera. This information is essential for data fusion or to give an accurate estimate of the absolute velocity of the vehicle.

The main contribution of this work can be summarised in two points the first one is introducing a novel description of the relation between the variation of the feature extracted from an image and the linear/angular velocities of the camera using qLPV representation, based on which a nonlinear observer is designed to estimate the depth, and the second point consists on reconstructing the camera linear velocity with respect to its x axis and y axis during its motion.

This paper is structured as follows: in section II, basic definitions are highlighted and the nonlinear model of camera is described. In section III, the new description using qLPV representation is explained for the nonlinear model of camera. In Section IV the design of the nonlinear observer is presented and the sufficient conditions are given in terms of LMIs based on Lyapunov theory. Simulation tests are conducted to discuss the performances of the proposed observer in section V. Finally, section VI draws some conclusions regarding our work.

All authors are with IBISC Lab, Evry Val d'Essonne (UEVE), Paris Saclay University, 43 Rue du Pelvoux, 91080 Courcouronnes. France rayane.benyoucef@univ-evry.fr

II. MATHEMATICAL BACKGROUND

In this section, we first provide some basic definitions and lemmas needed for the development of the proposed approach. Then, we recall the conventional camera model.

A. Notations and basic definitions

We represent matrices in upper case bold letters \mathbf{X} and vectors in lower case bold letters \mathbf{x} otherwise, the remaining notations represent scalars (x or X).

We recall in the following the theorems used in the analysis of the observer convergence:

Theorem 1 (Strong Detectability Condition): For every matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{F} \in \mathbb{R}^{n \times q}$ and $\mathbf{C} \in \mathbb{R}^{m \times n}$. We consider the following Linear Time Invariant (LTI) system:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{F}\mathbf{d}(t) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \end{cases} \quad (1)$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$ and $\mathbf{d} \in \mathbb{R}^q$ are respectively the state vector, the unknown input vector and the output vector. The system (1) is strongly detectable if:

$$\lim_{t \rightarrow \infty} \mathbf{y}(t) = 0 \Rightarrow \lim_{t \rightarrow \infty} \mathbf{x}(t) = 0 \quad (2)$$

regardless of the input and the initial state. Algebraically this is equivalent to:

$$\text{rank}(\mathbf{R}(p)) = n + q \quad (3)$$

where p represents the pole of the system and \mathbf{R} denotes the Rosenbrock matrix of system (1), given by:

$$\mathbf{R} = \begin{bmatrix} p\mathbf{I} - \mathbf{A} & -\mathbf{F} \\ \mathbf{C} & 0 \end{bmatrix} \quad (4)$$

Lemma 1: For every matrix $\mathbf{G} = \mathbf{G}^T > 0$, \mathbf{X} and \mathbf{Y} with appropriate dimensions, the property below holds:

$$\mathbf{X}^T \mathbf{Y} + \mathbf{Y}^T \mathbf{X} \leq \mathbf{X}^T \mathbf{G} \mathbf{X} + \mathbf{Y}^T \mathbf{G}^{-1} \mathbf{Y} \quad (5)$$

Lemma 2 (Schur complement lemma): Consider the following convex nonlinear inequalities:

$$\mathbf{R} > 0, \quad \mathbf{T} - \mathbf{S}\mathbf{R}^{-1}\mathbf{S}^T > 0 \quad (6)$$

where the matrices $\mathbf{T} = \mathbf{T}^T$, $\mathbf{R} = \mathbf{R}^T$ and \mathbf{S} are of appropriate dimension. Hence, the previous inequalities can be written in the following form:

$$\begin{bmatrix} \mathbf{T} & \mathbf{S} \\ \mathbf{S}^T & \mathbf{R} \end{bmatrix} > 0 \quad (7)$$

Note that the previous mathematical properties for the LTI systems hold for the case of qLPV systems considering the case of frozen parameter vectors.

B. Conventional camera model

Let P be a 3-D point of coordinates $\mathbf{p} = (X \ Y \ Z)^T$ defined in the camera frame \mathcal{F}_c . Its projection onto the image plane is obtained through the well-known Pinhole model. More precisely, the 3-D point p is projected in the image as a 2-D point with homogeneous coordinates given by the vector \mathbf{m} as:

$$\mathbf{m} = (x \ y \ 1)^T = \frac{1}{Z} \mathbf{p} \quad (8)$$

The velocity of the 3D point p is related to the camera special velocity by:

$$\dot{\mathbf{p}} = -v + \mathbf{p} \times \boldsymbol{\omega} = (-\mathbf{I} \ [\mathbf{p}]_{\times}) \mathbf{u} \quad (9)$$

where $[\]_{\times}$ refers to the skew-symmetric matrix of a given vector, $\mathbf{u} = (v^T \ \boldsymbol{\omega}^T)^T$ is the spatial velocity of the camera motion, with $v = (v_x \ v_y \ v_z)^T$ and $\boldsymbol{\omega} = (\omega_x \ \omega_y \ \omega_z)^T$ are respectively, the instantaneous linear and angular velocities of the camera frame. From (9), the dynamics of the inverse of the depth $\frac{1}{Z}$ is given by:

$$\frac{d}{dt} \left(\frac{1}{Z} \right) = \left(0 \ 0 \ -\frac{1}{Z^2} \ -\frac{y}{Z} \ \frac{x}{Z} \ 0 \right) \mathbf{u} \quad (10)$$

The time derivative of the image point \mathbf{m} is linked to the camera spatial velocity \mathbf{u} by the following interaction matrix [13]:

$$\dot{\mathbf{m}} = \begin{pmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & (1+y^2) & -xy & -x \end{pmatrix} \mathbf{u} \quad (11)$$

Let us now define the state vector as $\mathbf{X} = (\mathbf{s}^T, \chi)^T$ with $\mathbf{s} = (x \ y)^T \in \mathbb{R}^2$ is a *measurable* vector, and $\chi = \frac{1}{Z} \in \mathbb{R}$ is the *unmeasurable* 3D data that we want to estimate. Using (11) and (10), the dynamics of the state vector \mathbf{X} is given by:

$$\begin{cases} \dot{\mathbf{s}} = \mathbf{f}_m(\mathbf{s}, \mathbf{u}) + \boldsymbol{\Omega}^T(\mathbf{s}, \mathbf{u}) \chi \\ \dot{\chi} = \mathbf{f}_u(\mathbf{s}, \chi, \mathbf{u}) \end{cases} \quad (12)$$

where the vectors $\boldsymbol{\Omega}^T(\mathbf{s}, \mathbf{u}) \in \mathbb{R}^2$, $\mathbf{f}_m(\mathbf{s}, \mathbf{u}) \in \mathbb{R}^2$ and $\mathbf{f}_u(\mathbf{s}, \chi, \mathbf{u}) \in \mathbb{R}$ are generic and sufficiently smooth w.r.t their arguments and they are defined as:

$$\begin{cases} \mathbf{f}_m(\mathbf{s}, \mathbf{u}) = \begin{pmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{\Omega}(\mathbf{s}, \mathbf{u}) = (-v_x + xv_z \quad -v_y + yv_z) \\ \mathbf{f}_u(\mathbf{s}, \chi, \mathbf{u}) = v_z \chi^2 + (y\omega_x - x\omega_y) \chi \end{cases} \quad (13)$$

In the upcoming sections, the dynamic model given in (12) will be expressed in a qLPV form in order to design a proper nonlinear Unknown Input (UI) Observer to estimate the depth information χ and recover the linear velocities with respect to the x and y axis of the camera.

III. POLYTOPIC FORMULATION & DETECTABILITY ANALYSIS

We express in this section, the vision system model (13) into qLPV structure and analyze the existence of the nonlinear UI observer.

The objective of this paper is to estimate the depth information $\frac{1}{Z}$ and reconstruct the linear velocities during the camera motion using a nonlinear unknown input observer. For this purpose, we represent the system (12) in a state space form as follows:

$$\begin{cases} \dot{\mathbf{x}} &= \mathbf{A}(\mathbf{x}, \mathbf{u}) \mathbf{x} + \mathbf{B}(\mathbf{y}) \boldsymbol{\omega} + \mathbf{F} \mathbf{d} \\ \mathbf{y} &= \mathbf{C} \mathbf{x} \end{cases} \quad (14)$$

where:

$$\mathbf{A}(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} 0 & 0 & xv_z \\ 0 & 0 & yv_z \\ y\omega_x & x\omega_y & \chi v_z + \omega_x y - x\omega_y \end{pmatrix}$$

$$\mathbf{B}(\mathbf{y}) = \begin{pmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \\ -xy & -xy & 0 \end{pmatrix}$$

$$\mathbf{F} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \quad \mathbf{d} = \begin{pmatrix} -\chi v_x \\ -\chi v_y \end{pmatrix}$$

and \mathbf{y} represents the output of the system with:

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Using the sector nonlinearity approach, the previous system (14) can be represented in the polytopic form as follows:

$$\begin{cases} \dot{\mathbf{x}} &= \sum_{i=1}^r \mu_i(\mathbf{x}) (\mathbf{A}_i \mathbf{x} + \mathbf{B}(\mathbf{y}) \boldsymbol{\omega} + \mathbf{F} \mathbf{d}) \\ \mathbf{y} &= \mathbf{C} \mathbf{x} \end{cases} \quad (15)$$

where $\mathbf{A}_i \in \mathbb{R}^{3 \times 3}$, $\mathbf{B}(\mathbf{y}) \in \mathbb{R}^{3 \times 3}$ and $\mu_i, i = 1, \dots, r$ are the weighting functions with r is the number of sub-models that depends on the number of nonlinearities in the system (in our case we have five nonlinearities). These weighting functions satisfy the following convex sum property on the considered compact bounds of the nonlinearities of the system:

$$\begin{cases} 0 \leq \mu_i \leq 1 \\ \sum_{i=1}^r \mu_i = 1 \end{cases} \quad (16)$$

Note that in hereafter we are going to consider the discrete-time form of the continuous-time system (15) represented before and keep the same notations. Using Forward Euler Approximation for state space models, the previous system will have the following form:

$$\begin{cases} \mathbf{x}(k+1) &= \sum_{i=1}^r \mu_i(\mathbf{x}(k)) (\mathbf{A}_i \mathbf{x}(k) + \mathbf{B}(\mathbf{y}(k)) \boldsymbol{\omega}(k) + \mathbf{F} \mathbf{d}(k)) \\ \mathbf{y}(k) &= \mathbf{C} \mathbf{x}(k) \end{cases} \quad (17)$$

with k is the sampling instant.

The new description of the model enables us to synthesis a proper nonlinear unknown input observer to estimate the depth information and reconstruct the linear velocities. This type of observer exists under the following conditions:

- 1) $\text{rank}(\mathbf{C} \mathbf{F}) = \text{rank}(\mathbf{F})$.
- 2) the system (14) is strong detectable, that means, it satisfies the condition stated in theorem (1)

After verifying the two conditions above, we can state that the UI observer exists. In the next section, we discuss the observer design.

IV. DESIGN OF THE UI OBSERVER

In this section we present the UI Observer design given in the form:

$$\begin{cases} \mathbf{z}(k+1) &= \sum_{i=1}^r \mu_i(\hat{\mathbf{x}}(k)) (\mathbf{N}_i \mathbf{z}(k) + \mathbf{G}_i \boldsymbol{\omega}(k) + \mathbf{L}_i \mathbf{y}(k)) \\ \hat{\mathbf{x}}(k) &= \mathbf{z}(k) - \mathbf{E} \mathbf{y}(k) \end{cases} \quad (18)$$

where $\mathbf{z} \in \mathbb{R}^3$ is the state of the observer and $\hat{\mathbf{x}} \in \mathbb{R}^3$ the estimated state and the matrices \mathbf{N}_i , \mathbf{G}_i , \mathbf{L}_i and \mathbf{E} are the matrices gains to be computed such that the state estimation error given by (19) converges to zero.

$$\begin{aligned} \mathbf{e}(k) &= \mathbf{x}(k) - \hat{\mathbf{x}}(k) \\ &= (\mathbf{I} + \mathbf{E} \mathbf{C}) \hat{\mathbf{x}}(k) - \mathbf{z}(k) \end{aligned} \quad (19)$$

With $\mathbf{T} = \mathbf{I} + \mathbf{E} \mathbf{C}$, the error will be defined as:

$$\mathbf{e}(k) = \mathbf{T} \hat{\mathbf{x}}(k) - \mathbf{z}(k) \quad (20)$$

For sake of simplicity, in what follows we put: $\mathbf{e}(k) = e_k$. Thus, the expression of the estimation error is equivalent to:

$$\begin{aligned} \mathbf{e}_{k+1} &= \mathbf{T} \hat{\mathbf{x}}_{k+1} - \mathbf{z}_{k+1} \\ &= \sum_{i=1}^r \mu_i(\hat{\mathbf{x}}_k) (\mathbf{N}_i \mathbf{e}_k + (\mathbf{T} \mathbf{A}_i - \mathbf{K}_i \mathbf{C} - \\ &\quad \mathbf{N}_i) \mathbf{x}_k + \mathbf{T} \mathbf{F} \mathbf{d}_k + (\mathbf{T} \mathbf{B}_i - \mathbf{G}_i) \boldsymbol{\omega}_k) + \Delta \end{aligned} \quad (21)$$

with $\Delta = \mathbf{T} (\mu_i(\mathbf{x}_k) - \mu_i(\hat{\mathbf{x}}_k)) (\mathbf{A}_i \mathbf{x}_k + \mathbf{B}_i \boldsymbol{\omega}_k + \mathbf{F} \mathbf{d}_k)$ and $\mathbf{K}_i = \mathbf{N}_i \mathbf{F} - \mathbf{L}_i$.

We assume that all the elements in Δ are growth bounded with respect to \mathbf{e}_k . Thus we can say that Δ fulfills the calmness property at the origin:

$$\Delta^T \Delta = \|(\Delta)\|^2 < \alpha^2 \|\hat{\mathbf{x}}_k - \mathbf{x}_k\|^2 = \alpha^2 \|\mathbf{e}_k\|^2 \quad (22)$$

The notation $\|\cdot\|$ represents the 2-norm and $\alpha^2 > 0$ is constant of Lipschitz.

To ensure the stability of the error dynamics (21), the following conditions must be satisfied $\forall i = 1, \dots, 32$:

- 1) The system defined by: $\mathcal{N}_i = \sum_{i=1}^r \mu_i(\hat{\mathbf{x}}_k) \mathbf{N}_i$ is stable where: $\mathbf{e}_{k+1} = \mathcal{N}_i \mathbf{e}_k + \Delta$.
- 2) $\mathbf{T} \mathbf{A}_i - \mathbf{K}_i \mathbf{C} - \mathbf{N}_i = 0$
- 3) $\mathbf{T} \mathbf{B}_i - \mathbf{G}_i = 0$
- 4) $\mathbf{T} \mathbf{F} = 0$

The first condition implies that \mathcal{N}_e is Hurwitz subject to a vanishing disturbance Δ i.e: $\Delta \rightarrow 0$ when $\hat{x} \rightarrow X$ and to demonstrate that [16], we consider the following quadratic Lyapunov function:

$$V = \mathbf{e}_k^T \mathbf{P} \mathbf{e}_k \quad \mathbf{P} = \mathbf{P}^T > 0 \quad (23)$$

It follows that:

$$\begin{aligned} V_{k+1} - V_k &= \mathbf{e}_{k+1}^T \mathbf{P} \mathbf{e}_{k+1} - \mathbf{e}_k^T \mathbf{P} \mathbf{e}_k \\ &= \mathbf{e}_k^T \mathcal{N}_e^T \mathbf{P} \mathcal{N}_e \mathbf{e}_k + \Delta_k^T (X, \hat{X}) \mathbf{P} \mathcal{N}_e \mathbf{e}_k + \\ &\quad \mathbf{e}_k^T \mathcal{N}_e^T \mathbf{P} \Delta + \Delta^T \mathbf{P} \Delta \end{aligned} \quad (24)$$

To ensure the stability of the system the time derivative of the Lyapunov function must satisfy:

$$\mathbf{e}_k^T \mathcal{N}_e^T \mathbf{P} \mathcal{N}_e \mathbf{e}_k + \Delta_k^T (X, \hat{X}) \mathbf{P} \mathcal{N}_e \mathbf{e}_k + \mathbf{e}_k^T \mathcal{N}_e^T \mathbf{P} \Delta + \Delta^T \mathbf{P} \Delta < 0 \quad (25)$$

To attenuate the disturbance's effect Δ on the estimation error \mathbf{e}_k in the L_2 -gain sense, we define:

$$\sup_{\|\Delta\| \neq 0} \frac{\|\mathbf{e}_k\|}{\|\Delta\|} < \gamma^2 \quad (26)$$

which leads to the following inequality:

$$\mathbf{e}_k^T \mathbf{e}_k - \gamma^2 \Delta^T \Delta < 0 \quad (27)$$

This expression can be simplified using lemma 1, the resulting inequality is given by:

$$\Delta^T \mathbf{P} \mathcal{N}_e \mathbf{e}_k + \mathbf{e}_k^T \mathcal{N}_e^T \mathbf{P} \Delta < \epsilon \Delta^T \Delta + \frac{1}{\epsilon} \mathbf{e}_k^T \mathcal{N}_e^T \mathbf{P}^T \mathbf{P} \mathcal{N}_e \mathbf{e}_k \quad (28)$$

Then the following inequality is deduced:

$$\begin{aligned} V_{k+1} - V_k &< \mathbf{e}_k^T (\mathcal{N}_e^T \mathbf{P} \mathcal{N}_e - \mathbf{P} + \mathbf{I} + \frac{1}{\epsilon} \mathcal{N}_e^T \mathbf{P}^T \mathbf{P} \mathcal{N}_e) \mathbf{e}_k + \\ &\quad \Delta^T (\epsilon \mathbf{I} + \gamma^2 \mathbf{I} + \mathbf{P}) \Delta \end{aligned} \quad (29)$$

It follows:

$$\begin{aligned} \mathbf{e}_k^T (\mathcal{N}_e^T \mathbf{P} \mathcal{N}_e - \mathbf{P} + \mathbf{I} + \frac{1}{\epsilon} \mathcal{N}_e^T \mathbf{P}^T \mathbf{P} \mathcal{N}_e) \mathbf{e}_k + \\ \Delta^T (\epsilon \mathbf{I} - \gamma^2 \mathbf{I} + \mathbf{P}) \Delta < 0 \end{aligned} \quad (30)$$

Taken into account the Lipschitz condition (22), the following inequality holds:

$$\begin{aligned} \mathbf{e}_k^T (\mathcal{N}_e^T \mathbf{P} \mathcal{N}_e - \mathbf{P} + \mathbf{I} + \frac{1}{\epsilon} \mathcal{N}_e^T \mathbf{P}^T \mathbf{P} \mathcal{N}_e + \\ \alpha \epsilon \mathbf{I} - \alpha^2 \gamma^2 \mathbf{I} + \alpha^2 \mathbf{P}) \mathbf{e}_k < 0 \end{aligned} \quad (31)$$

The inequality (31) holds if and only if:

$$\mathcal{N}_e^T \mathbf{P} \mathcal{N}_e - \mathbf{P} + \mathbf{I} + \frac{1}{\epsilon} \mathcal{N}_e^T \mathbf{P}^T \mathbf{P} \mathcal{N}_e + \alpha^2 \epsilon \mathbf{I} - \alpha^2 \gamma^2 \mathbf{I} + \alpha^2 \mathbf{P} < 0 \quad (32)$$

Using Schur lemma 2, the inequality (32) can be expressed in an equivalent manner with the LMI constraints as:

$$\begin{bmatrix} \mathbf{I} - \mathbf{P} + \alpha^2 \epsilon \mathbf{I} - \alpha^2 \gamma^2 \mathbf{I} & \alpha \mathbf{P} & \mathcal{N}_e^T \mathbf{P} & \mathcal{N}_e^T \mathbf{P} \\ \alpha \mathbf{P} & \mathbf{P} & 0 & 0 \\ \mathbf{P} \mathcal{N}_e & 0 & -\epsilon \mathbf{I} & 0 \\ \mathbf{P} \mathcal{N}_e & 0 & 0 & -\mathbf{P} \end{bmatrix} < 0 \quad (33)$$

Substituting the term \mathcal{N}_e in the previous equation (33) yields:

$$\sum_1^r \mu_i \begin{bmatrix} \mathbf{I} - \mathbf{P} + \alpha^2 \epsilon \mathbf{I} - \alpha^2 \gamma^2 \mathbf{I} & \alpha \mathbf{P} & \mathbf{N}_i^T \mathbf{P} & \mathbf{N}_i^T \mathbf{P} \\ \alpha \mathbf{P} & \mathbf{P} & 0 & 0 \\ \mathbf{P} \mathbf{N}_i & 0 & -\epsilon \mathbf{I} & 0 \\ \mathbf{P} \mathbf{N}_i & 0 & 0 & -\mathbf{P} \end{bmatrix} < 0 \quad (34)$$

The inequality (34) is equivalent to following in more conservative manner, for all $i = 1, \dots, 32$:

$$\begin{bmatrix} \mathbf{I} - \mathbf{P} + \alpha^2 \epsilon \mathbf{I} - \alpha \gamma^2 \mathbf{I} & \alpha \mathbf{P} & \mathbf{N}_i^T \mathbf{P} & \mathbf{N}_i^T \mathbf{P} \\ \alpha \mathbf{P} & \mathbf{P} & 0 & 0 \\ \mathbf{P} \mathbf{N}_i & 0 & -\epsilon \mathbf{I} & 0 \\ \mathbf{P} \mathbf{N}_i & 0 & 0 & -\mathbf{P} \end{bmatrix} < 0 \quad (35)$$

From the second condition to ensure the stability of the error dynamics, one can write: $\mathbf{N}_i = \mathbf{T} \mathbf{A}_i - \mathbf{K}_i \mathbf{C}$. After substituting \mathbf{N}_i , We proceed to the following changing of variables: $\bar{\lambda} = \alpha^2 \gamma^2$, $\bar{\eta} = \alpha^2 \epsilon$, $\mathbf{Q} = \alpha \mathbf{P}$ and $\mathbf{W}_i = \mathbf{P} \mathbf{K}_i$ the inequality (35) becomes:

$$\begin{bmatrix} \mathbf{I} - \mathbf{P} + \bar{\eta} \mathbf{I} - \bar{\lambda} \mathbf{I} & \mathbf{Q} & \Psi^T \mathbf{P} & \Psi^T \mathbf{P} \\ \mathbf{Q} & -\mathbf{P} & 0 & 0 \\ \mathbf{P} \Psi & 0 & \epsilon \mathbf{I} & 0 \\ \mathbf{P} \Psi & 0 & 0 & -\mathbf{P} \end{bmatrix} < 0 \quad (36)$$

where $\Psi = \mathbf{T} \mathbf{A}_i + \mathbf{W}_i \mathbf{C}$.

Then the system (1) is stable if there exist a positive definite symmetric matrices $\mathbf{P} \in R^{3 \times 3}$, $\mathbf{Q} \in R^{3 \times 3}$, matrices $\mathbf{W}_i \in R^{3 \times 2}$, positive scalars $\bar{\eta}$ and $\bar{\lambda}$ so that the LMIs in (35) are satisfied and the resulting observer gains are given by $\mathbf{K}_i = \mathbf{P}^{-1} \mathbf{W}_i$

Note that in order to have feasible solution of the LMIs, the pair $(\mathbf{T} \mathbf{A}, \mathbf{C})$ should be observable or at least detectable, we study the detectability of the system by analysing the poles. As a consequence, to fulfil the requirement of detectability the following condition must be satisfied:

$$y \omega_x + \chi v_z - x \omega_y + 1 < 0 \quad (37)$$

To summarize, after ensuring the existence of the observer, we proceed to its design according to the given steps below:

- 1) deduce \mathbf{E} from the equation (4) Since the condition $rank(\mathbf{C} \mathbf{F}) = rank(\mathbf{F})$ holds.
- 2) calculate the matrix \mathbf{E} , the matrix \mathbf{T} is computed directly from (3) as well as the matrices \mathbf{G}_i from the equation (2).
- 3) ensure that the pair $(\mathbf{T} \mathbf{A}, \mathbf{C})$ is at least detectable and solve the LMIs constraints (35) to get the gains $\mathbf{K}_i = \mathbf{P}^{-1} \mathbf{W}_i$.
- 4) Finally, compute the gain matrices: $\mathbf{N}_i = \mathbf{T} \mathbf{A}_i - \mathbf{K}_i \mathbf{C}$ and $\mathbf{L}_i = \mathbf{K}_i - \mathbf{N}_i \mathbf{E}$.

The linear velocities in the x and y directions of the camera are expressed in the disturbance part of the system and they can be recovered once the estimated states converge to the real ones.

$$\mathbf{y}_{k+1} = \mathbf{C} \sum_{i=1}^r \mu_i (X_k) (\mathbf{A}_i X_k + \mathbf{B}(\mathbf{y}_k) \omega_k + \mathbf{F} \mathbf{d}_k) \quad (38)$$

It follows that:

$$\hat{\mathbf{d}}_k = (\mathbf{C}\mathbf{F})^{-1} \left[\mathbf{y}_{k+1} - \mathbf{C} \sum_{i=1}^r \mu_i(\hat{\mathbf{x}}_k) (\mathbf{A}_i \hat{\mathbf{x}}_k + \mathbf{B}(\mathbf{y}_k) \omega_k) \right] \quad (39)$$

The unknown input has the given form:

$$\mathbf{d}_k = \begin{pmatrix} -\chi v_x \\ -\chi v_y \end{pmatrix} \quad (40)$$

The estimate of linear velocity with respect to x and y axis of the camera is obtained as follows:

$$\begin{pmatrix} \hat{v}_x \\ \hat{v}_y \end{pmatrix} = -\hat{\mathbf{d}}_k \hat{\chi}^{-1} \quad (41)$$

V. SIMULATION RESULTS

In order to validate the proposed approach, we consider two sets of synthetic images generated at a rate of $20fps$ using a known camera motion. The real depth information χ of the tracked point feature as well as the linear velocities v_x and v_y are compared with the estimated ones and used in the discussion of the observer performance.

The LMIs conditions derived previously are solved and yield the following result:

$$\epsilon = 2.7666, \quad \bar{\lambda} = 4.5048, \quad \bar{\eta} = 1.6908$$

$$\mathbf{P} = \begin{pmatrix} 2.1209 & 0 & 0 \\ 0 & 2.1209 & 0 \\ 0 & 0 & 1.1217 \end{pmatrix}$$

$$\mathbf{Q} = 10^{-11} \begin{pmatrix} 0.0093 & 0.1220 & -0.0001 \\ 0.1220 & 0.0296 & -0.0035 \\ -0.0001 & -0.0035 & 0 \end{pmatrix}$$

The first set where original and final images are shown in figure (1), is generated using the following linear/angular velocities of the camera:

$$v_x = 0.2 \sin(\pi t) \quad v_y = -0.2 + 0.1t \quad v_z = -0.7$$

$$\omega_x = 0.1 \quad \omega_y = -0.2 \quad \omega_z = 0$$

The red dot in the images represents the tracked point that we want to estimate its depth.



Fig. 1: (a) the original and (b) the final images of the first set of images.

To better verify the performance of the observer, we consider now a second set of images where the original and final

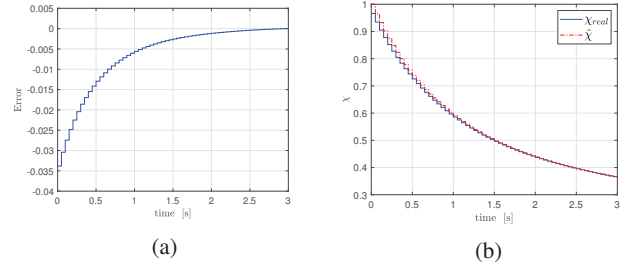


Fig. 2: (a) Estimation error (b) Real and estimated depths of the selected image point.

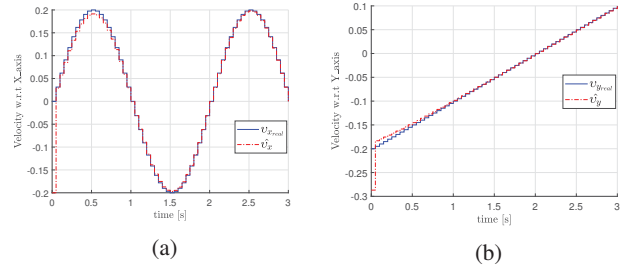


Fig. 3: Real and estimated linear velocities: (a) v_x and (b) v_y .

images are shown in figure (4). The set of images is generated using the linear/angular velocities of the camera defined below:

$$v_x = 0.4 \cos(2\pi t) \quad v_y = 0.5 \cos(\pi t) \quad v_z = -0.7 \cos(\pi t) - 0.3$$

$$\omega_x = 0.1 \quad \omega_y = -0.1 \quad \omega_z = 0.1$$



Fig. 4: (a) the original and (b) the final images of the second set of images.

Note that the initial value of the estimated depth information $\hat{\chi}$ is restricted by the Lipschitz condition (22). Therefore, a close initial value to the real value of the depth is required for the estimation as shown in figures (2b) and (5b).

It can be noticed from the evolution of the estimation error for the first and the second set depicted in figures (2a) and (5a) respectively, that the convergence is achieved within approximately 2.5 sec .

Figures (3) and (6) highlight the reconstructed velocities along the x and y axis respectively for both sets. From the depicted figures one can see that the velocities are

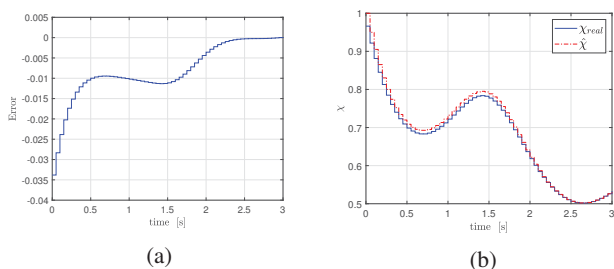


Fig. 5: (a) Estimation error (b) Real and estimated depths of the selected image point.

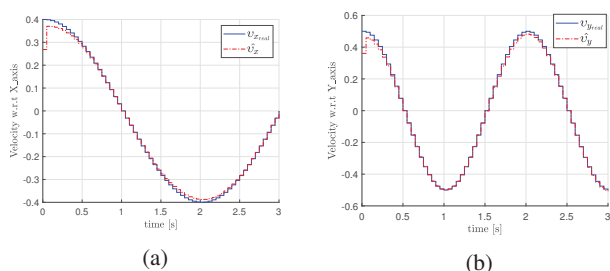


Fig. 6: Real and estimated linear velocities: (a) v_x and (b) v_y .

well recovered when the the estimated depth information $\hat{\chi}$ converges to the real value.

Note that, to correctly estimate the depth, the velocity should be perfectly known however, In real experiments, there are very few use cases where a perfect knowledge is available, so it is required to have velocities measurements close to accurate using different techniques of filtering or data fusion for the approach to work. Usually standard triangulation techniques used in computer vision don't require strong assumptions to recover the 3d structure but with the proposed approach the linear velocity of the camera along the x and y axis can be reconstructed while estimating the depth.

VI. CONCLUSIONS

In the present paper, we have proposed a solution to estimate the depth information of a feature point and to recover the linear velocities of the camera with respect to x and y axis. The nonlinear system describing the relationship between the feature point time variation and the camera spatial velocity has been represented by a discrete time Takagi-sugeno model. An adequate Unknown Input Observer has been designed to estimate the depth information and reconstruct the camera velocities. The convergence of the state estimation error has been analysed using the Lyapunov theory and the convergence conditions have been formulated as LMIs constraints. The performances of the proposed observer have been validated using two sets of synthetic images. Simulation results have shown the good convergence of the observer.

In future works, the proposed technique will be used to compute the relative distance between the considered UAV and the other flying robots with respect to its camera reference frame in a group formation, as well as to reconstruct the linear velocities of the UAV.

REFERENCES

- [1] Egodagamage, R., & Tuceryan, M. (2017). Distributed monocular SLAM for indoor map building. *Journal of Sensors*, 2017.
- [2] Spica, R., & Giordano, P. R. (2013, December). A framework for active estimation: Application to structure from motion. In *52nd IEEE conference on decision and control* (pp. 7647-7653). IEEE.
- [3] Nalpantidis, L., & Gasteratos, A. (2012). Stereo vision depth estimation methods for robotic applications. In *Depth Map and 3D Imaging Applications: Algorithms and Technologies* (pp. 397-417). IGI global.
- [4] De Luca, A., Oriolo, G., & Robuffo Giordano, P. (2008). Feature depth observation for image-based visual servoing: Theory and experiments. *The International Journal of Robotics Research*, 27(10), 1093-1116.
- [5] Marino, R., & Tomei, P. (1995). *Nonlinear control design: geometric, adaptive and robust* (Vol. 136). London: Prentice Hall.
- [6] Spica, R., & Giordano, P. R. (2013, December). A framework for active estimation: Application to structure from motion. In *52nd IEEE conference on decision and control* (pp. 7647-7653). IEEE.
- [7] Spica, R., Giordano, P. R., & Chaumette, F. (2014). Active structure from motion: Application to point, sphere, and cylinder. *IEEE Transactions on Robotics*, 30(6), 1499-1513.
- [8] Guerreiro, B. J., Batista, P., Silvestre, C., & Oliveira, P. (2013). Globally asymptotically stable sensor-based simultaneous localization and mapping. *IEEE Transactions on Robotics*, 29(6), 1380-1395.
- [9] Omari, S., & Ducard, G. (2013, July). Metric visual-inertial navigation system using single optical flow feature. In *2013 European Control Conference (ECC)* (pp. 1310-1316). IEEE.
- [10] Skog, I. (2007). *GNSS-aided INS for land vehicle positioning and navigation* (Doctoral dissertation, KTH).
- [11] Jia, Y. (2017). *Estimating the Linear and Angular Velocities of a Free-Flying Object*.
- [12] Hua, M. D., & Allibert, G. (2018, August). Riccati Observer Design for Pose, Linear Velocity and Gravity Direction Estimation using Landmark Position and IMU Measurements. In *2018 IEEE Conference on Control Technology and Applications (CCTA)* (pp. 1313-1318). IEEE.
- [13] Blanchini, F., & Miani, S. (2003). Stabilization of LPV systems: state feedback, state estimation, and duality. *SIAM journal on control and optimization*, 42(1), 76-97.
- [14] Chesi, G., Garulli, A., Tesi, A., & Vicino, A. (2009). *Homogeneous polynomial forms for robustness analysis of uncertain systems* (Vol. 390). Springer Science & Business Media.
- [15] Rosinová, D., & Valach, P. (2014, May). Switched system robust control: Pole-placement LMI based approach. In *Proceedings of the 2014 15th International Carpathian Control Conference (ICCC)* (pp. 491-496). IEEE.
- [16] Bergsten, P., & Palm, R. (2000). Thau-Luenberger observers for TS fuzzy systems. In *Ninth IEEE International Conference on Fuzzy Systems. FUZZ- IEEE*, 2, pp. 671-676.
- [17] Martinelli, A., & Siegwart, R. (2005, August). Observability analysis for mobile robot localization. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1471-1476). IEEE.
- [18] Saska, M. (2015, June). MAV-swarms: unmanned aerial vehicles stabilized along a given path using onboard relative localization. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 894-903). IEEE.
- [19] Benyoucef, R., Nehaoua, L., Hadj-Abdelkader, H., & Arioui, H. (2019, December). Depth Estimation for a Point Feature: Structure from motion. In *IEEE CONFERENCE ON DECISION AND CONTROL* (To appear). IEEE; 2019.



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session IV

Human vehicle interaction

- **Keynote speaker: Cristina Olaverri (Johannes Kepler Universitat, Austria)**

Title: Title: The Effect of Vehicle Automation on Road Safety

- **Round table: Human vehicle interaction**

Participants:

- **Henriette Cornet (TUMCREATE, Singapore)**
- **Li Haizhou (National University of Singapore)**
- **Cristina Olaverri (Johannes Kepler Universitat, Austria)**
- **Juraj Kabzan (Nutonomy, Singapore)**



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Session IV

Keynote speaker: **Cristina Olaverri**
(Johannes Kepler Universitat, Austria)

The Effect of Vehicle Automation on Road Safety

Abstract: The feasibility of incorporating new technology-driven functionality to vehicles has played a central role in automotive design. The overall diffusion in the application of digital technologies presents the possibility of designing systems, the functioning of which is based on intelligent technologies that simultaneously reside in multiple, interconnected applications. Consequently, the development of intelligent road-vehicle systems such as cooperative advanced driver assistance systems (co-ADAS) and with them the degree of vehicle automation is rapidly increasing. The advent of vehicle automation promotes a reduction of the driver workload. However, depending on the automation grade consequences for the passengers such as out-of-the-loop states can be foreseen. Also the protection of Vulnerable Road Users (VRUs) has been an active research topic in recent years. A variety of responses that exhibit several levels of trust, uncertainty and a certain degree of fear when interacting with driverless vehicles has been observed. In this context, P2V (Pedestrian-to-Vehicle) and V2P (Vehicle-to-Pedestrian) have become crucial technologies to minimize potential dangers, due to the high detection rates and the high user-satisfaction levels they achieve. This presentation gives an overview of the impact of such technologies on traffic awareness towards improving driving performance and reducing road accidents. Furthermore, the benefits and potential problems regarding vehicle automation will be outlined.

Biography: Prof. Dr. Cristina Olaverri-Monreal received her PhD from the Ludwig-Maximilians University (LMU) in Munich in cooperation with BMW. After working several years internationally in the industry and in the academia, in her current position as full professor and holder of the BMVIT endowed chair sustainable transport logistics 4.0 at the Johannes Kepler University Linz, in Austria her research aims at studying solutions for an efficient and effective transportation focusing on minimizing the barrier between users and road systems. To this end,



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

she relies on the automation, wireless communication and sensing technologies that pertain to the field of Intelligent Transportation Systems (ITS). Dr. Olaverri is Vice-president of Educational Activities in the IEEE ITS Society Executive Committee and chair of the Technical Activities Committee on Human Factors in ITS. In addition, she serves as an associate editor and editorial board member of several journals in the field, including the IEEE Intelligent Transportation Systems Transactions and the IEEE International Transportation Systems Magazine. She was recently recognized for her dedicated contribution to continuing education in the field of ITS with the 2017 IEEE Educational Activities Board Meritorious Achievement Award in Continuing Education.



The Effect of Vehicle Automation on Road Safety

11th Workshop on Planning, Perception and Navigation of Intelligent Vehicles

2019 IROS 2019
MACAU, CHINA

Univ. Prof. Dr. Cristina Olaverri-Monreal



Vehicular Robots Self-Driving Technologies in Use



- **Drifting warning**
alerts by lane deviation
- **Collision avoidance**
Radar-, laser-, or camera-based
- **Blind-spot detectors**
camera or radar-based
- **Enhanced cruise control**
adapted distance to vehicle ahead
- **Self parking**
camera or sonar-based



1

Driving Task Complexity

Driving Subtasks

Driving Subtasks

- Speed
- Distance
- Steering
- Traffic observation
- Decision making



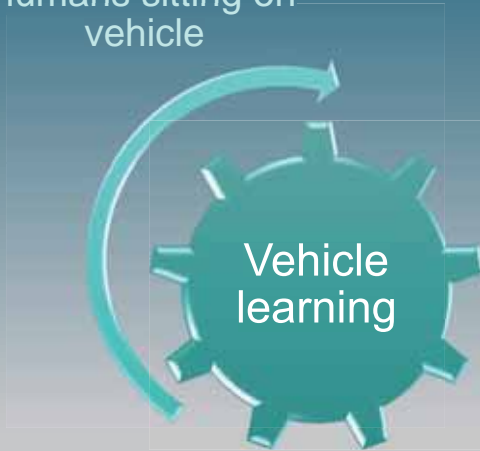
- Car controls
- In-vehicle information
- Traffic signs
- Rules awareness
- Risk evaluation



Driving Instruction for Autonomous Vehicles

By giving examples and training the vehicle

Recording data from humans sitting on vehicle



- Speed
- Driving tasks
 - Longitudinal direction
 - Lateral direction
- All driving subtasks
 - merging into moving traffic,
 - driving in roundabouts
 - negotiating intersections
 - avoiding obstacles
- Interaction with other road users



Driving Instruction for Autonomous Vehicles

- Perception
- Expectations
- Judgement
- Memory
- Planning
- Decision making
- Risk assessment

Human Behavior

Unexpected situations



<https://www.dolanlaw.com/motorcyclists-steer-clear-aggressive-drivers/>

Driving Instruction for Autonomous Vehicles

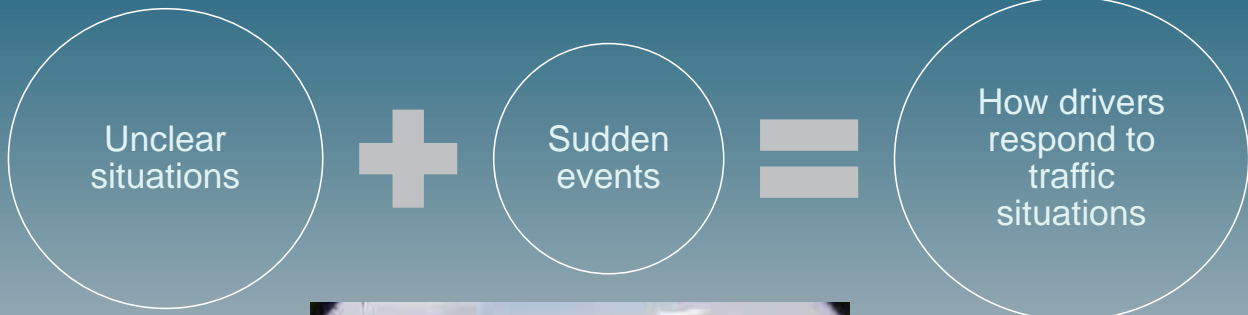


<http://novacriminalattorney.com/2015/05/20/when-the-consequences-in-virginia/>

Emotional artificial intelligence through human monitoring and analysis

- gesture capture
- eye tracking
- head pose
- face detection
- body posture
- voice recognition
- vocal emotion
 - hesitations, frequency, variation, pitch, energy, speed

Driving Instruction for Autonomous Vehicles



Human Errors



<http://www.4autosurinsurancequote.com/uncategorized/top-10-secrets-for-getting-cheap-car-insurance>

Processing and Decision Making



<https://www.fiveaa.com.au/news/SA-Police-Are-Cracking-Down-On-Running-Yellow-Lights>

Information interpretation

CO-ADAS based on V2I



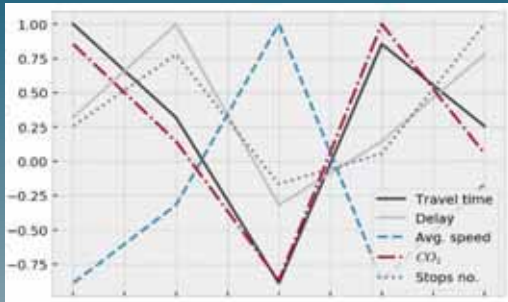
Traffic Light Assistance System

- Retrieves the traffic light timing program to calculate the optimal speed while approaching an intersection
- Shows recommended velocity based on:
 - vehicle's acceleration
 - speed,
 - phase state of traffic light
 - remaining phase duration

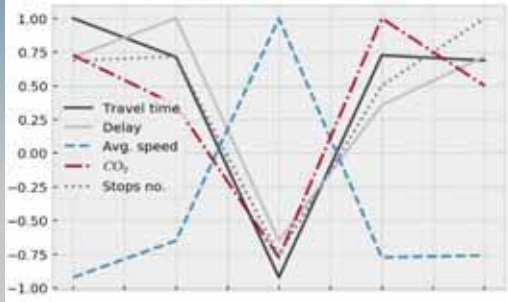
Olaverri-Monreal, C., Enea-Moreno, J., Diaz-Alvarez, A. (2018) "Implementation and Evaluation of a Traffic Light Assistance System in a Simulation Framework based on V2I Communication", Journal of Advanced Transportation, Special Issue "Cooperative Systems for Autonomous Vehicles" (CSAV), Hindawi Publishing Corporation.



CO-ADAS based on V2I



(a) Scenario 1 (with TLA)



Increase in driving efficiency after the drivers adjusted their velocity to the speed calculated by the system

- improvement of traffic flow
- reduced gas emissions
- waiting time at traffic lights

	W/o TLA		W/ TLA		T-Test ($\alpha = 0.05$)	
	Mean	SD	Mean	SD	p	t(24)
Travel time (s)	398	52.57	370.2	43.8	3.19	0.0038**
Delay (s)	108.59	35.82	71.38	26.64	6.82	4.69e-7***
Speed (kmh^{-1})	19.16	2.57	20.4	2.2	-2.88	0.008**
CO ₂ (mg)	411.91	28.12	406.33	22.11	1.19	0.24
Stops no.	13.6	3.88	8.88	4.4	5.66	7.69e-6***

Olaverri-Monreal, C., Errea-Moreno, J., Diaz-Alvarez, A. (2018) 'Implementation and Evaluation of a Traffic Light Assistance System in a Simulation Framework based on V2I Communication', Journal of Advanced Transportation, Special Issue "Cooperative Systems for Autonomous Vehicles" (CSAV), Hindawi Publishing Corporation.

2

Interaction in a Vehicular Environment Communication and Information Flow



Interaction in a Vehicular Environment



Social Traffic

Vehicular units

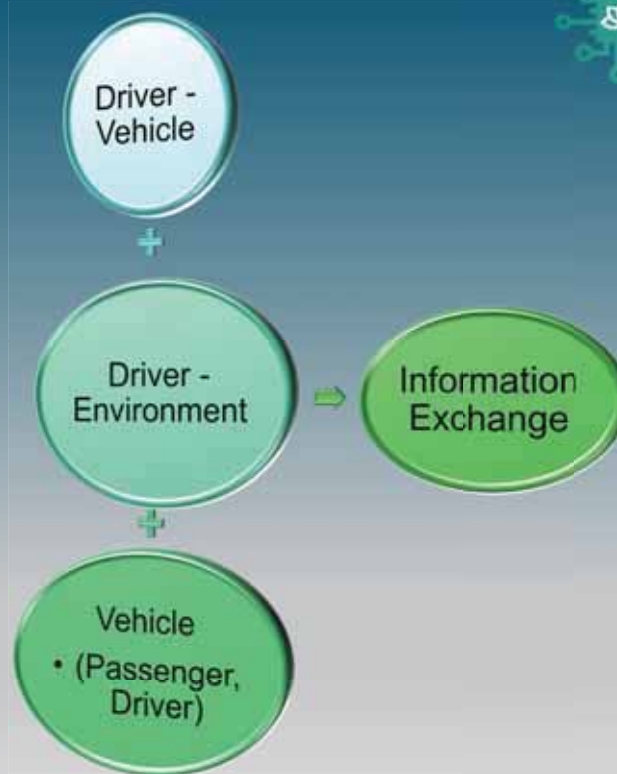
- share space
- take each other into account
- avoid a collision



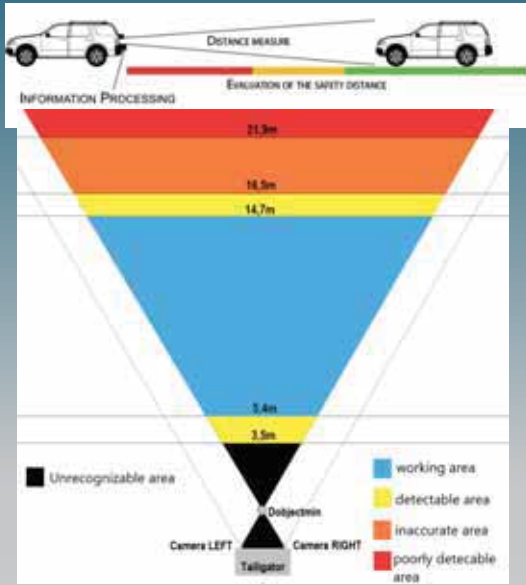
©CristinaOlaverriMonreal

Information Flow

Capacity of attentional resources and demand



Communication while Driving Information Location



Krizek, G. C., Hausleitner, R., Böhme, L., Olaverri-Monreal, C. (2019) "Empirical Analysis of Safe Distance Calculation by the Stereoscopic Capturing and Processing of Images through the Tailgator System" in Sensors Journal

Olaverri-Monreal, C., Krizek, G.C., Michaeler, F., Lorenz, R., Pichler, M. (2018) "Collaborative Approach for a Safe Driving Distance Using Stereoscopic Image Processing", in Future Generation Computer Systems (FGCS) Journal. Special Issue on "Advanced Technologies and Systems for collaboration" <https://doi.org/10.1016/j.future.2018.01.050> | Olaverri-Monreal, C., Lorenz, R., Michaeler, F., Krizek, G., Pichler, M. (2016) "Tailgator: Cooperative System for Safety Distance Observation", Proceedings 2016 International Conference on Collaboration Technologies and Systems, Orlando, Florida, USA, Nov. 2016, pp. 392 - 397.

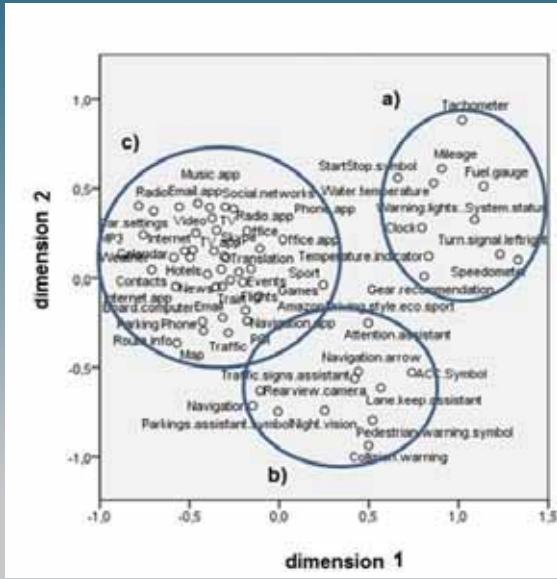
Communication while Driving Information Location



Metric	Baseline		In-Vehicle System		Leading-Vehicle System	
	Mean	SD	Mean	SD	Mean	SD
Velocity (km/h)	31.44	4.85	31.95	3.21	31.34	3.65
Headway (m)	9.87	1.65	12.12	1.71	10.72	1.68
TTC (s)	1.20	0.22	1.46	0.22	1.36	0.25
LP (m)	0.52	0.09	0.50	0.12	0.50	0.07
Deceleration change rate ((km/h)/s)	0.43	0.17	0.62	0.29	0.60	0.28
T-Test ($\alpha=0.05$)						
Metric	Baseline vs. In-Vehicle		Baseline vs. Leading-Vehicle System		In-Vehicle vs. Leading-Vehicle System	
	n(19)	p	n(19)	p	n(19)	p
Velocity (km/h)	-0.41	0.69	0.07	0.94	0.68	0.50
Headway (m)	-4.90	9.93E-05***	-2.32	0.031*	2.87	0.009**
TTC (s)	-5.33	3.81E-05***	-1.88	0.07	1.07	0.29
LP (m)	0.66	0.52	0.44	0.66	-0.19	0.84
Deceleration change rate ((km/h)/s)	-2.90	0.009***	-2.35	0.02*	0.30	0.76

Olaverri-Monreal, C., Gvozdic, M., Muthurajan, B. (2017) "Effect on Driving Performance of Two Visualization Paradigms for Rear-End Collision Avoidance", Proceedings IEEE Intelligent Transportation Systems Conference, Yokohama, Japan, October 2017, pp. 37-42.

Communication while Driving Information Location

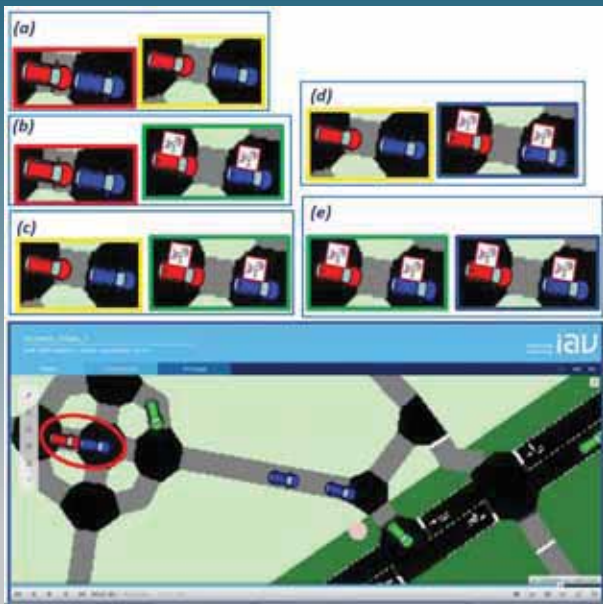


Claverri-Monreal, C., Hasan, A., Bulut, J., Körber, M., Bengler, K. (2014) "Impact of In-Vehicle Displays Location Preferences on Drivers' Performance and Gaze", IEEE Transactions on Intelligent Transportation Systems (IEEE T-ITS), Special Issue on "Human Factors in Intelligent Vehicles", Volume 15, Issue 4, pp. 1770 – 1780 | Claverri-Monreal, C., Lehning, C., Trübswetter, N., Schepp, C. A., Bengler, K. (2013) "In-Vehicle Displays: Driving Information Prioritization and Visualization", Proceedings IEEE Intelligent Vehicles Symposium, Gold Coast, Australia, pp. 660 – 665.

3

Connected and Automated Technologies Driving and Interaction with other Road Users

V2X and ADAS effect on Road Safety

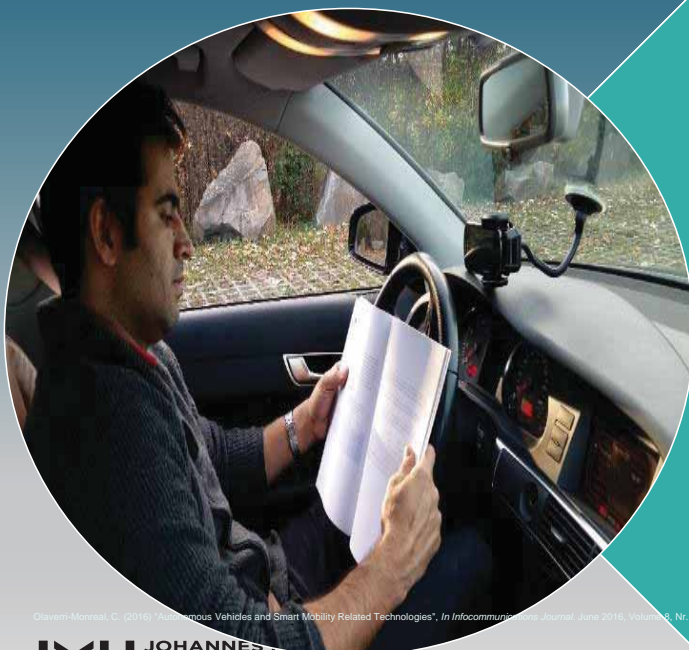


Technologies and applications that use data collected by sensors located in other vehicles, infrastructure or road users (V2X) to assist the driver

Key step toward a significant reduction of accidents

Validi, A., Ludwig, T., Olaverri-Monreal, C. (2017) "Analyzing the Effects of V2X and ADAS-ACC Penetration Rates on the Level of Road Safety in Intersections: Evaluating Simulation Platforms SUMO and Scene Suite", IEEE 2017 International Conference on Vehicular Electronics and Safety (ICVES 2017), Vienna, Austria, pp. 38 – 43
 Validi, A., Ludwig, T., Hussein, A., Olaverri-Monreal, C. (2018) "Examining the Impact on Road Safety of Different Penetration Rates of Vehicle-to-Vehicle Communication and Adaptive Cruise Control", IEEE Intelligent Transportation Systems Magazine

Automation



Autonomous vehicles represent an opportunity to continue working for increased road safety

- Human Intervention not required
- Congestion and air pollution reduction (platooning)
- Automation will be in charge of driving sub tasks

Olaverri-Monreal, C. (2016) "Autonomous Vehicles and Smart Mobility Related Technologies", In *Infocommunications Journal*, June 2016, Volume 8, Nr. 2, pp. 17-24.

Automation Levels



SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/Deceleration	Monitoring of Driving Environment	Fallback Performance of Dynamic Driving Task	System Capability (Driving Modes)
Human driver monitors the driving environment						
0	No Automation	The full-time performance by the human driver of all aspects of the dynamic driving task, even when enhanced by warning or intervention systems.	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	The driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the dynamic driving task.	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	The driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the dynamic driving task.	System	Human driver	Human driver	Some driving modes
Automated driving system ("system") monitors the driving environment						
3	Conditional Automation	The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task with the expectation that the human driver will respond appropriately to a request to intervene.	System	System	Human driver	Some driving modes
4	High Automation	The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task, even if a human driver does not respond appropriately to a request to intervene.	System	System	System	Some driving modes
5	Full Automation	The full-time performance by an automated driving system of all aspects of the dynamic driving task under all roadway and environmental conditions that can be managed by a human driver.	System	System	System	All driving modes

Adapted from SAE INTERNATIONAL'S LEVELS OF DRIVING AUTOMATION FOR ON-ROAD VEHICLES, Issued January 2014, http://www.sae.org/misc/pdfs/automated_driving.pdf

Take Over Request (Level 3)

- Construction zones, orange signs, cones etc.
- Railroad crossing
- Cars stopped in the road
- Pedestrians, cyclists
- Narrow paths (mountain roads)
- Lidar problems due to sunlight
- Interferences with other electronic devices



Allamehzadeh, A., Olaverri-Monreal, C. (2016) "Automatic and Manual Driving Paradigms: Cost-Efficient Mobile Application for the Assessment of Driver Inattentiveness and Detection of Road Conditions", Proceedings IEEE Intelligent Vehicles Symposium, IV '16, Gothenburg, Sweden, June 19-22, pp 26-31.

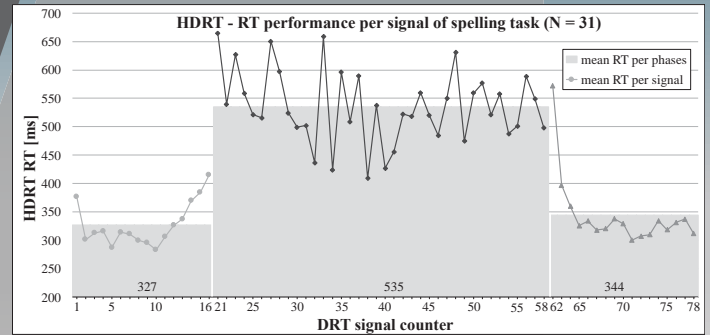
Take Over Request (Level 3)



- Hit Detection Response Task (HDRT) performance prior to and after a spelling task per signal and phases (before, during and post) spelling task (N = 31)
- It takes about 0.8 seconds for a driver to shift attention so that their eyes are on the road
- It takes even longer to assess the situation to make a helpful response.
- Drivers can be distracted up to 27 seconds after finishing a “highly distracting task and up to 15 seconds after a moderate one”



<http://www.nationaltruckdrivingjobs.com/node/362>



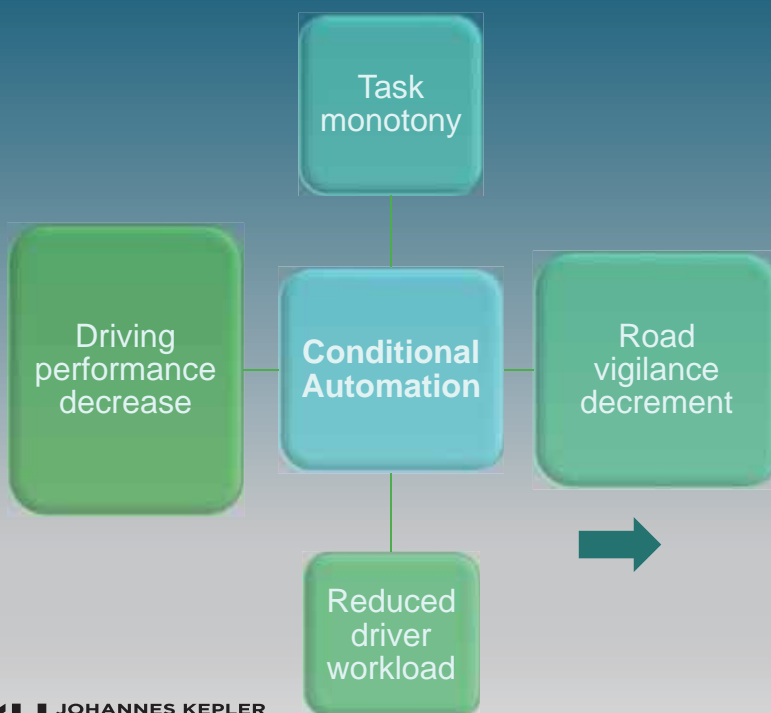
Winzer, O. M., Conti, A. S., Olaverri-Monreal, C., Bengler, K. (2017) 'Modifications of driver attention post-distraction: a detection response task study', In: Nah FH, Tan CH, (eds) HCI in Business, Government and Organizations. Interacting with Information Systems. HCIBGO 2017. Lecture Notes in Computer Science, vol 10293. Springer, Cham

Cristina Olaverri-Monreal - Keynote PPNIV / IROS 2019 23



Strayer, D., Cooper, J., and Siegel, L., "Up to 27 seconds of inattention after talking to your car or smartphone: Distraction rated 'high' for most devices while driving." 2015.

Hypovigilance (Level 3)



Allamehazadeh, A., Olaverri-Monreal, C. (2016) 'Automatic and Manual Driving Paradigms: Cost-Efficient Mobile Application for the Assessment of Driver Inattentiveness and Detection of Road Conditions', Proceedings IEEE Intelligent Vehicles Symposium, IV '16, Gothenburg, Sweden, June 19-22, pp 26-31



Cristina Olaverri-Monreal - Keynote PPNIV / IROS 2019 24



Hypovigilance (Level 3)



- Luminescence-based unobtrusive method
- Relying on peripheral vision, which is processed subconsciously

Tendency to respond faster to a TOR when peripheral vision detected stimulus

Capalar, J. and Olaverri-Monreal, C. (2017) "Hypovigilance in Limited Self-Driving Automation: Peripheral Visual Stimulus for a Balanced Level of Automation and Cognitive Workload", Proceedings IEEE Intelligent Transportation Systems Conference, Yokohama, Japan, October 2017, pp. 14-18.

Take Over Request (Level 3)

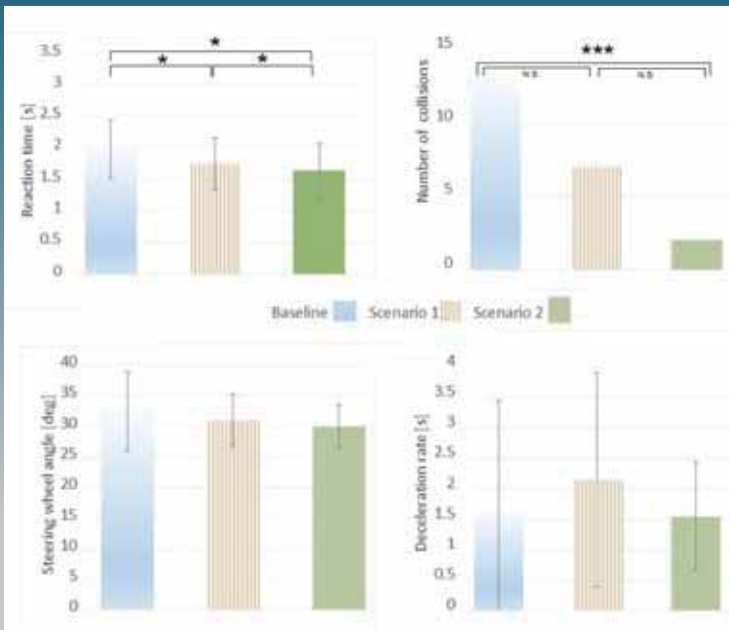


- Baseline conditions: No IDCS system activated
- Scenario 1: TOR display activated
- Scenario 2: IDCS informs the driver through the three display panels

Olaverri-Monreal, C., Kumar, S., Diaz-Alvarez, A. "Automated Driving: Interactive Automation Control System to Enhance Situational Awareness in Conditional Automation". IEEE Intelligent Vehicles Symposium 2018.



Take Over Request (Level 3)



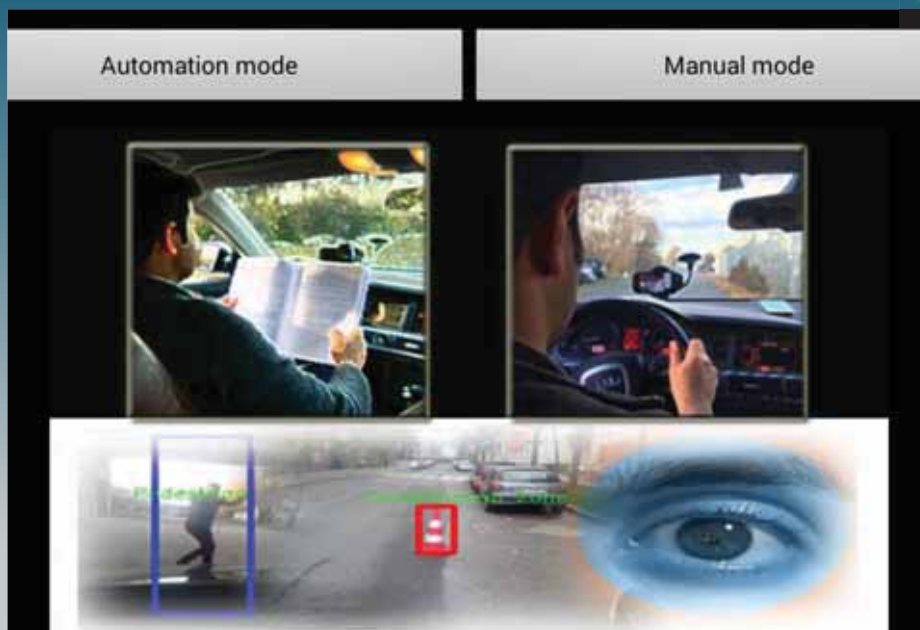
- Baseline conditions: No IDCS system activated
- Scenario 1: TOR display activated
- Scenario 2: IDCS informs the driver through the three display panels

Significant decrease in

- response time to TOR
- number of collisions

Olaverri-Monreal, C., Kumar, S., Diaz-Alvarez, A. "Automated Driving: Interactive Automation Control System to Enhance Situational Awareness in Conditional Automation". IEEE Intelligent Vehicles Symposium 2018.

Take Over Request (Level 3)

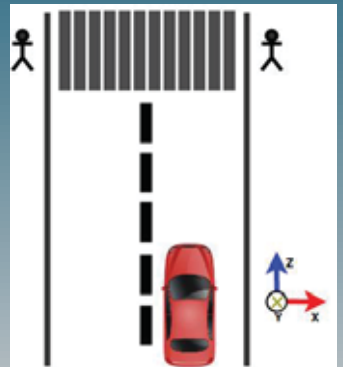


Allamehzadeh, A., Olaverri-Monreal, C. (2016) "Automatic and Manual Driving Paradigms: Cost-Efficient Mobile Application for the Assessment of Driver Inattentiveness and Detection of Road Conditions". Proceedings IEEE Intelligent Vehicles Symposium, IV '16, Gothenburg, Sweden. June 19-22, pp 26-31.

Vulnerable Road Users (VRU)



Pedestrian detection by pose estimation using OpenPose



Pedestrians crossing from both sides

Morales-Alvarez, W., Gomez-Silva, M. J., Fernandez-Lopez, G., Garcia-Fernandez, F., Olaverri-Monreal C. (2018) "Automatic Analysis of Pedestrian's Body Language in the Interaction with Autonomous Vehicles", Proceedings IEEE Intelligent Vehicles Symposium 2018, Changshu, China.

VRUs Response to Autonomous Vehicles



22 videos documented pedestrian behavior of 49 pedestrians in a marked crosswalk

Vehicle equipped with optical wheel encoders, a stereo-vision camera, a laser-range finder, a compass and GPS sensor

Morales Alvarez, W., de Miguel, M.A., Garcia, F., Olaverri-Monreal C. (2019) Analysis of VRUs response to communication signals from autonomous vehicles, IEEE ITS 2019 Auckland, New Zealand
de Miguel, M.A., Fuchshuber, D., Hussein, A., Olaverri-Monreal, C. (2019) Perceived Pedestrian Safety: Public Interaction with Driverless Vehicles, IEEE IV 2019, Paris, France

VRUs Response to Autonomous Vehicles



- Many pedestrians involved in the manipulation of their smartphones
- Vehicle attracted attention and curiosity
- Testing whether the vehicle really stopped
- Some pedestrians did not dare to approach it
- Uncertainty → hesitation before crossing

Morales Alvarez, W., de Miguel, M.A., Garcia, F., Olaverri-Monreal, C. (2019) Analysis of VRUs response to communication signals from autonomous vehicles, IEEE ITSC 2019 Auckland, New Zealand | de Miguel, M.A., Fuchshuber, D., Hussein, A., Olaverri-Monreal, C. (2019) Perceived Pedestrian Safety: Public Interaction with Driverless Vehicles, IEEE IV 2019, Paris, France

VRUs Response to Autonomous Vehicles



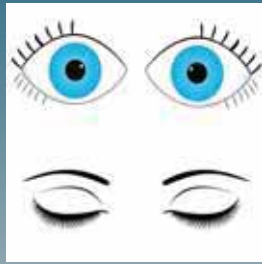
Crossing uncertainty due to:

- lack of knowledge about detection
- whether the vehicle was going to slow down
- not trusting the functioning of the sensors

Morales Alvarez, W., de Miguel, M.A., Garcia, F., Olaverri-Monreal, C. (2019) Analysis of VRUs response to communication signals from autonomous vehicles, IEEE ITSC 2019 Auckland, New Zealand | de Miguel, M.A., Fuchshuber, D., Hussein, A., Olaverri-Monreal, C. (2019) Perceived Pedestrian Safety: Public Interaction with Driverless Vehicles, IEEE IV 2019, Paris, France



VRU Response to Communication Signals from Autonomous Vehicles



70% of people preferred the eyes image compared with 30% that selected the color coded.

No statistically significant differences ($2(1, N = 21) = 7.54, p = .006$).

Morales Alvarez, W., de Miguel, M.A., Garcia, F., Olaverri-Monreal, C. (2019) Analysis of VRUs response to communication signals from autonomous vehicles, IEEE ITSC 2019 Auckland, New Zealand

de Miguel, M.A., Fuchshuber, D., Hussein, A., Olaverri-Monreal, C. (2019) Perceived Pedestrian Safety: Public Interaction with Driverless Vehicles, IEEE IV 2019, Paris, France

VRU Response to Communication Signals from Vehicular Robots

Pedestrian distance to the vehicle as well as TTC while crossing depending on the kind of display showed

Metric	Baseline		Red/green color		Opened/closed eyes	
	Mean	SD	Mean	SD	Mean	SD
Distance(m)	6.14	3.56	7.38	3.48	6.88	2.76
TTC (s)	7.31	4.64	4.9	6.23	5.10	7.91

T-test ($\alpha = 0.05$)

Metric	Baseline vs. Red/green color		Baseline vs. Opened/Closed eyes		Red/green color vs. Opened/Closed eyes	
	t(92)	p	t(92)	p	t(92)	p
Distance(m)	1.27	0.20	0.85	0.39	0.67	0.55
TTC (s)	1.51	0.13	1.07	0.28	0.90	0.12

Distributions of walked or stopped variables with red or closed eyes not statistically significant



The kind of display did not affect TTC neither the distance at which pedestrians crossed in front of the AV



Visual communication cues for are not necessarily required for a shared space in which informal traffic rules apply

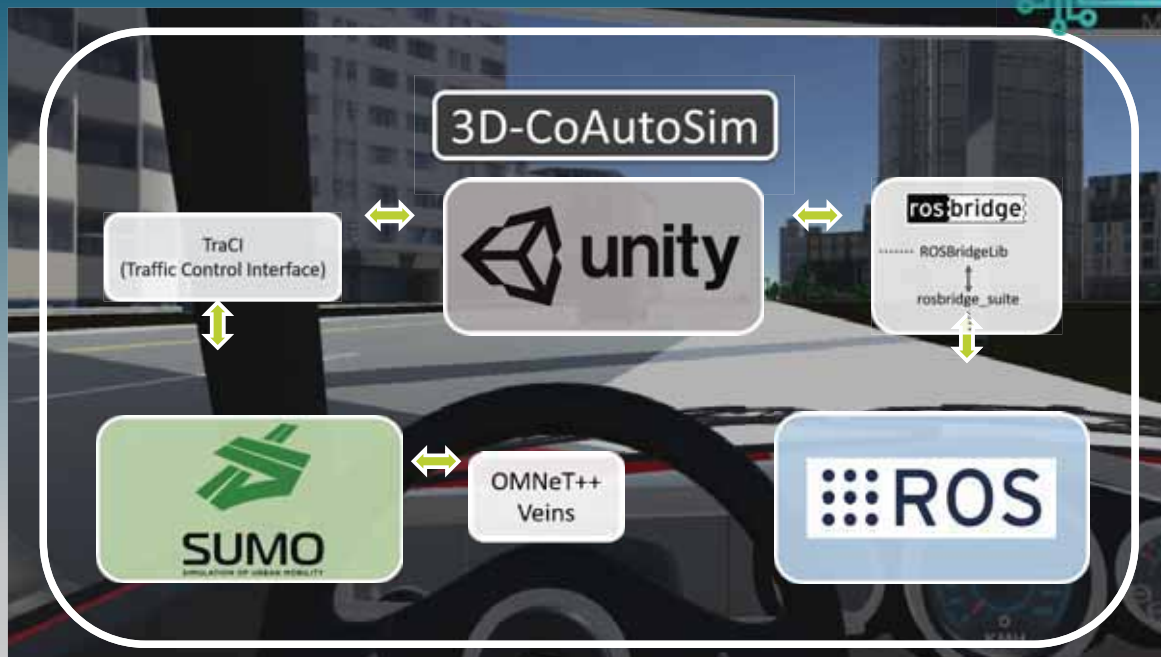
Effect of eye contact on interaction with the AV

T-test ($\alpha = 0.05$)

Metric	Without eye contact		Eye contact		T-Test($\alpha = 0.05$)	
	Mean	SD	Mean	SD	t(133)	p
Distance (m)	6.93	3.28	7.81	3.56	1.41	0.1599
TTC (m/s)	5.87	6.71	8.93	12.22	1.37	0.1726

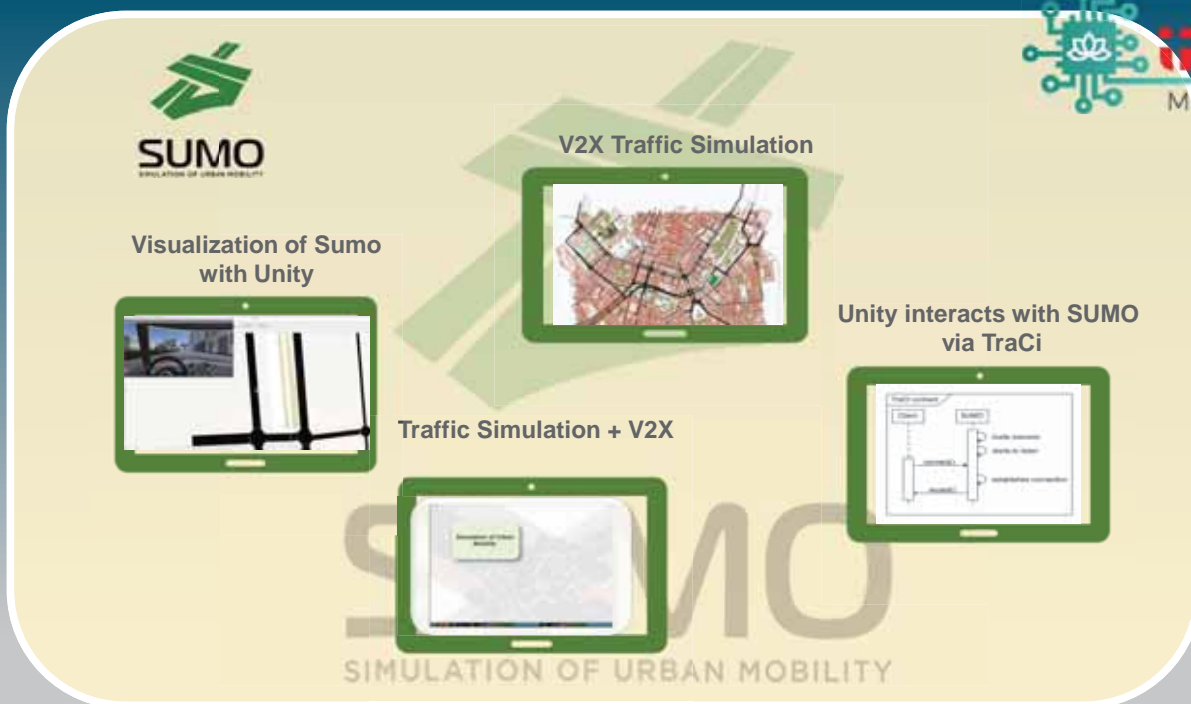
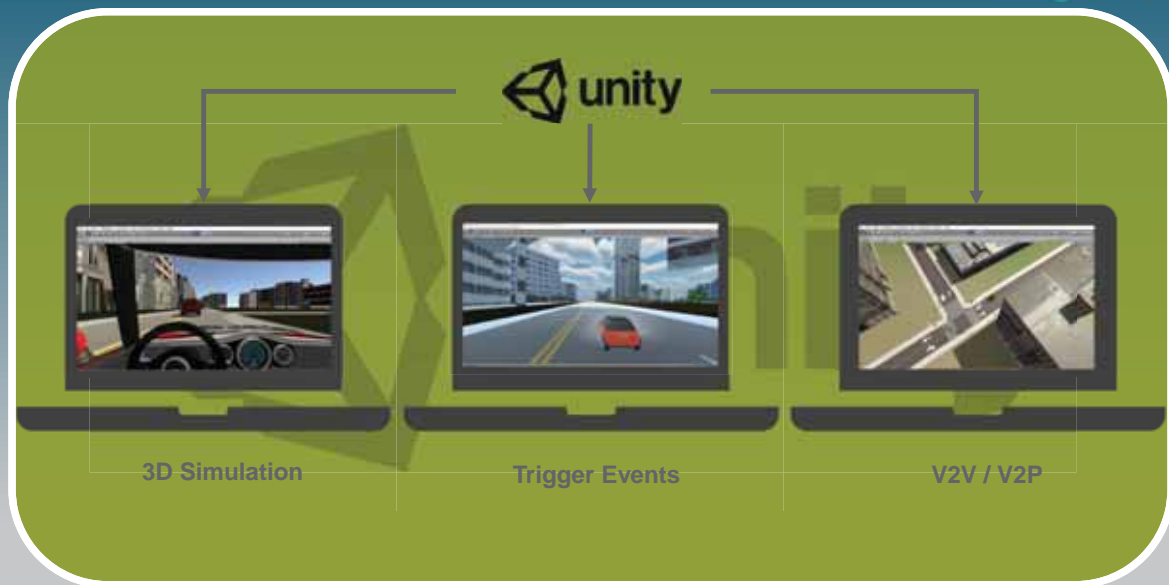
Morales-Alvarez, W., Gomez-Silva, M. J., Fernandez-Lopez, G., Garcia-Fernandez, F., Olaverri-Monreal, C. (2018) 'Automatic Analysis of Pedestrian's Body Language in the Interaction with Autonomous Vehicles', Proceedings IEEE Intelligent Vehicles Symposium 2018, Changshu, China.

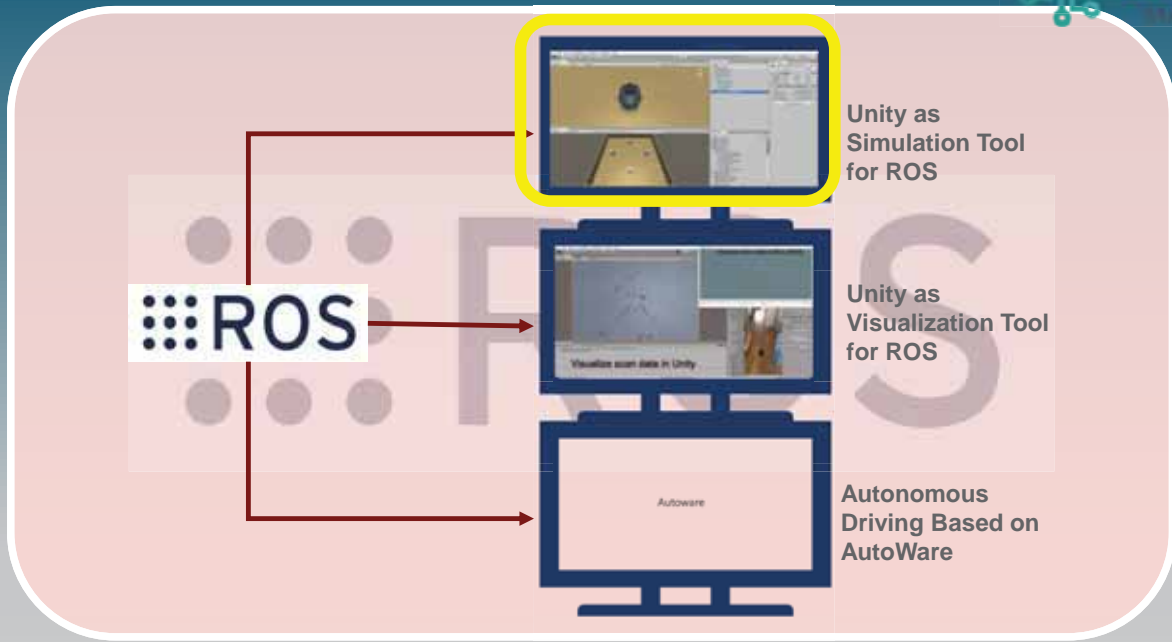
Morales Alvarez, W., de Miguel, M.A., Garcia, F., Olaverri-Monreal, C. (2019) Analysis of VRUs response to communication signals from autonomous vehicles, IEEE ITSC 2019 Auckland, New Zealand



Hussein, A., Diaz-Avarez, A., Armingol, J. M., and Olaverri-Monreal, C. (2018) "3D-CoAutoSim: Simulator for Cooperative ADAS and Automated Vehicles". Proceedings 21st International IEEE Conference on Intelligent Transportation Systems, ITSC2018, Hawaii, November 2018. | Artal-Villa, L., Hussein, A., Olaverri-Monreal, C. (2019) "Extension of the 3D-CoAutoSim to Simulate Vehicle and Pedestrian Interaction based on SUMO and Unity 3D". Proceedings IEEE Intelligent Vehicles Symposium 2019, Paris, France. | Michaeler, F. and Olaverri-Monreal, C. (2017) "3D Driving Simulator with VANET Capabilities to Assess Cooperative Systems: 3DSimVanet". Proceedings IEEE Intelligent Vehicles Symposium, Los Angeles, USA, June 2017, pp. 999-1004.







Thank You!



Univ.-Prof. Dr. Cristina Olaverri-Monreal
cristina.olaverri-monreal@jku.at



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Round table

Human vehicle interaction

Henriette Cornet (TUMCREATE, Singapore)

Li Haizhou (National University of Singapore)

Cristina Olaverri (Johannes Kepler Universitat, Austria)

Juraj Kabzan (Nutmomy, Singapore)



2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

Design for Autonomous Mobility

Dr Henriette Cornet

Principal Investigator 'Design for Autonomous Mobility'

henriette.cornet@tum-create.edu.sg

04.11.2019



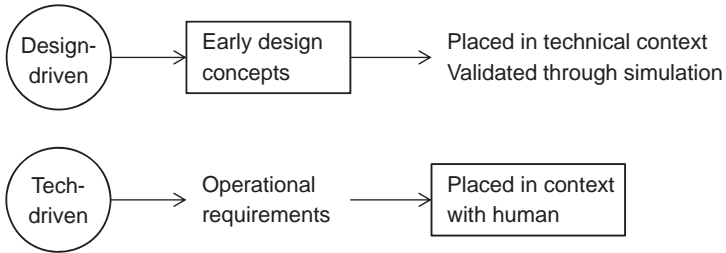
TUMCREATE's mission: Improve the travel experience for people of Singapore



Design for Autonomous Mobility

Human-Machine Interfaces:

- AV to pedestrians communication → Safety
- AV to passengers communication → Trust
- Robot and Virtual Companion for customer care → Trust



3

Two human-centred approaches: Emotional Design



Virtual Design Lab



4

Providing an Interface for Humans to Understand Machines

Haizhou Li (李海洲)

Dept of Electrical and Computer Engineering, NUS



1

Understanding Artificial Intelligence



	Humanly	Rationally
Thinking	Associate with human thinking, problem solving, decision making	The study of mental faculties through computational models
Acting	Turing Test (pattern classification, dialogue, translation)	Systems that act rationally

2

Virtual Bus Captain

ST Engineering launches 12m-long driverless bus; public trials to start on Jurong Island next year. *The Straits Times*-22 Oct 2019

THE STRAITS TIMES



November 04, 2019

nuTonomy - APTIV

Juraj Kabzan
APTIV Autonomous Mobility
juraj.kabzan@aptiv.com



About Me

- BSc. from TU Wien (Electrical Engineering)
- MSc. from ETH Zurich (Robotics Systems, and Control)



About Me

- I am a Research Engineer in Chief Scientist Office
Focus on planning, control and their fusion with ML



3

Juraj Kabzan | 04.11.2019

• APTIV •