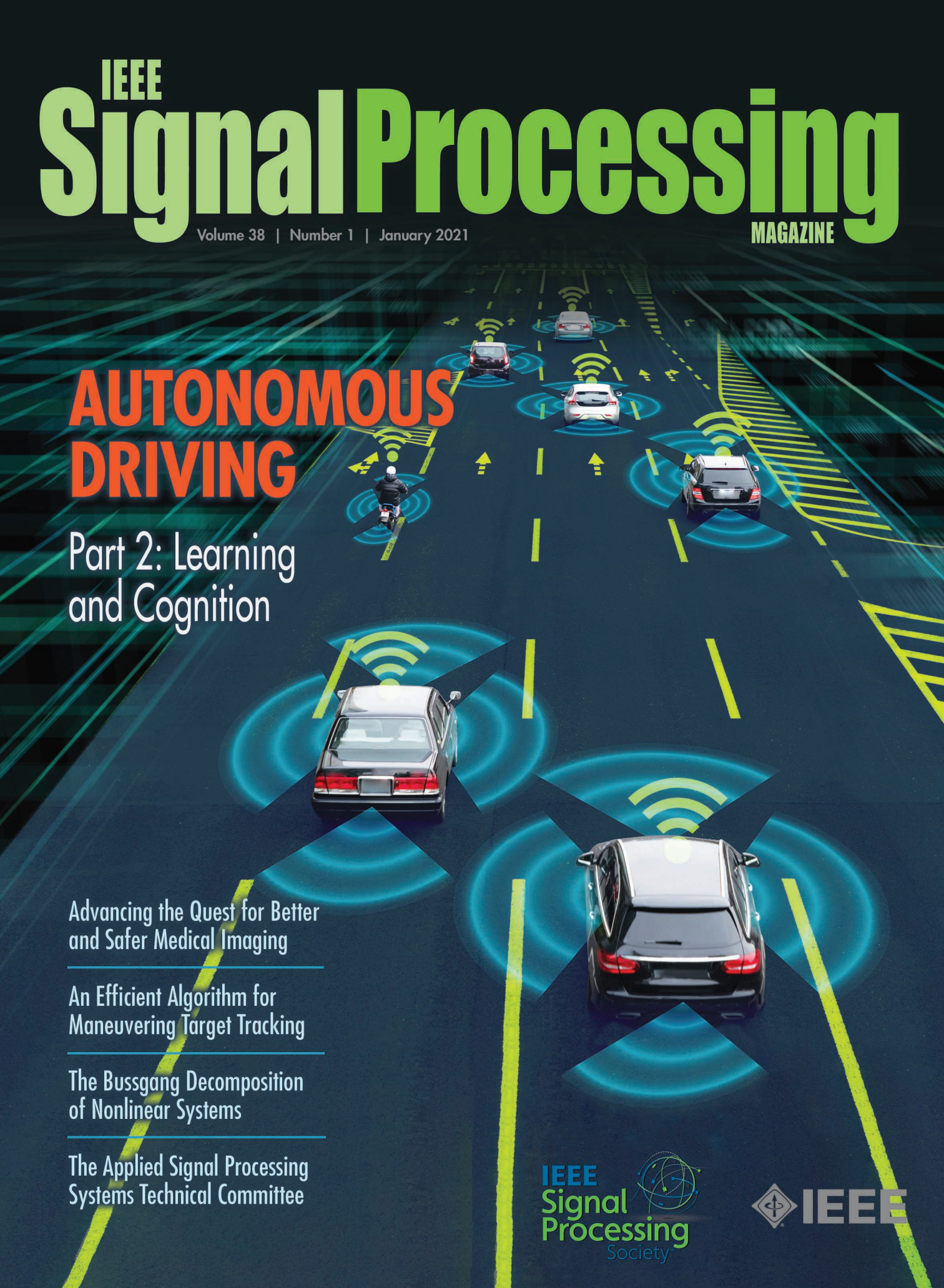
IEEE
Signal Processing
MAGAZINE
Volume 38 | Number 1 | January 2021



AUTONOMOUS DRIVING

Part 2: Learning and Cognition

IEEE Signal Processing Society

IEEE

Call for Papers

Signal Processing Magazine Special Issue on

Explainability in Data Science: Interpretability, Reproducibility, and Replicability

The growing importance of the closely-related fields of data science, machine learning, and signal processing highlights the fact that data-driven solutions are playing an increasing role in practical problems across many domains. With the shift away from traditional and model-driven approaches, we start to recognize the importance of explainability of our solutions. Explainability is critical not only for the simple reason that one would like to have confidence over the solutions, but also because one would like to produce further insights from the solutions. This includes interpretability and completeness so that one can not only "audit" them, but also ask appropriate questions to probe for insights beyond the initial solution.

Interpretability, i.e., ability to attach a physical meaning to the solution, along with reproducibility, and replicability are three key aspects of explainability. Following the definitions by the National Academies of Sciences, Engineering, and Medicine, reproducibility refers to obtaining consistent results using the same data and code—i.e., method,—as the original study, and replicability is obtaining consistent results across studies aimed at answering the same scientific question using new data or other computational methods.

This special issue emphasizes

- the broad view of explainability in data science, which not only addresses neural networks but also many other data-driven solutions including, but not limited to, graphical and kernel methods, matrix and tensor factorizations, learning models and methods for multiple application domains;
- the relationship of these methods and their varying degrees of explainability within and across application domains;
- all three topics under the broad umbrella of explainability: interpretability, (computational) reproducibility, and replicability;
- the need to respect history as the discussions on explainability, reproducibility, replicability, validation, and generalization are closely related and are not new but have been explored before.

We invite overviews in explainability that address multiple aspects of interpretability, reproducibility, and replicability including the closely-related concepts of robustness, validation and generalization across

- an array of data-driven methods in machine learning and signal processing, and
- application areas including audio, speech, and language processing, multi-sensor processing and data fusion, image and video processing, computer vision, computational imaging, communications, medical image analysis, bio-imaging, biomedical signal processing, autonomous systems, smart cities, and natural sciences among others.

Papers that provide comparative studies, best practices to obtain insights through explainability, and a critical view of the state-of-the-art are especially welcome.

White papers are required, and full articles will be invited based on the review of white papers. The white paper format is up to 4 pages in length, including the proposed title, motivation and significance of the topic, an outline of the proposed paper, and representative references. An author list with contact information and short bios should also be included.

Submitted articles must be of tutorial/overview/survey nature, written in an accessible style to a broad audience, and have a significant relevance to the scope of the Special Issue. Submissions must not have been published or be under review elsewhere, and must be made online through https://mc.manuscriptcentral.com/sps-ieee.
For submission guidelines, please see the IEEE Signal Processing Magazine Information for Authors page.

Schedule

White paper due: March 22, 2021
Invitation notification: April 26, 2021
Manuscript due: July 5, 2021
First review to authors: September 6, 2021
Revision due: November 8, 2021
Acceptance notification: January 11, 2022
Final package due: February 1, 2022

Guest Editors

Tülay Adalı, University of Maryland, Baltimore County, Baltimore, MD, USA
Rodrigo Capobianco Guido, São Paulo State University, SP, Brazil
Tin Kam Ho, IBM Watson Health, Yorktown Heights, NY, USA
Klaus-Robert Müller, TU Berlin, Germany and Korea University, Seoul, South Korea
Stephen Strother, Baycrest Hospital and University of Toronto, ON, Canada

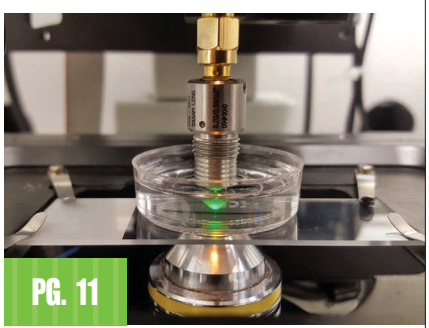
Digital Object Identifier 10.1109/MSP.2020.3038346

Contents

Volume 38 | Number 1 | January 2021

SPECIAL SECTION

AUTONOMOUS DRIVING: PART 2



PG. 11

ON THE COVER

This special issue on autonomous driving is Part 2: Learning and Cognition, which continues from Part 1: Sensing and Perception that was published in the July 2020 issue of *IEEE Signal Processing Magazine*.

COVER IMAGE: ©SHUTTERSTOCK.COM/METAMORWORKS

COLUMNS

PG. 139

IEEE SIGNAL PROCESSING MAGAZINE (ISSN 1053-5888) (ISPREG) is published bimonthly by the Institute of Electrical and Electronics Engineers, Inc., 3 Park Avenue, 17th Floor, New York, NY 10016-5997 USA (+1 212 419 7900). Responsibility for the contents rests upon the authors and not the IEEE, the Society, or its members. Annual member subscriptions included in Society fee. Nonmember subscriptions available upon request. **Individual copies:** IEEE Members US$20.00 (first copy only), nonmembers US$248 per copy. Copyright and Reprint Permissions: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of U.S. Copyright Law for private use of patrons: 1) those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923 USA; 2) pre-1978 articles without fee. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. **For all other copying, reprint, or republication permission,** write to IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854 USA. Periodicals postage paid at New York, NY, and at additional mailing offices. **Postmaster:** Send address changes to IEEE Signal Processing Magazine, IEEE, 445 Hoes Lane, Piscataway, NJ 08854 USA. Canadian GST #125634188 **Printed in the U.S.A.**

Digital Object Identifier 10.1109/MSP.2020.3028252

DEPARTMENTS

©SHUTTERSTOCK.COM/R.M. NUNES

The 2021 IEEE International Conference on Autonomous Systems is scheduled to be held 11–13 August in Montréal, Québec, Canada.

IEEE Signal Processing Magazine

Digital Object Identifier 10.1109/MSP.2020.3028253

SCOPE: *IEEE Signal Processing Magazine* publishes tutorial-style articles on signal processing research and applications as well as columns and forums on issues of interest. Its coverage ranges from fundamental principles to practical implementation, reflecting the multidimensional facets of interests and concerns of the community. Its mission is to bring up-to-date, emerging, and active technical developments, issues, and events to the research, educational, and professional communities. It is also the main Society communication platform addressing important issues concerning all members.

Christian Jutten | Editor-in-Chief | Christian.Jutten@grenoble-inp.fr

Starting a Three-Year Journey With *IEEE Signal Processing Magazine*

First, I would like to wish you a happy New Year and, especially, health for you and your families. I am very honored to be the new editor-in-chief (EIC) of *IEEE Signal Processing Magazine* (*SPM*) for the next three years. It is a great challenge for me, as it was probably for its previous EICs since *SPM* is not an ordinary magazine.

Let me introduce myself. My teenage dreams were driven by a few scientific and technological events. In particular, I remembered the first heart transplant by Barnard in 1967 and then Armstrong's first steps on the moon in 1969, the missions of the rockets *Soyuz* and the beginning of Ariane in Europe, and the first prototype of the fast train (TGV) in France. I was also passionate about electronics and built simple devices for fun. Naturally, I decided to follow scientific studies. But I hesitated between various topics that were all equally appealing to me: medicine, astronomy, physics, and electronics. Finally, I graduated from a higher school of electrical engineering in Grenoble. And I discovered there, in 1976, the basics of signal processing and digital signal processing (with Oppenheim's book, just published in 1975): it was love at first sight. In fact, very quickly, I understood, with my professors, the magic—although usually hidden—position of signal processing in many scientific and technological domains. I enjoyed the virtuous circle of the signal processing approach: starting from an actual problem, propose a mathematical model, design an algorithm, and validate the algorithm on the actual problem. Then, refine the model and the algorithm, and so on, if the results are not good enough. For me, signal processing was the opportunity to address problems in different domains, of course, while working with experts of these domains. So during my career, I have had the chance to work on very exciting problems related to multiple disciplines, including neuroscience, medicine, astrophysics, and chemical engineering.

> **I realize how some face-to-face encounters may have a strong impact on one's life, although it is usually much later we realize those impacts.**

During my Ph.D. studies, I worked under the supervision of Prof. Jeanny Hérault on artificial neural networks, which were actually based on electronic neurons. At that time, computers were indeed not powerful enough for doing large simulations. To give you an idea, in the early 1980s, I had access for about 2 h/day to a 16-bit computer, but the random-access memory was limited to 24 kilobytes, and that is both for algorithms and data! Artificial neural networks were at their infancy, especially given the limited computational power. Moreover, at that time, presenting a paper on this subject in signal processing conferences or journals was quite challenging since you passed for an alien! Hence, in 1982, with Prof. Hérault, we created a French interdisciplinary working group, called *Neurosciences and Engineering Sciences*, in which neuroscientists, biologists, physicists, mathematicians, and engineers met every other year during one week to discuss their works. Interestingly, some very famous scientists in machine learning, e.g., Yann Le Cun, Isabelle Guyon, and Léon Bottou, participated in some of these meetings. It was during one of these meetings that we met and discussed with neuroscientists working on vertebrate motion decoding. And these discussions were the starting point of blind source separation and independent component analysis, which dominated my scientific life. When I think of these discussions, I realize how some face-to-face encounters may have a strong impact on one's life, although it is usually much later we realize those impacts. Currently, with the COVID-19 pandemic, even if virtual conferences may offer some advantages, like reaching a larger audience, I believe that they cannot replace face-to-face discussions, with eyes that meet and friendly moments.

Now, machine learning is "in fashion" in signal and image processing as it is in many other fields. Besides other machine learning concepts, deep neural networks have been attracting much

Digital Object Identifier 10.1109/MSP.2020.3031649
Date of current version: 24 December 2020

attention as they, quite often, produce impressive results given large amounts of data and in a more or less blind fashion. But blindly following trends in research is dangerous [1]. So although usually powerful, results obtained by deep learning methods for solving difficult problems also raise many relevant and fundamental issues for being smartly used. First, we have the output uncertainty and robustness with respect to small input changes. Second, we have the role of the different layers and relationships among parameters. Third, we have bias provided by the data themselves since networks consider mainly what is statistically relevant and ignore rare data. You are able to see a yellow sailboat in the middle of the blue ocean, but for a machine learning algorithm, everything is blue. Such topics on explainability in data science will be the core of a special issue in *SPM*; see the inside cover of this issue. In addition to these problems, there is a social and environmental issue with machine learning approaches: Can we promote methods based on huge data and running greedy algorithms on exascale supercomputers including millions of CPU/GPU cores? What is the ecological cost of all machine learning studies? I believe that a very interesting issue for many scientists, and especially scientists in signal and image processing, is to wonder how we can reuse previous learning results and extrapolate them on new data: the problem of transfer learning is actually a very hot and promising direction of research. More generally, I believe that the question of the computational cost—and thus the ecological impact—of algorithms deserves to be taken into account at the design level, especially for heavily used algorithms.

During IEEE ICIP 2020, I attended an exciting workshop called Promoting Diversity in Signal Processing (also known as PROGRESS). In addition to gender diversity, other diversities were addressed during this workshop. One talk was about diversity in teaching and textbook design. The pandemic has also imposed the opportunity to do and test other ways of teaching. Concerning the low percentage of females in our domain in most universities, it seems that a solution could be to raise the interest of girls in high school and perhaps earlier: experiences in such scientific mediation activities at different levels could be imitated in other places to increase the percentage of female researchers and educators in signal processing. Since many of us are both researchers and professors, I believe it is interesting to share experiences and advice. Such articles have a place in the columns of *SPM*.

During the PROGRESS workshop, the questions of ethical research and scientific integrity were also addressed. During the recent months of the pandemic, we have observed some examples of researcher behaviors, mainly in medical research, concerning questionable experiments and publications about some of the effects of drugs against COVID-19. I believe that promoting and doing ethical and honest research is essential for three reasons. First, science is focused on improving knowledge, and there is no room for cheaters and liars in science. Second, it is thus very important to show our students the right way. Third, misconduct in science—even of only a few scientists—leads to mistrust from people, institutions, and governments of the whole scientific world. Discussing such issues is thus important and can be presented in *SPM* columns and forums.

Signal processing and image processing are relatively new domains in the sciences, even if some methods were proposed a long time ago, sometimes a few centuries, by visionary scientists. It has also been developed according to advances in numerical technologies, especially electronics, sensors, communications, and computer sciences. I believe that it is very instructive to recall the role of pioneers, to explain the context in which some discoveries have been done, and, more generally, to celebrate the outstanding scientists who contributed to our domain. Again, I believe that articles on such pioneers—like Fourier, Cooley and Tukey, or Widrow, to cite only a few—have their place in *SPM*.

SPM is very different from the transactions, which are focused on the latest advances in very accurate areas of research. Conversely, *SPM* presents review articles and special issues on mature domains and other topics, including columns and forums that are of interest to many readers. *SPM* articles are written for a large audience, including IEEE Signal Processing Society (SPS) members, but also for students and industrial practitioners. Outgoing EIC Prof. Robert Heath and his team of area editors, the editorial board, and associate editors were working hard to produce high-quality content in various areas, and the next issues will also be the result of their tremendous work. In his editorial in the November 2020 issue, Prof. Heath [2] explained how the roles of area editors, senior editorial board members, and associate editors are important. During his last few months as EIC, he, as well as *SPM*'s Managing Editor Jessica Welsh and SPS Publications Administrator Rebecca Wollman, also spent time explaining to me how *SPM* works, and I thank them for their kind help. I am confident that I can also count on them and their advice in the future. Thus, it is challenging for me to propose, with a new team of area editors that I am currently constituting, content of the same high quality and diversity, especially in special issues and features. As I explained, I also would like to stimulate other kinds of articles, e.g., sharing teaching experiences, recalling historical milestones achieved by some pioneers in our domain, and discussing the human values concerning scientific integrity and ethics that are defended by the SPS. Quoting Prof. Heath's [3] first editorial in January 2018: "*SPM* is a magazine for all of us. I look forward to your feedback and ideas." I also look forward to your contributions.

References

[1] T. Adali, J. Trussell, L. K. Hansen, and V. D. Calhoun, "The dangers of following trends in research: Sparsity and other examples of hammers in search of nails," *Proc. IEEE*, vol. 106, no. 6, pp. 1014–1018, June 2018. doi: 10.1109/JPROC.2018.2823428.

[2] R. W. Heath, "Signing off as editor-in-chief," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 3–4, Nov. 2020. doi: 10.1109/MSP.2020.3019614.

[3] R. W. Heath, "Taking the next step for *IEEE Signal Processing Magazine*," *IEEE Signal Process. Mag.*, vol. 35, no. 1, p. 4, Jan. 2018. doi: 10.1109/MSP.2017.2770481.

SP

Ahmed Tewfik | IEEE Signal Processing Society President | tewfik@austin.utexas.edu

Il N'y a ni Mauvaises Herbes ni Mauvais Hommes

A little over a century and a half ago, Victor Hugo wrote "Il n'y a ni mauvaises herbes ni mauvais hommes. Il n'y a que de mauvais cultivateurs," which translates to "there are no weeds and no bad men. There are only bad cultivators." These two sentences provide a stark reminder of the heavy responsibility we all bear, as parents, educators, mentors, members of professional societies, and citizens of states, nations, and earth. Indeed, arguably our main goal as a professional society is to help develop our human capital. Everything else flows from there. As we empower our members and help them innovate and flourish, we advance the state of the art and make our world a better place for all of its inhabitants. This also underlines the importance of careful strategic planning and execution.

By the time you read this "President's Message," the IEEE Signal Processing Society (SPS) will have completed the first phase of its strategic planning. That first phase was led by several ad hoc committees made up of volunteer members of the Society and members of its Executive Committee and Board of Governors. In the second phase, the SPS will engage with all of you in a series of interactive webinars, centered on finances, conferences, innovations, publications, membership, and education, to finalize its plans by the end of the first quarter of 2021.

The operations of the SPS—including, publications, conferences, technical initiatives, committee support, and staff and member services—are supported essentially by income it derives from IEEE *Xplore* subscriptions and conferences. Because of the peculiarities of IEEE accounting and finance rules, the Society has consistently run a surplus, which at times has been substantial due to favorable IEEE investment returns. However, the changing publications and conference models are raising concerns about the sustainability of our current revenue streams. As such, it behooves us to carefully consider the financial implications of these changes and begin to invest in new revenue sources. These new revenue sources will also force the Society to make sure that it offers services that industry and society at large value and are willing to pay for through new channels, such as grants and funds designated to support specific or general Society activities and initiatives.

As I noted in one of my columns last year, publications are clearly moving to an open access model. While this model is beneficial to humanity at large and increases the visibility and impact of our authors, it raises questions about the ability of authors to pay to have their articles published and the larger economics of publishing articles. Our goal is to understand how we can offer economical open-access publications while subsidizing authors' publications fees with industry and government grants or income derived by new revenue sources. In addition, the ArXiv preprint model is upending the current peer-review publication model. Reimagining peer review to embrace preprints and accelerate the review and publication processes is a challenge that our strategic planning exercise also aims to address. Cutting-edge innovations are happening at the intersection of disciplines, bolstering our commitment to high-quality technical publications and conferences co-sponsored with sister IEEE Societies. Finally, we clearly learned this year that we are not serving the needs of many of our members through our conferences and publications. There is no question that we must develop new publications that specifically address the needs of our practitioners and uphold our commitment to quality and excellence.

Our flagship conferences, ICASSP and ICIP, this year established that all of our future meetings will have a virtual component. Effective virtual components that deliver real value to virtual presenters and attendees and enrich the experience of physical attendees are challenging us on multiple fronts: technology; conference formats that support multiple time zones, networking, and productive random productive exchanges; and conference registration fees that democratize participation in our conferences. Keeping and expanding the large number of ICASSP and

Digital Object Identifier 10.1109/MSP.2020.3031648
Date of current version: 24 December 2020

ICIP first-time attendees and offering new conference experiences that attract the vast majority of Society members who do not attend our meetings should be central to our thinking and execution. Despite the continued COVID-19 uncertainty in 2021, we will need to move forward with our plans to create a new conference format that makes us the preferred meeting spot for an innovation ecosystem that brings together start-ups, venture capital, government, industry, and academia.

Our education and membership initiative planning are focusing on bulking up our diversity, inclusion, and outreach initiatives that are meant to address systemic inequities within our workplaces, professional societies, and throughout the globe. Another goal is identifying the education niche that the Society should serve, complementing the offerings from the academic institutions to which many our members belong. The challenges in these two areas are perhaps more formidable than in other areas, as few of us have developed a clear understanding of how we can implement our vision and how we can measure progress.

I have no doubt that, together, we will come up with compelling initiatives. In deciding what to pursue, we should focus on a careful examination of the potential reach, impact, confidence in our ability to deliver and resource the requirements of each initiative. Please allow me to reiterate what I wrote last year: this is *your* Society; let us all participate in its governance.

Let me close by expressing my sincere gratitude to two of our vice presidents who stepped down on 31 December 2020: Fernando Pereira and Sergios Theodoridis. Fernando devoted considerable efforts to reimagining our conference model and converting our conferences to be fully virtual in 2020. No words can express the magnitude of the debt we owe Fernando as a society. Sergios led our efforts regarding open access with the launch of our first open access journal. He spearheaded our partnerships with other Societies in launching new interdisciplinary journals on artificial intelligence and machine learning on communications. Both helped us navigate the real challenges we faced in 2020. We are glad that they both have agreed to continue to serve on our ad hoc strategic planning committees. On 1 January 2021, Fernando was succeeded by Ana Pérez-Neira and Sergios by Marc Moonen. The Society remains in good hands: Ana worked closely with Fernando on moving ICASSP 2020 to a fully virtual format and Marc led the efforts of EURASIP in open access. Tremendous thanks are given to you for your service, Sergios and Fernando, and welcome Ana and Marc.

SP

CALL FOR PAPERS

IEEE Journal of Selected Topics in Signal Processing

Special Issue on Joint Communication and Radar Sensing for Emerging Applications

For the sake of enhancing the exploitation of the permanently allocated, but potentially under-utilized spectral resources, sharing the frequency bands between radar and communication systems has attracted substantial attention, which motivates the research of joint communication and radar (JCR) designs. In general, there are two main research directions in JCR: 1) Radar-communication coexistence (RCC) and 2) Dual-functional Radar-Communication (DFRC) system design. The first category of research aims at developing efficient interference management techniques, so that the two systems can operate without unduly interfering with each other. On the other hand, DFRC techniques focus on designing joint systems that can simultaneously perform wireless communication and remote sensing, which benefits both sensing and signaling operations via real-time cooperation, de-congests the RF environment, and allows a single hardware platform for both functionalities. This type of work has been extended to numerous novel applications, including vehicular networks, indoor positioning and covert communications. This special issue seeks to bring together contributions from researchers and practitioners in the area of wireless communications and radar with an emphasis on new approaches and techniques for joint communication and sensing designs. We solicit high-quality original research papers on topics including, but not limited to:

- Spectrum analysis and management for communication and radar systems
- Opportunistic spectrum sharing for communication and radar systems
- Interference channel modeling and estimation for communication and radar spectrum sharing
- Interference mitigation techniques for communication and radar spectrum sharing
- Fundamental theory of dual-functional radar-communication systems
- Joint precoding/beamforming design for dual-functional radar-communication systems
- Joint receiver design for dual-functional radar-communication systems
- Security and privacy issues in communication and radar spectrum sharing
- MIMO and massive MIMO dual-functional radar-communication systems
- Millimeter wave dual-functional radar-communication systems
- Joint sensing and communication design for unmanned aerial vehicle (UAV) network
- Joint sensing and communication design in 5G vehicular-to-everything (V2X) network
- Wi-Fi based indoor positioning and target detection/recognition
- Radar-assisted low-probability-of-intercept (LPI) communications
- Machine learning based approaches for radar and communication signal classification
- Experimental demonstrations and prototypes

In addition to technical research results, we invite very high quality submissions of a tutorial or overview nature; we also welcome creative papers outside of the areas listed here but related to the overall scope of the special issue. Prospective authors can contact the Guest Editors to ascertain interest on topics that are not listed.

Prospective authors should visit http://www.signalprocessingsociety.org/publications/periodicals/jstsp/ for information on paper submission. Manuscripts should be submitted using the Manuscript Central system at http://mc.manuscriptcentral.com/jstsp-ieee. Manuscripts will be peer-reviewed according to the standard IEEE process.

Manuscript Submission:	January 31, 2021
First review completed:	April 30, 2021
Revised manuscript due:	June 15, 2021
Second review completed:	July 31, 2021
Final manuscript due:	September 15, 2021
Publication date:	November 2021

Guest Editors

Prof. Christos Masouros (Lead Guest Editor), University College London, UK, email: c.masouros@ucl.ac.uk
Prof. Robert W. Heath, Jr., North Carolina State University, USA, email rwheathjr@ncsu.edu
A/Prof. J. Andrew Zhang, University Technology Sydney, Australia, email: andrew.zhang@uts.edu.au
Prof. Zhiyong Feng, Beijing University of Posts and Telecommunications, China, email: fengzy@bupt.edu.cn
Dr. Le Zheng, Aptiv, USA, email: le.2.zheng@aptiv.com
Prof. Athina Petropulu, Rutgers University, USA (athinap@soe.rutgers.edu)

Digital Object Identifier 10.1109/MSP.2020.3041908

Top Downloads on IEEE *Xplore*

Each "Reader's Choice" column focuses on a different publication of the IEEE Signal Processing Society. In this issue, we highlight articles in *IEEE Signal Processing Magazine* (*SPM*).

SPM publishes tutorial-style articles on signal processing research and applications as well as columns and forums on issues of interest. Its coverage ranges from fundamental principles to practical implementation, reflecting the multidimensional facets of interests and concerns of the community. Its mission is to bring up-to-date, emerging, and active technical developments, issues, and events to the research, educational, and professional communities. It is also the main Society communication platform addressing important issues concerning all members.

This issue's "Reader's Choice" column lists the top 15 *SPM* articles most downloaded from January 2018 to September 2020. Your suggestions and comments are welcome and should be sent to Associate Editor H. Vicky Zhao (vzhao@tsinghua.edu.cn).

Deep Reinforcement Learning: A Brief Survey

Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A.

This article reviews the recent progress in deep reinforcement learning (RL). The survey begins with an introduction to RL's general field and then progresses to the main streams of value- and policy-based methods. Important algorithms in deep RL are explored, including the deep Q-network, trust region policy optimization, and asynchronous advantage actor critic. The unique advantages of deep neural networks (DNNs) are also surveyed, focusing on visual understanding via RL.

2017

An Introduction to Compressive Sampling

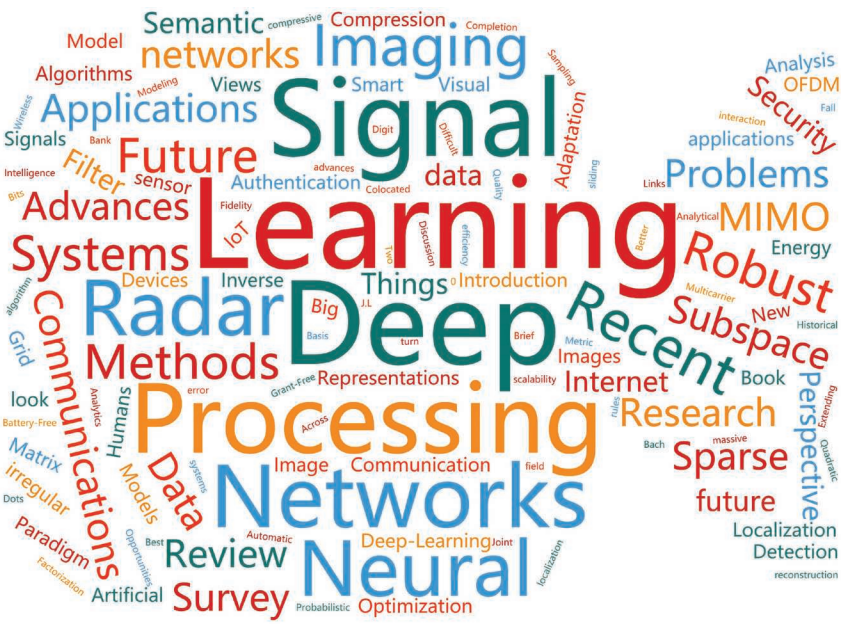
Candes, E.J.; Wakin, M.B.

This article surveys the compressive sampling theory, also known as *compressed sensing* (*CS*), a novel sensing/sampling paradigm that goes against the common wisdom in data acquisition. The CS theory asserts that one can recover certain signals and images from far fewer samples or measurements than traditional methods use. The article highlights that randomness can lead to very effective sensing mechanisms, explains why compressive sensing is a concrete protocol for sensing and compressing data simultaneously, and evaluates important applications.

2008

Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups

Hinton, G.; Deng, L; Yu, D.; Dahl, G.E.; Mohamed, A..; Jaitly, N.; Senior, A.;



Digital Object Identifier 10.1109/MSP.2020.3032087
Date of current version: 24 December 2020

Vanhoucke V.; Nguyen, P.; Sainath, T.N.; Kingsbury B.

This article explores recent progress in using DNNs for acoustic modeling in speech recognition. It starts with the two-stage training procedure used for fitting the DNNs. In the first stage, layers of feature detectors are initialized by fitting a stack of generative models. In the second stage, each generative model is used to initialize one layer of hidden units in a DNN, and the whole network is fine-tuned to predict the target hidden Markov model states. Exploratory experiments on the TIMIT database demonstrate the power of the DNNs for acoustic modeling.

2012

Geometric Deep Learning: Going Beyond Euclidean Data

Bronstein, M.M.; Bruna, J.; LeCun, Y.; Szlam, A.; Vandergheynst, P.

Geometric deep learning refers to the emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The non-Euclidean nature of such data implies that there are no such familiar properties as global parameterization, a common system of coordinates, vector space structure, or shift invariance. Consequently, basic operations like convolution are not well defined in non-Euclidean domains. This article studies different methods to translate the key ingredients of successful deep learning methods to non-Euclidean data and presents available solutions, difficulties, applications, and future research directions in this nascent field.

2017

Automotive Radars: A Review of Signal Processing Techniques

Patole, S.M.; Torlak, M.; Wang, D.; Ali, M.

Various signal processing techniques have been developed to provide better resolution and estimation performance for automotive radar systems. This article summarizes various aspects of automotive-radar signal processing techniques, including waveform design, possible radar architectures, estimation algorithms, complexity-resolution tradeoff, adaptive processing for complex environments, and unique problems associated with automotive radars, such as pedestrian detection.

2017

HTTPS://PIXABAY.COM/ZH/ILLUSTRATIONS/ARTIFICIAL-NEURAL-NETWORK-ANN-3501528

Scaling Up MIMO: Opportunities and Challenges With Very Large Arrays

Rusek, F.; Persson, D.; Lau, B.K.; Larsson, E.G.; Marzetta, T.L.; Edfors, O.; Tufvesson, F.

The application of very large multiple-input, multiple-output (MIMO) arrays is a new research field in communication theory, propagation, and electronics, motivating the entirely new theoretical research of signal processing, coding, and network design. This article surveys some of the challenges with very large arrays. It discusses ultimate information–theoretic performance limits, practical algorithms, the influence of channel properties on the system, and practical constraints on the antenna arrangements.

2013

The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains

Shuman, D.I.; Narang, S.K.; Frossard, P.; Ortega, A.; Vandergheynst, P.

The emerging field of signal processing on graphs merges algebraic and spectral graph theoretic concepts with computational harmonic analysis. This tutorial outlines the main challenges of the area, discusses different ways to define graph spectral domains, and highlights the importance of incorporating the irregular structures of graph data domains when processing signals on graphs. It then discusses methods to generalize fundamental operations, such as filtering, translation, and modulation to the graph setting. It surveys the localized, multiscale transforms to extract information from high-dimensional data on graphs. Open issues and possible extensions are also discussed.

2013

Compressive Sensing [Lecture Notes]

Baraniuk, R.G.

Addressing the Nyquist–Shannon sampling theory's limitations in real applications, this lecture note presents a new method to capture and represent compressible signals at a rate significantly below the Nyquist one. This method, called *compressive sensing*, employs nonadaptive linear projections that preserve the structure of the signal, which is then reconstructed from these projections using an optimization process. The ideas presented illustrate the links among data acquisition, compression, dimensionality reduction, and optimization in undergraduate and graduate digital signal processing, statistics, and applied mathematics courses.

2007

Deep Multimodal Learning: A Survey on Recent Advances and Trends

Ramachandram, D.; Taylor, G.W.

The success of deep learning has been a catalyst for solving increasingly complex machine learning (ML) problems, which often involve multiple data modalities. This article reviews recent advances in deep multimodal learning, highlights the state-of-the-art works, and presents the gaps and challenges in this active research field. It first classifies deep multimodal learning architectures and then discusses methods to fuse learned multimodal representations in deep learning architectures. Two research topics are highlighted as exciting areas for future work: regularization strategies and methods that learn or optimize multimodal fusion structures.

2017

IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security?

Xiao, L.; Wan, X.; Lu, X.; Zhang, Y.; Wu, D.

The Internet of Things (IoT) integrates a variety of devices into networks to provide advanced and intelligent services. It has to protect user privacy and address attacks, such as spoofing and denial-of-service attacks, jamming, and eavesdropping. The attack model for IoT systems is investigated, and the ML-based IoT security solutions are accessed, including ML-based IoT authentication, access control, secure offloading, and malware-detection schemes to protect data privacy. The article then discusses the challenges that need to be addressed to implement these ML-based security schemes in practical IoT systems.

2018

MIMO Radar for Advanced Driver-Assistance Systems and Autonomous Driving: Advantages and Challenges

Sun, S.; Petropulu, A.P.; Poor, H.V.

MIMO radar technology has been used in the current-generation automotive radar for advanced driver-assistance systems and in the next-generation, high-resolution imaging radar for autonomous driving. This article examines MIMO radar basics, highlighting the features that make this technology a good fit for automotive radar, and surveys important theoretical results for increasing the angular resolution. It also describes the challenges that arise when applying existing MIMO radar theory to automotive radar, providing interesting problems for signal processing researchers.

2020

Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation [Lecture Notes]

Faragher, R.

Named after Rudolf E. Kálmán, the Kalman filter's great success is due to its small computational requirement, elegant recursive properties, and its status as the optimal estimator for 1D linear systems with Gaussian error statistics. Typical uses of the Kalman filter include smoothing noisy data and providing estimates of parameters of interest. This article provides a simple and intuitive derivation of the Kalman filter, aiming to teach this useful tool to students from disciplines that do not require a strong mathematical background.

2012

Semidefinite Relaxation of Quadratic Optimization Problems

Luo, Z.; Ma, W.; So, A.M.; Ye, Y.; Zhang, S.

Semidefinite relaxation (SDR) is a powerful and computationally efficient approximation technique for a host of challenging optimization problems. In particular, it can be applied to many nonconvex, quadratically constrained quadratic programs. This article provides general, comprehensive coverage of the SDR technique, from its practical deployments and scope of applicability to key theoretical results. It also showcases several representative applications, namely MIMO detection, B1 shimming in magnetic resonance imaging, and sensor network localization.

2010

Generative Adversarial Networks: An Overview

Creswell, A.; White, T.; Dumoulin, V.; Arulkumaran, K.; Sengupta, B.; Bharath, A.A.

Generative adversarial networks (GANs) provide a way to learn deep representations without extensively annotated training data. They achieve this by deriving backpropagation signals through a competitive process involving a pair of networks. The representations that can be learned by GANs may be used in a variety of applications, including image synthesis, semantic image editing, style transfer, image superresolution, and classification. This article provides an overview of GANs for the signal processing community, drawing on familiar analogies and concepts where possible. In addition to identifying different methods to train and construct GANs, it also points to remaining challenges in their theory and application.

2018

Convolutional Neural Networks for Inverse Problems in Imaging: A Review

McCann, M.T.; Jin, K.H.; Unser, M.

This article discusses recent uses of convolutional NNs (CNNs) to solve inverse problems in imaging, such as denoising, deconvolution, superresolution, and medical image reconstruction. It evaluates the recent experimental works in these areas, focusing on the critical design decisions: From where do the training data come? What is the architecture of the CNN? How is the learning problem formulated and solved? It also mentions a few key theoretical articles that offer perspectives on why CNNs are appropriate for inverse problems and points to some next steps in the field.

2017

SP

John Edwards

Signal Processing Advances the Quest for Better and Safer Medical Imaging

Imaging breakthroughs are saving lives by giving radiologists and physicians sharper and safer views inside the human body

Medical imaging has progressed in impressive ways since the discovery of X-rays more than a century ago. Modern radiologists can now visualize body parts and organs to help physicians diagnose, treat, and monitor disease and/or injury in intricate detail, using techniques such as computed tomography (CT), positron emission tomography, and various other modalities.

The downside to many current imaging approaches is that they expose patients to ionizing radiation, which potentially increases their risk of eventually developing cancer. Signal processing is now helping researchers develop new medical imaging technologies that are not only noninvasive but also virtually risk-free.

A smart brain biopsy needle

Researchers and clinicians at the University of Adelaide and the University of Western Australia have developed a "smart brain biopsy needle" that's designed to reduce the risk of dangerous brain bleeds in patients undergoing brain biopsies.

A needle biopsy is currently a common way to identify the exact type of brain tumor under investigation. The process involves inserting a large needle into the patient's brain and snipping out a tissue sample. A neurosurgeon will typically take anywhere from six to 10 samples.

"When they take the tissue sample, there's a risk they will also cut any blood vessels next to the needle," says project leader Prof. Robert McLaughlin, chair of Biophotonics at the University of Adelaide's Medical School. The surgeon has to be very careful to avoid an intracranial bleed. "Approximately 1% of patients having a brain biopsy will die from intracranial bleeds, and 2–3% will be left permanently disabled," he observes.

Signal processing is now helping researchers develop new medical imaging technologies that are not only noninvasive but also virtually risk-free.

Developed jointly by the University of Adelaide and the University of Western Australia, the new smart needle (Figure 1) contains an optical fiber probe that's capable of detecting when the needle is next to a blood vessel and automatically alerting the neurosurgeon to the fact. The approach takes advantage of optical coherence tomography (OCT), a technology already commonly used in ophthalmology and cardiology. "We're adapting it for use in neurosurgery," McLaughlin says.

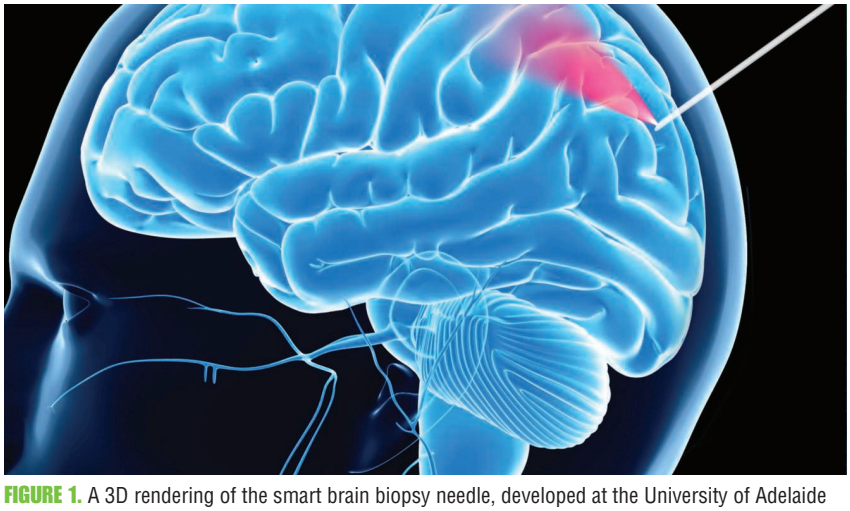


FIGURE 1. A 3D rendering of the smart brain biopsy needle, developed at the University of Adelaide and the University of Western Australia, in operation. (Source: Miniprobes Pty. Ltd.; used with permission.)

Digital Object Identifier 10.1109/MSP.2020.3032349
Date of current version: 24 December 2020

The researchers created a miniaturized OCT imaging probe and placed it inside a brain biopsy needle. "We did this by fabricating a tiny lens on the end of an optical fiber the thickness of a human hair (0.2 mm)," McLaughlin explains. "We then shine light through the optical fiber and out of a tiny hole near the tip of the needle." The process allows the probe to take an OCT scan of the tissue that's immediately next to the tip to determine if a blood vessel is present.

The new smart needle contains an optical fiber probe that's capable of detecting when the needle is next to a blood vessel and automatically alerting the neurosurgeon to the fact.

OCT is analogous to ultrasound (US). "In US, the scanner sends sound waves into the tissue, and by analyzing the echoes that are detected, it's able to build up an image of the inside of the body," McLaughlin says. OCT performs the same task but substitutes near-infrared light waves for sound waves. "Light waves are much smaller than sound waves, so we can see much smaller things," he notes. The probe's imaging resolution is approximately 5–20 μm, creating a single pixel. "However, light doesn't penetrate very far into the body, so we can only see 1–2 mm below the surface," McLaughlin notes. "This means that OCT is capable of seeing very small things but needs to be very close to them."

US and OCT both excel at blood vessel detection and share a unique characteristic. "If you look closely at an US or OCT image, you will see that it's completely covered in a type of salt and pepper noise—speckle noise," McLaughlin says. While speckle noise is little more than obstructive clutter to an US probe user, it's a treasure trove of vital information to an OCT probe operator. The researchers took advantage of this capability by developing signal processing algorithms that analyze and differentiate the speckle patterns generated by moving blood as opposed to the speckling generated by stationary tissue.

Speckle noise is simply a random collection of bright and dark spots that don't change over time. "If I take an OCT scan of an area of tissue, and if I don't move and the tissue doesn't move, then all the bright and dark spots stay in the same location," McLaughlin explains. The trick to detecting blood vessels lies in using OCT to image the tissue as the needle moves incrementally. "As we pass through stationary tissue, the speckle will change very slowly," he says. "If we suddenly reach a point where the speckle is changing quickly, we know that something else must be moving." In a clinical situation, it's a strong indication that blood is flowing through a vessel.

The signal processing algorithm is designed to measure changes in the speckle noise intensity. "We can't predict the change in an individual pixel, but we can calculate the statistics of the intensity change over a small region of pixels," McLaughlin notes. "We can then relate the rate of intensity change to the speed at which things are moving." If the speckle intensity is changing very slowly, it's a sign that the needle is next to stationary tissue. If the speckle is changing somewhat faster, it's likely near a slow-flowing blood vessel. "If the speckle is changing very quickly, then we must be imaging a fast-flowing blood vessel," he explains.

The imaging needle project recently completed a successful initial validation with 11 patients at Sir Charles Gairdner Hospital in Western Australia. Prof. Christopher Lind, a consultant neurosurgeon at Sir Charles Gairdner Hospital and the University of Western Australia, led the clinical trial.

McLaughlin says that working on the project was personally satisfying. "What makes this work so fun is the way it combines signal processing, physics, hardware design, and a really compelling medical need," he notes. "As an electronic engineer who started his research career in signal processing, I love the way that this project allows me to work in so many different areas of engineering at the same time."

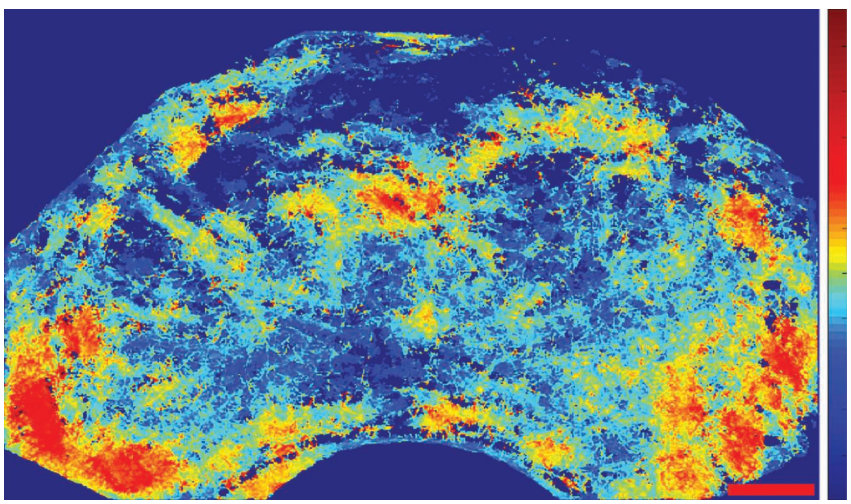


FIGURE 2. A prostate scanned by the superresolution US technology developed by Heriot-Watt University researchers. The technology promises a 5–10× resolution improvement over standard US images. (Source: Heriot-Watt University; used with permission.)

Superresolution US

Heriot-Watt University scientists have developed a superresolution US technology that promises a 5–10× resolution improvement over standard US images. The system allows entire organs to be scanned in superresolution for the first time, potentially leading to earlier cancer diagnoses and allowing medical staff to create better targeted treatment strategies.

With its impressive image resolution leap, superresolution promises to

revolutionize medical diagnosis, says research team leader Vassilis Sboros, a Heriot-Watt associate professor. "It's literally like looking into the human body with a microscope," he observes (Figure 2). "As such, we expect to visualize processes that have not been seen before."

A number of groups are involved in the research, Sboros notes. "Our main contribution is that we demonstrated that it's possible to achieve strong superresolution results with existing clinical equipment and within reasonable patient examination times," he says.

The methodology borrows from algorithms used in astronomy and light microscopy that employ microbubble suspensions. "This research effort started with basic physics—both experimental and theoretical—[and] investigations on the nonlinear scatter properties of US microbubbles," Sboros says. "This knowledge was then injected into signal processing and, eventually, into image formation."

The researchers initially developed a methodology that used several well-known signal processing tools, such as Kalman filters and spectral methods, yet changes are planned. "I feel that there are a number of exciting new methods already implemented in other fields of sensing that we have not been able to use to date," Sboros says.

The researchers haven't yet decided on a final signal processing method to replace the current set of conventional signal processing approaches. "It's a very big challenge … in clinical equipment," Sboros notes. "We do not feel at this stage that there is one signal processing method that is better than all the others."

Given the complexities inherent in US imaging, the vascular system, and microbubble nonlinearity, the team prefers that any new methodology selected is well understood and tested prior to inclusion. "Having said that, we have evidence to suggest that methods like minimum variance and sharpness-based beamforming can adapt to single-particle imaging and provide viable solutions," Sboros says.

One possible option is spectral analysis, which uses a Bayesian approach to provide superresolved spectral information. "It appears that we can identify and tag individual particles by means of spectral identification," Sboros states. "This is exciting, and we haven't yet had the chance to implement it in real imaging." He reports that the researchers have also used minimum variance—via a Capon filter—to create high-resolution particle images. "The challenge is to do this in the clinical setting and for live imaging," he observes.

While speckle noise is little more than obstructive clutter to an US probe user, it's a treasure trove of vital information to an OCT probe operator.

Besides an array of cancers, superresolution US also has the potential to be applied to any noncommunicable disease that can currently be investigated with US imaging. "It could be used to diagnose and target treatment for a number of vascular-related diseases such as diabetes, liver disease, transplant rejection, stroke, and Alzheimer's, to name a few," Sboros predicts.

Photoacoustic imaging

A Ryerson University-led research team, working out of the Institute for Biomedical Engineering, Science and Technology at Toronto's St. Michael's Hospital, is hoping to raise photoacoustic (PA) imaging to an entirely new level.

Like US imaging, PA imaging creates a visual image of biological structures by collecting soundwaves. While US imaging transmits soundwaves into biological structures, PA imaging utilizes light.

PA imaging projects light into biological structures, such as blood vessels, that can absorb the energy. Light waves generate a tiny amount of heat inside the target structure, which triggers an almost imperceptible expansion in volume, generating sound.

Using a technique they call *F-Mode*, the researchers were able to subdivide PA signals into different frequency bands. They then successfully demonstrated the selective enhancement of features in samples ranging from biological cells to live zebrafish larvae without relying on the contrast dyes typically required by other imaging techniques (Figure 3).

One of the best things about the technique is that it can, in principle, be applied to data acquired from any conventional US or PA system, provided that the user has access to unprocessed digitized signal data and doesn't require any special hardware,

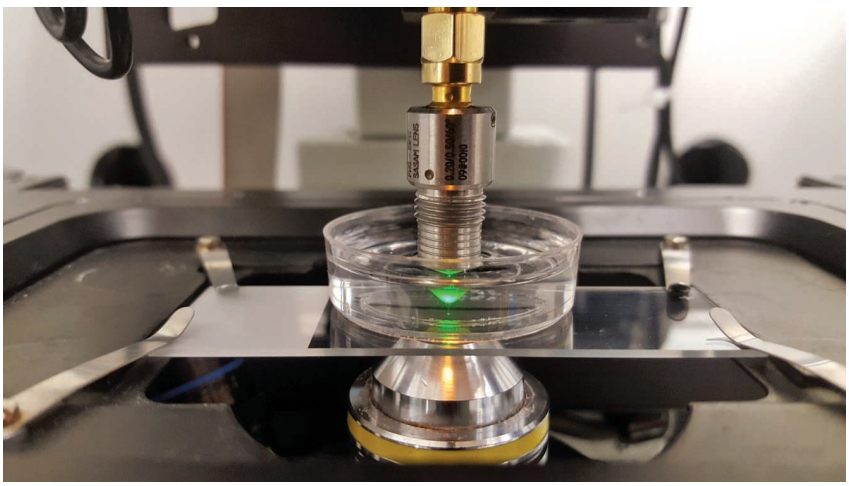


FIGURE 3. A close-up image of the PA microscope used for acquiring F-Mode images. A 532-nm laser is focused through a 10× optical objective (bottom of figure) into a petri dish containing the sample to be imaged. A 200-MHz US transducer, immersed in the dish, records the resultant PA signals. (Source: Ryerson University; used with permission.)

microbubbles," Sboros team member Michael Moore, currently a medical physics resident at Grand River Hospital in Ontario, Canada.

Moore believes that the technique is optimal for zeroing in on objects of specific sizes. For example, when imaging blood vessels, vessel sizes within a specific area can vary by several orders of magnitude, only some of which would be of interest in a given clinical situation. "The idea is that by using F-Mode, the user would be able to hone in on the size of the object that they're interested in, removing some of the objects that would otherwise clutter the image," he explains.

Besides an array of cancers, superresolution US also has the potential to be applied to any noncommunicable disease that can currently be investigated with US imaging.

F-Mode also opens the way to size-specific contrast levels in PA imaging. "Usually in PA, contrast comes from the relative optical absorption properties of the sample that you're analyzing," Moore states. "In F-Mode, you can have two objects with the exact same optical absorption characteristics but make either one of those objects completely disappear just by changing the frequency band used to construct the image."

One of the main reasons F-Mode works so well for PA imaging is that the PA signals produced are inherently broadband, akin to the acoustic signal that's generated by a popping balloon. Most current PA imaging techniques simply measure amplitude, displaying areas emitting louder sounds with brighter pixels. F-Mode, on the other hand, uses the frequency content of sounds emitted from biological structures to determine pixel brightness.

Just as with the human voice, the frequency content of any given signal is unique. In PA, the content is dictated by both the object's size and its shape. Smaller objects produce signals with more high-frequency content than large objects. "Interestingly, within the broadband power spectrum of the signal, there are some 'null' frequencies, or frequency bands that contribute very little to the overall signal energy," Moore says.

Null frequencies vary depending on the size and shape of the object producing the sound. One object may have a maximum frequency in one band, while another might have a minimum. By appropriately subdividing the power spectra of the radio-frequency (RF) data into bands, we can choose to reconstruct an image from only a single frequency band," Moore says. "If we choose one of the special bands where the contribution of one object is high while the contribution of the other object is at a minimum, one object will appear much brighter than the other in the resultant image."

F-Mode is heavily reliant on signal processing. "Without it, we would never be able to access the size-specific data encoded in our time-domain signals," Moore says. "We first acquire PA signals from objects of different size using either a PA microscope or PA CT system," he says. This typically involves standard techniques, such as signal averaging, and the application of a Tukey or Hamming filter to remove some of the high-frequency noise from outside the transducer bandwidth, Moore notes.

In conventional image formation, the maximum amplitude of the PA signal at each spatial location within the scan region is used to determine the intensity of the corresponding pixel in the resulting image—a technique known as *maximum amplitude projection imaging*. "In F-Mode, we take those PA signals and apply the fast Fourier transform to them and compute their power spectrum," Moore says. "By appropriately subdividing the power spectra of the RF data into bands, spectral features, such as nulls within the divisions, can be isolated and used as a mechanism for generating image contrast," he explains. Displaying the contents of one such band at each location in the scan data set results in images that are highly sensitive to small deviations in object size and shape.

The research team, including Ryerson biomedical physics doctoral candidates Eno Hysi and Muhannad Fadhel, was led by PA imaging pioneer Michael Kolios, a professor in Ryerson's Physics Department and an associate dean of science for research and graduate studies. The work was funded by the Natural Sciences and Engineering Research Council of Canada, the Canadian Cancer Society, and the Canadian Foundation for Innovation.

The researchers next hope to investigate additional in vivo PA applications in small animals, such as mice. Their ultimate goal is to use the technique to selectively enhance human blood vessels of various sizes for improved disease detection.

Author

John Edwards (edwards@johnedwardsmedia.com) is a technology writer based in the Phoenix, Arizona, USA, area. Follow him on Twitter @TechJohn Edwards.

SPS Announces the 2021 Class of Distinguished Lecturers and Distinguished Industry Speakers

The IEEE Signal Processing Society (SPS) Distinguished Lecturer (DL) Program provides a means for Chapters to have access to well-known educators and authors in the fields of signal processing to lecture at Chapter meetings. While many IEEE Societies have similar programs, the SPS provides a substantial amount of financial support for the Chapters to take advantage of this service.

The Distinguished Industry Speaker (DIS) Program provides a means for Chapters to have access to individuals who are recognized experts with a background in industrial applications in the signal processing area and are well versed in the ongoing issues/activities in industry to lecture at Chapter meetings. The main difference and goal of the DIS Program is to educate and interact with Society members about topics that are of primary importance to industry and the signal processing community at large. The DIS Program will supplement and closely mirror the regular DL Program.

Five colleagues were honored with the appointment of DL: Pier Luigi Dragotti (Imperial College London), Karen Livescu (Toyota Technological Institute at Chicago), Venkatesh Saligrama (Boston University), Dimitri Van De Ville (École polytechnique fédérale de Lausanne), and Dong Xu (University of Sydney, Australia).

Five colleagues were honored with the appointment of DIS: Achintya K. (Achin) Bhowmik (Starkey Hearing Technologies), Chienchung Chang (Qualcomm Technologies), Mérouane Debbah (Huawei France Research Center), Dilek Hakkani-Tür (Amazon Alexa AI), and Xiaodong He (JD.COM Inc.).

Chapters interested in arranging lectures by a DL or a DIS can visit the following websites:

- DL Program: http://signalprocessingsociety.org/professional-development/distinguished-lecturer-program
- DIS Program: https://signalprocessingsociety.org/professional-development/distinguished-industry-speaker-program.

Lectures can also be arranged by sending an e-mail to sp.info@ieee.org.

Achintya K. Bhowmik

Achintya K. (Achin) Bhowmik is the chief technology officer and executive vice president of engineering at Starkey Hearing Technologies, a privately held medical devices business with more than 5,000 employees and operations in more than 100 countries worldwide. In this role, he is responsible for the company's technology strategy, global research, product development, and engineering departments. He is also responsible for leading the drive to transform hearing aids into multifunction wearable health and communication devices with advanced sensors and artificial intelligence (AI) technologies.

Prior to joining Starkey, Dr. Bhowmik was the vice president and general manager of the Perceptual Computing Group at Intel Corporation. There, he was responsible for the R&D, engineering, operations, and businesses in the areas of 3D sensing and interactive computing, computer vision (CV) and AI, autonomous robots and drones, and immersive virtual and merged reality devices.

Dr. Bhowmik is an adjunct professor at the Stanford University School of Medicine, where he advises research and lectures in the areas of multisensory cognition, perceptual augmentation, and intelligent systems. He has also held adjunct and guest professor positions at the University of California, Berkeley; the Liquid Crystal Institute of the Kent State University; the Kyung Hee University, Seoul; and the Indian Institute of Technology, Gandhinagar. He received his B.Tech. degree (1996) from the Indian Institute of Technology, Kanpur and his Ph.D. degree (2000) from Auburn University. He has authored more than 200 publications, including two books and 38 issued patents.

Dr. Bhowmik serves on the board of trustees for the National Captioning Institute, the board of directors for OpenCV, the executive board for the Society for Information Display (SID) as the president-elect, and the board of advisors for the Fung Institute for Engineering Leadership at University of California, Berkeley. He is on the board of directors and advisors for several technology start-up companies. His awards and honors include Time's Best

Digital Object Identifier 10.1109/MSP.2020.3031650
Date of current version: 24 December 2020

Inventions, the Artificial Intelligence Breakthrough Award, the Red Dot Design Award, the Industrial Distinguished Leader Award from the Asia-Pacific Signal and Information Processing Association, and the Fellow of the SID.

Dr. Bhowmik's lecture topics include transforming hearing aids into multipurpose devices as a gateway to health and information; enhancing and augmenting human perception with sensors and AI; evolving medtech in the era of digitalization and AI; AI: from pixels and phonemes to semantic understanding and interactions; virtual and augmented reality (VR and AR): toward life-like immersive experiences; cognitive neuroscience: how do we sense and understand the world?; and perceptual computing: enabling machines to sense and understand the world.

Chienchung Chang

Chienchung Chang received his B.S. degree from National Tsing Hua University, Hsinchu, Taiwan (1982) and his M.S. and Ph.D. degrees from the University of California, San Diego, La Jolla, (1987 and 1991, respectively), all in electrical and computer engineering. Dr. Chang has worked with Qualcomm since 1991. Currently, Dr. Chang is the vice president of engineering at Qualcomm Technologies, where he serves as the department head of the Multimedia R&D and Standards group, with a major focus in forward-looking research in the fields of speech, video, imaging, CV, and AI and extended reality (XR) (VR/AR) technologies.

Dr. Chang pioneered to introduce video, camera, and display technologies into Qualcomm Snapdragon products. In 2005, he successfully led Qualcomm Code-Division Multiple Access (CDMA) Technologies Segment's first multimedia-centric chipset, MSM6550, from design to commercialization. MPEG-4/H.263 video codecs, the 4MP CMOS image sensor, and GPU were introduced into CDMA/WCDMA handsets for the first time. MSM6550 became the best-selling chipsets in the company's history then. The chipsets not only created a watershed moment for wireless smartphone booming, but also laid a solid foundation to foster multimedia technology innovation within Snapdragon platforms up to today.

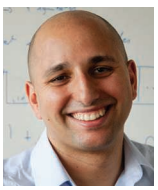
Dr. Chang was commissioned to start the Multimedia R&D and Standards Group focusing on forward-looking multimedia research in speech, video, imaging, CV and XR. Under his leadership, Qualcomm won the ITU-T/MPEG H.265/HEVC, SVC, H.266/VVC, EVC video codec and 3rd Generation Partnership Project EVS speech codec standard competitions. These codec standards are expected to benefit smartphone, automotive, XR, and Internet of Things (IoT) applications for years to come.

In 2017, Dr. Chang helped Qualcomm published 3D depth sensor technology, based on structured lights (3DSL), and its own programmable hardware, RICA. Together with 3D face authentication and relevant CV technologies, he proactively helped Qualcomm deliver the most competitive biometric solutions for Android handsets to counter iPhone FaceID. Lately, Dr. Chang has led XR research, augmented by CV and machine learning research, in rolling out leading-edge perception technologies, such as six degrees of freedom, 3D reconstruction, hand/object detection and tracking, and digital human and split XR. These technologies enhance Qualcomm technology leadership and has helped it become the world's largest VR/AR chipset vendor today.

Dr. Chang's research interests include speech compression, speech recognition, imaging and video processing, CV, pattern recognition, and machine learning. He was recognized as the Distinguished Alumni of the College of Electrical Engineering and Computer Science, National Tsing Hua University, Taiwan, in April 2018.

Dr. Chang's lecture topics include unleashing multimedia technologies on smartphones, automotive, XR, and IoT; boundless XR; and disruptive 3D sensing.

Mérouane Debbah



Mérouane Debbah received his M.Sc. and Ph.D. degrees from the Ecole Normale Supérieure Paris-Saclay, France. He was with Motorola Labs, Saclay, France, from 1999 to 2002, and also with the Vienna Research Center for Telecommunications, Vienna, Austria, until 2003. From 2003 to 2007, he was an assistant professor with the Mobile Communications Department, Institut Eurecom, Sophia Antipolis, France. In 2007, he was appointed full professor at CentraleSupelec, Gif-sur-Yvette, France. From 2007 to 2014, he was the director of the Alcatel-Lucent Chair on Flexible Radio. Since 2014, he has been vice president of the Huawei France Research Center. He is jointly the director of the Mathematical and Algorithmic Sciences Lab as well as the director of the Lagrange Mathematical and Computing Research Center.

Dr. Debbah is an IEEE Fellow, a WWRF fellow, and a membre émérite of SEE. He was a recipient of the European Research Council (ERC) Grant, MORE (Advanced Mathematical Tools for Complex Network Engineering) (2012–2017); Mario Boella Award (2005); IEEE Glavieux Prize Award (2011); Qualcomm Innovation Prize Award (2012); and IEEE Radio Communications Committee Technical Recognition Award (2019). He received more than 20 best paper awards, among which include the 2007 IEEE GLOBECOM Best Paper Award, Wi-Opt 2009 Best Paper Award, 2010 Newcom++ Best Paper Award, WUN CogCom Best Paper 2012 and 2013 Award, 2014 IEEE WCNC Best Paper Award, 2015 ICC Best Paper Award, 2015 IEEE Communications Society Leonard G. Abraham Prize, 2015 IEEE Communications Society Fred W. Ellersick Prize, 2016 IEEE Communications Society Best Tutorial Paper Award, 2016 European Wireless Best Paper Award, 2017 EURASIP Best Paper Award, 2018 IEEE Marconi Prize Paper Award, 2019 IEEE Communications Society Young Author Best Paper Award, and the Valuetools 2007, Valuetools 2008, CrownCom 2009, Valuetools 2012, SAM 2014, and 2017 IEEE Sweden VT-COM-IT Joint Chapter Best Student Paper Awards. He was an associate editor-in-chief of *Journal Random Matrix: Theory and Applications* and an associate area editor and senior area editor of *IEEE Transactions on Signal Processing*, from 2011 to 2013 and from 2013 to 2014, respectively.

Dr. Debbah's research interests lie in fundamental mathematics, algorithms, statistics, information, and communication sciences research.

Dr. Debbah's lecture topics include wireless AI: from cloud AI to on-device AI; rebuilding the theoretical foundations of communication and computing; the fundamentals of 5G; random matrix theory/theory and applications; and an outlook on beyond 5G.

Pier Luigi Dragotti

Pier Luigi Dragotti is a professor of signal processing in the Electrical and electronic engineering Department at the Imperial College London, U.K. He received the laurea degree (summa cum laude) in electronic engineering from the University Federico II, Naples, Italy (1997); a master's degree in communications systems from the Swiss Federal Institute of Technology of Lausanne, EPFL, Switzerland (1998); and a Ph.D. degree from EPFL, Switzerland (April 2002). Before joining the Imperial College in November 2002, he was a senior researcher at EPFL working on distributed signal processing for the Swiss National Competence Center in Research on Mobile Information and Communication Systems.

Prof. Dragotti has also held several visiting positions. He was a visiting student at Stanford University (1996); summer researcher at the Mathematics of Communications Department at Bell Labs, Lucent Technologies, Murray Hill, New Jersey (2000); and visiting scientist at Massachusetts Institute of Technology (MIT) (2011).

Prof. Dragotti is an IEEE Fellow (2017). He was the editor-in-chief of *IEEE Transactions on Signal Processing* (2018–2020); member of the IEEE SPS Fellow Evaluation Committee (2020); associate editor of *IEEE Transactions on Image Processing* (2006–2009); elected member of the IEEE Image, Video, and Multidimensional Signal Processing Technical Committee (2008–2013), where he acted as the chair of the award subcommittee (2011–2013); member of the IEEE Signal Processing Theory and Methods Technical Committee (2013–2018); member of the Computational Imaging Technical Committee (2015–2020); and technical cochair of the European Signal Processing Conference (2012). Prof. Dragotti is also the recipient of an ERC Investigator Award, which is awarded to "exceptional research leaders to pursue ground-breaking, high-risk projects" (2011–2016).

Prof. Dragotti's lecture topics include new sampling methods: sparse sampling based on timing information and sampling along trajectories; deep dictionary learning approaches for image superresolution; and computational imaging for art investigation and for neuroscience.

Dilek Hakkani-Tür

Dilek Hakkani-Tür is a senior principal scientist at Amazon Alexa AI and a visiting distinguished professor at University of California, Santa Cruz, USA, focusing on enabling natural dialogues with machines. Prior to joining Amazon, she was leading the dialogue research group at Google (2016–2018) and a principal researcher at Microsoft Research (2010–2016), the International Computer Science Institute (2006–2010), and the AT&T Labs-Research (2001–2005). She received her B.Sc. degree from the Middle East Technical University in 1994 and her M.Sc. and Ph.D. degrees from Bilkent University, Department of Computer Engineering in 1996 and 2000, respectively.

Dr. Hakkani-Tür is the recipient of three best paper awards for her work on active learning for dialog systems, from the ISPS (2008), the International Symposium on Computer Architecture (ISCA) (2007), and EURASIP (2007). She served as the associate editor of *IEEE Transactions on Audio, Speech, and Language Processing* (2005–2008); as a member of the IEEE Speech and Language Processing Technical Committee (2009–2014); as an area editor for speech and language processing for *Elsevier's Digital Signal Processing Journal* and *IEEE Signal Processing Letters* (2011–2013); and served on the ISCA Advisory Council (2015–2018). She is the editor-in-chief of *IEEE/ACM Transactions on Audio, Speech and Language Processing* (2019–2021) and a Fellow of IEEE (2014) and ISCA (2014).

Dr. Hakkani-Tür's research interests include conversational AI, natural language and speech processing, spoken dialogue systems, and machine learning for language processing.

Dr. Hakkani-Tür's lecture topics include conversational machines: toward bridging the chasm between task-oriented and social conversations; deep learning for task-oriented dialogue systems; and neural network-based response generation in social conversational systems.

Xiaodong He

Xiaodong He is the vice president of technology of JD.COM Inc., deputy managing director of JD AI Research, and the head of the Deep Learning, NLP, and Speech Lab. He is also an affiliate professor at the Electrical and Computer Engineering Department of the University of Washington (Seattle). Dr. He joined JD.COM, the largest online retailer in China, in 2018. Prior to that, he was the principal researcher and research manager of the Deep Learning Technology Center at Microsoft Research, Redmond, Washington. He holds a bachelor's degree from Tsing Hua University, Beijing, an M.S. degree from the Chinese Academy of Sciences, Beijing, and a Ph.D. degree from the University of Missouri, Columbia.

Dr. He is an IEEE Fellow "for contributions to multimodal signal processing in human language and vision technologies," and a fellow of the China Association of Artificial Intelligence. He has held editorial positions in *Transactions of the Association for Computational Linguistics* and multiple IEEE journals, including *IEEE Signal Processing Magazine* and *IEEE Signal Processing Letters* (2017–2018), and he has served on the organizing committees/program committees of major AI conferences. He was the chair on the IEEE Seattle Section (2016–2017) and served on the IEEE Speech and Language Processing Technical Committee (2015–2017).

Dr. He's work—including deep structured semantic models, hierarchical attention networks, bottom-up and top-down attention models, stacked attention networks, MS-Celeb-1M, AttnGAN, CaptionBot, DistMult, STAGG-QA, and RNN-SLU—is widely applied to important scenarios in natural language processing, CV, dialogue systems, multimodal human–machine interaction, IR, and knowledge graphs. He also led the development of the industry-first, emotion-aware conversational system that provides large-scale smart customer services to more than 300 million users of JD.COM.

Dr. He has received multiple best paper awards: ICASSP (2011); ACL (2015); and IEEE/ACM Transactions on Audio, Speech, and Language Processing (2018). Dr. He won the following major AI challenges: NIST Machine Translation Evaluation (2008), International Conference on Spoken Language Translation (2011), COCO Captioning Challenge (2015), Visual Question Answering (2017), and WikiHop-QA (2019).

Dr. He's research interests are mainly in natural language, vision, and multimodal intelligence, which is connected to deep learning, natural language processing, speech recognition, information retrieval, CV, and other relevant fields.

Dr. He's lecture topics include the progress in vision-and-language multimodal intelligence; language understanding, question answering, and dialogue—the evolving of language intelligence; and multimodal conversational AI for smart customer service systems.

Karen Livescu

Karen Livescu is an associate professor at Toyota Technological Institute at Chicago. She received her bachelor's degree in physics from Princeton University in 1996 and her Ph.D. degree in electrical engineering and computer science from MIT in 2005.

Dr. Livescu is an associate editor of *IEEE Open Journal of Signal Processing* (present); associate editor of *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2014–2017); a member of the IEEE Speech and Language Processing Technical Committee (2012–2017); and a technical cochair of the IEEE Workshop on Automatic Speech Recognition and Understanding (2015, 2017, and 2019). Outside of IEEE, she has served as a program cochair of the International Conference on Learning Representations (2019); a subject editor of *Speech Communication* journal; and an area chair for a number of speech processing, machine learning, and natural language processing conferences. She has won best paper awards at the ACL Workshop on Representation Learning for NLP in 2016 and 2017 and a Best Student Paper Award at Interspeech (2012). Her work has been acknowledged with an Amazon AWS Machine Learning Research Award (2020) and Google Research Awards (2014, 2015), and she was awarded a Clare Boothe Luce Post-Doctoral Fellowship (2005–2007) and an NSF Graduate Research Fellowship (1997–2000).

Dr. Livescu's main research interests are in speech and language processing and machine learning.

Dr. Livescu's lecture topics include embeddings for spoken language and automatic recognition of sign language in video.

Venkatesh Saligrama

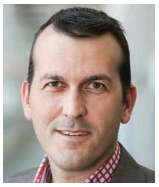
Venkatesh Saligrama is a professor in the Departments of Electrical and Computer Engineering, Computer Science (by courtesy), and Systems Engineering at Boston University, Massachusetts, USA. He is a founding faculty member of Computing and Data Sciences at Boston University. He received his Ph.D. degree from MIT.

Dr. Saligrama is an IEEE Fellow; recipient of several awards, including the Presidential Early Career Award, ONR Young Investigator Award, and the NSF Career Award; and he has received best paper awards at several conferences. He has served as an associate editor of *IEEE Transactions on Signal Processing* (2005–2007) and *IEEE Transactions on Information Theory*; has edited special issues of *IEEE Journal of Selected Topics in Signal Processing* and *IEEE Transactions on Signal Information Processing Over Networks*; has been chair of the Big Data Special Interest Group (2020); and has served on Technical Program Committees of several IEEE conferences.

Dr. Saligrama's current research interests are in machine learning, with particular emphasis on resource-efficient learning and zero-shot and limited-shot learning; statistical testing of graphs; and, more broadly, on the societal impact of AI.

Dr. Saligrama's lecture topics include machine learning on the edge and resource-efficient learning; zero-shot learning and learning with limited or no supervision in the target domain; and testing changes in graphs/networks with applications to social and physical sciences.

Dimitri Van De Ville



Dimitri Van De Ville received his M.S. degree in computer sciences and his Ph.D. degree in computer science engineering from Ghent University, Belgium, in 1998 and 2002, respectively. He was a postdoctoral fellow (2002–2005) at the lab of Prof. Michael Unser at EPFL, Switzerland, before becoming a group leader for the Signal Processing Unit at the University Hospital of Geneva, Lausanne, Switzerland, as part of the Center d'Imagerie Biomédicale. In 2009, he received a Swiss National Science Foundation professorship and since 2015 has been a professor of bioengineering at EPFL (Institute of Bioengineering), jointly affiliated with the University of Geneva (Department of Radiology and Medical Informatics), Switzerland.

Dr. Van De Ville serves as a senior editor of *IEEE Transactions on Signal Processing* (2019–present) and an editor of *SIAM Journal on Imaging Science* (2018–present). He has served as an associate editor of *IEEE Transactions on Image Processing* (2006–2009); an associate editor of *IEEE Signal Processing Letters* (2004–2006); a chair of the Bio

Imaging and Signal Processing Technical Committee (2012–2013); a founding chair of EURASIP Biomedical Image and Signal Analytics (2016–2018); and a cochair of the Biennial Wavelets and Sparsity series conferences. He is the *recipient of the Pfizer Research Award* (2012); NARSAD Independent Investigator Award (2014); and the Leenaards Foundation Award (2016).

Dr. Van De Ville's research interests include wavelets, sparsity, graph signal processing, and their applications in computational neuroimaging.

Dr. Van De Ville's lecture topics include human brain imaging: dynamics of functional brain networks, whole-brain connectomics, cognitive and clinical biomarkers, computational neuroimaging, and functional magnetic resonance imaging and graph signal processing: spectral transforms, graph slepians, nonparametric surrogate data generation, and modularity-based graph signal processing.

Dong Xu

Dong Xu is a chair in Computer Engineering and an Australian Research Council Future Fellow at the School of Electrical and Information Engineering, the University of Sydney, Australia. He received his B.Eng. and Ph.D. degrees from the University of Science and Technology of China in 2001 and 2005, respectively. Before joining the University of Sydney, he worked as a postdoctoral research scientist at Columbia University (2006–2007) and a faculty member at Nanyang Technological University (2007–2015). He was selected as the Clarivate Analytics Highly Cited Researcher in the field of Engineering in 2018 and awarded the IEEE Computational Intelligence Society Outstanding Early Career Award in 2017.

He will serve/has served as the program chair of the IEEE International Workshop on Machine Learning for Signal Processing (2021); the program cochair of the IEEE Signal and Data Science Forum (2016); the program cochair of the IEEE International Conference on Multimedia and Expo (ICME 2014); and the program cochair of the Pacific-Rim Conference on Multimedia (2012). He served as a steering committee member of ICME (2016–2017). He will serve/has served as an area chair of the AAAI Conference on Artificial Intelligence (2020); International Conference on Computer Vision (2017), ACM MM 2017; European Conference on Computer Vision (ECCV 2016); the IEEE Conference on Computer Vision and Pattern Recognition (2012); as well as a track chair of the International Conference on Pattern Recognition (2016). He is a member of the Image, Video, and Multidimensional Signal Processing Technical Committee (2018–2020) and Machine Learning for Signal Processing Technical Committee (2017–2020) and was a member in the Multimedia Signal Processing Technical Committee (2014–2019). He received the Best Associate Editor Award of *IEEE Transactions on Circuits and Systems for Video Technology* (*T-CSVT*) in 2017. He is a Fellow of IEEE and the International Association of Pattern Recognition.

Prof. Xu is/was on the editorial boards of *IEEE Transactions on Image Processing*, *T-CSVT*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, and *IEEE Transactions on Neural Networks and Learning Systems* (*T-NNLS*). He is serving/served as a guest editor of more than 10 special issues of *International Journal of Computer Vision*, *T-NNLS*, *T-CSVT*, *IEEE Transactions on Cybernetics*, *IEEE Transactions on Multimedia*, *ACM Transactions on Multimedia Computing, Communications and Applications*, *Computer Vision and Image Understanding*, and other journals.

Prof. Xu's lecture topics include transfer learning for image and video recognition; advances of machine learning in biometrics and visual applications; and deep learning for video compression.

SP

Lina J. Karam, Jay Katupitiya, Vicente Milanés, Ioannis Pitas, and Jieping Ye

Autonomous Driving: Part 2—Learning and Cognition

This special issue covering autonomous driving is presented in two parts: Part 1—Sensing and Perception was published in the July 2020 issue of *IEEE Signal Processing Magazine* (*SPM*) [1], and this issue, Part 2—Learning and Cognition. Learning and cognition models and, in particular, deep learning-based models are at the core of autonomous vehicles and automated driving. Autonomous driving and, more generally, automated driving are receiving increasing attention, and significant resources are being deployed to enable safe, reliable, and efficient automated mobility in real-world environments. Some of the needed enabling technologies include affordable sensing platforms, reliable simultaneous localization and mapping, machine learning that can effectively handle varying conditions and unforeseen events, "machine learning-friendly" signal processing, hardware and software co-design for efficient real-time performance, robust platforms that can withstand adversarial attacks and failures, and frameworks that can enable effective testing of emerging autonomous driving advances.

In this issue

The aim of this special issue is to provide researchers and professionals with tutorial-style articles covering the state of the art as well as emerging trends in the development and deployment of learning and cognition technologies for autonomous and automated driving. In particular, deep neural networks have been widely adopted and integrated as part of these technologies. Part 2 describes key concepts and the latest advances underlying the operation of such learning and cognition approaches. It also sheds light on remaining challenges that need to be addressed to enable reliable and safe operation in autonomous driving.

Digital Object Identifier 10.1109/MSP.2020.3033086
Date of current version: 24 December 2020

Overview

This issue contains eight articles. Four of them deal with the robustness of learning and perception models under adverse conditions and/or adversarial attacks. The others cover various aspects of learning and cognition for autonomous driving. The first article, "Deep Neural Network Perception Models and Robust Autonomous Driving Systems," by Shafiee et al., is concerned with perception models and robustness in autonomous driving, with a focus on adversarial attacks. In "Self-Supervised Learning for Autonomous Vehicles Perception," Chiaroni et al. address self-supervised learning and the applications that the technology enables for autonomous driving. "The Vulnerability of Semantic Segmentation Networks to Adversarial Attacks in Autonomous Driving," by Bär et al., discusses the susceptibility of convolutional neural networks (CNNs) to adversarial attacks when these CNNs are deployed for semantic segmentation in the context of autonomous driving. The authors also review existing adversarial defense strategies. The fourth article, "Object Detection Under Rainy Conditions for Autonomous Vehicles," by Hnewa and Radha, is concerned with autonomous driving under adverse weather conditions, with a focus on rainy conditions. The authors review object detection methods that are being considered for integration into autonomous vehicles. They also survey and discuss state-of-the-art methods for mitigating the effect of rain on autonomous driving.

The fifth article, "3D Point Cloud Processing and Learning for Autonomous Driving," by Chen et al., summarizes cutting-edge processing and learning methods for 3D point clouds and offers perspectives on open issues that remain to be solved. "Deep Inverse Reinforcement Learning for Behavior Prediction in Autonomous Driving," by Fernando et al., is concerned with behavior modeling in autonomous driving, with a focus on deep inverse reinforcement learning. In "Novel Arithmetic in Deep Neural Network Signal Processing for Autonomous Driving," Cococcioni et al. review current and emerging arithmetic for deep neural network (DNN) signal processing. The authors also highlight the issues in implementing DNN accelerators to achieve low-complexity processing of automotive sensor signals without compromising

accuracy. Deter et al. close the issue with "Simulating the Autonomous Future," which provides an overview of simulation tools for scene and scenario creation and describes open autonomous vehicle data sets, with a focus on constructing and validating virtual vehicle environments to replicate a range of test scenarios for autonomous driving.

Acknowledgments

We would like to extend our appreciation to Robert Heath, *SPM*'s editor-in-chief at the time of this writing, and Namrata Vaswani, *SPM*'s area editor, special issues, for their valuable input. We would also like to thank Rebecca Wollman, IEEE Signal Processing Society (SPS) publications administrator, and IEEE Magazines Managing Editor Jessica Welsh for their support. Last but not least, special thanks go to the contributors and reviewers without whom this issue would not have come to fruition. This special issue is technically sponsored by the SPS Autonomous Systems Initiative.

Guest Editors

Lina J. Karam (lina.karam@lau.edu.lb) is a professor in, and the dean of, the School of Engineering, Lebanese American University, Beirut, Lebanon. She is also an emerita professor at Arizona State University, Tempe. She is the editor-in-chief of *IEEE Journal on Selected Topics in Signal Processing* and a member of the IEEE TechRxiv Advisory Board, *IEEE Access* Editorial Board, and IEEE Signal Processing Society (SPS) Awards and Publications Boards. She served as general chair of the 2016 IEEE International Conference on Image Processing and as general cochair of the 2019 IEEE International Conference on Multimedia & Expo. She is a recipient of the National Science Foundation CAREER, NASA Technical Innovation, IEEE Region 6, IEEE SPS Best Paper, and Intel Outstanding Researcher Awards. She has written more than 240 technical publications and holds seven U.S. patents. She is a Fellow of IEEE.

Jay Katupitiya (j.katupitiya@unsw.edu.au) received his Ph.D. degree in engineering from the Catholic University of Leuven, Belgium. He is currently an associate professor at the University of New South Wales, Sydney, Australia, where he is a former deputy head of the School of Mechanical and Manufacturing Engineering and where he helped establish the Mechatronic Engineering degree program, which he subsequently led. His research interests include unmanned field vehicles, and he has contributed to the development of a number of field-scale unmanned systems for agriculture, mining, and road construction. As a secondary area of research, he conducts space robotics research, developing space robots to capture foreign objects in orbit.

Vicente Milanés (vicente.milanes@renault.com) received his Ph.D. degree in electronic engineering from the University of Alcalá, Madrid, Spain, in 2010. He has worked in the research department at Renault since 2016. Previously, he was with the AUTOPIA program at the Center for Automation and Robotics (UPM-CSIC, Spain) from 2006 to 2011 and was then awarded a two-year Fulbright fellowship at California PATH, University of California, Berkeley. In 2014, he joined the Robotics for Intelligent Transportation Systems team at the National Institute for Research in Computer Science and Automation, Rocquencourt, France. He is the author or coauthor of more than 120 refereed publications in international journals, book chapters, and conference proceedings, and he holds more than 10 industrial patents. His research interests include multiple aspects in the autonomous vehicle field.

Ioannis Pitas (pitas@aiia.csd.auth.gr) is a professor in the Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece, where he is the director of the Artificial Intelligence and Information Analysis lab. He has been a visiting professor at several universities; published more than 900 papers; contributed to 47 books; and edited, authored, and coauthored 11 books. He has more than 30,800 citations to his credit and an h-index of 83+. A past general or technical chair of four international conferences, he has participated in 70 R&D projects, primarily funded by the European Union, and was the principal investigator for most of them. His research interests include computer vision, machine learning, autonomous systems, and intelligent digital media. He is the chair of the IEEE Signal Processing Society Autonomous Systems initiative, an IEEE Distinguished Lecturer, and a Fellow of IEEE and the European Association for Signal Processing.

Jieping Ye (yejieping@didichuxing.com) is the vice president of Didi Chuxing, Beijing, China, and a professor at the University of Michigan, Ann Arbor, Michigan, USA. He has served as an associate editor of *Data Mining and Knowledge Discovery*, *IEEE Transactions on Knowledge and Data Engineering*, and *IEEE Transactions on Pattern Analysis and Machine Intelligence*. He won the National Science Foundation CAREER Award in 2010, the INFORMS Wagner Prize in 2019, the Outstanding Student Paper Award at the 2004 International Conference on Machine Learning, the Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Best Research Paper runner-up in 2013, and the SIGKDD Best Student Paper Award in 2014. His research interests include big data, machine learning, and data mining with applications in transportation and biomedicine. He is a Fellow of IEEE.

Reference

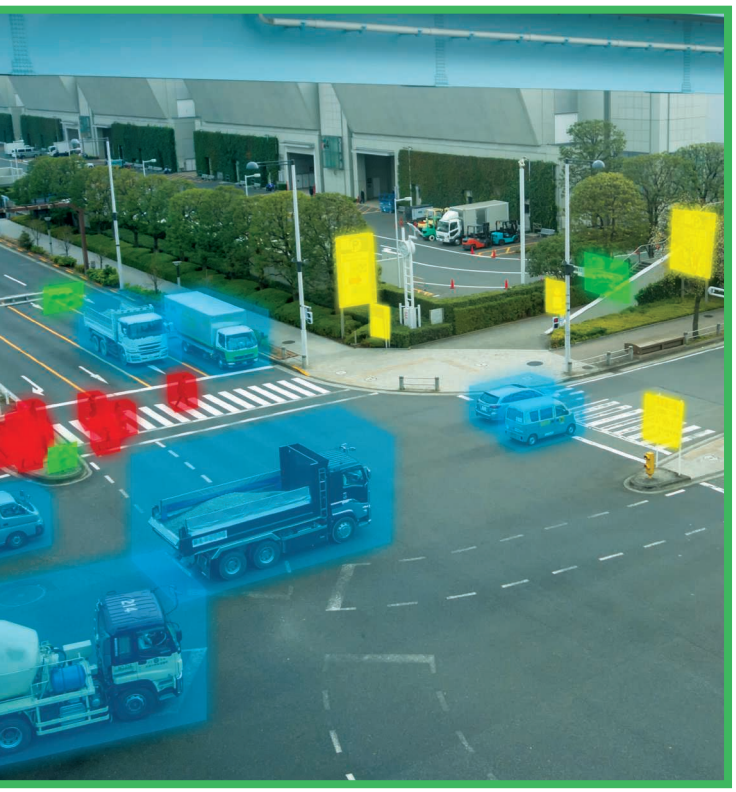
[1] L. J. Karam, J. Katupitiya, V. Milanes, I. Pitas, and J. Ye, "Autonomous driving: Part 1-Sensing and perception [From the Guest Editors]," *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 11–13, July 2020. doi: 10.1109/MSP.2020.2990330.

SP

Mohammad Javad Shafiee, Ahmadreza Jeddi,
Amir Nazemi, Paul Fieguth, and Alexander Wong

Deep Neural Network Perception Models and Robust Autonomous Driving Systems

Practical solutions for mitigation and improvement



Digital Object Identifier 10.1109/MSP.2020.2982820
Date of current version: 24 December 2020

The National Highway Traffic Safety Administration reported that more than 90% of in-road accidents in 2015 occurred purely because of drivers' errors and misjudgments, with such factors as fatigue and other sorts of distractions being the main cause of these accidents [1]. One promising solution for reducing (or even resolving) such human errors is via autonomous or computer-assisted driving systems. Autonomous vehicles (AVs) are currently being designed with the aim of reducing fatalities in accidents by being insusceptible to typical driver errors. Moreover, in addition to improved safety, autonomous systems offer many other potential benefits to society: 1) improved fuel efficiency beyond that of human driving, making driving more cost beneficial and environmentally friendly, 2) reduced commute times due to improved driving behaviors and coordination among AVs, and 3) a better driving experience for individuals with disabilities, to name a few.

Given the extensive global interest toward the deployment of AV technologies, recent studies have introduced new guidelines and regulations for speeding up AV development and pushing AVs into the market in a more effective manner. At the same time, there have been significant efforts to inform the public on the capabilities and limitations of AV systems.

The most widely used approach to categorize AV systems is to classify them based on their level of automation, as standardized by the Society of Automotive Engineers [2], ranging from level 0 (no automation) to level 5 (completely autonomous). Although a level 5 AV is the ideal, the majority of current AV systems are at levels 1 and 2. The limited autonomous driving capabilities of current AV systems are due to a range of challenges, such as the high cost of sensors, a lack of acceptance by the public, a lack of appropriate safety evaluations, and the high error rates of existing technologies. In this study, we focus on those challenges associated with level 3 or higher [2].

Although different levels of automation can lead to some variations in the developed systems, the general architecture of an autonomous driving system consists of five main components, as shown in Figure 1, grouped into two main areas of 1) perception and 2) decision/control. Perception includes all of

the hardware and software attempting to find the current state of the AV system with respect to its surrounding environment, such that this information can be used as the input to decision and control. Sensors, algorithms to process the sensed data, environmental mapping, and localizing the AV with respect to the generated map are all components within perception. Sensor devices that are typically used in AV perception include lidar, various types of visual cameras, GPS, and radar as well as Internet of Vehicles devices.

Perception includes all of the hardware and software attempting to find the current state of the AV system with respect to its surrounding environment.

Raw sensory data are processed by a variety of algorithms to generate useful information regarding the environment around the AV, of which three examples include:

1) *Object detection*: responsible for taking sensor data and detecting important objects of interest, such as traffic lights, traffic signs, road, lanes, pedestrians, and other vehicles
2) *Semantic segmentation*: responsible for segmenting the road and its participants from sidewalk and objects that are not within the road
3) *Scene reconstruction*: responsible for generating 3D scenes based on 2D images and/or lidar devices.

The perception algorithms employed are clearly a function of the automated driving level [2]. For example, whereas cruise control (based on lidar or radar sensors) is a functionality in level 1, lane centering (visible images and a segmentation model) is one of the main tasks in level 2. Level 3 driving systems are ones that can truly be considered autonomous, a level of autonomy that can allow drivers to sit back and relax. Several models are being used to perform the automated driving. Traffic signs and traffic lights are detected by object detection models, typically based on visible camera images; other cars driving in the road can be detected using object detection models by fusing sensory data from different sources; and the valid area in which the car can drive is identified by segmentation models. Although drivers can be hands-off in level 3 vehicles, they need to be ready to engage and take control at any time. In contrast, drivers can be mind-off in level 4 vehicles, but only in certain, geofenced traffic areas, or in level 5 vehicles in any situation.

Different perception models may be used independently for different tasks, such as lane segmentation, traffic sign identification, and traffic light detection (some state-of-the-art comparison results on two different computer vision applications can be found in the longer version of this manuscript [3]). However, this information should be fused for path planning and decision making, for example, to be able to associate different signs and lights to a particular lane in the road. Nonetheless, such fusion is still a challenging task, and using predefined maps can mitigate this problem to some extent in a practical scenario and increases association accuracy. Localization and mapping systems provide a map of the environment surrounding the AV and determine the current state (i.e., position and orientation) of the AV with respect to this detailed map.

Given the mapping of the environment and the AV's current state, the decision system makes decisions on what actions to be taken to optimally reach a goal; that is, the system is responsible for generating route proposals, motion planning for each of those routes, evaluating compliance with passenger preferences and the law [4], calculating safety probabilities of each proposed trajectory, and making decisions on which trajectory to select. The control system acts upon these decisions and controls the vehicle, generating appropriate commands and ensuring that the proposed trajectory is followed. There remain several challenges regarding the interaction of these two parts. In particular, the decision-control system

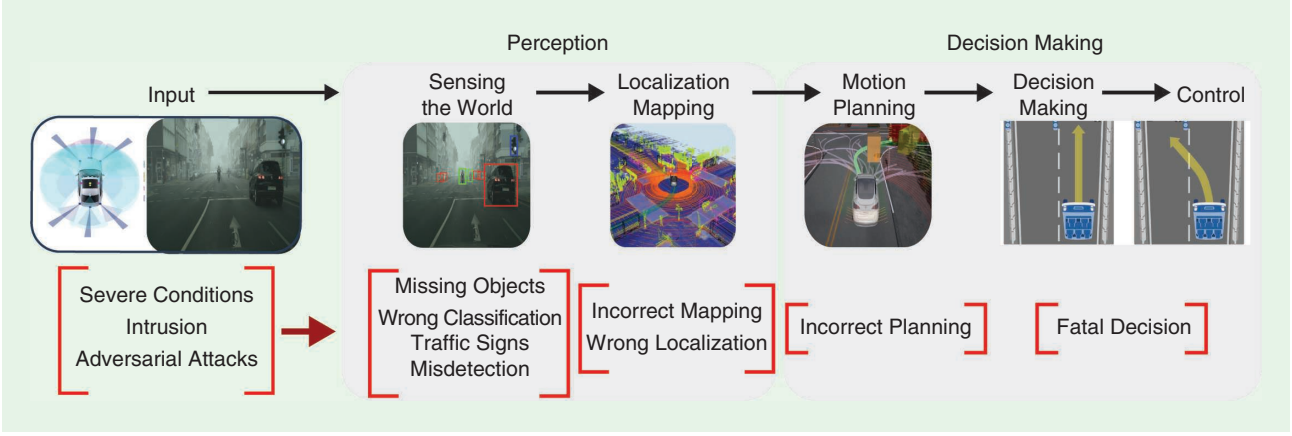


FIGURE 1. The general architecture of an autonomous driving system, which is composed of the two main components of perception and decision making. The perception system typically consists of a set of machine learning algorithms and provides a semantic understanding of the world around the vehicle. The perception system can be negatively affected by different types of intrusion algorithms, commonly referred to as *adversarial attacks*. The perception system (and especially the sensing the world module) is the first step between the outside world and the autonomous driving system; therefore, any incorrect conclusions from the perception system, due to adversarial attacks, will propagate to later components, leading to potentially fatal decisions being made. The decision-making component of an AV is composed of motion planning, decision making, and control. These modules are responsible to identify the best path for the vehicle and to actuate the car toward that.

should reevaluate the possible risks in different situations constantly and predict the intentions of human drivers around the vehicle. Effectively estimating uncertainty is very important; however, understanding human driver intention is still not a common practice in the field and is usually relaxed in the problem formulation.

A more granular description of the five components of an autonomous driving system is as follows:

1) *Sensing the world*: consists of various sensors and algorithms processing the available data and provides a semantic scene understanding [5].
2) *Localization and mapping*: computes the AV pose (location and orientation) with respect to the surrounding environment, which is frequently addressed via simultaneous localization and mapping techniques [6], well known in robotics and offline premapping.
3) *Motion planning*: provides different trajectories as sequences of states, given the environment information, initial states, and the final goal [7].
4) *Decision making*: selects the optimal trajectory while considering such other factors as safety, compliance, and so forth [8].
5) *Control*: actuates components and ensures that the AV follows the selected path [8].

The decision-control system should reevaluate the possible risks in different situations constantly and predict the intentions of human drivers around the vehicle.

The preceding modular structure requires harmonizing all components together, training all components together, and reducing the propagation of erroneous decision making from level to level. As a result of these limitations, end-to-end autonomous driving systems have been proposed, in which the components are learned as a single system. For example, Bojarski et al. [9] proposed PilotNet, a deep convolutional neural network (CNN)-based framework that takes visual data from cameras as sensory input and delivers a simple autonomous system for lane following by outputting steering angles based on a fully end-to-end approach. Caltagirone et al. [10] used a fully convolutional network to generate path proposals from lidar sensory data, a partial end-to-end paradigm.

Whether fully end to end, partial, or broken into classical components, deep learning methods are now at the heart of essentially all AV technologies. Although deep neural network (DNN) models provide the state-of-the-art performance in most scene-understanding algorithmic tasks, the robustness of neural network models has become a major concern in the research community. In the context of AV systems, extreme weather conditions and possible intrusion attacks by adversaries are two key situations where these models may be particularly vulnerable.

As seen in Figure 1, incorrect scene understanding by the sensing the world module, the first module in the sequence, can propagate wrong information through the pipeline of consecutive modules and end up with an incorrect decision, possibly a fatal outcome. False information may be generated for a variety of reasons, such as severe weather conditions, complicated urban scenarios, and intrusion by adversaries, raising major concerns on the safety and security of DNN models.

In this article, we study how adversarial attacks can upend the claimed or assessed robustness of DNNs, specifically in the application to AVs. We mainly discuss how these attacks can impose danger to AV perception and can propagate through the entire pipeline. We analyze the existing mechanisms to address these issues and describe the best practices to improve AV systems.

Robustness challenges in autonomous driving systems

In this section, we describe two main factors that can influence the robustness of machine learning models specifically for autonomous driving applications. Although several different criteria can affect the robustness of a machine learning model, here we focus on adverse conditions and adversarial attacks as prominent factors. Autonomous navigation requires an understanding of the environment around the car, and machine learning models play an important role in fulfilling this task. In particular, these models need to perform perfectly in different environments and scenarios, a degree of generalization that will be an important factor in reliable autonomous systems.

Wide variations and adverse conditions

Generalization can be defined as the ability of a trained model to deal with samples that were not seen during training. In contrast to generalization, models can also be subject to overfitting, whereby a model essentially memorizes the training samples and indeed performs with high accuracy on that training data but providing poor performance on unseen testing data. To measure the generalization of a model, the generalization error is defined as the difference between the expected and empirical errors. The empirical error is defined as the model error on the available sample data, whereas the expected error measures how the model can perform over variation of the data based on their true (but normally unknown) underlying distribution. The available test set should be large enough to be able to calculate a reliable empirical error to quantify model generalization.

Understanding generalization is key in autonomous driving systems because of the safety-critical aspects of these systems. A generalized autonomous system should perform reliably in a wide variety of different conditions and variations, particularly challenging in autonomous driving applications due to the extremely stochastic environments around the vehicle, which can cause a distribution drift. For example, an AV trained in one country and tested in another one will face inputs that did not exist in training.

Generalization and robustness issues have certainly been studied for different applications. Visual domain adaptation and generative adversarial networks (GANs) are two different techniques proposed to improve model generalization.

Tzeng et al. [11] took advantage of GAN methods to generate new samples, and they used discriminative modeling and weight sharing to improve the domain adaptation on different classification tasks. Yang et al. [12] proposed a method to keep the generalization ability of an autonomous driving system by mapping real data into a unified domain and to make decisions on the virtual data.

Any autonomous driving system needs to be functional in real-world contexts and outdoor environments, performing reliably in different weather conditions or different lighting situations. A first simple but important situation is the system performance at nighttime, particularly the functionality of vision-based models. Taking advantage of lidar or far-infrared sensors and data fusion is one approach; however, there is a body of research [13] on directly improving the robustness of vision-based models in dark environments.

Dai and Van Gool [13] proposed a new adaptation mechanism to address the semantic segmentation problem at nighttime. They used twilight images as an intermediate step to adapt the models to darker environments before fine-tuning the model for nighttime scenes, a so-called gradual model adaptation process.

Any autonomous driving system needs to be functional in real-world contexts and outdoor environments, performing reliably in different weather conditions or different lighting situations.

The main issue in generalizing models to adverse conditions is the availability of training data to use in model learning. For example, thick fog is observable only during 0.01% of typical driving in North America; therefore, having enough samples for annotation and preparing training data are very challenging. As a result, generating synthetic data may be highly desirable. Sakaridis et al. [14] introduced a new approach to synthesize foggy driving scene data from clear-weather outdoor scenes; their results showed that using synthetic data helps the model to generalize better on foggy scenarios and leads to more robust predictions.

Although providing enough data with sufficiently varied/adverse conditions is necessary, devising proper frameworks and methods to handle these conditions is crucial as well. Using lidar or far-infrared data might resolve nighttime scenarios; however, acquiring reliable lidar information might be challenging in severe conditions, such as rain, snow, or foggy situations. As a result, fusing information from different sources and sensors is a common approach. Several methods have focused on when and in which step such fusion should be performed. Methods can be divided into two main streams of early fusion [15], where the features extracted from each sensor are intertwined and knowledge fusion is performed in an early stage of the network, versus late fusion [16], where each sensor datum is processed independently and the results are combined at the end.

Intruding autonomous driving systems

Besides the natural challenges just discussed, unnatural factors may challenge autonomous driving systems as well. Essentially every software system is prone to some sort of intrusion, whether via physical access to the system or intrusions without any explicit interaction between intruder and physical system. For AV systems, it is the machine learning models that are our main concern in this article; because learned models understand the world based on sensory data, it is possible to intrude the system by providing deceiving sensory input that fools the machine learning models.

Adversarial attacks

Generally, attacks can be applied on any part of an intelligent system, such as on the training data (training-set poisoning), model output (model theft), and manipulated inputs (adversarial examples). Because careful model learning should avoid or limit the first two effects, it is the most likely attack, that of manipulating sensor inputs as adversarial attacks, that will be our focus.

Adversarial examples are those input samples that can fool a trained model with a high confidence. In particular, those that reliably fool the trained model are, at the same time, only barely (or not at all) perceptible to the human eye.

In an adversarial attack, the main goal is to find an input sample x' close to the true sample x that changes the prediction from conclusion $y = f(x)$ to conclusion $y' = f(x')$, such that $y \neq y'$. Ideally the difference between x and x', the perturbation size, is very small, making the input perturbation close to imperceptible. Typically, the perturbation is measured using an l_p norm:

$$\eta = \|x' - x\|_p. \tag{1}$$

There are two types of adversarial attacks, white-box attacks and black-box attacks [17]. In a white-box attack, the adversary has full access to the trained model and knowledge regarding the model structure and parameters to allow for a fairly informed attack. In a black-box attack, the adversarial method does not have access to the model, so the adversary has to query the target model to estimate those needed aspects of the model's interior structure. The fast gradient sign method (FGSM) [18] and DeepFool [19] are two simple but well-known examples of white-box attacks. These two attacks fall into the class of first-order adversaries, which use the gradients of the network loss function with respect to the input data to perturb input samples into adversarial examples.

Black-box attacks are more applicable in real-world scenarios and certainly more applicable in autonomous driving systems, because these systems are not accessible, in general, to outsiders. Black-box attacks may be undertaken by ensemble-based approaches, which use multiple trained models and generate common adversarial examples, which are then validated on the target model. In other words, ensemble-based attacks generate adversarial samples using a white-box attack, and the samples are then used to attack the target model in black-box form, what is known as a *transferable attack*. A second black-box approach is that

of zeroth-order optimization [20], which tries to estimate the gradient and Hessian of the network function using the inputs and outputs of the model.

Another perspective is to divide adversarial methods into targeted and nontargeted attacks. A targeted approach tries to change the input in a way to have the model predict a particular specified (targeted) class label. For example, inducing the network to recognize a stop sign as a speed limit sign could be the outcome of this method. A nontargeted method manipulates a trained model to misclassify the input data without constraining the objective toward any specific class label.

Furthermore, for applications like object detection there are positive and negative classes, and adversarial samples are generated to either fool the model to not detect the object or to classify the object to a wrong class label. For example, a vision-based autonomous driving system may misclassify a tree as a traffic sign (false positive) or may not classify a traffic sign at all (false negative). Table 1 summarizes the most well-known adversarial attack algorithms.

The main issue in generalizing models to adverse conditions is the availability of training data to use in model learning.

Attacks are characterized based on whether they are white box or black box (that is, having access to model details or not), the number of network queries to be able to generate the perturbed input (attack frequency), and how they measure the amount of perturbation added to the input data. These characteristics are described based on the most common approaches that these methods use; however, it is possible to extend these methods to change the characteristics as well. More details and the references to these methods can be found in the longer version of this manuscript [3].

Defense mechanisms

In addition to the development of algorithms to challenge/attack DNN models, a variety of defense mechanisms have been proposed to improve network robustness or mitigate the issue of facing adversarial perturbations. Model robustness against adversarial attacks can be addressed during training [18], most simply by augmenting the training set with adversarial examples. Goodfellow et al. [18] regularized the training of a DNN model by an adversarial objective function based on the fast gradient sign method:

$$\hat{J}(\theta, x, y) = \alpha \cdot J(\theta, x, y) + (1 - \alpha) \cdot J(\theta, x + \epsilon \cdot sign(\nabla_x J(\theta, x, y))), \quad (2)$$

where $J(\cdot)$ is the training objective function. Given the universal approximator theorem, Goodfellow et al. argued that if adversarial examples are encoded in the training process, the model should learn those examples and become more robust in dealing with those (and hopefully other) examples. (The universal approximator theorem states that a neural network with at least one hidden layer can represent any function to an arbitrary degree of accuracy, assuming the hidden layer to have enough units.) That is, the model is continually supplied with adversarial examples, such that they essentially resist the current version of the model, applied in an iterative process to make the model progressively more robust. Although this technique does improve the robustness of DNNs, Moosavi-Dezfooli et al. [19] showed that there is still an effective and yet universal adversarial example to fool even such adversarially trained networks; nevertheless, this approach is still the most common in increasing DNN robustness.

The robustness of DNN models can also be improved by simply performing a preprocessing step on the input data. The adversarial attacks are usually generated as an additive perturbation, almost noise [18], on the input data. The attack can be defended by heuristically removing input noise using any number of signal processing techniques, such as a moving average operator or taking advantage of compression methods [21] to remove high-frequency values.

Motivated by these ideas, Xie et al. [22] illustrated that random resizing or random padding of input images reduces the effectiveness of adversarial attacks. At the same time, it is worth noting that these changes to the input data may also reduce the accuracy of the model. Goldblum et al. [23] proposed a new distillation-based approach, which incorporates robust training of a student network. The proposed method follows a similar technique as adversarial training but in the context of distillation, where a student network is sought to mimic the teacher's outputs within an ϵ-ball of training samples.

It is also possible to take advantage of postprocessing techniques to improve the network robustness. Using an ensemble of DNNs [24] allows decision making to be aggregated across several models, improving robustness against adversarial attacks. This strategy can be combined with averaging [25] or noise perturbation [26] to make adversarial attacks more difficult. Table 2 summarizes the main techniques to improve robustness of DNNs against adversarial attacks. These approaches can be performed in preprocessing, during training, in postprocessing, or even by changing the network architecture.

Table 1. The summary characteristics of common adversarial attacks.

Method	White Box/ Black Box	Targeted/ Nontargeted	Attack Frequency	Measurement
FGSM	White box	Nontargeted	One time	Elementwise
DeepFool	White box	Nontargeted	Iterative	l_p
C&W	White box	Targeted	Iterative	l_1, l_2, l_∞
PGD	White box	Both	Iterative	l_∞
ZOO	Black box	Both	Iterative	l_2
One pixel	Black box	Both	Iterative	l_0

C&W: Carlini & Wagner; PGD: projected gradient descent; ZOO: zeroth-order optimization.

Adversarial attacks for autonomous driving systems

Autonomous driving systems are among the pioneering fields in advancing machine learning and deep learning techniques,

and complex deep learning networks are used in all phases from perception to decision making and control. As a result, the effect of adversarial attacks has been significantly investigated for autonomous driving applications.

Chen et al. [27] proposed a new approach to fool the well-known faster region-based CNN (R-CNN) object detection. They generated adversarial attacks that trick the model into not detecting stop signs in a scene. They took advantage of expectation over transformation [28], where a random noise is added in each iteration of the optimization to infer more robust adversarial perturbations. Lu et al. [29] proposed an optimization method searching through different sets of stop sign examples to craft new stop sign images that are not detectable by faster R-CNN or you-only-look-once network architectures.

However, it has been argued [30] that designing adversarial perturbation for real systems, in practice, is harder than those analyzed in the literature and in laboratory environments. Lu et al. [30] explained that because object detection and decision making in autonomous driving cars are performed based on a sequence of frames, it is far more difficult to fool a model for all frames than for one frame.

To be sure, recent algorithms have been proposed to craft adversarial attacks in real environments. Eykholt et al. [31] introduced robust physical perturbation methods that can generate perturbations under different physical conditions. Figure 2 demonstrates an example of this approach, where a physical perturbation is added to a stop sign, causing the model to misclassify the stop sign. They took several conditions into account, including environmental constraints, spatial constraints, and physical limits on imperceptibility, while optimizing the perturbation noise. These conditions are incorporated into the loss function during the optimization step. It is worth mentioning that these types of changes to traffic signs are usually considered as ordinary changes that typically do not bring up any suspicions for human inspectors. In other words, these types of perturbations might not be noticeable to human inspection as viable intrusions.

Sitawarin et al. [32] proposed a new algorithm that modifies logos and advertisements such that the deep learning model detects them as different traffic signs in a scene. Figure 3 shows an example of this scenario. The refinement on the advertisement logo can fool the targeted model to classify the image as a bicycle crossing sign.

Although most of the research in the literature has focused on DNNs taking red, green, blue (RGB) images as input, these issues are not RGB specific. Recently, Cao et al. [33] analyzed the vulnerability of lidar-based methods in autonomous driving systems. They proposed an optimization-based approach to generate real-world adversarial objects, evading the lidar-based detection framework, generating 3D objects that can fool the network and be invisible to detection.

Table 2. The different defense mechanisms to improve the robustness of DNN models against adversarial attacks.

Method	Procedure	Description
Random resizing/ random padding [21]	Preprocessing	Changing the size of the input image before passing to the network.
Adversarial training [17]	Training	Adding targeted perturbed samples to the training data.
Compression [20]	Preprocessing	Compressing and decompressing the input samples before passing to the network.
Distillation [22]	Training/ postprocessing	Training a student network given the original (teacher) network.
Ensemble [23]	Postprocessing	Aggregating the decision of several networks to mitigate the effect of adversary.
Noise perturbation [25]	Architectural change/training	Adding auxiliary noise modules in the network to neuter the effect of adversarial perturbation.

FIGURE 2. An example of a physical perturbation designed to fool a DNN model to misclassify a stop sign. (a) Adding these types of perturbations may not often draw a human's attention because they may be assumed to be regular artifacts on the traffic sign. (b) This image demonstrates a physical attack that would cause the model to misclassify the stop sign. The example is extracted directly from [31].

FIGURE 3. An example of the adversarial attack proposed by [32], where logos and advertisements are modified to cause a wrong interpretation by the DNN model. (a) An advertisement logo and (b) the modified logo designed to cause a perception model to recognize it as a bicycle crossing sign. The example is extracted from [32].

Best practices toward robust autonomous driving

In the previous section, we discussed the factors and issues influencing the robustness of an autonomous driving system. In this section, we will discuss the best practices to achieve more reliable systems for autonomous driving applications.

The key part of an autonomous driving system that might challenge the robustness of the system is machine learning models working to navigate the car. As a result, these models should be evaluated thoroughly before any deployment. Although the robustness evaluation of each model individually is a first step, the robustness of these models must also be evaluated in a combined, closed-loop framework before deployment, as evaluated by Tuncali et al. [34]. The proposed framework is a simulated environment that tries to identify problematic test scenarios by a falsification method. The proposed simulated environment is composed of four different parts: 1) the perception system (the machine learning model to be examined), 2) the controller, 3) the vehicle and the environment, and 4) the renderer. The controller takes the detection information from the machine learning model and object class information and then estimates the actual positions of the objects (e.g., pedestrians or vehicles). The role of vehicle and environment modeling and the renderer is to generate driving scenarios, improving testing and making it closer to real-world situations without deployment.

Generating test cases leveraging real-world changes in driving conditions like rain, fog, snow, and lighting conditions are important in the evaluation of autonomous driving systems.

Generating test cases leveraging real-world changes in driving conditions like rain, fog, snow, and lighting conditions are important in the evaluation of autonomous driving systems. Having the capability of generating real-world conditions would help to identify cases that lead to problematic behavior by the autonomous system. This approach decreases the need for manual testing of rare-case scenarios to some extent. To this end, Tian et al. [35] proposed DeepTest, an automated approach to generate samples subject to a wide variety of environmental conditions to detect erroneous behaviors of DNN models. The proposed framework uses linear and convolutional transformations to change the brightness or contrast or add fogginess or rain into real images. The idea is that autonomous driving systems should behave similarly for specific scenes with these types of variations; for example, that the steering angle (determined by a learned network) should not change significantly for a given scene but with different lighting or weather conditions applied. This testing approach can help to pinpoint corner cases of inconsistent system behavior.

There are several research studies examining the effectiveness of simulated data in improving the accuracy and robustness of perception models in autonomous driving. Ros et al. [36] introduced SYNTHIA, a synthetic data set of urban scenes to improve the performance of semantic segmentation models in autonomous applications. Their experimental results demonstrated that using synthetic data in conjunction with real-world data can boost the average per-class accuracy. This improvement is significant in classes having limited data set instances, such as pedestrians, cars, and cyclists. Furthermore, it has been stated by different autonomous driving groups, such as Waymo, that taking advantage of simulated driving systems improves the accuracy and performance of models in real-world scenarios and in public roads.

Simulated environments are important in finding corner cases effectively and resolving them by providing better training data or more effective learning. However, it is also important to improve the intrinsic robustness of machine learning algorithms. One of the main issues with deep learning models is that they are fairly deterministic, and furthermore, there is a lack of measuring the uncertainty in the decision-making process, making it easy to estimate and predict a network's decision making and making networks vulnerable to attack.

There have been improvements in DNN models to provide uncertainty while making predictions [37]. Model uncertainty measures a DNN's confidence associated with a prediction; for example, an AV DNN system could be exposed to test data that have a different distribution from that of the training data (for example, trained on urban data and tested in highway environments) and would ideally be expected to be uncertain in their predictions [38], where uncertainties may be aleatoric (data dependent) or epistemic (model dependent) [4]. Unfamiliar test data distributions, as a result of either insufficient original training data or a distributional shift, are common examples causing model uncertainty [39].

Gal [40] argued that softmax outputs in a DNN classifier should not be interpreted as its prediction confidence. They experimentally showed that for out-of-distribution test samples, a model can have high softmax outputs at times when predictions should be highly uncertain. Moreover, Gal and Ghahramani [37] illustrated that by using stochastic regularization techniques, DNNs can be viewed as Bayesian approximators of Gaussian process. Using this Bayesian deep learning (BDL) view, one can easily estimate a model's confidence without needing to change its architecture.

For end-to-end autonomous systems in which a DNN takes raw sensory data as input and maps them to controlling commands (e.g., steering, braking, acceleration, and so forth), finding model uncertainties is a simple procedure. Michelmore et al. [38] proposed to add dropout layers to Nvidia's PilotNet (i.e., an end-to-end self-driving car system) as a stochastic regularizer in the training step. However, they were also used at test time to extract the model confidence by computing the model output multiple times for each input image.

Figure 4 shows an illustration of two different end-to-end autonomous systems with and without model uncertainty. Bayesian DNNs can be substituted to provide uncertainty in

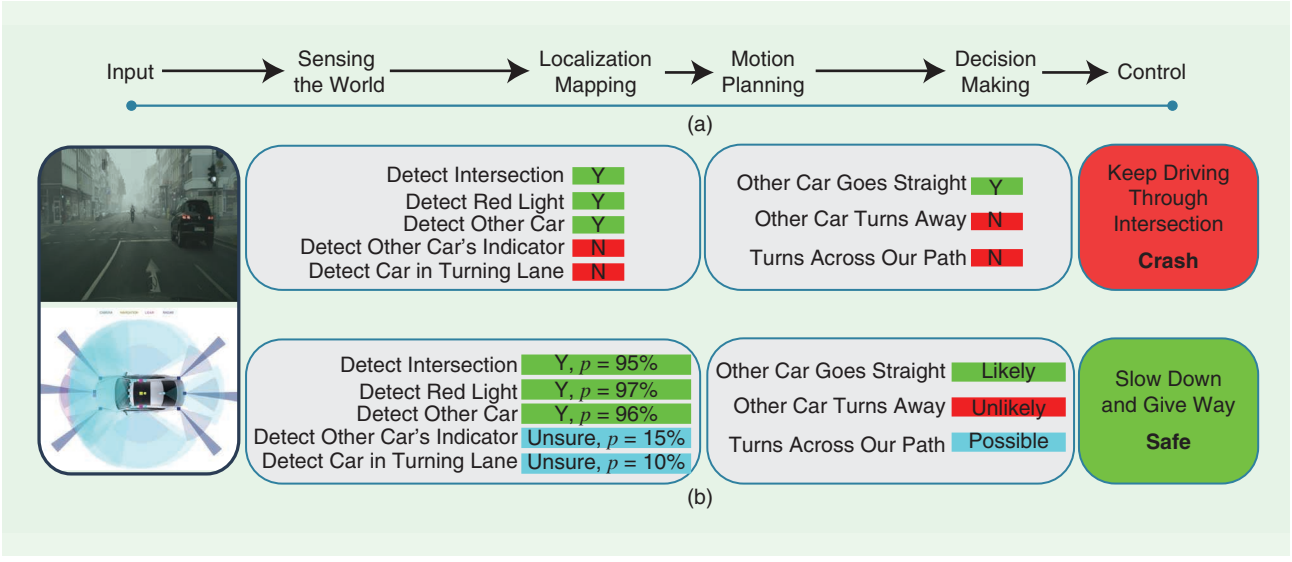


FIGURE 4. The effect of BDL. Model (a) is an end-to-end DNN model with no uncertainty evaluation, whereas model (b) is BDL based, calculating and propagating uncertainties. In this example, the BDL-based model [4] makes a better decision at the end and prevents a fatal accident. The diagram is drawn following the structure introduced in [4].

the decision-making process. As seen in Figure 4, by incorporating uncertainty in the decision-making process and taking advantage of probabilistic approaches, the system can provide more reliable predictions, better decision making, and safer actions.

However, in a modularized autonomous architecture, where the system is organized as a pipeline of subsystems, evaluating model uncertainties is a much bigger challenge than that of end-to-end architectures. McAllister et al. [4] stated that to prevent errors generated in perception subsystems to not propagate through the rest of the pipeline and affect the entire decision-making process, all of the subsystems should be equipped with BDL tools to allow uncertainty distributions to be taken into and propagated by each subsystem.

Unfamiliar test data distributions, as a result of either insufficient original training data or a distributional shift, are common examples causing model uncertainty.

Conclusions

Autonomous driving offers potentially major advantages to society, such as reducing injury and gasoline usage while also decreasing insurance costs. The past few years have witnessed remarkably significant progress toward fully automated vehicles being present on public roads. However, there do still remain concerns regarding the reliability of the computer vision and data analysis models operating within AVs and, even more significantly, their robustness in different situations. In this article, we examined the robustness of autonomous systems, focusing on proper functioning in adverse conditions/environments and in the presence of intrusions and adversarial attacks. Practical solutions to mitigate these issues and to improve the robustness of these models were discussed, ranging from extending data sets by simulated data, simulated evaluation environments to uncover corner cases, and new techniques to better calculate the uncertainty of such models in decision making, all strategies that can help to improve the performance of models in real-world applications.

The tremendous success of autonomous driving has opened a vast range of opportunities for researchers across a wide range of domains but also for members of society beginning to imagine a different future. This excitement has led to raised expectations and optimistic timelines about how soon such vehicles might be expected; however, for reasons of safety and engineering ethics, it is essential to fully understand the robustness and reliability of the designed systems.

Authors

Mohammad Javad Shafiee (mjshafiee@uwaterloo.ca) is with the Waterloo Artificial Intelligence Institute and the Department of Systems Design Engineering, University of Waterloo, Ontario, Canada. His research interests include statistical learning, graphical models with random fields, and deep learning approaches, with a focus on computer vision, machine learning, and biomedical image processing. He is a Member of IEEE.

Ahmadreza Jeddi (a2jeddi@uwaterloo.ca) is with the David Cheriton School of Computer Science, University of Waterloo, Ontario, Canada. His research interests include deep learning, computer vision, and big data frameworks and analytics.

Amir Nazemi (anazemi@uwaterloo.ca) is with the Waterloo Artificial Intelligence Institute and the Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada. His current research interests

include adversarial machine learning, continual learning, and video object segmentation.

Paul Fieguth (pfieguth@uwaterloo.ca) is with the Waterloo Artificial Intelligence Institute and the Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada. His research interests include machine learning for computer vision and statistical image processing, with a focus on hierarchical algorithms for large problems, particularly in simplifying modeling and interpretation. He is a Senior Member of IEEE.

Alexander Wong (a28wong@uwaterloo.ca) is with the Waterloo Artificial Intelligence Institute and the Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada. His research interests include artificial intelligence, computer vision, computational imaging, and multimedia systems, with a particular focus on scalable deep learning and explainable deep learning for operational scenarios. He is a Senior Member of IEEE.

References

[1] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," National Center for Statistics and Analysis, Washington, D.C., Tech. Rep. DOT HS 812 115, 2015.

[2] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, SAE International J3016, 2016.

[3] M. J. Shafiee, A. Jeddi, A. Nazemi, P. Fieguth, and A. Wong, Deep neural network perception models and robust autonomous driving systems. 2020. [Online]. Available: arXiv:2003.08756

[4] R. McAllister, Y. Gal, A. Kendall, M. Van Der Wilk, A. Shah, R. Cipolla, and A. V. Weller, "Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning," in *Proc. Int. Joint Conf. Artificial Intelligence*, 2017, pp 47454753. doi: 10.24963/ijcai.2017/661.

[5] H. Cheng, *Autonomous Intelligent Vehicles: Theory, Algorithms, and Implementation*. New York: Springer-Verlag, 2011.

[6] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Trans. Robot. Autom. (1989–June 2004)*, vol. 17, no. 3, pp. 229–241, 2001. doi: 10.1109/70.938381.

[7] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *J. Guid. Control Dyn.*, vol. 25, no. 1, pp. 116–129, 2002. doi: 10.2514/2.4856.

[8] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, 2016. doi: 10.1109/TIV.2016.2578706.

[9] M. Bojarski et al., End to end learning for self-driving cars. 2016. [Online]. Available: arXiv:1604.07316

[10] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "Lidar-based driving path generation using fully convolutional neural networks," in *Proc. 2017 IEEE 20th Int. Conf. Intelligent Transportation Systems (ITSC)*, pp. 1–6. doi: 10.1109/ITSC.2017.8317618.

[11] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 7167–7176.

[12] L. Yang, X. Liang, and E. Xing, Unsupervised real-to-virtual domain unification for end-to-end highway driving. 2018. [Online]. Available: arXiv:1801.03458

[13] D. Dai and L. Van Gool, "Dark model adaptation: Semantic image segmentation from daytime to nighttime," in *Proc. 2018 21st Int. Conf. Intelligent Transportation Systems (ITSC)*, pp. 3819–3824. doi: 10.1109/ITSC.2018.8569387.

[14] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data," *Int. J. Comput. Vis.*, vol. 126, no. 9, pp. 973–992, 2018. doi: 10.1007/s11263-018-1072-8.

[15] M. Bijelic, F. Mannan, T. Gruber, W. Ritter, K. Dietmayer, and F. Heide, Seeing through fog without seeing fog: Deep sensor fusion in the absence of labeled training data. 2019. [Online]. Available: arXiv:1902.08913

[16] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *Proc. 2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 1–8. doi: 10.1109/IROS.2018.8594049.

[17] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, Adversarial attacks and defences: A survey. 2018. [Online]. Available: arXiv:1810.00069

[18] I. J. Goodfellow, J. Shlens, and C. Szegedy, Explaining and harnessing adversarial examples. 2014. [Online]. Available: arXiv:1412.6572

[19] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 1765–1773.

[20] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop Artificial Intelligence and Security*, 2017, pp. 15–26. doi: 10.1145/3128572.3140448.

[21] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten, Countering adversarial images using input transformations. 2017. [Online]. Available: arXiv:1711.00117

[22] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *Proc. IEEE Int. Conf. Computer Vision*, 2017, pp. 1369–1378.

[23] M. Goldblum, L. Fowl, S. Feizi, and T. Goldstein, Adversarially robust distillation. 2019. [Online]. Available: arXiv:1905.09747

[24] S. Kariyappa and M. K. Qureshi, Improving adversarial robustness of ensembles with diversity training. 2019. [Online]. Available: arXiv:1901.09981

[25] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019, pp. 501–509.

[26] A. Jeddi, M. J. Shafiee, M. Karg, C. Scharfenberger, and A. Wong, "Learn2perturb: An end-to-end feature perturbation learning to improve adversarial robustness," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2020, pp. 1–13.

[27] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau, "Shapeshifter: Robust physical adversarial attack on faster R-CNN object detector," in *Proc. Joint European Conf. Machine Learning and Knowledge Discovery Databases*, 2018, pp. 52–68. doi: 10.1007/978-3-030-10925-7_4.

[28] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. 2018. [Online]. Available: arXiv:1802.00420

[29] J. Lu, H. Sibai, and E. Fabry, Adversarial examples that fool detectors. 2017. [Online]. Available: arXiv:1712.02494

[30] J. Lu, H. Sibai, E. Fabry, and D. Forsyth, No need to worry about adversarial examples in object detection in autonomous vehicles. 2017. [Online]. Available: arXiv:1707.03501

[31] K. Eykholt et al., Robust physical-world attacks on deep learning models. 2017. [Online]. Available: arXiv:1707.08945

[32] C. Sitawarin, A. N. Bhagoji, A. Mosenia, P. Mittal, and M. Chiang, Rogue signs: Deceiving traffic sign recognition with malicious ads and logos. 2018. [Online]. Available: arXiv:1801.02780

[33] Y. Cao, C. Xiao, D. Yang, J. Fang, R. Yang, M. Liu, and B. Li, Adversarial objects against lidar-based autonomous driving systems. 2019. [Online]. Available: arXiv:1907.05418

[34] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *Proc. 2018 IEEE Intelligent Vehicles Symp. (IV)*, pp. 1555–1562. doi: 10.1109/IVS.2018.8500421.

[35] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th Int. Conf. Software Engineering*, 2018, pp. 303–314. doi: 10.1145/3180155.3180220.

[36] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 3234–3243. doi: 10.1109/CVPR.2016.352.

[37] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Machine Learning*, 2016, pp. 1050–1059.

[38] R. Michelmore, M. Kwiatkowska, and Y. Gal, Evaluating uncertainty quantification in end-to-end autonomous driving control. 2018. [Online]. Available: arXiv:1811.06817

[39] S. Shafaei, S. Kugele, M. H. Osman, and A. Knoll, "Uncertainty in machine learning: A safety perspective on autonomous driving," in *Proc. Int. Conf. Computer Safety, Reliability, and Security*, 2018, pp. 458–464. doi: 10.1007/978-3-319-99229-7_39.

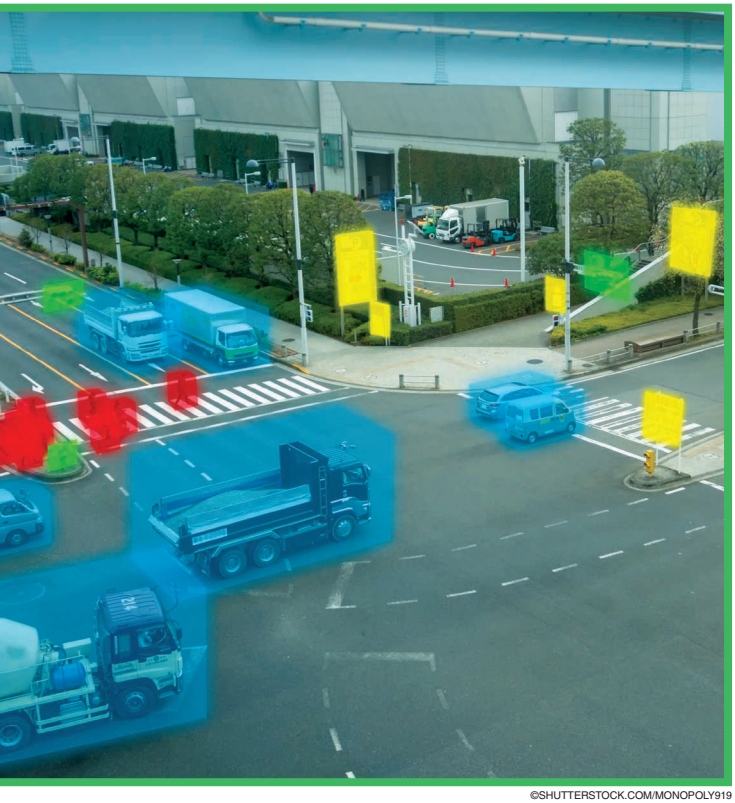
[40] Y. Gal, "Uncertainty in deep learning," Ph.D. thesis, Univ. Cambridge, U.K., 2016.

SP

Florent Chiaroni, Mohamed-Cherif Rahal,
Nicolas Hueber, and Frédéric Dufaux

Self-Supervised Learning for Autonomous Vehicles Perception

A conciliation between analytical and learning methods



The interest in autonomous driving has continuously increased in the last two decades. However, to be adopted, such critical systems need to be safe. Concerning the perception of the ego–vehicle environment, the literature has investigated two different types of methods. On the one hand, traditional analytical methods generally rely on handcrafted designs and features while on the other hand, learning methods aim at designing their own appropriate representation of the observed scene.

Analytical methods have demonstrated their usefulness for several tasks, including keypoints detection [1], [2], optical flow, depth map estimation, background subtraction, geometric shape detection, tracking, and simultaneous localization and mapping (SLAM) [3]. Those methods have the advantage of being easily explainable. However, it is difficult to apply them to high-dimensional data for semantic scene analysis. For example, identifying the other road users or understanding the large variety of situations present in an urban scene requires extracting complex patterns from high-dimensional data captured by camera sensors.

Learning methods are now the most adapted in terms of prediction performances for complex pattern-recognition tasks [4] implied in autonomous vehicles scene analysis and understanding. However, state-of-the-art results are often obtained with large and fully labeled training data sets [5]. Hand labeling a large data set for a given specific application has a cost. Another difficulty is apprehending the learned representations from end to end. To overcome the former limitation, transfer learning and weakly supervised learning methods have been proposed. Some of them can exploit partially labeled [6], [7] or noisy labeled data sets [8], [9]. Concerning the latter problem, under mild theoretical assumptions on the learning model, we can interpret the predicted outputs. For instance, it is possible to automatically detect overfitting of the training [10], to estimate the fraction of mislabeled examples [11], or to estimate the uncertainty in the prediction outputs [12].

In addition to the difficulty of obtaining a large labeled training data set, another challenge for learning methods is to prevent unpredictable events. Indeed, some scenes unseen

Digital Object Identifier 10.1109/MSP.2020.2977269
Date of current version: 24 December 2020

during training can appear frequently in the context of the autonomous vehicle. For instance, an accident on the road can drastically change the appearance and location of potential obstacles. Thus, even if it is possible to predict when the model does not know what it observes, it may be interesting to confirm it through an analytical process and to adapt the learning model to this novel situation.

It turns out that self-supervised learning methods, consisting of combined analytical and learning techniques, have been shown in the literature to have the ability to address such issues.

It turns out that self-supervised learning (SSL) methods, consisting of combined analytical and learning techniques, have been shown in the literature to have the ability to address such issues. For instance, the SSL system in [13] won the 2005 DARPA Grand Challenge thanks to its adaptability to changing environments. SSL for autonomous driving vehicles perception is most often based on learning from data that is automatically labeled by an upstream method, similar to feature learning in [14]. In this article, we address the following aspects of SSL.

- Abilities such as sequential environment adaptation on the application time, referred to as *online learning*, self-supervised evaluation, unnecessity of hand-labeled data, fostering of multimodal techniques [13], and self-improvement are included. For example, iterative learning progressively reduces the corrupted predictions [15].
- We also address tasks made possible thanks to those advantages, such as depth map estimation [15], [16], temporal predictions [17], moving obstacles analysis [18], and long-range vision [13], [19]. For example, the SSL system in [19] learns to extrapolate the appearance of obstacles and traversable areas observable by stereo vision in a short range and to identify them at a longer distance beyond the detection range of the stereo vision.

While the cited SSL techniques are designed for specific use case applications, they present some similarities. In particular, a shared underlying idea is to learn to predict, from given spatiotemporal information (e.g., a single camera frame [13], [16], [19], [21], [22]), something (e.g., traversable area segmentation [13], [19], depth estimation [16], or moving obstacles segmentation [21], [22]) that can be automatically labeled in another way using additional spatiotemporal information (e.g., a stereo-vision camera [16], [19], temporal sequence [23], or depth sensor [13]).

We propose to highlight those interdependencies. In this way, we aim to provide readers with some analytical, learning, and hybrid tools that are transversal to the final application use cases. In addition, the limitations of the presented frameworks are discussed and highlighted in Table 1, as well as the perspectives of improvement for self-evaluation, self-improvement, and self-adaptation, to address future autonomous driving challenges.

Analytical methods

Before the recent growing interest in deep learning methods, many analytical methods (without learning) were proposed, bringing baseline reference tools for multiple challenging perception tasks in the context of autonomous driving. Some of the most investigated tasks considered in this article are briefly introduced.

- *Keypoints feature detection*: Before analyzing the sensor data from a relatively high level, analytical techniques often require performing spatial or temporal data matching using feature-detection methods. More specifically, these methods consist of detecting and extracting local features in the sensor data. These handcrafted features can be small regions of interest [24]. To enable the matching of sensor data, captured from the same scene with different spatial or temporal points of view, such features need to be as invariant as possible to scale, translation, and rotation transformations. The most common sensor data is an image captured by a camera. In this case, competitive feature detectors include scale-invariant feature transform (SIFT) [1], speeded-up robust features [25], and oriented fast and rotated brief [26]. When a depth sensor is also available, the depth information can be exploited to further improve feature detection. For instance, the tridimensional rotational invariant surface keypoints method [2] is specifically designed for red-green-blue-depth (RGB-D) images. More recently, lidar has enabled the acquisition of point clouds. To tackle this new form of sensor data, some feature-detection techniques are derived from image ones (e.g., Harris and SIFT). Alternatively, some new approaches such as intrinsic shape signatures [27] are exclusively designed for point clouds. From a practical point of view, implementing common image feature detectors can be found in image libraries as OpenCV [28] and in point clouds libraries as PCL [29]. Feature detectors are exploited by several autonomous driving perception techniques that require matching of sensor data, including optical flow, disparity map, visual odometry, SLAM, and tracking techniques.
- *Optical flow*: This dense [30] or sparse [31] motion pattern can be obtained by

Table 1. A comparison of state-of-the-art analytical, learning, and SSL methods for autonomous vehicle perception challenges.

Methodology	No Hand Labeling	Dense Complex Pattern Analysis	Online Self-Evaluation and Adaptation	Knowledge Extrapolation	Low-Cost Sensor Requirements
Analytical	+++	+	++	+	+
Supervised learning	+	+++	+	+	+++
SSL	+++	++	+++	+++	++

+: inappropriate; ++: intermediary; +++: appropriate.

computing points or features transformations throughout a temporal images sequence captured from a static or mobile ego–camera point of view. In the context of autonomous driving perception, optical flow is relevant for background subtraction and motion estimation of the ego–vehicle and surrounding moving obstacles, as proposed by Menze and Geiger [32]. It can also be exploited, as in the case of a monocular mobile camera without any additional information, for relative depth map estimation [33] of the surrounding static environment.

- *Depth map estimation*: This task aims at providing image pixel depths, namely the relative or absolute distance between the camera and captured objects. Several techniques exist to address this undertaking. One of the most common and effective approaches is to compute a disparity map from a stereo camera. Combined with the extrinsic camera parameters, such as the baseline distance separating both cameras, the disparity map can be converted into an inversely proportional absolute depth map. Another approach is to project lidar points on some of the camera image pixels. It also requires extrinsic spatial and temporal calibrations between both sensors. As mentioned previously, a relative depth map of a static scene can also be derived from the optical flow obtained with a moving camera. Under some assumptions, such as with additional accurate GPS and IMU sensors, information concerning the absolute pose transformations of the moving camera, the absolute depth map can then be obtained. The depth map can also be directly obtained with some RGB-D sensor and is interesting for identifying the 3D shape of objects in the scene. More specifically, in autonomous driving, an absolute depth map is relevant for estimating the distance between the ego–vehicle and detected obstacles. However, we should note that absolute depth map estimation is more challenging compared to a relative depth map, as at least two jointly calibrated sensors are necessary. Consequently, it implies a relatively higher financial cost in production. Moreover, extrinsic calibrations can be sensitive to the ego–vehicle physical shocks. Finally, such sensor fusions can only offer depth estimation in a limited range, due to fixed baselines with stereo cameras or sparse point cloud projections with dispersive lidar sensors. Nevertheless, a relative depth map is sometimes sufficient to detect obstacles and traversable areas. For example, considering the traversable area as a set of planes in the depth map 3D point cloud projection, some template-matching techniques can be used [19].
- *Geometric shape detection*: Techniques such as Hough transform and random sample consensus (RANSAC) [34] were initially aimed at identifying some basic geometric shapes such as lines for lane marking detection, ellipses for traffic lights detection, or planes for road segmentation. To deal with sophisticated template-matching tasks, techniques such as the Hough transform have been generalized for arbitrary shape detection [35]. Nonetheless, these techniques require an exact model definition of the shapes to be detected. Consequently, they are sensitive to noisy data and are impractical for detecting complex and varying shapes such as obstacles encountered in the context of autonomous driving. Indeed, such objects typically suffer from outdoor illumination changes, background clutter, or nonrigid transformations.
- *Motion tracking*: This technique aims at following some data points, features, or objects through time. Tracking filters such as the extended Kalman filter (EKF) predict the next motion using prior motion knowledge. Conversely, object tracking can be achieved by features or template matching between consecutive video frames. Pixel points and features tracking is relevant for dense or sparse optical flow, as well as visual odometry estimation [36]. Conversely, obstacle object tracking is very important in autonomous driving for modeling or anticipating their trajectories into the ego–vehicle environment. However, on the whole, while some techniques integrate uncertainty, they remain limited when managing complex real motion patterns. Predicting pedestrian and driver behaviors typically requires knowledge of their context. Moreover, mobile obstacles' appearances can drastically change depending on their orientation.
- *SLAM techniques*: The complementarity between the concepts mentioned has been demonstrated through the problem of simultaneously localizing the ego–vehicle and mapping the surrounding environment (SLAM) [3]. Features matching provides the pose transformations of the moving ego–vehicle. In turn, 3D scaled projections of depth maps combined with the successive estimated poses provide the environment mapping. Tracking filters and template matching may offer some robustness against sensor data noise and drifting localization estimation, as proposed in the EKF SLAM [37] and SLAM++ [38] approaches, respectively.

In autonomous driving, an absolute depth map is relevant for estimating the distance between the ego–vehicle and detected obstacles.

To summarize, analytical methods can successfully handle several perception tasks of significant interest in the context of autonomous driving. In particular, a self-driving vehicle embedding these techniques is able to carry out physical analysis such as the 3D reconstruction modeling of the environment, as well as dynamic estimations concerning the ego–vehicle and the encountered surrounding mobile obstacles. Moreover, these techniques have the advantage of being easily explainable in terms of design. This facilitates the identification and prevention of failure modes. However, some critical limitations persist, including the following issues.

- A lack of landmarks and salient features combined with the presence of dynamic obstacles may entail a severe degradation of feature detection and matching.
- Severe noisy sensor data induces the same risks.

- It is impossible to achieve dense real-time semantic scene analysis of environments including a wide range of complex shape patterns.

By recognizing and predicting complex patterns with generalization abilities, learning methods aim at overcoming such issues, as developed in the next section.

Learning methods

Learning methods have demonstrated state-of-the-art prediction performances for semantic analysis tasks in the last decade. Autonomous driving is a key application that can greatly benefit from these recent developments. For instance, learning methods have been investigated in this context, for identifying the observed scene context using classification, for detecting the other road users surrounding the ego–vehicle, for delineating the traversable area surface, or for dynamic obstacle tracking.

- *Classification*: This learning method aims at predicting, for a given input sensor sample, an output class label. To deal with high-dimensional data containing complex patterns, the first stage is generally to extract relevant features using handcrafted filters or learned feature extractors. For image feature extraction, the state-of-the-art techniques use convolutional neural network (CNN) architectures composed of a superposition of consecutive layers of trainable convolutional filters. Then, a second stage is to apply a learning classifier on the feature maps generated as output of these filters. Some commonly used classifiers are the support vector machine (SVM) and the multilayer perceptron (MLP). Both require training that is usually performed in a fully supervised way on labeled data. The CNN and MLP deep learning models are trained by backpropagating the output prediction error on the trainable weights up to the input. Concerning the evaluation of these models, a test data set is required, which is labeled as well. The accuracy metric is commonly used to evaluate the prediction performances, while the F1 score, a harmonic mean of the precision and recall, is relevant for information retrieval. An image-classification application example in autonomous driving is for categorizing the context of the driven road [39].
- *Detection*: Generally, this learning method localizes the regions of interest in a visual sensor data, which in turn can be classified. A commonly used strategy, invariant to scales and translations, applies an image classifier on sliding windows over an image pyramid. Then, several advanced competitive image-detection techniques, such as region proposal networks [40] or Yolo [41], have been developed more recently and have been adapted for road users detection [39].
- *Segmentation*: As its name suggests, this task provides a segmentation of visual sensor data. The following three distinct applications can be considered.
  - *Semantic segmentation* assigns a semantic class label to each pixel. An example is road segmentation [39]. State-of-the-art methods for autonomous vehicle perception can exploit an autoencoder architecture, but also dilated or atrous convolutions, as well as an image context modeling strategy as reviewed in [42]. In the context of image segmentation, these models are trained to predict as output a pixel-wise classification of the input image.
  - *Instance segmentation* aims at detecting and segmenting each object instance. Examples include foreground segmentation and object detection of potentially moving obstacles [43].
  - *Panoptic segmentation* [4] is a unification of the two previously mentioned segmentation tasks.

Some models dealing with these segmentation tasks have been adapted for performing per-pixel regression tasks such as dense optical flow estimation [44] or depth map estimation [45].

- *Temporal object tracking*: This task follows the spatial location of selected objects along a temporal data sequence. State-of-the-art learning techniques use variants of the recurrent neural network (RNN) model [46]. Compared to standard filtering techniques, RNNs have the ability to learn complex and relatively long-term temporal patterns in the context of autonomous driving.

Obstacle object tracking is very important in autonomous driving for modeling or anticipating their trajectories into the ego–vehicle environment.

These methods can be combined in a unified framework, for instance by sharing the same encoded latent feature maps, as proposed in MultiNet [39] for joint real-time scene classification, vehicle detection, and road segmentation. While demonstrating competitive prediction performances, the mentioned learning techniques are fully supervised. In other words, they have the limitation to require large-scale fully annotated training data sets in common. To alleviate this issue, the following other learning strategies have been investigated.

- *Weakly supervised learning*: These techniques can be trained with a partially labeled data set [6] and eventually with a fraction of corrupted labels [8], [9]. Advantageously, these approaches drastically reduce the need for labeled data.
- *Clustering*: These approaches can be defined as an unlabeled classification strategy that aims to gather the data depending on their similarities without supervision. A huge advantage is that no labels are required. However, if it is necessary to associate the resulting clusters with semantic meanings understandable by humans, then a final step of punctual per-cluster hand labeling is required. State-of-the-art methods [47] dealing with complex real images mix trainable feature extractors with standard clustering methods such as a Gaussian mixture model (GMM) [48].
- *Pretraining*: Some relevant generic visual feature extractors can be obtained by performing a preliminary pretraining of the CNN model on unlabeled or labeled data coming from the target application domain [19] or even from a different one [49].

We also note that to apprehend the learned representations from end to end, it is possible to identify training overfitting [10] of deep learning models without validation test supervision. Furthermore, some learning approaches can estimate the prior of a noisy labeled training data set [11] or the model uncertainty [12], [50].

Now that some considered analytical and learning methods have been treated separately, the next section shows the complementarity between these two different types of approaches through several SSL systems developed in the context of the perception of the autonomous driving vehicle.

SSL autonomous driving applications

In the context of autonomous driving applications, we can organize the SSL perception techniques in two main categories:

1) High-level scene understanding including road segmentation to discriminate the traversable path from obstacles to be avoided; dynamic obstacles detection and segmentation; and obstacles tracking and motion anticipation predictions.
2) Low-level sensor data analysis, with a particular focus on dense depth map estimation, which is potentially relevant input data information for dealing with the previously enumerated scene understanding challenges.

If the road aspect changes suddenly, then it is impossible with this SSL strategy to correctly segment this novel road region.

Scene understanding

To navigate safely, smoothly, or swiftly when it is required, a self-driving car must perform path planning adapted to the surrounding environment. The planned trajectories must pass through traversable areas, while ensuring that surrounding static and dynamic obstacles are avoided. For this purpose, it is necessary to detect and delineate these potential obstacles in advance, and also to anticipate future positions of the mobile ones.

Traversable area segmentation

A traversable area can be identified by performing its segmentation over the mapped physical environment. Two different strategies have been successively applied. The former is mainly dedicated to off-road unknown terrain crossing. It entails fully self-supervised training systems (i.e., without hand-labeled data). The latter, which appeared more recently, is dedicated to urban road analysis. The main difference is that the SSL online systems are initialized with a supervised pretraining on hand-labeled data. This preliminary step aims at replacing the lack of landmarks on urban asphalt roads that have uniform textures, by prior knowledge.

SSL off-road systems

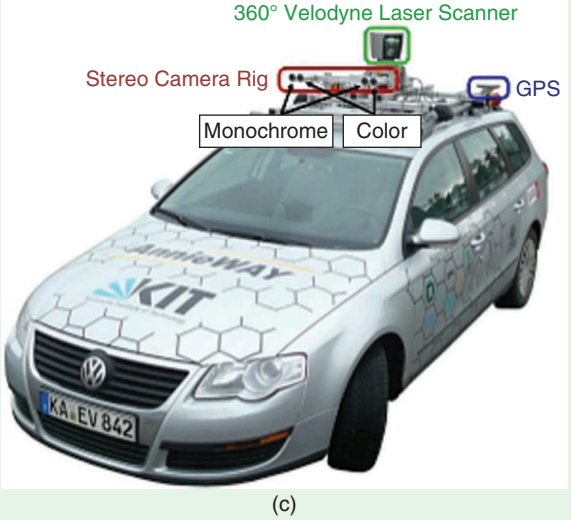
A road segmentation is proposed in [51] by exploiting temporal past information concerning the road's appearance on monocular camera images. It considers the close observable area on the current monocular camera frame in front of the car as a traversable road. Next, it propagates optical flow on this area from the current frame up to the past captured frames. Then, it can deduce this close area's appearance when it was spatially farther in the past. This past appearance of the actual close traversable area is exploited for producing horizontal line templates using the sum of squared differences matching measure. It is combined with a Hough transform-based horizon detector to define the image horizontal lines of pixels on which to apply the horizontal 1D template matching. Next, with the assumption that the actual distant traversable area has roughly the same appearance as the actual close area had in the past, the 1D templates are applied over the current frame to segment the distant traversable area. If the best template-matching measure changes abruptly, then it is supposed that the ego–vehicle is going off the road or that the road's appearance has suddenly and drastically changed. The approach in [51] is relevant for providing a long-range road image segmentation using a monocular camera only. However, a major issue is the critical assumption of considering the close area as always traversable. If the road aspect changes suddenly, then it is impossible with this SSL strategy to correctly segment this novel road region.

Another SSL road-segmentation approach is proposed in [13], dealing with this issue. Instead of using temporal information with the assumption that the close area is always traversable, and in addition to the monocular camera, a lidar sensor is used for detecting the obstacles close to the ego–vehicle. Projected on the camera images, lidar depth points enable automatically and sparsely labeling the close traversable area on images pixels. Then, a learning GMM is trained online to recognize the statistical appearance of these sparse analytically labeled pixels. Next, the trained model is applied on the camera pixels, which cannot benefit from the sparse lidar points projection, to classify them as road pixels or not. In this way, the vehicle can anticipate the far obstacles observable in the monocular camera images, but not in the dispersive lidar data. This SSL system enabled the Stanley self-driving car, presented in Figure 1(a), to win the DARPA Grand Challenge [52] by smoothing the trajectories and increasing the vehicle speed thanks to the anticipation of distant obstacles. This highlighted the interest in combining multiple sensors in a self-driving car.

More recently, with the growing interest for deep learning methods, Hadsell et al. [19] propose to use a CNN classifier model instead of the earlier template-matching or GMM learning techniques. Moreover, an additional paired camera (i.e., stereo camera) replaces the lidar sensor as in [13]. As off-road terrain traversable areas are not always completely flat, a multiground plane segmentation is performed in [19], on the short-range point cloud projection, obtained with the stereo-vision depth map by using a Hough transform plane detector. This technique provides several automatic labels for image patches that are observable in the short-range region. Then, addressing the long-range vision segmentation, the authors first train a classifier to predict patch labels automatically estimated within the short-range region. Next, the trained classifier predicts the same labels on the long-range

(a)

(b)



(c)

FIGURE 1. Three self-driving cars are shown: (a) the self-driving car Stanley that won the DARPA Grand Challenge using an SSL system equipped with a calibrated monocular camera and a lidar sensor [13] (Source: Wikipedia); (b) the autonomous mobile robot LAGR, which integrates another SSL vision approach [19] able to identify online the obstacles and road segmentation from a short-range stereo vision up to a long-range monocular vision (Source: DARPA); and (c) the car equipped with the perception sensors used to generate the KITTI data set [20].

observable image region patches by using a sliding window classification strategy.

Concerning the prediction performances, the authors have demonstrated that the online fine tuning of the classifier and the offline pretraining of its convolutional layers using an unsupervised autoencoder architecture can improve prediction performances. Moreover, an interesting point to note is that instead of using uncertainty or noisy labeled learning techniques, the authors created transition class labels for the boundary image surfaces separating the obstacles from the traversable area. Finally, from an initial 11–12-m short-range stereo vision, the developed SSL system is able to extrapolate a long-range vision up to 50–100 m. Nonetheless, to estimate the short-range stereo 3D reconstruction, including planar sets of points corresponding to the off-road traversable area, this approach requires the presence of salient visual features in the road regions. This may be impractical, for instance on the uniform visual texture of asphalt roads commonly encountered in urban scenarios, as illustrated in Figure 2.

Pretrained SSL urban road systems

Some other online SSL techniques deal with this issue by exploiting a classifier pretrained offline on hand-labeled data [53], [54]. The automatic labeling step previously performed with analytical methods is replaced in [53] by an SVM classifier pretrained offline using a human annotated data set. In this way, this approach is also compatible with uniform asphalt road surfaces. However, compared to the previously presented SSL off-road approaches, it requires hand-labeled data.

A hybrid path-segmentation technique is proposed in [54]. It combines a 3D traversability cost map obtained by stereo vision and an SVM classifier pretrained offline over a human annotated data set. Six different ground surfaces are considered to train the classifier: asphalt, large gravel, small gravel, soil, grass, bushe,s and stones. The strategy is as follows. Online, SVM predictions refine the cost map concerning the flat regions. In turn, the 3D traversability cost map, obtained without supervision, is exploited to update some misclassifications of the pretrained classifier online.

To sum up these road-segmentation SSL approaches, we notice that while the sensor data and the analytical and learning models are different, the online process remains essentially the same. The first stage always consists of generating automatic labels by using additional temporal [51], sensor [13], [19], or prior knowledge information [53], [54]. Then, a second stage trains or updates an online classifier, such that it can be used to provide a long-range or refined road segmentation. Overall, while the short-range visions based on depth sensors aims at ensuring the reliable detection of close obstacles, using such SSL vision techniques in static environments directly enables anticipating the path-planning evolution at a long range. Consequently, it is possible to increase the maximum speed velocity of the self-driving car [13] while preserving smooth trajectories [19].

Now that we have presented some SSL techniques dealing with limited depth sensors in static environments, we focus on dynamic obstacles, as they represent the other potential road users interacting with the ego–vehicle in the shared surrounding environment.

Dynamic obstacles analysis

This section starts by presenting an SSL approach [21] based on a binary per-pixel segmentation of dynamic obstacles. Then, we introduce its extension [18] for dynamic obstacles instance segmentation, such that the different road users can be separated.

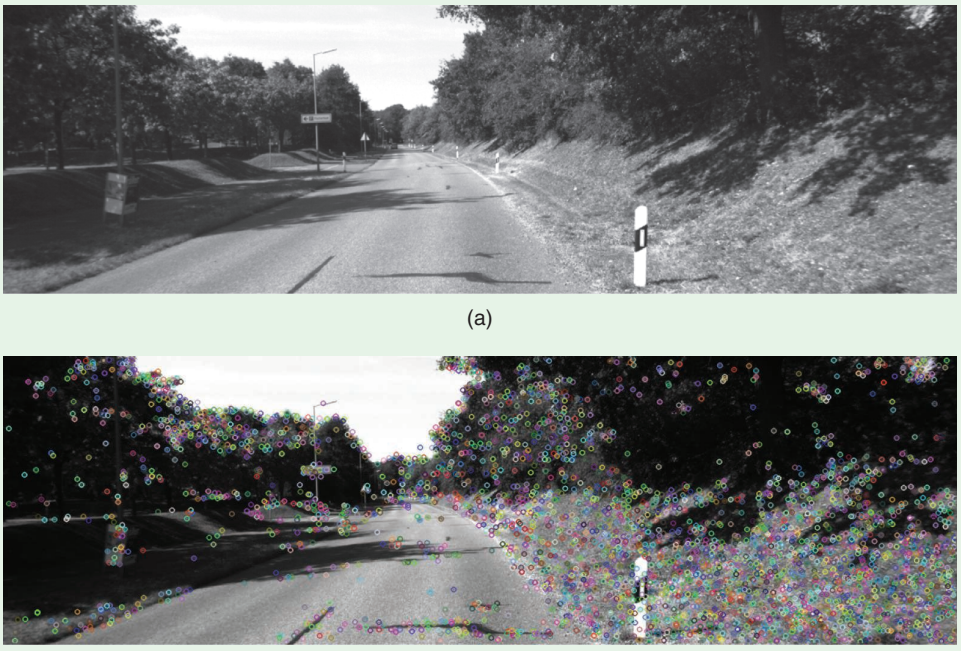
SSL for dynamic obstacles pixel-wise segmentation

A pixel-level binary segmentation of dynamic obstacles is proposed in [21], using temporal image sequences captured with a monocular camera installed on a mobile urban vehicle. The approach firstly separates sparse dynamic keypoints features from the static ones, by applying a RANSAC technique over the optical flow between consecutive frames. Then, the automatically produced per-pixel dynamic labels are transferred as input of a learning Gaussian process (GP) model. Next, the trained model extrapolates this knowledge to label as dynamic the pixels of the same visual properties instead of the ones previously automatically identified as dynamic. The whole process is achieved during an online procedure. The system is evaluated on a hand-labeled data set. This SSL strategy has the advantage to provide the background subtraction from a moving camera, while extrapolating a dense per-pixel segmentation of the dynamic obstacles from sparse detected keypoints. However, this technique cannot provide per-obstacles analysis as it merely predicts a binary mask of pixels corresponding to dynamic obstacles.

It is possible to increase the maximum speed velocity of the self-driving car while preserving smooth trajectories.

The technique in [18] extends the previous approach for SSL multi-instance segmentation by using temporal image sequences captured with a monocular camera installed on a mobile urban vehicle. The authors apply, over the mobile keypoints detected by [21], a clustering method using the tracked keypoints information such as their spatial location and motion pattern features. The multi-instance segmentation of dynamic obstacles is evaluated on a hand-labeled video sequence of the KITTI data set [20].

Overall, the authors state that some issues shared with analytical methods persist in their approach. If the dynamic obstacles shadows are projected on the background, then the latter are considered as dynamic as well. Moreover, the segmentation of distant dynamic obstacles can be missed if the corresponding keypoints variations are considered as noise due to the difficulty to detect the corresponding slight optical flow variations. Furthermore, if a dynamic obstacle, either large or close to the sensor, represents the majority of the image keypoints, then this given obstacle is likely to be treated as the static background scene.



(a)

(b)

FIGURE 2. Salient features location in an urban ego–vehicle environment. (a) An arbitrary frame, extracted from the KITTI data set [20], illustrating an urban, asphalt road with the surrounding environment. (b) Keypoints detected on the left input image using the SIFT detector. The distribution is dense in the off-road region and sparse on the asphalt road in the image center.

Nonetheless, it is important to bear in mind that these approaches present state-of-the-art competitive performances for dynamic obstacles detection and segmentation without training or pretraining on annotated data. In addition, the method in [18] provides interesting tools to analyze on the move the dynamic obstacles, for example to separately track them and learn to predict their intention.

The next focus is on SSL techniques designed for object tracking and temporal predictions in urban road scene evolution, including dynamic obstacles.

Temporal tracking predictions

To deal with object appearance changes, a competitive SSL tracking technique [55] proposes an online adaptive strategy combining tracking, learning, and object detector real-time modules. However, in the context of autonomous driving, it may often be necessary to simultaneously track, and even anticipate, the trajectories of several surrounding road users. Moreover, being able to consider the interactions between each road user requires some complex motion pattern analysis.

In the context of autonomous driving, it may often be necessary to simultaneously track, and even anticipate, the trajectories of several surrounding road users.

It turns out that some SSL approaches propose to deal with this challenge by focusing the prediction effort on the entire scene in a unified way, rather than on every obstacle independently. The SSL deep tracking system [23] learns to predict the future state of a 2D lidar occupancy grid. This is achieved by training an RNN on the latent space of a CNN autoencoder, which is applied on the input occupancy grid considered as an image. Such an approach could be categorized as unsupervised. However, in this article we make the choice to consider that exploiting an additional future temporal information during the training, not available during the prediction step, is a type of self-supervision.

Each cell of the grid is represented by a pixel, which can be color-coded as occluded, void, or obstacle surface. Consequently, the model can be trained from end to end by learning to predict the next occupancy grid states using the past and current conditions. Solely the prediction output error of nonoccluded cells is backpropagated during the training. By definition, this system can perform a self-evaluation by computing a per-pixel photometric error between the predicted occupancy grid and the real future observed occupancy grid at the same temporal instant. This technique has the advantage of being compatible with complex motion patterns compared to Bayesian and Kalman tracking techniques. In addition, the training process enables predicting the obstacle trajectories, even during occlusions. The major interest of deep tracking is that, as the model learns to predict a complete scene, it naturally considers interactions between each dynamic obstacle present in the scene. In [17], the deep tracking model is extended for a real mobile lidar sensor by adding a spatial transformer module to take into consideration the displacements of the ego–vehicle with respect to its environment during object tracking.

In turn, these tracking approaches provide the tools to collect motion pattern information of surrounding dynamic obstacles, such that this information may help to classify obstacles depending on their dynamic properties [56].

Low-level sensor data analysis

This section addresses the sensor data analysis for low-level information estimation in the context of autonomous driving. Compared to the previous methods, the attention has mainly focused recently on SSL depth map estimation from monocular or stereo cameras.

SSL depth map estimation

The self-supervised depth map estimation approach presented in [16] predicts a depth map from a monocular camera without relying on annotated depth maps. The pose transformation between both left and right cameras is known. The SSL strategy is as follows. First, the left camera frame is provided as input to a CNN model trained from scratch to predict the corresponding depth map. Second, an inverse warping is performed by combining the predicted left depth map with the right camera frame to output a synthesized frame similar to the input left frame. In this way, an SSL photometric reconstruction error can be computed as output of the decoder part. Next, this per-pixel error is directly used to train the encoder weights using the stochastic gradient descent optimization technique. While requiring neither pretraining nor annotated ground-truth depths, this approach presents prediction performances comparable with the state-of-the-art fully supervised monocular techniques. However, the ground-truth pose transformation, related to the interview displacement between both cameras, is required.

Following a similar idea, another technique is proposed in [15]. It is trained to reconstruct, from a given frame, the second frame taken from a different point of view. It generates a depth map using a stereo camera during the training step, and also during the prediction step. This makes the approach more robust, such that it becomes competitive with standard stereo-matching techniques. Moreover, the constraint of preserving two cameras and the pose transformation ground truth for predictions enables its counterpart to perform online learning. This may be interesting for dealing with unseen novel ego–vehicle environments during the training.

To overcome the necessity of the pose transformation ground truth, Zhou et al. [57] propose to predict, from a temporal sequence of frames, the depth map with a learning model and the successive camera pose transformations with another learning model. Both models are trained together from end to end to make the novel view synthesis of the next frame. However, the pose transformation estimation implies that the predicted depth map is defined up to a scale factor.

A more modular technique [49] exploits either temporal monocular sequences of frames as in [57], the paired frames

of a stereo camera as in [15], or jointly exploited temporal and stereo information. This framework also deals with the false depth estimation of moving obstacles by ignoring the pixels not varying between two consecutive temporal frames during training. It also deals with occluded pixels when the captured point of view changes by using a minimum reprojection loss.

To summarize, low-level analysis techniques for depth map estimation have demonstrated that SSL strategies without using ground-truth labels can bring state-of-the-art solutions competitive with fully supervised techniques. Overall, the SSL techniques presented in this section support the following conclusion: By exploiting the complementarity between analytical and learning methods, it is possible to address several autonomous driving perception tasks, without necessarily requiring an annotated data set. Presented methodologies are summarized in Figure 3 along with Table 2.

By exploiting the complementarity between analytical and learning methods, it is possible to address several autonomous driving perception tasks, without necessarily requiring an annotated data set.

Limitations and future challenges

In the context of autonomous driving, some limitations remain in the presented SSL perception systems and open future research perspectives.

Catastrophic forgetting

During the online learning procedure, the trainable weights of the model may require unnecessary repetitive updates for detecting a given pattern throughout the environment exploration. In fact, when a learning model is continuously specialized for dealing with the latest data, the likelihood increases that the model simultaneously forgets the potentially relevant formerly learned patterns. It turns out that it is possible to deal with this catastrophic forgetting issue when using neural networks [61]. For future research directions, it may be interesting to combine such incremental learning techniques with the presented SSL frameworks.

The following issues concern the scene depth map estimation solely based on temporal analysis.

- The presence of dynamic obstacles in the scene during the learning stage can result in poor estimates of the observed scene. As discussed in [21], further research on SSL for potentially dynamic obstacle delineations on the sensor data may help to deal with this issue.
- The current state-of-the-art techniques cannot estimate the real depth map without requiring a supervised scaling factor. The latter is generally obtained by estimating the real

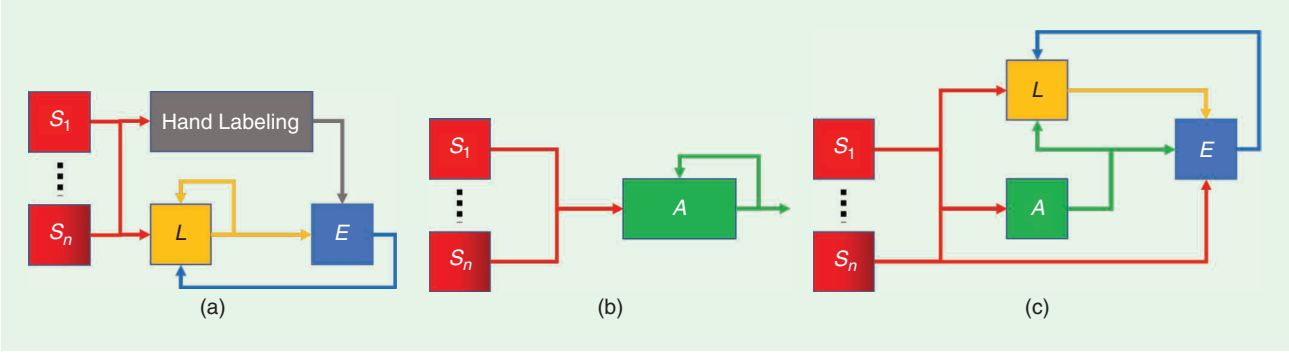


FIGURE 3. Function diagrams showing the common connections depending on the strategy. (a), (b), and (c), respectively, refer to learning, handcrafted, and self-supervised learning strategies. Functional blocks represent a single monocular camera frame S_1, additional sensor data (e.g., temporal frame sequence, stereo camera, or lidar data) S_n, a learning model L, an analytical method A, and evaluation method E.

Table 2. The functional block connections of presented SSL methodologies depending on the application. Experimental data sets are exploited and relative prediction performances are reported whenever available.

SSL Methodologies	$S_1 \rightarrow L$	$S_n \rightarrow L$	$S_1 \rightarrow A$	$S_n \rightarrow A$	$S_n \rightarrow E$	$A \rightarrow L$	$L \rightarrow E$	$A \rightarrow E$	$E \rightarrow L$	Data Sets	Performances
(Off)road segmentation [13], [19], [51], [53], [54]	√		√	√		√	√	√	√	–	–
Dynamic obstacles analysis [18], [21]	√		√	√		√				KITTI [20] Sidney [21]	–
Temporal tracking predictions [17], [23]	√	√			√		√		√	Oxford Robotcar data set [55]	–
Depth map estimation [15][1], [16], [49][1], [57]	√	√[1]			√		√		√	KITTI Make3D [60]	[59]* > [49] > [16] > [57] > [61]*

*: supervised methods. The superscript 1 specifies that the methods in [15] and [49] exclusively use the connection $Sn \rightarrow L$.

metric values of the pose transformation between two consecutive camera viewpoints. As proposed in the Deep MANTA supervised detector [62], it may be interesting to recover this scale factor by using some template-matching techniques on the observable objects of the scene.

Concerning the online self-evaluation, some of the presented systems require an analytically obtained baseline reference [19]. However, if we consider that the analytical processes, considered as ground-truth labeling techniques, are likely to generate some noisy labels, it may be interesting to investigate some future research on how to evaluate this prior noise from the learning model point of view [11], and how to deal with it [9].

Authors

Florent Chiaroni (f.chiaroni@net.estia.fr) received his Dipl-Ing degree in computer science and electronics from Ecole Superieure des Technologies Industrielles Avancées, Bidart, France, in 2016 and his M.Sc. degree in robotics and embedded systems from the University of Salford Manchester, United Kingdom, in 2016. He received his Ph.D. degree in signal and image processing from the University of Paris-Saclay in 2020, with VEDECOM Institute, Versailles, France, and Université Paris-Saclay, Centre national de la recherche scientifique, CentraleSupélec, Laboratoire des signaux et systèmes, Gif-Sur-Yvette, France. His current research focuses on weakly supervised learning for autonomous vehicle perception.

Mohamed-Cherif Rahal (mohamed.rahal@vedecom.fr) received his state computer science engineering diploma from the Computer Science Institute, Constantine-Algeria, in 2002 and his Ph.D. degree in theoretical computer science from the Université Paris Dauphine-PSL, France, in 2010. Since 2013, he has been the head of the transversal and multidisciplinary domain team at VEDECOM, Versailles, France. He participated in and led six European and seven national projects dealing with connected and autonomous vehicles. He is also the author or coauthor of more than 50 publications and he cosupervised five Ph.D. theses. His research interests include concern perception, artificial intelligence, and data fusion for autonomous driving.

Nicolas Hueber (nicolas.hueber@isl.eu) received his Ph.D. degree in optoelectronics and systems from the University of Haute Alsace (UHA), Mulhouse, France, in 2007, when he joined the French-German Research Institute of Saint-Louis (ISL), France, as a researcher. Since June 2009, he has been with the ISL-European Laboratory for Sensory Intelligence research team, France, which is focused on edge computing and embedded artificial intelligence for automatic object detection, recognition, and tracking. He is also a UHA lecturer in the domain of computer vision. He received a best paper award at the Machine Intelligence and Bio-Inspired Computation IX Conference in 2015 and the 42nd Ingénieur Général Chanson Innovation Prize in 2016.

Frédéric Dufaux (frederic.dufaux@l2s.centralesupelec.fr) received his M.Sc. degree in physics and his Ph.D. degree in electrical engineering from Ecole Polytechnique Fédérale de Lausanne, Switzerland, in 1990 and 1994, respectively. He is a CNRS research director at the Laboratory for Signals and Systems, Gif-sur-Yvette, France, where he heads the Telecom and Networks Division. He currently serves as the chair of the Multimedia Signal Processing Technical Committee. He was vice general chair of the 2014 IEEE International Conference on Image Processing, general chair of the 2018 IEEE International Workshop on Multimedia Signal Processing, chair of the 2018 IEEE Multimedia Signal Processing technical committee, and technical program cochair of the 2019 IEEE International Conference on Image Processing. He was the editor-in-chief of *Signal Processing: Image Communication* from 2010 to 2019. He is the chair of the European Association for Signal Processing Technical Area Committee on visual information processing. He currently serves as the chair of the Multimedia Signal Processing Technical Committee. He is a Fellow of IEEE.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004. doi: 10.1023/B:VISI.0000029664.99615.94.

[2] M. Karpushin, G. Valenzise, and F. Dufaux, "Keypoint detection in RGBD images based on an anisotropic scale space," *IEEE Trans. Multimedia*, vol. 18, no. 9, pp. 1762–1771, 2016. doi: 10.1109/TMM.2016.2590305.

[3] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 2, no. 3, pp. 194–220, 2017. doi: 10.1109/TIV.2017.2749181.

[4] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollr, Panoptic segmentation. 2018. [Online]. Available: arXiv:1801.00868

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth et al., "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223. doi: 10.1109/CVPR.2016.350.

[6] G. Niu, M. C. du Plessis, T. Sakai, Y. Ma, and M. Sugiyama, "Theoretical comparisons of positive-unlabeled learning against positive-negative learning," in *Proc. Advances Neural Information Processing Systems*, 2016, pp. 1199–1207. doi: 10.5555/3157096.3157231.

[7] F. Chiaroni, M.-C. Rahal, N. Hueber, and F. Dufaux, "Learning with a generative adversarial network from a positive unlabeled dataset for image classification," in *Proc. IEEE Int. Conf. Image Processing*, 2018, pp. 1368–1372. doi: 10.1109/ICIP.2018.8451831.

[8] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. M. Erfani, S.-T. Xia, S. Wijewickrema, and J. Bailey, Dimensionality-driven learning with noisy labels. 2018. [Online]. Available: arXiv:1806.02612

[9] F. Chiaroni, M. C. Rahal, N. Hueber, and F. Dufaux, "Hallucinating a cleanly labeled augmented dataset from a noisy labeled dataset using GAN," in *Proc. IEEE Int. Conf. Image Processing*, 2019, pp. 3616–3620. doi: 10.1109/ICIP.2019.8803632.

[10] M. E. Houle, "Local intrinsic dimensionality I: An extreme-value-theoretic foundation for similarity applications," in *Proc. Int. Conf. Similarity Search and Applications*. Springer, 2017, pp. 64–79. doi: 10.1007/978-3-319-68474-1_5.

[11] S. Jain, M. White, and P. Radivojac, "Estimating the class prior and posterior from noisy positives and unlabeled data," in *Advances Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 2693–2701.

[12] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, Dept. Eng., Univ. Cambridge, 2016.

[13] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski, "Self-supervised monocular road detection in desert terrain," in *Proc. Robotics: Science and Systems*, Philadelphia, 2006, vol. 38.

[14] L. Jing and Y. Tian, Self-supervised visual feature learning with deep neural networks: A survey. 2019. [Online]. Available: arXiv:1902.06162

[15] Y. Zhong, Y. Dai, and H. Li, Self-supervised learning for stereo matching with self-improving ability. 2017. [Online]. Available: arXiv:1709.00930

[16] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised CNN for single view depth estimation: Geometry to the rescue," in *Proc. European Conf. Computer Vision*. Springer, 2016, pp. 740–756. doi: 10.1007/978-3-319-46484-8_45.

[17] J. Dequaire, P. Ondruska, D. Rao, D. Wang, and I. Posner, "Deep tracking in the wild: End-to-end tracking using recurrent neural networks," *Int. J. Robot. Res.*, vol. 37, nos. 4–5, pp. 492–512, 2017. doi: 10.1177/0278364917710543.

[18] A. Bewley, V. Guizilini, F. Ramos, and B. Upcroft, "Online self-supervised multi-instance segmentation of dynamic objects," in *Proc. 2014 IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 1296–1303. doi: 10.1109/ICRA.2014.6907020. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/6907020/

[19] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, "Learning long-range vision for autonomous off-road driving," *J. Field Robot.*, vol. 26, no. 2, pp. 120–144, 2009. doi: 10.1002/rob.20276.

[20] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The Kitti vision benchmark suite," in *Proc. 2012 IEEE Conf. Computer Vision and Pattern Recognition*. pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

[21] V. Guizilini and F. Ramos, "Online self-supervised segmentation of dynamic objects," in *Proc. 2013 IEEE Int. Conf. Robotics and Automation (ICRA)*, 2013, pp. 4720–4727. doi: 10.1109/ICRA.2013.6631249.

[22] D. Pathak, R. Girshick, P. Dollr, T. Darrell, and B. Hariharan, "Learning features by watching objects move," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 6024–6033. doi: 10.1109/CVPR.2017.638.

[23] P. Ondruska and I. Posner, Deep tracking: Seeing beyond seeing using recurrent neural networks. 2016. [Online]. Available: arXiv:1602.00991

[24] C. G. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vision Conf.*, 1988, pp. 147–151.

[25] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proc. European Conf. Computer Vision*. Springer, 2006, pp. 404–417. doi: 10.1007/11744023_32.

[26] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "ORB: An efficient alternative to sift or surf," in *Proc. ICCV*, 2011, pp. 2564–2571. doi: 10.1109/ICCV.2011.6126544.

[27] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3D object recognition," in *2009 IEEE 12th Int. Conf. Computer Vision Workshops, ICCV Workshops*. pp. 689–696. doi: 10.1109/ICCVW.2009.5457637.

[28] OpenCV. Accessed on: Apr. 7, 2020. [Online]. Available: https://opencv.org/

[29] PCL. Accessed on: Apr. 7, 2020. [Online]. Available: http://pointclouds.org//

[30] G. Farnebck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis (Lecture Notes in Computer Science)*, J. Bigun and T. Gustavsson, Eds. Berlin, Heidelberg, Germany: Springer Berlin Heidelberg, 2003, pp. 363–370.

[31] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artificial Intelligence*, 1981. pp. 674–679. doi: 10.5555/1623264.1623280.

[32] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015, pp. 3061–3070. doi: 10.1109/CVPR.2015.7298925.

[33] K. Prazdny, "Egomotion and relative depth map from optical flow," *Biol. Cybern.*, vol. 36, no. 2, pp. 87–102, 1980. doi: 10.1007/BF00361077.

[34] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981. doi: 10.1145/358669.358692.

[35] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recogn.*, vol. 13, no. 2, pp. 111–122, 1981. doi: 10.1016/0031-3203(81)90009-1.

[36] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, 2011. doi: 10.1109/MRA.2011.943233.

[37] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 6, pp. 1052–1067, 2007. doi: 10.1109/TPAMI.2007.1049.

[38] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2013, pp. 1352–1359. doi: 10.1109/CVPR.2013.178.

[39] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, MultiNet: Real-time joint semantic reasoning for autonomous driving. 2016. arXiv:1612.07695

[40] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Advances Neural Information Processing Systems*, 2015, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks

[41] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition*, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[42] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Appl. Soft Comput.*, vol. 70, pp. 41–65, 2018. doi: 10.1016/j.asoc.2018.05.018.

[43] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Oct. 2017. pp. 2980–2988. doi: 10.1109/ICCV.2017.322.

[44] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers et al., "Flownet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Dec. 2015. pp. 2758–2766. doi: 10.1109/ICCV.2015.316.

[45] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2024–2039, Oct. 2016. doi: 10.1109/TPAMI.2015.2505283.

[46] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Proc. 31st AAAI Conf. Artificial Intelligence*, 2017.

[47] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proc. European Conf. Computer Vision (ECCV)*, 2018, pp. 132–149. doi: 10.1007/978-3-030-01264-9_9.

[48] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Process. Mag.*, vol. 13, no. 6, pp. 47–60, 1996. doi: 10.1109/79.543975.

[49] C. Godard, O. M. Aodha, and G. Brostow, Digging into self-supervised monocular depth estimation. 2018. arXiv:1806.01260

[50] A. Kendall, V. Badrinarayanan, and R. Cipolla, Bayesian SegNet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. 2015. arXiv:1511.02680

[51] D. Lieb, A. Lookingbill, and S. Thrun, "Adaptive road following using self-supervised learning and reverse optical flow," *Robotics: Sci. Syst.*, pp. 273–280, 2005. doi: 10.15607/RSS.2005.I.036.

[52] "The grand challenge," DARPA, Arlington, VA. Accessed on: Apr. 7, 2020. [Online]. Available: https://www.darpa.mil/about-us/timeline/-grand-challenge-for-autonomous-vehicles

[53] S. Zhou, J. Gong, G. Xiong, H. Chen, and K. Iagnemma, "Road detection using support vector machine based on online learning and evaluation," in *Proc. 2010 IEEE Intelligent Vehicles Symp.*, pp. 256–261. doi: 10.1109/IVS.2010.5548086.

[54] H. Roncancio, M. Becker, A. Broggi, and S. Cattani, "Traversability analysis using terrain mapping and online-trained terrain type classifier," in *Proc. 2014 IEEE Intelligent Vehicles Symp.*, 2014, pp. 1239–1244. doi: 10.1109/IVS.2014.6856427.

[55] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, July 2012. doi: 10.1109/TPAMI.2011.239.

[56] M. Fathollahi and R. Kasturi, Autonomous driving challenge: To Infer the property of a dynamic object based on its motion pattern using recurrent neural network. 2016. arXiv:1609.00361

[57] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858. doi: 10.1109/CVPR.2017.700.

[58] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *Int. J. Robot. Res. (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017. doi: 10.1177/0278364916679498. [Online]. Available: http://dx.doi.org/10.1177/0278364916679498

[59] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2018, pp. 2002–2011. doi: 10.1109/CVPR.2018.00214.

[60] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D scene structure from a single still image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 824–840, 2008. doi: 10.1109/TPAMI.2008.132.

[61] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. Advances Neural Information Processing Systems*, 2014, pp. 2366–2374.

[62] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan et al., "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci.*, vol. 114, no. 13, pp. 3521–3526, 2017. doi: 10.1073/pnas.1611835114.

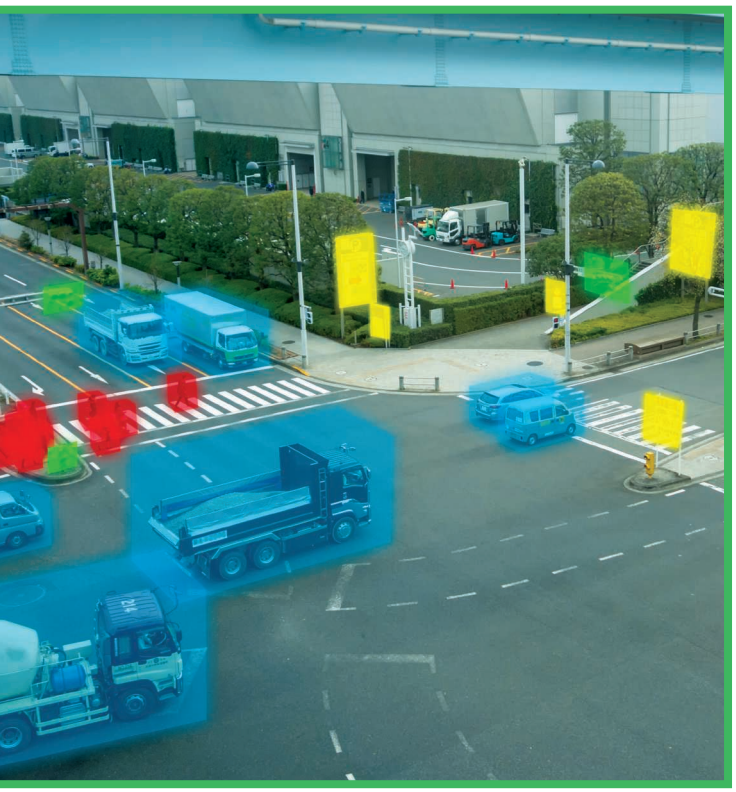
[63] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau, "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 1827–1836. doi: 10.1109/CVPR.2017.198.

SP

Andreas Bär, Jonas Löhdefink, Nikhil Kapoor,
Serin John Varghese, Fabian Hüger,
Peter Schlicht, and Tim Fingscheidt

The Vulnerability of Semantic Segmentation Networks to Adversarial Attacks in Autonomous Driving

Enhancing extensive environment sensing



©SHUTTERSTOCK.COM/MONOPOLY919

Digital Object Identifier 10.1109/MSP.2020.2983666
Date of current version: 24 December 2020

Enabling autonomous driving (AD) can be considered one of the biggest challenges in today's technology. AD is a complex task accomplished by several functionalities, with environment perception being one of its core functions. Environment perception is usually performed by combining the semantic information captured by several sensors, i.e., lidar or camera. The semantic information from the respective sensor can be extracted by using convolutional neural networks (CNNs) for dense prediction. In the past, CNNs constantly showed state-of-the-art performance on several vision-related tasks, such as semantic segmentation of traffic scenes using nothing but the red-green-blue (RGB) images provided by a camera. Although CNNs obtain state-of-the-art performance on clean images, almost imperceptible changes to the input, referred to as *adversarial perturbations*, may lead to fatal deception. The goal of this article is to illuminate the vulnerability aspects of CNNs used for semantic segmentation with respect to adversarial attacks, and share insights into some of the existing known adversarial defense strategies. We aim to clarify the advantages and disadvantages associated with applying CNNs for environment perception in AD to serve as a motivation for future research in this field.

Introduction

The desire for mobility is a driving force in progressing technology, with AD clearly being the next major step in automotive technology, along with electromobility. An AD vehicle is a highly complex system with several sensors and subcomponents, one of which is vehicle-to-everything (V2X) communication.

In the context of AD, V2X communication has several applications, e.g., path planning and decision making [29], or systems used for localization and cooperative perception [14]. All autonomous systems need a perception stage, which constitutes the first step in the process chain of sensing the environment. The purpose of cooperative perception systems in AD is the exploitation of information stemming from other traffic participants to increase safety, efficiency, and comfort aspects while driving [13]. The common concept lies in information

 1053-5888/21©2021IEEE

transmission between various vehicles as well as between vehicles and back-end servers over any kind of (wireless) transmission channel. The transmitted information ranges from trajectories of the ego vehicle and other traffic participants over vehicle state information to sensor data coming from radar, lidar, and the camera, and assists with constructing a more complete model of the physical world.

Each decision of an AD vehicle is based on the underlying environment perception and is intended to lead to an appropriate action. Hence, the proper perception of the environment is an essential ingredient for reducing road accidents to a bare minimum to foster public acceptance of AD. The most common sensors of a single AD vehicle's environment perception system [5], [16], [26] are shown in Figure 1.

Several external sensors, i.e., radar, lidar, and the camera, are mounted on an AD vehicle. Radar sensors are already widely used in multiple automotive functions and play a key role in enabling AD [10], [24]. Lidar sensors are capable of detecting obstacles [16] and have been used in numerous AD competitions [5]. Camera sensors, on the other hand, are mainly used for detecting lane markings or traffic signs [26] but can also be used for object detection and semantic segmentation [30]. The data captured by the three sensor groups are gathered within a central processing unit to extract semantic information from the environment.

Over the past few years, the interest in employing deep neural networks (DNNs) increased noticeably as they constantly achieved state-of-the-art performance in multiple vision-related tasks and benchmarks, including semantic segmentation for AD [8], [17]. Semantic segmentation is a classical computer vision task, where each pixel of an RGB image is assigned to a corresponding semantic class [see Figure 2(a) and (b)]. Because such camera-based technology is both cheaper and uses less data compared to lidar-based technology, it is of special interest for AD. Recent progress in semantic segmentation enables real-time processing [30], making this an even more promising technology for AD applications.

Nevertheless, the environment perception system of an AD vehicle is a highly safety-relevant function. Any error can lead to catastrophic outcomes in the real world. Although DNNs reveal promising functional performance in a wide variety of tasks, they show vulnerability to certain input patterns, denoted as *adversarial examples* [25]. Adversarial examples are almost imperceptibly altered versions of an image and are

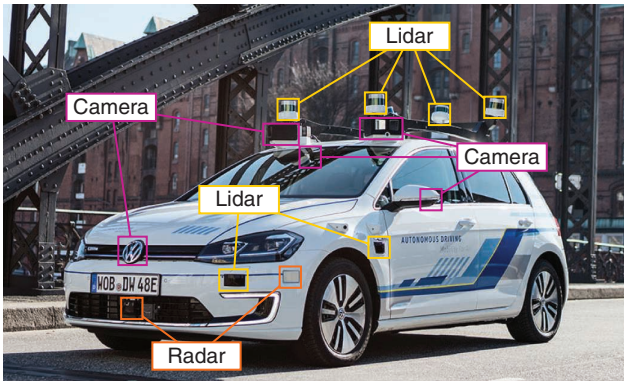


FIGURE 1. An AD research vehicle equipped with radar (orange), lidar (yellow), and camera sensors (purple). The sensors are placed at different locations to obtain an extensive environment sensing.

able to fool state-of-the-art DNNs in a highly robust manner [see Figure 2(c) and (d)]. Assion et al. [2] showed that a virtually unlimited set of adversarial examples can be created on each state-of-the-art machine learning model. This intriguing property of DNNs is of special concern when looking at their applications in AD and needs to be addressed further by DNN certification methods [9], [27] or means of uncertainty quantification [19]. Cooperative perception, for example, can be seen as one of the weak spots in data processing during the environment perception of an AD vehicle; it can be used as a loophole to intrude adversarial examples to fool AD vehicles in range. Note that this is only one of many possible scenarios that show how adversarial examples can find their way into the system.

In this article, we examine the vulnerability of DNNs toward adversarial attacks, while focusing on environment perception for AD. For this purpose, we chose semantic segmentation as the underlying function that we want to perform adversarial attacks on because it is a promising technology for camera-based environment perception. The article is intended to sensibilize the reader toward vulnerability issues of DNNs in environment perception for AD and to stir interest in the development of new defense strategies for adversarial attacks.

Semantic segmentation

An RGB image is a high-dimensional source of data, with pixels being the smallest units of semantic information. Semantic segmentation is a popular method used to extract the semantic information from an RGB image, where each pixel is tagged

FIGURE 2. A simple adversarial attack using the iterative least-likely class method [15] to fool the image cascade network (ICNet) [30] on a hand-picked image from the Cityscapes validation set. (a) A clean input image, (b) a semantic segmentation of clean input image, (c) an adversarial example, and (d) a semantic segmentation of an adversarial example.

with a label taken from a finite set of classes. Today's state of the art in semantic segmentation is dominated by CNNs, a special form of DNNs. This section introduces some mathematical notation regarding CNNs and gives an overview of the CNN architecture used for semantic segmentation throughout this article.

A mathematical notation

For the sake of simplicity, we first assume having a CNN, which takes one input image and outputs only a corresponding class for the entire image. Hence, we begin with simple image classification and then extend to semantic segmentation.

First of all, the input image is denoted as $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{G}^{H \times W \times C}$, with image height in pixels H, image width in pixels W, number of color channels C, data set $\mathcal{X}$, and the set of integer gray values $\mathbb{G}$. Each image contains gray values $\boldsymbol{x}_i \in \mathbb{G}^C$ at each pixel position $i \in \mathcal{I}$, with $\mathcal{I}$ being the set of pixel positions, having the cardinality $|\mathcal{I}| = H \cdot W$. The smaller patches of an image are denoted as $\boldsymbol{x}_{\mathcal{I}_i} \in \mathbb{G}^{h \times w \times C}$, with patch height in pixels h, crop width in pixels w, and the set of pixel positions $\mathcal{I}_i \subseteq \mathcal{I}$ with i being the center pixel and $|\mathcal{I}_i| = h \cdot w$. For the special case of $\mathcal{I}_i = \mathcal{I}$, we obtain $\boldsymbol{x}_{\mathcal{I}_i} = \boldsymbol{x}$. A CNN usually consists of several layers $\ell \in \mathcal{L}$ containing feature map activations $\boldsymbol{f}_\ell(\boldsymbol{x}) \in \mathbb{R}^{H_\ell \times W_\ell \times C_\ell}$ of the respective layer $\ell \in \mathcal{L}$, and first-layer input image $\boldsymbol{x}$, with the set of layers $\mathcal{L}$, feature map height H_ℓ, feature map width W_ℓ, and number of feature maps C_ℓ. Fed using the input image $\boldsymbol{x}$, a CNN for image classification outputs a probability score $P(s|\boldsymbol{x}) \in \mathbb{I}$ for each class $s \in \mathcal{S}$, with $\mathbb{I} = [0, 1]$, and the set of classes $\mathcal{S}$, with the number of classes $N = |\mathcal{S}|$, leading to

$$\mathfrak{F}_{\text{classification}} : \mathbb{G}^{H \times W \times C} \to \mathbb{I}^N. \tag{1}$$

For better readability, the CNN parameters $\boldsymbol{\theta}$ are omitted in our notation. The predicted class $s^*(\boldsymbol{x}) \in \mathcal{S}$ for the input image $\boldsymbol{x}$ is then obtained by

$$s^*(\boldsymbol{x}) = \operatorname*{argmax}_{s \in \mathcal{S}} P(s|\boldsymbol{x}). \tag{2}$$

From now on, a CNN is considered, which is capable of performing semantic segmentation. The respective CNN outputs a probability $P(s|i, \boldsymbol{x})$ for each pixel position $i \in \mathcal{I}$ of the input image $\boldsymbol{x}$ and class $s \in \mathcal{S}$. Altogether, it outputs class scores $\boldsymbol{p}(\boldsymbol{x}) = \mathfrak{F}_{\text{segmentation}}(\boldsymbol{x}) \in \mathbb{I}^{H \times W \times N}$ for all pixel positions $i \in \mathcal{I}$ and classes $s \in \mathcal{S}$, leading to

$$\mathfrak{F}_{\text{segmentation}} : \mathbb{G}^{H \times W \times C} \to \mathbb{I}^{H \times W \times N}. \tag{3}$$

The semantic segmentation mask $\boldsymbol{m}(\boldsymbol{x}) \in \mathcal{S}^{H \times W}$ containing the predicted class $m_i(\boldsymbol{x}) = s_i^*(\boldsymbol{x})$ at each pixel position $i \in \mathcal{I}$ of the input image $\boldsymbol{x}$ is then obtained by

$$\boldsymbol{m}(\boldsymbol{x}) = \operatorname*{argmax}_{s \in \mathcal{S}} \boldsymbol{p}(\boldsymbol{x}). \tag{4}$$

The performance of such a CNN is measured by the mean intersection-over-union (mIoU)

$$\text{mIoU} = \frac{1}{N} \sum_{s \in \mathcal{S}} \frac{\text{TP}(s)}{\text{TP}(s) + \text{FP}(s) + \text{FN}(s)}, \tag{5}$$

with the class-specific true positives $\text{TP}(s)$, false positives $\text{FP}(s)$, and false negatives $\text{FN}(s)$.

Architecture for semantic segmentation

Today's state-of-the-art CNN architectures for semantic segmentation are often based on the work of Long et al. [17]. They proposed to use a CNN, pretrained on image classification, as a feature extractor and further extend it to recover the original image resolution. The extended part is often referred to as the *decoder* and fulfills the task of gathering, reforming, and rescaling the extracted features for the task of semantic segmentation. One characteristic of this proposed network architecture is the absence of fully connected layers. Such CNNs are therefore called *fully convolutional networks* (*FCNs*).

Especially for AD, a real-time capable state-of-the-art CNN being robust to minimal changes in the input is needed. Arnab et al. [1] analyzed the robustness of various CNNs for semantic segmentation toward simple adversarial attacks [11], [15] and concluded that CNNs using the same input with different scales are often most robust. The image cascade network (ICNet) developed by Zhao et al. [30] comprises both, a lightweight CNN architecture with multiscale inputs. The overall structure of the ICNet is depicted in Figure 3. The ICNet is designed to extract multiscale features by taking different scales of the image as inputs. The extracted multiscale features are fused before being upsampled to obtain a full-resolution semantic segmentation mask. The ICNet mainly profits from the combination of high-resolution low-level features (i.e., edges) with low-resolution high-level features (i.e., spatial context).

For the sake of reproducibility, an openly available reimplementation (https://github.com/hellochick/ICNet-tensorflow) of the ICNet based on TensorFlow is used and tested on the widely applied Cityscapes data set [8]. Cityscapes serves as a good data set for exploring CNNs using semantic segmentation for AD, having pixel-wise annotations for 5,000 images (validation, training, and test set combined), with relevant classes such as pedestrians and cars. The reimplementation of the ICNet achieves 67.26% mIoU on the Cityscapes validation set and runs at roughly 19 and 26 frames/s (fps) on our Nvidia Tesla P100 and Nvidia Geforce GTX 1080Ti, respectively, with an input resolution of 1,024 × 2,048. These numbers are promising and indicate that semantic segmentation could serve as a technology for the environment perception system of AD vehicles.

Adversarial attacks

Although CNNs exhibit state-of-the-art performance in several vision-related fields of research, Szegedy et al. [25] revealed their vulnerability toward certain input patterns. The CNN topologies they investigated were fooled by just adding small, imperceptible patterns to the input image. An algorithm producing such adversarial perturbations is called an *adversarial attack*, and a perturbed image is referred to as an *adversarial example*.

Based on the observations of Szegedy et al., new approaches arose for crafting adversarial examples more efficiently [3], [7], [11], [15], [21] and were even extended to dense prediction tasks, e.g., semantic segmentation [2], [18], [22]. In the following section, two types of adversarial attacks are introduced: individual adversarial attacks, which aim to fool on the basis of one particular input image, as well as universal adversarial attacks, which seek to fool on the basis of a whole bunch of images at the same time.

Individual adversarial perturbations

For the sake of simplicity, CNNs used for image classification are considered in this section to describe the basic nature of targeted and nontargeted adversarial attacks using individual adversarial perturbations. As shown previously, image classification can be easily extended to semantic segmentation.

Common adversarial attacks aim at fooling a CNN so that the predicted class $s^*(\boldsymbol{x})$ does not match with the ground-truth class $s(\boldsymbol{x}) \in \mathcal{S}$ of the input image $\boldsymbol{x}$. One example of such an adversarial type of attack is the fast gradient sign method (FGSM) introduced by Goodfellow et al. [11]. FGSM adopts the loss function $J(\boldsymbol{x}, s(\boldsymbol{x}))$ used during the training of the underlying CNN and computes the adversarial examples by

$$\boldsymbol{x}^{\text{adv}} = \boldsymbol{x} + \boldsymbol{r} = \boldsymbol{x} + \lambda \, \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}, s(\boldsymbol{x}))), \tag{6}$$

with the adversarial perturbation $\boldsymbol{r} \in \mathbb{R}^{H \times W \times C}$, the step-size $\lambda \in \mathbb{R}^+$, and the gradient with respect to the input image $\nabla_{\boldsymbol{x}} J(\boldsymbol{x}, s(\boldsymbol{x}))$. Note that $\text{sign}(\cdot) \in \{\pm 1\}^{H \times W \times C}$. FGSM lets the perturbation $\boldsymbol{r}$ effectively increase the loss in each dimension by manipulating the input image into positive ("+") gradient direction. Thus, one is not limited to using the ground truth $s(\boldsymbol{x})$ as depicted in (6), but can, in fact, use the output of the respective DNN $s^*(\boldsymbol{x})$.

Kurakin et al. [15] extended FGSM by using an iterative algorithm, changing the adversarial perturbation slightly in each iteration by a small λ. To prevent the adversarial perturbation's magnitude from getting too large, it is upper-bounded by

$$\| \boldsymbol{r} \|_\infty \leq \epsilon, \tag{7}$$

with $\epsilon \in \mathbb{R}^+$ being the upper bound of the infinity norm and $\epsilon \geq \lambda$. This way, the perceptibility of the adversarial perturbation is controlled by adjusting ϵ accordingly. For the iterative case, (6) extends to

$$\begin{aligned} \boldsymbol{x}_0^{\text{adv}} &= \boldsymbol{x}, \\ \boldsymbol{x}_{\tau+1}^{\text{adv}} &= \boldsymbol{x}_\tau^{\text{adv}} + \boldsymbol{r}_{\tau+1} \\ &= \boldsymbol{x}_\tau^{\text{adv}} + \lambda \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_\tau^{\text{adv}}, s(\boldsymbol{x}))), \end{aligned} \tag{8}$$

with $\tau \in \{0, 1, 2, \ldots\}$ being the current iteration index and therefore $\boldsymbol{x}_\tau^{\text{adv}}$ the adversarial example at iteration τ [the total number of iterations is set by flooring $\lfloor \min(\epsilon + 4, 1.25\epsilon) \rfloor$].

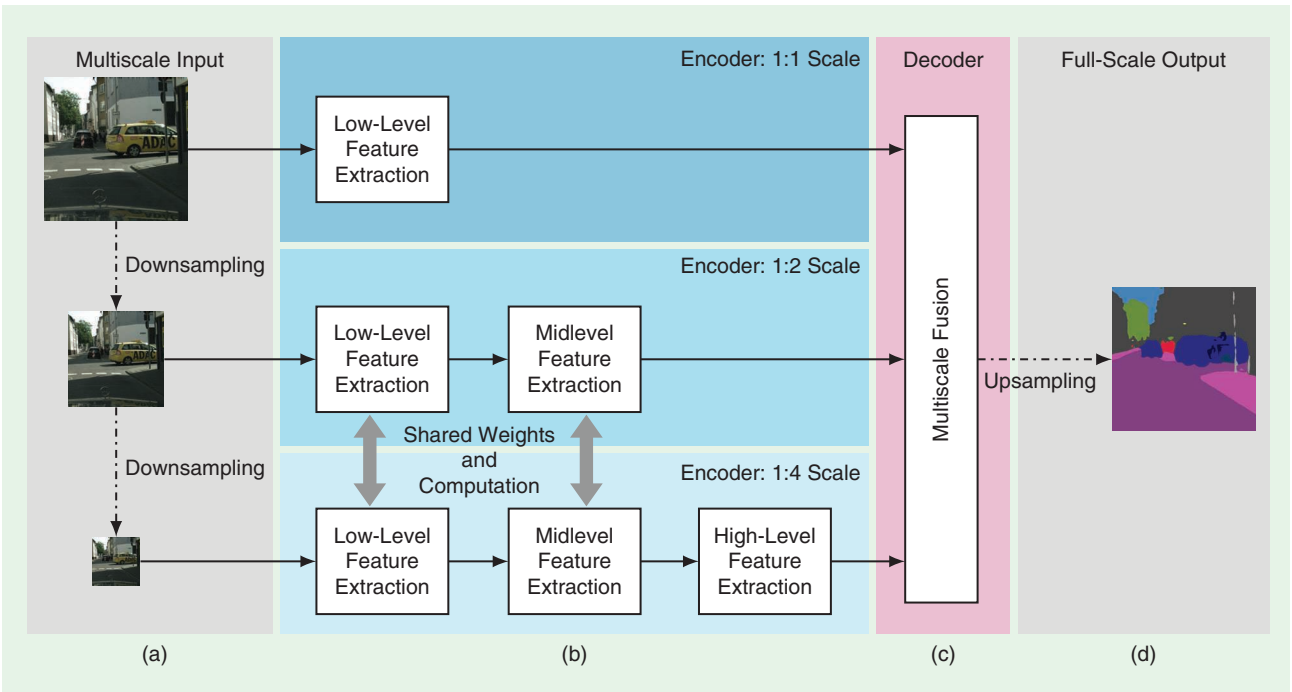


FIGURE 3. An architectural overview of ICNet [30]. The ICNet takes different scales of an RGB image as (a) inputs to output a (d) semantic segmentation mask. The encoder consists of (b) three scale-dependent parts to extract multiscale features from the (b) inputs. Each of these three encoder parts perform a downsampling by a factor of eight during feature extraction. To save computational complexity, the bigger scales are limited to low-level and midlevel feature extraction. The extracted multiscale features are then fused within the decoder by a (c) multiscale fusion block before performing a final upsampling to obtain a full-resolution semantic segmentation mask with respect to the input.

Considering AD vehicles, there exists no ground truth for the data being inferred. As previously noted, a naive attacking idea in this setup would be finding an adversarial perturbation $\boldsymbol{r}$, such that (classification)

$$s^*(\boldsymbol{x}^{\text{adv}}) \neq s^*(\boldsymbol{x}). \tag{9}$$

Such an attack is the least-likely class method (LLCM) introduced by Kurakin et al. [15]. LLCM seeks to find an adversarial perturbation $\boldsymbol{r}$ to obtain

$$s^{\text{o}}(\boldsymbol{x}) = \underset{s\in\mathcal{S}}{\operatorname{argmax}}\, P(s\,|\,\boldsymbol{x}^{\text{adv}}) = \underset{s\in\mathcal{S}}{\operatorname{argmin}}\, P(s\,|\,\boldsymbol{x}), \tag{10}$$

with the least-likely class $s^{\text{o}}(\boldsymbol{x})$ of the input image $\boldsymbol{x}$. Different from before, the adversarial example using LLCM is obtained by taking a step into the negative direction of the gradient with respect to the input image $\boldsymbol{x}$, according to

$$\boldsymbol{x}^{\text{adv}} = \boldsymbol{x} - \lambda \operatorname{sign}(\nabla_x J(\boldsymbol{x}, s^{\text{o}}(\boldsymbol{x}))), \tag{11}$$

minimizing the loss function. Similar to FGSM, LLCM can also be performed in an iterative fashion, where in each step, a small adversarial perturbation is added to the respective input image.

Another well-known approach used for crafting adversarial examples is DeepFool [21], introduced by Moosavi-Dezfooli et al. Compared with FGSM and LLCM, DeepFool not only searches for individual adversarial perturbations but also tries to find the minimal adversarial perturbation with respect to an l_p-norm, changing the network's output. This leads us to the following equation:

$$\boldsymbol{r}_{\min} = \underset{\boldsymbol{r}}{\operatorname{argmin}} \|\boldsymbol{r}\|_p, \text{ s.t. } s^*(\boldsymbol{x}+\boldsymbol{r}_{\min}) \neq s^*(\boldsymbol{x}), \tag{12}$$

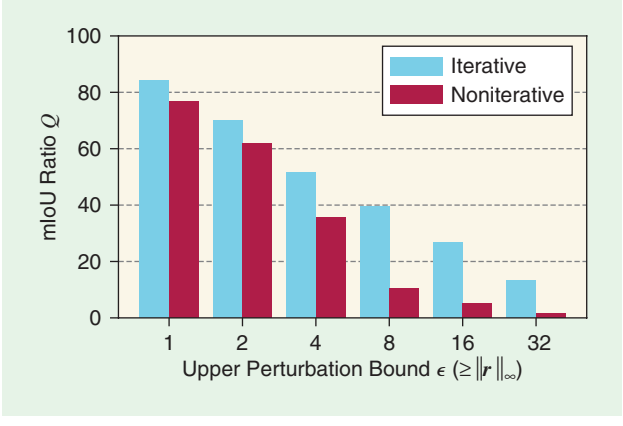


FIGURE 4. Adversarial attacks on the ICNet using the iterative or noniterative LLCM from Kurakin et al. [15] on the Cityscapes validation set with different values for ϵ, an upper bound of the l_∞- norm of the adversarial perturbation $\boldsymbol{r}$. We set $\lambda = \epsilon$ for the noniterative LLCM and $\lambda = 1$ for the iterative LLCM. A lower mIoU ratio Q means a stronger adversarial attack. Note that the noniterative LLCM appears to be even more aggressive than that of the iterative LLCM.

with $\|\cdot\|_p$ being the l_p-norm restricting the magnitude of $\boldsymbol{r}_{\min}$. Moosavi-Dezfooli et al. primarily experimented with $p = 2$, showing DeepFool's superiority in terms of speed and magnitude compared to FGSM when targeting the same error rate for the respective CNN. We do not go into further detail here, but instead refer the interested reader to [21] for more information about DeepFool.

Carlini and Wagner [7] proposed an approach that was shown to be extremely effective with regard to adversarial example-detection mechanisms. They used

$$\begin{aligned}\boldsymbol{r}_{\min} = \underset{\boldsymbol{r}}{\operatorname{argmin}}(\|\boldsymbol{r}\|_2 + c\cdot f(\boldsymbol{x}+\boldsymbol{r})),\\ \text{s.t. } s^*(\boldsymbol{x}+\boldsymbol{r}_{\min}) \neq s^*(\boldsymbol{x}),\end{aligned} \tag{13}$$

as an objective function, with c being a hyperparameter and $f(\boldsymbol{x}+\boldsymbol{r})$ being a loss function. Athalye et al. [3] adopted this approach and, as a result, managed to circumvent several state-of-the-art defense mechanisms. We refer the interested reader to [3] and [7] for more fine-grained information about both approaches and their specific variations.

Thus far, we have introduced adversarial attacks that were successfully carried out on image classification. Arnab et al. [1] conducted the first extensive analysis on the behavior of different CNN architectures for semantic segmentation using FGSM and LLCM, both performed iteratively and noniteratively. They report results on a large variety of CNN architectures, including both lightweight and heavyweight CNN architectures. The main observation was that the network models using residual connections are often more robust when it comes to adversarial attacks. In addition, lightweight CNN architectures tend to be almost equally as robust as do heavyweight CNN architectures. In summary, the results on the Cityscapes data set demonstrated the vulnerability of CNNs in general. We show a typical attack in Figure 2 using the iterative LLCM on the ICNet with the hyperparameters $\lambda = 1$ and $\epsilon = 8$. Despite being mostly imperceptible to the human eye, the adversarial example leads to a dramatically altered network output. To show the overall effect on the Cityscapes validation set, we computed the mIoU ratio for the iterative and noniterative LLCM using different values for ϵ. The mIoU ratio Q is defined by

$$Q = \frac{\text{mIoU}_{\text{adv}}}{\text{mIoU}_{\text{clean}}}, \tag{14}$$

with mIoU_{adv} being the mIoU on adversarially perturbed images $\boldsymbol{x}^{\text{adv}}$, and $\text{mIoU}_{\text{clean}}$ being the mIoU on clean images $\boldsymbol{x}$. The results are plotted in Figure 4. As expected, the stronger the adversarial perturbation (in terms of ϵ) the lower the mIoU on adversarial examples mIoU_{adv} and thus a lower mIoU ratio Q is obtained. As pointed out by Arnab et al. [1], we also observe that the noniterative LLCM is even stronger than its iterative counterpart, which contradicts the original observation made by Kurakin et al. [15] on image classification. Arnab et al. argue that this phenomenon might be a data set property of Cityscapes because the effect does not occur on their second

data set (Pascal VOC 2012). Nonetheless, we do not investigate this further as we prefer to focus on more realistically looking adversarial attacks discussed later in this section.

Metzen et al. [18] introduced new adversarial attacks for semantic segmentation. Instead of only fooling the CNN, they additionally wanted the respective CNN to output more realistically looking semantic segmentation masks. To do so, Metzen et al. developed two methods. The first method uses a fake semantic segmentation mask $\boldsymbol{m}(z)$ instead of the original semantic segmentation mask $\boldsymbol{m}(\boldsymbol{x})$, with $\boldsymbol{x} \neq z \in \mathcal{X}$, meaning that the fake segmentation mask refers to an existing image of data set $\mathcal{X}$. The overall assumption of Metzen et al. is that a possible attacker may invest time to create a few uncorrelated fake semantic segmentation masks him or herself. Assuming that the attacker wants to use the same fake semantic segmentation mask to fool the respective CNN on several images, he is restricted to stationary situations to operate unnoticed, i.e., the AD vehicle does not move and thus the scenery captured by the camera sensor changes only slightly. Because of this operational constraint, we call this method the *stationary segmentation mask method* (*SSMM*). The second method modifies the CNN's original semantic segmentation mask $\boldsymbol{m}(\boldsymbol{x})$ by replacing a predefined objective class $o = m_i(\boldsymbol{x}) \in \mathcal{S}$ at each corresponding pixel position $i \in \mathcal{I}_o(\boldsymbol{x}) \subset \mathcal{I}$ by the spatial nearest-neighbor class $n_i(\boldsymbol{x}) = m_j(\boldsymbol{x}) \in \mathcal{S}$, with $m_j(\boldsymbol{x}) \neq m_i(\boldsymbol{x})$. Here, $\mathcal{I}_o(\boldsymbol{x})$ is the set of all pixel positions, where $m_i(\boldsymbol{x}) = o$ holds. By completely removing the objective class o from the semantic segmentation mask, we obtain

$$m_i^{\mathrm{DNNM}}(\boldsymbol{x}) = \begin{cases} m_i(\boldsymbol{x}), & \text{if } m_i(\boldsymbol{x}) \neq o. \\ n_i(\boldsymbol{x}), & \text{otherwise,} \end{cases} \tag{15}$$

with $m_i^{\mathrm{DNNM}}(\boldsymbol{x})$ being the new target class at pixel position $i \in \mathcal{I}$. Metzen et al. suggested using the Euclidean distance of two pixel positions i and j to find the nearest-neighbor class satisfying $n_i(\boldsymbol{x}) = m_j(\boldsymbol{x}) \neq m_i(\boldsymbol{x}) = o$. In contrast to SSMM, the realistically looking fake semantic segmentation mask created using this method is now unique for each real semantic segmentation output. Additionally, specific properties, such as the correlation between two consecutive real semantic segmentation outputs, are transferred to the created fake ones.

Altogether, a possible attacker is able to create a sequence of correlated realistically looking fake semantic segmentation masks making this kind of attack suitable for situations where the respective AD vehicle moves. Due to these properties, we call this method the *dynamic nearest-neighbor method* (*DNNM*). The application of DNNM has the potential to create safety-relevant perception errors for AD. This can be seen in Figure 5, where DNNM is used to remove pedestrians or cars from the scene. The adversarial examples were created by setting $\epsilon = 10$, followed by the same procedure that was used with iterative LLCM. Astonishingly, the semantic classes different from the objective class are completely preserved and the nearest-neighbor class seems to be a good estimate for the regions occluded by the objective class, thereby dangerously providing a plausible, but wrong, semantic segmentation mask.

Universal adversarial perturbations

Thus far, we have discussed approaches that generate adversarial perturbations for single-input images. In reality, however, it is difficult for a possible attacker to generate adversarial examples for each incoming image of an environment perception system, considering a camera running at 20 fps. Therefore, in AD applications, a special interest lies in single-adversarial perturbations that are capable of fooling a CNN on a set of input images, e.g., a video sequence. This class of adversarial perturbations is called *universal adversarial perturbation* (*UAP*).

One of the first works toward finding UAPs was done by Moosavi-Dezfooli et al. [20]. Their idea was to find a UAP $\boldsymbol{r}_{\mathrm{uni}}$ that fools nearly all the images in some image set $\mathcal{T}$ in an image classification task (again, only one class per image). To achieve this, they used the DeepFool algorithm in an iterative fashion to solve the optimization problem

$$s^*(\boldsymbol{x} + \boldsymbol{r}_{\mathrm{uni}}) \neq s^*(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathcal{T}' \subset \mathcal{T}, \tag{16}$$

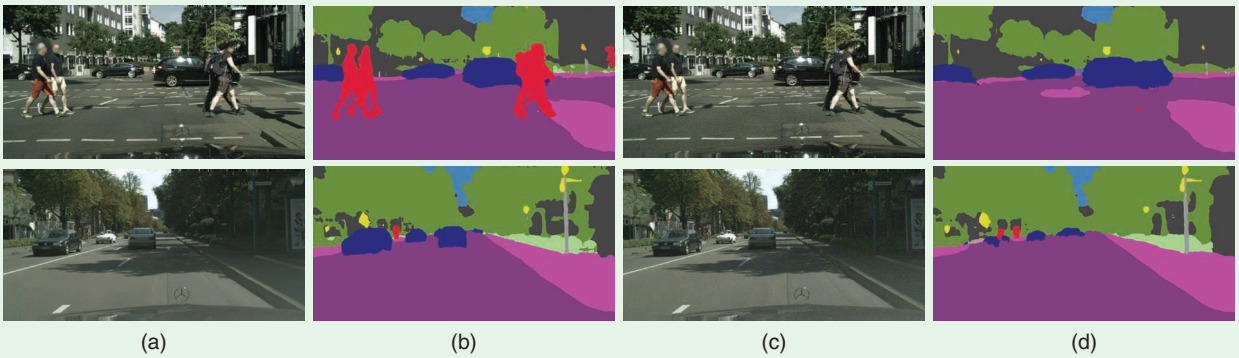


FIGURE 5. Adversarial attacks on the ICNet using a DNNM [18] on two example images, one with pedestrians and one with cars, from the Cityscapes validation set. The adversarial examples aim at removing pedestrians (top row) and cars (bottom row) from the scene. (a) A clean input image, (b) a semantic segmentation output on a clean input image, (c) an adversarial example created by the DNNM, and (d) a semantic segmentation output on an adversarial example created by the DNNM.

with the subset of respective images $\mathcal{T}'$ for which the CNN is fooled, and the set of all the respective images $\mathcal{T}$ the UAP is optimized on. The UAP is again constrained by

$$\|\boldsymbol{r}_{\text{uni}}\|_p \leq \epsilon, \tag{17}$$

with $\|\cdot\|_p$ being the l_p-norm of $\boldsymbol{r}_{\text{uni}}$ and ϵ being its upper bound. In their experiments, Moosavi-Dezfooli et al. obtained the best results by setting $p = \infty$ and $\epsilon = 10$. Different from all of the attacks shown before, the UAP optimized on $\mathcal{T}$ generalizes well, meaning that the UAP can fool even a respective system on a disjointed set of images $\mathcal{V}$, with $\mathcal{V} \cap \mathcal{T} = \varnothing$, on which the UAP was not optimized.

Although Moosavi-Dezfooli et al. used samples from a set of images $\mathcal{T}$ to craft UAPs, Mopuri et al. [23] introduced a data set-independent method named *fast feature fool* (*FFF*). In this section, we consider the formulation of FFF from their extended work [22]. Adopting the overall objective in (16), FFF aims at finding a UAP that increases the mean activation in each layer $\ell \in \mathcal{L}$ without having any knowledge about the respective images $\boldsymbol{x} \in \mathcal{T}'$ to fool. This is done by minimizing the following loss function:

$$J(\boldsymbol{r}_{\text{uni}}) = -\log\left(\prod_{\ell \in \mathcal{L}} \|\boldsymbol{f}_\ell(\boldsymbol{r}_{\text{uni}})\|_2\right), \tag{18}$$

with respect to $\boldsymbol{r}_{\text{uni}}$ (as a first-layer input image), constrained by (17) with $p = \infty$. We used FFF on the ICNet to show the effectiveness and transferability of UAPs on several images taken from the Cityscapes validation set by following Mopuri et al. in choosing $\epsilon = 10$. The obtained results for some images are presented in Figure 6. Although it does not generate the same realistically looking semantic segmentation masks that the DNNM does, FFF still completely fools the ICNet on several diverse images and needs to be computed only once to obtain $\boldsymbol{r}_{\text{uni}}$. Moreover, safety-critical classes such as pedestrians and cars are removed from the scene in all examples, underlining again the risk of adversarial attacks for AD. Note that the particular danger of this method for AD lies in the fact that it just requires a generic adversarial pattern to be added to any unknown sensorial data $(\boldsymbol{x} + \boldsymbol{r}_{\text{uni}})$ during driving, causing major errors in the output segmentation mask.

Adversarial defense

Thus far, we have demonstrated that DNNs can be fooled in many different ways by means of almost imperceptible modifications of the input image. This behavior of DNNs presents challenges to their application within environment perception in AD. Therefore, appropriate adversarial defense strategies are needed to decrease the risk of DNNs being completely fooled by adversarial examples. In this section, some adversarial defense strategies that have been hypothesized and developed to defend against adversarial attacks are presented. In general, adversarial defense strategies can be distinguished as being specific or agnostic to a model at hand. In the following section, we provide a brief introduction to model-specific defense techniques but then focus on model-agnostic ones.

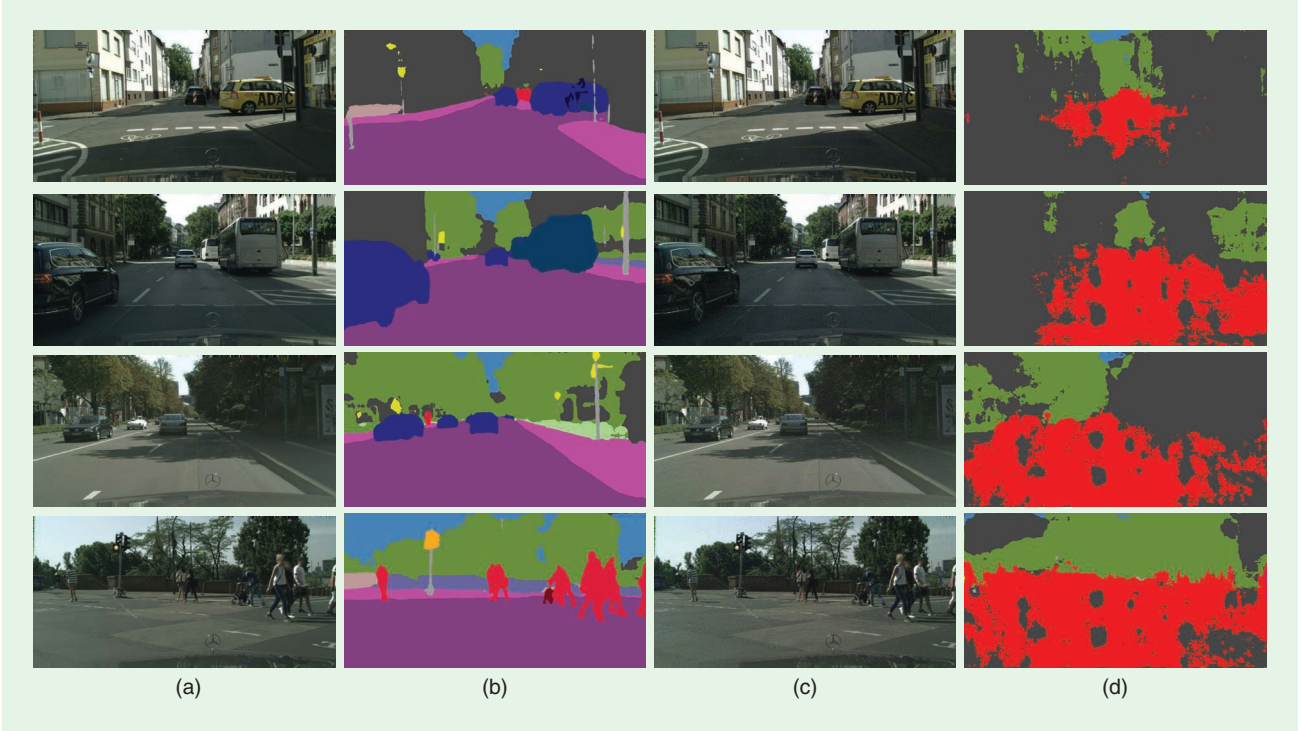


FIGURE 6. Adversarial attacks on the ICNet using a (single) UAP $\boldsymbol{r}_{\text{uni}}$ created by an FFF [22]. We show its effectiveness in fooling the ICNet on four example images with cars or pedestrians from the Cityscapes validation set. Each row corresponds to one attack scenario. (a) A clean input image, (b) a semantic segmentation of a clean input image, (c) an adversarial example created by the FFF, and (d) a semantic segmentation of an adversarial example created by the FFF.

Model-specific defense techniques

Model-specific defense techniques aim at modifying the behavior of a specific DNN in such a way that the respective DNN becomes more robust toward adversarial examples. Note that such a technique can most often be alternatively applied to numerous DNN topologies, however, once it is applied it always defends only the specific DNN at hand. One well-known and intuitive method of model-specific defense techniques is adversarial training. In adversarial training, the original training samples of the DNN are extended using their adversarial counterparts, e.g., created by the FGSM from Goodfellow et al., as shown previously, and then retrained with this set of clean and adversarially perturbed images.

Whereas the performance of the DNN on adversarial examples increases [11], [20], the effect is still marginal [21]. More importantly, it is also not clear what amount or type of adversarial example is sufficient to increase the DNN's robustness up to a desired level. Xie et al. [28] investigated the effect of adversarial examples on the feature maps in several layers. Their observation was that adversarial examples create noise-like patterns in the feature maps. To counter this, they proposed adding trainable denoising layers containing a denoising operation followed by a convolution operation. Xie et al. obtained the best results by using the nonlocal means (NLM) algorithm [6] for feature denoising. Bär et al. [4] explored the effectiveness of teacher–student approaches in defending against adversarial attacks. Here, an additional student DNN is included to increase the robustness against adversarial attacks, assuming that the potential attacker has a difficult time dealing with a constantly adapting student DNN. It was concluded that, in combination with simple output voting schemes this approach could be a promising model-specific defense technique. Nevertheless, a major drawback of model-specific defense techniques is that the respective DNN has to be retrained, or one has to modify the network architecture, which is not always possible when using pretrained DNNs.

Model-agnostic defense techniques

In contrast to model-specific defense techniques, model-agnostic defense techniques, once developed, can be applied in conjunction with any model, as they do not modify the model itself but rather the input data. In particular, the model does not need to be retrained. Hence, it serves in image preprocessing, where the adversary is removed from the input image.

Guo et al. [12] analyzed the effectiveness of nondifferentiable input transformations in destroying adversarial examples. Nondifferentiability is an important property of adversarial defense strategies considering that the majority of adversarial attacks is built on gradient-based optimization. Guo et al. used image quilting (IQ) among some other input transformation techniques and observed IQ to be an effective way of performing a model-agnostic defense against several adversarial attacks. IQ is a technique, wherein the input image $\boldsymbol{x}$ is viewed as a puzzle of small patches $\boldsymbol{x}_{\mathcal{I}_i}$, with i being the position of the center pixel. To remove potential adversaries from an image, each of its patches $\boldsymbol{x}_{\mathcal{I}_i}$, irrelevant of being adversarially perturbed or not, is replaced by a nearest-neighbor patch $\hat{\boldsymbol{x}}_{\mathcal{I}_i} \in \mathcal{P} \subset \mathbb{G}^{h\times b\times C}$ to obtain a quilted image $\boldsymbol{x}^{\mathrm{IQ}}$, with $\mathcal{P}$ being a large set of patches created beforehand from random samples of clean images. The aim is to synthetically construct an adversary-free image having the original semantic content.

Another model-agnostic defense technique is the NLM from Buades et al. [6]. NLM aims at denoising the input image. To accomplish this, NLM replaces each pixel value $\boldsymbol{x}_i$ by

$$\boldsymbol{x}_i^{\mathrm{NLM}} = \sum_{j\in\mathcal{I}} w_{i,j}\boldsymbol{x}_j, \tag{19}$$

with the NLM-denoised pixel $\boldsymbol{x}_i^{\mathrm{NLM}}$, the interpixel weighting factor $w_{i,j} \in [0,1]$ for which $\sum_{j\in\mathcal{I}} w_{i,j} = 1$ holds, and the pixel value $\boldsymbol{x}_j$ at position j. The interpixel weighting factor $w_{i,j}$ relates the respective pixel $\boldsymbol{x}_i$ at pixel position i to the pixel $\boldsymbol{x}_j$ at pixel position j. It is defined by

$$w_{i,j} = \frac{1}{\alpha_i}\exp\left(-\frac{\left\|\boldsymbol{x}_{\mathcal{I}_i} - \boldsymbol{x}_{\mathcal{I}_j}\right\|_{2,a}^2}{h^2}\right), \tag{20}$$

with patches $\boldsymbol{x}_{\mathcal{I}_i}$ and $\boldsymbol{x}_{\mathcal{I}_j}$ centered at pixel positions i and j, the squared Gaussian-weighted Euclidean distance $\|\cdot\|_{2,a}^2$, with $a > 0$ as the standard deviation of the Gaussian kernel, the hyperparameter for the degree of filtering h, and the normalizing factor α_i. By incorporating the squared Gaussian-weighted Euclidean distance, a large weight is put to pixels $\boldsymbol{x}_j$, whose neighborhood $\boldsymbol{x}_{\mathcal{I}_j}$ looks similar to $\boldsymbol{x}_{\mathcal{I}_i}$ (the neighborhood of the respective pixel $\boldsymbol{x}_i$ to be denoised).

The idea behind NLM is to remove the high local dependency of adversarial perturbations. Nevertheless, applying NLM on the complete input image, as stated in (19), can be computationally demanding. Thus, the search window is often reduced to an image region $\mathcal{R}_i \subset \mathcal{I}$ of size $|\mathcal{R}_i| = R \times R$. Note that $\mathcal{I}_i \subset \mathcal{R}_i$, with $|\mathcal{I}_i| = |\mathcal{I}_j| < |\mathcal{R}_i|$.

Now let us look into the results of both model-agnostic defense methods, IQ and NLM, on the adversarial examples shown before. For IQ, the patch data set $\mathcal{P}$ was created using samples from the Cityscapes training set. Here, we followed Guo et al. and collected $|\mathcal{P}| = 1$ million patches of size 5×5 pixels in total. Increasing the size of the patch data set will lead to better approximations of the patches, but on the other hand, also increases the search space. The same holds when decreasing the size of the patches up to a certain level.

For NLM, patches $\mathcal{I}_i$ and $\mathcal{I}_j$ of size 7×7 were used and the image region for the neighboring pixel was restricted according to $|\mathcal{R}_i| = 9 \times 9$ to keep an adequate algorithm complexity. The degree of filtering h was computed by $h = 2.15\tilde{\sigma}(\boldsymbol{x})$, with $\tilde{\sigma}(\boldsymbol{x})$ being an estimate for the Gaussian noise standard deviation on the input image $\boldsymbol{x}$.

Using these settings, we tested IQ, NLM, and a combined version of both, denoted as IQ+NLM, on the adversarial attacks shown in the "Adversarial Attacks" section (see Figures 5 and 6). It is important to note that we applied both defense methods without any extensive hyperparameter search. The adversarial defenses on DNNM-attacked images are depicted in Figure 7. From Figure 7(a) to (e), the original semantic segmentation

mask is reconstructed progressively better, with the combination of NLM and IQ showing the best results [Figure 7(e)]. Comparing NLM and IQ separately, it can be seen that IQ is able to reconstruct the original semantic segmentation mask even more precisely. The same behavior can be observed when looking at the mIoU values in Figure 7, where we report averages over the entire Cityscapes validation set. Altogether, the results show that by combining NLM with IQ, one can lever the destructiveness of DNNM—an important and reassuring observation.

The adversarial defenses on FFF-attacked images are illustrated and supported by the corresponding average mIoU values on the Cityscapes validation set shown in Figure 8. Here, it is not trivial to judge by looking only at the images, which defense is superior, IQ or NLM. In some cases, NLM seems to lead to better results, whereas in other cases, IQ appears to outperform NLM. Yet, looking at the average mIoU values for the entire Cityscapes validation set leads to the conclusion that, overall, NLM is superior to IQ. Moreover, combining NLM with IQ again shows the best results, leading to an overall significant improvement in the restoration of the segmentation masks. This observation is both extremely important and relieving, as the existence of UAPs is particularly dangerous for the use case of DNNs in AD.

Even though we observe a certain level of effectiveness in using model-agnostic defense methods, there is still room for improvement in defending against adversarial attacks. The work of Carlini and Wagner [7] and Athalye et al. [3] are just two of many representative examples. Carlini and Wagner bypassed several state-of-the-art detection systems for adversarial examples with their approach, whereas Athalye et al. circumvented the nondifferentiality property of some state-of-the-art defenses by using different gradient approximation methods.

Summary and future directions

DNNs are one of the most promising technologies for the use case of environment perception in AD. Assuming the environment perception system consists of several camera sensors, a DNN trained for semantic segmentation can be used to perform extensive environment sensing in real time. Nevertheless, today's state-of-the-art DNNs still reveal flaws when fed with specifically crafted inputs, denoted as *adversarial examples*. It was demonstrated step by step that it is quite easy and intuitive to craft adversarial examples for individual input images using the LLCM or the DNNM by simply performing gradient updates on the clean input image. It is even possible to craft adversarial examples to fool not only one but a set of images using the FFF method, without any knowledge of the respective input image to be perturbed. This in turn highlights the importance of appropriate defense strategies. From a safety-concern perspective, the lack of robustness shown by DNNs is a highly relevant and important challenge to deal with before AD vehicles are released for public use.

DNNs' lack of robustness evoked the need for defense strategies and other fallback strategies regarding the safety relevance for AD applications. Model-agnostic defense strategies only modify the potentially perturbed input image to decrease the effect of adversarial attacks. This way, an already-pretrained DNN can be used without the need for retraining or modifying the DNN itself. We explored two model-agnostic defense strategies, namely, IQ and NLM, both on DNNM and FFF attacks, where the combination of IQ and NLM showed the best results on nearly all of the images. Nevertheless, although clearly robustifying the DNNs toward adversarial attacks, the current state of research in model-agnostic defense strategies also showed that the vulnerability of DNNs is not entirely solved yet. However, ensembles of model-agnostic defenses could be promising for tackling adversarial attacks as well as intelligent redundancy (e.g., by teacher–student approaches). We also point out that certification methods [9], [27] should be further investigated to really obtain provable robustness.

What does this mean regarding the application of DNNs for AD? Are today's DNNs not suitable for safety-critical applications in AD? We argue that this is to some extent true, if we only consider applying model-agnostic defenses without certification. DNN training and understandability are two highly dynamic academic fields of research. Research thus far has focused mainly on increasing the performance of DNNs,

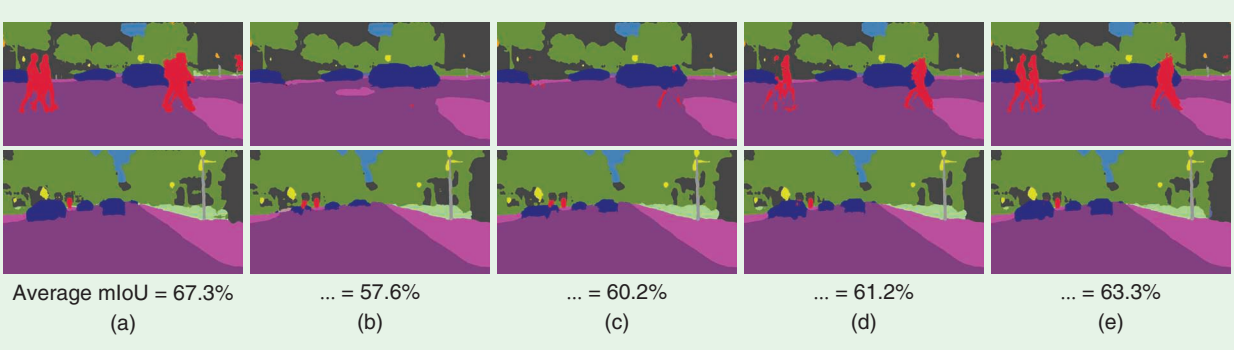


FIGURE 7. Adversarial attacks on the ICNet using a DNNM [18], defended by IQ [12] and the NLM [6]. Both image rows correspond to the examples shown in Figure 5. The top row contains an example of where the DNNM was used to remove pedestrians from the scene, while the bottom row contains an example of where the DNNM was used to remove cars instead. (a) A clean output, (b) an adversarial output using the DNNM, (c) an adversarial output using the DNNM and defended by the NLM, (d) an adversarial output using the DNNM and defended by IQ, and (e) an adversarial output using the DNNM and defended by the NLM and IQ combined. The mIoU values below the bottom row refer to the average mIoU over the entire Cityscapes validation set.

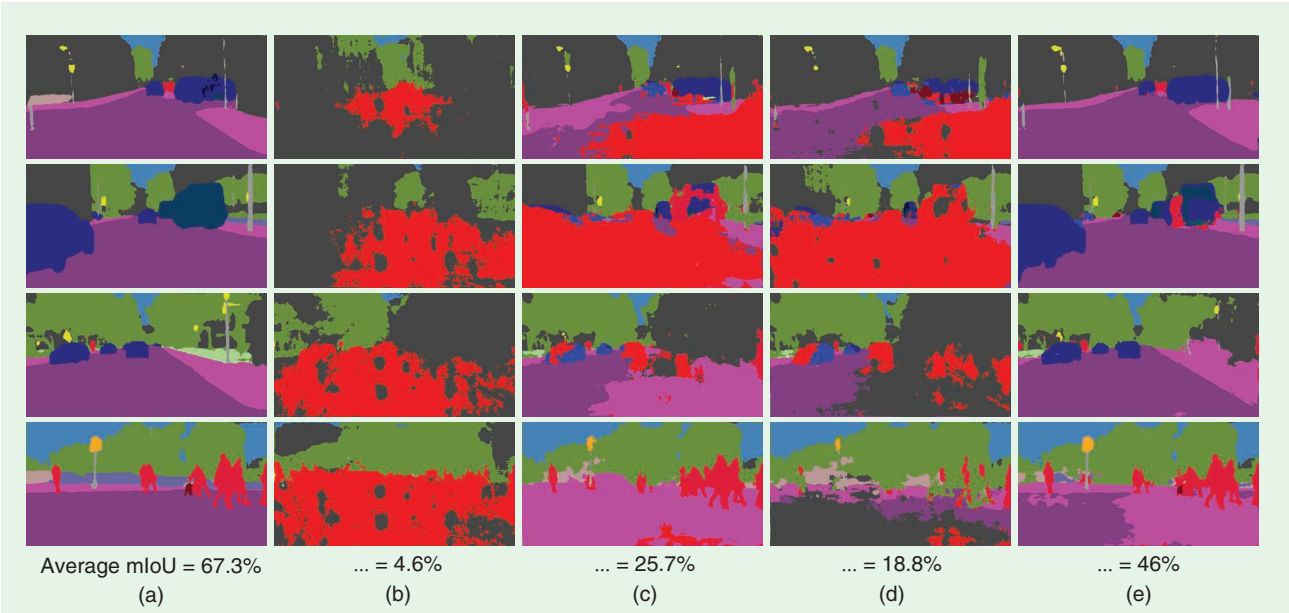


FIGURE 8. Adversarial attacks on the ICNet using an FFF [22], defended by IQ [12] and an NLM [6]. We show results on the four semantic segmentation outputs from Figure 6 using the Cityscapes validation set. Each row corresponds to an attack scenario to be defended by the NLM, IQ, or a combination of both. (a) A clean output, (b) an adversarial output using the FFF, (c) an adversarial output using the FFF and defended by the NLM, (d) an adversarial output using the FFF and defended by IQ, and (e) an adversarial output using the FFF and defended by the NLM and IQ combined. The mIoU values in the bottom row refer to the average mIoU over the entire Cityscapes validation set.

widely neglecting their robustness and certification. To develop employable machine learning-based functions that are realistically usable in a real-world setting, it is extremely important to establish their robustness against slight input alterations in addition to improving the task performance. Furthermore, new mature defense and certification strategies are needed, including fusion approaches, redundancy concepts, and modern fallback strategies. We especially recommend that automotive companies focus on the certification of DNNs; otherwise, doors would be opened to potentially fatal attacks, which in turn would have consequences on the public acceptance of AD.

Acknowledgments

We gratefully acknowledge the support of Volkswagen Group Automation, Wolfsburg, Germany, and would like to thank Nico M. Schmidt and Zeyun Zhong for their help with setting up final experiments.

Authors

Andreas Bär (andreas.baer@tu-bs.de) received his B.Eng. degree from Ostfalia University of Applied Sciences, Wolfenbüttel, Germany, in 2016, and his M.Sc. degree from Technische Universität Braunschweig, Braunschweig, Germany, in 2018, where he is currently a Ph.D. degree candidate in the Faculty of Electrical Engineering, Information Technology, and Physics. His research interests include convolutional neural networks for camera-based environment perception and the robustness of neural networks to adversarial attacks. In 2020, he won the Best Paper Award at the Workshop on Safe Artificial Intelligence for Automated Driving, held in conjunction with the IEEE Conference on Computer Vision and Pattern Recognition, along with coauthors Serin John Varghese, Fabian Hüger, Peter Schlicht, and Tim Fingscheidt.

Jonas Löhdefink (j.loehdefink@tu-bs.de) received his B. Eng. degree from Ostfalia University of Applied Sciences, Wolfenbüttel, Germany, in 2015, and his M.Sc. degree from Technische Universität Braunschweig, Braunschweig, Germany, in 2018, where he is currently a Ph.D. degree candidate in the Faculty of Electrical Engineering, Information Technology, and Physics. His research interests include learned image compression and quantization approaches by means of convolutional neural networks and generative adversarial networks.

Nikhil Kapoor (nikhil.kapoor@volkswagen.de) received his B.Eng. degree from the Army Institute of Technology, Pune, India, in 2012, and his M.Sc. degree from RWTH Aachen University, Germany, in 2018. Currently, he is a Ph.D. degree candidate at Technische Universität Braunschweig, Braunschweig, Germany, in cooperation with Volkswagen Group Research. His research focuses on training strategies that range from improving the robustness of neural networks for camera-based perception tasks to augmentations and adversarial perturbations using concept-based learning.

Serin John Varghese (john.serin.varghese@volkswagen.de) received his B.Eng. degree from the University of Pune, India, in 2013, and his M.Sc. degree from Technische Universität Chemnitz, Germany, in 2018. Currently, he is a Ph.D. degree candidate at Technische Universität Braunschweig, Braunschweig, Germany, in cooperation with Volkswagen Group Research. His research is focused on compression techniques for convolutional neural networks used

for perception modules in automated driving, with a focus on not only inference times but also maintaining, and even improving, the robustness of neural networks.

Fabian Hüger (fabian.hueger@volkswagen.de) received his M.Sc. degree in electrical and computer engineering from the University of California, Santa Barbara, as a Fulbright scholar in 2009. He received his Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from the University of Kassel, Germany, in 2010 and 2014, respectively. He joined Volkswagen Group Research, Germany, in 2010, and his current research is focused on safe and efficient use of artificial intelligence for autonomous driving.

Peter Schlicht (peter.schlicht@volkswagen.de) received his Ph.D. degree in mathematics from the University of Leipzig, Germany. After a two-year research stay at the Ecole Polytechnique Fédérale, Lausanne, Switzerland, he joined Volkswagen Group Research, Wolfsburg, Germany, in 2016 as an artificial intelligence (AI) architect. There he deals with research questions on AI technologies for automatic driving. His research interests include methods used for monitoring, explaining, and robotizing deep neural networks as well as securing them.

Tim Fingscheidt (t.fingscheidt@tu-bs.de) received his Dipl.-Ing. and Ph.D. degrees in electrical engineering, both from RWTH Aachen University, Germany, in 1993 and 1998, respectively. Since 2006, he has been a full professor with the Institute for Communications Technology, Technische Universität Braunschweig, Braunschweig, Germany. He received the Vodafone Mobile Communications Foundation prize in 1999 and the 2002 prize of the Information Technology branch of the Association of German Electrical Engineers (VDE ITG). In 2017, he coauthored the ITG award-winning publication, "Turbo Automatic Speech Recognition." He has been the speaker of the Speech Acoustics Committee ITG AT3 since 2015. He served as an associate editor of *IEEE Transactions on Audio, Speech, and Language Processing* (2008–2010) and was a member of the IEEE Speech and Language Processing Technical Committee (2011–2018). His research interests include speech technology and vision for autonomous driving. He is a Senior Member of IEEE.

References

[1] A. Arnab, O. Miksik, and P. H. S. Torr, "On the robustness of semantic segmentation models to adversarial attacks," in *Proc. CVPR*, Salt Lake City, UT, June 2018, pp. 888–897. doi: 10.1109/CVPR.2018.00099.

[2] F. Assion, P. Schlicht, F. Gressner, W. Gunther, F. Hüger, N. Schmidt, and U. Rasheed, "The attack generator: A systematic approach towards constructing adversarial attacks," in *Proc. CVPR—Workshops*, Long Beach, CA, June 2019, pp. 1370–1379. doi: 10.1109/CVPRW.2019.00177.

[3] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. ICML*, Stockholm, Sweden, July 2018, pp. 274–283.

[4] A. Bär, F. Hüger, P. Schlicht, and T. Fingscheidt, "On the robustness of redundant teacher-student frameworks for semantic segmentation," in *Proc. CVPR—Workshops*, Long Beach, CA, June 2019, pp. 1380–1388. doi: 10.1109/CVPRW.2019.00178.

[5] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 4, pp. 6–22, Oct. 2014. doi: 10.1109/MITS.2014.2336271.

[6] A. Buades, B. Coll, and J.-M. More, "A non-local algorithm for image denoising," in *Proc. CVPR*, San Diego, CA, June 2005, pp. 60–65. doi: 10.1109/CVPR.2005.38.

[7] N. Carlini and D. Wagne "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. AISec*, New York, Nov. 2017, pp. 3–14. doi: 10.1145/3128572.3140444.

[8] M. Cordts et al., "The cityscapes dataset for semantic urban scene understanding," in *Proc. CVPR*, Las Vegas, NV, June 2016, pp. 3213–3223. doi: 10.1109/CVPR.2016.350.

[9] K. Dvijotham, M. Garnelo, A. Fawzi, and P. Kohli, "Verification of deep probabilistic models," in *Proc. NIPS—Workshops*, Montréal, QC, Dec. 2018, pp. 1–5.

[10] F. Engels, P. Heidenreich, A. M. Zoubir, F. K. Jondral, and M. Wintermantel, "Advances in automotive radar: A framework on computationally efficient high-resolution frequency estimation," *IEEE Signal Process. Mag.*, vol. 34, no. 2, pp. 36–46, Mar. 2017. doi: 10.1109/MSP.2016.2637700.

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. ICLR*, San Diego, CA, May 2015, pp. 1–10.

[12] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," in *Proc. ICLR*, Vancouver, BC, Apr. 2018, pp. 1–12.

[13] L. Hobert, A. Festag, I. Llatser, L. Altomare, F. Visintainer, and A. Kovacs, "Enhancements of V2X communication in support of cooperative autonomous driving," *IEEE Commun. Mag.*, vol. 53, no. 12, pp. 64–70, 2015. doi: 10.1109/MCOM.2015.7355568.

[14] S.-W. Kim, W. Liu, M. H. Ang, E. Frazzoli, and D. Rus, "The impact of cooperative perception on decision making and planning of autonomous vehicles," *IEEE Intell. Transp. Syst. Mag.*, vol. 7, no. 3, pp. 39–50, 2015. doi: 10.1109/MITS.2015.2409883.

[15] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. ICLR*, Toulon, France, Sept. 2017, pp. 1–17.

[16] J. Levinson et al., "Towards fully autonomous driving: Systems and algorithms," in *Proc. Intelligent Vehicles Symp. (IV),* Baden-Baden, Germany, June 2011, pp. 163–168. doi: 10.1109/IVS.2011.5940562.

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, Boston, MA, June 2015, pp. 3431–3440. doi: 10.1109/CVPR.2015.7298965.

[18] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer, "Universal adversarial perturbations against semantic image segmentation," in *Proc. ICCV*, Venice, Italy, Oct. 2017, pp. 2774–2783. doi: 10.1109/ICCV.2017.300.

[19] R. Michelmore, M. Wicker, L. Laurenti, L. Cardelli, Y. Gal, and M. Kwiatkowska, Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. Sept. 2019. [Online]. Available: arXiv:1909.09884

[20] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal Adversarial Perturbations," in *Proc. CVPR*, Honolulu, HI, July 2017, pp. 1765–1773 doi: 10.1109/CVPR.2017.17.

[21] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. CVPR*, Las Vegas, NV, June 2016, pp. 2574–2582 doi: 10.1109/CVPR.2016.282.

[22] K. R. Mopuri, A. Ganeshan, and V. B. Radhakrishnan, "Generalizable data-free objective for crafting universal adversarial perturbations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2452–2465, Oct. 1, 2019. doi: 10.1109/TPAMI.2018.2861800.

[23] K. R. Mopuri, U. Garg, and R. V. Babu, "Fast feature fool: A data independent approach to universal adversarial perturbations," in *Proc. BMVC*, London, Sept. 2017, pp. 1–12.

[24] S. M. Patole, M. Torlak, D. Wang, and M. Ali, "Automotive radars: A review of signal processing techniques," *IEEE Signal Process. Mag.*, vol. 34, no. 2, pp. 22–35, Mar. 2017. doi: 10.1109/MSP.2016.2628914.

[25] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. ICLR*, Montréal, QC, Dec. 2014, pp. 1–10.

[26] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *Proc. Intelligent Vehicles Symp. (IV)*, Gold Coast, QLD, June 2013, pp. 763–770. doi: 10.1109/IVS.2013.6629559.

[27] M. Wu, M. Wicker, W. Ruan, X. Huang, and M. Kwiatkowska, "A game-based approximate verification of deep neural networks with provable guarantees," *Theor. Comput. Sci.*, vol. 807, pp. 298–329, Mar. 2019. doi: 10.1016/j.tcs.2019.05.046.

[28] C. Xie, Y. Wu, L. van der Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proc. CVPR*, Long Beach, CA, June 2019, pp. 501–509 doi: 10.1109/CVPR.2019.00059.

[29] L. Zeng, K. Zhang, Q. Han, S. Chen, L. Ye, R. Wang, J. Lei, and Q. Xie, "Research of path planning model based on hotspots evaluation," in *Proc. Intelligent Vehicles Symp. (IV)*, Paris, France, June 2019, pp. 2193–2198. doi: 10.1109/IVS.2019.8814163.

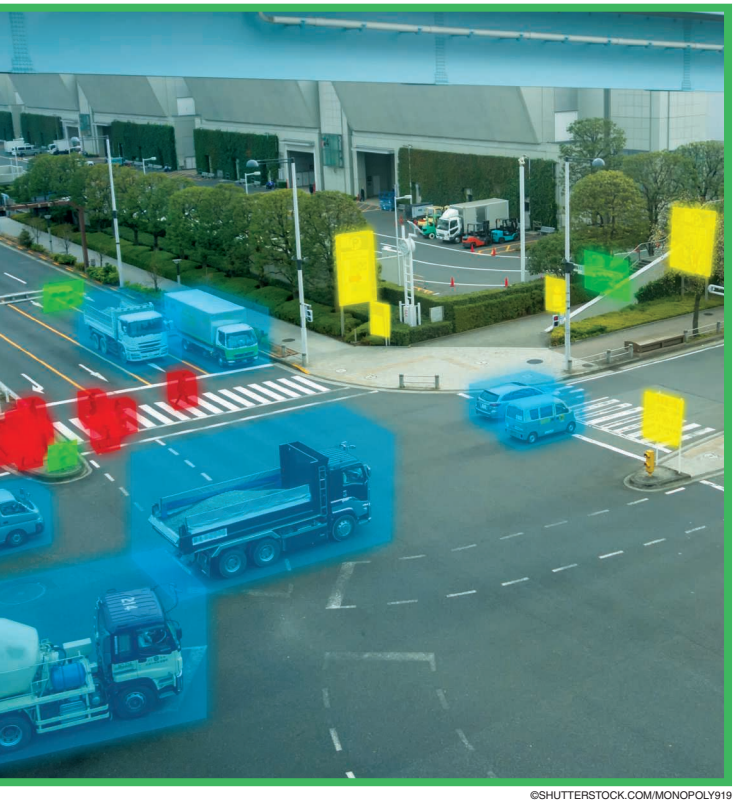
[30] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *Proc. ECCV*, Munich, Germany, Sept. 2018, pp. 418–434. doi: 10.1007/978-3-030-01219-9_25.

SP

Mazin Hnewa and Hayder Radha

Object Detection Under Rainy Conditions for Autonomous Vehicles

A review of state-of-the-art and emerging techniques



©SHUTTERSTOCK.COM/MONOPOLY919

Digital Object Identifier 10.1109/MSP.2020.2984801
Date of current version: 24 December 2020

Advanced automotive active safety systems, in general, and autonomous vehicles, in particular, rely heavily on visual data to classify and localize objects, such as pedestrians, traffic signs and lights, and nearby cars, to help the corresponding vehicles maneuver safely in their environments. However, the performance of object detection methods could degrade rather significantly in challenging weather scenarios, including rainy conditions. Despite major advancements in the development of deraining approaches, the impact of rain on object detection has largely been understudied, especially in the context of autonomous driving.

Introduction

Visual data plays a critical role in enabling automotive advanced driver-assistance systems and autonomous vehicles to achieve high levels of safety while the cars and trucks maneuver in their environments. Hence, emerging autonomous vehicles are employing cameras and deep learning-based methods for object detection and classification [1]–[3]. These methods predict bounding boxes that surround detected objects and classify probabilities associated with each bounding box. In particular, convolutional neural network (CNN)-based approaches have shown very promising results in the detection of pedestrians, vehicles, and other objects [4]–[10]. These neural networks are usually trained using a large amount of visual data captured in favorable clear conditions. However, the performance of such systems in challenging weather, such as rainy conditions, has not been thoroughly surveyed or studied.

The quality of visual signals captured by autonomous vehicles can be impaired and distorted in adverse weather conditions, most notably in rain, snow, and fog. Such conditions minimize the scene contrast and visibility, and this could lead to a significant degradation in the ability of the vehicle to detect critical objects in the environment. Depending on the visual effect, adverse weather conditions can be classified as steady (such as fog, mist, and haze) or dynamic, which have more complex effects (such as rain and

snow) [11]. In this article, we focus on rain because it is the most common dynamic challenging weather condition that impacts virtually every populated region of the globe. Furthermore, there has been a great number of recent efforts that attempt to mitigate the effect of rain in the context of visual processing. While addressing the effect of other weather conditions has been receiving some, yet minimal, attention, the volume of work regarding the mitigation of rain is far more prevalent and salient within different research communities.

It is worth noting that rain consists of countless drops that have a wide range of sizes and complex shapes, and rain spreads quite randomly, with varying speeds when falling on roadways, pavement, vehicles, pedestrians, and other objects in the scene. Moreover, raindrops naturally cause intensity variations in images and video frames. In particular, every raindrop blocks some of the light that is reflected by objects in a scene. In addition, rain streaks lead to low contrast and elevated levels of whiteness in visual data. Consequently, mitigating the effect of rain on visual data is arguably one of the most challenging tasks that autonomous vehicles will have to perform, due to the fact that it is quite difficult to detect and isolate raindrops, and it is equally problematic to restore the information that is lost or occluded by rain.

The impact of rain on object detection has largely been understudied, especially in the context of autonomous driving.

Meanwhile, there has been noticeable progress in the development of advanced visual deraining algorithms [12]–[17]. Thus, one natural and intuitive solution for mitigating the effect of rain on active safety systems and autonomous vehicles is to employ robust deraining algorithms and then apply the desired object detection approach to the resulting derained signal. State-of-the-art deraining algorithms, however, are designed to remove the visual impairments caused by rain, while attempting to restore the original signal with minimal distortion. Hence, the primary objective of these algorithms, in general, is to preserve the visual quality as measured by popular performance metrics, such as the peak signal-to-noise-ratio and structure similarity index (SSIM) [18]. These metrics, however, do not reflect a viable measure for analyzing the performance of the system for more complex tasks, such as object detection.

The main objective of this article is to survey and present a tutorial on state-of-the-art and emerging techniques that are leading candidates for mitigating the influence of rainy conditions on an autonomous vehicle's ability to detect objects. In that context, our goal includes surveying and analyzing the performance of object detection methods that are representatives of state-of-the-art frameworks that are being considered for integration into autonomous vehicles' artificial intelligence (AI) platforms. Furthermore, we survey and highlight the inherent limitations of leading deraining algorithms, deep learning-based domain adaptation, and image translation frameworks in the context of rainy conditions.

While surveying a variety of relevant techniques in this area, we present experimental results with the objective of highlighting the urgent need for developing new paradigms for addressing the challenges of autonomous driving in severe weather conditions. Although generative model-based image translation and domain adaptation approaches do show some promise, one overarching conclusion that we aim to convey through this article is that current solutions do not adequately mitigate the realistic challenges for autonomous driving in diverse weather conditions. This overarching conclusion opens the door for the research community to pursue and explore new frameworks that address this timely and crucial problem area. The architectures highlighting the main parts of this article are highlighted in Figure 1.

Object detection for autonomous vehicles in clear and rainy conditions

The level of degradation in the performance of an object detection method, trained in certain conditions, is influenced heavily by 1) how different the training and testing domains are and 2) the type of deep learning-based architecture used for object detection. Most recent object detectors are CNN-based networks, such as the single-shot multibox detector [9], region-based fully convolutional network [10], You Only Look Once (YOLO) [8], RetinaNet [7], and Faster Regions With CNNs (R-CNNs) [6]. To that end, we review two major classes of object detection frameworks that are both popular and representative of deep learning-based approaches. As we see later in this tutorial, these two classes of architectures exhibit different levels and forms of degradation in response to challenging rainy conditions, and they also perform rather differently in conjunction with potential rain mitigation frameworks.

In particular, we briefly describe the underlying architectures for Faster R-CNN and YOLO as representatives of two major classes of object detection algorithms. Faster R-CNN is arguably the most popular of the object detection algorithms that are based on a two-stage deep learning architecture; one stage is for identifying region proposals (RPs), and the second is for refining and assigning class probabilities for the corresponding regions. YOLO, on the other hand, is a representative of detection frameworks that directly operate on the whole image.

Deep learning-based methods for object detection

The utility of CNNs for object detection was well established prior to the introduction of the notion of RPs, commonly known as *R-CNN* [4], where "R" stands for regions or RPs. A fast version of R-CNN was later introduced [5], and then Ren et al. [6] presented the idea of the RP network (RPN) that shares convolutional layers with Fast R-CNN [5]. The RPN is merged with the Fast R-CNN into one unified network that is known as *Faster R-CNN* to achieve more computationally efficient detection. Under Faster R-CNN, an input image is fed to a feature extractor, such as the ZF model [19] or VGG-16 [20], to produce a feature map. Then, the RPN utilizes this

feature map to predict RPs (regions in the image that could potentially contain objects of interest).

In that context, many RPs are quite overlapped with each other, with significant numbers of pixels common among multiple RPs. To filter out the substantial redundancy that might occur with such a framework, nonmaximum suppression (NMS) [21] is used to remove redundant regions while keeping the ones that have the highest prediction scores. Subsequently, each regional proposal that survives the NMS process is used by a region-of-interest (RoI) pooling layer to crop the corresponding features from the feature map. This cropping process produces a feature vector that is fed to two fully connected layers: one predicts the offset values of a bounding box of an object with respect to the regional proposal, and the other predicts class probabilities for the predicted bounding box. Figure 2 shows a high-level architecture for Faster R-CNN.

The quality of visual signals captured by autonomous vehicles can be impaired and distorted in adverse weather conditions.

On the other hand, Redmon et at. [8] proposed to treat object detection as a regression problem, and they developed a unified neural network that is called *YOLO* to predict bounding boxes and class probabilities directly from a full image in one evaluation. Under YOLO, an input image is divided into a specific set of grid cells, and each cell is responsible for detecting objects whose centers are located within that cell. To that end, each cell predicts a certain number of bounding boxes, and it also predicts the confidence scores for these boxes in terms of the likelihood that they contain an object. Furthermore, it predicts conditional class probabilities, given that it has an object. In this case, there are potentially many wrongly predicted bounding boxes. To filter them out and provide the final detection result, a threshold is used on the confidence scores of the predicted bounding boxes. Figure 2 illustrates the general architecture of YOLO.

Object detection performance for neural network architectures in clear and rainy conditions

Here, we provide an insight into the level of degradation caused by rainy conditions on the performance of the two major deep learning architectures described previously. In particular, we focus on the following fundamental question: how much degradation a deep neural network that is trained in clear conditions will suffer when tested in rainy weather. In that context, we first describe the data set that we used for training and testing; this is followed by presenting some visual and numerical results. For the purpose of this tutorial, we needed a rich data set that was captured in diverse weather conditions. Despite the fact that there are few notable data sets [22]–[24], which are quite popular among the computer vision and AI research communities in terms of training deep NNs, there is only one (arguably two [25], [26]) that is properly labeled and annotated for our purpose and hence that could be used for training and testing for different weather conditions. In particular, we use the Berkeley Deep Drive (BDD100K) data set

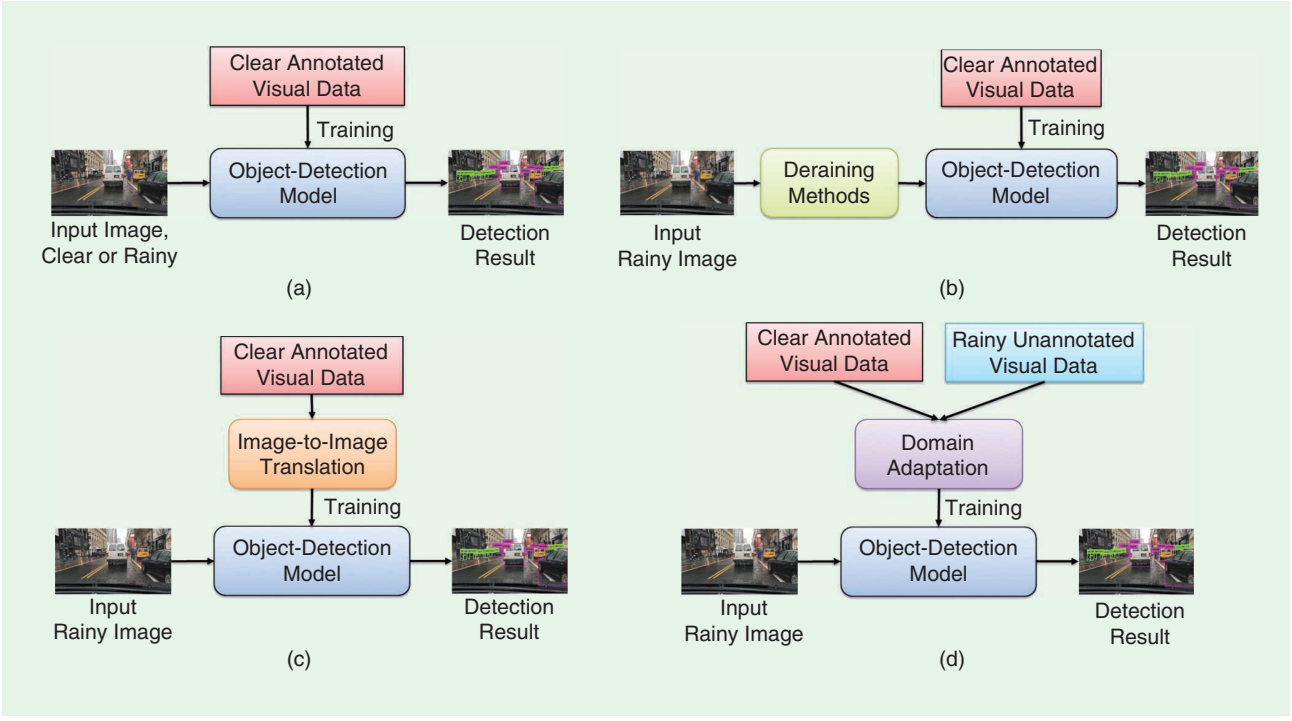


FIGURE 1. The architectures highlighting the main parts of the article. The (a) "Object Detection for Autonomous Vehicles in Clear and Rainy Conditions" section, (b) "Deraining in Conjunction With Object Detection" section, (c) "Unsupervised Image-to-Image Translation" section, and (d) "Domain Adaptation" section [25].

[25] because it contains image tagging for weather (i.e., each image in the data set is labeled with its weather condition, such as clear, rainy, foggy, and so on). Meanwhile, although some other data sets, such as nuScenes [26], might contain some visuals captured in rainy conditions, they do not have weather tagging. Hence, choosing the BDD100k data set was influenced by the fact that we could select images illustrating a specific weather condition.

Moreover, the BDD100K has 100,000 video clips captured in diverse geographic, environmental, and weather conditions. It is worth noting that only one selected frame from each video is annotated with object bounding boxes as well as image-level tagging. Examples of annotated frames in clear and rainy weather are shown in Figure 3. In this article, we consider the four classes (vehicle, pedestrian, traffic light, and traffic sign) that are labeled and provided as ground-truth objects within the BDD100K data set. Naturally, these four classes are among the most critical objects for an autonomous vehicle. In this tutorial, we use images that are captured in clear weather from the designated training set of the BDD100K to form our underlying training data set. We refer to this training data as the *train clear* set, which we used consistently to train the detection methods for the different scenarios covered in this article. For testing, we use a collection of clear weather images from the testing set of the BDD100K. We refer to this latter group as the *test clear* set. Table 1 gives the number of annotated objects in the train clear and test clear data sets.

One approach to demonstrate the impact of rain on object detection methods that are trained in clear conditions is by rendering synthetic rain [27]–[29] within the images of the test clear set. Then, the synthetic rainy data can be used to test the already trained object detection methods. The benefit of this approach is that one would have the exact same underlying content in both testing data sets in terms of the

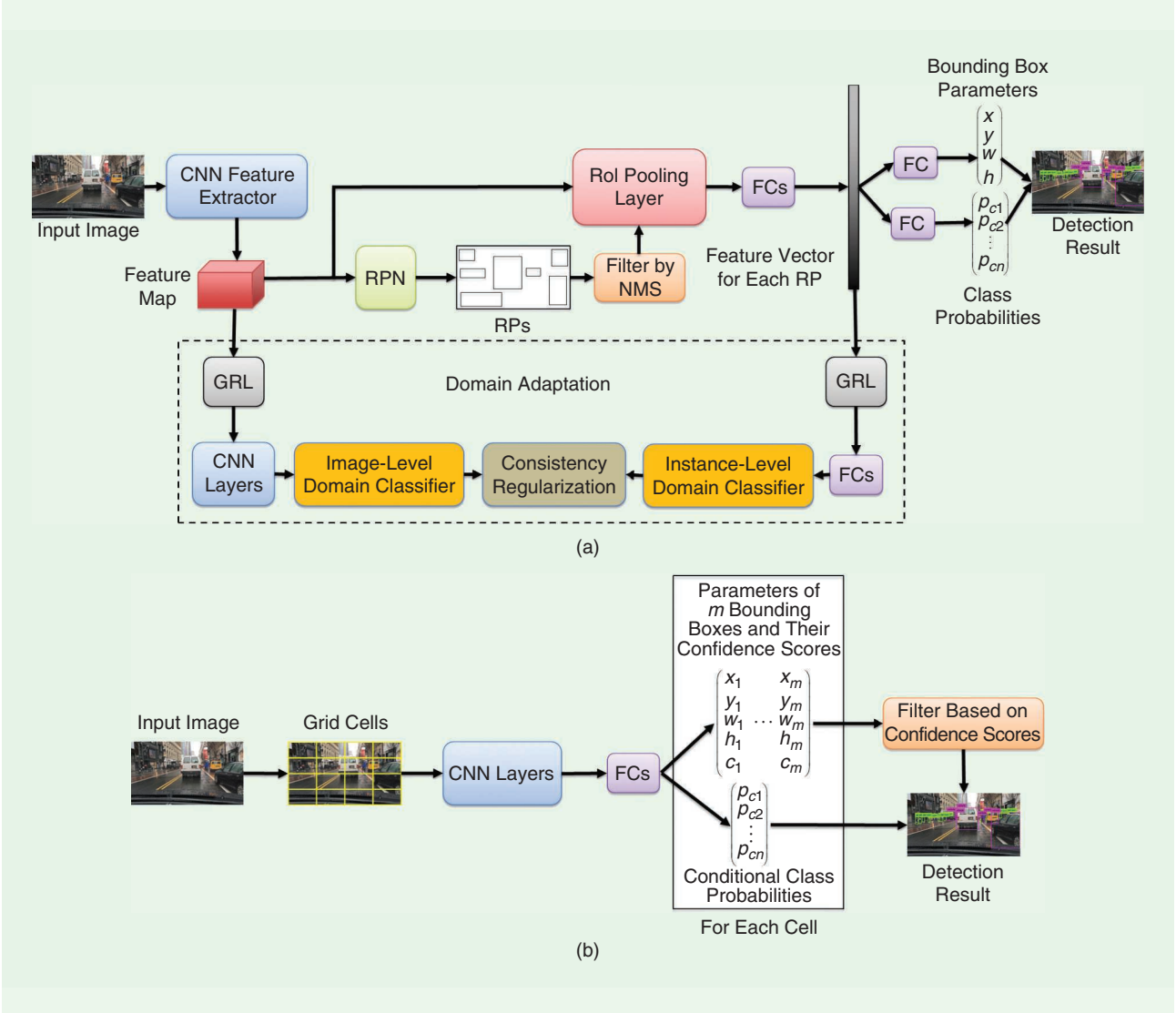


FIGURE 2. The high-level architectures of the detection methods that are used in this tutorial. The (a) Faster R-CNN and (b) YOLO. The domain adaptation of Faster R-CNN is explained in the "Domain Adaptation" section [25]. GRL: gradient reversal layer.

FIGURE 3. The examples of the annotated images in the BDD100K data set [25]. Images in (a) are tagged as clear weather, and images in (b) and (c) are tagged as rainy weather. However, images in (c) are wrongly tagged; they actually show clear or cloudy weather.

objects within the scene, with one set representing the original clear-weather content when the data was captured and another set with the synthetic rain. This would plainly show the impact of rain, as the visual objects are the same in both tested sets (the test clear set and a test synthetic rain set). However, from our extensive experience in this area, we noticed that most well-known rain simulation methods do not render realistic rain that viably captures actual and true rainy weather conditions, especially for a driving vehicle. Thus, when comparing the two scenarios, this discrepancy between synthetic and natural (real) rainy conditions will lead to domain mismatch. As a result, we do not test the detection methods using synthetic rain in our study because those techniques will not demonstrate the impact of true natural rain on a driving vehicle.

Alternatively, we use images captured in real rainy conditions from the training and testing sets of the BDD100K data set to test the object detection methods. It is worth noting that several images in the data set are wrongly tagged as rainy weather when they actually show clear or cloudy conditions, such as the examples in Figure 3(c). To solve this problem, we manually selected the images that were truly captured in rainy weather to form what we refer to as the *test rainy* set. Equally important, we elected to have both the test clear and test rainy sets include approximately the same number of annotated objects, as shown in Table 1, to provide statistically comparable results.

Table 1. The number of annotated objects in the training and testing sets that are used in our study.

Set	Vehicles	Pedestrians	Traffic Signs	Traffic Lights
Train clear	149,548	16,777	43,866	26,002
Test clear	13,721	2,397	3,548	4,239
Test rainy	13,724	2,347	3,551	4,246

It is important to make one final critical note regarding the currently available data sets for training NNs designed for object detection. The lack of data sets captured in diverse conditions, including rain, snow, fog, and other weather scenarios, represents one of the most challenging aspects of achieving a viable level of training for autonomous vehicles. Even for the BDD100K data set, which is one of very few publicly available data sets with properly annotated objects captured in different weather conditions, there is not a sufficient amount of annotated visual content that is truly viable for training in rainy weather. This fundamental issue with the lack of real training data for rainy and other conditions

has clearly become a major obstacle, to the extent that leading high-tech companies working in the area have begun a focused effort designated specifically for collecting data in rainy conditions. For example, Waymo recently announced plans to begin collecting data for autonomous driving in rainy conditions [30].

Performance metric

To evaluate the detection performance, we compute the mean average precision (mAP). This metric has been the most popular performance measure since the time when it was originally defined in the PASCAL Visual Object Classes Challenge 2012 for evaluating detection methods [31]. To determine the mAP, a precision/recall curve is first computed based on the prediction result against the ground truth. A prediction is considered a true positive if its bounding box has 1) an intersection-over-union value greater than 0.5 relative to the corresponding ground-truth bounding box and 2) the same class label as the ground truth. Then, the curve is updated by making the precision monotonically decrease. This is achieved by setting the precision for recall r to the maximum precision obtained for any recall $r > r$. The AP is the area under the updated precision/recall curve. It is computed by numerical integration. Finally, the mAP is the mean of the AP among all classes.

Results and discussion

We trained the detection methods (Faster R-CNN and YOLO) using the train clear set, which is described in the "Object Detection Performance for Neural Network Architectures in Clear and Rainy Conditions" section. We used the same training settings and hyperparameters that were employed in the original papers [6], [8]. Then, we tested the trained models by using the test clear and test rainy sets to illustrate the impact of rain. Table 2 presents the AP for each class as well as the mAP evaluated based on the AP values of the classes. From the table, we observe that the mAP clearly declines in rainy weather compared to clear weather using both Faster R-CNN and YOLO. Consequently, these results undoubtedly illustrate that the performance of an object detection framework that is trained using clear visuals could significantly degrade in rainy weather conditions. The performance decreases due to the fact that rain covers and distorts important details of the underlying visual features, which are used by detection methods to classify and localize objects. Figure 4 provides examples when the detection methods fail to perceive most objects in rainy conditions.

Moreover, one can notice that in rainy conditions, the AP for the pedestrian and traffic light classes declines more significantly than the decrease in performance for vehicle and traffic sign classes. This discrepancy in performance degradation for different objects is due to a variety of factors. For example, vehicles usually occupy larger regions within an image frame than other types of objects; hence, even when raindrops or rain streaks cover a visual of a vehicle, there are still sufficient features that can be extracted by the detection method. Furthermore, traffic signs are normally made from materials that have high reflectivity, which makes it easier for an object detection method to achieve higher accuracy, even when a traffic sign visual is distorted by some rain. Overall, in both cases, the important features needed for reliable detection are still salient within the underlying deep NNs of the detection algorithms. Nevertheless, rain could still impact the detection of vehicle and traffic signs, as shown in the bottom three rows of Figure 4.

Deraining in conjunction with object detection

Deraining methods attempt to remove the effect of rain and restore an image of a scene that has been distorted by raindrops or rain streaks, while preserving important visual details. In this tutorial, we review three recently developed deraining algorithms [14]–[16] that employ deep learning frameworks for the removal of rain from a scene. The high-level architectures of these methods are illustrated in Figure 5. In the following, we briefly describe these three deraining methods and

Table 2. The AP for each class and mAP evaluated based on the AP values of the classes.

	Faster R-CNN					YOLO-V3				
Mitigating Technique	**V-AP**	**P-AP**	**TL-AP**	**TS-AP**	**mAP**	**V-AP**	**P-AP**	**TL-AP**	**TS-AP**	**mAP**
None (clear conditions*)	72.61	40.99	26.07	38.12	44.45	76.57	37.12	46.22	50.56	52.62
None (rainy conditions**)	67.84	32.58	20.52	35.04	39	74.15	32.07	41.07	50.27	49.39
Deraining: DDN [14]	67	28.55	20.02	35.55	37.78	73.07	29.89	40.05	48.74	47.94
Deraining: DeRaindrop [15]	64.37	29.27	18.32	33.33	36.32	70.77	30.16	37.7	48.03	46.66
Deraining: PReNet [16]	63.69	24.39	17.4	31.68	34.29	70.83	27.36	35.49	43.78	44.36
Image translation: UNIT [32]	**68.47**	**32.76**	18.85	**36.2**	**39.07**	74.14	**34.19**	**41.18**	48.41	**49.48**
Domain adaptation [33]	67.36	**34.89**	19.24	**35.49**	**39.24**	Not applicable				

V-AP: vehicle AP; P-AP: pedestrian AP; TL-AP: traffic light AP; TS-AP: traffic sign AP; DDN: deep-detail network.
*The top row shows the performance in clear conditions (i.e., using the test clear set), while all other rows show the performance in rainy conditions (i.e., using the test rainy set).
**Significant degradation in performance can be observed due to rainy conditions (text in red) relative to the performance in clear conditions (top row). Improvements in performance by mitigating the effect of rain can be observed using generative model-based image translation and/or domain adaptation (highlighted in bold). Meanwhile, deraining algorithms do not improve, and most of the time further degrade, the performance.

highlight their limitations when employing them in conjunction with object detection methods.

Deep detail network

Fu et at. [14] proposed a deep detail network (DDN) to remove rain from a single image. They employed a CNN, which is a residual neural network (ResNet) [34], to predict the difference between clear and rainy images, and used this difference to remove rain from a scene. In particular, the DDN exploits only the rainy image's high-frequency details, and it uses such details as input to the ResNet while ignoring the low-frequency background (interference) of the same underlying scene.

Attentive generative adversarial network

Qian et at. [15] proposed an attentive generative adversarial network (GAN) that is called *DeRaindrop* to remove raindrops from images. In this method, a GAN [35] with visual attention is employed to learn raindrop areas and their surroundings. The first part of the generative network, known as the *attentive recurrent network* (*ARN*), produces an attention map to guide the next stage of the DeRaindrop framework. The ARN includes the ResNet, long short-term memory (LSTM) [36], and CNN layers. The second stage of DeRaindrop, which is known as the *contextual autoencoder*, operates on the attention map, and hence it focuses on (or pays more attention to) the raindrop areas. The overall process from the two stages is expected to clean images free of raindrops. The architecture also includes a discriminative network, which assesses the generated rain-free images to verify that they are similar to real ones that have been used during the training process.

Progressive image deraining network

Ren et at. [16] proposed a PReNet to recursively remove rain from a single image. At each iteration, some rain is removed, and the remaining rain can be progressively erased during subsequent iterations. Consequently, after a certain number of iterations, most of the rain should be removed, leading to a rain-free, quality image. In addition to several

FIGURE 4. The (a) ground truth shown with example detection results using (b) Faster R-CNN and (c) YOLO for different visual scenes from the test rainy set [25].

residual blocks of ResNet, the PReNet includes a CNN layer that operates on both the original rainy image and the current output image. The PReNet also includes another CNN layer to generate the output image. Furthermore, a recurrent layer is appended to exploit dependencies in the deep features across iterations via convolutional LSTM. To train the PReNet, a single negative SSIM [18] or mean-square-error loss is used.

Relying purely on state-of-the-art deraining solutions does not represent a viable approach for mitigating the impact of rain.

Results and discussion

To demonstrate the performance of the deraining methods outlined previously, we apply the pretrained deraining models provided by the corresponding authors to the test rainy set as a prepossessing step. After applying the deraining algorithms, which are anticipated to remove the rain from the input visual data and generate rain-free clear visuals, we feed the derained images into the object detection methods. Table 2 shows the performance of the detection methods after applying the deraining approaches. It can be seen that the deraining algorithms actually degrade the detection performance when compared to directly using the rainy images as input into the corresponding detection frameworks. This is true for both Faster R-CNN and YOLO. One important factor for this degradation in performance is that the deraining process tends to smooth out the input image, and hence it distorts the meaningful information and distinctive features of a scene while attempting to remove the effect of rain.

In particular, it is rather easy to observe that state-of-the-art deraining algorithms smooth out the edges of objects in an image, which leads to a loss of critical information and features, which are essential for enabling the detection algorithms to classify and localize objects. The images in the top two rows of Figure 6, representing outputs of Faster R-CNN and YOLO, show some of the objects that are not detected after using the deraining methods but that are successfully detected if rainy images are directly used as input into the detection algorithms.

A related critical issue to highlight for current deraining algorithms is their inability to remove natural raindrops found in realistic scenes captured by moving vehicles. The root cause of this issue is the fact that deraining algorithms have been largely designed and tested using synthetic rain visuals superimposed on the underlying scenes. What aggravates this issue is that, at least in some cases, the background environments employed to design and test deraining algorithms are predominantly static scenes with a minimal number of moving objects. Consequently, the salient differences between such synthetic scenarios and the realistic environment encountered by a vehicle that is moving through natural rain represents a domain mismatch that is too

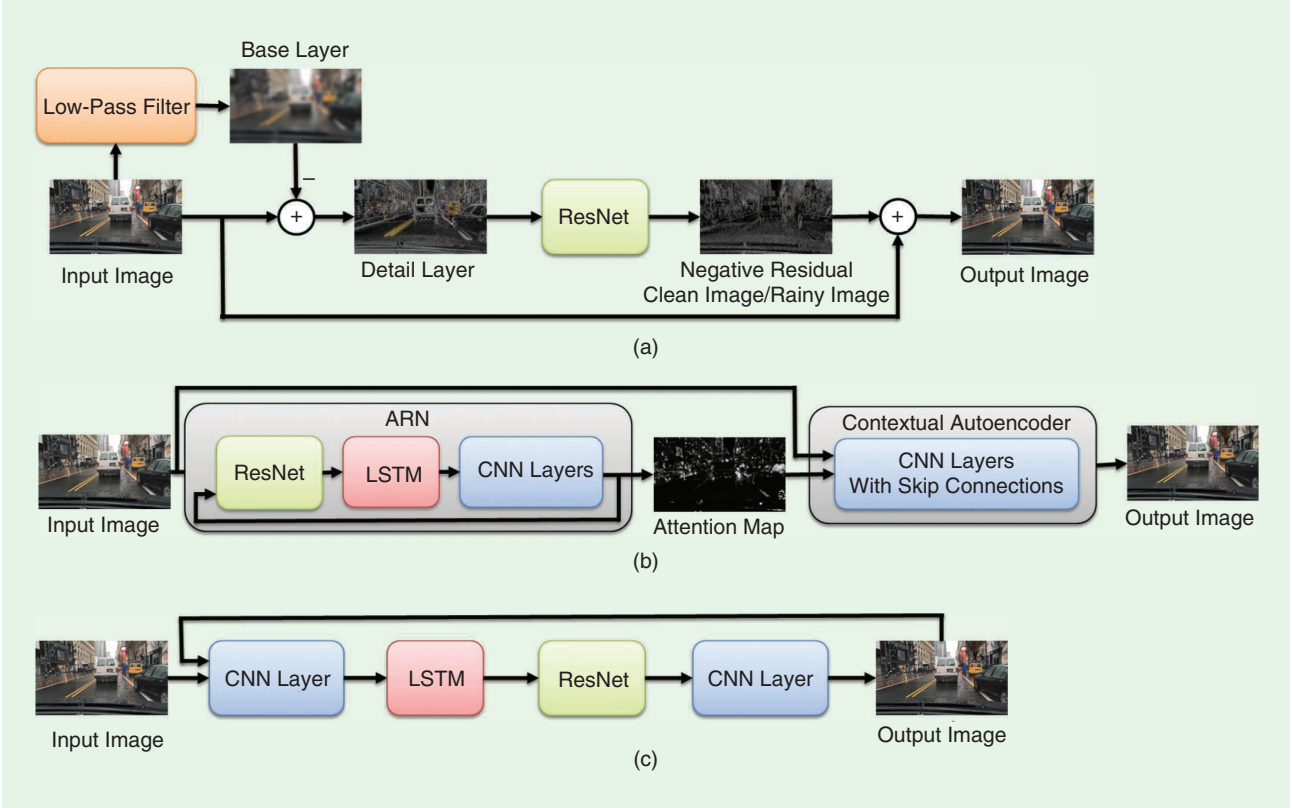


FIGURE 5. The general architectures of the used deraining methods, including (a) the DDN [14], (b) DeRaindrop [15], and (c) progressive recurrent network (PReNet) [16]. LSTM: long short-term memory; ARN: attentive recurrent network.

much to handle for current deraining algorithms, and this leads to the algorithms' failure under realistic conditions for autonomous vehicles. Hence, overall, we believe that relying purely on state-of-the-art deraining solutions does not represent a viable approach for mitigating the impact of rain on object detection. The images in Figure 6(a) and (b), especially some of the cases in the bottom two rows, illustrate examples of the failure of deraining methods to improve the performance of object detection.

Alternative training approaches for deep learning-based object detection

The requirement that autonomous driving systems must work reliably in different weather conditions is at odds with the fact that the training data are usually collected in dry weather with good visibility. Thus, the performance of object detection algorithms degrades in challenging weather conditions, as we showed in the first "Results and Discussion" section.

FIGURE 6. The example detection results for different visual scenes where no deraining methods were employed and where deraining methods (DDN [14], DeRaindrop [15], and PReNet [16]) were used in conjunction with detection methods. Objects were perceived using (a) Faster R-CNN [6] and (b) YOLO [37].

One simple approach to address this problem is to train a given CNN for the detection of objects using images captured in real rainy weather. As we highlighted earlier, sufficient annotated data sets captured by moving vehicles in realistic urban environments in natural rainy conditions are not readily available. To that end, and in spite of the fact that some data sets are available, the very few data sets captured under real rainy conditions are not properly annotated [25]. Having such small data sets inherently makes them inadequate to reliably train deep learning architectures for objection detection. Furthermore, annotating the available data captured in real rainy conditions with accurate bounding boxes is an expensive and time-consuming process.

An alternative approach for addressing the lack-of-real-data issue is to train detection methods using synthetic rain data. However, and as we highlighted earlier, the trained methods generalize poorly on real data due to the domain shift between synthetic and natural rain. To solve this issue, we review approaches that can be employed for training the detection methods using annotated clear data in conjunction with unannotated rainy data. In particular, we review and survey two emerging frameworks for addressing this critical issue: image translation and domain adaptation.

Unsupervised image-to-image translation

Image-to-image translation (I2IT) is a well-known computer vision framework that translates images from one domain (e.g., captured in clear weather) to another domain (e.g., rainy conditions) while preserving the underlying and critical visual content of the input images. In short, I2IT attempts to learn a joint distribution of images in different domains. The learning process can be classified into a supervised setting, where the training data set consists of paired examples of the same scene captured in both domains (e.g., clear and rainy conditions), and an unsupervised setting, where different examples of both domains are used for training; hence, these examples do not have to be taken from the same corresponding scene.

The unsupervised case is inherently more challenging than supervised learning. More importantly, to address the main issue we face in the context of the lack of data needed to train object detection architectures in realistic conditions, we consequently need an unsupervised setting. In particular, the requirement of having a very large set of image pairs, where each pair of images must be of the same scene captured in different domains, renders supervised I2IT solutions virtually useless for our purpose. In fact, this requirement imposes more constraints than the lack-of-data issue that we are already trying to address. Hence, and despite the availability of well-known supervised learning-based techniques in this area [38], [39], we have to resort to unsupervised solutions to address the problem at hand.

Recently, GANs [35] have been achieving very promising performance results in the area of image translation [32], [38]–[41]. In general, a GAN consists of a generator and a discriminator. The generator is trained to fool the discriminator, while the latter attempts to distinguish (or discriminate) between real natural images, on the one hand, and fake images, which are generated by the trained generator, on the other hand. By doing this, GANs align the distribution of translated images with real images in the target domain.

As mentioned earlier, data sets that have paired clear–rainy images in driving environments are not publicly available. As a result, we use unsupervised I2IT (UNIT) [32] to translate clear images to rainy ones since the training process for the UNIT framework does not require paired images of the same scene. In other words, UNIT training requires two independent sets of images, where one consists of images in one domain, and the other includes images in a different domain. The high-level architecture of the UNIT model is shown in Figure 7. First, the encoder network maps an input image to a shared latent code (a shared, compact representation of an image in both domains). Then, the generator network uses the shared latent code to generate an image in the desired domain.

To train the UNIT model that learns the mapping from clear images to rainy ones, we use the train clear set that consists of clear-weather annotated images as the source domain. For the target rainy domain, we extract a sufficiently large number of images from the rainy videos in the BDD100K data set. Subsequently, we apply images in the train clear set to the trained UNIT model to generate rainy images. We refer to the images that are generated by the UNIT model as the *train-gen-rainy* set. Examples of generated rainy images appear in Figure 8.

Eventually, we use the train-gen-rainy data set to train the detection methods. This is followed by using the test rain data set to evaluate the AP performance of the detection methods, which are now trained using the generated rainy set. We also calculate the mAP as we have done for other approaches. Table 2 shows the performance of detection methods that are trained using generated rainy images by the UNIT model.

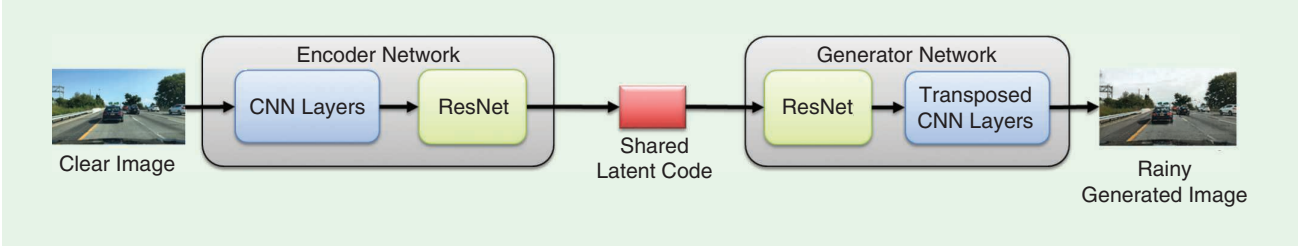


FIGURE 7. The high-level architecture of the UNIT model [25] to generate images.

Domain adaptation

Domain adaptation is another potentially viable framework that could be considered to address the major challenges that we have been highlighting in this tutorial regarding 1) the salient mismatch between the two domains (clear and rainy weather conditions) and 2) the lack of annotated training data captured in rainy conditions. In particular, a domain adaptation framework [33] has been designed and developed specifically for Faster R-CNN due to the fact that it is among the most popular object detection approaches. (At this point, we are not aware of other domain adaptation frameworks that have been designed and developed for YOLO. Consequently, given the tutorial nature of this article, we review only domain adaptation that has already been developed for Faster R-CNN [33].) The framework developed in [33] adapts deep learning-based object detection to a target domain that is different from the training domain, without requiring any annotations in the target domain. In particular, it employs the adversarial training strategy [35] to learn robust features that are domain invariant. In other words, it makes the distribution of features extracted from images in the two domains indistinguishable.

The architecture for the domain-adaptive Faster R-CNN model [33] is shown in Figure 2. There are two levels of domain adaptation that are employed. First, an image-level domain classifier is used. At this level, the global attributes (such as the image style, illumination, and so forth) of the input image are used to distinguish between the source and target domains. Thus, the (global) feature map resulting from the common CNN feature extractor of the Faster R-CNN detector is used as input toward the image-level domain classifier. Second, an instance-level domain classifier is employed. This classifier uses the specific features associated with a particular region to distinguish between the two domains. Hence, the instance-level domain classifier uses the feature vector resulting from

(a) (b)

FIGURE 8. The example images generated by the trained UNIT model. The (a) original clear images and (b) generated rainy images [25].

the FCs at the output of the RoI pooling layer of the Faster R-CNN detector. The two classifiers, the image- and instance-level ones, should naturally agree in terms of their binary classification decision about whether the input image belongs to the source or target domain. Consequently, a consistency regularization stage combines the output of the two classifiers to promote consistency between their outcomes.

While the two domain adaptation classifiers are optimized to differentiate between the source and target domains, the Faster R-CNN detector must be optimized such that it becomes domain-independent or domain invariant. In other words, the Faster R-CNN detector must distinguish objects regardless of the input image domain (clear or rainy). Hence, the feature map resulting from the Faster R-CNN feature extractor must be domain invariant. To that end, this feature extractor should be trained and optimized to maximize the domain classification error achieved by the domain adaptation stage. Thus, while both the image- and instance-level domain classifiers are designed to minimize the binary classification error (between the source and target domains), the Faster R-CNN feature extractor is constructed to maximize the same binary classification error.

The lack of data, and especially annotated data, that captures the truly diverse nature of rainy conditions for moving vehicles is arguably the most critical and fundamental issue in this area.

To achieve these contradictory objectives, a GRL [42] is employed. Thus, the GRL is a bidirectional operator that is used to realize two different optimization objectives. In the feed-forward direction, the GRL acts as an identity operator. This leads to the standard objective of minimizing the classification error when performing local backpropagation within the domain adaptation network. On the other hand, for backpropagation toward the Faster R-CNN network, the GRL becomes a negative scalar. Hence, in this case, it leads to maximizing the binary classification error, and this maximization promotes the generation of a domain invariant feature map by the Faster R-CNN feature extractor.

Consequently, for the purpose of this tutorial, we developed and employed a domain-adaptive Faster R-CNN [33] for rainy conditions. To train this model, we prepare the training data to include two sets: source data, which consists of images captured in clear weather (this set includes data annotations in terms of bounding box coordinates and object categories), and target data, which includes only images captured in rainy conditions without any annotations. To validate the trained model using domain adaptation, we tested it using

FIGURE 9. The (a) ground truth, (b) original training, and (c) alternative training, where the example detection results using alternative training approaches for Faster R-CNN and YOLO are shown. The top-right image depicts the improvement in vehicle and traffic sign detection when images generated by I2IT (the UNIT model [25]) are used to train Faster R-CNN. The middle-right image shows the improvement in pedestrian detection when domain adaptation [33] is employed to train Faster R-CNN. The bottom-right image illustrates the improvement in pedestrian detection when images generated by I2IT (the UNIT model [25]) are harnessed to train YOLO.

the test rainy set. The performance of the detection method (Faster R-CNN) that is trained by the domain adaptation approach is shown in the bottom row of Table 2.

Discussion

Based on the results in Table 2, we observe that while deraining algorithms degrade the AP performance when tested on scenes distorted by natural rain, improvements can be achieved when employing I2IT and domain adaptation as mitigating techniques. Different cases are presented in Figure 9. In terms of the AP, and as an example, rainy conditions degrade the pedestrian detection capabilities for YOLO by more than 5% (from approximately 37% to 32%), but by using image translation, the performance improves to an AP of more than 34%, consequently narrowing the gap between the clear- and rainy-condition performances. Similarly, both image translation and domain adaptation improve the traffic sign detection performance for Faster R-CNN. Furthermore, image translation seems to improve the vehicle detection performance under Faster R-CNN.

In other cases, for example, the traffic light detection performance under Faster R-CNN, the domain adaptation and image translation do not seem to perform well when tested on natural rainy images (even when using natural rainy images as the target domain for training these techniques). One potential factor for this poor performance in some of these cases is the fact that small objects, such as traffic lights, are quite challenging to detect to start with. This can be seen from the very low AP value, even in clear conditions, which is a mere 26%. Naturally, the impact of raindrops or rain streaks on such small objects in the scene could be quite severe, to the extent that a mitigating technique might not be able to recover the salient features of these objects.

In summary, employing domain adaptation or generating rainy weather visuals using UNIT translation, and then using these visuals for training, seems to narrow the gap in performance due to the domain mismatch between clear and rainy weather conditions. This promising observation becomes especially clear when considering the disappointing performance of deraining algorithms. Nevertheless, it is also evident that there is still much room for improvement toward reaching the same level of performance in clear conditions. There are key challenges that need to be addressed, though, when designing any new mitigating techniques for closing the aforementioned gap. These challenges include the broad and diverse scenarios for rainy conditions, especially in driving environments.

Generative models could still play a crucial role in training object detection methods to be more robust and resilient in challenging conditions.

These diverse cases and scenarios can't be learned in a viable way by using state-of-the-art approaches. For example, raindrops have a wide range of possible appearances, and they come with various sizes and shapes, especially when falling on the windshield of a vehicle. Another factor is the influence of windshield wipers on altering the amount of rain, and even the shapes and sizes of raindrops, between wipe cycles. Other external factors include reflections from the surrounding wet pavement, mist in the air, and splash effects. Current state-of-the-art image translation techniques and domain adaptation are not robust enough to capture this wide variety of rain effects. Figure 10 provides images from the test rainy set that illustrate several rainy weather scenarios and effects for driving vehicles.

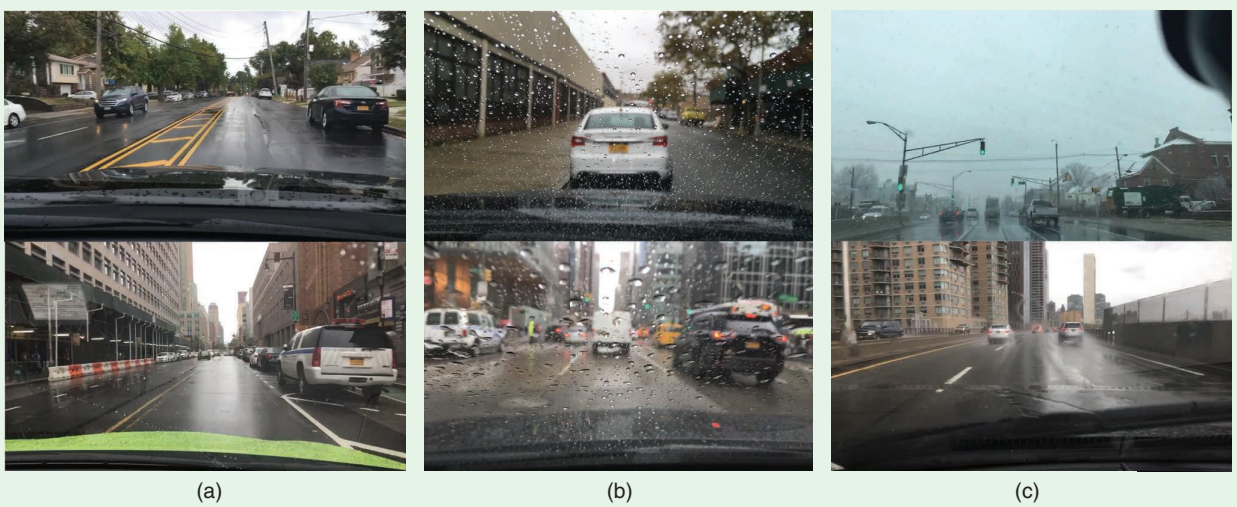


FIGURE 10. The images from the test rainy set that illustrate several rainy weather scenarios and effects for driving vehicles, with (a) no raindrops, (b) various raindrops, and (c) mist and pavement reflection [25].

Conclusions

Besides outlining state-of-the-art frameworks for object detection, deraining, I2IT, and domain adaptation, this tutorial highlighted crucial results and conclusions regarding current methods in terms of their performance in rainy weather conditions. In particular, we believe there is an overarching consistent message regarding the limitations of the surveyed techniques in handling and mitigating the impact of rain for visuals captured by moving vehicles. This consistent observation has serious implications for autonomous vehicles since the aforementioned limitations impact autonomous vehicles' core safety capabilities. To address these issues, we recap some of our key findings and point out potential directions.

There is a need for novel deep learning architectures and solutions that have adequate capacity for handling object detection under diverse conditions.

1) The lack of data, and especially annotated data, that captures the truly diverse nature of rainy conditions for moving vehicles is arguably the most critical and fundamental issue in this area. Major industry players are becoming more willing to tackle this problem and more open about addressing this issue publicly. Consequently, a few related efforts have just been announced and actually commenced by high-tech companies. These efforts are specifically dedicated to operating fleets of autonomous vehicles in challenging and diverse rainy weather conditions, explicitly for the sake of collecting data under these conditions [30]. After years of testing and millions of driven miles conducted primarily in favorable and clear weather, there is a salient admittance and willingness to divert important resources toward data collection in challenging weather conditions that will be encountered by autonomous vehicles.
2) Despite the recent efforts to collect more diverse data, we believe that generative models could still play a crucial role in training object detection methods to be more robust and resilient in challenging conditions. In particular, we believe that novel and more advanced frameworks for UNIT could play a viable role for generating meaningful data for training. Due to the fact that these frameworks do not require annotated data, their underlying generative models could be useful in many ways. First, they could fill the gap that currently exists in terms of the lack of real annotated data in different weather conditions; hence, progress in terms of training and testing new object detection methods could be achieved by using these generative models.

   Second, even after a reasonable amount of annotated data captured in natural rainy conditions becomes available, the generative models could still play a pivotal role in both the basic training and coverage of diverse scenarios. In other words, UNIT models could always generate more data that can compliment real data, and this, on its own, could be quite helpful to further the basic training of object detection methods. Furthermore, despite the number of various rainy condition scenarios that real data actually represent, there will always be a need for capturing certain scenarios that are not included in a real data set. In that context, generative models could be used to produce data representing the scenarios that are missing from the real data sets, and hence they could increase the coverage and diversity of the cases that object detection methods can handle.
3) There is a need for novel deep learning architectures and solutions that have adequate capacity for handling object detection under diverse conditions. Designing a neural network that performs quite well in one domain yet degrades in others is not a viable strategy for autonomous vehicles. In general, training the leading object detection architectures through a diverse set of data does not necessarily improve the performance of these architectures relative to their results when trained on a narrow domain of cases and scenarios. We believe that this issue represents an opportunity for researchers in the field to make key contributions.

Acknowledgments

This work has been supported by the Ford Motor Company under the Ford–Michigan State University Alliance Program. We would like to acknowledge Ford's advanced research engineers, Jonathan Diedrich and Mark Gehrke, for their invaluable input and contributions throughout the collaborative effort that resulted in this article.

Authors

Mazin Hnewa (mazin@msu.edu) received his B.S. degree in computer engineering from the University of Baghdad, Iraq, in 2005 and his M.S. degree in electrical engineering from Michigan State University (MSU), East Lansing, in 2012. He was a faculty member at the University of Karbala, Iraq, in 2013–2017 and is currently working toward his Ph.D. degree at MSU, East Lansing, Michigan, USA. His research interests include object detection in adverse weather conditions, deep learning, domain adaptation, and autonomous vehicles.

Hayder Radha (radha@egr.msu.edu) received his Ph.M. and Ph.D. degrees from Columbia University, New York, in 1991 and 1993, respectively. He is currently the Michigan State University (MSU) Foundation Professor and director of the Connected and Autonomous Networked Vehicles for Active Safety program at MSU, East Lansing, Michigan, USA. He was a fellow at Philips Research (1996–2000) and a distinguished member of the technical staff at Bell Laboratories (1986–1996). He received the Amazon Research Award (2019), Semiconductor Research Consortium Award (2019), Google Faculty Research Award (2014 and 2015), Microsoft Research Award, AT&T Bell Labs Ambassador and AT&T Circle-of-Excellence Awards, and William J. Beal Outstanding Faculty Award (2015). He is a Fellow of IEEE.

References

[1] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Computer Vision*, 2015, pp. 2722–2730. doi: 10.1109/ICCV.2015.312.

[2] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops*, 2017, pp. 129–137.

[3] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *Proc. 2018 IEEE Intelligent Vehicles Symp. (IV)*, pp. 1013–1020. doi: 10.1109/IVS.2018.8500504.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2014, pp. 580–587. doi: 10.1109/CVPR.2014.81.

[5] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Computer Vision*, 2015, pp. 1440–1448. doi: 10.1109/ICCV.2015.169.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, June 2017. doi: 10.1109/TPAMI.2016.2577031.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Oct 2017, pp. 2999–3007.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. European Conf. Computer Vision*, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[10] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Advances Neural Information Processing Systems*, 2016, pp. 379–387.

[11] K. Garg and S. K. Nayar, "Detection and removal of rain from videos," in *Proc. 2004 IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, vol. 1, p. I. 10.1109/CVPR.2004.1315077.

[12] L. Kang, C. Lin, and Y. Fu, "Automatic single-image-based rain streaks removal via image decomposition," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1742–1755, Apr. 2012. doi: 10.1109/TIP.2011.2179057.

[13] W. Yang, R. T. Tan, J. Feng, J. Liu, Z. Guo, and S. Yan, "Deep joint rain detection and removal from a single image," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 1357–1366. doi: 10.1109/CVPR.2017.183.

[14] X. Fu, J. Huang, D. Zeng, Y. Huang, X. Ding, and J. Paisley, "Removing rain from single images via a deep detail network," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 3855–3863. doi: 10.1109/CVPR.2017.186.

[15] R. Qian, R. T. Tan, W. Yang, J. Su, and J. Liu, "Attentive generative adversarial network for raindrop removal from a single image," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2018, pp. 2482–2491. doi: 10.1109/CVPR.2018.00263.

[16] D. Ren, W. Zuo, Q. Hu, P. Zhu, and D. Meng, "Progressive image deraining networks: A better and simpler baseline," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019, pp. 3937–3946. doi: 10.1109/CVPR.2019.00406.

[17] T. Wang, X. Yang, K. Xu, S. Chen, Q. Zhang, and R. W. Lau, "Spatial attentive single-image deraining with a high quality real rain dataset," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019, pp. 12270–12279. doi: 10.1109/CVPR.2019.01255.

[18] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004. doi: 10.1109/TIP.2003.819861.

[19] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. European Conf. Computer Vision*, 2014, pp. 818–833. doi: 10.1007/978-3-319-10590-1_53.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learning Representations, ICLR 2015*.

[21] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, 2009. doi: 10.1109/TPAMI.2009.167.

[22] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 1–8. doi: 10.1109/CVPR.2012.6248074.

[23] M. Cordts et al., "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1–18. doi: 10.1109/CVPR.2016.350.

[24] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *Proc. Int. Conf. Computer Vision (ICCV)*, pp. 4990–4999, 2017. [Online]. Available: https://www.mapillary.com/dataset/vistas

[25] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, Bdd100k: A diverse driving video database with scalable annotation tooling. 2018. [Online]. Available: arXiv:1805.04687

[26] H. Caesar et al., Nuscenes: A multimodal dataset for autonomous driving. 2019. [Online]. Available: arXiv:1903.11027

[27] P. Rousseau, V. Jolivet, and D. Ghazanfarpour, "Realistic real-time rain rendering," *Comput. Graph.*, vol. 30, no. 4, pp. 507–518, 2006. doi: 10.1016/j.cag.2006.03.013.

[28] K. Garg and S. K. Nayar, "Photorealistic rendering of rain streaks," *ACM Trans. Graph. (TOG)*, vol. 25, no. 3, pp. 996–1002, 2006. doi: 10.1145/1141911.1141985.

[29] Adobe Inc., Sab Jose, CA. Cycore Rainfall Simulation, Adobe after Effects CC 2019. 2019. [Online]. Available: https://www.adobe.com/products/aftereffects.html

[30] K. Korosec, Waymo self-driving cars head to Florida for rainy season. Aug. 20, 2019. [Online]. Available: https://techcrunch.com/2019/08/20/waymo-self-driving-cars-head-to-florida-for-rainy-season/

[31] M. Everingham and J. Winn, "The PASCAL visual object classes challenge 2012 development kit," in *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, 2011. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/devkit_doc.pdf

[32] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Proc. Advances Neural Information Processing Systems*, 2017, pp. 700–708.

[33] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, "Domain adaptive faster R-CNN for object detection in the wild," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2018, pp. 3339–3348. doi: 10.1109/CVPR.2018.00352.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Advances Neural Information Processing Systems*, 2014, pp. 2672–2680.

[36] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Advances Neural Information Processing Systems*, 2015, pp. 802–810.

[37] J. Redmon and A. Farhadi, Yolov3: An incremental improvement. 2018. [Online]. Available: arXiv:1804.02767

[38] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 1125–1134. doi: 10.1109/CVPR.2017.632.

[39] C. Ledig et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 4681–4690. doi: 10.1109/CVPR.2017.19.

[40] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Computer Vision*, 2017, pp. 2223–2232. doi: 10.1109/ICCV.2017.244.

[41] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *Proc. Advances Neural Information Processing Systems*, 2017, pp. 465–476.

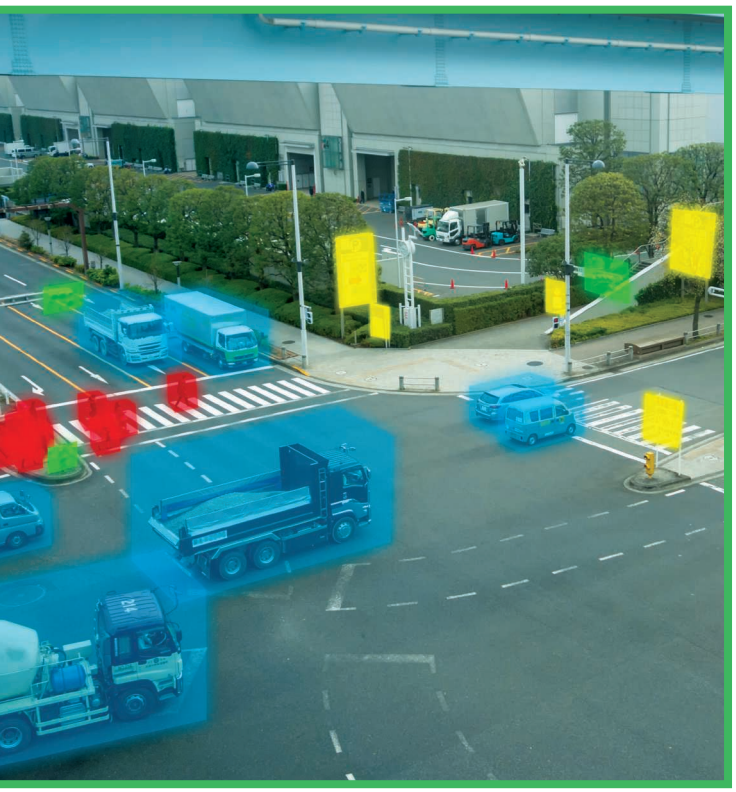
[42] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. 32nd Int. Conf. Machine Learning*, 2015, vol. 37, pp. 1180–1189.

SP

Siheng Chen, Baoan Liu, Chen Feng,
Carlos Vallespi-Gonzalez, and Carl Wellington

3D Point Cloud Processing and Learning for Autonomous Driving

Impacting map creation, localization, and perception



©SHUTTERSTOCK.COM/MONOPOLY919

Digital Object Identifier 10.1109/MSP.2020.2984780
Date of current version: 24 December 2020

We present a review of 3D point cloud processing and learning for autonomous driving. As one of the most important sensors in autonomous vehicles (AVs), lidar sensors collect 3D point clouds that precisely record the external surfaces of objects and scenes. The tools for 3D point cloud processing and learning are critical to the map creation, localization, and perception modules in an AV. Although much attention has been paid to data collected from cameras, such as images and videos, an increasing number of researchers have recognized the importance and significance of lidar in autonomous driving and have proposed processing and learning algorithms that exploit 3D point clouds. We review the recent progress in this research area and summarize what has been tried and what is needed for practical and safe AVs. We also offer perspectives on open issues that are needed to be solved in the future.

Introduction and motivation

As one of the most exciting engineering projects of the modern world, autonomous driving is an aspiration for many researchers and engineers across generations. It is a goal that might fundamentally redefine the future of human society and everyone's daily life. Once autonomous driving becomes mature, we will witness a transformation of public transportation, infrastructure, and the appearance of our cities. The world is looking forward to exploiting autonomous driving to reduce traffic accidents caused by driver errors, save drivers' time, and liberate the workforce, as well as to save parking spaces, especially in urban areas [1].

Autonomous driving: History and current state

It has taken decades of effort to get closer to the goal of autonomous driving. From the 1980s through the DARPA Grand Challenge in 2004 and the DARPA Urban Challenge in 2007, the research on autonomous driving was primarily conducted in the United States and Europe, yielding incremental progresses in driving competence in various situations [2]. In 2009, Google started a research project on self-driving cars and later created Waymo to commercialize the accomplishment based on their

 1053-5888/21©2021IEEE

early technical success. Around 2013–2014, the rise of deep neural networks brought on the revolution of practical computer vision and machine learning. This emergence made people believe that many technical bottlenecks of autonomous driving could be fundamentally solved. In 2015, Uber created the Uber Advanced Technologies Group with the aim to enable AVs to complete scalable ride-sharing services. This goal has become a common deployment strategy within the industry.

SAE International, a transportation standards organization previously known as the Society of Automotive Engineers, introduced the J3016 standard, which defines six levels of driving automation; visit https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic for more information. The standard ranges from SAE L0 (no automation) to SAE L5 (full automation). One turning point occurs between L2 and L3, where the driving responsibility shifts from a human driver to an autonomous system, and another turning point occurs between L3 and L4, where the human no longer drives under any circumstances. Currently, there are numerous high-tech companies, automobile manufacturers, and start-up companies working on autonomous-driving technologies, including Apple, Aptiv, Argo AI, Aurora, Baidu, General Motors Cruise, Didi, Lyft, Pony.ai, Tesla, Zoox, the major automobile companies, and many others [3]. These companies have ambitious goals to achieve SAE L4 in the near future.

Although there has been significant progress across many groups in industry and academia, there is still much work to be done. Efforts from both industry and academia are needed to achieve autonomous driving. Recently, there have been many discussions and hypotheses about the progress and future of autonomous driving; however, few thoughts from those who push industrial-level self-driving technologies from the frontline are publicly accessible. In this article, we provide a unifying perspective from both practitioners and researchers.

In industry, an autonomous system usually includes a series of modules with complicated internal dependencies. Most modules are still far from being perfect due to a number of technical bottlenecks and the long-tail issues [4]. Additionally, a small error from one module can cause problems in subsequent modules and potentially result in a substantial failure at the system level. There has been some initial research on end-to-end systems where the entire system is trained end to end and information can flow from sensors directly to the final planning or control decisions. These systems offer the promise to reduce internal dependency challenges; however, these systems often lack explainability and are difficult to analyze. Although significant progress has been made, there remain many open challenges in designing a practical autonomous system that can achieve the goal of full self-driving.

A tour of an autonomous system

An autonomous system typically includes the sensing, map-creation, localization, perception, prediction, routing, planning, and control modules [5] (see Figure 1). A high-definition (HD) map is created offline. At runtime, the online system is given a destination. The system then senses its environment, localizes itself to the map, perceives the world around it, and makes corresponding predictions of future motion for these objects. The motion planner uses these predictions to plan a safe trajectory for an AV to follow the route to the destination executed by the controller.

Sensing module

To ensure reliability, autonomous driving usually requires multiple types of sensors. Cameras, radar, lidar, and ultrasonic sensors are most commonly used. Among those sensors, lidar is particularly interesting because it directly provides a precise 3D representation of a scene. Although the techniques for

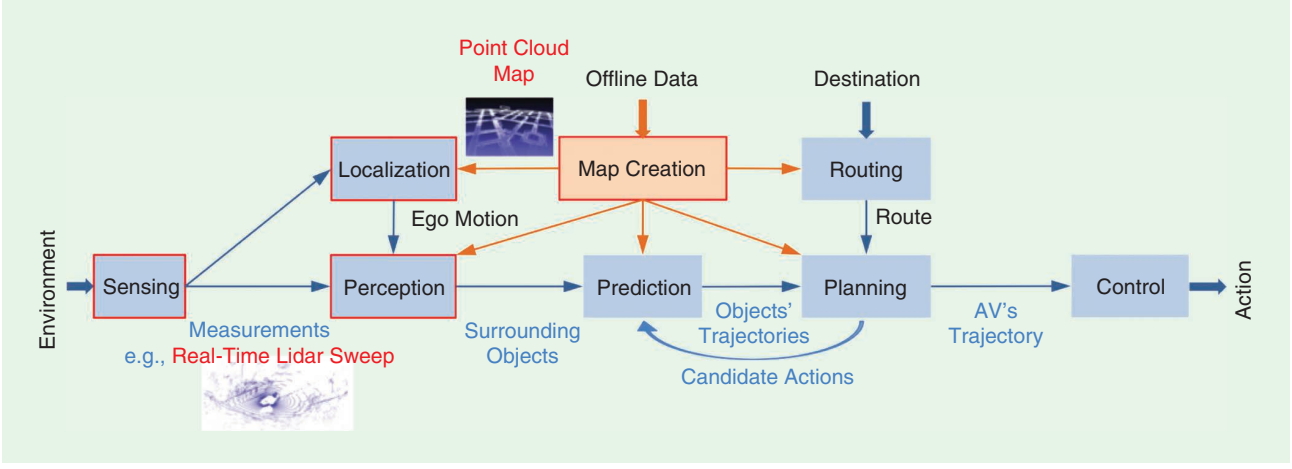


FIGURE 1. The high-level architecture of a typical autonomous system. An HD map is built offline. At runtime, the online system is given a destination. The system then senses its environment, localizes itself to the map, perceives the world around it, and makes corresponding predictions of the future motion for these objects. The motion planner uses these predictions to plan a safe trajectory for an AV to follow the route to the destination, which is executed by the controller. Note that two types of 3D point clouds are used in this autonomous system: a point cloud map, created by the map creation module and consumed by the localization module, and a real-time lidar sweep, collected by the sensing module and consumed by the localization and perception modules.

3D reconstruction and depth estimation based on 2D images have been significantly improved with the development of deep learning-based computer vision algorithms, the resulting estimations are still not always precise or reliable. Besides algorithmic constraints, fundamental bottlenecks also include inherent exponential range error growth in depth estimation, poor performance in low light, and the high computational cost of processing high-resolution images. On the other hand, lidar measures 3D information through direct physical sensing. A real-time lidar sweep consists of a large number of 3D points, called a *3D point cloud*. (The measurements from radar and ultrasound are also called 3D point clouds, but our focus is on the 3D point clouds collected by lidar.) Each 3D point records the range from the lidar to an object's external surface, which can be transformed into the precise 3D coordinate. These 3D point clouds are extremely valuable for an AV to localize itself and detect surrounding objects in the 3D world.

The vast majority of companies and researchers rely heavily on lidar to build a reliable AV [6]. This is why we believe that advanced techniques for 3D point cloud processing and learning are indispensable for autonomous driving.

Map creation module

Map creation is the task of creating an HD map, a precise heterogeneous map representation of the static 3D environment and its traffic rules. An HD map usually contains two map layers: a point cloud map, representing 3D geometric information of the surroundings, and a traffic rule-related semantic feature map, containing road boundaries, traffic lanes, traffic signs, traffic lights, and so on. These two map layers are aligned together in the 3D space and provide detailed navigation information. As one map layer, the point cloud map is a dense 3D point cloud and is mainly used for providing localization prior. Different from common maps designed for humans, an HD map is designed for AVs. The map creation module is crucial because an HD map provides valuable prior environmental information, as discussed in the "3D Point Cloud Processing for HD Map Creation" section.

Localization module

Localization is the task of finding the ego position of an AV relative to a reference position in the HD map. This module is crucial because an AV must localize itself to use the correct lane and other important priors in the HD map. One of the core techniques is 3D point cloud registration; that is, estimating the precise location of an AV by matching real-time lidar sweeps to the offline HD map, as mentioned in the "3D Point Cloud Processing for HD Map Creation" section.

Perception

Perception is the task of perceiving the surrounding environment and extracting information related to navigation. This module is crucial because the perception module is the visual system of an AV, which should detect, track, and classify objects in the 3D scene. It used to be considered the technical bottleneck of autonomous driving. Recently, with large-scale training data and the development of advanced machine learning algorithms, the overall performance of the perception module has improved tremendously. Some core techniques include 2D and 3D object detection. 2D object detection becomes relatively mature, while 3D object detection is based on real-time lidar sweeps and becomes an increasingly hot research topic, as discussed in the "3D Point Cloud Processing for Localization" section.

Prediction

Prediction is the task of predicting the future potential trajectories of each object in the 3D scene. This module is crucial because an AV must know the possible future behaviors of nearby objects to plan a safe trajectory.

Routing

Routing is the task of designing a high-level path from the AV's starting position to its destination. The output of this module provides a high-level guideline for the planning module.

Planning

Motion planning is the task of designing a trajectory for an AV based on the state of current cars and its surrounding environment and destination. This module is crucial because an AV must know how to react to the surrounding environment.

Control

Control is the task of executing the commands from the planning module. It takes charge of controlling the actuators of the steering wheel, throttle, and brakes.

Overview of 3D point cloud processing and learning

As mentioned previously, lidar provides indispensable 3D information for autonomous driving. In the following sections, we discuss the processing and learning techniques that convert raw lidar measurements into useful information.

Usages in autonomous driving

Two types of 3D point clouds are commonly used in an AV: a real-time lidar sweep and a point cloud map, which is one layer in the HD map (see Figure 1). A point cloud map provides previous environmental information: the localization module uses a point cloud map as a reference in 3D point cloud registration to determine the position of the AV, and the perception module uses a point cloud map to help split the foreground and background. On the other hand, real-time lidar sweeps are consumed by the localization module to register against the point cloud map and by the perception module to detect surrounding objects in the 3D scene. Therefore, 3D point cloud processing and learning are critical to build the map creation, localization, and perception modules in an autonomous system.

Recent progress in academia

Sensors capture data and data feeds algorithms. During the development of radar, acoustic sensors, and communication

systems, 1D signal processing experienced a rapid growth during the past century, leading to a revolutionary impact on digital communication systems. With the popularization of cameras and televisions, 2D image processing experienced a rapid growth during the past 30 years, resulting in a significant change to photography, entertainment, and surveillance. Due to the increasing needs from industrial robotics, autonomous driving, and augmented reality, 3D sensing techniques is experiencing rapid development recently. At the same time, the algorithms used to process and learn from 3D point clouds are starting to receive much attention in academia. The discussion in the following sections is divided into two parts: 3D point cloud processing, which handles 3D point clouds from a signal processing perspective, and 3D point cloud learning, which handles 3D point clouds from a machine learning perspective.

3D point cloud processing

3D point cloud processing is the analyzing and modifying of a 3D point cloud to optimize its transmission, storage, and quality through various mathematical and computational algorithms. Even though the processing algorithms could be significantly different, many processing tasks are naturally extended from 1D signal processing and 2D image processing. For example, 3D point cloud compression is the 3D counterpart of image compression and aims to reduce the cost for storage or transmission of a 3D point cloud; 3D point cloud denoising is the 3D counterpart of image denoising and seeks to remove noise from a 3D point cloud; 3D point cloud registration is the 3D counterpart of image registration and attempts to align two or more 3D point clouds of the same scene; and 3D point cloud downsampling and upsampling are the 3D counterpart of image scaling and look to change the resolution (number of points) in a 3D point cloud.

3D point cloud learning

3D point cloud learning is the process of interpreting and understanding a 3D point cloud. Using the powerful tools of deep neural networks, computer vision researchers aim to extend the success from images and videos to 3D point clouds. Two primary learning problems are 3D point cloud recognition and segmentation. Similar to the cases for 2D images, 3D point cloud recognition aims to classify a given 3D point cloud into a predefined class category and 3D point cloud segmentation aims to partition a given 3D point cloud into multiple segments. Due to the irregular format of 3D point clouds, one of the biggest challenges for designing a learning algorithm is to formulate efficient data structures to represent 3D point clouds. Some algorithms transform 3D point clouds to regular 3D voxels so that 3D convolutions can be used for analysis; however, they must make a tradeoff between resolution and memory. To handle raw point clouds directly, PointNet [7] uses pointwise multilayer perceptrons (MLPs) and max-pooling to ensure the permutation invariance. After that, a series of 3D deep learning methods follow PointNet as their base networks.

Relationship between academia and industry

The technical transition from 1D time series to 2D images is quite natural because both types of data are supported on regular-spacing structures; however, the technical transition from 2D images to 3D point clouds is not straightforward because those points are irregularly scattered in a 3D space. Numerous popular methods used to handle 3D point clouds are proposed heuristically by practitioners. Therefore, there is substantial room for researchers and practitioners to collaborate and solve fundamental tasks on 3D point cloud processing and learning so that we can accelerate the progress of autonomous driving.

Key ingredients of 3D point cloud processing and learning

In this section, we introduce the basic tools of 3D point cloud processing and learning. We start with the key properties of 3D point clouds. Next, we evaluate some options for representing a 3D point cloud. Finally, we review a series of popular tools used to handle 3D point clouds. These tools have received great attention in academia. Even though some of them may not be directly applied to an autonomous system, it is still worth mentioning because they could inspire new techniques, which are potentially useful to autonomous driving. As discussed in the "Representative Tools" section, we consider two typical types of 3D point clouds in autonomous driving: real-time lidar sweeps and point cloud maps.

Properties

Real-time lidar sweeps

Because of the sensing mechanism, for each 3D point in a real-time lidar sweep, we can trace its associated laser beam and captured time stamp. One real-time lidar sweep can naturally be organized on a 2D image, whose x-axis is the time stamp and y-axis is the laser ID. We thus consider each individual real-time lidar sweep as an organized 3D point cloud. For example, a Velodyne HDL-64E has 64 separate lasers, and each laser fires thousands of times per second to capture a 360° field of view. We thus obtain a set of 3D points associated with 64 elevation angles and thousands of azimuth angles. In a real-time lidar sweep, the vertical resolution is usually much lower than that of the horizontal resolution. Each collected 3D point is associated with a range measurement, an intensity value, and a high-precision GPS time stamp. Note that for a global-shutter image, the pixel values are collected by a charge-coupled device at the same time; however, for a real-time lidar sweep, the 3D points are collected at various time stamps. For the same laser, firings happen sequentially to collect 3D points. For different lasers, firings are not synchronized either; thus, the collected 3D points are not perfectly aligned on a 2D regular lattice.

Because the arrangement of 64 lasers follows a regular angular spacing, the point density of a real-time lidar sweep changes over the range; that is, we collect many more 3D points from nearby objects than from faraway objects. Moreover, a real-time lidar sweep naturally suffers from the occlusion; that

is, we obtain 3D points only from the sides of objects facing the lidar. To summarize, some key properties of a real-time lidar sweep include:

- *Pseudo 3D*: A real-time lidar sweep arranges 3D points approximately on a 2D lattice. Due to the imperfect synchronization, 3D points are not perfectly aligned on a 2D lattice. Meanwhile, unlike a 3D point cloud obtained from multiple views, a real-time lidar sweep reflects only a specific view; we thus consider its dimension to be pseudo 3D.
- *Occlusion*: Each individual real-time lidar sweep records the 3D environment from nearly a single viewpoint. change slightly. A front object would occlude the other objects behind it.
- *Sparse point clouds*: Compared to a 2D image, a real-time lidar sweep is usually sparse representations of objects, especially for faraway objects. It cannot provide detailed the 3D shape information of objects.

Point cloud maps

To create a point cloud map, one needs to aggregate real-time lidar sweeps scanned from multiple AVs across time. Because there is no straightforward way to organize a point cloud map, we consider it an unorganized 3D point cloud. For example, for a 200×200-m^2 portion of an HD map, one needs to aggregate the lidar sweeps around that area for 5–10 trials, leading to more than 10 million 3D points. Because lidar sweeps could be collected from significantly different views, an HD map after aggregation gets denser and presents a detailed 3D shape information. To summarize, some key properties of a point cloud map include:

- *Full 3D*: A point cloud map aggregates multiple lidar sweeps from various views, which is similar to 3D data collected from scanning an object on a turntable. A point cloud map captures information on more objects' surfaces, providing a denser and more detailed 3D representation.
- *Irregularity*: 3D points in a point cloud map are irregularly scattered in the 3D space. They come from multiple lidar sweeps and lose the laser ID association, causing an unorganized 3D point cloud.
- *No occlusion*: A point cloud map is an aggregation of 3D points collected from multiple viewpoints. It depicts the static 3D scene with much less occlusion.
- *Dense point clouds*: A point cloud map provides a dense point cloud, which contains detailed 3D shape information, such as high-resolution shapes and the surface normals.
- *Semantic meanings*: As another layer in the HD map, a traffic rule-related semantic feature map contains the semantic labels of a 3D scene, including road surfaces, buildings, and trees. Because a traffic rule-related semantic feature map and a point cloud map are aligned in the 3D space, we can trace the semantic meaning of each 3D point. For example, 3D points labeled as trees in a point cloud map would help improve perception as lidar points on leaves of trees are usually noisy and difficult to recognize.

Matrix representations

Representations have always been at the heart of most signal processing and machine learning techniques. A good representation lays the foundation to uncover hidden patterns and structures within data and is beneficial for subsequent tasks. A general representation of a 3D point cloud is through a set, which ignores any order of 3D points. Let $\mathcal{S} = \{(\mathbf{p}_i, \mathbf{a}_i)\}_{i=1}^{N}$ be a set of N 3D points, whose ith element $\mathbf{p}_i = [x_i, y_i, z_i] \in \mathbb{R}^3$ represents the 3D coordinate of the ith point and $\mathbf{a}_i$ represents other attributes of the ith point. A real-time lidar sweep usually includes the intensity $\mathbf{a}_i = r_i \in \mathbb{R}$ and a point cloud map usually includes surface normals $\mathbf{n}_i \in \mathbb{R}^3$; thus, $\mathbf{a}_i = [r_i, \mathbf{n}_i] \in \mathbb{R}^4$. For generality, we consider the feature of the ith point as $\mathbf{x}_i = (\mathbf{p}_i, \mathbf{a}_i) \in \mathbb{R}^d$.

For efficient storage and scientific computation, a matrix (or tensor) representation is appealing. Let f be the mapping from a set of 3D points $\mathcal{S}$ to a matrix (or tensor) X with a pending shape. A matrix representation of a 3D point cloud is thus $\mathrm{X} = f(\mathcal{S})$. Next we discuss a few typical approaches used to implement the mapping $f(\cdot)$.

Raw points

The most straightforward matrix representation of a 3D point cloud is to list each 3D point in the set $\mathcal{S}$ as one row in the matrix. Consider

$$\mathrm{X}^{(\mathrm{raw})} = [\mathbf{x}_1 \;\; \mathbf{x}_2 \;\; \ldots \;\; \mathbf{x}_N]^T \in \mathbb{R}^{N \times d}, \tag{1}$$

whose ith row $\mathrm{X}_i^{(\mathrm{raw})} = \mathbf{x}_i \in \mathbb{R}^d$ is the features of ith point in the 3D point cloud.

The advantages of raw-point-based representation are that 1) it is simple and general and 2) it preserves all of the information in the original set of 3D points; however, its shortcoming is that it does not explore any geometric property of 3D points. This representation is generally used in the map and the localization module of an autonomous system, where high precision is needed.

3D voxelization

To enjoy the success of 2D image processing and computer vision, we can discretize the 3D space into voxels and use a series of voxels to represent a 3D point cloud. A straightforward discretization is used to partition the 3D space into equally spaced nonoverlapping voxels from each of three dimensions, as presented in Figure 2(a). Let a 3D space with range H, W, and D along the x-, y-, and z-axes, respectively. Each voxel is of size h, w, and d, respectively. The (i, j, k)th voxel represents a 3D voxel space, $\mathcal{V}_{i,j,k} = \{(x, y, z) | (i-1)h \le x < ih,\ (j-1)w \le y < jw,\ (k-1)d \le z < kd\}$. We then use a three-mode tensor to represent this 3D point cloud. Let $\mathrm{X}^{(\mathrm{VOX})} \in \mathbb{R}^{H \times W \times D}$, whose (i, j, k)th element is

$$\mathrm{X}_{i,j,k}^{(\mathrm{vox})} = \begin{cases} 1, & \text{when } \mathcal{V}_{i,j,k} \cap \mathcal{S} \neq \varnothing; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The tensor $\mathrm{X}^{(\mathrm{vox})}$ records the voxel occupancy.

The advantages of the 3D voxelization-based representation are that 1) the resulting voxels are associated with a natural hierarchical structure and all of the voxels have a uniform spatial size and 2) we can use off-the-shelf tools such as 3D convolutions to analyze data; however, its shortcomings are that 1) it does not consider specific properties of organized 3D point clouds, 2) it usually leads to an extremely sparse representation where most voxels are empty, and 3) it involves a serious tradeoff between the resolution and the memory. This representation can be used in the perception module of autonomous driving as well as the storage of 3D point clouds.

Range view

As discussed in the "Properties" section, a real-time lidar sweep is essentially a series of range measurements from a single location with a certain angular field of view [see Figure 2(b)]. We can approximately organize the 3D points in a real-time lidar to a 2D range-view image. Each pixel in the range-view image corresponds to a frustum in the 3D space. The pixel value is the range from the lidar to the closest 3D point inside the frustum. Specifically, we partition the 3D space along the azimuth angle $\alpha \in [0, 2\pi)$ and the elevation angle $\theta \in (-\pi/2, \pi/2]$ with the resolution of azimuth angle α_0 and the resolution of elevation angle θ_0. The (i, j)th pixel corresponds to a frustum space, $\mathcal{V}_{i,j} = \{(x, y, z) | \alpha_0(i-1) \leq a\cos(x/\sqrt{x^2+y^2}) < \alpha_0 i,\ \theta_0(j-1) \leq a\tan(z/\sqrt{x^2+y^2}) + \pi/2 < \theta_0 j\}$. We then use a 2D matrix to represent a 3D point cloud. Let $\mathrm{X}^{(\mathrm{FV})} \in \mathbb{R}^{H \times W}$, whose (i, j)th element is

$$X_{i,j}^{(\mathrm{FV})} = \begin{cases} \min_{(x,y,z) \in \mathcal{V}_{i,j} \cap \mathcal{S}} \sqrt{x^2+y^2+z^2}, & \mathcal{V}_{i,j,k} \cap \mathcal{S} \neq \varnothing; \\ -1, & \text{otherwise.} \end{cases} \tag{3}$$

We consider the smallest range value in each frustum space. When no point falls into the frustum space, we set a default value as –1. Note that the range-view-based representation could also use nonuniform-spaced elevation angles according to the lidar setting.

The advantages of the range-view-based representation are that 1) it naturally models how lidar captures 3D points, reflecting a 2D surface in the 3D space and 2) most associated frustum spaces have one or multiple 3D points, leading to a compact range-view image; however, its shortcoming is that it is difficult to model an unorganized point cloud, such as the point cloud map in an HD map. This representation can be used in the perception module.

Bird's-eye view

The bird's-eye view (BEV)-based representation is a special case of 3D voxelization, as it ignores the height dimension. It projects 3D voxels to a BEV image, as shown in Figure 2(c). Let a 3D space with range H, W along the x- and y-axes, respectively. Each pixel is of size h, w, respectively. The (i, j)th pixel in the BEV image represents a pillar space, $\mathcal{V}_{i,j} = \{(x, y, z) | (i-1)h \leq x < ih, \quad (j-1)w \leq y < jw\}$. We then use a 2D matrix to represent a 3D point cloud. Let $\mathrm{X}^{(\mathrm{BEV})} \in \mathbb{R}^{H \times W}$, whose (i, j)th element is

$$X_{i,j}^{(\mathrm{BEV})} = \begin{cases} 1, & \text{when } \mathcal{V}_{i,j} \cap \mathcal{S} \neq \varnothing; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The matrix $\mathrm{X}^{(\mathrm{BEV})}$ records the occupancy in the 2D space. Note that there are a few variations of the BEV-based representations. For example, instead of using a binary value, MV3D [8] uses a few statistical values in each pillar space to construct $\mathrm{X}^{(\mathrm{BEV})}$.

The advantages of the BEV-based representation are that 1) it is easy to apply 2D vision-based techniques, 2) it is easy to merge with information from the HD map; for example, drivable areas and the positions of intersections encoded in the HD map can be projected onto the same 2D space and fused with lidar information, 3) it is easy to use for subsequent modules, such as prediction and motion planning, and 4) objects are always the same size regardless of range (contrasting with the range-view-based representation), which is a strong prior and makes the learning problem much easier. However, the shortcoming

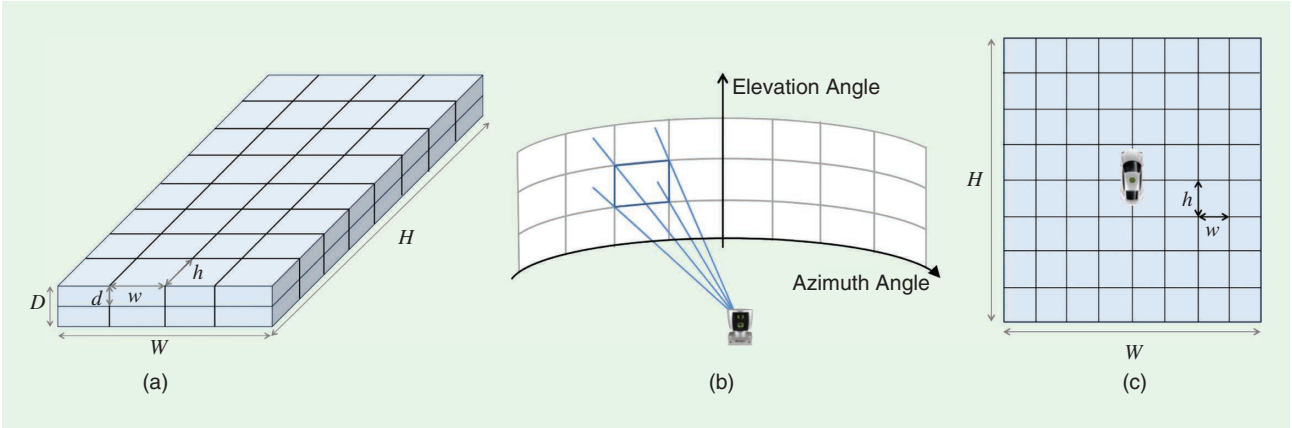


FIGURE 2. The common approaches used to discretize the 3D space. (a) The 3D voxel-based representation is used to discretize the 3D space into equally spaced nonoverlapping voxels from each of the three dimensions, (b) the range-view-based representation is used to discretize the 3D space along the azimuth and elevation angles, and (c) the BEV-based representation is used to discretize the 3D space along the x- and y-axes, omitting the height dimension.

of this voxelization is that 1) it also involves a serious tradeoff between resolution and memory, causing excessive quantization issues related to obtaining detailed information on small objects, 2) it does not consider the specific properties of organized 3D point clouds and cannot reason the occlusion, and 3) it causes the sparsity issue because most pixels are empty. This representation can be used in the perception module of autonomous driving.

Representative tools

3D point clouds have been studied across various communities, including robotics, computer graphics, computer vision, and signal processing. In the following sections, we introduce a few representative tools used to process and learn from 3D point clouds. We mainly emphasize deep neural network-based approaches because of their practical applications in autonomous driving.

Raw 3D point clouds are inherently unordered sets, and PointNet was designed to respect this property and produce the same output regardless of the ordering of the input data.

Nondeep learning methods

Before the emergence of deep learning, there have been many traditional methods used to handle 3D point clouds for various tasks. However, unlike deep neural networks, those conventional methods can hardly be described in a single methodological framework. This is because handcrafted tools are specifically designed to cater to the needs of each individual task. For example, in 3D point cloud segmentation and 3D shape detection, traditional techniques have been developed based on either region growth with simple geometric heuristics or graph-based optimization, or robust estimation methods, such as random sample consensus [9]. As another important task, 3D key point matching is closely related to 3D point cloud registration and 3D point cloud recognition. To tackle this task, many statistics-based methods have been developed in a handcrafted fashion and aim to describe the geometric structures around 3D key points or objects. For a more comprehensive discussion, see [10].

Convolutional neural networks

The motivation for using convolutional neural networks (CNNs) is to leverage off-shelf deep learning tools to process 3D point clouds. As regularized versions of MLPs, CNNs employ a series of convolution layers and are commonly applied to analyzing images and videos. A convolution layer operates a set of learnable filters on input data to produce the output that expresses the activation map of filters. The beauty of a convolution layer is weight sharing, that is, the same filter coefficients (weights) are applied to arbitrary positions in a 2D image, which not only saves a lot of learnable weights but also ensures shift invariance and helps avoid overfitting to limited training data. As a general and mature learning framework, CNNs and common variations are widely used in various computer vision tasks, including classification, detection, and segmentation and have achieved state-of-the-art performance in most tasks.

Based on the success of CNNs in images and videos, CNNs have been applied to 3D point cloud data as well. Multiple representations have been used, including the 3D voxelization-based representation (2), the range-view-based representation (3), and the BEV-based representation (4). A benefit of using CNNs to handle a 3D point cloud is that a convolution operator naturally involves local spatial relationships. In PointNet, each 3D point is processed individually, while in CNNs, adjacent voxels or pixels are considered jointly, providing richer contextual information. The basic operator is a 3D convolution for the 3D voxelization-based representation and a 2D convolution for the range-view-based representation and BEV-based representation, respectively. Without a loss of generality, consider a four-mode tensor $\mathrm{X} \in \mathbb{R}^{I \times J \times K \times C}$, after convolving with C 3D filters $\mathrm{H} \in \mathbb{R}^{k \times k \times k \times C}$, the (i, j, k, c')th element of the output $\mathrm{Y} \in \mathbb{R}^{I \times J \times K \times C'}$ is

$$\mathrm{Y}_{i,j,k,c'} = \sum_{c=0}^{C-1} \sum_{\ell=0}^{k-1} \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} \mathrm{H}_{i-\ell, j-m, k-n, c'} \mathrm{X}_{\ell, m, n, c}.$$

For simplicity, we omit the boundary issue. 3D convolution is expensive in both computation and memory usage.

Because of their discretization, many techniques and architectures developed for 2D images can be easily extended to handle 3D point clouds. Even though their discretization causes an inevitable loss of information, CNNs usually provide reliable performances and are widely used in many tasks. As discussed previously, one critical issue about discretizing a 3D point cloud is that a resulting 3D volume or 2D image is sparse. A huge amount of computation is wasted in handling empty voxels.

In summary, CNNs handle a 3D point cloud in a discretized representation. This approach inevitably modifies the exact 3D position information, but still provides strong and promising empirical performances because of the spatial relationship prior and the maturity of CNNs. It is thus widely used in the industry.

PointNet-based methods

The motivation for using PointNet-based methods is to directly handle raw 3D points by deep neural networks without any discretization. PointNet [7] is a pioneering work that achieves this goal. Raw 3D point clouds are inherently unordered sets, and PointNet was designed to respect this property and produce the same output regardless of the ordering of the input data. The key technical contribution of PointNet is to use a set of shared pointwise MLPs followed by global pooling to extract geometric features while ensuring this permutation-invariant property of raw 3D data. Even though the architecture is simple, it has become a standard building block for numerous 3D point cloud learning algorithms and achieves a surprisingly strong performance on 3D point cloud recognition and segmentation.

PointNet considers the raw-point-based representation $\mathrm{X}^{(\mathrm{raw})}$ (1). Let $\mathrm{H} \in \mathbb{R}^{N\times D}$ be a local-feature matrix, where the ith row H_i represents the features for the ith point, and let $\mathbf{h} \in \mathbb{R}^D$ be a global-feature vector. A basic computational block of PointNet works as

$$\begin{aligned} \mathrm{H}_i &= \mathrm{MLP}^{(\mathrm{L})}(\mathrm{X}_i^{(\mathrm{raw})}) \in \mathbb{R}^D, \quad \text{for } i = 1, \ldots, N, \\ \mathbf{h} &= \mathrm{maxpool}(\mathrm{H}) \in \mathbb{R}^D, \end{aligned} \tag{5}$$

where $\mathrm{X}_i^{(\mathrm{raw})}$ is the ith 3D point's feature, and $\mathrm{MLP}^{(\mathrm{L})}(\cdot)$ denotes L-layer MLPs, which map each 3D point to a feature space, and maxpool$(\cdot)$ performs downsampling by computing the maximum values along the column (the point dimension) [see Figure 3(a)]. Note that each 3D point goes through the same MLPs separately.

Intuitively, the MLPs propose D representative geometric patterns and test whether those patterns appear around each 3D point. The max-pooling records the strongest response over all of the 3D points for each pattern. Essentially, the global-feature vector $\mathbf{h}$ summarizes the activation level of D representative geometric patterns in a 3D point cloud, which can be used to recognize a 3D point cloud. Meanwhile, because each 3D point goes through the same MLPs separately and the max-pooling removes the point dimension, the entire computational block is permutation invariant; that is, the ordering of 3D points does not influence the output of this block. To some extent, PointNet for 3D point cloud learning is similar to principal component analysis (PCA) for data analysis: it is simple, general, and effective. Just like PCA, PointNet extracts global features in a 3D point cloud.

To summarize, PointNet-based methods handle 3D point clouds in the raw-point-based representation and ensure the permutation invariance. The effectiveness has been validated in various processing and learning tasks.

Graph-based methods

The motivation for using graph-based methods is to leverage the spatial relationships among 3D points to accelerate the end-to-end learning of deep neural networks. One advantage of CNNs is that a convolution operator considers local spatial relationships; however, those relationships are between adjacent voxels (or adjacent pixels), not original 3D points. To capture the local relationships among 3D points, one can introduce a graph structure, where each node is a 3D point and each edge reflects the relationship between each pair of 3D points. This graph structure is a discrete proxy of the surface of an original object. A matrix representation of a graph with N nodes is an adjacency matrix $A \in \mathbb{R}^{N\times N}$, whose (i, j)th element reflects the pairwise relationship between the ith and jth 3D points [see Figure 3(b)]. Graph-based methods usually consider the raw-point-based representation (1). Each column vector in $\mathrm{X}^{(\mathrm{raw})}$ is then data supported on the A graph, called a *graph signal*.

There are several ways to construct a graph, such as K-nearest-neighbor (KNN), ϵ-nearest-neighbor, and learnable graphs. A KNN graph is one in which two nodes are connected by an edge when their Euclidean distance is among the Kth smallest Euclidean distances from one 3D point to all of the other 3D points. An ϵ-nearest-neighbor graph is one in which two nodes are connected by an edge when their Euclidean distance is smaller than a given threshold ϵ. Both KNN and ϵ-graphs can be efficiently implemented using efficient data structures, such as Octree [11]. A learnable graph is one whose adjacency matrix is trainable in an end-to-end learning architecture.

A general graph-based operation is a graph filter, which extends a classical filter to the graph domain and extracts features from graph signals. The most elementary nontrivial graph filter is called a *graph shift operator*. Some common options for a graph shift operator include the adjacency matrix A, the transition matrix $\mathrm{D}^{-1}\mathrm{A}$ (D is the weighted degree matrix, a diagonal matrix with $\mathrm{D}_{i,i} = \sum_j \mathrm{A}_{i,j}$ reflecting the density around the ith point), the graph Laplacian matrix $\mathrm{D} - \mathrm{A}$, and many other structure-related matrices. The graph shift replaces the signal value at a node with a weighted linear combination of values at its neighbors; that is, $\mathrm{Y} = \mathrm{A}\mathrm{X}^{(\mathrm{raw})} \in \mathbb{R}^N$, where $\mathrm{X}^{(\mathrm{raw})} \in \mathbb{R}^{N\times 3}$ is an input graph signal (an attribute of a point cloud). Every linear, shift-invariant graph filter is a polynomial in the graph shift

$$h(\mathrm{A}) = \sum_{\ell=0}^{L-1} h_\ell \mathrm{A}^\ell = h_0\mathrm{I} + h_1\mathrm{A} + \cdots + h_{L-1}\mathrm{A}^{L-1},$$

where $h_\ell, \ell = 0, 1, \ldots, L-1$ are filter coefficients and L is the graph filter length. A higher-order corresponds to a larger receptive field on the graph vertex domain. The output of graph filtering is given by the matrix-vector product $\mathrm{Y} = h(\mathrm{A})\mathrm{X}^{(\mathrm{raw})}$.

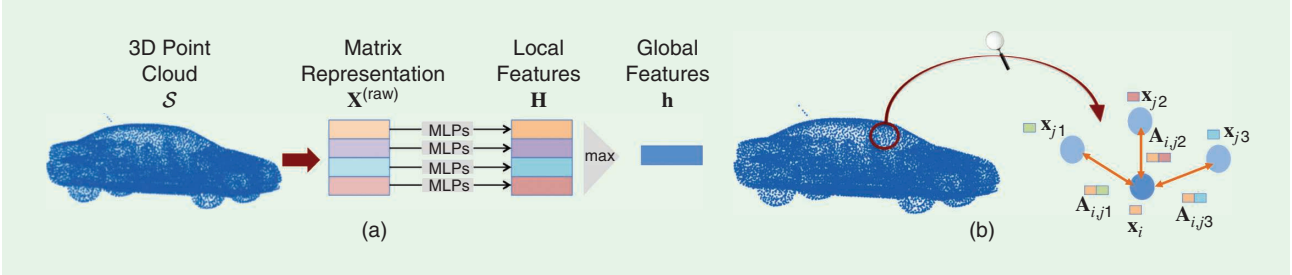


FIGURE 3. The representative tools. (a) PointNet uses a set of shared pointwise MLPs followed by max-pooling to extract the geometric features that exhibit the permutation-invariant property of raw 3D point clouds. (b) The graph-based methods introduce a graph structure that captures the local relationships among 3D points. In the graph, each node is a 3D point and each edge reflects the relationship between each pair of 3D points.

Graph filtering can be used in various processing tasks, such as 3D point cloud downsampling and denoising [12].

Inspired by the success of graph neural networks in social network analysis, numerous recent research works incorporate graph neural networks to handle a 3D point cloud. As the first such work, [13] introduces two useful techniques: the edge convolution operation and learnable graphs. The edge convolution is a convolution-like operation used to extract geometric features on a graph. The edge convolution exploits local neighborhood information and can be stacked to learn global geometric properties. Let $\mathrm{H} \in \mathbb{R}^{N \times d}$ be a local-feature matrix, where the ith row H_i represents the features for the ith point. A basic computational block works as $\mathrm{H}_i = \|_{(i,j)\in\mathcal{E}} g(\mathrm{X}_i^{(\mathrm{raw})}, \mathrm{X}_j^{(\mathrm{raw})}) \in \mathbb{R}^d$, where $\mathcal{E}$ is the edge set and $g(\cdot,\cdot)$ is a generic mapping, implemented by some neural networks, and $\|$ is a generic aggregation function, which could be the summation or maximum operation. To some extent, the edge convolution extends PointNet by inputting a pair of neighboring points' features. The edge convolution is also similar to graph filtering: both aggregate neighboring information; however, the edge convolution specifically models each pairwise relationship by a nonparametric function. The authors in [13] also suggest learning a graph dynamically; it always uses a KNN graph, but the distance metric is the Euclidean distance in the high-dimensional feature space.

Graph-based methods build graph structures that capture the distribution of a 3D point cloud and take advantage of local spatial relationships.

Subsequent research has proposed to use novel graph neural networks to handle 3D point cloud recognition and segmentation. As one of the most recent works in this area, [14] constructs the deepest-yet graph convolution network architecture, which has 56 layers. It transplants a series of techniques from CNNs, such as residual and dense connections as well as dilated graph convolutions, to the graph domain.

To summarize, graph-based methods build graph structures that capture the distribution of a 3D point cloud and take advantage of local spatial relationships. This approach handles 3D point clouds in the raw-point-based representation, ensuring the permutation invariance. This approach is less mature: even though leveraging a graph improves its overall performance, graph construction is more art than science and takes extra computational cost [13]. Additionally, deep architectures used for graph-based neural networks still need more exploration [14].

3D Point cloud processing for HD map creation

Overview of an HD map creation module

To precisely represent the static 3D environment and traffic rules, an HD map usually contains two map layers: a point cloud map, representing 3D geometric information of surroundings, and a traffic rule-related semantic feature map, containing road boundaries, traffic lanes, traffic signs, traffic lights, the height of the curbs, and so forth. The main reason for creating an offline HD map is that understanding traffic rules in real time is too challenging. For example, based on the current technology, it is difficult for an AV to determine the correct lane in real time when driving into at an intersection with complicated lane merging and splitting. In contrast, all traffic rules and environmental information can easily be encoded in an HD map, which goes through an offline process with human supervision and quality assurance. An HD map provides strong and indispensable priors and fundamentally eases the designs of multiple modules in an autonomy system, including localization, perception, prediction and motion planning. Therefore, an HD map is widely believed to be an indispensable component of autonomous driving.

Priors for localization

The role of localization is to localize the pose of an AV. In an HD map, the point cloud map and the traffic rule-related semantic features, such as lane markers and poles, are usually served as localization priors for the map-based localization. These priors are used to register real-time lidar sweeps to the point cloud map, such that one can obtain the real-time high-precision ego motion of an AV.

Priors for perception

The role of perception is to detect all objects in the scene as well as their internal states. The perception module can use an HD map to serve as a prior for detection. For example, the positions of traffic lights in an HD map are usually served as perception priors for traffic light-state estimation. Using the point cloud map as priors, one can separate a real-time lidar sweep into foreground and background points in real time. We can then remove background points, which are those lying on the static scenes, such as road surfaces and the trunks of trees, and feed only foreground points to the perception module. This formalism can significantly reduce the computational cost and improve the precision of object detection.

Priors for prediction

The role of prediction is to predict the future trajectory of each object in the scene. In an HD map, 3D road and lane geometries and connectivities are important priors to the prediction module. These priors can be used to guide the predicted trajectories of objects to follow the traffic lanes.

Priors for planning

The role of motion planning is to determine the trajectory of an AV. In an HD map, traffic rule-related semantic features such as lane geometries and connectivities, traffic lights, traffic signs and the speed limit of lanes are indispensable priors for the planning module. These priors are used to guide the designed trajectory to follow the correct lane and obey the stop signs and other traffic signs.

Because an HD map is critical to autonomous driving, it must be created with high precision and be up to date. To achieve this, it usually needs sophisticated engineering procedures to analyze

data from multiple modalities by exploiting both machine learning techniques and human supervision. A standard map creation module includes two core components: 3D point cloud stitching and semantic feature extraction (see Figure 4). 3D point cloud stitching merges real-time lidar sweeps collected from multiple vehicles across times into a point cloud map, and semantic feature extraction extracts semantic features, such as lane geometries and traffic lights, from the point cloud map. To view a video illustration of the industrial-level HD maps, visit https://vimeo.com/303412092 and additional illustrations in the supplementary material that accompanies this article on IEEE *Xplore*.

3D point cloud stitching

The goal of 3D point cloud stitching is to create a high-precision point cloud map from the sensor data collected by a fleet of vehicles across time. Because a point cloud map dominates the precision of all the map priors, centimeter-level precision is required for any local portion of the point cloud map. To promptly create and update city-scale HD maps, the process of 3D point cloud stitching must be highly robust and efficient.

One fundamental problem of 3D point cloud stitching is that of estimating the six-degree-of-freedom (DOF) pose of each lidar sweep, also called *lidar pose*. We consider the map frame as the standardized global frame, and the lidar frame as the ego frame of an AV at the time stamp when the corresponding real-time lidar sweep is collected. A lidar pose is then a transformation between the map frame and the lidar frame. It includes 3D translation and 3D rotation. Note that the 6-DOF pose can be represented as a 4×4 homogeneous transformation matrix. Using the lidar poses, all of the lidar sweeps can be synchronized to the standardized global frame and integrated to form a dense 3D point cloud. To estimate lidar poses, a commonly used technique is simultaneous localization and mapping (SLAM). Let $\mathcal{S}_i$ and $\mathcal{S}_j$ be the ith and jth real-time lidar sweeps, respectively. SLAM works as

$$\text{argmin}_p \left[\sum_{p_i} \sum_{p_j} h_{\mathcal{S}_i,\mathcal{S}_j}(p_i, p_j) + g(p_i) \right], \tag{6}$$

where p_i is the 6-DOF lidar pose associated with the ith real-time lidar sweep, $h_{\mathcal{S}_i,\mathcal{S}_j}(p_i, p_j)$ indicates the negative log likelihood of the measurement on the misalignment between $\mathcal{S}_i$ and $\mathcal{S}_j$ and $g(\cdot)$ indicates the negative log likelihood of the difference between the predicted lidar position in the map frame and the direct measurement of GPS [15]. A typical choice of $h_{\mathcal{S}_i,\mathcal{S}_j}(p_i, p_j)$ is the objective function of the iterative closest point (ICP) algorithm. We thus minimize the objective function of the ICP algorithm and assign the optimized value to $h_{\mathcal{S}_i,\mathcal{S}_j}(p_i, p_j)$.

SLAM is a big research field in robotics communities and there exists extensive research that aims to solve the optimization problem (6). For example, the filter-based SLAM solves the optimization problem (6) in an approximated and online fashion. It employs Bayes filtering to predict and optimize the map and lidar poses iteratively based on its online sensor measurements. On the other hand, the graph-based SLAM optimizes all of the lidar poses together by using all sensor measurements across time. It constructs a pose graph that models the relations among lidar poses. In the pose graph, the ith node is the ith lidar pose p_i; and the (i, j)th edge is the cost of misalignment between the ith and jth lidar poses, $h_{\mathcal{S}_i,\mathcal{S}_j}(p_i, p_j)$ (see the pose graph in Figure 4). Intuitively, each edge weight is either the total point-to-point distance or the total point-to-plane distance between two lidar sweeps. Solving (6) is thus equivalent to minimizing the total sum of the edge weights of a pose graph. For a city-scale map creation, the SLAM solution must satisfy the requirements discussed in the following sections.

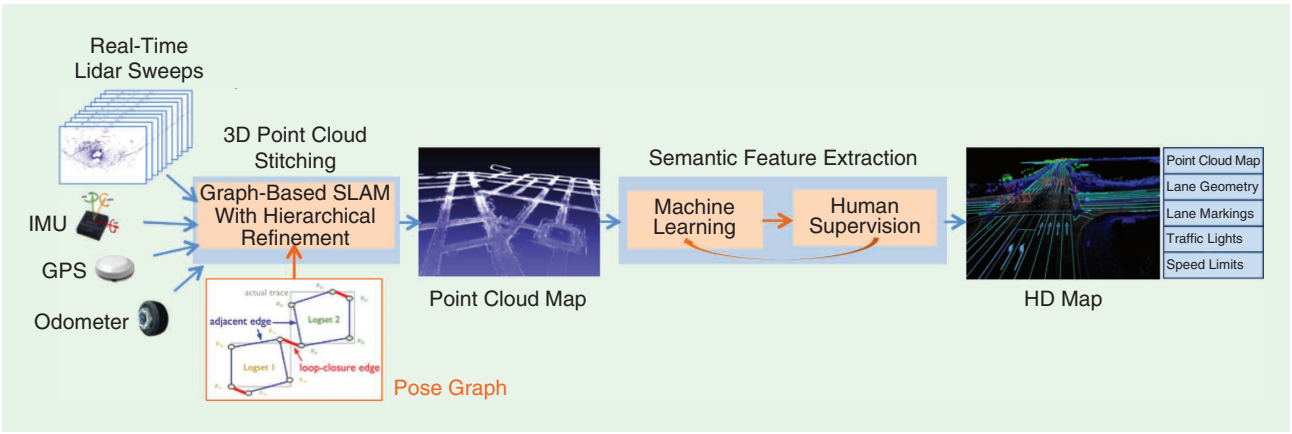


FIGURE 4. A standard HD map creation system includes two core components: a 3D point cloud stitching and semantic feature extraction. 3D point cloud stitching usually adopts graph-based simultaneous localization and mapping (SLAM) with hierarchical refinement, and semantic feature extraction contains iterative procedures of machine learning and human supervision. A key component in graph-based SLAM is a pose graph, which models the relations among lidar poses. The nodes are lidar poses and edges reflecting the misalignment level between two lidar poses. The final outputs include a point cloud map, which is a dense 3D point cloud, as well as a traffic rule-related semantic feature map, containing the positions of land markers, traffic signs, and traffic lights. IMU: inertial measurement unit.

High local and global precision

Local precision indicates that the lidar poses in a local region are accurate with respect to one another, and global precision indicates that all of the lidar poses in the entire HD map are accurate with respect to the standardized global frame. For the SLAM solution, centimeter-/microradian-level local precision must be achieved because autonomy software modules require highly accurate local surroundings from the HD map, and centimeter-level global precision is useful to accelerate the HD map update process, especially for the city-scale application.

High robustness

The SLAM solution requires handling the noisy sensor measurements collected by multiple vehicles driving in complicated scenes and complex driving conditions in the real world.

High efficiency

The SLAM solution necessitates handling the optimization of more than 100 million lidar poses. To achieve high precision and robustness, the graph-based SLAM is a better option than the filter-based SLAM because the global optimization formalism makes the graph-based SLAM inherently more accurate; however, it is still challenging to solve the city-scale graph-based SLAM problem with high efficiency and robustness. There are two main reasons for this. First, the scale of the problem is enormous. It is expensive to solve the optimization problem (6) in a brute-force way because the core step of the optimization algorithm is solving a series of equations associated with an $n \times n$ matrix, where n is the total number of lidar poses. For a city-scale map, n could be more than 100 millions, causing big issues for both computational efficiency and numerical stability of the optimization algorithm. Second, evaluating edge weights in a pose graph usually suffers from low precision because sensor data are collected in complex driving conditions. For example, the calculation of the misalignment between consecutive lidar sweeps will likely be compromised by the moving objects.

To effectively solve this problem, the graph-based SLAM with the hierarchical refinement formalism can be adopted [16]. The functionality of hierarchical refinement formalism provides a good initialization for the global optimization, making the optimization both fast and accurate. The hierarchical refinement formalism distinguishes two types of edges in a pose graph; that is, adjacent and loop-closure edges. Adjacent edges model the relations between two lidar poses whose corresponding lidar sweeps are consecutively collected from the same log set, and loop-closure edges model the relations between two lidar poses whose corresponding lidar sweeps are collected around the same location from different log sets (different vehicles or across time). To handle these two types of edges, the hierarchical refinement formalism includes two steps: 1) optimizing adjacent edges, including a chain of lidar poses from a single log set and 2) optimizing loop-closure edges, including lidar poses across log sets, as shown in Figure 4. In the first step, rather than relying simply on aligning lidar sweeps, sensor measurements from multiple modalities, including inertial measurement units (IMUs), GPS, odometer, cameras, and lidar, can be fused together to calculate the adjacent edges. Because consecutive lidar sweeps have similar lidar poses, this step is usually easy and provides extremely high precision. In the second step, the loop-closure edges are calculated by aligning lidar sweeps through the ICP algorithm. After these two steps, we then perform global optimization (6).

Because most edges in a pose graph are adjacent edges, which can be highly optimized through the first step, the hierarchical refinement formalism provides a good initialization for the global optimization. Therefore, the computational cost for optimizing the entire pose graph can be significantly reduced, and the robustness of the global optimization can be greatly improved by the hierarchical refinement formalism.

Semantic feature extraction

The goal of semantic feature extraction is to extract traffic rule-related semantic features, such as lane geometries, lane connectivities, traffic signs, and traffic lights, from the point cloud map. This component requires both high precision and recall. For example, missing a single traffic light prior in a city-scale HD map can potentially cause serious issues to the perception and motion planning modules, severely jeopardizing the safety of autonomous driving.

The semantic feature-extraction component usually contains two iterative steps. The first of which uses machine learning techniques to automatically extract features, while the second step introduces human supervision and quality assurance processes to ensure the high precision and recall of semantic features.

To automatically extract features, standard machine learning techniques are based on CNNs. The inputs are usually the combination of the lidar ground images and the camera images associated with the corresponding real-time lidar sweep. A lidar ground image renders the BEV-based representation of the point cloud map obtained in 3D point cloud stitching, where the values of each pixel are the ground height and laser reflectivity of each lidar point. The outputs are usually the semantic segmentation of either the lidar ground images or the camera images. The networks follow from standard image segmentation architectures.

After obtaining the output, the pixel-wise semantic labels are projected back to the point cloud map. By fitting the projected 3D points into 3D splines or 3D polygons, the traffic rule-related semantic feature map can then be obtained. Note that the human-editing outcomes also serve as an important source of training data for automatic feature-extraction algorithms, where these two steps therefore form a positive feedback loop to improve the precision and efficiency of HD map production. There still exist several challenges for HD map creation, which are presented in the next section.

Real-world challenges

Point cloud map with centimeter-level global precision

Global precision can greatly benefit the updating of a city-scale point cloud map. The changes of the urban appearance usually take place locally. Ideally, the map update should focus on the targeted portion of the pose graph; however, a point cloud map with high local precision but without high global precision cannot freely access the targeted portion from a global aspect and guarantee its precision. In comparison, given a point cloud map with high global precision, one can focus on updating the targeted portion of the pose graph, thus significantly reducing the scale of computation; however, it is challenging to enforce the global precision to the graph-based SLAM. This is because the global optimization formalism of graph-based SLAM tends to distribute the error of each edge uniformly in the graph. Therefore, even if the GPS observations are accurate, the corresponding lidar poses can be misaligned after global optimization. Enforcing centimeter-level global precision of a point cloud map can be especially challenging in the places where the GPS signal is unavailable, such as in a building canyon, a tunnel, or an underground garage.

The importance of the localization module to autonomous driving is that it bridges the HD map to the other modules in an autonomous system.

Automatic semantic feature extraction

Although there exists extensive research on semantic segmentation based on 3D point clouds and camera images, it is still challenging to automatically extract lane connectivities in the intersections and traffic lights that indicate lane-control relations. This is due to limited training labels and complex traffic conditions. Currently, the solution for extracting complex semantic features, such as traffic light to lane-control information, still relies largely on human supervision, which is both expensive and time consuming.

3D Point cloud processing for localization

Overview of the localization module

As introduced in the "A Tour of an Autonomous System" section, the localization module finds th ego position of an AV relative to the reference position in the HD map. It consumes the real-time measurements from multiple sensors including lidar, the IMU, GPS, odometer, and cameras as well as the HD map (see Figure 5). Because of the 3D representation of an HD map, the ego position of an AV is a 6-DOF pose (translation and rotation), which is a rigid transformation between the map and lidar frames. The importance of the localization module to autonomous driving is that it bridges the HD map to the other modules in an autonomous system. For example, by projecting the HD map priors, such as the lane geometries to the lidar frame, the AV gains the knowledge of which lane it drives on and which lanes the detected traffic is on. To view a video illustration of real-time localization, visit (https://vimeo.com/327949958) and additional illustrations in the supplementary material that accompanies this article on IEEE *Xplore*.

To enable full autonomous driving, high precision and robustness are the most critical criteria for the performance of the localization module. High precision indicates that the error of translation and the error of rotation angle should be at the centimeter and microradian levels, respectively. It allows for the traffic detected from 1 km away to be associated with the correct lanes in the HD map, and the lane-change intentions of the nearer traffic can be predicted by measuring the distance between its wheels to the lane boundaries, which can significantly benefit motion planning and prediction modules. Robustness indicates that the localization module is expected to work in all driving conditions with changes in illumination, weather, traffic, and road conditions. Note that although the commercial-grade GPS/IMU unit with real-time kinematics mode has accurate position measurement in open areas, it is not robust enough for autonomous driving because it suffers from the low-precision issue in the city due to multipath effects.

To achieve these aforementioned criteria, the map-based localization with multisensor fusion is the standard approach. As discussed in previous sections, an HD map could be created

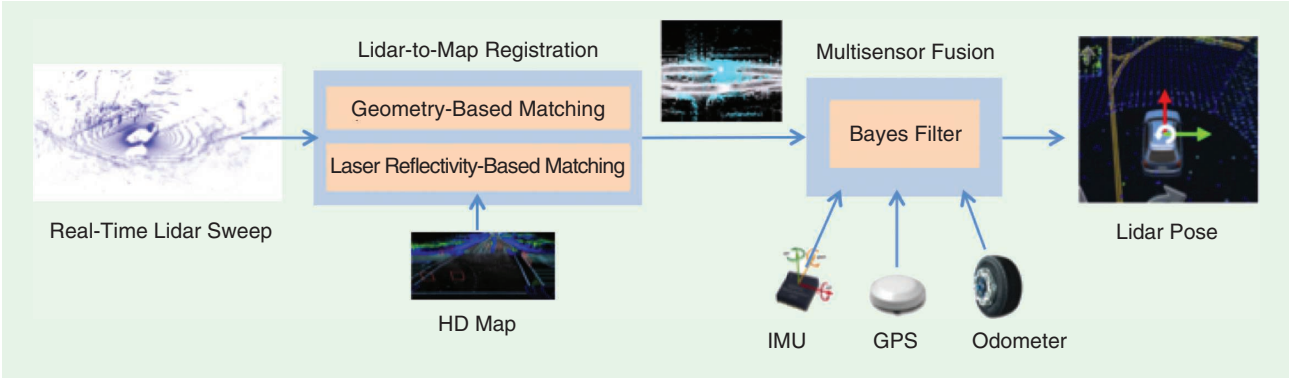


FIGURE 5. A standard map-based localization system includes two core components: lidar-to-map registration and multisensor fusion. Lidar-to-map registration uses geometry-based matching and laser reflectivity-based matching to achieve high precision and recall, and multisensor fusion adopts Bayes filters to merge multiple modalities.

beforehand to significantly ease the localization. On the contrary, the SLAM-based solution cannot satisfy these criteria.

Map-based localization

The basic idea of the map-based localization is to estimate the lidar pose by matching a lidar sweep to the point cloud map in an HD map by leveraging the measurements from the IMU, GPS, and cameras to make pose estimation robust. A map-based localization system usually consists of two components, as depicted in Figure 5. The first component is the lidar-to-map registration, which computes the lidar pose by registering lidar sweep to a point cloud map; The second component is the multisensor fusion, which estimates the final pose from the IMU, odometer, and GPS as well as the estimation from the lidar-to-map registration.

Lidar-to-map registration

The lidar-to-map registration component directly estimates the lidar pose by matching the lidar sweep to the point cloud map. Let $\mathcal{S}, \mathcal{S}^{(\text{map})}$ be a real-time lidar sweep and the point cloud map, respectively. The problem of lidar-to-map registration can be formulated as

$$\text{argmin}_p \left[\sum_{\mathbf{x}_i \in \mathcal{S}} g(f_p(\mathbf{x}_i), \mathcal{S}^{(\text{map})}{}_{i^*}) \right], \tag{7}$$

where p is the lidar pose, $\mathbf{x}_i$ is the ith 3D point in the lidar sweep, and $\mathcal{S}^{(\text{map})}{}_{i^*}$ is the 3D point in the point cloud map, which is associated with the ith 3D point in the lidar sweep. The associated index i^* is usually chosen from the closest point in the Euclidean distance. $f_p: R^3 \rightarrow R^3$ is the function that transforms a 3D point x_i in the lidar frame into the map frame based on the lidar pose p, and $g(\cdot)$ indicates a loss function measuring the misalignment between the points from the lidar sweep and the HD map. Usually, $g(\cdot)$ takes the form of the point-to-point, point-to-line, or point-to-plane distances between the associated points in the lidar sweep and the point cloud map.

To solve (7) and achieve high precision and recall, the following two major approaches are used:

- *Geometry-based matching*: This approach calculates the high-precision 6-DOF pose by matching the lidar sweep to the point cloud map based on the ICP algorithm [17]. This approach usually works well in heavy traffic and challenging weather conditions, such as snow, because a point cloud map contains abundant geometry priors for lidar sweeps to match with; however, in geometry-degenerated scenes, such as tunnels, bridges, and highways, the ICP calculation could diverge because of the loss of geometric patterns, hence causing bad precision.
- *Laser reflectivity-based matching*: This approach calculates the pose by matching a lidar sweep to a point cloud map based on laser reflectivity signals. The matching can be done in either the dense 2D image-matching method or the feature extraction-based ICP-matching method. For the first method, the laser reflectivity readings of the lidar sweep and the point cloud map are first converted into gray-scale 2D images [following the BEV-based representation (4)] and then the pose is calculated using image-matching techniques. Note that this method only calculates the x, y, and yaw components of the pose. To obtain the 6-DOF pose, the z, roll, and pitch components are estimated based on the terrain information in the HD map. For the second method, the region-of-interest objects, such as lane markers and poles, are first extracted from the lidar sweep based on the laser-reflectivity readings [18]. The ICP algorithm can then be used to calculate the lidar pose by matching the region-of-interest objects between the real-time lidar sweeps and the priors in the HD map. This approach usually outperforms geometry-based matching in the highway and bridge scenarios because those scenarios lack geometry features but have rich laser reflectivity textures on the ground (e.g., dashed lane markers). This approach does not work well in challenging weather conditions such as heavy rain and snow, where the laser reflectivity of the ground will change significantly.

To achieve the best performance, both of these two strategies can simultaneously be used to estimate lidar poses; however, lidar-to-map registration alone cannot 100% guarantee the precision and recall for pose estimation over time. To give an extreme example, if lidar is totally occluded by trucks driving side by side or front and back, the lidar-to-map registration component would fail. To handle extreme cases and make the localization module robust, the multisensor fusion component is required.

Multisensor fusion

The multisensor fusion component is used to estimate a robust and confident pose from the measurements of multiple sensors, including the IMU, GPS, odometer, and cameras as well as the poses estimated by the lidar-to-map registration module. The standard approach of multisensor fusion is to employ a Bayes-filter formalism, such as a Kalman, extended, or particle filter. Bayes filters consider an iterative approach to predict and correct the lidar pose and other states based on the vehicle motion dynamics and multisensor readings. In autonomous driving, the states tracked and estimated by Bayes filters usually include motion-related states such as pose, velocity, acceleration, and so on, and sensor-related states such as IMU bias and so forth.

Bayes filters work in two iterative steps: prediction and correction. In the prediction step, during the gaps between sensor readings, the Bayes filter predicts the states based on the vehicle motion dynamics and the assumed sensor model. For example, by taking the constant acceleration approximation as the vehicle motion dynamics during a short period of time, the evolution of pose, velocity, and acceleration can be predicted by Newton's laws. The IMU bias states can be predicted by assuming that it behaves as white noise.

In the correction step, when receiving a sensor reading or a pose measurement, the Bayes filter corrects the states based on the corresponding observation models. For example, when

an IMU reading is received, the states of acceleration, angular velocities, and the IMU bias are corrected. When a pose measurement is received, the pose state is corrected. Note that the states require correction because the prediction step is not prefect and there are accumulated errors over time.

Real-world challenges

The real-world challenges of the localization module are that it works in extreme scenes. For example, when an AV drives through a straight tunnel without a dashed lane marker, there are few geometric and texture features, causing the failure of the lidar-to-map registration. When an AV is surrounded by large trucks, lidar could be totally blocked, also causing the failure of the lidar-to-map registration. When the failure of the lidar-to-map registration lasts for several minutes, the lidar pose estimated by the multisensor fusion component will drift significantly and the localization module will lose precision.

A late-fusion module unifies the bounding box information from multiple pipelines and outputs a final 3D bounding boxes with tracks.

3D Point cloud processing for perception

Overview of the perception module

As introduced in the "A Tour of an Autonomous System" section, the perception module is the visual system of an AV that enables the perception of the surrounding 3D environment. The input of the perception module usually includes the measurements from cameras, lidar, radar, and ultrasound as well as the ego-motion pose output from the localization module and the priors from the HD map. The outputs of the perception module are typically traffic light states and objects' 3D bounding boxes with tracks.

As discussed in the "A Tour of an Autonomous System" section, multiple sensing modalities are used to ensure the robustness of the perception module. Depending on the mechanism used to fuse those modalities, a perception module can be categorized into late and early fusion. Late fusion combines modalities in a semantic space, which usually happens in the final step, and early fusion unites modalities in a feature space, which usually happens in an early or intermediate step.

Figure 6(a) shows the standard framework of a late-fusion-based perception module. To obtain objects' 3D bounding boxes with tracks, a late-fusion-based perception module uses an individual pipeline to handle each sensor input. Each pipeline includes the detection and the association and tracking components. The detection component finds bounding boxes, and the association and tracking component tracks bounding boxes across frames to assign a unique identity for each individual object. A late-fusion module unifies the bounding box information from multiple pipelines and outputs a final 3D bounding boxes with tracks. In comparison, Figure 6(b) shows an early-fusion-based perception module. It uses an early-fusion detector to take the outputs from all the sensing modalities and produces all of the 3D bounding boxes. It then uses an association and tracking component to associate 3D bounding boxes across frames and assigns an identity for each object. To estimate traffic light states, a traffic light-state estimator extracts the traffic light regions from images according to the position priors in an HD map and then uses machine learning techniques to analyze the image and identify the traffic light state.

The late-fusion-based approach is much more mature, while the early-fusion-based approach is believed to have bigger potential [8]. The industry has adopted the late-fusion-based approach for decades because it modularizes the tasks and makes each sensor pipeline easy to implement, debug, and manage. The early-fusion-based approach carries the spirit of end-to-end learning and enables the mutual promotion of multiple sensing modalities in a high-dimensional feature space; however, there are still significant challenges in this research direction, and many companies still use the late-fusion-based approach.

A robust perception module usually includes multiple intermediate components, such as lane detection, 2D and 3D object detection, semantic segmentation, and object tracking to achieve the final goal. Among those components, 3D object detection is particularly interesting and challenging because it must handle real-time lidar sweeps and can directly produce the 3D bounding boxes for all of the objects in the scene. Recently, this task has drawn much attention when combined with the power of deep learning [8]. In the next sections, we focus on 3D object detection.

3D object detection

The task of 3D object detection is to detect and localize objects in the 3D space with the representation of bounding boxes based on one or multiple sensor measurements. 3D object detection usually outputs 3D bounding boxes of objects, which are the inputs for the component of object association and tracking. Based on the usage of sensor measurements, we can categorize 3D object detection into lidar- [see Figure 4(a)] and fusion-based detection [see Figure 4(b)]. Qualitative performances are illustrated in the supplementary material that accompanies this article on IEEE *Xplore*.

Lidar-based detection

Let $\mathcal{S}$ be a real-time lidar sweep. A lidar-based detector aims to find all of the objects in the sweep, that is,

$$\{\mathbf{o}_i\}_{i=1}^{O} = h(\mathcal{S}), \tag{8}$$

where $\mathbf{o}_i = [\mathbf{y}_i, \mathbf{b}_i]$ is the ith object in the 3D scene with $\mathbf{y}_i$ as the object's category, such as the vehicle, bikes, and pedestrians, and $\mathbf{b}_i$ as the corners of the bounding box. Now the detection function $h(\cdot)$ is typically implemented with deep neural network-based architectures.

The main difference between 2D and 3D object detection is the input representation. Different from a 2D image, a real-time lidar sweep could be represented in various ways, leading

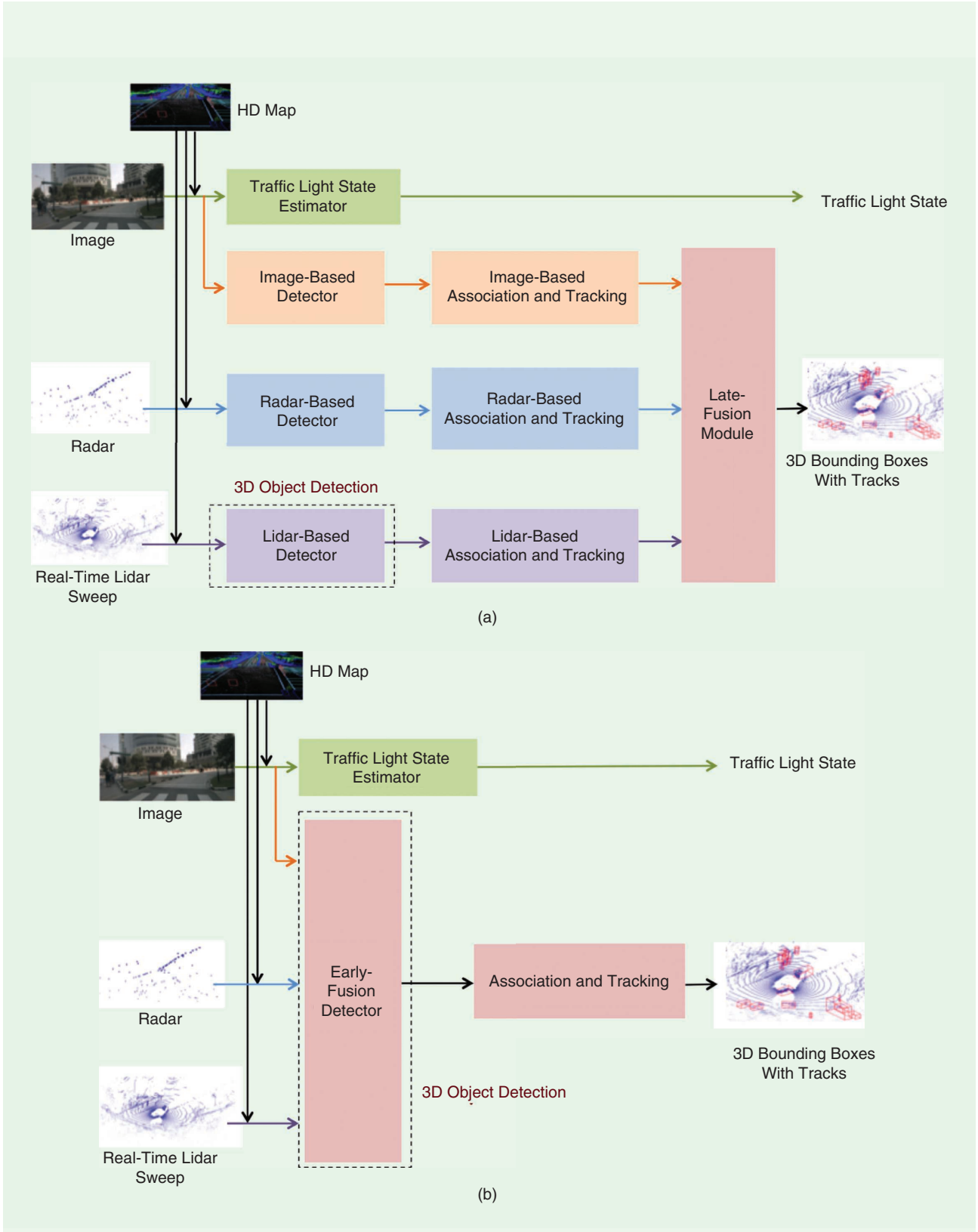


FIGURE 6. A perception module takes multiple sensing modalities and outputs traffic light states and objects' 3D bounding boxes with tracks. Depending on the mechanism used to fuse modalities, a perception module is categorized into (a) late fusion, which fuses in a semantic space, or (b) early fusion, which fuses in a feature space.

to corresponding operations in subsequent components. For example, PointRCNN [19] adopts the raw-point-based representation (1) and then uses PointNet with multiscale sampling and grouping to learn pointwise features, fully convolutional network [20] takes on the 3D voxelization-based representation (2) and uses 3D convolutions to learn voxel-wise features, PIXOR [21] assumes the BEV-based representation (4) and then uses 2D convolutions to learn pixel-wise features, and LaserNet [6] employs the range-view-based representation (3) and then uses 2D convolutions to learn pixel-wise features.

Some other methods consider hybrid representations. VoxelNet [22] proposes a voxel feature encoding (VFE) layer that combines the advantages of both the raw-point-based representation and the 3D voxelization-based representation. VFE first groups 3D points according to the 3D voxel in which they reside, then uses PointNet to learn pointwise features in each 3D voxel, and finally aggregates pointwise features to obtain voxel-wise feature for each 3D voxel. The benefit of VFE is that it converts raw 3D points to a 3D voxelization-based representation and simultaneously learns 3D geometric features in each 3D voxel.

Bin-based localization first finds the bin associated with the center location of an object and then regresses the residual.

Similar to 2D objection detection, there are usually two paradigms of 3D object detection: single- and two-stage detection. Single-stage detection directly estimates bounding boxes, while two-stage detection first proposes coarse regions that may include objects and then estimates bounding boxes. Single-stage detection directly follows (8). To implement the detection function $h(\cdot)$, a deep neural network architecture usually includes two components: a backbone, which extracts deep spatial features, and a header, which outputs the estimations. For a backbone, all of these methods use 2D/3D CNNs with a multiscale, pyramidal hierarchical structure. One off-the-shelf backbone structure is feature pyramid networks [23]. A header is usually a multitasking network that handles both category classification and bounding box regression. Typically, it is small and efficient. Some off-the-shelf header structures are a single-shot detector [24] and other small CNNs. The two-stage detection implements the detection function $h(\cdot)$ in two stages, that is,

$$\{r_i\}_{i=1}^{R} = h_1(\mathcal{S}), \tag{9a}$$

$$\{\mathbf{o}_i\}_{i=1}^{O} = h_2(\mathcal{S}, \{r_i\}_{i=1}^{R}), \tag{9b}$$

where r_i is a set of parameters that describes the ith proposed region in the 3D space. The proposal-generation stage (9a) proposes several 3D regions that may include objects inside, and the bounding box-estimation stage (9b) extracts 3D points from those proposed regions and estimates the precise object positions.

For example, PointRCNN is a recent work that follows the two-stage detection [19]. In the proposal-generation stage, PointRCNN uses PointNet++ as the backbone and recommends bin-based localization to offer regions. Bin-based localization first finds the bin associated with the center location of an object and then regresses the residual. In the bounding box-estimation stage, PointRCNN use canonical transformation to align 3D points in each proposed region and PointNet to estimate the parameters of 3D bounding boxes.

In summary, the input representation plays a crucial role in lidar-based detection. The raw-point-based representation provides complete point information, but lacks the spatial prior. PointNet has become a standard method used to handle this issue and extract features from the raw-point-based representation. The 3D voxelization-based and BEV-based representations are simple and straightforward but result in a lot of empty voxels and pixels. Feature pyramid networks with sparse convolutions can help address this issue. The range-view-based representation is more compact because the data are represented in the native frame of the sensor, leading to efficient processing, and it naturally models the occlusion. But, objects at various ranges would have significantly different scales in the range-view-based representation, as they usually requires more training data to achieve high performance. VFE introduces hybrid representations that take advantage of both the raw-point-based and 3D voxelization-based representations. The one-stage detection tends to be faster and simpler and naturally enjoys a high recall, while the two-stage detection tends to achieve higher precision [25].

Fusion-based detection

A real-time lidar sweep provides a high-quality 3D representation of a scene; however, the measurements are generally sparse and only return instantaneous locations, making it difficult for lidar-based detection approaches to estimate objects' velocities and detect small objects, such as pedestrians, at range. On the other hand, radar directly provides motion information and 2D images provides dense measurements. It is possible to naively merge detections from multiple modalities to improve overall robustness, but the benefit of this approach is limited.

Following the end-to-end fashion in deep neural networks, early fusion is believed to be a key technique to significantly improve detection performance; however, it remains an unresolved problem to design an effective early-fusion mechanism. The main challenges are 1) the measurements from each modality come from different measurement spaces; for example, 3D points are sparsely scattered in a continuous 3D space, while images contain dense measurements supported on a 2D lattice, 2) the measurements from each modality are not perfectly synchronized; lidar, cameras, and radar capture the scene at their own sampling frequencies, and 3) the sensing modalities have unique characteristics. The low-level processing of the sensor data depends on the individual sensor modality, but high-level fusion needs to consider the characteristics across multiple modalities.

Some existing early-fusion-based detection systems include MV3D [8], F-PointNet [26], PointFusion [27],

ContinuousConvolution [28], and LaserNet++ [29]. Each of these works has shown that adding image data can improve detection performance, especially when lidar data are sparse; however, the benefit is not substantial and there is no consensus on a system prototype or a basic operation. This makes it difficult for the industry to overturn previous late-fusion-based approaches.

In summary, it remains an open problem to design an early-fusion-based detection system. Most designs are based on the concatenation of intermediate features from both images and 3D point clouds, allowing the networks to figure out how to merge them. Thus far, there has been no specific design created to handle the unsynchronization issue of multiple sensors, which may be implicitly handled by learning from large-scale training data.

Data sets

High-quality data sets are required to train any of the aforementioned machine learning models. KITTI [30] is the most commonly used autonomous-driving data set, which was released in 2012 and has been updated several times since then. Most 3D object detection algorithms are validated on KITTI; however, KITTI is a relatively small data set and does not provide detailed map information. Several autonomous-driving companies have recently released their data sets, such as nuScenes' (https://www.nuscenes.org/), Argoverse's (https://www.argoverse.org/), Lyft's Level 5 AV data set (https://level5.lyft.com/dataset/), and Waymo's Open data set (https://waymo.com/open).

Evaluation metrics

To evaluate detection performance, standard evaluation metrics used in academia are the precision-recall (PR) curve and average precision (AP); however, there is no standard platform to evaluate the running speed of each model. On the other hand, industry considers more detailed evaluation metrics to check the detection performances. For example, practitioners would check the performances at various ranges, shapes, sizes, appearances, and occlusion levels to get more signals. They would also check the influences on subsequent modules, such as object tracking, future trajectory prediction, and motion planning to obtain system-level metrics.

Real-world challenges

With the growth of deep learning, the perception module has achieved tremendous improvements. Some practitioners no longer consider it the technical bottleneck of autonomous driving; however, the perception module is still far from perfect. In the next sections, we describe a series of challenges in the perception module.

High cost

A self-driving vehicle is usually equipped with one or more lidars and computing devices, such as GPUs and other specialized processors, which are expensive. Their high cost makes it formidable to maintain a scaled fleet of AVs. It remains an open problem to exploit information from real-time lidar sweeps using low-cost computation.

Tradeoffs between effectiveness and efficiency

A self-driving vehicle should react to its surroundings in real time. It would be meaningless to pursue a high-precision perception module when it introduces too much latency; however, researchers tend to focus much more on the effectiveness than on the efficiency of an algorithm.

Training data deluge

A modern perception module heavily depends on machine learning techniques, which usually need as much training data as possible; however, it takes a lot of time and computational resources to handle large-scale training data. It remains a yet-to-be-resolved problem to effectively choose a representative subset of training data from the entire data set, which would significantly accelerate product development.

Long-tail issues

There are countless traffic conditions where large-scale training data cannot cover all the possibilities. It remains an unresolved problem to find and handle corner cases, especially detecting objects that never appear in the training data.

Research conversion

In academia, research tends to design algorithms based on clean, small-scale data sets. It turns out that many effective algorithms work well for those clean, small-scale data sets but are ineffective on noisy, large-scale data sets. Meanwhile, some algorithms that work well on large-scale data sets do not work well on small-scale data sets [6]. These discrepancies can reduce the usefulness of research results when applied to real-world problems. Industry should consider providing representative data sets and perhaps even a computational evaluation platform that allows for people to compare various methods at full industrial scale.

Evaluation metrics

Objects in a scene have various levels of interactions with an AV. Incorrect estimations of some objects would lead to much bigger consequences than that of other objects; however, the PR curve and AP give uniform weights to all of the samples. Additionally, the PR curve and AP do not clearly reflect corner cases, which have only a small sample size. Thus, improving the PR curve and AP do not necessarily lead to better behavior of an AV. It is often more important to slice the test data and look at the performance over subsets of high-impact cases in addition to overall AP. A standardized simulator could also be developed to provide some system-level metrics.

Summary and open issues

The field of autonomous driving is experiencing rapid growth. Many techniques have become relatively mature; however, an ultimate solution for autonomous driving has yet to be determined. At the current stage, lidar is an indispensable sensor used for building a reliable AV, and advanced techniques for

3D point cloud processing and learning are critical building blocks for autonomous driving. In this article, we surveyed recent developments in the area of 3D point cloud processing and learning and presented their applications to autonomous driving. We described how 3D point cloud processing and learning makes a difference in three important modules in autonomous driving: map creation, localization, and perception.

With the rapid development of 3D point cloud processing and learning, the overall performances of the map creation, localization, and perception modules in an autonomous system have been significantly improved; however, quite a few challenges remain ahead. Here we briefly mention a few important open issues from a big-picture perspective.

How should we make processing and learning algorithms scalable and efficient?

Now we are still in the developing phase, and AVs are tested in a limited number of canonical routes or over a small area. In the near future, AVs may be tested on a city/country scale, which needs a city/country-scale HD map. This requires scalable algorithms that create and update HD maps. Now an AV is usually equipped with a 64-line lidar, which still produces relatively sparse point clouds. In the near future, lidar might have many more lines and produce much denser point clouds. This requires more efficient algorithms to achieve lidar-to-map localization and 3D object detection in real time.

How should we make processing and learning algorithms robust enough to handle corner cases?

We can collect large amounts of real-world sensor data and generate significant amounts of simulated sensor data, but we need to deliberately select the most representative data to improve the generality of the algorithms. At the same time, one has to face the fact that all learning algorithms depend on training data, which can never cover all of the possibilities. To address this issue, one key research area is to improve the uncertainty estimation of an algorithm because this allows for a system to react conservatively when the learned components are not confident. This requires reasoning both about the known uncertainty from the training data and also the more challenging uncertainty from cases that are not covered by the training data.

How should we develop processing and learning algorithms with a faster iteration speed?

We want more data and more complicated algorithms to achieve better performance for autonomous driving. Meanwhile, we want efficient and practical algorithms to accelerate product development, which is also critical. Practitioners in industry should collaborate closely with researchers in academia to increase the research conversion rate.

How should we evaluate processing and learning algorithms?

Currently, most of the processing and learning algorithms are evaluated on specific model-level metrics to meet the criteria of the corresponding tasks; however, these model-level metrics often do not fully correlate with system-level metrics that reflect the overall behavior. Along these same lines, the research community often focuses on improving the average performance, but there needs to be an increased focus on improving the rare long-tail cases that are really critical for a real-world system.

Acknowledgment

This article has supplementary downloadable material available at https://doi.org/10.1109/MSP.2020.2984780, provided by the authors.

Authors

Siheng Chen (schen@merl.com) received his bachelor's degree in electronics engineering from the Beijing Institute of Technology, China, in 2011, and he received his two master's degrees in electrical and computer engineering and machine learning and his doctorate in electrical and computer engineering, all from Carnegie Mellon University. Currently, he is a research scientist at Mitsubishi Electric Research Laboratories, Cambridge, Massachusetts, USA. Previously, he was an autonomy engineer at Uber Advanced Technologies Group, working on the perception and prediction systems of self-driving cars. Prior to joining Uber, he was a postdoctoral research associate at Carnegie Mellon University. He was the recipient of the 2018 IEEE Signal Processing Society Young Author Best Paper Award. His coauthored paper received the Best Student Paper Award at IEEE GlobalSIP 2018. His research interests include graph signal processing, graph neural networks, and 3D computer vision.

Baoan Liu (chensiheng1989@gmail.com) received his bachelor's degrees in electrical and computer engineering and applied physics from Shanghai Jiao Tong University, China, in 2011 and his doctorate in mechanical engineering from Carnegie Mellon University in 2016. He is the founder and chief technology officer of Precivision Technologies, Inc. (acquired by DeepMap Inc. in 2019), Pittsburgh, Pennsylvania, USA, where he leads the technology and product development of high-definition mapping and high-precision vehicle-localization solutions for autonomous driving.

Chen Feng (cfeng@nyu.edu) received his bachelor's degree in geospatial engineering from Wuhan University, China, his master's degree in electrical engineering, and his Ph.D. degree in civil engineering in 2015, both from the University of Michigan, Ann Arbor, where he studied robotic vision and learning and attempted to apply them in civil engineering. Previously, he was a research scientist in the Computer Vision Group at Mitsubishi Electric Research Labs, focusing on visual simultaneous localization and mapping and deep learning. In August 2018, he became an assistant professor jointly in the Department of Mechanical and Aerospace Engineering and the Department of Civil and Urban Engineering in the New York University Tandon School of Engineering, New York, USA, where his lab, AI4CE, aims to advance robotic vision and machine learning through multidisciplinary

use-inspired research that originates from civil/mechanical engineering domains.

Carlos Vallespi-Gonzalez (cvallespi@uber.com) received his B.Sc. degree (honors) in software engineering from La Salle School of Engineering and received his M.Sc. degree in robotics from Carnegie Mellon University. Currently, he is a senior staff engineer and technical lead manager at Uber Advanced Technologies Group, Pittsburgh, Pennsylvania, USA, leading the R&D of perception algorithms deployed in Uber's self-driving cars. Previously, he worked for more than 10 years at the National Robotics Engineering Center in the automation of agricultural machinery, including the development and deployment of fully autonomous tractors in orange orchards in Florida. He has authored or coauthored more than 30 patents as well as several publications at major computer vision conferences. His research interests are in the fields of machine learning and computer vision.

Carl Wellington (cwellington@uber.com) received his doctorate degree in robotics from Carnegie Mellon University in 2005. Currently, he is a director of engineering at Uber Advanced Technologies Group (ATG), Pittsburgh, Pennsylvania, USA. He was one of the founding members of Uber ATG when it began in 2015 and currently leads the perception and prediction teams of Uber's self-driving effort. Prior to that, he spent 10 years as an applied researcher at Carnegie Mellon's National Robotics Engineering Center (NREC) developing perception systems for autonomous agricultural and military vehicles that ranged in size from small utility carts to large tractors and multiton unmanned ground vehicles. His work at the NREC included several large-scale field trials and multiple computer vision systems that have since become commercial products. His research interests include machine learning for robotic perception.

References

[1] A. Taeihagh and H. Si Min Lim, "Governing autonomous vehicles: Emerging responses for safety, liability, privacy, cybersecurity, and industry risks," *Transp. Rev.*, vol. 39, no. 1, pp. 103–128, Jan. 2019. doi: 10.1080/01441647.2018.1494640.

[2] National Research Council, *Technology Development for Army Unmanned Ground Vehicles*. Washington, D.C.: National Academies Press, 2002.

[3] C. Badue, R. Guidolini, R. Vivacqua Carneiro, P. Azevedo, V. Brito Cardoso, A. Forechi, L. Ferreira Reis Jesus, R. Ferreira Berriel et al., Self-driving cars: A survey. Jan. 2019. [Online]. Available: arXiv:1901.04407

[4] M. Bansal, A. Krizhevsky, and A. S. Ogale, ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst. 2018. [Online]. Available: arXiv:abs/1812.03079

[5] C. Urmson, J. Anhalt, D. Bagnell, C. R. Baker, R. Bittner, M. N. Clark, J. M. Dolan, D. Duggins et al., "Autonomous driving in urban environments: Boss and the urban challenge," in *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, M. Buehler, K. Iagnemma, and S. Singh, Eds. Victorville, CA: George Air Force Base, 2009, pp. 1–59.

[6] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "LaserNet: An efficient probabilistic 3D object detector for autonomous driving," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2019, pp. 12,677–12,686. doi: 10.1109/CVPR.2019.01296.

[7] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2017, pp. 77–85. doi: 10.1109/CVPR.2017.16.

[8] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2017, pp. 6526–6534. doi: 10.1109/CVPR.2017.691.

[9] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981. doi: 10.1145/358669.358692.

[10] X.-F. Hana, J. S. Jin, J. Xie, M.-J. Wang, and W. Jiang, A comprehensive review of 3D point cloud descriptors. 2018. [Online]. Available: arXiv:1802.02297

[11] J. Peng and C.-C. Jay Kuo, "Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 609–616, July 2005. doi: 10.1145/1073204.1073237.

[12] S. Chen, D. Tian, C. Feng, A. Vetro, and J. Kovačević, "Fast resampling of three-dimensional point clouds via graphs," *IEEE Trans. Signal Process.*, vol. 66, no. 3, pp. 666–681, 2018. doi: 10.1109/TSP.2017.2771730.

[13] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graphics*, vol. 38, no. 5, pp. 1–12, Nov. 2019. doi: 10.1145/3326362.

[14] G. Li, M. Müller, A. K. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?" in *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 9266–9275. doi: 10.1109/ICCV.2019.00936.

[15] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transport. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, 2010. doi: 10.1109/MITS.2010.939925.

[16] D. Droeschel and S. Behnke, "Efficient continuous-time SLAM for 3D lidar-based online mapping," in *Proc. 2018 IEEE Int. Conf. Robotics and Automation, ICRA*, Brisbane, Australia, pp. 5000–5007. doi: 10.1109/ICRA.2018.8461000.

[17] P. J. Besl and N. D. McKay, "A method for registration of 3D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992. doi: 10.1109/34.121791.

[18] A. Y. Hata and D. F. Wolf, "Road marking detection using LIDAR reflective intensity data and its application to vehicle localization," in *Proc. 17th Int. IEEE Conf. Intelligent Transportation Systems (ITSC 2014)*, Qingdao, China, 2014, pp. 584–589. doi: 10.1109/ITSC.2014.6957753.

[19] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, Long Beach, CA, June 2019. doi: 10.1109/CVPR.2019.00086.

[20] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *Proc. 2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2017)*, Vancouver, BC, pp. 1513–1518. doi: 10.1109/IROS.2017.8205955.

[21] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660. doi: 10.1109/CVPR.2018.00798.

[22] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, Salt Lake City, UT, June 2018, pp. 4490–4499. doi: 10.1109/CVPR.2018.00472.

[23] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2017, pp. 936–944. doi: 10.1109/CVPR.2017.106.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. 14th European Conf. Computer Vision (ECCV 2016)*, Amsterdam, The Netherlands, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[25] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Computer Vision (ICCV 2017)*, Venice, Italy, 2017, pp. 2999–3007. doi: 10.1109/ICCV.2017.324.

[26] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3D object detection from RGB-D data," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2018, pp. 918–927. doi: 10.1109/CVPR.2018.00102.

[27] D. Xu, D. Anguelov, and A. Jain, "PointFusion: Deep sensor fusion for 3D bounding box estimation," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2018, pp. 244–253. doi: 10.1109/CVPR.2018.00033.

[28] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3D object detection," in *Proc. 15th European Conf. Computer Vision (ECCV 2018),* Munich, Germany, 2018, pp. 663–678. doi: 10.1007/978-3-030-01270-0_39.

[29] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez, Sensor fusion for joint 3D object detection and semantic segmentation. 2019. [Online]. Available: arXiv:abs/1904.11466

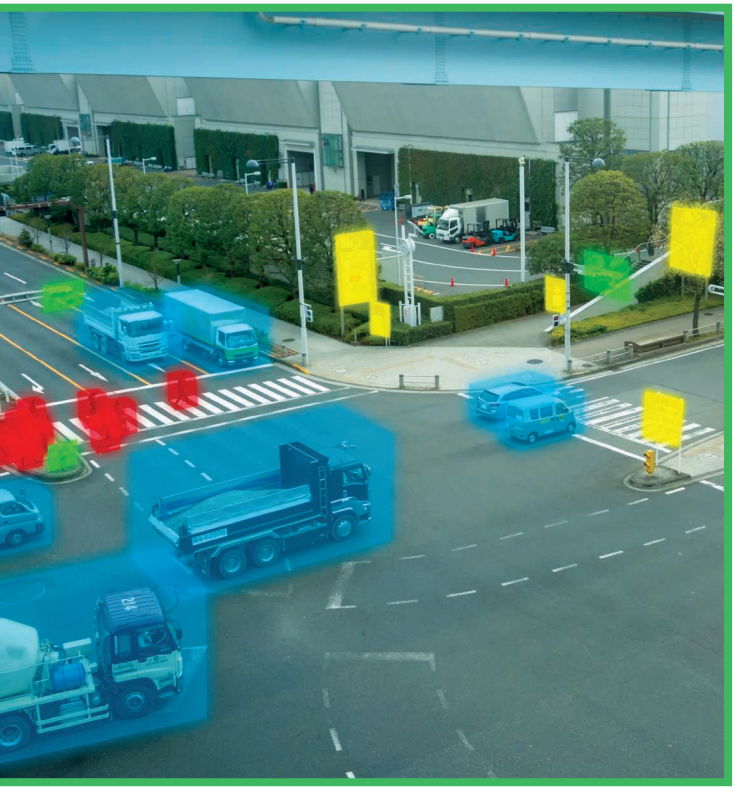
[30] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, Providence, RI, June 2012, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

SP

Tharindu Fernando, Simon Denman,
Sridha Sridharan, and Clinton Fookes

Deep Inverse Reinforcement Learning for Behavior Prediction in Autonomous Driving

Accurate forecasts of vehicle motion



©SHUTTERSTOCK.COM/MONOPOLY919

Digital Object Identifier 10.1109/MSP.2020.2988287
Date of current version: 24 December 2020

Accurate behavior anticipation is essential for autonomous vehicles when navigating in close proximity to other vehicles, pedestrians, and cyclists. Thanks to the recent advances in deep learning and inverse reinforcement learning (IRL), we observe a tremendous opportunity to address this need, which was once believed impossible given the complex nature of human decision making. In this article, we summarize the importance of accurate behavior modeling in autonomous driving and analyze the key approaches and major progress that researchers have made, focusing on the potential of deep IRL (D-IRL) to overcome the limitations of previous techniques. We provide quantitative and qualitative evaluations substantiating these observations. Although the field of D-IRL has seen recent successes, its application to model behavior in autonomous driving is largely unexplored. As such, we conclude this article by summarizing the exciting pathways for future breakthroughs.

Introduction

Consider the example shown in Figure 1. If you are driving the blue car and want to turn right at the intersection, you will try to predict the behavior of the yellow car considering aspects such as the yellow car's speed and acceleration, the distance the yellow car is from intersection, and the amount of time it would take for you to turn. You will make this decision intuitively in a split second, based on years of driving experience with similar instances as well using your intuition of human social behavior. We pose the question: How do we teach driverless cars to make these same predictions, judgments, and decisions?

Social prediction is an extraordinary feat that human drivers routinely employ to assist their decision making while traveling in close proximity to other vehicles, with conflicting objectives and incomplete information regarding the objectives of other people in the scene [1]. As such, prediction is a pivotal component in self-driving cars. Recognizing this, in August 2017, Sam Anthony, Harvard neuroscientist, chief technology officer, and cofounder of Perceptive Automat (an autonomous vehicle software

company) said, "Self-driving cars should learn human intuition and human social behavior before they can become a part of urban life" [2]. Later, in July 2018, he mentioned that one of the key challenges for safety in self-driving cars is the inability of machine learning algorithms to look at a person on the road and, irrespective of whether they are walking, driving a car, or riding a bike, predict their future behavior [3].

A major hindrance to making accurate future predictions comes from the tradeoffs that humans make between arbitrary complex factors (i.e., their surroundings, the route, behavior, risk, resource, and goal-oriented factors) when making their own decisions. Through experience as humans, we have mastered this process over our lifetime, and we seamlessly adapt our behavior. To date, making such predictions autonomously has eluded the machine learning and autonomous driving community. However, recent developments in areas such as IRL have the potential to address this limitation.

Behavior modeling in autonomous driving: A review

Model-based learning and supervised learning

The main modules in a generalized autonomous driving framework can be broadly categorized as sensor fusion, localization, prediction, and motion control. The sensory inputs are captured and fused to localize and predict the future trajectory of the agents in the local neighborhood. Utilizing these predictions, the future trajectory of an autonomous vehicle is generated and subsequently passed to the motion control subsystem to generate the control commands. The prediction module in an autonomous car uses behavior-modeling techniques, and these algorithms can be broadly categorized into model- and learning-based approaches.

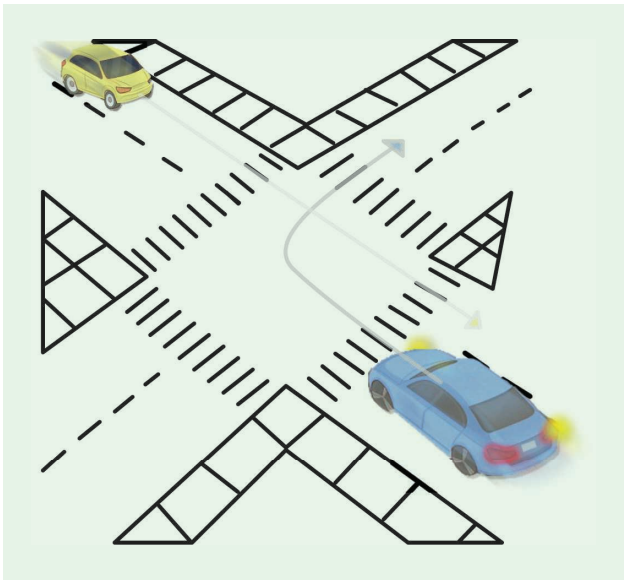


FIGURE 1. A sample driving scenario. The blue car is waiting to turn right at an intersection while there is a car coming from the opposite direction. The driver in the blue car should anticipate the future behavior of the yellow vehicle before determining its next action.

In model-based approaches, the factors that inform human behavior are hand engineered and combined to optimize a predefined objective, such as proximity to other vehicles, the number of lane changes, or the risk of taking a particular trajectory. In contrast, in learning-based systems, the underlying factors that influence human sociological factors are recovered from the data.

Among model-based approaches, Li et al. [4] uses a trajectory-planning scheme that samples trajectories from the global reference path leading to the goal state. A velocity profile generates the speed for each state along the generated path. Finally, the best path is chosen based on a cost function considering safety and comfort. The trajectory-generation algorithm of [5] computes a course by minimizing the distance to the goal state, the distance to the centerline path, and maximizing the proximity to the obstacles. This was extended in [6] by augmenting the proximity cost to discourage picking paths that are close to dangerous drivers, cyclists, and pedestrians. Despite these attempts, it is infeasible to hand engineer a cost function that can consider all of the factors that influence the future behavior of people in the vicinity of an autonomous vehicle.

As opposed to model-based systems, learning-based systems try to automatically recover these factors from the data. A popular family of learning-based algorithms is supervised learning. These algorithms apply the past observed trajectories of the autonomous agent(s) in the local neighborhood and learn to predict the future trajectory of the autonomous agent. The process is data driven, as the model minimizes the distance between the predicted and ground-truth trajectories using a predefined loss function, such as the mean square error (MSE) [7].

In [8], the authors propose the utilization of social pooling to capture interdependencies between neighboring vehicles in motion. The authors encode the past trajectories of the autonomous vehicle as well as neighboring agents using long short-term memory (LSTM) networks [9], and to capture the interdependencies of nearby agents they pool out hidden states of LSTM based on their spatial configuration in the scene. These states are subsequently passed through a series of convolutional and pooling layers, and the future trajectory is generated by a decoder LSTM. The framework is trained to minimize the negative log-likelihood loss between the predicted and ground-truth trajectories. In [10], the authors extend this encoder–decoder LSTM framework for joint trajectory prediction and maneuver classification. They illustrate that maneuver-dependent trajectory prediction is comparatively more resilient than predicting the trajectory alone.

Most recently, Zhao et al. [1] proposed a framework that encodes the past trajectories of neighboring agents using LSTMs and captures the scene context using convolutional neural networks (CNNs). Then, this information is fused and passed to a decoder LSTM to generate the future trajectory of the autonomous agent. This framework is learned through a combination of MSE and adversarial loss, which is achieved through a generative adversarial network (GAN) learning process [11].

In addition to [1], which has achieved favorable results on both highway driving and pedestrian trajectory data sets, it is worth noting that supervised learning systems such as Soft + Hardwired Attention [12] and Social GAN [13] have been proposed to automatically recover human social navigation behavior in crowded environments. However, these systems were developed and evaluated using pedestrian trajectory data.

Generative adversarial imitation learning

Despite their reasonable success, supervised learning approaches cannot recover the underlying factors that influence human social behavior [14], as they operate using a predefined cost function, which does not fully capture human reasoning. There exists another class of algorithms, that being generative adversarial imitation learning (GAIL) [15], which seeks to directly mimic the expert's policy and has been extensively applied for autonomous driving tasks [16]–[18].

One of the most popular approaches used for solving IRL problems is maximum entropy (MaxEnt)-IRL, where the expert behavior is modeled as a distribution to the one of the highest entropy.

Let the decision-making process of the pedestrians be modeled as a Markov decision process (MDP) [19]. The MDP $M = [S, A, \tau, R]$ is composed of state space S: a set of possible actions; A: a transition matrix; τ: a reward function; and R: a policy. π defines the selection of an action, given a particular state. We are presented with a set of demonstrations $D = [\zeta^1, \zeta^2, \ldots, \zeta^N]$, where each demonstration ζ^i is composed of state (s_t) and action (a_t) pairs, $\zeta^i = [s_0, s_1, \ldots, s_{T_{\text{obs}}}]$. Then the GAIL objective is denoted by

$$\min_\theta \max_w V(\theta, w) = \mathbb{E}_{\pi_\theta}[\log D_w(s, a)] + \mathbb{E}_{\pi_E}[\log D_w(s, a)], \tag{1}$$

where policy π_θ is a neural network parameterized by θ, which directly generates the policy imitating π_E, and D_w is the discriminator network parameterized by w, which tries to distinguish state–action pairs from π_θ and π_E. $\mathbb{E}_\pi[f(s, a)]$ denotes the expectation of f over the state–action pairs generated by policy π.

Numerous works [16]–[18] have utilized GAIL for predicting trajectories in simulated highway driving scenarios. In [17], the authors use eight features including vehicle speed, length, lane curvature, and distance-to-lane markers as state features and, employing the GAIL formulation, they predicted the relevant actions given this state representation. The authors in [16] propose a system to leverage variability among different expert demonstrations. They apply the information-maximization theorem to automatically discover and disentangle latent factors in the underlying expert demonstrations. In our previous work [18], we propose the use of neural memory networks (NMNs) [20] to capture relationships at a subtask level and determine how they are temporally linked in a given expert demonstration.

Similar to supervised methods, GAIL does not attempt to recover the reward function. Instead, it attempts to directly mimic the expert's policy. Hence, its applicability to environments with data constraints and its generalizability to new environments remain questionable [21].

IRL

IRL, however, has shown promise in being able to address the deficiencies of supervised and imitation learning. Unlike GAIL, which directly tells the learner how to act, IRL recovers the underlying reward function, which provides a better understanding regarding modeled behavior [21]. In an IRL framework, given a set of demonstrations $D = [\zeta^1, \zeta^2, \ldots, \zeta^N]$, we recover reward function R followed by the demonstrators in the samples. Then, using the recovered reward function, a machine can imitate natural human behavior.

IRL-based behavior-prediction techniques segregate the underlying semantics of the scene such that the goal or intent of the agents can be recovered from the modeled reward function. This makes the system more tractable and able to generalize to new environments [21] while demonstrating more accurate predictions into the distant future [22], [23].

One of the most popular approaches used for solving IRL problems is maximum entropy (MaxEnt)-IRL [24], where the expert behavior is modeled as a distribution to the one of the highest entropy [14]. The MaxEnt formulation assumes that the reward function can be calculated as a weighted linear combination of features $\Phi(s)$, where Φ is a function that outputs the features of the state s and the set of weights θ:

$$R(\Phi(s)) = [\theta]^\top \Phi(s). \tag{2}$$

Capitalizing on the merits of IRL, several works [25]–[27] have applied it for behavior prediction. In [25], the authors first cluster the trajectories in the training set and train a multiclass classifier to label the cluster identity of a given trajectory. The authors utilize hidden Markov models to transform the observed trajectories in each cluster into a set of finite states. Then they recover the reward matrices R_i for each cluster i using an IRL framework. In the test phase, given an observed partial trajectory, they first predict the cluster identity, and using the recovered reward matrix of that particular cluster and the Viterbi algorithm [28], they find the most probable sequence of states for its future trajectory.

In [26], the authors investigate the tradeoff between social accessibility and task-related constraints for navigation. For each demonstrated trajectory, they define an acceptability-dependent criteria based on its social acceptability. Then, combining this feature together with other task-related features such as acceleration, steering, velocity, and deviation from lane centers, they apply the MaxEnt algorithm to learn different acceptability-dependent behaviors.

The authors in [27] address the exploding state-space problem in IRL. They propose to replace the RL inner loop in IRL

with deep Q-networks to extend the IRL framework to larger state spaces.

Despite these capabilities, the original MaxEnt-IRL framework [24] and subsequent works [25]–[27] assume that the reward function can be calculated as a weighted linear combination of the features [21]. This linear mapping from features to the reward severely restricts the reward structure that can be modeled [23].

D-IRL

The recent works of Wulfmeier et al. [14] extend IRL to a deep learning setting, lifting the MaxEnt-IRL constraints and permitting a nonlinear mapping, which allows more flexibility for the learned reward structure. Hence,

$$R(\Phi(s)) = f(\theta, \Phi(s)), \tag{3}$$

where f is a nonlinear function. The authors of [14] try to maximize the log-likelihood of the demonstrated trajectories:

$$L(\theta) = \log \prod_{\zeta^i \in D} P(\zeta^i, \theta), \tag{4}$$

where $P(\zeta^i, \theta)$ is the probability of the trajectory ζ^i in demonstration D and

Algorithm 1. The MED-IRL.

Input:
D: Demonstrations; S: state space; A: set of possible actions; τ: transition matrix; γ: discount factor for the value-iteration algorithm; and α: learning rate of the deep neural network.
Output: Reward network parameters θ^*

1: **for** iteration $i = 1$ to M **do**
2: $R^i(\Phi(s)) = f(\theta^i, \Phi(s)) \quad \forall_{s \in S}$ // forward pass in the reward network
3: $\pi^i = Value\_Iteration(R^i, S, A, \tau, \gamma)$ // planning step
4: $\mathbb{E}[\mu^i] = compute\_SVF(\pi^i, S, A, \tau)$
5: $\frac{\delta L_D^i}{\delta R^i} = \mu_D - \mathbb{E}[\mu^i]$ // gradient calculation
6: $\theta^{i+1} = back\_propagate\left(\theta^i, \frac{\delta L_D^i}{\delta R^i}, \alpha\right)$ // reward network update
7: **end**
8: **return** θ.

Algorithm 2. The value iteration.

Input:
R: Current approximation of the reward function; S: state space; A: set of possible actions; τ: transition matrix; and γ: discount factor.
Output: ϕ

1: $V(s) = -\infty$ **repeat**
2: $V_t(s) = V(s)$
3: $Q(s, a) = r(s, a) + E_{\tau(s,a,s')}[V(s')]$
4: $V(s) = \max_a(Q_i(s, a))$
5: **until** $\max_s(V(s) - V_t(s)) < \epsilon$;
6: **return** $\phi(a|s) = e^{Q(s,a) - V(s)}$.

$$\frac{\delta L_D}{\delta \theta} = \mu_D - \mathbb{E}[\mu] \frac{\delta R(\Phi(s))}{\delta \theta}, \tag{5}$$

where μ_D and $\mathbb{E}[\mu]$ are the state visitation frequencies from the demonstrated and inferred reward functions, respectively. Algorithm 1 illustrates the process of refining the reward network in the MaxEnt-deep-IRL (MED-IRL) framework proposed in [14], where γ is a discount factor for the value-iteration algorithm (see Algorithm 2), and α is the learning rate of the deep NN (DNN). In each iteration i of the algorithm, they first evaluate the reward based on the state features $\Phi(s)$ and the current reward network parameters θ^i. Then, using the current reward function, they apply value iteration [24] to solve the forward RL problem, determining the current policy π^i based on the current approximation of the reward $R^i(\Phi(s))$ and the transition matrix τ. The value-iteration algorithm is illustrated in Algorithm 2. Within Algorithm 1, line 5 computes the gradient with respect to the reward, which determines how to update the reward network parameters (line 6). The process is presented in Figure 2.

Recently, MED-IRL has been applied for autonomous driving tasks [14], [23], [29]. Wulfmeier et al. [14] demonstrated the utility of fully convolutional neural (FCN) networks for mapping the lidar scans of urban environments to traversability maps, which are automatically learned through MED-IRL. The proposed multiscale fully convolutional network (MSFCN) architecture (see Figure 3) employs a pooling-based substream to capture spatial invariant features from lidar and a fully convolutional network (FCN) stream, which preserves the location information from the input data. The proposed system was able to learn end-to-end mapping from raw inputs to a reward map, utilizing more than 25,000 trajectories from more than 120 km of driving.

In [23], Zhang et al. couple low-level lidar scan features together with kinematic features to augment the performance of the MED-IRL framework. The network architecture used in [23] is shown in Figure 4. The authors in [23] argue that, in motion planning, human drivers consider kinematic aspects such as the vehicle's current velocity and past trajectory in addition to evaluating the spatial attributes of the environment, such as the distance to obstacles. Hence, they propose to augment the MED-IRL framework of [14] to incorporate this information in two stages. In the first stage, they apply a four-layer FCN to encode a color-coded point cloud. In the second stage, the authors utilize two feature maps encoding each grid cell, that is, the x and y positions of the grid cell in a vehicle centered, world-aligned frame. Another three feature maps are generated encoding kinematic information: Δx, Δy, and the curvature of the input trajectory. Their evaluations demonstrated greater robustness in predictions compared to both supervised learning and MaxEnt-IRL systems.

In [29], the authors refine the MaxEnt [24] formulation, considering both linear and nonlinear (MED-IRL) settings to maximize the entropy of the joint distribution over short data pieces. They show that long demonstrations

are hard to use in a model-free IRL setting, as the prediction error is accumulated over long time horizons. However, this system is validated in simulations of highway driving where the environment is simplified compared to complex urban driving.

Considering that the aforementioned systems do not account for the motion of the neighbors when predicting future motion, most recently, we proposed a novel MED-IRL framework for pedestrian trajectory prediction. The trajectories of the agent of interest and the neighboring trajectories are encoded

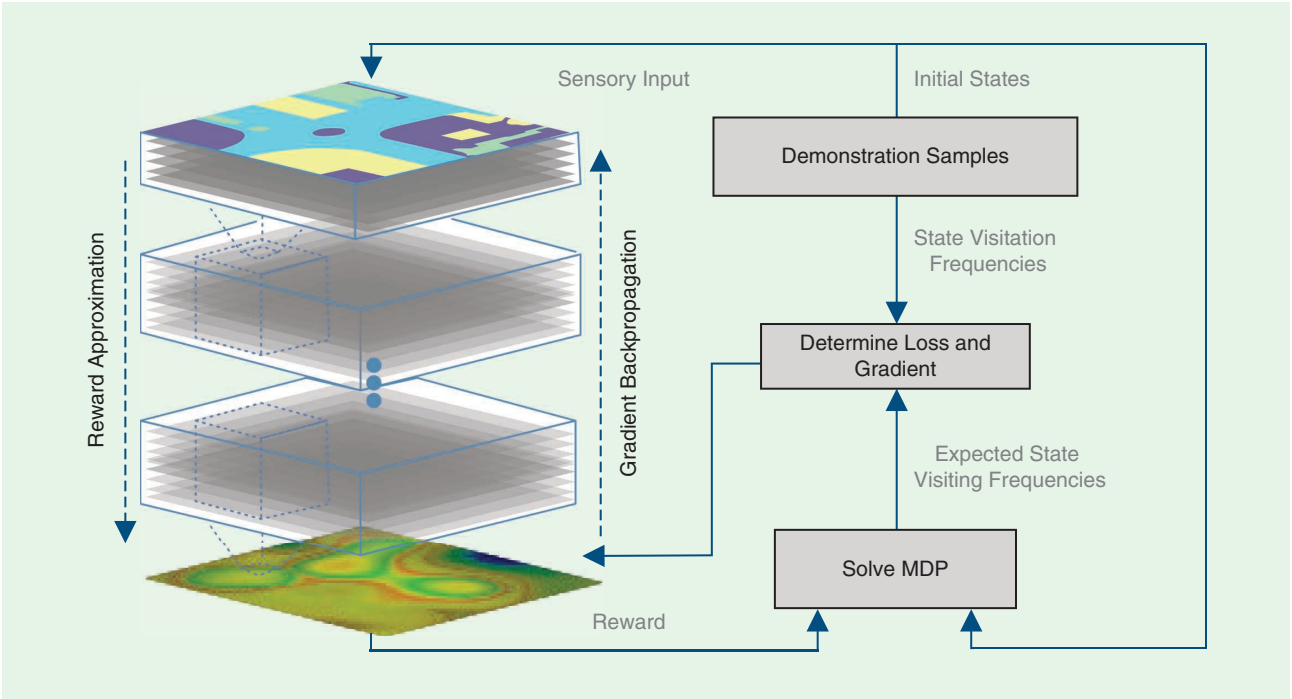


FIGURE 2. The schema proposed in [14] for training DNNs using MaxEnt-IRL. Given a set of demonstrations, a DNN is utilized to approximate the reward function. Then we calculate the difference between the state visitation frequencies from the demonstrated trajectories and from the inferred reward function. This difference acts as the network's loss and we backpropagate its gradients, updating the network.

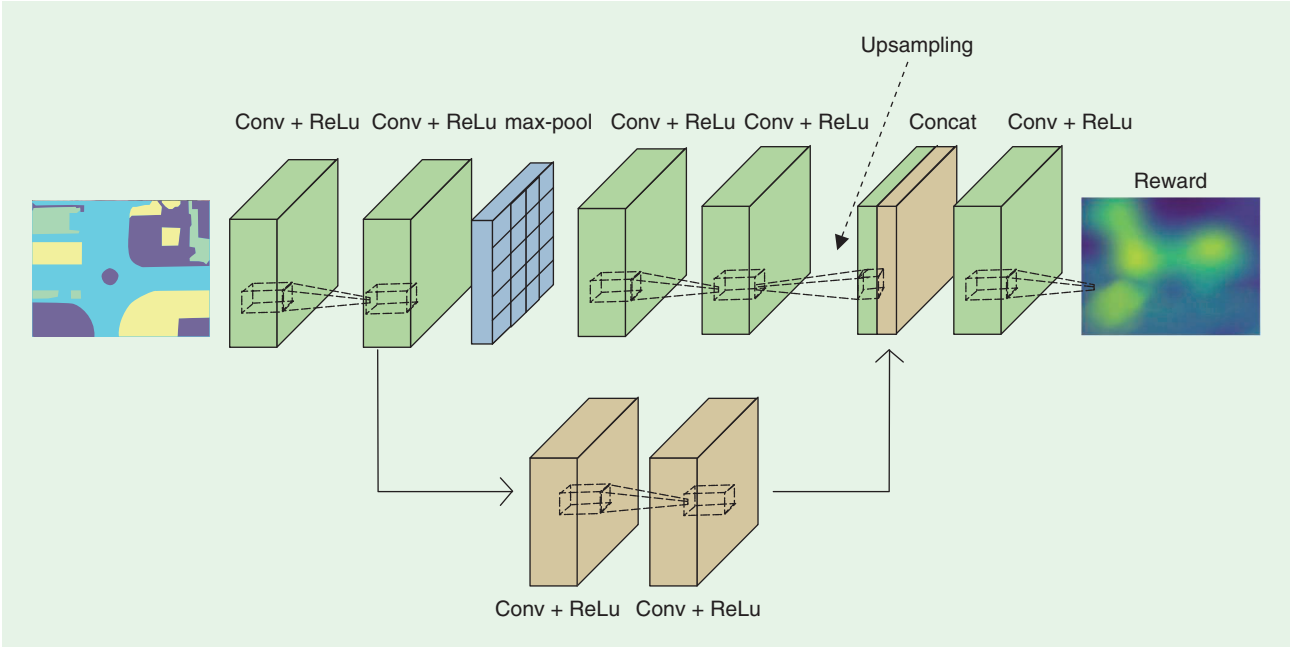


FIGURE 3. An illustration of the FCN architecture used by [14] as the reward network. By using a two-stream architecture, with an FCN-based mainstream and a pooling-based substream, we propose capturing spatially variant and invariant features, respectively. conv: convolutional; concat: concatenated; ReLu: rectified linear unit.

using LSTMs. Then we develop a combination of soft and hard-wired attention [12] to aggregate the encoded trajectory information to a context vector.

As the reward network, similar to [14] and [24], we utilize an FCN. We first generate an empty map G of the environment and then assign values $\hat{h}_t^k$ from the pedestrian of interest k and $\tilde{h}_j^n$ from the neighbors to grid G based on the Cartesian coordinates that the specific hidden state comes from (i.e., based on the position of the trajectory). Then, using the FCN, we map G to a reward map R. This architecture is illustrated in Figure 5.

Experimental evaluations

In this section, we report the evaluation results of current state-of-the-art supervised learning, GAIL, linear-IRL (L-IRL), and D-IRL systems on the publicly available Next Generation Simulation (NGSIM) trajectories US-101 [31] data set and a portion of the nuScenes data set [32].

Data sets

The NGSIM US-101 [31] data set contains trajectories of real freeway traffic captured from fixed overhead cameras placed over a 640-m span of US-101, recorded at 10 Hz over a 45-min period. This data set consists of more than 6,000 vehicle annotations and provides varying traffic conditions where the traffic flow varies from mild to moderate to congested.

In addition, we use trajectories from the nuScenes data set [32], which is captured in multiple cities, from multiple sensors including six cameras, a lidar, five radars, a GPS sensor, and an inertial measurement unit sensor. The complete data set contains 15 h of driving data covering 242 km with dense traffic and highly challenging driving situations. The data set is divided into 1,000 scenes by the database authors, and to ensure a compatible size between the two evaluations, we use only scenes 61, 69, and 234. To generate the trajectories, we used object-bounding box annotations, and the center of the bounding box is taken to be the object position at each time step.

We report the results in terms of the root-MSE (RMSE) of the predicted trajectory with respect to the ground-truth future trajectory over different prediction horizons ranging from 1 to 5 s. Similar to [17], we simulate the behavior prediction through a trajectory-prediction task where we select each car, iteratively, to be the autonomous car and predict the future behavior of this car, exploiting the past behavior of neighboring vehicles.

Evaluated models

The following models were assessed:

- *Supervised learning*: we use the models of [8] [convolutional social (CS)-LSTM] and [1] [Multi-Agent Tensor Fusion (MATF)-GAN].
- *GAIL-gated recurrent unit (GRU)*: we consider the GAIL model from [17].
- *L-IRL*: we use the L-IRL model proposed in [22].
- *D-IRL*: to demonstrate the utility of D-IRL models, we use the models proposed in [14] (D-IRL), [23] [deep kinematics (DK)-IRL], and [30] [deep neighborhood (DN)-IRL].

For all of the considered systems, similar to [1], the neighbors appearing in the 640-m span are studied in the reasoning and prediction process. In the original works of [14] and [23], the authors utilize terrain maps captured using lidar. As this information is not available in the NGSIM US-101 data, we use the semantic segmentation of the scene, which indicates the traversable lanes, and for the nuScenes we use the traversability maps generated through lidar.

For the GAIL-GRU baseline, we follow the policy network architecture of [17], which uses five feedforward layers that decrease in size from 256 to 32 neurons, and an additional GRU layer consisting of 32 neurons. We use the implementation released by the authors (https://github.com/sisl/gail-driver).

For the D-IRL and L-IRL systems, we consider a grid size of 120 × 120 and map the x, y coordinates to grid cells. As they generate a probability distribution over the cells, we

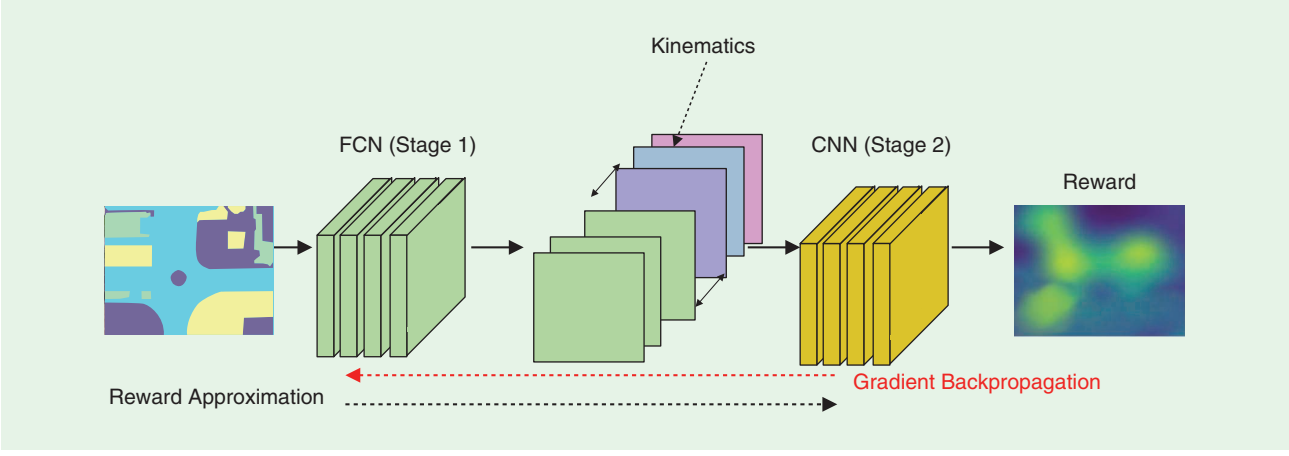


FIGURE 4. The two-state architecture proposed in [23]. We captured environmental context from input terrain maps in the first-stage network, and the resulting feature maps are concatenated with the kinematic context in the second-stage network, which outputs a reward representation. The difference between the state visitation frequencies from the demonstrated trajectories and the learned policy is used to compute the gradients for backpropagation.

sample 1,000 trajectories from the distribution and measure the average RMSE between the ground truth and samples. We map predictions back to the image coordinate space for clear comparison. For the D-IRL baseline, we strictly adhere to the recommendations of the authors and used the FCN architecture introduced in [14]. This takes the semantic segmentation map as the input and generates the reward map purely based on the environment.

For the DK-IRL baseline, we follow the two-stage architecture of [23] and used the FCN model from D-IRL as the network for the first state. For the second stage, following [23] we generate two feature maps encoding each grid cell, that it, the x and y positions of the grid cell in a pedestrian centered, world-aligned frame. Another three feature maps are generated encoding the kinematic information: Δx, Δy, and the input trajectory curvature. We use the codebase released by the authors (https://github.com/yfzhang/vehicle-motion-forecasting), which also provided an implementation of the D-IRL framework in [14].

For the DN-IRL baseline, as per [30] we consider the trajectories of the 10 closest neighbors in the front, left, and right directions. If there are more than 10 neighbors in any direction, we choose the closest nine and the mean trajectory of the rest. If there are less than 10 neighbors, we create a dummy trajectory such that we have 10 neighbors for each direction and set the dummy trajectory hardwired weights to zero. For all the LSTMs, we use a hidden-state dimension of 50 units.

Results

Quantitative evaluations of the performance of the considered frameworks are presented in Table 1. To clearly demonstrate the utility of the D-IRL framework, we perform the trajectory predictions under different prediction horizons, estimating the trajectories from 1- to 5-s ahead. For each trajectory, we use the coordinates (positions) for the previous 3 s as the observed portion of the trajectory. We evaluated the performance for different prediction horizons, predicting trajectories from 1 s ahead to 5 s ahead. We report the RMSE as the error metric (lower is better). For clarity, supervised learning methods are shown with a blue background, the GAIL-GRU method with an orange background, the L-IRL method with a yellow background, and the D-IRL methods with a green background.

From Table 1, we observe that the performance of the supervised learning methods degrades when predicting behavior into the distant future. This is caused by deficiencies in the supervised learning structure, as these models try to directly map inputs to targets without paying attention to the end goal or intent of the driver. Furthermore, the linear IRL

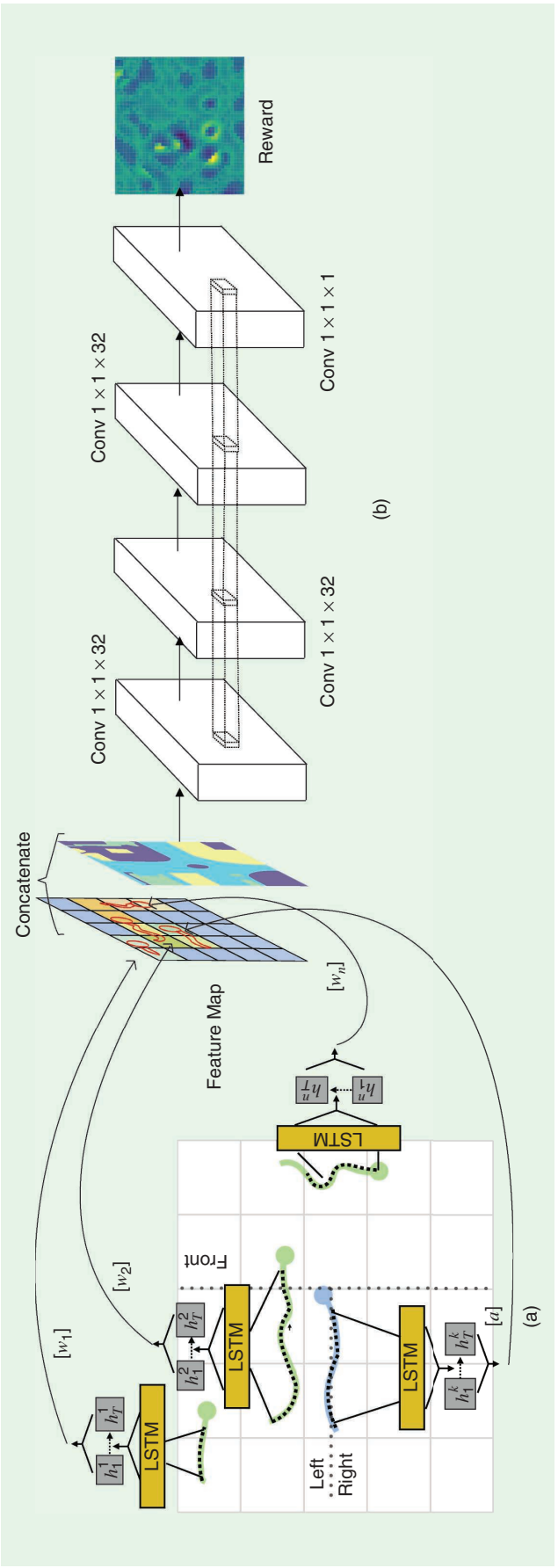


FIGURE 5. (a) The architecture used to embed neighborhood context: the trajectory of the pedestrian of interest is shown in blue, with three neighbors in green. The heading directions are indicated with circles. The trajectories are encoded using LSTMs, with soft attention used to embed information from the pedestrian of interest and hardwired attention used for the neighbors. Next, a feature map is created to spatially embed this information based on the cartesian points of each trajectory. (b) The architecture of the four-layer FCN used to map the feature map G to the reward map R. The first three layers contain 32 1×1 convolution kernels with an ReLu activation, and the final layer contains one 1×1 convolution kernel.

system fails to generate satisfactory results due to constraints with the learned feature-to-reward mapping structure.

With the introduction of the nonlinear reward mapping from the D-IRL framework, we observe a slight performance increase with the DK-IRL methods compared to the L-IRL method of [22]; however, these methods fail to outperform the MATF-GAN system. This is a result of the lack of input information that DK-IRL frameworks receive regarding the neighborhood context, which is a highly influential factor when navigating in congested environments. However, the LSTM-based neighborhood embedding scheme in the DN-IRL framework is able to capture a notion of the neighborhood, resulting in superior performance. We observe a substantial performance increase, especially when predicting behavior into the distant future.

Due to the architectural differences between the GAIL-GRU and DN-IRL methods, their performance is not directly comparable. Further evaluation is necessary with identical network architectures to compare their relative strengths and weaknesses. However, such comparisons are currently constrained by the nonpublic availability of such advanced GAIL architectures.

Qualitative results of the DN-IRL method and the recovered reward representation for four examples from the NGSIM data set [31] are given in Figure 6. By analyzing the predictions in Figure 6 we observe that the DN-IRL method achieves good performance when predicting lengthier trajectories. Furthermore, we observe that the DN-IRL method, by virtue of its neighborhood modeling, is capable of predicting complex maneuvers such as lane changes and overtaking.

Limitations and open research challenges

Although MED-IRL provides flexibility and robustness for behavior anticipation, it has yet to be widely adopted for autonomous driving systems. Despite great potential for generating realistic hypotheses of human behavior, there are several open research questions requiring further study.

Table 1. The evaluation results for NGSIM US-101 [31] and nuScenes [32].

Results for the NGSIM US-101 Data Set

	Prediction Horizon				
Method	**1 s**	**2 s**	**3 s**	**4 s**	**5 s**
CS-LSTM [8]	0.61	1.27	2.09	3.1	4.37
MATF-GAN [1]	0.66	1.34	2.08	2.97	4.13
GAIL-GRU [17]	0.69	1.51	2.55	3.65	4.71
L-IRL [22]	1.12	2.29	2.31	3.38	4.45
D-IRL [14]	1.35	2.57	2.83	3.69	4.88
DK-IRL [23]	1.09	2.05	2.27	2.91	4.4
DN-IRL [30]	0.54	1.02	1.91	2.43	3.76

Results for the nuScenes Data Sets

	Prediction Horizon				
Method	**1 s**	**2 s**	**3 s**	**4 s**	**5 s**
Social GAN [13]	0.93	1.49	2.67	3.32	5.89
GAIL-GRU [17]	1.39	2.02	2.98	4.05	5.87
L-IRL [22]	1.44	2.68	3.57	3.59	5.51
D-IRL [14]	1.61	2.93	3.12	4.21	5.19
DK-IRL [23]	1.23	2.53	3.03	3.52	4.94
DN-IRL [30]	0.75	1.25	2.35	2.59	4.55

To the best of our knowledge, the only D-IRL framework that considers complex, dynamic environments with multiple agents in motion is presented in [30]. Yet, in [30], the temporal nature of the agent's motion is ill-represented through the current formulation of the reward network. Furthermore, the hardwired attention formulation of [30] is perhaps less impactful in a driving context than it is for pedestrian motion, for which [30] is originally proposed. In addition, the type of neighbor, i.e., a car, truck, motorbike, or pedestrian, may also be important, yet it is not considered. Hence, further investigation is required to determine effective ways to learn the spatiotemporal contextual factors that impact an agent's behavior when there are large numbers of different types of mobile agents.

Another interesting pathway for investigation is a methodology to capture subtle differences among different expert demonstrations via the reward network formulation. There are often clear differences in expert behavior due to varied user preferences and domain knowledge, even though all experts perform the same task. Li et al. [16] learns these differences by conditioning the learned low-level actions on a latent variable and discriminates expert demonstrations based on their structure in the GAIL setting. In our previous work, we investigate using NMNs to capture these factors in GAIL [18] and supervised learning settings [33]. The viability of these methods in the MED-IRL setting is an open question. In addition, MED-IRL assumes a fixed transition model τ; however, this formulation may limit the robustness of learned policies when there are changes in dynamics such as significant environmental variations (i.e., changes in weather or traffic conditions).

The work of Fu et al. [21] investigates applying an adversarial IRL (A-IRL) framework to disentangle the policy and reward function. A-IRL has been formulated by combining GAIL and guided cost learning (GCL) [34]. Compared to GAIL, it learns both the reward function and the policy, and compared to GCL, it learns in an adversarial learning setting. Although the evaluations in [21] demonstrate increased robustness in high-dimensional environments with significant domain shifts between demonstrations, further investigation is required to enable the method to mitigate suboptimality in the given samples when the demonstrators do not follow optimal behavior.

In the current formulation of the MED-IRL algorithm, value iteration (see Algorithm 2) is used to solve the forward RL problem in the loop. Numerous works have demonstrated that value iteration has a very slow convergence rate [23], [35]. In the work of Zhang et al. [23], the authors utilized a technique called *annealed softmax* where they artificially increase the probability of the most likely action being chosen. However, more investigation is required to determine the best ways to speed up the convergence of value iteration.

In addition, little effort has been made to leverage the multimodal data captured by autonomous vehicles. In [14] and [23],

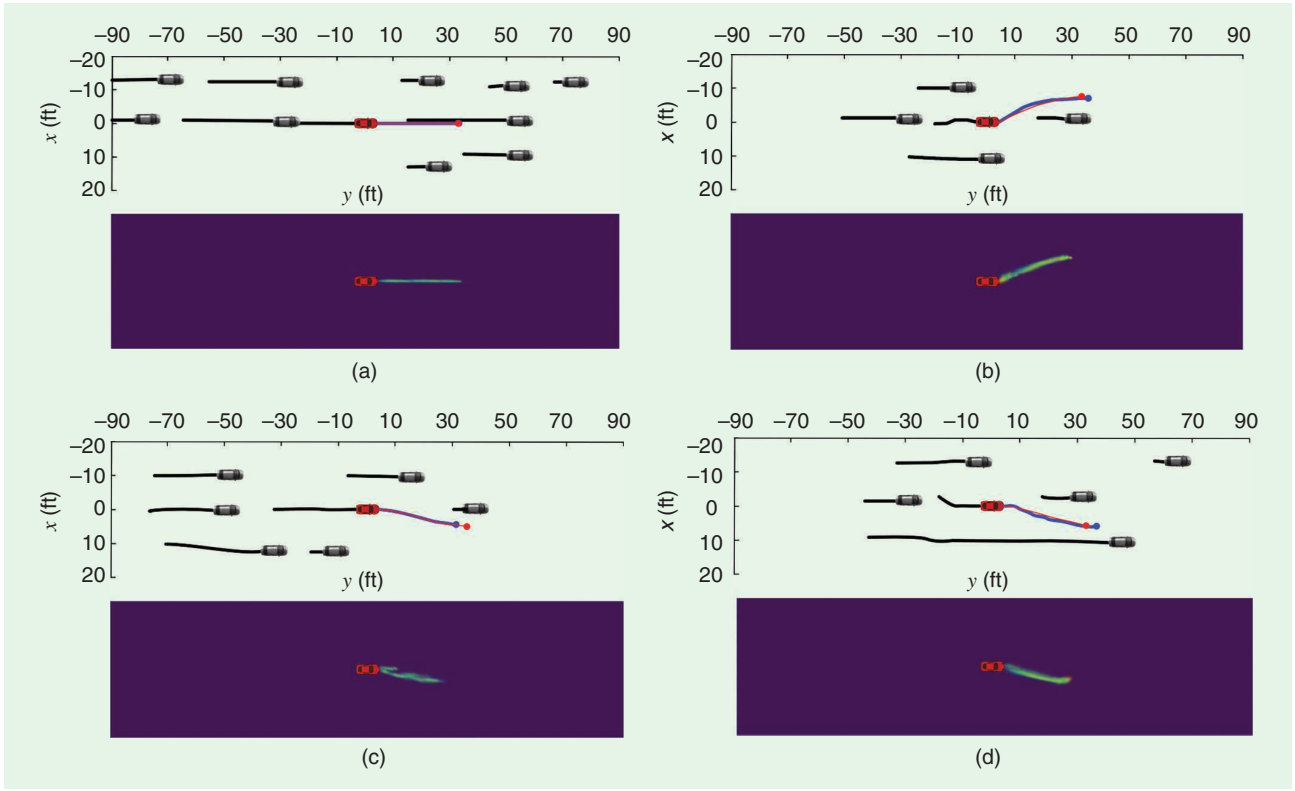


FIGURE 6. The qualitative results of the DN-IRL method: (a) a lane following behavior is shown in the ground truth and a higher probability in the prediction is given for this behavior; (b)–(d) predictions for scenarios where an overtaking behavior exists in the ground truth. The observed part of the trajectories is shown in black. The autonomous agent is indicated by the red car, and the neighboring vehicles are denoted by black vehicles. The ground-truth future trajectory is given in blue, while the predictions are in red. In the probability map, the colors from blue to yellow indicate low to high probability.

the terrain maps are captured using lidar scans; however, systems can be designed to exploit the complementary information available through sources such as red-green-blue, infrared and thermal cameras, and radar sensors, which are readily available in a typical autonomous driving setting. These sensors could provide information at different granularities and different ranges, enabling better neighborhood modeling for decision making.

The lack of availability of well-annotated public benchmarks poses another hinderance. Only a limited number of data sets, such as nuScenes [32] and KITTI [36], have annotations relating to other agents in the scene, including pedestrians and cyclists. Hence, introducing public benchmarks with richer annotations could promote the swift implementation and evaluation of behavioral prediction systems for real-world autonomous driving systems.

Conclusions

In this article, we presented an overview of the current state-of-the-art techniques applied for behavior prediction in autonomous driving. We reviewed popular approaches, including both model-based and supervised learning and GAIL, IRL, and D-IRL methods. We quantitatively and qualitatively evaluated these frameworks on two public driving benchmark data sets and demonstrated the utility of D-IRL, especially when making predictions into the distant future. Despite the undoubted potential of D-IRL methods, there are several shortcomings at present, and a number of promising research avenues for future breakthroughs were discussed to further advance the field and realize the goal of fully autonomous vehicles.

Authors

Tharindu Fernando (t.warnakulasuriya@qut.edu.au) received his B.Sc. (special degree in computer science) and Ph.D. degrees from the University of Peradeniya, Sri Lanka, and Queensland University of Technology (QUT), Brisbane, Australia, respectively. He is currently a postdoctoral research fellow in the Speech, Audio, Image, and Video Technologies research program with the School of Electrical Engineering and Computer Science at QUT, Brisbane, Australia. His research interests focus mainly on human behavior analysis and prediction. He is a Member of IEEE.

Simon Denman (s.denman@qut.edu.au) received his B.Eng. degree in electrical engineering and his Ph.D. degree in the area of object tracking from Queensland University of Technology (QUT), Brisbane, Australia. He is currently a senior lecturer with the School of Electrical Engineering and Computer Science at QUT, Brisbane, Australia. His research interests include intelligent surveillance, video analytics, and video-based recognition. He is a Member of IEEE.

Sridha Sridharan (s.sridharan@qut.edu.au) received his B.Sc. degree in electrical engineering and his M.Sc. degree in

communications engineering, both from the University of Manchester, United Kingdom, and his Ph.D. degree from the University of New South Wales, Kensington, Australia. He is currently with Queensland University of Technology (QUT), Brisbane, Australia, where he is a professor with the School Electrical Engineering and Computer Science. He has authored more than 600 publications in the areas of image and speech technologies. He leads the Speech, Audio, Image, and Video Technologies research program at QUT, with a strong focus in the areas of computer vision, pattern recognition, and machine learning. He is a Life Member of IEEE.

Clinton Fookes (c.fookes@qut.edu.au) received his B.Eng. degree in aerospace/avionics and his M.B.A. and Ph.D. degrees from Queensland University of Technology (QUT), Brisbane, Australia. He is currently a professor and the head of discipline for vision and signal processing with the Science and Engineering Faculty at QUT, Brisbane, Australia. He serves on the editorial board of *IEEE Transactions on Information Forensics and Security*. His research focus is in the areas of computer vision, machine learning, and pattern recognition. He is a Senior Member of IEEE, an Australian Institute of Policy and Science Young Tall Poppy, an Australian Museum Eureka Prize winner, and a Senior Fulbright Scholar.

References

[1] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, "Multi-agent tensor fusion for contextual trajectory prediction," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019, pp. 12,126–12,134. doi: 10.1109/CVPR.2019.01240.

[2] S. Anthony, "Self-driving cars still can't mimic the most natural human behavior," Quartz, 2017. [Online]. Available: https://qz.com/1064004/self-driving-cars-still-cant-mimic-the-most-natural-human-behavior/

[3] "Introducing perceptive automata: Human intuition for self-driving cars," Medium, 2018. [Online]. Available: https://medium.com/perceptive-automata/introducing-perceptive-automata-human-intuition-for-self-driving-cars-3d2aaa05c083

[4] X. Li, Z. Sun, D. Cao, D. Liu, and H. He, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles," *Mech. Syst. Signal Process.*, vol. 87, pp. 118–137, Mar. 2017. doi: 10.1016/j.ymssp.2015.10.021.

[5] V. Cardoso, J. Oliveira, T. Teixeira, C. Badue, F. Mutz, T. Oliveira-Santos, L. Veronese, and A. F. De Souza, "A model-predictive motion planner for the IARA autonomous car," in *Proc. 2017 IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 225–230. doi: 10.1109/ICRA.2017.7989028.

[6] A. B. D. Manocha, "Behavior modeling for autonomous driving," in *Proc. AAAI Fall Symp. Reasoning and Learning Real-World Systems Long-Term Autonomy (LTA)*, 2018, pp. 16–21.

[7] E. L. Lehmann and G. Casella, *Theory of Point Estimation*. New York: Springer-Verlag, 2006.

[8] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1468–1476. doi: 10.1109/CVPRW.2018.00196.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[10] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based LSTMs," in *Proc. 2018 IEEE Intelligent Vehicles Symp. (IV)*, pp. 1179–1184. doi: 10.1109/IVS.2018.8500493.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Int. Conf. Advances Neural Information Processing Systems*, 2014, pp. 2672–2680. doi: 10.5555/2969033.2969125.

[12] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Soft+ hardwired attention: An LSTM framework for human trajectory prediction and abnormal event detection," *Neural Netw.*, vol. 108, pp. 466–478, Dec. 2018. doi: 10.1016/j.neunet.2018.09.002.

[13] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264. doi: 10.1109/CVPR.2018.00240.

[14] 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," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1073–1087, 2017. doi: 10.1177/0278364917722396.

[15] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Advances Neural Information Processing Systems*, 2016, pp. 4565–4573.

[16] Y. Li, J. Song, and S. Ermon, "InfoGAIL: Interpretable imitation learning from visual demonstrations," in *Proc. Advances Neural Information Processing Systems*, 2017, pp. 3812–3822.

[17] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *Proc. 2017 IEEE Intelligent Vehicles Symp. (IV)*, pp. 204–211. doi: 10.1109/IVS.2017.7995721.

[18] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Learning temporal strategic relationships using generative adversarial imitation learning," in *Proc. 17th Int. Conf. Autonomous Agents and MultiAgent Systems*. 2018, pp. 113–121. doi: 10.5555/3237383.3237407.

[19] R. Bellman, "A Markovian decision process," *J. Math. Mech.*, vol. 6, no. 4, pp. 679–684, 1957. doi: 10.1512/iumj.1957.6.56038.

[20] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Pedestrian trajectory prediction with structured memory hierarchies," in *Proc. Joint European Conf. Machine Learning and Knowledge Discovery in Databases*. New York: Springer-Verlag, 2018, pp. 241–256. doi: 10.1007/978-3-030-10925-7_15.

[21] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," in *Proc. Int. Conf. Learning Representation, (ICLR)*, 2018, pp. 1–15.

[22] K. Saleh, M. Hossny, and S. Nahavandi, "Long-term recurrent predictive model for intent prediction of pedestrians via inverse reinforcement learning," in *Proc. 2018 Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–8. doi: 10.1109/DICTA.2018.8615854.

[23] Y. Zhang, W. Wang, R. Bonatti, D. Maturana, and S. Scherer, "Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories," in *Proc. Conf. Robot Learning (CoRL)*, 2018, pp. 1–12.

[24] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. Conf. AAAI*, Chicago, IL, 2008, pp. 1433–1438.

[25] T. V. Le, S. Liu, and H. C. Lau, "A reinforcement learning framework for trajectory prediction under uncertainty and budget constraint," in *Proc. 22nd European Conf. Artificial Intelligence*. Amsterdam, The Netherlands: IOS Press, 2016, pp. 347–354.

[26] M. Herman, V. Fischer, T. Gindele, and W. Burgard, "Inverse reinforcement learning of behavioral models for online-adapting navigation strategies," in *Proc. 2015 IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 3215–3222. doi: 10.1109/ICRA.2015.7139642.

[27] S. Sharifzadeh, I. Chiotellis, R. Triebel, and D. Cremers, "Learning to drive using inverse reinforcement learning and deep Q-networks," in *Proc. NIPS workshop Deep Learning for Action and Interaction*, 2016, pp. 1–7.

[28] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, 1967. doi: 10.1109/TIT.1967.1054010.

[29] C. You, J. Lu, D. Filev, and P. Tsiotras, "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robot. Auton. Syst.*, vol. 114, pp. 1–18, Apr. 2019. doi: 10.1016/j.robot.2019.01.003.

[30] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Neighbourhood context embeddings in deep inverse reinforcement learning for predicting pedestrian motion over long time horizons," in *Proc. IEEE Int. Conf. Computer Vision Workshops*, 2019, vol. 108, pp. 466–478. doi: 10.1109/ICCVW.2019.00149.

[31] J. Colyar and J. Halkias, "US highway 101 dataset," Federal Highway Administration (FHWA), Washington, D.C., Tech. Rep. FHWA-HRT-07-030, 2007.

[32] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan et al., nuScenes: A multimodal dataset for autonomous driving. 2019. [Online]. Available: arXiv:1903.11027

[33] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Going deeper: Autonomous steering with neural memory networks," in *Proc. IEEE Int. Conf. Computer Vision Workshops*, 2017, pp. 214–221. doi: 10.1109/ICCVW.2017.34.

[34] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proc. Int. Conf. Machine Learning*, 2016, pp. 49–58. doi: 10.5555/3045390.3045397.

[35] D. Wingate, "Solving large MDPS quickly with partitioned value iteration," Ph.D. dissertation, Brigham Young Univ., Provo, UT, 2004.

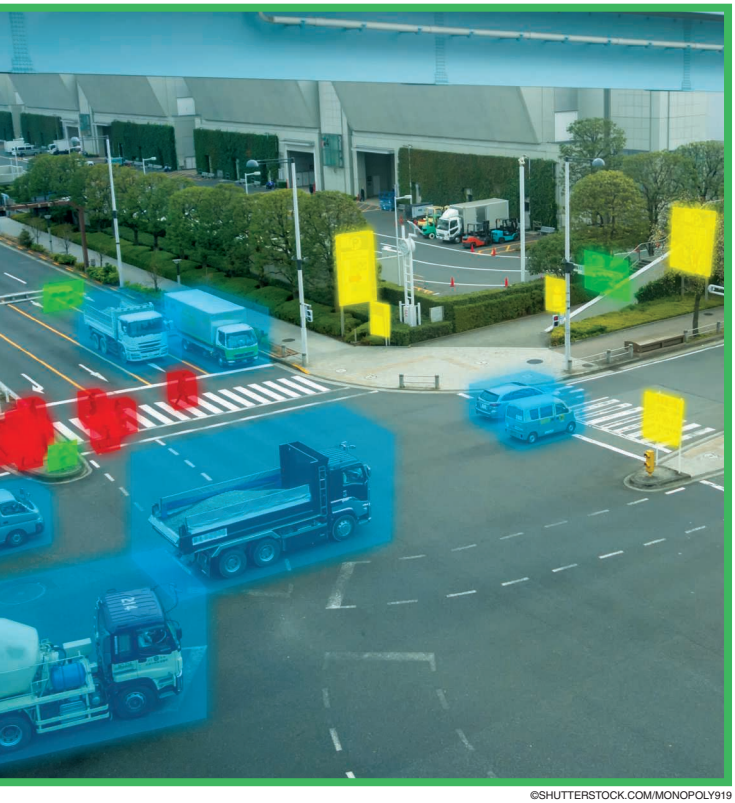
[36] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. 2012 IEEE Conf. Computer Vision and Pattern Recognition*, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

SP

Marco Cococcioni, Federico Rossi, Emanuele Ruffaldi,
Sergio Saponara, and Benoît Dupont de Dinechin

Novel Arithmetic in Deep Neural Network Signal Processing for Autonomous Driving

Challenges and opportunities



©SHUTTERSTOCK.COM/MONOPOLY919

Digital Object Identifier 10.1109/MSP.2020.2988436
Date of current version: 24 December 2020

This article focuses on the trends, opportunities, and challenges of novel arithmetic for deep neural network (DNN) signal processing, with particular reference to assisted- and autonomous driving applications. Due to strict constraints in terms of the latency, dependability, and security of autonomous driving, machine perception (i.e., detection and decision tasks) based on DNNs cannot be implemented by relying on remote cloud access. These tasks must be performed in real time in embedded systems on board the vehicle, particularly for the inference phase (considering the use of DNNs pretrained during an offline step). When developing a DNN computing platform, the choice of the computing arithmetic matters. Moreover, functional safe applications, such as autonomous driving, impose severe constraints on the effect that signal processing accuracy has on the final rate of wrong detection/decisions. Hence, after reviewing the different choices and tradeoffs concerning arithmetic, both in academia and industry, we highlight the issues in implementing DNN accelerators to achieve accurate and low-complexity processing of automotive sensor signals (the latter coming from diverse sources, such as cameras, radar, lidar, and ultrasonics). The focus is on both general-purpose operations massively used in DNNs, such as multiplying, accumulating, and comparing, and on specific functions, including, for example, sigmoid or hyperbolic tangents used for neuron activation.

Introduction

The use of DNNs as a general tool for signal and data processing is increasing in both the automotive industry and academia, proposing a set of algorithms for most of the autonomous driving tasks. The effort in computing these artificial intelligence (AI) algorithms is an open challenge in the field of computing platforms. In particular, when considering strict requirements, such as lowering the power consumption, maximizing the throughput, and minimizing the latency, the computational complexity becomes more critical. Moreover, with modern achievements in sensor components, the complexity and requirements scale further, with data coming in higher volumes and dimensions and at higher speeds [1].

This survey work is focused on the trends, opportunities, and challenges of the adoption of DNN signal processing techniques for autonomous driving and the needs of signal processing acceleration as well as the relevant computing arithmetic. Indeed, autonomous driving is a safety-critical application, as also specified in functional safety standards, such as International Organization for Standardization 26262, with strict requirements in terms of real time (both throughput and latency) [1], [2]. In levels $\mathcal{L}1$ and $\mathcal{L}2$ of the Society of Automotive Engineers autonomous driving scale [3], only assistance to the human driver is needed. Therefore, signal processing based on deterministic algorithms is still enough; e.g., fast Fourier transform-based processing of frequency-modulated continuous-wave radar was done in [1]. Instead, for high autonomous driving levels, from $\mathcal{L}3$ to $\mathcal{L}5$, the complexity of the scenario and need for signal processing, not only for sensing but for localization, navigation, decisions, and actuation, is so high that in recent state-of-the-art DNNs, signal processing is proposed to be used on board [1], [2], [4], [5]. This trend is confirmed by the rise of the Autonomous Systems Initiative within the IEEE Signal Processing Society [6]. DNNs have reached state-of-the-art status in several signal processing domains, such as image processing, segmentation, classification, tracking [7]–[10], computer vision [11], and related areas [12]–[14].

In the automotive field, while sensor raw data processing (from cameras, lidar, radar, and ultrasonics) can be still performed using classical signal processing techniques, DNNs are emerging as more appropriate solutions to solve complex and high-level tasks, such as data fusion, classification, and planning in harsh, unstructured, and continuously changing environments. Tasks such as scene understanding (e.g., image segmentation, region-of-interest extraction, subscene classification, and so on) must be done on board vehicles since cloud-based computing scenarios (where signal processing is done on remote cloud servers and on board, there is only a client unit generating requests to the server) suffer from several issues: privacy, authentication, integrity, connection latency and contention, and even communication unavailability in uncovered areas (highway tunnels, etc.). Onboard DNN signal processing can be done only if a low-computational complex algorithm is used and performing hardware (HW) is adopted. Hence, onboard computing units for DNNs should be optimized in terms of the ratio between the signal processing throughput performance and resources (memory, bandwidth, power consumption, and so on) [15]–[17]. This is the trend that big players, including Google, NVIDIA, and Intel, are following as they try to enter the autonomous driving market. Tesla recently announced its full self-driving (FSD) chip. This concept is also the core of the automotive stream in the Horizon 2020 European Processor Initiative (EPI) (embedded high-performance computing for autonomous driving, with BMW Group as the technology's main end user [17]), where this article's authors are involved.

To address the preceding issues, new computing arithmetic styles are appearing in the state of the art [18]–[26] to overcome the classic fixed-point (INT) versus IEEE Standard 754 floating-point duality in the case of embedded DNN signal processing. Just as an example, Google is proposing Brain Floating-Point Format (BFLOAT) 16, which is equivalent to a standard single-precision floating-point value with a truncated mantissa field. BFLOAT16 is supported in the Google Cloud tensor-processing unit (TPU) and TensorFlow and Intel AI processors. Intel is also proposing Flexpoint [18], [19], a 16-bit-block floating-point format aiming to replace Float32. NVIDIA's Turing architecture supports, in its tensor cores, Float16 to Float16 or Float32 matrix multiply–add operations as well as integer 4 (INT4) or INT8 to INT32 matrix multiply–add operations, the latter for inferencing workloads that tolerate quantization [24]. The Tesla FSD chip exploits a neural processing unit using eight-by-eight-bit integer multiplication and 32-bit integer addition. Transprecision computing for DNNs is also proposed in the state of the art by academia [20] and industry, e.g., IBM and Greenwaves in [21]. Recently, a novel way to represent real numbers, called *Posit*, has been proposed [25], [26]. Basically, the Posit format can be thought of as a compressed floating-point representation, where more mantissa bits are used for numbers around one, with fewer stepping away from one, within a fixed-length format with variable-size fields (the exponent bits adapt accordingly to maintain the format fixed in length).

Review of state-of-the-art DNN signal processing in autonomous driving

Autonomous driving is deeply bounded to vehicle navigation, including vehicle self-localization, motion, mapping, and interaction. A relevant survey on trends and technologies for autonomous driving is presented in [27]. The localization task is aimed at knowing the vehicle's pose (position and orientation) as it is referred to a relative or absolute coordinate system. Traditional approaches to localization include satellite communication, such as GPS. However, these are typically weak radio signals that can be easily occluded by trees and buildings in a metropolitan scenario. There exist other types of equipment, such as inertial measurement units, that, combined with GPS, real-time kinematics, and Kalman-based predictors, can solve this problem, but they increase the implementation cost. Since the task of constantly knowing the vehicle's position is critical, one cannot rely only on these signals.

The mapping task introduces a further level of context awareness. With a map-matching approach, a vehicle is able to know not only its position but its surroundings. An important mapping technique is simultaneous localization and mapping (SLAM) [28], which enables a vehicle to bypass or minimize the need for satellite navigation. SLAM considers the surroundings as a probability distribution of points rather than a snapshot of the context in time, building a world model by making use of lidar sensors or similar devices. The typical output of these sensors is point clouds that represent the surrounding environment and must be processed to give more information about the area. In [29], a way to classify lidar images using DNNs is presented. In [30], a benchmark challenge for DNNs for the German Traffic Road Sign Recognition

Benchmark (GTRSB) is proposed, and in [31], there are some advanced DNN techniques, such as data augmentation and region-of-interest extraction, to maximize DNN recognition and detection accuracy, reaching top-level accuracy on road sign recognition and detection benchmarks. Moreover, with advanced developments in computer vision, vehicles can be equipped with cameras whose signals can be processed by DNNs as well. For example, in [32]–[34], a semantic segmentation of city landscapes challenge is presented, providing benchmarks for DNNs to prove their ability to identify the main components of a road (such as lanes, other vehicles, and pedestrians) from image or video signals. On the industry side, with the advent of companies including Tesla and Google's Waymo, the use of DNNs in processing lidar and camera signals has become more central.

Low-precision DNNs

Academia and industry have proposed multiple solutions to the problem of reducing the number of bits used to represents DNNs' weights, compressing the size from 32 to 16, eight, four, and even one bit, resulting in little to no degradation in performance when tested with common DNN tasks and benchmarks. As an emerging trend in the state of the art, the literature is starting to explore the possibility of using the newly introduced Posit representation to halve the weights' size while maintaining the same accuracy and to further reduce the weights' size while sacrificing little to no DNN precision. A very interesting work has been presented in [35], where network weights have been binarized, dramatically reducing the network footprint and increasing the training and inference speed. On the industry side, NVIDIA has led the reduction of weight bits with its TPU, introducing integer weight types, such as eight- and four-bit integers. In [36], a novel method is introduced to train neural networks (NNs) with extremely low precision (e.g., one bit), weights, and activations at runtime. In [37], the authors studied the training of NNs using low-precision fixed-point computations and evaluated the impact of different rounding techniques.

The aim of this article is to develop an NN accelerator based entirely on Posits, while also embedding look-up tables (LUTs) for low-bit Posits, such as four–12-bit Posits. In this way, we ensure a homogeneity of representations that is lost in the NVIDIA approach due to the discontinuity introduced when switching from floating-point half precision to eight- or four-bit integers.

Alternative representations for real numbers

In this section, we review the most interesting representation for real numbers, which could be used as an alternative to the floating-point representation (the IEEE 754 standard, 2008, which will be referred to simply as *Float* from now on). In the following, we will use a homogeneous representation for the different number representation "Type Bits[,Exp]," where Type is the name of the representation (Float, Posit, and Fixed), Bits is the number of bits, and Exp is the number of bits used for the exponent. For fixed-point Exp representations, the scaling factor to be applied to the number is considered to be a signed integer (e.g., Fixed16,8 represents a value with eight bits in the integer part and eight bits in the fractional part). For Float when Exp is missing, the standard value is assumed: 11, eight, and five for Float64, Float32, and Float16, respectively, corresponding to binary64, binary32, and binary16 of the IEEE standard.

BFLOAT16

The research on DNNs has demonstrated that 16-bit Floats could be enough for many classification problems. From this research came the idea to give HW support to the standard half precision (Float16,5) too, in addition (or as an alternative) to Float32. The problem is that pretrained DNN models are usually available with Float32, and thus, lowering them to five bits of exponent could introduce alterations to the classification and affect the overall classification performance. For this reason, the BFLOAT16 format (Brain Float 16-bits, namely, Float16,8 in the present notation) has been recently introduced with eight bits of exponent instead of five. Having the same size of the exponent of Float32, the use of BFLOAT16 introduces a loss of numerical precision but no loss of dynamic range. Also, the conversion with Float32 is bitwise.

Flexpoint

Flexpoint numbers [18] are characterized by a shared tensor exponent used for all number representations in a given NN layer (e.g., a 16-bit Flexpoint plus a five-bit shared exponent). Moreover, the magnitude of the common exponent is dynamically adjusted according to the required numerical range during training. The Flexpoint approach, although interesting and powerful, cannot be used as a drop-in replacement for Floats: changes are required to the DNN software (SW) libraries. This also makes the reuse of pretrained DNNs cumbersome.

Type-3 uniform numbers: Posits

Type-3 uniform numbers are the third proposal of universal numbers offered, again, by Gustafson. They can be exact (Valids) or inexact (Posits). Posits are particularly interesting because they are a drop-in replacement for Floats, while Valids are not. Posits will be presented and deeply investigated in the next section. Before that, we discuss, in the next two subsections, two further representations that are somehow related to Posits.

Universal coding of real numbers using bisection

The bisection method proposed by Lindstrom in [38] is based on Elias codes. It encodes each real number in a binary string based on bisecting intervals, starting from the base interval $(-\text{inf}, +\text{inf})$. Each bit of the string is the result of a comparison with a value contained in a given interval. The framework proposed as universal coding enables building new number systems by defining a generator function to produce the various intervals and a so-called refinement operator to compute the average value between two numbers. Theoretically speaking, this encoding is very interesting due to the possibility to rapidly prototype and verify the representation. However, the encoding is quite

inefficient, involving elaborate expressions in its computations, thus becoming HW-unfriendly. This suggests that this particular encoding is not so interesting in the high-performance HW accelerator topic discussed here, although Posit numbers can also be generated using this powerful encoding technique.

Logarithmic numbers and the Kulisch accumulator

As pointed out by Johnson, a researcher at Facebook AI Research, the problem in [39] with floating-point operations in HW is that the transistors needed to perform multiplication and division occupy the main part of the floating-point unit (FPU), which is significantly more complex than for addition/subtraction. To overcome this problem, the logarithmic number system (LNS) was proposed decades ago in [40]. The LNS consists of representing a number as $y = 2^x$, i.e., in a pure logarithmic way. This makes multiplication and division a matter of just adding and subtracting logarithmic numbers.

However, this requires huge HW LUTs to compute the sum or difference of two logarithmic numbers [39]. This has been one of the main bottlenecks for the format since handling these tables can be more expensive than basic HW multipliers. To avoid common fused multiplication and adding complexity, the Kulisch accumulation can be used. The idea is to not accumulate with a floating-point-type but instead, maintain an accumulator in a fixed-point type. As a drawback, this approach leads to a significant increase in logic circuitry and power consumption, due to the bit count requirements of the Kulisch accumulator. Although this approach is really promising and can be combined with the Posit philosophy, it has not yet been demonstrated that logarithmic numbers are more effective than Floats for DNNs. Thus, more research is clearly needed before resorting to this solution.

A deeper investigation of Posits

Posit numbers have been proposed by Gustafson in [26]. The format is a fixed-length representation for real numbers, and it has two parameters: the total number of bits (totbits) and the number of exponent bits (esbits). It is composed of a maximum of four fields (see Figure 1):

- one-bit sign field S
- variable-length regime field R (1.rebits)
- exponent field E, which has a predetermined maximum length of esbits (field E can even be absent)
- variable-length fraction field F (it can be absent, too).

With the adopted notations, PositN,E refers to a Posit with N total bits and E esbits.

Both the total number of bits and the maximum size of the exponent field (esbits) are decided empirically a priori, depending on the application. These two lengths are those that fully characterize the Posit representation. The regime field length is determined by the number of consecutive zeros after the sign bit ended by one or, vice versa, by the number of consecutive ones ended by one zero. In the former case, the regime value is negative. After having determined the regime length, the associated value k can be retrieved according to the procedure illustrated in Figure 2. The bits that follow the regime field are, if present, the ones associated to the exponent. Their number can be, at maximum, equal to the esbits (the a priori predetermined maximum number of exponent bits). When the field is missing, the exponent e is assumed to be zero. When fewer bits than esbits are present, the value of e can be obtained by filling the missing bits with zeros before decoding it (see Figure 3).

If there are additional bits after the exponent field, they are the ones associated to the fractional part of the mantissa. If the Posit is negative (the first bit is equal to one) before decoding it to retrieve k, e, and f, the two's complement of its remaining bits must be computed. Therefore, let p be the integer represented by the Posit bit string, k the correspondent integer indexed by the regime bits into a run-length table (see Figure 2), e the unsigned integer associated to the exponent field E, and $f = 0.f_1 f_2 \ldots f_n$ [the fractional part of the mantissa m $(m = 1 + f)$, associated to the F field]; the expression that maps the bit set to the real value is

$$x = \begin{cases} 0, & \text{if } p = 0 \\ \text{NaR}, & \text{if p} = -2^{(\text{totbits}-1)} \\ \text{sign}(p) \times u^k \cdot 2^e \cdot (1+f), & \text{otherwise} \end{cases},$$

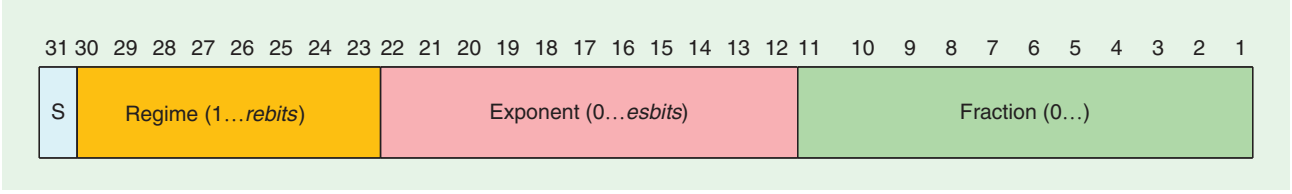


FIGURE 1. The 32-bit Posit data type.

Binary	0000	0001	001x	01xx	10xx	110x	1110	1111
Numerical Meaning, k	–4	–3	–2	–1	0	1	2	3

FIGURE 2. The mapping table between the regime bits and k value for a five-bit string. Amber bits represent the regime bits, and brown ones terminate the regime run.

where

$$u = 2^{2^{\text{esbits}}}.$$

Notably, it is possible to prove that for PositN,0, the numbers in the range $[-1, 1]$ are encoded as signed fixed points across $N-1$ bits. This property is important for the level-1 (L1) operations discussed later. Figure 3 provides an example of Posit16,3 (16 b, with a maximum of three exponent bits) and its decoding procedure.

Posit advantages over Floats and industrial adoption

As shown in [26], the main advantages of Posits over IEEE floating points are represented by less waste of representations (such as unique-zero and not-a-number bit configurations) and higher decimal accuracy when compared to the same bit length floating point. Moreover, the simplicity of the Posit number systems theoretically facilitates a more HW-friendly implementation, simplifying circuitry and thus reducing area occupation and power consumption. Even if the Posit format is relatively new, it has already attracted the attention of researchers from Facebook, IBM, Google, Microsoft, Intel, Bosch, Huawei, Fujitsu, Qualcomm, Kalray, Micron, Altair, Etaphase, Posit Research, Rex Computing, Stillwater Supercomputing, and Comma, as reported by Gustafson during a recent talk [41].

SW and HW implementations of Posits

SW implementations

Having SW implementation of Posit arithmetic is useful to test the applicability of the type to existing libraries and algorithms to compare performance against traditional floats and in the absence of proper HW support for Posit operations.

SoftPosit

This is a library endorsed by the Next Generation Arithmetic committee. Among its positive factors, it is multiplatform, supporting C, C++, Julia, and Python. However, it presents hard-coded Posit configurations and nonmodern implementation without templatized classes for the various configurations. It also lacks support for tabulated Posits.

Beyond Floating Point

Beyond Floating Point is one of the first C++ Posit arithmetic libraries developed. However, it is still incomplete and does not support Posit tabulation.

StillWater

StillWater is a complete library with modern C++ features and class templatization, although it is computationally heavy and missing Posit tabulation.

cppPosit

This library (available in [42]), developed by the authors of the present work, exploits some of the modern C++ features, such as templates (i.e., generic programming), and traits. It supports Posit tabulation and logic separation between the front-end interface and back-end underlying type used for computation: the front end is the Posit number expressed in its packed form, while the back end enables choosing different approaches for performing mathematical operations.

The library identifies four operational levels, with increasing computational cost. At level 1 (called L1), operations are just bit manipulation of the bits of the encoding. The cost is the same as integer operations performed in the arithmetic logic unit (ALU). At level 2 (L2), Posit data are extracted to fields (sign, regime, exponent, and fraction), with no need to compute the exponent completely. Computations are performed in these fields, and the cost includes encoding and decoding of the format. At level 3 (L3), we have the unpacked version that is completely built (including the sign, exponent, and fraction). In addition to L2 operations, here, there is the need to build the full exponent. At level 4 (L4), the unpacked version is used to perform the operations in either SW or HW floating point or using fixed-point representations. The most efficient level is, of course, L1 since it comprehends operations that only require bit manipulation of the Posit representation, which can be computed on existing ALUs without having to wait for Posit-processing units. Table 1 reports the most important L1 operations provided by the library. The library offers the possibility to use different back ends for Posit operations:

- a fixed-number back end (using a quire-like approach)
- a tabulated back end (see the section "The LUT Approach")

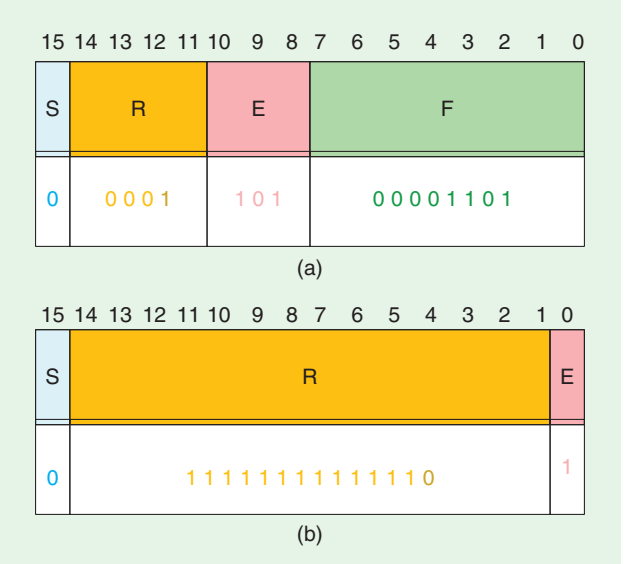


FIGURE 3. Two examples of Posit16,3, that is a 16-bit Posit with esbits = 3. (a) The associated real value is $+256^{-3} \cdot 2^{5} \cdot (1 + 13/256)$ (13/256 is the value of the fraction, and 1 + 13/256 is the value of the mantissa). The final value is therefore $+1.907348 \times 10^{-6} \cdot (1 + 13/256) \cong +2.0042 \times 10^{-6}$. (b) The associated real value is: $+256^{+12} \cdot 2^{4} \cdot (1 + 0)$ (since the fractional part of the mantissa is missing, we set it to zero). The final value is therefore $2^{96} \cdot 2^{4} \cong +1.2676506 \times 10^{30}$. The second example enables us to clarify that 1) the fractional part can be missing and 2) the exponent field can be shorter than its maximum size (in that case, the missing bits are assumed to be zero: the exponent four comes from reconstructed exponent field 100).

- a floating-point back end: either SW (SoftFloat) or HW (FPU).

Each L3 operation in the cppPosit library undergoes three different phases: 1) decode, 2) operation back end, and 3) encode. Each of these phases requires different functionalities in the processor architecture:

- *Decode*: mostly bit manipulation; the core function that is used here is the count-leading zeros (CLZ) built-in function.
- *Back end*:
  - *Fixed*: requires big-integer (64–128 bit) support
  - *Float*: requires an FPU
  - *SoftFloat*: requires 32/64-bit integer manipulations
- *Encode*: bitwise operations.

Table 2 shows a summary of the requirements support for two common architectures (both have been used for the benchmarks executed in the next sections; respectively, they are Intel i7560u and ARM Cortex A72). The two architectures do not differ in terms of HW requirements for the aforementioned phases. However, speaking about big-integer support, the Intel instruction set architecture (ISA) offers a single instruction (mulq) to perform a 64–128-bit integer multiplication; on the other hand, the ARM ISA requires the execution of two instructions.

HW implementations of the Posit processing unit

Some work has already been done to implement Posit units on field-programmable gate arrays (FPGAs) to provide efficient and optimized HW implementation of Posit arithmetic. In [44], an algorithmic flow and architecture generator for Posit numbers is proposed, including a Float-to-Posit converter unit and base arithmetic units. For the converter, the flow follows two major parts: floating-point unpacking and Posit construction. The first part works as any FPU, while the other determines the impact of the design on the HW. This has been implemented on a Xilinx Virtex-6 device, resulting in roughly 600 FPGA slices for a 32-bit Posit adder and 300 for a 16-bit Posit adder.

In [45], a Posit core generator, called *Poisson Generator* (*POSGEN*), is proposed. In addition, the FPGA design has been enriched with an extension of the Basic Linear Algebra Subprograms (BLAS) library for the Posit numbers, called *Posit BLAS*, to connect the FPGA through the Intel Open Computing Language libraries. The results show that the maximum frequency reached by the proposed implementation matches the state-of-the-art floating-point cores (FloPoCo) floating-point implementation. However, the area consumed by the POSGEN implementation is much higher than the FloPoCo one.

Another Posit arithmetic core, called the *Posit arithmetic unit*, generator is presented in [46], where generators for the Posit adder and multiplier are proposed. The design results show a reduction in the area occupation, referring to [44] for both the adder and multiplier as well as a reduction in power consumption for eight-bit Posits. For 16-bit Posits, the results are overturned in favor of the other implementation as well as for 32-bit Posits. Moreover, from the comparison between the Posit realization and the standard IEEE floating point, it is evident that a 32-bit Posit adder occupies less area and has a lower delay than a 32-bit Float adder. The 32-bit multiplier, instead, occupies the same area but with a higher delay. Finally, a 16-bit adder occupies a higher area with a higher delay.

In [47], another Posit arithmetic core generator has been introduced, called *PACoGEN*. The work presents different generators for HW description language adder/subtractor and multiplier/division cores. An interesting aspect of this implementation is the pipelined Posit arithmetic architecture, aimed at increasing the throughput of the unit and trying to produce a new result at each clock cycle (when at regime), making the three phases of an operation independent (Posit data extraction, core arithmetic process, and Posit construction). Design results show that the proposed implementation has a lower area (LUT) · period (ns) when compared to proposals in the literature, such as [46]. However, when the design is compared to standard floating-point ones, the results show that 32-bit Posit adder/multiplier units occupy more area than some 32-bit floating-point ones.

An accelerator for Posit-based BLAS operations is proposed in [48]. The work presents a modular framework for Posit arithmetic with the common three-step dataflow: Posit data extraction, operation, and construction. The implementation consists in a Posit adder, multiplier, and Posit accumulator. The proposed BLAS library enables vectorized operations, such as element-wise addition, subtraction, and multiplication, as well as the dot product and vector sum. Experimental results show a consistent speedup obtained when using the vectorized approach compared to an SW implementation.

When considering FPGA implementation of Posit arithmetic units, we need to consider the area occupation (thus, the power consumption) of the realized design and compare it to an FPU realization. Having a 32-bit HW Posit unit makes sense if the area of the realized Posit unit is less than the FPU one. If this does not hold, it still makes sense to have a 16-bit

Table 1. The cppPosit's most important L1 operations.

Operation	Approximated	Requirements
$2x$	No	esbits = 0
$x/2$	No	esbits = 0
$1-x$	No	esbits = 0, $x \in [-1, 1]$
$1/x$	Yes	esbits = 0
FastSigmoid	Yes	esbits = 0
FastTanh [43]	Yes	esbits = 0
FastELU	Yes	esbits = 0

The table shows whether the operation produces an exact or approximated result and reports the requirements to be fulfilled. For instance, notice how $1-x$ can be computed using fast bit manipulations only when $x \in [-1, 1]$.

Table 2. The requirements support of Intel and ARM for the cppPosit library.

Requirements	Intel Seventh Generation (Kaby Lake)	ARM v8
CLZ built in	✓	✓
Big integer	✓ (one single instruction)	✓ (two instructions needed)
FPU	✓	✓
Integer manipulation	✓	✓
Bitwise operations	✓	✓

HW Posit unit if its area is less than a 32-bit FPU one since 16-bit Posits achieve similar performance to 32-bit Floats in different application fields (in the "Benchmark Data Sets and Examples of Achievable Results" section, we show that in DNNs, even a Posit8 can match the performance of a Float32).

Posit-based DNNs for signal processing

Nonlinear activation functions are a very important part of DNNs. Their efficient implementation is therefore crucial. In the next sections, we will see how some widely used activation functions can be efficiently computed when using Posits.

DNN activation functions

In this section, we present special implementations of well-known mathematical functions and algorithms adapted to the Posit format. When considering these implementations, it is crucial to build them mostly with L1 operations (see the "cppPosit" section).

Sigmoid

The sigmoid function

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

has a very efficient approximation when using a Posit format with zero exponent bits, consisting only in a manipulation of the representation's bits. This discovery is due to Yonemoto and Gustafson [26]. Although this formula is appealing in NNs since it leads to faster training, there are intrinsic limitations when reducing the total number of bits (precision). Indeed, the sigmoid function does not exploit the dynamic range of the Float or Posit format enough since its codomain varies in [0, 1]. For this reason, we have developed a fast approximation of the hyperbolic tangent (see the next section).

Hyperbolic tangent

To solve said problem, an expression for the hyperbolic tangent has been derived using a linear combination of the sigmoid function:

$$\tanh(x) = 2 \cdot \text{sigmoid}(2 \cdot x) - 1.$$

This leads to a fast and approximated version of the hyperbolic tangent (FastTanh, from now on) when using the aforementioned fast sigmoid approximation:

$$\text{FastTanh}(x) = 2 \cdot \text{FastSigmoid}(2 \cdot x) - 1.$$

To have an L1 expression, we initially restrict the domain to negative numbers only. The doubling operation and sigmoid function are L1 when using zero exponent bits, and the result of the first term of the expression is contained in the unitary range. This means that computing $-(1-y)$ is also an L1 operation, according to Table 1. Finally, thanks to Tanh symmetry, we can also extend back the domain to positive numbers. Figure 4 shows the time comparison between the fast approximated version and the exact version of the hyperbolic tangent. As we can see, the FastTanh approximated version is six times faster than the exact Tanh version. Moreover, we computed the mean square error (MSE) between the two, resulting in $\text{MSE} = 2.947 \cdot 10^{-3}$ in the entire Posit interval.

A similar approach can be applied to the extended linear unit (ELU) activation function. This function solves the common problem of vanishing gradients of sigmoid-like functions, such as the hyperbolic tangent, and the effects of the flattening of the rectified linear unit (ReLu) for negative numbers:

$$\text{ELU}(x) = \begin{cases} e^x - 1, & \text{if } x \le 0 \\ x & \text{otherwise} \end{cases}.$$

Starting from the Sigmoid function, we can obtain the negative argument case as follows, where each step of the ensuing equation can be executed as an L1 operation with contained approximation:

$$\text{ELU}(x) = 2 \cdot \left(\frac{1}{2 \cdot \text{Sigmoid}(-x)} - 1 \right).$$

If we switch from the Sigmoid to the fast-approximated version already exploited with the hyperbolic tangent, we can get a fast approximation of the ELU (called *FastELU*). Table 3 shows an example of accuracy and timing improvements when using the approximated ELU function in place of the exact one. We trained a LeNet-like [49] model with the different activation functions until negligible improvements in the validation accuracy were obtainable. Then, we tested the three previously mentioned trained models with the different Posit types, reporting the accuracy and processing time. As we can see, the approximated FastELU model outperforms the ReLu model in terms of accuracy and, in particular, the type Posit8,0 shows lower degradation in terms of accuracy

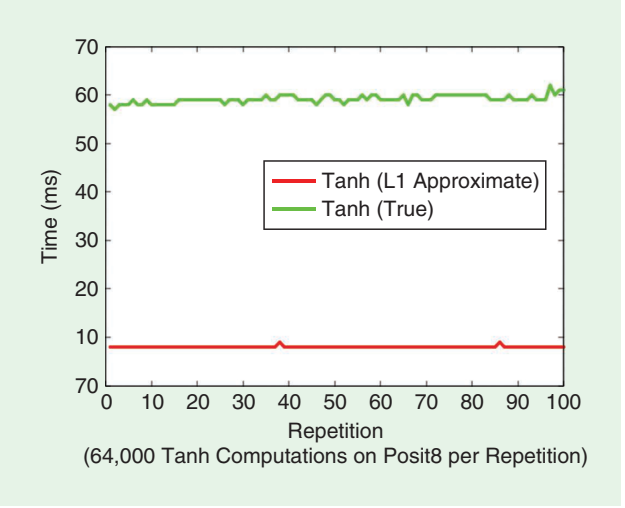


FIGURE 4. The time comparison in various repetitions of ~60,000 executions of Tanh and FastTanh for Posit16,0 [benchmarks were executed on an Intel seventh-generation (Kaby Lake) i7-7560U processor with two cores at 2.4 GHz]. The latter appears to be roughly six times faster, with a computed mean square error of $2.947 \cdot 10^{-3}$.

with FastELU/ELU rather than ReLu. In terms of timing, the FastELU and ReLu are comparable with PositN,0, both being L1 operations, while ELU is costlier. More mathematical details about FastTanh and FastELU can be found in [50].

The LUT approach

When using a low number of bits, the application of LUTs quickly becomes appealing. In theory, one could profile a specific application (i.e., computing the histogram of the most-used values and the most significant range) and then create an ad hoc series of values. For this set of values, one has only to compute the four LUTs for the four elementary operations plus the tabulation of significant unary functions (exponential, log, trigonometric functions, square root, square, and so on). There also exist some optimized soft mathematical libraries in the Sun Cephes collection ([51]). The collection consists of more than 400 mathematical functions entirely implemented in *C* and mostly delivered in different arithmetic precisions (32-, 64-, 80-, 96-, 144-, and 336-bit operands).

LUTs for Posits

The Posit LUT size depends on the overall number of Posit bits. Without any optimization, a table for a binary operation for x-bit Posits is a square one, with the number of rows and columns equal to $R = C = 2^x - 1$. Each table entry occupies b bits, depending on the underlying type used to hold the Posit number. The overall occupation for a naive table is thus $S = R \cdot C \cdot b$. For an eight-bit Posit represented across an eight-bit unsigned integer type, a single table occupies 64 kilobits. To reduce the table size, the symmetry of addition/subtraction operations can be exploited to halve the table size and number. Moreover, multiplication and division tables can be discarded by exploiting logarithm properties, thus using just the addition/subtraction tables.

Multiply and accumulate

The task of multiplying two numbers and summing the result into an accumulator is very common during DNN operations (such as convolution or matrix multiplication). The presence of an HW multiplier–accumulator is crucial since it helps in reducing by one the number of roundings involved in the computation at each step. The authors of [52] present the implementation of an exact multiply-and-accumulate (MAC) function for low-precision Posits and other floating/fixed-point types, resulting in eight-bit Posit matching and even overcoming 32-bit Floats.

Fused/exact dot product

When dealing with low-bit number representations, the dot product is a critical operation. The dot product is intensively used in DNNs during convolution operations, and overflows can occur with high probability during the accumulation of term products. To avoid most of these overflows, two solutions can be adopted.

Fused dot product

While a MAC technique computes the product result, rounds it, adds it to the accumulator, and then rounds it again, a fused dot product (FDP) (also known as *fused multiply–add*) computes the entire expression at the maximum available precision, typically using an accumulator that has twice the bits of the single operands. In [26], the potentiality of Posits for overcoming rounding issues when using fused operations is shown, such as the possibility to use 32-bit Posits for high-performance computing instead of 64-bit Posits, thus increasing the computation speed and reducing the power consumption and storage requirements.

Exact dot product

The exact dot product technique makes use of the concept of quires (a very-high-bit-count scratch area) as the accumulator, deferring rounding only at the very last operation, thus minimizing rounding errors. The concept of quires was introduced by Kulisch in [53] to minimize the number of transistors used to build a fixed-size register inside a processor. A quire is a very-high-bit-count fixed-size scratch area used to perform arithmetic operations at the maximum possible precision given by that fixed-size type. If the quire is properly dimensioned, the rounding error will affect only the very last operation when converting the result back to the original low-precision type. To prevent the quire from underflow or overflow during these operations, we need to dimension it depending on the Posit configuration (https://posithub.org/docs/Posits4.pdf). Suppose that to have a totbits–bit Posit, the maximum possible value for the Posit will be $maxpos = u^{\text{totbits}-2}$, while the minimum possible value will be $minpos = 1/maxpos$, where $u = 2^{2^{\text{esbits}}}$; each number is,

Table 3. The comparison of different activation functions when applied to an NN for traffic sign classification.

Activation	GTRSB								
	FastELU			ReLu			ELU		
	Accuracy (%)	Time (ms)	NCT[1]	Accuracy (%)	Time (ms)	NCT[1]	Accuracy (%)	Time (s)	NCT[1]
Posit16,0	94	5.8	—	92	5	—	94.2	6.4	—
Posit14,0	94	4.6	0.79	92	4.3	0.86	94.2	5.2	0.81
Posit12,0	94	4.6	0.79	92	4.3	0.86	94.2	5.1	0.79
Posit10,0	94	4.6	0.79	92	4.2	0.84	94.2	5	0.78
Posit8,0	92	4.6	0.79	86.8	4	0.8	91.8	5	0.78

Benchmarks were executed on an Intel seventh-generation (Kaby Lake) i7-7560U processor with two cores at 2.4 GHz.
[1]NCT: normalized computing time. (Posit computing times are normalized against the Posit16,0 computing times.)

then, an integer multiple of *minpos*. Suppose we need to perform the dot product $\{maxpos, minpos\} \cdot \{maxpos, minpos\}$; we'll need the quire to be able to accommodate the value $maxpos^2/minpos^2$. After some transformation, we can compute the maximum value to hold as

$$2^{(4 \cdot \text{totbits} - 8) \cdot 2^{\text{esbits}}}.$$

Moreover, one bit has to be reserved for the sign, and more bits must be held to handle the sum (e.g., Gustafson chooses 30 more bits to guarantee the absence of overflows). Practically, for example, this means that with an eight-bit Posit (esbits = 0), we will need one 64-bit quire register; for a 16-bit Posit (esbits = 1), we will need a 256-bit quire register (four 64-bit registers); and for a 32-bit (esbits = 2) Posit, we will need a 512-bit quire register (eight 64-bit registers).

Kalray massively parallel processor array approach

To address the challenges of high-performance embedded computing with time predictability, Kalray has been refining a homogeneous manycore architecture, called the *massively parallel processor array* (*MPPA*), based on very-long instruction word (VLIW) cores. On the third-generation MPPA processor [54], each VLIW core is paired with a coprocessor designed for 2D data processing, especially the mixed-precision tensor operations of deep learning inference. In particular, each coprocessor implements matrix multiply–accumulate operations on INT8/32 and Float16/32, where we use the forward slash to describe the two bandwidths of the multiplicand and accumulator. Exploitation of INT8/32 operations relies on TensorFlow Lite quantization support [55], while exploitation of the Float16/32 arithmetic through standard frameworks is the same as for NVIDIA general-purpose GPUs. However, unlike the NVIDIA tensor cores, the Kalray MPPA-3 coprocessors perform exact dot–product processes inside the Float16.32 matrix multiply–accumulate operations by applying Kulisch's principles on an $80 + \epsilon$ accumulator [56].

Following [52], the Posit8 numbers have been identified by Kalray as an effective compressed representation for the Float32 network parameters: instead of rounding the Float32 parameter values to Float16 values, the results of rounding can be restricted to Posit8,0 or Posit8,1 numbers, with the primary benefit of reducing by half the memory capacity and bandwidth required by the network parameters. Kalray focuses on the Posit8,0 and Posit8,1 numbers because they are exactly represented as Float16 numbers and thus can benefit from the exact Float16/32 dot–product operator of the MPPA-3 coprocessors. Conversely, the Posit8,2 numbers include eight values of magnitude 65,536 and larger that are out of range of the Float16 numbers, while the Posit8,3 numbers overflow even the range of the BFLOAT16 numbers. Evaluation of the HW costs and application benefits of using Posit8,0 numbers as a compressed format for Float32 network parameters is ongoing. This evaluation should lead to the inclusion of new arithmetic instructions to expand Posit8,0 to Float16 in the MPPA Internet Protocol delivered to the Horizon 2020 EPI.

Preliminary results obtained by comparing the use of Float32, Float16, and Posit8,E (with an E from zero to three) for data storage (while computation is still done in Float32) during the inference phase using network models for both the classification task [e.g., SqueezeNet, Alexnet, Visual Geometry Group (VGG)-16, VGG-19, GoogleNet and a custom convolutional NN (CNN) on the Modified National Institute of Standards and Technology (MNIST) database, and Canadian Institute for Advanced Research (CIFAR)-100] and detection task [e.g., You Only Look Once (YOLO) v3] show that Posit8,1 or Posit8,2 offers the best performance, with an accuracy loss below 1% versus Float32 but a data compression of factor four. This will lead to reduced complexity for the data transfer and storage that are dominating DNN applications. It should be noted that 1) the networks were pretrained using Float32 and 2) the used data sets in the reported results had thousands of images. Indeed, the ImageNet Large Scale Visual Recognition Challenge 2012 data set has been used for classification and the Visual Object Classes Challenge 2012 data set for detection.

Vectorization of Posit operations (tested on random images)

While in the absence of proper HW support for Posits (i.e., a posit processing unit), we can still accelerate DNN core functions and operators using already-existing HW accelerators. This is the case of the ARM Scalable Vector Extension (SVE) single instruction, multiple data engine. We have also ported our cppPosit library to provide a vectorized version of Posit functions exploiting the ARM SVE library. When talking about vectorized functions, L1 operations are the easiest ones to vectorize. In fact, since they rely only on integer arithmetic and logic, we can effortlessly exploit the native ARM SVE vectorization of integer operations. Benchmarks were executed on a HiSilicon Hi1616 CPU with a 32-bit, 2.4-GHz ARM Cortex-A72 processor, using the ARM SVE Instruction Emulator.

Table 4 shows some timing results between the vectorized and nonvectorized approaches. Furthermore, we have provided an interface between the Posit floating-point back end and ARM SVE types to vectorize L3/4 operations, as well. This enabled implementation of the Posit-accelerated version of convolution and pooling operations. Table 5 provides an example of the timing results with 3×3 convolution and maximum-pooling operations. Finally, Table 6 gives the vectorization performance in terms of processing time on the low-precision inference on Posit8,0. The performance was obtained on the tiny-DNN library on various very-deep NNs. All benchmarks have been executed on the ARM instruction emulator. As reported, the processing time with SVE vectorization enabled dramatic speedups. Note that, in terms of absolute values, the processing time is quite large. Clearly, this is due to the fact that SVE-enabled HW is not available at the time of writing, and all benchmarks are executed inside the ARM SVE instruction emulator.

DNN signal processing performance: Accuracy and complexity

In [52] and [57], Carmichael et al. show an architecture using Posits in DNNs called *Deep Positron*, using an exact MAC

technique on eight-bit low-precision formats. The architecture has been tested on the MNIST, Fashion–MNIST, and other data sets, reporting no drop in accuracy with regard to Float32. Another approach to deep learning with low-bit numbers has been tested in [39], using logarithmic numbers with a residential NN (ResNet)-50 architecture on Imagenet, resulting in a 0.9 percentage point drop when shifting from Float32 to logarithmic representation. We have integrated the cppPosit library in a DNN C++ library called *tiny-DNN* [58] that is capable of supporting various computing arithmetic, such as BFLOAT16, Flexpoint, and Posits. Then, we tested the accuracy of different network models in image classification benchmarks, such as MNIST, Fashion–MNIST, CIFAR-10, and GTRSB, using the FDP technique. For the MNIST data set, we registered a drop of 0.9 percentage points when testing the model from Float32 to Posit8. For GTRSB, we registered a drop of 0.2 percentage points, instead. For other Posit configurations with 16, 14, 12, and 10 bits, we registered no drop in accuracy from Float32 to the Posit type.

Table 4. The L1 operations performance processing-time comparison between nonvectorized (naive) and vectorized (SVE-X) approaches.

	FastSigmoid (ms)		FastTanh (ms)		FastELU (ms)	
Posit	**8,0**	**16,0**	**8,0**	**16,0**	**8,0**	**16,0**
Version						
Naive	3.08	3.41	5.76	7.24	8.12	8.54
SVE-128	0.73	1.51	1.32	2.65	1.29	2.6
SVE-256	0.59	1.05	1.18	1.83	1.16	1.79
SVE-512	0.43	0.62	0.69	1.09	0.69	1.05
SVE-1024	0.29	0.39	0.48	0.72	0.46	0.68
SVE-2048	0.22	0.28	0.36	0.5	0.35	0.47

Each timing result comes from the function computation on a vector of 8,192 items.

Table 5. The 3 × 3 convolution and pooling processing-time comparison on two common Posit configurations with 225 × 225 random images.

	Maximum Pooling (ms)		Convolution (ms)	
Posit	**8,0**	**16,0**	**8,0**	**16,0**
Version				
Naive	49.7	59.41	80.67	80.84
SVE-128	9.51	26.52	24.02	37.99
SVE-256	8.89	22.06	11.66	21.49
SVE-512	6.96	14.69	6.85	14.03
SVE-1024	5.12	11.84	6.38	12.88
SVE-2048	4.13	9.76	3.65	8.81

The naive approach is the nonvectorized one. The other approaches are with incremental SVE–vector registers.

Table 6. The image processing time for various very deep NN models using Posit8,0.

Version	Alexnet Time(s)	ResNet-34 Time(s)	VGG-16 Time(s)	VGG-19 Time(s)	ResNet-150 Time(s)
Naive	40.06	146.07	590.68	675.32	779.7
SVE-128	2.76	10.07	40.74	46.57	53.77
SVE-256	2.64	9.61	38.88	44.45	51.32
SVE-512	2.54	8.93	36.12	41.3	47.68
SVE-1024	2.44	8.92	36.06	41.23	47.6
SVE-2048	2.34	8.9	35.97	41.13	47.48

For this benchmark, random red–green–blue 224 × 224 images are employed.

Benchmark data sets and examples of achievable results

We have considered different standard data sets, such as the one shown in Figure 5, and standard CNN architectures, including the one in Figure 6. In particular, for the MNIST and GTSRB benchmarks, we trained customized CNN variants of the one reported in Figure 6, including Posit-related optimizations to the convolutional and activation layers. For the Fashion–MNIST benchmark, we used a pretrained model with a starting accuracy of 95%. For CIFAR-10, we used the VGG-16 pretrained model [59]. All the networks were initially trained using Float32 and then evaluated on the corresponding test sets, converting each Float32-trained model using different Posit configurations. Furthermore, to provide a fair timing–accuracy tradeoff comparison, the Float32 model has been tested exploiting the SoftFloat library for SW-emulated floating-point numbers.

MNIST, Fashion–MNIST, and CIFAR-10

Table 7 presents the results obtained on three well-known classification benchmarks: MNIST, Fashion–MNIST, and CIFAR-10. MNIST is a digit-recognition problem, while Fashion–MNIST has been designed as a more complex drop-in replacement for the MNIST data set, providing more general classes to be recognized (such as fashion products). Furthermore, CIFAR-10 consists in an even more complex task, bringing three-channel images in the data set. As reported, the tests on the model with the different types show that Posits with zero exponent bits and sized from 12 to 14 bits can be a perfect replacement for Float32, while those with 10 and eight bits can replace Float32 with some drop in accuracy. The same holds for the Fashion–MNIST data set.

Note how the processing time [on an Intel seventh-generation (Kaby Lake) i7 processor] for a single-image inference of the VGG-16 model on a CIFAR-10 sample is expressed in

FIGURE 5. The GTRSB data set example.

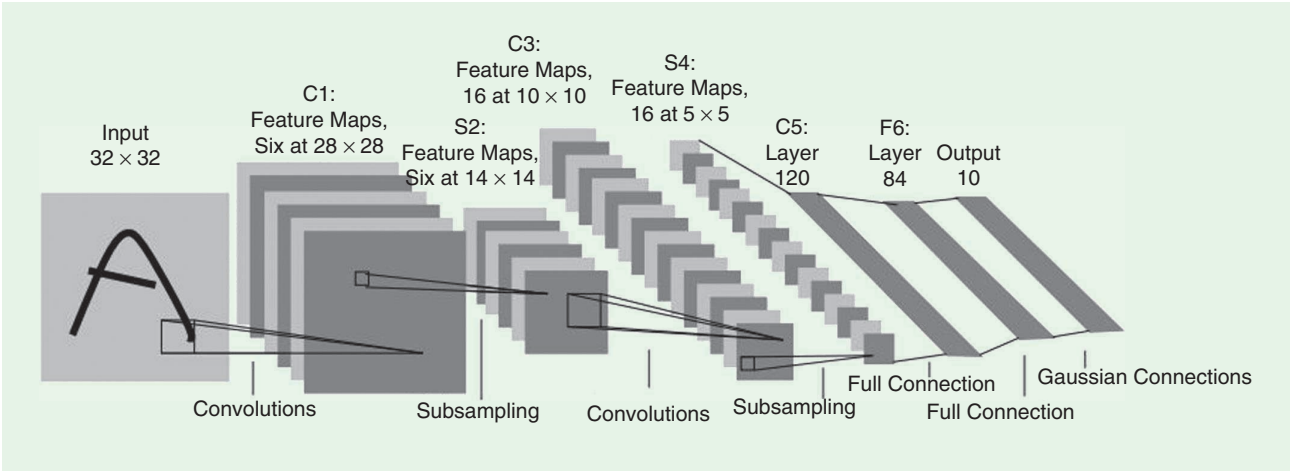


FIGURE 6. The LeNet-5 architecture as described in [49]. Some customization has been added to the network to better fit our goals: the activation function has been changed to FastTanh (as described before) for the MNIST data set and to a fast approximation of the ELU for the GTSRB data set. The input size of the first layer has been extended to hold the $64 \times 64 \times 3$ color images of the GTSRB data sets.

Table 7. The accuracy and processing time obtained on MNIST, Fashion–MNIST, and CIFAR-10 data sets.

Type	MNIST			Fashion–MNIST			CIFAR-10		
	Accuracy (%)	Time (ms)	NCT[1]	Accuracy (%)	Time (ms)	NCT[1]	Accuracy (%)	Time (s)	NCT[1]
SoftFloat32	99.4	8.8	—	95	41.9	—	93.75	7.75	—
Posit16,0	99.4	5.2	0.59	95	13.6	0.32	93.75	2.55	0.32
Posit14,0	99.4	4.6	0.52	95	13.5	0.32	93.75	2.49	0.32
Posit12,0	99.4	4.6	0.52	95	13.5	0.32	93.75	2.44	0.31
Posit10,0	99.3	4.6	0.52	95	13.4	0.32	93.75	2.4	0.3
Posit8,0	98.5	3.8	0.43	94	13.4	0.32	85	2.34	0.3

The processing time is evaluated as the mean per-sample inference time on the test set of the relative data set.
[1]Posit computing times are normalized against SoftFloat32 computing times.

Table 8. The accuracy–processing time tradeoff obtained on the GTRSB data set.

	GTRSB		
Type	Accuracy (%)	Time (ms)	NCT[1]
SoftFloat32	94	15.86	—
Posit16,0	94	6.37	0.4
Posit14,0	94	5.21	0.32
Posit12,0	94	5.08	0.32
Posit10,0	94	5	0.31
Posit8,0	93.8	4	0.25

[1]Posit computing times are normalized against SoftFloat32 computing times.

seconds, highlighting the infeasibility of this model for traditional CPU architectures. However, we are moving toward GPU-enabled DNN libraries, as described in the "Conclusions and Road Maps" section. For comparison, an entire training epoch of 60,000 CIFAR-10 samples on a Resnet-50 architecture takes only approximately 30 s on a dual-GPU (Tesla T4) configuration, thus only 0.5 ms for the forward and backward passes (including the weight update). It should also be noted that to make the comparison fair, we evaluate, in Tables 6 and 7, the SW implementation of Posits (using our developed cppPosit library) against an SW implementation of Floats (the SoftFloat library). From Tables 6 and 7, we can observe that moving from SoftFloat32 to Posit8,0, we get (roughly) the same classification accuracy on all considered data sets but with a reduction in the computing time of roughly a factor of three.

Automotive benchmarks: The traffic sign recognition problem

In this section, we report the results obtained on a classification benchmark related to assisted/autonomous driving. Benchmarks were executed on an Intel seventh-generation (Kaby Lake) i7-7560U processor with two cores at 2.4 GHz. The GTRSB is a baseline benchmark for road sign recognition, which is very interesting as an automotive task. Table 8 shows that, in this case also, Posits from 12 to 16 bits, and even 10 bits, can be a perfect replacement for Float32, while Posit8,0 performs well with a little drop in accuracy. We have also started an activity to assess the performance of Posits using the YOLO approach [60], [61] and Apollo [62] (http://apollo.auto/) heterogeneous framework, and the achieved results confirm what we already obtained with the GTRSB, MNIST, and Fashion–MNIST data sets. Moreover, we began an activity to assess Posit performance in semantic segmentation tasks (such as pixel- and instance-level classification [33], [34]) on famous data sets, such as CityScapes (see Figure 7).

FIGURE 7. The CityScapes data set example of the semantic segmentation of a road in Stuttgart, Germany.

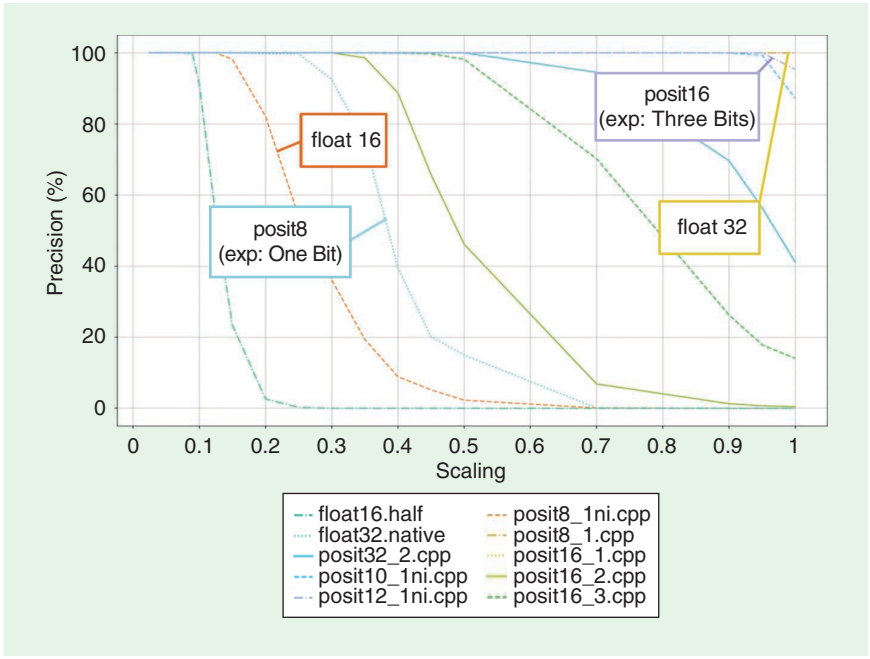


FIGURE 8. The performance of the *k*-NN using various data types on a single data set employing different values for the scaling factor.

The results we are obtaining are in line with those from the MNIST, Fashion–MNIST, and GTRSB data sets.

k-nearest neighbors results

The *k*-nearest neighbors (*k*-NN) algorithm is ubiquitous in pattern-recognition problems. It can be used to segment images and to compute the normal vectors to each point of a point cloud obtained by a lidar sensor mounted on a car. The *k*-NN algorithm finds the *k* nearest neighbors of a given point from those in a given data set. We have compared the performance of the *k*-NN when using Posits and Floats and, again, found that the accuracy of Posit16,0 is very close to that of Float32 (see Figure 8) and that a Posit8,0 outperforms Float16. These results have been obtained on a single data set, scaling it multiple times to reduce the dynamic range of the input data (thus enabling low-precision data types to be competitive with Float32). More details can be found in [63]. The obtained results confirm that Posits are powerful in a number of machine learning applications, meaning that implementing Posit-based HW accelerators will be beneficial for numerous different applications.

Next experiments

We are working toward the implementation of other fast approximated functions (e.g., the ELU). We are currently porting our cppPosit-based tiny-DNN library on the ARM instruction emulator used within the Horizon 2020 EPI [64] to exploit the SVE-2 as much as possible (providing a vectorization back end for the cppPosit library). We are also planning to test our SW on available simulators, such as GEM5, SESAM, and MUSA, to provide useful feedback to the ongoing EPI processor codesign process.

Conclusions and road maps

In this article, we have reviewed the state of the art of DNN signal processing for autonomous driving applications and the quest for novel representations of real numbers that must be both efficient and reliable. We have seen how Posit is a suitable drop-in replacement for the IEEE 754 standard, and we have assessed its potentialities in autonomous driving applications. Implementations with both SW libraries and HW–SW embedded systems, from academia and industry, have been discussed. The achieved results when combining Posit arithmetic with DNNs are promising in terms of the tradeoff between accuracy and processing time. From this and related works, it is clear that the current challenges are 1) the development of real-time and low-power accelerators for performing DNN inference at the edge, 2) the development of methods for DNN verification and validation for the high coverage rates required by standards for safety-critical applications, and 3) moving toward a GPU-enabled DNN library, such as Tensorflow, to build, train, and evaluate even more complex models once they are integrated with our cppPosit library. Furthermore, we plan to test our approach on GPU-enabled ARM devices, such as NVIDIA

Jetson boards; mobile devices that do not employ GPUs; and even without the FPU.

Acknowledgments

This work was partially funded by the Horizon 2020 European Processor Initiative (grant agreement 826647) and the Italian Ministry of Education and Research through the CrossLab project (departments of excellence).

Authors

Marco Cococcioni (marco.coccioni@unipi.it) has been an associate professor in the Department of Information Engineering, University of Pisa, Pisa, Italy, since 2016. He is on the editorial board of four journals indexed by Scopus and coauthored more than 90 scientific contributions. He has been the general chair of three IEEE conferences and a program committee member of 50-plus international conferences in the area of computational intelligence. He is a member of three IEEE task forces: Genetic Fuzzy Systems, Computational Intelligence in Security and Defense, and Intelligent System Applications. He is a Senior Member of IEEE and ACM.

Federico Rossi (federico.rossi@ing.unipi.it) is a Ph.D. student in the Department of Information Engineering, University of Pisa, Pisa, Italy. In 2019, he received his M.S. degree magna cum laude in computer engineering. He is currently involved in the European Processor Initiative project. His research interests include alternative real-number representations and their applications to deep neural networks for the automotive environment.

Emanuele Ruffaldi (emanuele.ruffaldi@mmimicro.com) is a senior software engineer at MMI, Pisa, Italy, working on robotic-assisted microsurgery. Formerly, he was an assistant professor in the Perceptual Robotics Laboratory, Scuola Superiore Sant'Anna, Pisa, Italy. His research interests include machine learning for human–robot interaction and embedded artificial intelligence. He coauthored more than 100 scientific publications and one patent. He is a Senior Member of IEEE and has served as publicity chair for the IEEE Haptics Technical Committee.

Sergio Saponara (sergio.saponara@unipi.it) is a full professor at the University of Pisa, Pisa, Italy, where he is responsible for automotive-electronics, hardware-security, and embedded-system activities; president of B.Sc. and M.Sc. degrees in electronic engineering; and head of the university's participation in the European Processor Initiative project. He is also director of a summer school on the industrial Internet of Things and a specialization course on automotive powertrain electrification. The coauthor of more than 300 scientific publications and approximately 20 patents, he is an associate editor of many peer-reviewed journals. He is a Senior Member of IEEE and is an IEEE Distinguished Lecturer.

Benoît Dupont de Dinechin (benoit.dinechin@kalray.eu) is the chief technology officer at Kalray, Montbonnot-Saint-Martin, France. He received his engineering degree from ISAE-SUPAERO, Toulouse, France, and his Ph.D. degree in computer systems from the Université Pierre et Marie Curie, Paris, France. He completed his postdoctoral studies at McGill University, Montréal. He is the main architect of the Kalray very-long instruction word cores and coarchitect of the company's massively parallel processor arrays. He defined the Kalray software road map and still contributes to its implementation. Before joining Kalray, he led R&D for the Software Tools division at STMicrelectronics, becoming a fellow in 2008. Prior to STMicrelectronics, he developed the software pipeliner of the Cray T3E production compilers at the Cray Research Park, Eagan, Minnesota.

References

[1] S. Saponara and B. Neri, "Radar sensor signal acquisition and multidimensional FFT processing for surveillance applications in transport systems," *IEEE Trans. Instrum. Meas.*, vol. 66, no. 4, pp. 604–615, 2017. doi: 10.1109/TIM.2016.2640518.

[2] L. L. Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent advances and trends in on-board embedded and networked automotive systems," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 1038–1051, 2019. doi: 10.1109/TII.2018.2879544.

[3] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, SAE Standard J3016_201806, 2018.

[4] J. Royo-Alvarez, M. Martínez-Ramón, J. Muñoz-Marí, and G. Camps-Valls, Eds., *From Signal Processing to Machine Learning*. Hoboken, NJ: Wiley, 2018, ch. 1, pp. 1–11.

[5] L. Deng, "Artificial intelligence in the rising wave of deep learning: The historical path and future outlook [perspectives]," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 180–177, 2018. doi: 10.1109/MSP.2017.2762725.

[6] IEEE Signal Processing Society (SPS) ASI. Accessed: Oct. 1, 2020. [Online]. Available: https://ieeeasi.signalprocessingsociety.org/

[7] M. T. McCann, K. H. Jin, and M. Unser, "Convolutional neural networks for inverse problems in imaging: A review," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 85–95, 2017. doi: 10.1109/MSP.2017.2739299.

[8] A. Arnab, S. Zheng, S. Jayasumana, B. Romera-Paredes, M. Larsson, A. Kirillov, B. Savchynskyy, C. Rother et al., "Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 37–52, 2018. doi: 10.1109/MSP.2017.2762355.

[9] S. F. Dodge and L. J. Karam, "Quality robust mixtures of deep neural networks," *IEEE Trans. Image Process.*, vol. 27, no. 11, pp. 5553–5562, 2018. doi: 10.1109/TIP.2018.2855966.

[10] P. Nousi, A. Tefas, and I. Pitas, "Deep convolutional feature histograms for visual object tracking," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP 2019)*, 2019, pp. 8375–8379. doi: 10.1109/ICASSP.2019.8683004.

[11] A. Voulodimos, N. D. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, Feb. 2018, Art. no. 7068349. doi: 10.1155/2018/7068349.

[12] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, 2017. doi: 10.1109/MSP.2017.2693418.

[13] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017. doi: 10.1109/MSP.2017.2743240.

[14] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, "Advanced deep-learning techniques for salient and category-specific object detection: A survey," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 84–100, 2018. doi: 10.1109/MSP.2017.2749125.

[15] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 126–136, Jan. 2018. doi: 10.1109/MSP.2017.2765695.

[16] P. Nousi, E. Patsiouras, A. Tefas, and I. Pitas, "Convolutional neural networks for visual information analysis with limited computing resources," in *Proc. 25th IEEE Int. Conf. Image Processing (ICIP'18)*, 2018, pp. 321–325. doi: 10.1109/ICIP.2018.8451600.

[17] D. Reinhardt, U. Dannebaum, M. Scheffer, and M. Traub, "High performance processor architecture for automotive large scaled integrated systems within the european processor initiative research project," SAE International, Warrendale, PA, SAE Tech. Paper 2019-01-0118, 2019.

[18] U. Köster, T. Webb, X. Wang, M. Nassar, A. K. Bansal, W. Constable, O. Elibol, S. Gray et al., "Flexpoint: An adaptive numerical format for efficient training of deep neural networks," in *Proc. Advances Neural Information Processing Systems*, 2017, pp. 1742–1752.

[19] V. Popescu, M. Nassar, X. Wang, E. Tumer, and T. Webb, "Flexpoint: Predictive numerics for deep learning," in *Proc. IEEE 25th Symp. Computer Arithmetic (ARITH)*, June 2018, pp. 1–4. doi: 10.1109/ARITH.2018.8464801.

[20] G. Tagliavini, A. Marongiu, and L. Benini, "FlexFloat: A software library for transprecision computing," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 39, no. 1, pp. 145–156, 2020. doi: 10.1109/TCAD.2018.2883902.

[21] A. C. I. Malossi, M. Schaffner, A. Molnos, L. Gammaitoni, G. Tagliavini, A. Emerson, A. Toms, D. S. Nikolopoulos et al., "The transprecision computing paradigm: Concept, design, and applications," in *Proc. Design, Automation Test Europe Conf. Exhibition (DATE)*, Mar. 2018, pp. 1105–1110. doi: 10.23919/DATE.2018.8342176.

[22] G. Venkatesh, E. Nurvitadhi, and D. Marr, "Accelerating deep convolutional networks using low-precision and sparsity," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 2861–2865. doi: 10.1109/ICASSP.2017.7952679.

[23] G. Srivastava, D. Kadetotad, S. Yin, V. Berisha, C. Chakrabarti, and J. Seo, "Joint optimization of quantization and structured sparsity for compressed deep neural networks," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP 2019)*, 2019, pp. 1393–1397. doi: 10.1109/ICASSP.2019.8682791.

[24] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, Quantized neural networks: Training neural networks with low precision weights and activations. 2016. [Online]. Available: http://arxiv.org/abs/1609.07061

[25] M. Cococcioni, E. Ruffaldi, and S. Saponara, "Exploiting posit arithmetic for deep neural networks in autonomous driving applications," in *Proc. Int. Conf. Electrical and Electronic Technologies Automotive*, 2018, pp. 1–6. doi: 10.23919/EETA.2018.8493233.

[26] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: Posit arithmetic," *Supercomput. Front. Innov.*, vol. 4, no. 2, pp. 71–86, 2017. doi: 10.14529/jsfi170206.

[27] P. Mallozzi, P. Pelliccione, A. Knauss, C. Berger, and N. Mohammadiha, "Autonomous vehicles: State of the art, future trends, and challenges," in *Automotive Systems and Software Engineering*, Y. Dajsuren and M. van den Brand, Eds. Cham, Switzerland: Springer-Verlag, 2019, pp. 347–367.

[28] A. Woo, B. Fidan, and W. W. Melek, "Localization for autonomous driving," in *Handbook of Position Location: Theory, Practice, and Advances*. Hoboken, NJ: Wiley, 2019, ch. 29, pp. 1051–1087.

[29] Y. Wenhui and Y. Fan, "Lidar image classification based on convolutional neural networks," in *Proc. Int. Conf. Computer Network, Electronic and Automation (ICCNEA)*, 2017, pp. 221–225. doi: 10.1109/ICCNEA.2017.37.

[30] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Netw.*, vol. 32, pp. 323–332, Aug. 2012. doi: 10.1016/j.neunet.2012.02.016.

[31] V.-D. Hoang, M.-H. Le, T. T. Tran, and V.-H. Pham, "Improving traffic signs recognition based region proposal and deep neural networks," in *Intelligent Information and Database Systems*, N. T. Nguyen, D. H. Hoang, T.-P. Hong, H. Pham, and B. Trawiński, Eds. Cham, Switzerland: Springer-Verlag, 2018, pp. 604–613.

[32] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth et al., "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223.

[33] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. European Conf. Computer Vision (ECCV)*, 2018, pp. 833–851.

[34] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu et al., "Searching for MobileNetV3," in *Proc. Int. Conf. Computer Vision (ICCV)*, 2019, pp. 1314–1324.

[35] M. Courbariaux, Y. Bengio, and, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY: Curran Associates Inc., 2015, pp. 3123–3131.

[36] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.

[37] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proc. 32nd Int. Conf. Machine Learning (ICML'15)*, 2015, vol. 37, pp. 1737–1746.

[38] P. Lindstrom, "Universal coding of the reals using bisection," in *Proc. Conf. Next Generation Arithmetic (CoNGA'19)*, 2019, pp. 7:1–7:10. doi: 10.1145/3316279.3316286.

[39] J. Johnson, Rethinking floating point for deep learning. 2018. [Online]. Available: http://arxiv.org/abs/1811.01721

[40] M. G. Arnold, J. Garcia, and M. J. Schulte, "The interval logarithmic number system," in *Proc. 16th IEEE Symp. Computer Arithmetic*, 2003, pp. 253–261. doi: 10.1109/ARITH.2003.1207686.

[41] J. Gustafson, "Posits and quires: Freeing programmers from mixed-precision decisions," in *Proc. 34th Int. Supercomputing Conf. High Performance (ISC'19)*, Frankfurt, Germany, June 16–20, 2019.

[42] E. Ruffaldi, "cppPosit," GitHub, San Francisco. Accessed: Oct. 1, 2020. [Online]. Available: https://github.com/eruffaldi/cppPosit

[43] M. Cococcioni, F. Rossi, E. Ruffaldi, and S. Saponara, "A fast approximation of the hyperbolic tangent when using Posit numbers and its application to deep neural networks," in *Applications in Electronics Pervading Industry, Environment and Society*, S. Saponara and A. De Gloria, Eds. Cham, Switzerland: Springer-Verlag, 2020, pp. 213–221.

[44] M. K. Jaiswal and H. K.-H. So, "Universal number posit arithmetic generator on FPGA," in *Proc. Design, Automation Test Europe Conf. Exhibition (DATE)*, 2018, pp. 1159–1162. doi: 10.23919/DATE.2018.8342187.

[45] A. Podobas and S. Matsuoka, "Hardware implementation of POSITs and their application in FPGAs," in *Proc. IEEE Int. Parallel and Distributed Processing Symp. Workshops (IPDPSW)*, 2018, pp. 138–145. doi: 10.1109/IPDPSW.2018.00029.

[46] R. Chaurasiya, J. Gustafson, R. Shrestha, J. Neudorfer, S. Nambiar, K. Niyogi, F. Merchant, and R. Leupers, "Parameterized posit arithmetic hardware generator," in *Proc. IEEE 36th Int. Conf. Computer Design (ICCD)*, 2018, pp. 334–341. doi: 10.1109/ICCD.2018.00057.

[47] M. K. Jaiswal and H. K. So, "PACoGen: A hardware Posit Arithmetic Core Generator," *IEEE Access*, vol. 7, pp. 74,586–74,601, June 2019. doi: 10.1109/ACCESS.2019.2920936.

[48] L. van Dam, J. Peltenburg, Z. Al-Ars, and H. P. Hofstee, "An accelerator for posit arithmetic targeting posit level 1 BLAS routines and pair-HMM," in *Proc. Conf. Next Generation Arithmetic (CoNGA'19)*, 2019, pp. 5:1–5:10. doi: 10.1145/3316279.3316284.

[49] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. doi: 10.1038/nature14539.

[50] M. Cococcioni, F. Rossi, E. Ruffaldi, and S. Saponara, "Fast approximations of activation functions in deep neural networks when using Posit arithmetic," *Sensors*, vol. 20, no. 5, p. 1515, 2020. doi: 10.3390/s20051515.

[51] Netlib, "Cephes mathematical function library." Accessed: Oct. 1, 2020. [Online]. Available: http://www.netlib.org/cephes/

[52] Z. Carmichael, H. F. Langroudi, C. Khazanov, J. Lillie, J. L. Gustafson, and D. Kudithipudi, "Performance-efficiency trade-off of low-precision numerical formats in deep neural networks," in *Proc. Conf. Next Generation Arithmetic 2019 (CoNGA'19)*, 2019, pp. 3:1–3:9. doi: 10.1145/3316279.3316282.

[53] U. Kulisch, "An axiomatic approach to rounded computations," *Numerische Mathematik*, vol. 18, pp. 1–17, Feb. 1971. doi: 10.1007/BF01398455.

[54] B. D. de Dinechin, "Consolidating high-integrity, high-performance, and cybersecurity functions on a manycore processor," in *Proc. 56th ACM/IEEE Design Automation Conf. (DAC'19)*, 2019, pp. 1–4, doi: 10.1145/3316781.3323473.

[55] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2018)*, Salt Lake City, UT, June 18–22, 2018, pp. 2704–2713. doi: 10.1109/CVPR.2018.00286.

[56] N. Brunie, "Modified fused multiply and add for exact low precision product accumulation," in *Proc. IEEE 24th Symp. Computer Arithmetic (ARITH)*, July 2017, pp. 106–113. doi: 10.1109/ARITH.2017.29.

[57] Z. Carmichael, H. F. Langroudi, C. Khazanov, J. Lillie, J. L. Gustafson, and D. Kudithipudi, "Deep Positron: A deep neural network using the Posit number system," in *Design, Automation and Test Europe Conf. and Exhibition (DATE'19)*, 2019, pp. 1421–1426. doi: 10.23919/DATE.2019.8715262.

[58] "tiny-dnn," GitHub, San Francisco. Accessed: Oct. 1, 2020. [Online]. Available: https://github.com/tiny-dnn/tiny-dnn

[59] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition. 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[60] M. Yang, S. Wang, J. Bakita, T. Vu, F. D. Smith, J. H. Anderson, and J. Frahm, "Re-thinking CNN frameworks for time-sensitive autonomous-driving applications: Addressing an industrial challenge," in *Proc. IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS)*, 2019, pp. 305–317. doi: 10.1109/RTAS.2019.00033.

[61] S. Goel, A. Baghel, A. Srivastava, A. Tyagi, and P. Nagrath, "Detection of emergency vehicles using modified YOLO algorithm," in *Intelligent Communication, Control and Devices*, S. Choudhury, R. Mishra, R. G. Mishra, and A. Kumar, Eds. Singapore: Springer-Verlag, 2020, pp. 671–687.

[62] H. Tabani, L. Kosmidis, J. Abella, F. J. Cazorla, and G. Bernat, "Assessing the adherence of an industrial autonomous driving framework to ISO 26262 software guidelines," in *Proc. 56th ACM/IEEE Design Automation Conf. (DAC'19)*, 2019, pp. 1–6. doi: 10.1145/3316781.3317779.

[63] M. Cococcioni, F. Rossi, E. Ruffaldi, and S. Saponara, "Novel arithmetics to accelerate machine learning classifiers in autonomous driving applications," in *Proc. 26th IEEE Int. Conf. Electronics Circuits and Systems (ICECS'19)*, 2019, pp. 779–782. doi: 10.1109/ICECS46596.2019.8965031.

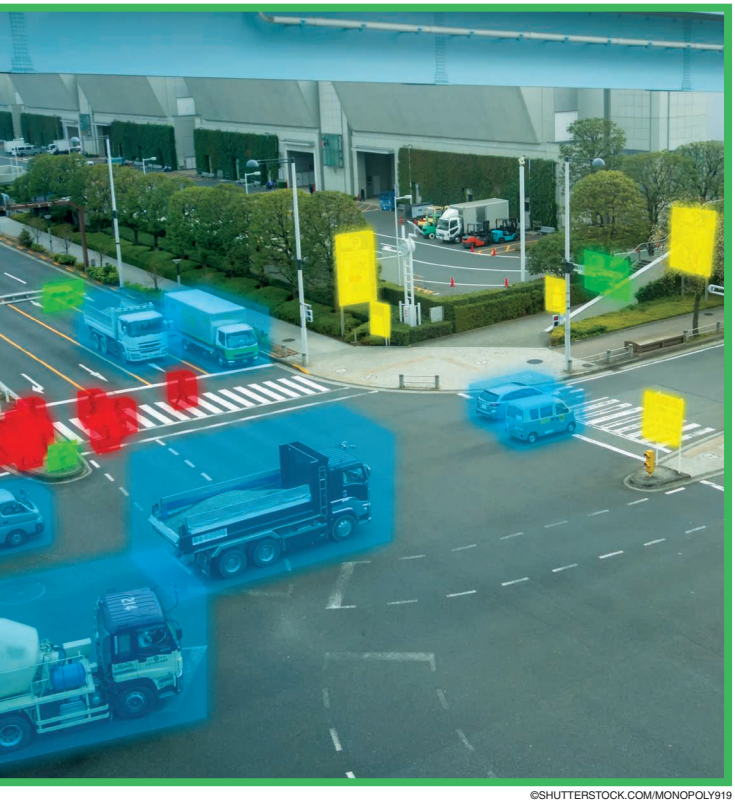
[64] European Processor Initiative, "European Processor Initiative, an H2020 project," 2019–2021. [Online]. Available: https://www.european-processor-initiative.eu/

SP

Dean Deter, Chieh (Ross) Wang,
Adian Cook, and Nolan Perry

Simulating the Autonomous Future

A look at virtual vehicle environments and how to validate simulation using public data sets



©SHUTTERSTOCK.COM/MONOPOLY919

The rapid evolution of autonomous vehicles (AVs) has exposed the need for fast-paced development and testing processes of a variety of perception, planning, and control algorithms. To expedite development, the AV industry and researchers leverage virtual vehicle environments to simulate a range of test scenarios that may otherwise be costly or difficult to conduct on a real test track. However, the various virtual environments may have different results depending on the fidelity of various simulation features, such as vehicle dynamics, sensor simulation, and environment recreation. This tutorial article examines a proposed framework for constructing, parameterizing, and validating a virtual vehicle environment using an existing AV data set. First, an overview of several open source and commercially available simulation tools, including their associated workflows, for scene and scenario creation is presented. Next, various open AV data sets are examined to inform the data set selection for the validation framework. Then, an example workflow of recreating a real-world scene from the selected data set in a simulation tool with various emulated sensors parameterized to match the data set is demonstrated. Finally, an example AV-perception algorithm is subjected to data streams from virtual and real-world environments and suggested metrics for analyzing the results are discussed.

Introduction

One of the primary focus areas for advancing the capability of AVs is to develop, train, and test AV algorithms using imagery and sensor data under a variety of environmental and driving conditions. A main challenge of this focus area is the sheer volume of testing and validation needed to complete many of the edge cases presented by AV operation. As a start, millions of miles have been driven by AVs, and a large amount of data has been collected on test tracks, closed roads, and public roads by various AV entities. For example, real-world data sets, such as the KITTI Vision Benchmark Suite [1], the Berkeley Deep Drive (BDD) [2], the Lyft Level 5 AV data set [3], the Waymo Open Dataset [4], and the nuScenes data set [5] have been collected by vehicles instrumented with a variety of sensors, such

Digital Object Identifier 10.1109/MSP.2020.2984428
Date of current version: 24 December 2020

as cameras, lidars, radars, Global Navigation Satellite System (GLONASS), and inertial measurement units (IMUs). These data sets have been used for benchmarking and as a toolset for training and testing AV algorithms, yet, researchers realize that there are still many arbitrary and/or dangerous situations that cannot be covered by the current extensive testing and data collection efforts. Some data sets, including the INTERACTION data set [6], have tried to address these situations but still lack vehicle data such as real latitude, longitude, raw image data, or vehicle sensor streams typically used for testing and validation.

To address these challenges, researchers have developed open source virtual environments using game engines such as Unreal Engine and Unity. These virtual environments, such as CARLA, LGSVL Simulator, SYNTHIA [7], and Virtual KITTI [8], aim to provide a flexible platform for developing and testing AV algorithms under a variety of possible traffic, lighting, weather, and other environmental and driving conditions. Although these virtual environments can be photorealistic, they are primarily used to generate scenarios for on-road edge cases, computer vision development, and machine learning (ML) applications; none of them have yet emulated a full suite of high-fidelity or physics-based AV sensors (e.g., cameras, lidars, and radars).

In addition to open source environments, several companies including AVSimulation, Cognata, dSPACE, ESI, IPG Automotive, Metamoto, MSC Software, and Siemens offer commercial toolchains for the development and validation of customers' AV algorithms. In contrast to most open source toolchains, many of these environments offer full sensor support and characterization as well as a high level of scenario customization. Furthermore, these tools often contain support for hardware-in-the-loop (HIL) (e.g., controller-in-the-loop, camera-in-the-loop, and sensor-in-the-loop), have full development suites, and can have high levels of simulation photorealism. Established ties to other commercial tools are also common, including software packages such as Aimsun, MATLAB/Simulink, GT-Suite, rFpro, SUMO, and VISSIM, which can help streamline workflow and cut development time for researchers, computer scientists, and engineers. A selection of both open source and commercial toolchains are covered in this article, as both solutions work to archive answers and results to varying research goals and challenges for AV testing and development as well as academic pursuits.

All of these tools have highlighted a significant need for R&D into the ability to generate synthetic sensor data streams (in addition to synthetic imagery) from virtual environments injected into AV controllers or a full AV on a dynamometer. The AV research community has begun to investigate whether artificial intelligence (AI) drivers and ML perception algorithms can be trained on purely synthetic data, a combination of synthetic and recorded data, or if only real-world data are acceptable. Furthermore, whether AI algorithms developed to drive the vehicle in a virtual environment translate to controlling the vehicle in the real world where dynamics and perception are noticeably different must also be determined.

To answer these questions and bridge the gap between virtual environments and existing real-world data sets, there is a need to develop a framework that not only creates a "digital twin" of the real-world environment but also emulates the data streams of a full suite of sensors that are commonplace on AVs to increase fidelity and accuracy. This would enable realistic autonomous driving simulations in virtual environments. Moreover, it is important to evaluate how similarly or differently various perception and control algorithms respond to synthetic data injection versus data feeds from real-world sensors. All of these improvements depend on the ability of the aforementioned tools to continuously improve and serve as the centerpiece of AV testing and development.

Current state of selected simulators

When choosing a simulator for AV algorithm testing and deployment, there are several factors to be considered. Looking specifically at vision-based perception and controls validation, the sensors made available from the data sets and the graphics performance are critical in determining the proper environment. Proper instrumentation and realistic feedback from sensor streams ensure that perception and control algorithms are being thoroughly tested. The resulting algorithms are more robust and can be deployed in real situations on a shorter timeline. Other factors, such as weather, environment photorealism, and ties to external hardware, are also important to consider. For example, tools may have additional capabilities that allow for the support of real-world sensor inputs (i.e., cameras) that can be integrated with the simulation. Rapidly testing various configurations in both hardware and software can also shorten development time and push algorithms faster to market. On the front end of development, the ability to leverage map-generation packages such as open tools (i.e., OpenDRIVE [9], OpenCRG [10], and OpenSCENARIO [11]) and other map tools can impact synthetic environment selection. Regarding cloud capabilities, some tools may offer either local or cloud computing or a combination of both. A simple overview of all the available tools presented and their supported sensor types can be found in Table 1 and are further discussed in the next sections.

MSC software virtual test drive

Virtual test drive (VTD) is a vehicle simulation package developed by VIRES Simulationstechnologie [12], which was later bought by MSC Software. Several open standards and tools developed by VIRES that were mentioned previously, such as OpenDRIVE, OpenCRG, and OpenSCENARIO, are fully supported in this environment. These associated open standards and tools allow support for road networks, road surfaces, and dynamic content, respectively. VTD includes support for a variety of sensors including cameras, radars, and lidars. In a sample workflow, road design and population could be done using the Open toolchain (i.e., OpenDRIVE, OpenCRG, and OpenSCENARIO) and customized using the VTD Road Designer. Road Designer includes road designs for Europe, the United States, and China with extensive libraries of objects and textures. Using v-Traffic and v-Scenario simulation modules, the scenario can be further

Table 1. A summary of sensor emulation available in different virtual environment simulators.

	openDRIVE	GLONASS	IMU	Camera	Lidar	Radar	Cloud Only
MSC VTD	Y	N	N	Y	Y	Y	N
dSPACE ASM	Y	Y	Y	Y	Y	Y	N
CARLA	Y	Y	N	Y	Y	N	N
Metamoto	Y	Y	Y	Y	Y	Y	Y
Cognata	Y	Y	Y	Y	Y	Y	N
IPG CarMaker	Y	Y	Y	Y	Y	Y	N

VTD: Virtual test drive; ASM: automotive simulation models.

customized with traffic, pedestrians, or other simulation assets. v-IG provides real environmental effects such as weather, camera effects, and real-time ray-tracing for the simulated environment. Finally, v-Task Control provides the core module responsible for task management, simulation control, and playback ability. VTD also provides cosimulation ability, with support for commonly used external tools such as MATLAB/Simulink.

dSPACE Automotive Simulation Models

Automotive Simulation Models (ASM) is a vehicle simulation package offered by dSPACE, Inc. [13]. Included with a significant MATLAB/Simulink integration, dSPACE supports GLONASS/IMU, cameras, lidars, and radars. In a typical workflow, a Simulink vehicle model can be virtually instrumented with a chosen set of sensors and vehicle characteristics to provide realistic vehicle dynamics. Next, the Simulink model links with dSPACE ModelDesk, where the user can parameterize the vehicle, associated simulation objects, the road network, and the scenario under test. Then, ModelDesk communicates with MotionDesk to visualize the vehicle, objects, and road in the scene performing the scenario under test. The environmental conditions and superficial objects can be further edited in MotionDesk. Finally, virtual sensors can be added to the vehicle and parameterized in MotionDesk, which streams sensor feedback to the Simulink model. As ASM is a dSPACE product, it leverages all of the additional hardware and software native to the dSPACE family, including the SCALEXIO HIL systems, MicroAutoBox controllers, and other dSPACE products.

CARLA

CARLA is an open source simulation tool built in the Unreal Engine environment [14]. The CARLA simulator includes sensor support for cameras, lidars, depth, semantic segmentation, obstacles, collision, and lane invasion. Several additional tools for map generation are available, including a CARLA standalone graphical user interface (GUI) or other external tools such as VectorZero's RoadRunner [15]. Because of the tool's integration with Unreal Engine, heavy customization is possible using Unreal Editor and associated asset libraries. In a typical workflow, the map is first generated using RoadRunner or Carla's GUI, ported to Unreal Engine for asset population or environment customization, and then compiled for use with the Python application programming interface (API). The API allows for interfacing to the host vehicle, the placement of sensors, and the execution of user scripts. The CARLA build includes a breadth of python classes for the API and gives a solid basis on which to generate user libraries.

Metamoto

Metamoto provides another option for a simulation platform, which includes a Designer, Director, and Analyzer tool [16]. All three tools can run in either Windows 10 or a web-based client using cloud computing. Currently, Metamoto provides support for GLONASS, IMU, cameras, lidars, and radars; this includes support for customer sensor models using actual sensor firmware. In a sample workflow, the Designer tool would be used to instrument the vehicle, add dynamic assets to the static environment, or add driving maneuvers. Using the Director tool, the test can be scheduled, the environment can be customized by time of day, weather, and so forth, and other scenario variables can be customized and parameterized. The Analyzer tool can then be used to debug and replay data for postprocessing, benchmarking, and algorithm results analysis. Because Metamoto offers a pay-as-you-go service, all simulations run in the Metamoto cloud; simulations of higher complexity (several sensors or actors) require heavier computation and, in turn, a higher overall cost.

Cognata

Cognata Studio is another tool available for AV development, with a heavy focus on software-in-the-loop. Cognata currently supports GLONASS, IMU, cameras, lidars, and radars in simulation [17]. In a typical workflow using Cognata Studio, OpenDRIVE or satellite data would be imported into the tool to generate the backbone of the simulation. Using these standard data sources, Cognata's tool can use AI to rapidly generate digital twin static environments. On this generated static backbone, assets such as AI drivers, pedestrians, vehicle cameras, or vehicle lidar sensors can be added. In the simulation step, the vehicle can be tested and visualized in a high-definition environment based on custom pass/fail criteria and user-developed rules. Leveraging the company's partnership with NVIDIA, Cognata aims to target the Drive Constellation platform with their future work.

IPG Automotive CarMaker

CarMaker/TruckMaker is a simulation tool developed by IPG Automotive [18]. IPG has a dedicated Simulink interface that includes libraries for global navigation sensors, lidars

(realistic and ideal), and radars (realistic and ideal). The tool's integration with Simulink allows engineers and researchers to leverage existing vehicle models and hardware, including external toolchains such as dSPACE and National Instruments [19]. Map generation leverages HERE's ADAS RP as well as IPG Road, while simulation is handled with an IPG GUI and Simulink [20]. The typical workflow for IPG with an emphasis on the transition from the real to virtual environment using CarMaker will be thoroughly discussed in later sections.

Summary

Simulators commonly used for AV algorithm development and testing were summarized in the previous sections. The previous discussion of toolchains is not exhaustive, however, as the simulator landscape is rapidly changing and evolving. Referencing the Autonomous Vehicle Landscape report, the number of listed simulators grew from 19 to 29 between February and August 2019 [21], [22]. The aforementioned platforms were specifically chosen due to their popularity as well as their capability in sensor emulation. In addition, many of the previously mentioned tools follow a similar workflow. Development tools (e.g., MATLAB/Simulink) or perception stacks (e.g., RTMaps) were omitted, as they do not fall into the category of a true total simulation platform. Other simulation platforms, such as LGSVL Simulator, NVIDIA DRIVE Constellation, Pro-SiVIC, rFpro, and SCANeR were omitted due to our unfamiliarity with these toolchains; however, they may be included in future research.

Current state of data sets

With the rise in interest for autonomous capability in vehicles, the need to effectively test associated vision and control algorithms increases. The high complexity of mimicking human driving decision making poses a difficult task for developers and engineers. To address this need, test vehicles are being instrumented with an array of sensors such as cameras, lidars, and radars to collect vast amounts of data for public use. These data sets aim to bridge the gap between safe autonomous driving and effective controls development by teams of computer scientists, engineers, and researchers. In addition, many teams have developed associated software development kits (devkits) to help accelerate development in their respective data sets. These data sets are presented in Table 2 and discussed in the following sections. Note that specifications and configurations of sensors may vary drastically among different data sets, and readers should reference specific data providers in the next sections for more detailed information.

KITTI

The KITTI Vision Benchmark Suite was recorded with a Volkswagen Passat station wagon in Karlsruhe, Germany [23]. The data set includes recorded data from two gray-scale cameras, two color cameras, one lidar, and one real-time kinematic GPS and IMU unit each. In the raw data set provided, the data are separated into the categories "city," "residential," "road," "campus," "person," and "calibration." In each frame, the data set provides the raw data (cameras, lidar, and GPS/IMU) and object annotations in the form of 3D bounding boxes captured at 10 Hz. Gray-scale and color camera data are provided in.png format, lidar data are in binary format, and GPS/IMU and calibration data are provided as text files, while the 3D object labels are given in XML format. The KITTI data set also includes a raw data devkit [24] to parse the associated files using C++ or a MATLAB wrapper.

BDD

The BDD data set is a collection of more than 100,000 annotated crowdsourced videos. Most of the data are taken with a camera mounted on the dash of a multitude of vehicles, resulting in a diverse array of data. The data include multiple cities, weather and lighting conditions, times of day, and different scene types. Similar to the KITTI data set and many of the data sets discussed in this article, the BDD raw data are annotated with image tagging, road object bounding boxes, drivable areas, lane markings, and full-frame instance segmentation. In total, the data set includes bounding box annotation for 10 categories as well as if the object is occluded or truncated [2].

Waymo

The Waymo Open Data Set [4] is a subset of data taken from Waymo's instrumented fleet of self-driving vehicles. The data include lidar and camera data from 1,000 20-s segments captured at 10 Hz. The resultant data include labels for vehicles, pedestrians, cyclists, and signs; these include labels and bounding boxes on both lidar and camera data. In addition, the data set includes a diverse array of scenarios, weather conditions, and times of day. The provided code repository for development leverages Python and TensorFlow to parse and use the data set.

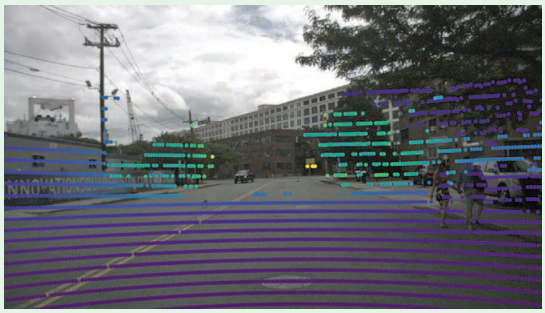
Aptiv nuScenes

The nuScenes data set is a large-scale data set developed by Aptiv Autonomous Mobility using two Renault Zoe vehicles [5]. Each vehicle is instrumented with GPS, one IMU unit, six cameras (see Figure 1), five radars, and one lidar (see Figure 2). A total of 1,000 driving scenes was collected in Boston and Singapore, with each scene cut to 20 s. A subset of 10 scenes is

Table 2. Publicly available data sets and their attributes.

	Size (GB)	GLONASS	IMU	Camera	Lidar	Radar	Location
KITTI	180	1	1	4	1	0	Karlsruhe
BDD	1,800	0	0	1	0	0	New York, Berkeley, and San Francisco
Waymo	1,000	0	0	5	5	0	25 U.S. cities
nuScenes	400	1	1	6	1	5	Boston and Singapore
Lyft	41.6	0	0	7	3	0	Phoenix

FIGURE 1. The captured camera output from six cameras mounted on the nuScenes development vehicle using the nuScenes devkit provided by libraries [5], [25].



(a)

(b)

FIGURE 2. An example of (a) lidar and (b) radar data streams overlaid on a front camera from the nuScenes data set [5], [25].

available in a mini data set, 850 for the training data set with full annotation, and 150 test scenes with no annotation. All of the objects in the nuScenes data set come with a semantic category as well as a 3D bounding box and attributes for frame. Twenty-three separate object categories are annotated, for a total of 1,166,187 annotations. A Python devkit and a full tutorial are also available (https://github.com/nutonomy/nuscenes-devkit), which allow the user to parse and manipulate the raw data as well as develop custom user libraries [25].

Lyft

Using the same format and devkit as the nuScenes data set, the Lyft Level 5 AV data set uses a fleet of Ford Fusion vehicles [3]. Each vehicle is instrumented with seven cameras and three lidar units, with slight variations in individual camera and lidar parameters depending on the vehicle deployed. The data set also includes human-labeled 3D bounding boxes of traffic agents and a semantic map. Because it leverages the nuScenes data set devkit, the same scripts and methods described in the nuScenes documentation can be utilized to parse and use the Lyft data. The training set is currently the only subset of data available for public use, but the testing and validation sets will be released at a later date.

Summary

A nonexhaustive list of publicly available data sets and their characteristics were discussed in the previous sections. These data sets were chosen due to their availability, general popularity in research, and overall usability. In addition, many of these data sets can be used for synthetic environment generation due to their variety of sensor data and data density. As such, some other data sets, e.g., the INTERACTION data set [6] and those referenced in the Autonomous Vehicle Landscape [22], were omitted.

Evaluation metrics

In the context of autonomous driving, the ability of an algorithm to detect and track objects in the driving environment is critical. A good replica of the real-world scene should yield very similar object detection and tracking results when the same algorithm is applied. Hence, to evaluate how well the virtual environment and its emulated sensors represent real-world scenarios, results of the same perception algorithm applied to detect and track objects in the real-world data set and the simulated environment are compared based on some commonly used object detection and tracking metrics summarized in this section.

Fundamentally, all object detection evaluation metrics use the following measures as their basis: true positive (TP) and false positive (FP), which represent the number of positive detections that correctly and incorrectly match the real-world objects, respectively. Similarly, true negative (TN) and false negative (FN) denote the number of negative detections (i.e., nondetections) that correctly and incorrectly match the real-world situation, respectively. To define how well the prediction matches the ground truth, metrics such as the *intersection over union* (*IoU*), defined as the ratio of the intersection of the detection and the ground-truth bounding boxes to the union of the two bounding boxes [26], [27] as well as the 2D center distance on the ground plane in meters [5] have been defined. These metrics are primarily used as a threshold for determining TP, FP, TN, and FN, which are then used to calculate the other metrics described in the following sections.

Object detection metrics

Detection results can be measured using basic metrics such as accuracy (A), recall (R), and precision (P). *Accuracy* is the ratio of correctly predicted results to the total observations, i.e., $A = (TP + TN)/(TP + TN + FP + FN)$. *Recall* denotes the ratio of the number of true positives to the number of real (actual) positives, i.e., $R = TP/(TP + FN)$. *Precision* typically denotes the ratio of the number of true positives to the number of predicted (detected) positives, i.e., $P = TP/(TP + FP)$; however, some defined have precision as $P = TP/(TP + FP + FN)$, which accounts for both the precision and recall metrics mentioned previously [28].

Average precision (AP) denotes the area below the precision-recall curve with precision monotonically decreasing [26], [29]. AP can be calculated based on a single IoU threshold (e.g., IoU = 0.5 [26] or IoU = 0.75 [27]); the higher the IoU, the stricter the metric is. AP can also be calculated based on multiple IoU thresholds, also known as the *mean AP* (*mAP*). For example, the COCO data set calculates its mAP based on an IoU = 0.5:0.05:0.95 (i.e., an IoU from 0.5 to 0.95 with a step-size of 0.05) [27].

With the recent advances in sensing and computing technologies, many detection algorithms can now detect and locate objects in the 3D environment [5]. As a result, evaluation metrics have also evolved from 2D to 3D. For example, the 2D center distance on the ground plane used by nuScenes [5] is only possible when the detection is in 3D. Also, instead of a 2D, area-based metric, IoU can now be a 3D, volume-based metric.

The CLassification of Events, Activities, and Relationships (CLEAR) workshop defined some additional metrics used for object detection and tracking [30], [31]. As shown in [1], the multiple object-detection precision of frame t is defined as the ratio of OverlapRatio to the number of mapped objects in frame t. It evaluates the alignment between the annotated and predicted bounding boxes. OverlapRatio in [2] is defined similarly to the concept of the sum of IoU, where $G_i^{(t)}$ denotes the ith ground-truth object in the tth frame and $D_i^{(t)}$ denotes the detected object for $G_i^{(t)}$ [30]:

$$\mathrm{MODP}(t) = \frac{\mathrm{OverlapRatio}}{N_{\mathrm{mapped}}^t}, \tag{1}$$

$$\mathrm{OverlapRatio} = \sum_{i=1}^{N_{\mathrm{mapped}}^t} \frac{|G_i^{(t)} \cap D_i^{(t)}|}{|G_i^{(t)} \cup D_i^{(t)}|}. \tag{2}$$

The Multiple Object-Detection Accuracy (MODA) metrics assesses the accuracy of the detection performance. As shown in [3], it compares false positives m_t and missed targets fp_t to the total number of ground-truth objects in frame t. Here, c_m and c_f are cost functions for the missed targets and false positives:

$$\mathrm{MODA}(t) = 1 - \frac{c_m(m_t) + c_f(fp_t)}{N_G^t}. \tag{3}$$

Object-tracking metrics

To evaluate the ability of an algorithm to accurately track the movement of objects, the multiple object-tracking precision (MOTP) and multiple object-tracking accuracy (MOTA) metrics were further defined by CLEAR [31]. As shown in [4] and [5], MOTP evaluates the ability of the algorithm to estimate precise object positions, and MOTA assesses the performance of the algorithm to correctly detect the number of objects and keep consistent trajectories. Here, c_t denotes the number of matches found for time t. For each of these matches, d_t^i denotes the distance between the object i and its corresponding detection. Finally, mme_t denotes the number of mismatch errors for frame t. MOTP and MOTA have been adopted by the Waymo Open Dataset as their tracking metrics [4]:

$$\mathrm{MOTP} = \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t}, \tag{4}$$

$$\mathrm{MOTA} = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}. \tag{5}$$

Summary

Several commonly used metrics in object-detection and tracking challenges were summarized in the preceding sections. The use of these metrics can be dependent on the algorithms being evaluated. For algorithms that detect and locate objects in images, object-detection metrics such as AP and mAP can be used. These metrics can be calculated in an area-based (2D) or volume-based (3D) manner depending on the sensors and data streams used. For algorithms that track objects through the sequence of multiple images, on the other hand, object-tracking metrics such as MOTP and MOTA can be used.

General workflow for synthetic environment creation and validation

The purpose of this article is to develop an overview of a framework and workflow that enables the emulation of a full suite of sensors and the generation of synthetic sensor data streams in virtual environments (via one of the discussed simulators), which replicates real-world scenarios using real data sets. As

mentioned in the following sections, this article discusses the approach in a step-by-step fashion.

Selection of a real-world data set for virtual environment replication

The question of which real-world data set to use for virtual environment and sensor feedback generation is important because it will help inform which simulators are acceptable for full scene generation or which data set is best suited for a user's available simulators. Choosing the correct data set for scene generation is threefold. First, data sets can provide limited sensor data or a full suite of sensors, including object-detection/perception sensors (e.g., cameras, radars, and lidars) and position sensors (GPS and IMU) [5]. Typically, it is best to find the most complete data set possible that covers the available sensor suite in the chosen simulation tool. Second, the chosen simulation tool must be able to appropriately model all of the chosen sensor streams for effective and more complete algorithm baselining. Finally, choosing a complete data set can save time for scene generation; for example, a data set with several cameras gives the scene creator a stronger basis upon which to build the virtual environment.

Generating the digital twin virtual environment

All of the simulation tools mentioned in this article have a primary interface for the development and testing of the virtual environment and the generation of the road network. With the GPS feedback or mapped location data provided from a chosen data set, a virtual digital twin environment can then be generated using a mapped database and software package, such as HERE's map database and development tools [32], or data from a 3D mapping service provider like 3D Mapping Solutions [33]. These services often have the ability to directly import into the simulator of choice, but the fidelity of the imported road network and the amount of adjustment and cleanup needed to achieve an acceptable scenario vary greatly.

Developing driving scenarios

After generating the scene, the user must place and define all of the simulation actors that range widely from cars and trucks to pedestrians, animals, and other moving objects. Replicating these objects as they are in the scene can be a very difficult step in the process, as these actors not only have to be present in the scene but also have a number of other variables and attributes that need to be correctly calibrated. These parameters include proper object type, size, color, surface properties, location, orientation, and velocity. Note that, although behavioral models such as car following, collision avoidance, and lane changing could be considered in this step, they are not necessary for this application because most real-world data sets provide detailed object locations frame by frame and thus can be manually replicated by the user.

Simulating full-suite synthetic sensors for collecting data in the virtual environment

All of the aforementioned toolchains include a full suite of synthetic sensors that can be deployed onto the vehicle in the virtual environment. Often the simulators have multiple levels of fidelity in the sensor models (e.g., ideal, statistical, probabilistic, physics based, and so on). These sensors, including lidars, radars, cameras, GLONASS, IMU, and ultrasonics, among others, can be used to collect synthetic data in the virtual environment based on the created road network, generated scene, and scenario built and deployed earlier. Figure 3 shows an example of synthetic versus real-world lidar data point clouds.

Shaping and restructuring synthetic sensor feeds

The synthetic sensor data from a simulator can then be restructured to match the configuration and delivery intervals of the sensor feeds in the real-world data sets. To complete this restructuring, custom code often needs to be written to unpack and repack synthetic data into the appropriate format.

Comparing real-world and synthetic data

Following the virtual environment generation, sensor deployment, data collection, and data restructure, real-world and synthetic data can be compared side by side using a variety of metrics to evaluate how the data collected by sensors in the real world compare to those collected virtually by their synthetic clones. Comparison can also be done by applying existing perception algorithms and controls on both the real-world and the synthetic data streams, and the results of these tests can be further evaluated.

Example workflow and use case

The detailed steps of a workflow (see Figure 4) that we used previously are described in the following sections as an example use case to better illustrate the workflow for using real-world data sets and virtual environments for perception algorithm evaluation.

Selection of scenes from the nuScenes data set

The nuScenes data set has the variety of sensors needed for a full AV, including data streams of lidars, radars, and cameras. It also includes a devkit that provides a suite of prebuilt Python functions to parse through scenes as well as a template for user functions and full tutorials on how to use and manipulate the data. Using these functions, one specific scene in the Boston Seaport was chosen for recreation. As shown in Figure 1 in the "Current State of Data Sets" section, this scene has a slew of different object classes available for recreation and validation in the virtual environment.

Generation of the environment using HERE and IPG

Using IPG CarMaker's HERE ADAS RP interface, it is possible to import road descriptions and basic geometry to lay the initial road backbone for the virtual simulation. Starting with the aforementioned example, the location of the Boston scene was found using the nusc.renderegoposesonmap() function in nuScenes, as displayed in Figure 5. Next, the Carmaker-ADAS RP interface is leveraged to export the Boston scene from HERE's map database to IPG's scenario editing tool, IPG Road. Note that the electronic horizon functionality of HERE's ADAS RP package was not utilized in this example; only the

map import feature was used. Then, in IPG Road, the initial backbone was manually populated by the user with additional assets that are parameterized to match the desired environment, as depicted in the selected nuScenes scene. In this case, the assets included pedestrians, a dog, parked vehicles, one moving truck, a stop sign, buildings, trees, and various objects surrounding the road and sidewalk. For the purpose of this example, the standard actor models provided by IPG were given user-defined locations, trajectories, and paths that were used to approximate the behavior of the actors in the scene.

Recreation of sensor behavior in the virtual environment

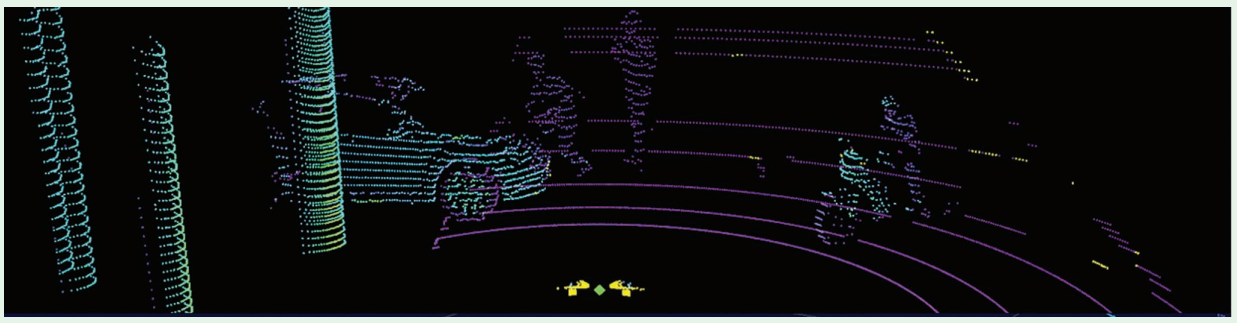
To begin sensor replication for proper testing, it is critical to replicate exact sensor placement on the vehicle as well as sensor parameters to best leverage the capability of the IPG toolchain. For this scene, the behavior of the front camera and the single lidar must be replicated and tested. Looking in the IPG environment, the sensors can be easily parameterized, as shown in Figure 6. For each real-world sensor and its synthetic replicate, conducting a visual one-to-one comparison to quantify the similarities/differences of the data streams is helpful, as presented in Figure 7.

Moreover, the response of existing perception algorithms such as sign, lane, and object-detection algorithms to virtual and real-world data streams can be evaluated. Referencing the nuScenes documentation, the lidar parameters are a 20-Hz capture frequency, 32 channels, a 360° horizontal field of view (FOV) with a +10 to –30° FOV, an 80–100-m range with ±2-cm accuracy, and up to 1.39 million points per second. The camera parameters are red green blue, a 12-Hz capture frequency, a 1/1.8-in CMOS sensor, 1,600 × 900 resolution, auto exposure, and JPEG compressed. The virtual and real-world data streams using these parameters are presented in Figure 7.

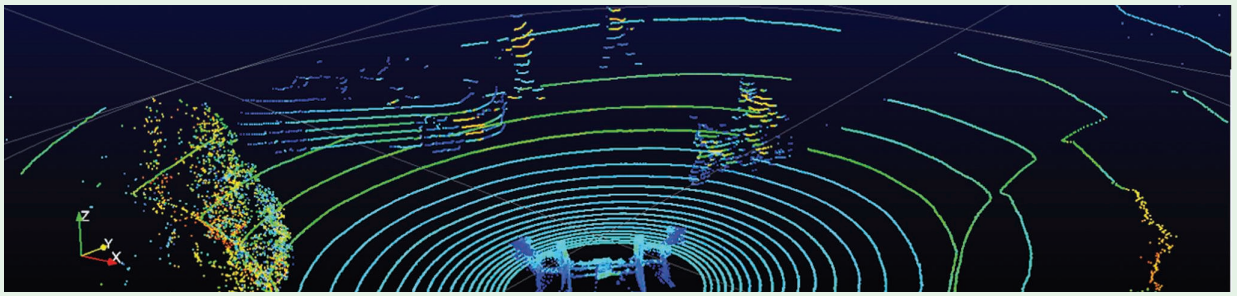
Testing of vision algorithms for synthetic baselining

For the purpose of comparing synthetic and real-sensor data streams, a selection of sensor(s) and perception algorithms must be made. In this example, we compared real data from the front camera module in the nuScenes data set to a synthetic front camera in the virtual environment of IPG CarMaker. With the sensor data streams selected, the perception algorithm was applied to detect and classify objects in these two data streams. In this example, You Only Look Once (YOLO) was chosen because of the availability of Darknet at https://www.pjreddie.com/darknet/yolo/, an open source framework for constructing a neural network architecture, as well as the final weights of the YOLOv3-416 model trained on Microsoft's COCO data set [27], [34]. In addition to the pretrained weights, the Darknet framework includes integration with NVIDIA CUDA for faster calculation on GPU and OpenCV to facilitate the input of various image formats and video stream to the model [35].

An open source variant of Darknet available on GitHub enabled the ability to provide a video to the model and receive an output video stream with the objects detected, classified, and localized using a bounding box [36]. By making slight alterations to the source code, the framework was able to output the object class, bounding box location, and classification confidence in a file format suitable for further data analysis. Leveraging these features, a comparison of the perception results from the real video stream with the synthetic video



(a)



(b)

FIGURE 3. (a) A lidar emulation in the IPG CarMaker environment using a ray-tracing algorithm. (b) A real-time data stream capture from a Velodyne HDL-32 lidar on an Oak Ridge National Laboratory (ORNL) vehicle.

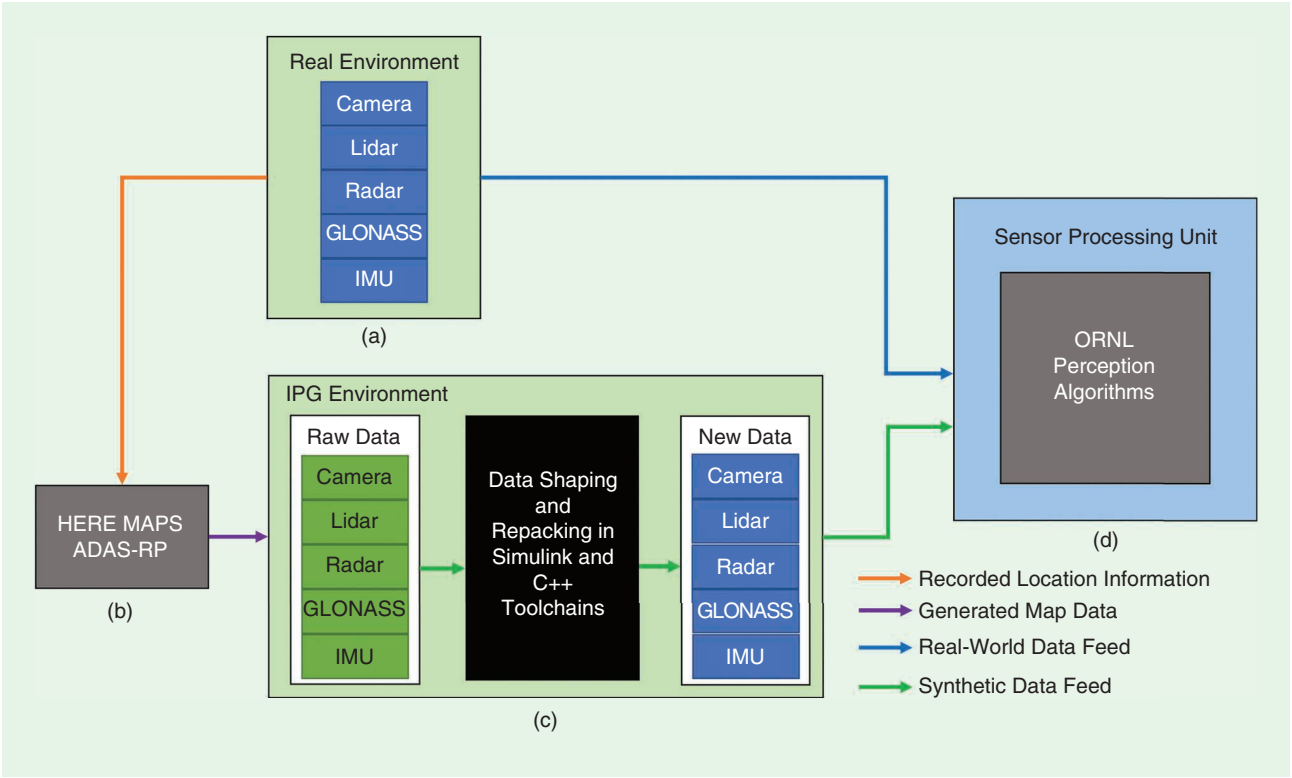


FIGURE 4. A diagram of ORNL's workflow using real-world data sets and virtual environments to compare the resulting sensor and visual data streams. (a) A real-world data set, (b) map and environment generation, (c) scenario creation and vehicle parameterization, and (d) the testing of vision- or sensor-detection algorithms using a real-time sensor processing unit.

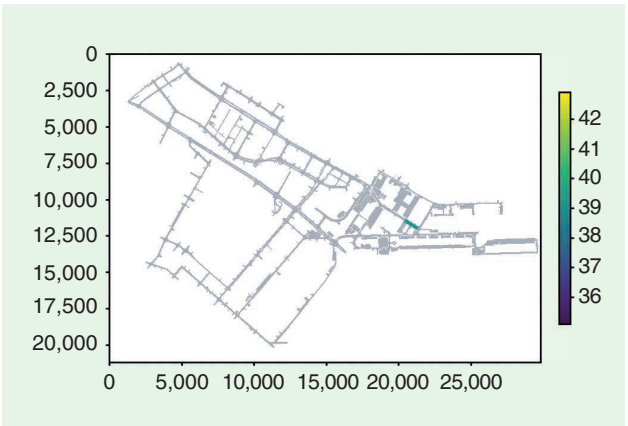


FIGURE 5. The rendering of a selected scene ego vehicle position depicting the number of ego poses within a 100-m area of the Boston Seaport [25].

stream generated in CarMaker was possible (see Figure 8). Metrics such as the AP and mAP with different 2D area-based IoU thresholds can be used to evaluate the detection results.

Conclusions

This article served as a tutorial for developing a framework to evaluate simulators and their sensor emulation in the virtual environment. With a review of commonly used data sets and simulators for testing and developing autonomous driving applications, this article outlined the process of selecting real-world data sets to generate a digital twin in the virtual environment, described options for simulation tools to emulate a full suite of AV sensor data streams, and discussed object-detection and tracking metrics that can be used to evaluate how AV-perception algorithms respond to virtual versus real-world sensor data streams. In the future, the validation framework could be expanded in the following directions:

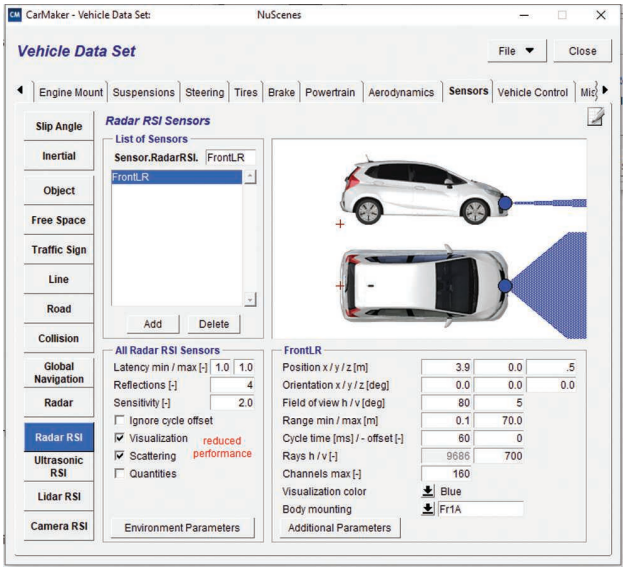
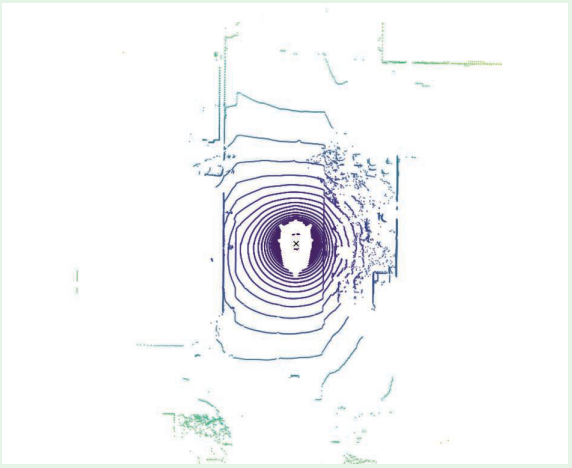
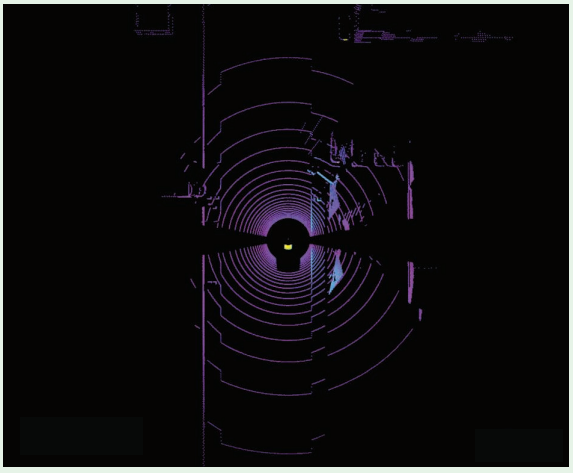


FIGURE 6. IPG CarMaker's interface includes extensive parameterization for a variety of sensors.



(a)



(b)

FIGURE 7. A comparison of (a) lidar point clouds from the nuScenes data set [5], [25] versus (b) a virtual replication in IPG CarMaker.

- *Quantitative analysis of perception algorithm results*: Although our team used visual inspection to analyze the synthetic and real-world data streams from lidar and camera units, a quantitative evaluation has not been performed. Going forward, we will examine the quantitative metrics further and determine how they clarify the validation of the simulation.
- *HIL applications*: Our team will evaluate virtual and real-world data streams ingested by Oak Ridge National Laboratory's (ORNL's) sensor processing unit and bench-top vehicle controller. The data will then be used as real-time inputs to the AV-perception algorithms that affect vehicle control.
- *Other perception and control algorithms*: This article touched on the use of object-detection algorithms in both virtual and real environments. In addition, the evaluation could potentially expand to other algorithms such as object segmentation, drivable areas, traffic signal-state detection, traffic sign detection and recognition, lane detection, path planning, dynamic behaviors of automated vehicles, and so forth.

(a)

(b)

FIGURE 8. A comparison of detection on the (a) nuScenes data set [5] and (b) the IPG CarMaker virtual replications.

Acknowledgments

This article was authored by the University of Tennessee-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy (DOE). The U.S. government retains it, and the publisher, by accepting the article for publication, acknowledges that the U.S. government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this article, or allows others to do so, for U.S. government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Authors

Dean Deter (deterdd@ornl.gov) is the principal engineer for the Vehicle Systems Integration and Connected and Automated Vehicle Environment Labs at Oak Ridge National Laboratory (ORNL), Oak Ridge, Tennessee, USA. Prior to this role, he led hardware-in-the-loop (HIL) and vehicle electrification focused efforts at ORNL focused primarily on the medium- and heavy-duty sectors. He specializes in advanced powertrain and vehicle R&D utilizing HIL practices and methodologies. He is experienced in vehicle/component modeling and simulation, HIL testing methods, virtual vehicle environments, and sensor data emulation. He is a Member of IEEE.

Chieh (Ross) Wang (cwang@ornl.gov) received his Ph.D. degree in civil and environmental engineering (with a focus in transportation systems engineering) from the Georgia Institute of Technology in 2017. Currently, he is an R&D staff member

at Oak Ridge National Laboratory, Oak Ridge, Tennessee, USA. He has extensive experience in transportation data analytics, including data collection, harvesting, integration, mining, and visualization. He has also been developing smart city solutions (e.g., optimized signal timing control and vehicle speed advisory) and computer vision algorithms for connected and automated vehicles. His research focuses on transportation data analytics and smart cities. He is a Member of IEEE.

Adian Cook (cookas@ornl.gov) received his B.S. degree in physics from the University of the South in 2009 and his M.S. degree in mechanical engineering from the University of Tennessee, Knoxville (UTK) in 2016. He is currently a vehicle systems R&D engineer at Oak Ridge National Laboratory (ORNL), Oak Ridge, Tennessee, USA. In 2019, he returned to UTK to pursue his Ph.D. degree in mechanical engineering with a specific emphasis on machine learning and artificial intelligence. Prior to joining ORNL in 2017, he competed in an advanced vehicle competition at UTK and worked for a small mechatronics company in Ann Arbor, Michigan. His research interests include embedded software, algorithm development, vehicle integration, and hardware-in-the-loop. He is a Member of IEEE.

Nolan Perry (perryn@ornl.gov) received his B.S. degree in mechanical engineering in 2015 and his M.S degree in mechanical engineering with a focus on robotics and controls in 2016, both from the University of Tennessee, Knoxville (UTK). He is currently a vehicle systems R&D engineer at Oak Ridge National Laboratory (ORNL), Oak Ridge, Tennessee, USA. Prior to joining ORNL in 2019, he competed in an advanced vehicle competition at UTK and worked for three years in vehicle software development and calibration at an automotive industry original equipment manufacturer in Detroit, Michigan. His research interests include software development, vehicle perception and control, and robotics. He is a Member of IEEE.

References

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. 2012 IEEE Conf. Computer Vision and Pattern Recognition*, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

[2] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, BDD100K: A diverse driving video database with scalable annotation tooling. 2018. [Online]. Available: arXiv:1805.04687

[3] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low et al., "Lyft level 5 AV dataset 2019," Lyft, San Francisco, CA, 2019. [Online]. Available: https://level5.lyft.com/dataset

[4] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou et al., Scalability in perception for autonomous driving: An open dataset benchmark. 2019. [Online]. Available: https:arXiv:abs/1912.04838

[5] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan et al., nuScenes: A multimodal dataset for autonomous driving. 2019. [Online]. Available: https:arXiv:abs/1903.11027

[6] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof et al., INTERACTION dataset: An INTERnational, adversarial and cooperative moTION dataset in interactive driving scenarios with semantic maps. 2019. [Online]. Available: arXiv:1910.03088

[7] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition*, pp. 3234–3243. doi: 10.1109/CVPR.2016.352.

[8] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition*, pp. 4340–4349. doi: 10.1109/CVPR.2016.470.

[9] ASAM, "ASAM OpenDRIVE," Hoehenkirchen, Germany, 2020. [Online]. Available: https://www.asam.net/standards/detail/opendrive/

[10] ASAM, "ASAM OpenCRG," Hoehenkirchen, Germany, 2020. [Online]. Available: https://www.asam.net/standards/detail/opencrg/

[11] ASAM, "ASAM OpenSCENARIO," Hoehenkirchen, Germany, 2020. [Online]. Available: https://www.asam.net/standards/detail/openscenario/

[12] MSC Software Corp., "Virtual test drive," Newport Beach, CA, 2020. [Online]. Available: https://www mscsoftware.com/product/virtual-test-drive

[13] dSPACE Inc., "Automotive simulation models," Wixom, MI, 2020. [Online]. Available: https://www.dspace.com/en/inc/home/products/sw/automotive_simulation_models.cfm

[14] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, CARLA: An open urban driving simulator, 2017. [Online]. Available: arXiv:1711.03938

[15] VectorZero, "Roadrunner," Carlsbad, CA, 2019. [Online]. Available: https://www.vectorzero.io/roadrunner

[16] Metamoto, "Simulation as a service," Redwood City, CA, 2020. [Online]. Available: https://www.metamoto.com

[17] Cognata, Rehovot, Israel, 2020. [Online]. Available: https://www.cognata.com/

[18] IPG Automotive, Karlsruhe, Germany, 2020. [Online]. Available: https://ipg-automotive.com

[19] "A powerful package. IPG Automotive makes CarMaker/HIL compatible with National Instruments hardware," IPG Automotive, Karlsruhe, Germany, 2015. [Online]. Available: https://press.ipg-automotive.com/press-release/article/a-powerful-package

[20] "Product extensions: Benefit from even greater performance," IPG Automotive, Karlsruhe, Germany, 2020. [Online]. Available: https://ipg-automotive.com/products-services/simulation-software/product-extensions/

[21] M. Amblard, "The autonomous vehicles landscape," Orsay Consulting, Palo Alto, CA, 2019. [Online]. Available: https://s3.amazonaws.com/spoke-profiles-prod-assets/uploads/9ba4516ff61c3652e15fdb88d52a23ca9b3c2596/original/Autonomous_Vehicles_Landscape_Feb_2019.pdf

[22] M. Amblard, "The autonomous vehicles landscape," Orsay Consulting, Palo Alto, CA, 2019. [Online]. Available: https://s3.amazonaws.com/spoke-profiles-prod-assets/uploads/ea6818b43b5faa46ae481fe71cd4cacfcae6cf61/original/Autonomous_Vehicles_Landscape_Aug_2019.pdf

[23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013. doi: 10.1177/0278364913491297.

[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Raw data," Karlsruhe Institute of Technology, Germany, 2020. [Online]. Available: http://www.cvlibs.net/datasets/kitti/raw_data.php

[25] "nuScenes devkit," GitHub, San Francisco, CA, 2020. [Online]. Available: https://github.com/nutonomy/nuscenes-devkit

[26] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge 2011 (VOC2011) development kit." [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2011/devkit_doc.pdf

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Computer Vision–ECCV 2014 (Lecture Notes Computer Science)*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer-Verlag, 2014, pp. 740–755. doi: 10.1007/978-3-319-10602-1_48.

[28] Lyft, "Lyft 3D object detection for autonomous vehicles," San Francisco, CA, 2019. [Online]. Available: https://www.kaggle.com/c/3d-object-detection-for-autonomous-vehicles

[29] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, pp. 98–136, Jan. 2015. doi: 10.1007/s11263-014-0733-5.

[30] R. Stiefelhagen, K. Bernardin, R. Bowers, J. Garofolo, D. Mostefa, and P. Soundararajan, "The clear 2006 evaluation," in *Multimodal Technologies for Perception of Humans (Lecture Notes in Computer Science)*, R. Stiefelhagen and J. Garofolo, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 1–44. doi: 10.1007/978-3-540-69568-4_1.

[31] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," *J. Image Video Process.*, vol. 2008, pp. 1–10, Jan. 2008. doi: 10.1155/2008/246309.

[32] HERE, "HERE maps," Amsterdam, The Netherlands, 2020. [Online]. Available: https://www.here.com/products/mapping/map-data

[33] 3D Mapping Solutions GmbH, "Automotive applications," Holzkirchen, Germany, 2019. [Online]. Available: https://www.3d-mapping.de/en/partners/automotive-applications

[34] J. Redmon and A. Farhadi, Yolov3: An incremental improvement. 2018. [Online]. Available: https:arXiv:abs/1804.02767

[35] J. Redmon, "Darknet: Open source neural networks in C," 2013–2016. [Online]. Available: http://pjreddie.com/darknet

[36] AlexeyAB, "GitHub Yolo-v3 and Yolo-v2 for Windows and Linux," 2019. [Online]. Available: https://www.github.com/AlexeyAB/darknet

SP

Arman Kheirati Roonizi

An Efficient Algorithm for Maneuvering Target Tracking

Maneuvering target tracking is an important technology in engineering applications [1]–[3]. The traditional methodologies for designing it can be divided into two categories: model-based and model-free algorithms. Almost all tracking algorithms are model based. The main idea behind model-based tracking algorithms is to choose a representation that fits the actual state trajectories of the target movement and then to estimate the state based on the noisy observations recorded by sensors. The Kalman filter and its extensions are the most popular methods to estimate the state of a system. However, the stability and convergence rate of these algorithms depend directly on the accurate initial state estimation, unknown parameters, and covariance matrices of the process and measurement noise.

This article introduces a simple and effective algorithm for maneuvering target tracking. An easy trick is used to modify the representation such that each state variable is described by an independent difference model [e.g., the autoregressive (AR) and the AR-moving average (ARMA) models]. Then, the target state variable is estimated using a regularized least-squares optimization algorithm. While the proposed approach is simple, it is inherently stable, as it can be implemented by forward-filtering the observation signal with a finite-impulse response (FIR) filter and backward-filtering the result with the same FIR filter; it depends only on a single parameter whose optimal value is obtained using an L-curve function, and its computational complexity is linear.

Digital Object Identifier 10.1109/MSP.2020.3029386
Date of current version: 24 December 2020

Introduction

Almost all maneuvering target tracking models assume that the target motion and its observation can be represented by the following state space model [4]:

$$\begin{cases} x_{k+1} = f_k(x_k, u_k) + w_k \\ y_k = h_k(x_k) + \eta_k \end{cases}, \tag{1}$$

where x_k is the state vector containing the terms of interest for the system, e.g., the target position (p_k), velocity (v_k), and acceleration (a_k); u_k is the control input (e.g., the braking force and the throttle setting); y_k is the observation; and w_k and η_k are process and measurement noise, respectively. The control input is typically deterministic and unknown. Therefore, it is naturally modeled with an unknown, deterministic process from measurement data during tracking.

Another approach, which is more popular than deterministic modeling, is to represent the control input as a random process. This method includes three groups [4]. In the first, the control input is modeled as white noise, such as the constant-velocity (CV) model and the constant-acceleration (CA) model. In the second, the control input is represented as a Markov model with a time correlation. This group includes the Singer model, which is the best known and most commonly used facsimile that covers the CV, the CA, and many other models [4]. In the last group, a semi-Markov jump process is employed to model the control input.

The linear counterpart of the target motion is described as [4]

$$\begin{cases} x_{k+1} = F_k x_k + G_k w_k \\ y_k = H_k x_k + \eta_k \end{cases}, \tag{2}$$

where F_k is the state transition matrix, which applies the effect of each state parameter at time k on the system state at time $k+1$; G_k is the process noise gain matrix; and H_k is a matrix that relates the state variables to the measurements. We desire to estimate the true target state or the motion parameters (the position, velocity, and acceleration) from the observations. As an example of an application, consider a simple tracking problem in which a truck moves along a straight line. The driver may apply an acceleration input or a braking input to the system, which can be considered as a function of an applied force (f_k) and the mass of the truck (m). According to Newton's second law of motion, the control input is $u_k = f_k/m$. The relationship between the position and the velocity of the truck during the time period

 1053-5888/21©2021IEEE

T_s (the elapsed period between time epochs $k-1$ and k) and the driver's input is given by the following equation:

$$\begin{cases} p_{k+1} = p_k + v_k T_s + \dfrac{f_k T_s^2}{2m} \\ v_{k+1} = v_k + \dfrac{f_k T_s}{m} \end{cases},$$

which can be written in the following linear form:

$$x_{k+1} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \dfrac{T_s^2}{2} \\ T_s \end{bmatrix} \frac{f_k}{m}.$$

In this case, the state vector x_k can be considered as $x_k = [x_{1,k}, x_{2,k}]^T = [p_k, v_k]^T$. Finally, we can equip the truck with a GPS unit that can provide an estimate of the target position according to the model

$$y_k = x_{1,k} + \eta_k.$$

In the CV and CA models, the control input is considered to be white noise. Another approach is to model the control input (the target acceleration) as a Markov model.

The Singer acceleration model

In his seminal paper [5], Singer proposed a zero-mean, first-order stationary Markov process for modeling the target acceleration a_k with the following autocorrelation:

$$R_a(\tau) = \mathbb{E}\{a_k a_{k+\tau}\} = \sigma^2 e^{-\alpha|\tau|},$$

where σ^2 is the variance of the target acceleration and α is the reciprocal of the maneuver (acceleration) time constant. It is worth noting that as the maneuver time constant τ increases, the acceleration becomes a white noise jerk model, and the Singer model reduces to the CV model. In cases where the maneuver time constant decreases, the acceleration becomes white noise, and the Singer model reduces to the CA model. In other instances, the Singer model corresponds to motion between these models.

The target acceleration is described by the following linear time-invariant model:

$$a_{k+1} = -\beta a_k + w_k,$$

where w_k is zero-mean white Gaussian noise with variance $\sigma^2(1-\beta^2)$, $\beta = e^{-\alpha T_s}$ and T_s is the sampling period. The state–space representation of the discrete-time Singer model is of the form [5]

$$x_{k+1} = \begin{bmatrix} 1 & T_s & \dfrac{\alpha T_s - 1 + \beta}{\alpha^2} \\ 0 & 1 & \dfrac{1-\beta}{\alpha} \\ 0 & 0 & \beta \end{bmatrix} x_k + \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} w_k, \tag{3}$$

where the exact value of m_i is a function of α and T_s and can be found in [5, eq. (19)]: see the boxed equation at the bottom of the page. The aim is to estimate the unobserved underlying state vector x_k from observation y_k.

Motivation

In fact, for linear dynamical systems subject to Gaussian noise with known model parameters, the Kalman filter is the optimal estimator in the minimum mean-square error (MMSE) sense [6]. However, the model parameters are usually unknown in real applications. Therefore, the stability and the convergence rate of this algorithm depend directly on the accurate initial state estimation, model parameters, and covariance matrices of the process and measurement noise. Moreover, the algorithm dependently estimates the state variables (the target position, target velocity, and target acceleration). This means that, in some applications that require the estimation of only one parameter, the estimation algorithm may require unnecessary parameters to be approximated, as well. In particular, the target velocity and target acceleration can be found from the target position.

Moreover, the observation for the velocity and acceleration are commonly not recorded by sensors. Tracking sensors usually measure the target position. Therefore, there is a need for a new estimation algorithm that provides better accuracy, with the capability of separable variable estimation. In the derivation that follows, we consider the Singer model, as it is the basic and most popular model-based tracking algorithm. We propose a simple and effective algorithm for estimating the target state of the representation.

The proposed approach

In this article, we propose an efficient algorithm for maneuvering target tracking. The basic idea of the proposed algorithm is to modify the model such that each state variable is described by an independent dynamic representation. The desired state variable is then estimated using a regularized least-squares method for only one unknown. While the proposed approach is simple, it depends only on a single parameter whose optimal value is obtained using an L-curve function. In the derivation that follows, we propose a solution to the target tracking problem in discrete time, where a simple trick is used to convert the problem into an independent difference model. In "Maneuvering Target Tracking in Continuous Time," we tackle the same problem for continuous-time systems by following an approach parallel to that used in the discrete-time case.

Let us define the forward shift operator z^i, which represents the operation for shifting a signal to the left by i samples (i.e., $z^i x_k = x_{k+i}$). Using it, (3) can be expressed as

$$z^1 \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ x_{3,k} \end{bmatrix} = \begin{bmatrix} 1 & T_s & \dfrac{\alpha T_s - 1 + \beta}{\alpha^2} \\ 0 & 1 & \dfrac{1-\beta}{\alpha} \\ 0 & 0 & \beta \end{bmatrix} \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ x_{3,k} \end{bmatrix} + \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} w_k. \tag{4}$$

$$m_1 = \sqrt{\frac{1}{2\alpha^5}\left[1 - \exp(-2\alpha T_s) + 2\alpha T_s + \frac{2\alpha^3 T_s^3}{3} - 2\alpha^2 T_s^2 - 4\alpha T_s \exp(-\alpha T_s)\right]},$$
$$m_2 = \sqrt{\frac{1}{2\alpha^3}\left[4\exp(-\alpha T_s) - 3 - \exp(-2\alpha T_s) + 2\alpha T_s\right]},$$
$$m_3 = \sqrt{\frac{1}{2\alpha}\left[1 - \exp(-2\alpha T_s)\right]}.$$

Maneuvering Target Tracking in Continuous Time

We tackle the problem of tracking maneuvering targets in continuous time and show that the proposed approach can be applied to both continuous and discrete-time models. Let us consider the continuous-time Singer model. Its discrete-time representation, (3), is often obtained from [5]

$$\frac{d}{dt}x(t)=\begin{bmatrix}0&1&0\\0&0&1\\0&0&-\alpha\end{bmatrix}x(t)+\begin{bmatrix}0\\0\\1\end{bmatrix}w(t).$$

Defining the differentiation operator s^i, which represents d^i/dt^i, i.e., $s^i x(t)=d^i/dt^i x(t)$, the continuous-time Singer model is expressed as

$$sx(t)=\begin{bmatrix}0&1&0\\0&0&1\\0&0&-\alpha\end{bmatrix}x(t)+\begin{bmatrix}0\\0\\1\end{bmatrix}w(t).$$

Applying the proposed trick, we find

$$x(t)=\begin{bmatrix}s&-1&0\\0&s&-1\\0&0&s+\alpha\end{bmatrix}^{-1}\begin{bmatrix}0\\0\\1\end{bmatrix}w(t).$$

After some simplifications, we find

$$\begin{cases}(s^3+\alpha s^2)x_1(t)=w(t)\\(s^2+\alpha s)x_2(t)=w(t)\\(s+\alpha)x_3(t)=w(t)\end{cases}.$$

Substituting the differential operator s^i with d^i/dt^i, we obtain

$$\begin{cases}\left(\frac{d^3}{dt^3}+\alpha\frac{d^2}{dt^2}\right)x_1(t)=w(t)\\\left(\frac{d^2}{dt^2}+\alpha\frac{d}{dt}\right)x_2(t)=w(t)\\\left(\frac{d}{dt}+\alpha\right)x_3(t)=w(t)\end{cases}.$$

Finally, in the continuous-time domain, the target position can be modeled by

$$\begin{cases}\frac{d^3}{dt^3}x_1(t)+\alpha\frac{d^2}{dt^2}x_1(t)=w(t)\\y(t)=x_1(t)+\eta(t)\end{cases},$$

which can be expressed as

$$\begin{cases}\theta(t)*x_1(t)=w(t)\\y(t)=x_1(t)+\eta(t)\end{cases},$$

where $\theta(t)=(d^3/dt^3)\delta(t)+\alpha(d^2/dt^2)\delta(t)$, $\delta(t)$ is the delta function and $*$ denotes the convolution operator. The desired target position can be estimated using the following regularized least-squares optimization algorithm:

$$x_1(t)=\underset{x_1(t)}{\operatorname{argmin}}\|y(t)-x_1(t)\|^2+\lambda\|\theta(t)*x_1(t)\|^2.$$

Setting the derivative of the preceding equation with respect to $x_1(t)$ to zero, and after some simplifications, we find [13]

$$x_1(t)=[\delta(t)+\lambda\theta(-t)*\theta(t)]^{\odot}*y(t),$$

where $^{\odot}$ denotes the deconvolution operator.

In the following, for simplifying the notation, we drop the "1" from "z^1" and simply refer to "z^1" as "z." After some simplifications, (4) can be expressed in the following form:

$$\begin{bmatrix}x_{1,k}\\x_{2,k}\\x_{3,k}\end{bmatrix}=\begin{bmatrix}z-1&-T_s&-\frac{\alpha T_s-1+\beta}{\alpha^2}\\0&z-1&-\frac{1-\beta}{\alpha}\\0&0&z-\beta\end{bmatrix}^{-1}\times\begin{bmatrix}m_1\\m_2\\m_3\end{bmatrix}w_k. \quad (5)$$

Equation (5) is equal to the equation at the bottom of the page.

Since the right-hand side depends only on w_k, each state $x_{i,k}$ depends only on w_k but not the other states: $x_{j,k},\ j\neq i$. Therefore, each state variable can be described by an independent (AR/ARMA) model:

$$2(z-1)^2(z-\beta)x_{1,k}=\left(m_1(z^2-(\beta+1)z+\beta)+m_2T_s(z-\beta)+m_3(T_s\frac{1-\beta}{\alpha^2}+(z-1)\frac{\alpha T_s-1+\beta}{\alpha^2})\right)w_k, \quad (6a)$$

$$(z-1)(z-\beta)x_{2,k}=\left(m_2(z-\beta)+m_3\frac{1-\beta}{\alpha}\right)w_k, \quad (6b)$$

$$(z-\beta)x_{3,k}=m_3w_k. \quad (6c)$$

According to (6a), the target position can be modeled by the following ARMA model:

$$\begin{cases}x_{1,k+3}+\phi_1x_{1,k+2}+\phi_2x_{1,k+1}+\phi_3x_{1,k}=\zeta_1w_{k+2}+\zeta_2w_{k+1}+\zeta_3w_k\\y_k=x_{1,k}+\eta_k\end{cases}, \quad (7)$$

where

$$\phi_1=-(\beta+2),\quad \phi_3=-\beta,$$
$$\zeta_2=-m_1(\beta+1)+m_2T_s+m_3\frac{\alpha T_s-1+\beta}{\alpha^2},$$
$$\phi_2=1+2\beta,\quad \zeta_1=m_1,$$
$$\zeta_3=m_1\beta-m_2T_s\beta+m_3\left(T_s\frac{1-\beta}{\alpha^2}-\frac{\alpha T_s-1+\beta}{\alpha^2}\right).$$

$$(z-1)^2(z-\beta)\begin{bmatrix}x_{1,k}\\x_{2,k}\\x_{3,k}\end{bmatrix}=\begin{bmatrix}(z-1)(z-\beta)&T_s(z-\beta)&T_s\frac{1-\beta}{\alpha^2}+(z-1)\frac{\alpha T_s-1+\beta}{\alpha^2}\\0&(z-1)(z-\beta)&(z-1)\frac{1-\beta}{\alpha}\\0&0&(z-1)^2\end{bmatrix}\begin{bmatrix}m_1\\m_2\\m_3\end{bmatrix}w_k.$$

Equation (7) can be expressed as [7]

$$\begin{cases} \dfrac{\Upsilon(z)}{\Gamma(z)} x_{1,k} = w_k \\ y_k = x_{1,k} + \eta_k \end{cases},$$

where

$$\Upsilon(z) = z^3 + \phi_1 z^2 + \phi_2 z + \phi_3,$$
$$\Gamma(z) = \zeta_1 z^2 + \zeta_2 z + \zeta_3.$$

Inspired by [7], the following regularized least-squares optimization algorithm is proposed to estimate the desired target position:

$$\hat{x}_{1,k} = \underset{x_{1,k}}{\operatorname{argmin}} \sum_{n=1}^{N} (y_n - x_{1,n})^2 + \lambda \sum_{n=1}^{N} \left(\frac{\Upsilon(z)}{\Gamma(z)} x_{1,n} \right)^2, \quad (8)$$

where N is the number of samples. In (8), the first term in the minimization is the difference between the measured location and the unknown actual location of the target, and the second Lagrangian term denotes the process noise power (i.e., λ is used to tradeoff between the tracking model error and the MMSE).

Equation (8) can be expressed in the following matrix notation:

$$\hat{x}_1 = \underset{x_1}{\operatorname{argmin}} \| \Gamma(y - x_1) \|^2 + \lambda \| \Upsilon x_1 \|^2,$$

where

$$\Upsilon = \begin{pmatrix} \phi_3 & \phi_2 & \phi_1 & 1 & 0 & \dots & 0 \\ 0 & \phi_3 & \phi_2 & \phi_1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \phi_3 & \phi_2 & \phi_1 & 1 \end{pmatrix} \text{ and}$$

$$\Gamma = \begin{pmatrix} \zeta_3 & \zeta_2 & \zeta_1 & 0 & \dots & 0 \\ 0 & \zeta_3 & \zeta_2 & \zeta_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \\ 0 & \dots & 0 & \zeta_3 & \zeta_2 & \zeta_1 \end{pmatrix}.$$

The optimal solution is [7]

$$\hat{x}_1 = (\Gamma^T \Gamma + \lambda \Upsilon^T \Upsilon)^{-1} \Gamma^T \Gamma y. \quad (9)$$

When T_s is sufficiently small so that $T_s \ll 1$, the values of ζ_i are very small, and the target position can be simply approximated by an AR model, instead:

$$\begin{cases} x_{1,k+3} + \phi_1 x_{1,k+2} + \phi_2 x_{1,k+1} + \phi_3 x_{1,k} = \varphi_k \\ y_k = x_{1,k} + \eta_k \end{cases},$$

where $\operatorname{var}\{\varphi_k\} = \sigma_\varphi^2 = (\zeta_1^2 + \zeta_2^2 + \zeta_3^2) \times \sigma_w^2$. In this case, the optimal solution is (the matrix Γ is replaced by identity matrix) [7]

$$\hat{x}_1 = (I + \lambda \Upsilon^T \Upsilon)^{-1} y, \quad (10)$$

where I is the identity matrix.

The proposed method is noncausal, as it uses all the available measurements [see (8)] to estimate the desired state. In the following, we call the method a *smoothing filter* or a *smoother*. The computational complexity of the proposed smoother is $O(n)$ [7]. See Figure 1 for an example, where the gray crosses denote the noisy data obtained by sampling the red, solid curve (the true target position). The observed data are the result of contaminating the true data with random noises that have signal-to-noise ratios (SNRs) of SNR = 20, SNR = 10, and SNR = 0 in Figure 1(a)–(c), respectively.

In this example, the Singer model parameters are $\alpha = 1.6$, $T_s = 0.01$, and $\sigma_w^2 = 1$. However, in real applications, the exact values of the model parameters are not available. Therefore, a nominal model is estimated and employed rather than the true model. The wrong estimated parameters were used to construct the nominal model. The nominal model curve is plotted with a yellow dash–dot curve. The proposed method was then harnessed to track the target position from the nominal model and the noisy observations. The only unknown parameter is λ. The optimal value of λ is related to the process noise power, σ_w^2, and the observation noise power, σ_v^2 (i.e., $\lambda = \sigma_v^2 / \sigma_w^2$) [7, eq. (30)]. In other words, the hyperparameter λ plays the role of balancing the optimization between the two noise terms.

However, in real problems, the noise variances are unknown. We propose to use an L- or U-curve to estimate the optimization parameter. To this end, the following bicriterion problem is employed [7]:

$$\text{minimize}(\text{w.r.t } R_+^2) \; (\| \Gamma(y - x_1) \|, \| \Upsilon x_1 \|).$$

Inspired by [7], the optimal regularization factor, λ, is obtained by plotting $\| \Gamma(y - x_1) \|$ versus $\| \Upsilon x_1 \|$ as λ varies across $(0, \infty)$. The λ corresponding to the point at which $\| \Gamma(y - x_1) \| + \| \Upsilon x_1 \|$ is minimized, i.e., the corner of the curve, is chosen as the optimal regularization factor. The L-curves corresponding to the noisy position are given in Figure 1(d). The L-curve has a clear knee near $\| \Upsilon x_1 \| \approx 480$ for SNR = 20, a clear knee near $\| \Upsilon x_1 \| \approx 1{,}850$ for SNR = 10, and a clear knee near $\| \Upsilon x_1 \| \approx 5{,}700$ for SNR = 0. The optimal value of λ is 1.58, 6.31, and 15.85 for SNR = 20, SNR = 10, and SNR = 0, respectively.

In Figure 1(e), we plot the sum of the norm of the residual term and the norm of the regularized term as λ varies across $(10^{-5}, 10^5)$. The λ corresponding to the minimum of this curve, which is equal to λ, and that corresponds to the corner of the L-curve is chosen as the optimal parameter. The optimal parameter was used to implement the proposed method. To verify the performance of the proposed smoothing filter, we employed it for tracking the target position in the previous examples. The results of the target tracking using the proposed smoothing filter (smoother) are illustrated with a black, dashed curve in Figure 1(a)–(c), demonstrating that the proposed smoothing filter is accurate enough for tracking the target position.

A causal filter

The matrix $\mathrm{M} = (\Gamma^T \Gamma + \lambda \Upsilon^T \Upsilon)^{-1} \Gamma^T \Gamma$ is a symmetric positive definite real matrix and has a unique Cholesky decomposition of the form $\mathrm{M} = \mathrm{L}\mathrm{L}^T = \mathrm{L}^T \mathrm{L}$, where L is a lower triangular matrix with real and positive diagonal entries. Therefore, (9) can be rewritten as $\hat{x}_1 = \mathrm{L}^T \mathrm{L} y$. Substituting L^T with JLJ, we obtain $\hat{x} = \mathrm{JLJL}y$, where J is the exchange matrix (a "row-reversed/column-reversed" version of the identity matrix). Therefore, the proposed smoothing filter can be implemented as a combination of forward-filtering the input signal y with a FIR filter $(x^f = \mathrm{L}y)$ and backward-filtering with the same FIR filter

$(\hat{x} = \mathrm{JLJ}x^f)$. The filter's output at time k, $x^f_{1,k}$ is obtained as

$$x^f_{1,k} = \sum_{i=0}^{k-1} \mathrm{L}(k, k-i)\, y_{k-i}.$$

The proposed FIR filter is inherently stable. We employed it to track the target position of the previous examples. The results are illustrated with the blue curve in Figure 1(a)–(c). The proposed filter uses only the current and past measurements to estimate the desired position, while the proposed smoother takes all the measurements (past, current, and future) into account. That is why the proposed smoother has better performance than the proposed filter. In the following section, simulations for verifying the performance of the proposed methods in comparison with the Kalman filter/smoother and the particle filter are given.

Experimental results

To verify the estimation performance of the proposed method in the target tracking application, we apply the approach to simulated data, which permits us to directly quantify the tracking error (no gold standard is available for tracking moving objects in real data). The performance of the proposed algorithm is compared with the Kalman filter/smoother and the particle filter using the noise-to-signal ratio (NSR) given by [7]

$$\mathrm{NSR} = \sqrt{\frac{\sum_k (x_{1,k} - \hat{x}_{1,k})^2}{\sum_k x^2_{1,k}}},$$

where $x_{1,k}$ and $\hat{x}_{1,k}$ are the true target position and the estimated target position, respectively. We consider four kinds of motion for objects (e.g., trucks, ships, and trains) in 2D [8], [9]:

1) uniform
2) uniform acceleration

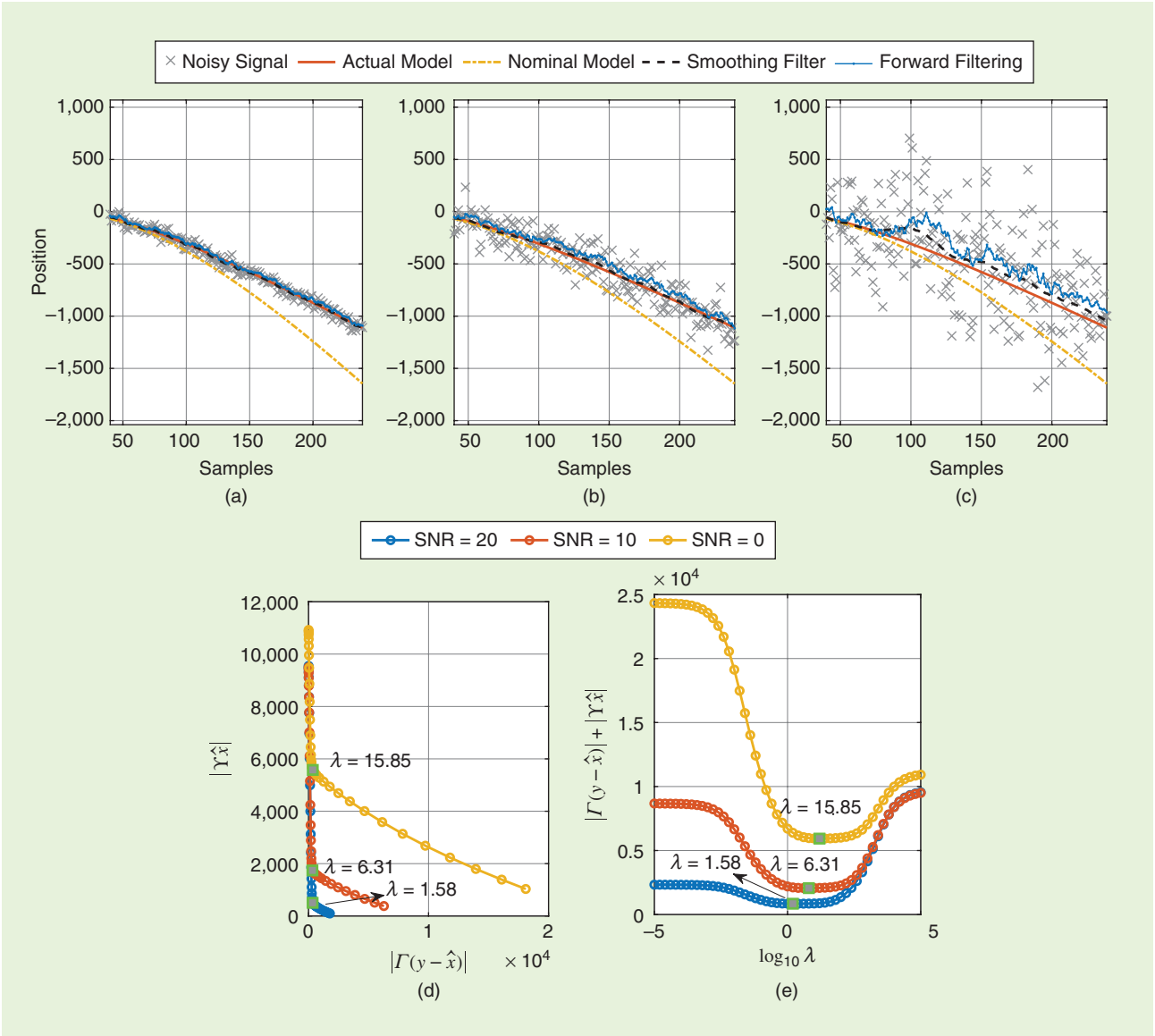


FIGURE 1. The tracked target position using the proposed smoothing procedure and the optimal-tradeoff L-curves. (a) SNR = 20. (b) SNR = 10. (c) SNR = 0. (d) The L-curves. (e) The sum of the norm of the residual and the regularized term.

3) varying acceleration
4) coordinated turn (CT).

In this experiment, we set T_s to 1.

As an example, for the path of a uniform moving target with initial states $x_0 = [0\ \text{m}\ 1\ \text{m/s}\ 0\ \text{m}\ 1\ \text{m/s}]^T$, process noise covariance $Q = 1$, and uniform acceleration motion with initial states $x_0 = [0\ \text{m}\ 1\ \text{m/s}\ 0.1\ \text{m/s}^2\ 0\ \text{m}\ 1\ \text{m/s}\ 0.1\ \text{m/s}^2]^T$, the process noise covariance $Q = 10$ is respectively plotted in Figure 2(a) and (b), including the actual value (the red, solid curve) and the noisy value (the gray, dotted curve), the results of the Kalman smoother (the green dash–dot curve), the particle filter with 1,000 particles (the purple dot curve), and the proposed smoother (the blue, dashed curve). We did not include the results of the Kalman filter and the proposed filter to avoid an unreadably complex diagram. Figure 2(c) shows the path of a CT model with a turn rate of $\Omega = 0.05$, with initial states $x_0 = [1{,}000\ \text{m}\ 200\ \text{m/s}\ 1{,}000\ \text{m}\ 200\ \text{m/s}]^T$, and the results of the position tracking using the preceding methods.

Compared to other methods, the tracked results using the proposed method are closer to the true path. To compare the tracking performance of each method, Monte Carlo simulations with 100 replications were performed. We produced signals varying the power of η_k. The SNR was modulated from 0 to 40 dB. In this experiment, different process noises with various powers $(0.001 \le Q \le 10)$ are considered. We calculate the NSRs, considering those process noises, and report the average value for each SNR. Figure 3(a) presents the results of the target tracking using the preceding procedures for uniform motion tracking. The results show that the NSR for the proposed smoother is smaller than the Kalman filter/smoother and the particle filter. Figure 3(b) and (c), respectively, shows the results of target tracking algorithms for varying acceleration motion and the CT motion model. It is seen that, for these two models, the NSR for the proposed smoother is smaller than it is for the other processes. It is notable that this system is linear, and the particle filter is particularly suited for nonlinear, non-Gaussian systems.

Application to power line interference and baseline wander cancellation

This section focuses on a specific application of the proposed method for power line interference (PLI) cancellation and baseline wander (BW) removal in biosignal measurement systems. PLI and BW are always a problem in biosignal measurement systems, in particular, electrocardiograms (ECGs).

PLI cancellation

The PLI (which may be considered as a sinusoid with a frequency of 50 or 60 Hz, depending on the geographic location) can be modeled by the following dynamical representation [10]:

$$x_{k+1} = \begin{bmatrix} 2\cos(\omega_0) & -1 \\ 1 & 0 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} w_k,$$

where $x_k = [x_k, x_{k-1}]^T$, $\omega_0 = 2\pi f_0/f_s$, and f_0 and f_s are the PLI and the sampling frequency. The biosignals contaminated by PLI can be represented as $y_k = x_k + \eta_k$, where x_k is the PLI and η_k represents other undesired signals/noise and assumed to be a zero-mean random term [10].

The proposed method can be used to track and remove the PLI from ECG recordings. To validate the proposed method, real ECG signals were taken from the Atrial Fibrillation Termination Challenge Database (http://archive.physionet.org/physiobank/database/aftdb/), which is a part of the PhysioNet project described in [11]. The data set contains 80 records that were originally provided for the 2004 PhysioNet/Computers in Cardiology Challenge [12]. Each record, extracted from a two-lead, 128-Hz sampling frequency Holter ECG recording, is 1 min in length. We

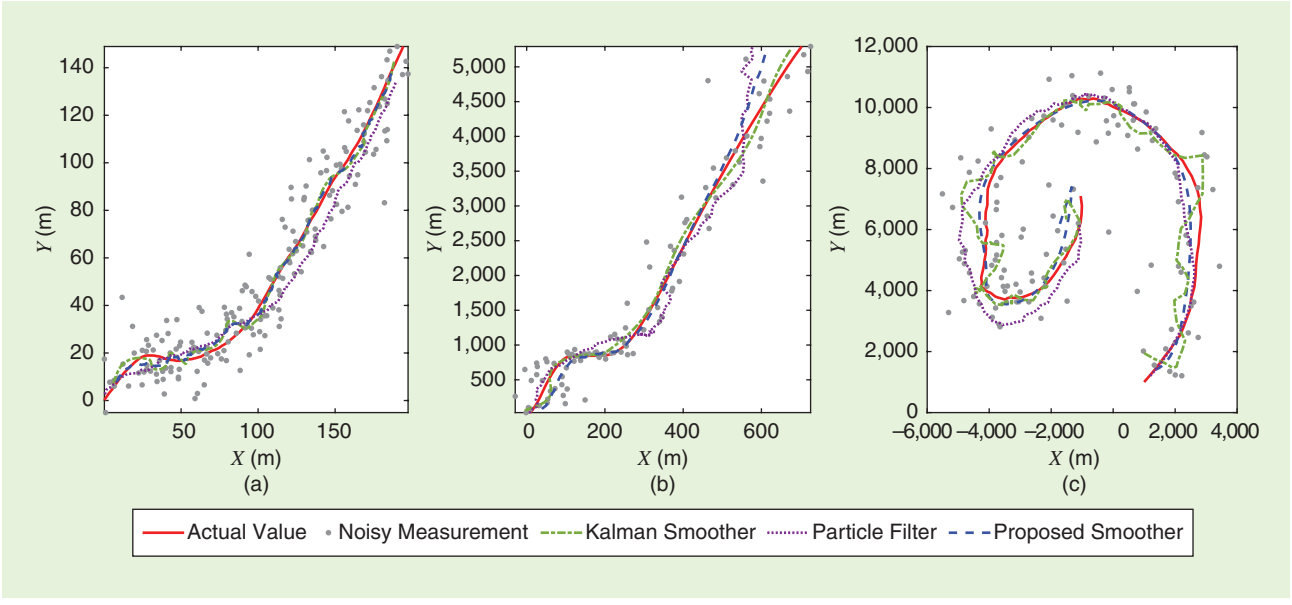


FIGURE 2. An example of (a) uniform motion ($Q = 1$), (b) uniform acceleration motion ($Q = 10$), and (c) the CT model ($\Omega = 0.05$; $Q = 10$).

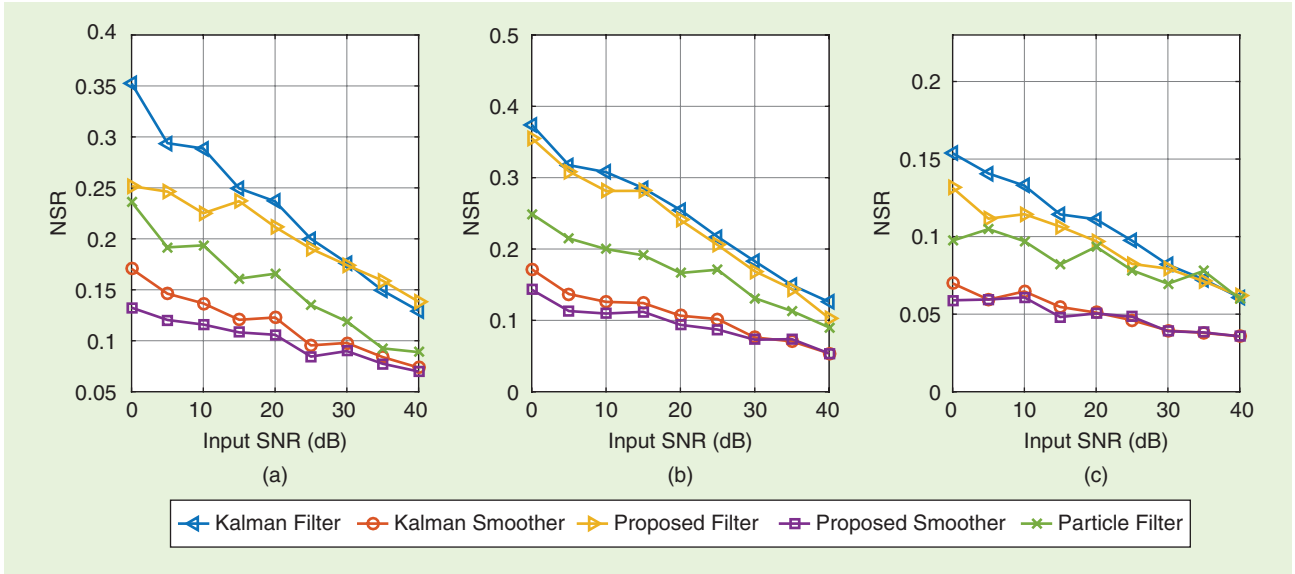


FIGURE 3. The results of (a) uniform motion, (b) uniform acceleration motion, and CT (c) motion tracking using Monte Carlo simulations.

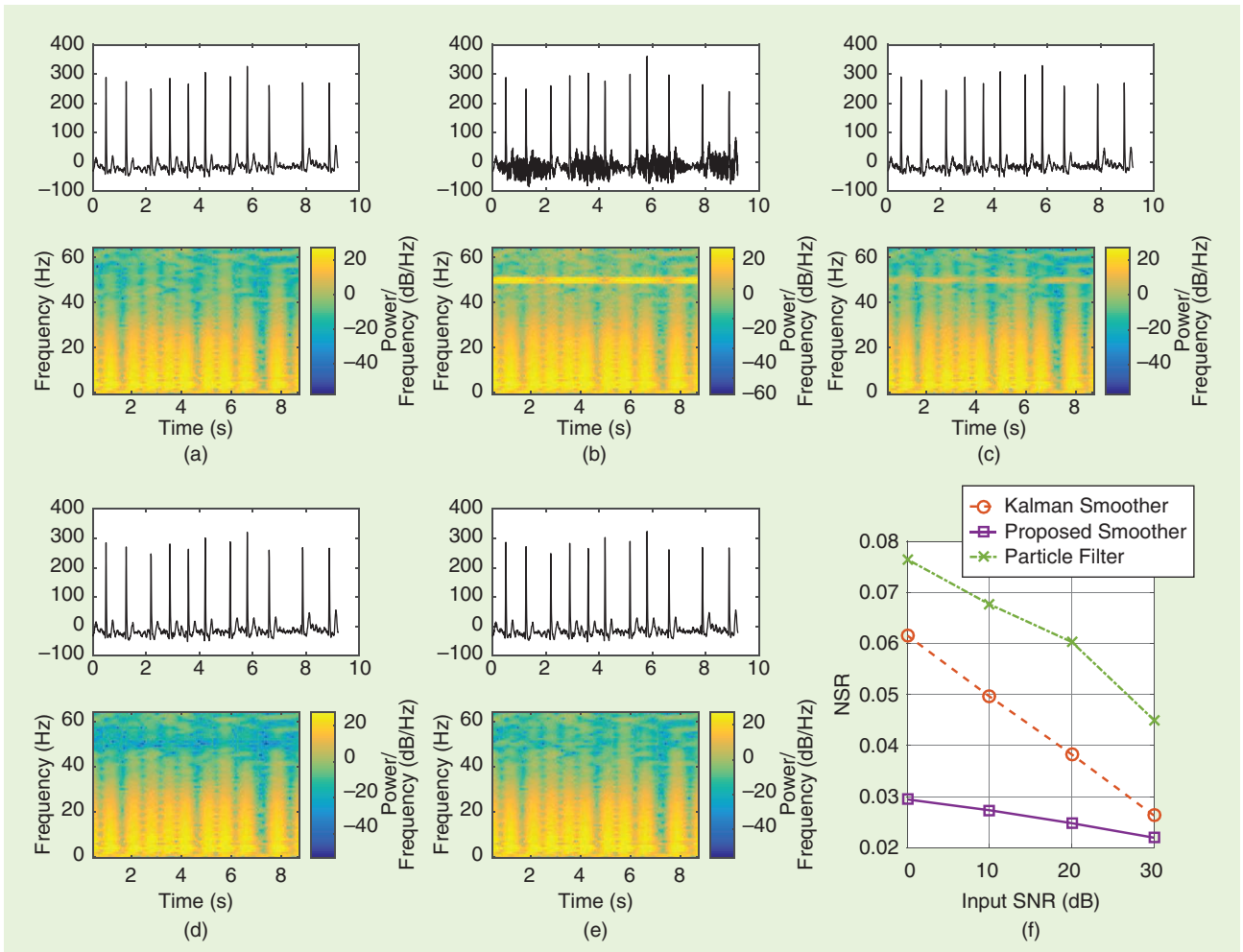


FIGURE 4. ECG PLI denoising for record a01m from the 2004 PhysioNet/Computers in Cardiology Challenge. (a) The ECG record. (b) The noisy ECG. (c) The denoised ECG provided by the particle filter with 1,000 particles. (d) The Kalman smoother. (e) The proposed smoother. (f) The mean values of the NSR for ECG PLI cancellation as a function of the power of PLI corrupting the input signal.

generated synthetic PLI (which was modulated with a 0.2-Hz sinusoid to represent respiratory-coupled changes in the PLI amplitude) and added it to clean ECG signals. The proposed trick was employed to modify the PLI model. Then, the PLI was estimated using the proposed method and removed from the ECG signals.

As a preliminary example, we plot a specific case (record a01m from the 2004 PhysioNet/Computers in Cardiology Challenge [12]) at the top of Figure 4(a). At the bottom of that figure, we show the spectral representation. At the top of Figure 4(b), we plot the same ECG but corrupted with PLI ($f_0 = 50$ Hz). The results of the PLI cancellation using the particle filter with 1,000 particles, the Kalman smoother, and the proposed smoother as well as their spectral representation are reported in Figure 4(c)–(e). The denoised ECG obtained with the proposed smoother is visually closer to the true ECG, compared to the denoised ECG derived through the particle filter and the Kalman smoother. The spectral representation of the signals is also presented and visually confirms the superiority of the proposed smoother (see the spectrograms in Figure 4).

For quantitatively evaluating the performance of the methods, we generated synthetic PLI with a variable SNR in the range of 0–30 dB and added it to ECG signals. The methods were then harnessed to estimate and remove the PLI from the ECG signals. The results of the tracking procedures are displayed in Figure 4(f). The values of the NSR show that the proposed method outperformed the particle filter and the Kalman smoother in PLI cancellation.

BW removal

BW is low-frequency additive noise affecting ECG signals. It can be represented using the CV and CA models, and its state can be estimated with the proposed method. As an example, a segment of the record A00001m from the 2017 PhysioNet/Challenge Training Set database [11] is shown in Figure 5(a). This record is affected by baseline drift and difficult to analyze. To remove the baseline drift, the BW was represented by the CA model and tracked using the Kalman smoother and the proposed smoother. After BW removal using the Kalman smoother and the proposed smoother, respectively, the same record is displayed in Figure 5(b) and (c). The BW has been successfully tracked and removed with the proposed smoother. The Kalman smoother leads to the distortion of P waves, while the proposed smoother does not, as highlighted using the ellipse in Figure 5(b) and (c).

Summary

This article proposed a simple and effective algorithm for maneuvering target tracking. The method is derived from a simple mathematical trick. The basic idea is to modify the model such that each state variable is described by a separate AR or ARMA representation. Then, the desired target state variable is estimated using a regularized least-squares optimization algorithm. Compared to other traditional methods (e.g., the Kalman filter and the particle filter), the proposed approach is inherently stable; it depends only on a single parameter whose optimal value is obtained using the L-curve function. It is simple, and its computational complexity is linear. Through comprehensive simulations and two real applications, it has been shown that the proposed method

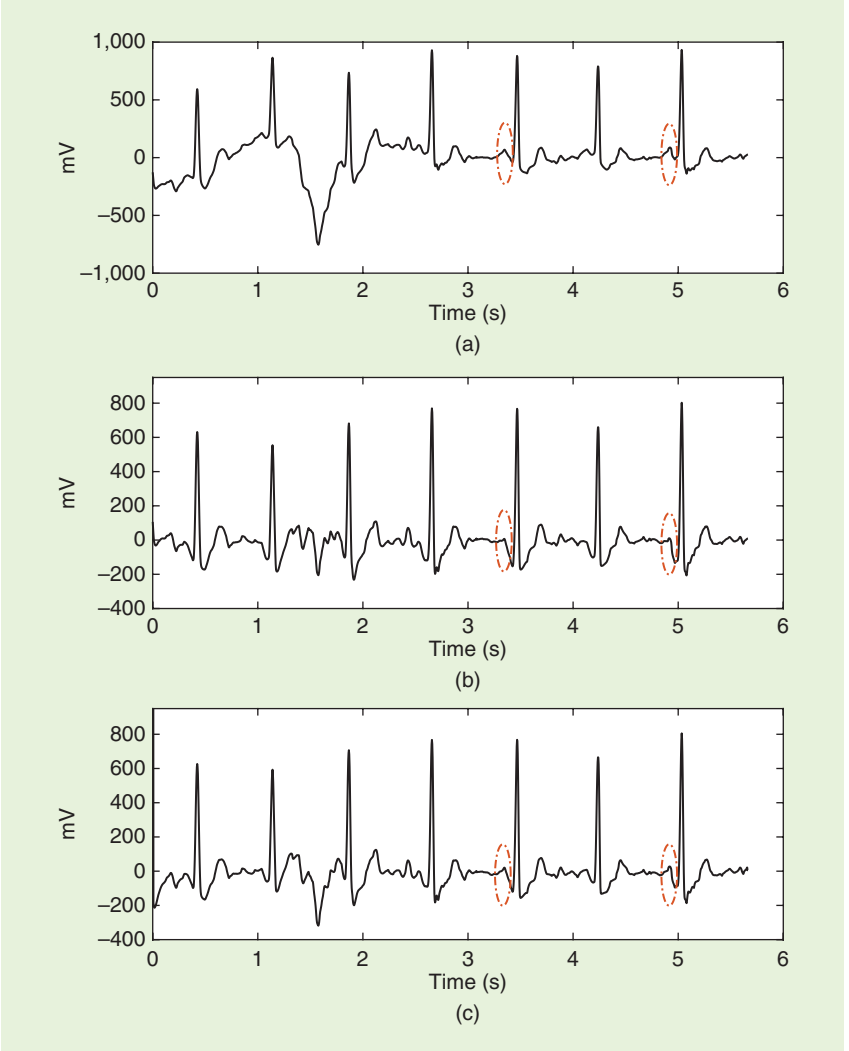


FIGURE 5. An ECG from real data: a segment of record challenge/2017/training/A00001m from PhysioNet and the ECG after the BW removal using a Kalman smoother and the proposed smoother. (a) ECG record A00001m. (b) The BW removal using a Kalman smoother. (c) The BW removal using the proposed smoother.

provides better estimation accuracy than the Kalman and particle filters for linear systems. Although the provided analyses and the proposed methodologies were given for linear systems, there are some improvements that can still be made. The extension of the proposed method to nonlinear systems can be considered as one of the future additions to this article.

Acknowledgments

We would like to thank Prof. Roberto Togneri, *IEEE Signal Processing Magazine* (*SPM*) area editor of columns and forums; Prof. Yuan-Hao Huang, *SPM* associate editor of columns and forums; and the anonymous reviewers for providing valuable suggestions to improve the manuscript.

Author

Arman Kheirati Roonizi (ebad.kheirati.roonizi@gmail.com) received his B.A.Sc. degree in computer engineering and his M.A.Sc. degree in bioelectrical engineering, both from Shiraz University, in 2006 and 2011, respectively, and his Ph.D. degree in computer science from the University of Milan, Italy, in 2017. He is currently an assistant professor in the Department of Computer Science, Fasa University, Fasa, Iran. During 2017–2018, he held a postdoctoral research position with GIPSA-lab, Grenoble, France. His research interests include signal modeling/processing, cardiac modeling and simulation, multimodal signal processing, and functional data analysis.

References

[1] S. Challa, M. Morelande, D. Mušicki, and R. Evans, *Fundamentals of Object Tracking*. Cambridge, U.K.: Cambridge Univ. Press, 2011. [Online]. Available: https://books.google.com/books?id=DOKJGnNh9HgC

[2] K. Saho, "Kalman filter for moving object tracking: performance analysis and filter design," in *Kalman Filters-Theory Advanced Applications*, G. L. Serra, Eds. 2018, pp. 233–252.

[3] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 425–437, 2002. doi: 10.1109/78.978396.

[4] X. Rong Li and V. P. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1333–1364, 2003. doi: 10.1109/TAES.2003.1261132.

[5] R. A. Singer, "Estimating optimal tracking filter performance for manned maneuvering targets," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-6, no. 4, pp. 473–483, 1970. doi: 10.1109/TAES.1970.310128.

[6] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Upper Saddle River, NJ: Prentice Hall, 1993.

[7] A. Kheirati Roonizi, "A new approach to ARMAX signals smoothing: Application to variable-Q ARMA filter design," *IEEE Trans. Signal Process.*, vol. 67, no. 17, pp. 4535–4544, 2019. doi: 10.1109/TSP.2019.2928986.

[8] B. Han, G. Xin, J. Xin, and L. Fan, "A study on maneuvering obstacle motion state estimation for intelligent vehicle using adaptive Kalman filter based on current statistical model," *Math. Probl. Eng.*, vol. 2015, pp. 1–14, Aug. 2015. doi: 10.1155/2015/515787.

[9] M. Eltoukhy, M. O. Ahmad, and M. Swamy, "An adaptive turn rate estimation for tracking a maneuvering target," *IEEE Access*, vol. 8, pp. 94,176–94,189, May 2020. doi: 10.1109/ACCESS.2020.2995672.

[10] R. Sameni, "A linear Kalman notch filter for power-line interference cancellation," in *Proc. 16th CSI Int. Symp. Artif. Intell. Signal Process. (AISP 2012)*, vol. AES-9, pp. 604–610. doi: 10.1109/AISP.2012.6313817.

[11] A. L. Goldberger et al., "Physiobank, Physiotoolkit, and Physionet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2003. doi: 10.1161/01.CIR.101.23.e215.

[12] G. Moody, "Spontaneous termination of atrial fibrillation: A challenge from physionet and computers in cardiology 2004," *Comput. Cardiol.*, vol. 31, pp. 101–104, July 2004. doi: 10.1109/CIC.2004.1442881.

[13] A. Kheirati Roonizi and C. Jutten, "Forward-backward filtering and penalized least-squares optimization: A unified framework," *Signal Process.*, vol. 178, p. 107,796, Jan. 2021. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168420303406. doi: 10.1016/j.sigpro.2020.107796.

Özlem Tuğfe Demir and Emil Björnson

The Bussgang Decomposition of Nonlinear Systems

Basic theory and MIMO extensions

Many of the systems in various signal processing applications are nonlinear due to, for example, hardware impairments, such as nonlinear amplifiers and finite-resolution quantization. The Bussgang decomposition is a popular tool used when analyzing the performance of systems that involve such nonlinear components. In a nutshell, the decomposition provides an exact probabilistic relationship between the output and the input of a nonlinearity: the output is equal to a scaled version of the input plus uncorrelated distortion. The decomposition can be used to compute either exact performance results or lower bounds, where the uncorrelated distortion is treated as independent noise. This lecture note explains the basic theory, provides key examples, extends the theory to complex-valued vector signals, and clarifies some potential misconceptions.

Relevance

The origin of the decomposition is a technical report by Julian J. Bussgang in 1952 [1]. Interestingly, the decomposition is not explicitly stated in his report but, rather, was a consequence of his results. In fact, it is mainly nontrivial extensions of his results that are utilized in current research; for example, applications to complex-valued multiple-input, multiple-output (MIMO) systems are popular in the communication community. There is no standard reference that presents and proves those extended results, and it can be hard to differentiate between which results are exact and which are mere approximations. This lecture note fills these gaps.

Digital Object Identifier 10.1109/MSP.2020.3025538
Date of current version: 24 December 2020

Prerequisites

This lecture note requires basic knowledge of random variables, linear algebra, signals and systems, and estimation theory.

Problem statement

Let us consider two jointly Gaussian continuous-time stationary random processes $f(t)$ and $g(t)$. One of the processes, say $f(t)$, undergoes a nonlinear memoryless distortion represented by the function $U(\cdot)$. The resulting non-Gaussian random process is $F(t) = U(f(t))$. The problem at hand is to obtain the cross correlation between samples of the distorted process $F(t)$ and $g(t)$ as well as $f(t)$ in a tractable form. The aim is to analyze the impact of nonlinearities commonly encountered in signal processing applications. Two further problems are to extend the results to MIMO systems and to generalize to the case of a non-Gaussian input process $f(t)$.

Solution: Bussgang decomposition

In the original paper [1], Bussgang computed the cross correlation of the two random variables obtained by sampling $F(t)$ and $g(t)$ at specific time instances. Let $x = f(t_1) \in \mathbb{R}$ and $y = g(t_2) \in \mathbb{R}$ denote the zero-mean Gaussian random variables obtained by sampling at time t_1 and t_2, respectively. Moreover, let $z = F(t_1) = U(x) \in \mathbb{R}$ be the sampled output of the nonlinear distortion function. We then have the following main result from [1, Sec. III].

Theorem 1: The Bussgang theorem

The cross correlation of $z = U(x)$ and y is

$$C_{zy} = \mathbb{E}\{U(x)y\} = \underbrace{\frac{\mathbb{E}\{U(x)x\}}{\mathbb{E}\{x^2\}}}_{\triangleq B}\mathbb{E}\{xy\} = BC_{xy}, \tag{1}$$

where B is called the Bussgang gain, and $C_{xy} \triangleq \mathbb{E}\{xy\}$ is the cross correlation of x and y.

The Bussgang theorem shows that the cross correlation between two Gaussian signals is the same before and after one of them has passed through a nonlinear function, except for a scaling factor B. The value of B depends on the choice of $U(\cdot)$ but the theorem holds for any function. After the original paper [1], the Bussgang theorem was recognized as a special case of the Price theorem [2], which provides an alternative computation method for the Bussgang gain that we will return to later. In the remainder of this lecture note, we focus on the

sampled random variables $x = f(t_1)$, $y = g(t_2)$, and $z = F(t_1)$. However, it is important to remember that the discrete-time random variables are obtained from underlying continuous-time random processes, that the distortion is memoryless, and that x and z are samples taken at the same time.

A consequence of Theorem 1 is that the output signal can be decomposed as

$$z = U(x) = Bx + \eta, \tag{2}$$

where η is a zero-mean random variable that is uncorrelated to both x and y. This can be shown by multiplying both sides of (2) with x or y, taking the expectation, and using Theorem 1. Hence, the classical relationship in (2) has given rise to the name *Bussgang decomposition*. The Bussgang decomposition in its elementary form shows that the output contains the useful part Bx and the distortion part η. In other words, the output of a nonlinear function is equal to a scaled version of the input plus the uncorrelated distortion η. Note that η and x are not independent. Since $\eta = U(x) - Bx$ is a deterministic function of x, the distortion term is non-Gaussian distributed and statistically dependent on x. Even if the Bussgang decomposition is named after Bussgang, the decomposition is not explicitly stated in [1].

The Bussgang decomposition can be viewed as the linear minimum-mean squared error (MMSE) estimate of z given x, with η being the estimation error. Hence, the decomposition holds even if x is not Gaussian distributed, but the Bussgang decomposition also guarantees that the distortion signal η is uncorrelated to any other jointly Gaussian random variable y, which does not hold when considering the non-Gaussian distributed x and y.

Bussgang decomposition for complex random variables

The Bussgang theorem was extended to the complex case in [3]. We present this result and then provide a direct proof from [4] that uses the linear MMSE estimator. For notational convenience, in the remainder of this lecture note, we use $C_x \triangleq \mathbb{E}\{|x|^2\}$ to denote the power of a signal x and we use $C_{xy} \triangleq \mathbb{E}\{xy^*\}$ to denote the cross correlation between x and y.

Theorem 2: The complex Bussgang theorem

Consider the jointly circularly symmetric complex Gaussian random variables $x \in \mathbb{C}$ and $y \in \mathbb{C}$. Let $z = U(x) \in \mathbb{C}$ be the output of a deterministic function. The cross correlations $C_{zy} \triangleq \mathbb{E}\{zy^*\}$ and $C_{xy} \triangleq \mathbb{E}\{xy^*\}$ are then related as

$$C_{zy} = \mathbb{E}\{U(x)y^*\} = \underbrace{\frac{\mathbb{E}\{U(x)x^*\}}{\mathbb{E}\{|x|^2\}}}_{\triangleq B = C_{zx}/C_x}\mathbb{E}\{xy^*\} = BC_{xy}. \tag{3}$$

Proof

We begin by decomposing y into two parts:

$$y = \frac{\mathbb{E}\{yx^*\}}{\mathbb{E}\{|x|^2\}}x + \underbrace{\left(y - \frac{\mathbb{E}\{yx^*\}}{\mathbb{E}\{|x|^2\}}x\right)}_{\triangleq \epsilon}, \tag{4}$$

which is equivalent to computing an MMSE estimate of y given x, with ϵ representing the estimation error. Hence, it follows that the second part, ϵ, in (4) is uncorrelated with x:

$$\begin{aligned}\mathbb{E}\{\epsilon x^*\} &= \mathbb{E}\left\{\left(y - \frac{\mathbb{E}\{yx^*\}}{\mathbb{E}\{|x|^2\}}x\right)x^*\right\} \\ &= \mathbb{E}\{yx^*\} - \frac{\mathbb{E}\{yx^*\}}{\mathbb{E}\{|x|^2\}}\mathbb{E}\{|x|^2\} \\ &= 0.\end{aligned} \tag{5}$$

Since x and y are jointly Gaussian, the fact that x and ϵ are uncorrelated implies that they are also independent complex Gaussian variables. By using the decomposition in (4), it follows that

$$\begin{aligned}C_{zy} &= \mathbb{E}\{U(x)y^*\} \\ &= \frac{\mathbb{E}\{U(x)x^*\}}{\mathbb{E}\{|x|^2\}}\mathbb{E}\{xy^*\} + \underbrace{\mathbb{E}\{U(x)\epsilon^*\}}_{=0} \\ &= BC_{xy},\end{aligned} \tag{6}$$

by using that the independence between x and ϵ implies $\mathbb{E}\{U(x)\epsilon^*\} = \mathbb{E}\{U(x)\}\mathbb{E}\{\epsilon^*\} = 0$. ■

The complex Bussgang theorem is the natural complex-valued extension of Theorem 1. The corresponding complex Bussgang decomposition is given by (2) with the only exception that the Bussgang gain is now computed as $B = C_{zx}/C_x = \mathbb{E}\{U(x)x^*\}/\mathbb{E}\{|x|^2\}$ instead.

A first use case of the Bussgang decomposition is to quantify the signal-to-distortion ratio (SDR) at the output of the distortion function. The SDR is the power ratio of the desired signal Bx to the additive distortion η:

$$\text{SDR} = \frac{\mathbb{E}\{|Bx|^2\}}{\mathbb{E}\{|\eta|^2\}} = \frac{|B|^2C_x}{C_z - |B|^2C_x}, \tag{7}$$

where we have used that the additive distortion η is uncorrelated with the desired signal x.

A second use case is to analyze the performance of a communication system where $x \sim \mathcal{N}_{\mathbb{C}}(0, C_x)$ is the transmitted information signal. Suppose the receiver obtains the noisy distorted signal $U(x) + y = z + y$, where $U(\cdot)$ models the hardware distortion and y is thermal noise with power σ^2, which is uncorrelated to x. The hardware distortion might, for example, be caused of a sequence of nonideal blocks in the receiver hardware [5], as illustrated in Figure 1. The first block is the low-noise amplifier (LNA), which can distort both the amplitude and phase of the complex input signal. In the yellow part of the figure, the amplitude distortion is exemplified and clipping occurs for input signals with large amplitudes. The second block is the in-phase/quadrature (I/Q) demodulator that might have mismatches between its branches leading to I/Q imbalance. In the green curve, the effect of I/Q imbalance on a quadrature phase-shift keying constellation is shown, where the actual transmitted points are affected by leakage from the mirror subcarriers. Finally, in the analog-to-digital converter (ADC) block, the real and imaginary parts of the received signal are quantized to be represented by a finite number of bits. Quantization distortion is inevitable even if a large number of ADC bits are used [6]. We can use the Bussgang decomposition in (2) to rewrite the received signal as

$$U(x) + y = \underbrace{Bx}_{\text{Desired signal}} + \underbrace{\eta + y}_{\text{Uncorrelated signal}}. \tag{8}$$

This signal contains a desired part Bx and an uncorrelated additive "noise" term $\eta + y$. Since the latter term is

uncorrelated with x, we can utilize the *worst-case uncorrelated additive noise* theorem from [7] to compute an achievable data rate. That theorem states that the worst distribution of $\eta + y$ from a rate perspective is independent complex Gaussian, in which case, the rate is

$$\log_2\left(1+\frac{\mathbb{E}\{|Bx|^2\}}{\mathbb{E}\{|\eta|^2\}+\sigma^2}\right) = \log_2\left(1+\frac{|B|^2 C_x}{C_z-|B|^2 C_x+\sigma^2}\right) \text{ bits per channel use,} \quad (9)$$

where we have used that the distortion noise η is uncorrelated with y by Theorem 2. In fact, this is the exact expression for the generalized mutual information of the channel under Gaussian inputs and nearest-neighbor decoding [8].

An alternative computation of the Bussgang gain and two examples

If the distortion function $U(x)$ is differentiable or has finite jump discontinuities where the first derivative can be represented using the Dirac function, there is an alternative way of computing the Bussgang gain B that might be easier. We exemplify this way in the real-valued case where $x \sim \mathcal{N}(0, C_x)$ has the probability density function $p(x) = \left(1/\sqrt{2\pi C_x}\right) e^{-x^2/(2C_x)}$. Since its derivative is $p'(x) = -(x/C_x)p(x)$, we can then rewrite the Bussgang gain as

$$B = \frac{\mathbb{E}\{U(x)x\}}{C_x} = \int_{-\infty}^{\infty} \frac{U(x)x}{C_x} p(x)\,dx \overset{(a)}{=} -\int_{-\infty}^{\infty} U(x)p'(x)\,dx \overset{(b)}{=} \int_{-\infty}^{\infty} U'(x)p(x)\,dx = \mathbb{E}\{U'(x)\}, \quad (10)$$

where we identify $p'(x)$ in (a) and integrate by parts to get (b). The last expression in (10) reveals that the Bussgang gain can be also computed as the expected value of the first derivative of the distortion function. This result is a special case of the Price theorem [2].

Example 1: One-bit quantization [2, Sec. III]

Consider a real-valued signal $x \sim \mathcal{N}(0, C_x)$ that enters the nonlinear distortion function $z = U(x) = sgn(x)$, which represents one-bit quantization. The Bussgang gain can then be found as $B = \mathbb{E}\{U'(x)\} = 2\mathbb{E}\{\delta(x)\} = 2p(0) = \sqrt{2/(\pi C_x)}$, where $\delta(x)$ is the Dirac function. The same Bussgang gain can be computed as $B = \mathbb{E}\{U(x)x\}/\mathbb{E}\{x^2\} = \mathbb{E}\{|x|\}/C_x$.

A similar alternative way of computing the Bussgang gain exists in the complex-valued case, where the derivative of the distortion function $U(x)$ is defined as [9]

$$\frac{\partial U(x)}{\partial x} = \frac{1}{2}\left(\frac{\partial U(x)}{\partial \Re\{x\}} - j\frac{\partial U(x)}{\partial \Im\{x\}}\right). \quad (11)$$

One can then show that the Bussgang gain can be computed as [9]

$$B = \mathbb{E}\left\{\frac{\partial U(x)}{\partial x}\right\}. \quad (12)$$

Example 2: Third-order nonlinearity

Consider a complex-valued signal $x \sim \mathcal{N}_{\mathbb{C}}(0, C_x)$ that enters the third-order nonlinear distortion function $z = U(x) = |x|^2 x$, which might model a nonlinear amplifier [4], [10]. The Bussgang gain can be obtained as $B = \mathbb{E}\{|x|^4\}/\mathbb{E}\{|x|^2\} = 2C_x$. The same number is found by evaluating $B = \mathbb{E}\{\partial U(x)/\partial x\} = \mathbb{E}\{2|x|^2\} = 2C_x$ using (11).

The additive quantization noise model is nothing but the Bussgang decomposition

The Bussgang decomposition is unique in the sense that it is the only decomposition $z = U(x) = Bx + \eta$ of a distorted signal having the property that the additive distortion noise η is uncorrelated with the input signal x and any other jointly Gaussian signal y. No other value of B can be used to achieve that.

One seemingly different decomposition is the additive quantization noise model (AQNM) originally proposed in [6] to model quantization errors. This model is sometimes described as an alternative decomposition; however, the AQNM is nothing but the Bussgang decomposition tailored to quantization. In [6, Lemma 1], a scalar quantizer function $Q(\cdot)$ is considered, which has the property $\mathbb{E}\{x|Q(x)\} = Q(x)$, which means that each quantization interval is represented by its mean value. When the input is $x \sim \mathcal{N}_{\mathbb{C}}(0, C_x)$, the AQNM says that the output can be expressed as a summation of a scaled version of x plus an uncorrelated distortion term η:

$$z = Q(x) = (1-\beta)x + \eta, \quad (13)$$

where $\beta = \mathbb{E}\{|x - z|^2\}/C_x$ and $\mathbb{E}\{|\eta|^2\} = \beta(1-\beta)C_x$.

We will show that (13) equals the Bussgang decomposition $x = Bx + \eta$, where the Bussgang gain $B = C_{zx}/C_x$ equals $1-\beta$. Using the assumption $\mathbb{E}\{x|Q(x)\} = Q(x)$ from [6], we have

$$C_{zx} = \mathbb{E}\{Q(x)x^*\} = \mathbb{E}\{\mathbb{E}\{Q(x)x^*|Q(x)\}\} = \mathbb{E}\{Q(x)Q^*(x)\} = C_z. \quad (14)$$

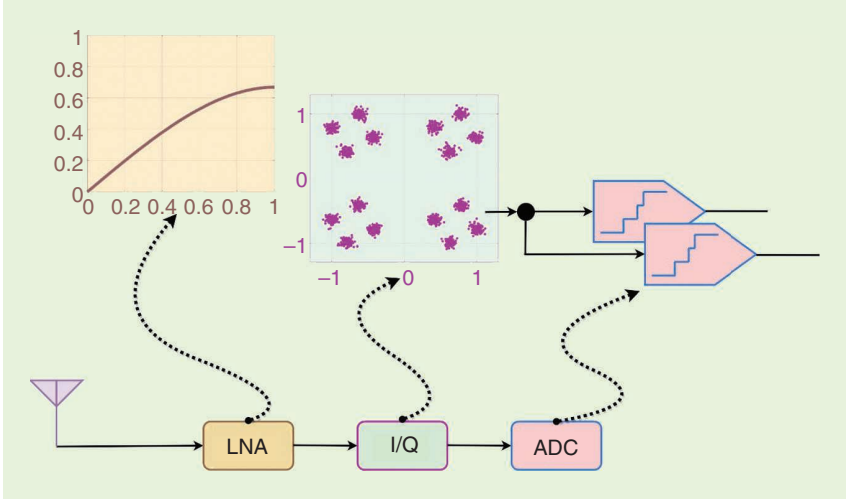


FIGURE 1. Some common sources of hardware impairments in a wireless receiver.

By utilizing this result, the scaling $1-\beta$ in (13) can be rewritten as

$$\begin{aligned} 1-\beta &= 1-\frac{\mathbb{E}\{|x-z|^2\}}{C_x} \\ &= 1-\frac{C_x+C_z-C_{zx}-C_{zx}^*}{C_x} \\ &= \frac{C_{zx}}{C_x} = B. \end{aligned} \tag{15}$$

Hence, the AQNM is a special case of the Bussgang decomposition for distortion functions that satisfy the condition $\mathbb{E}\{x\,|\,Q(x)\} = Q(x)$. The bottom line is that the Bussgang decomposition is unique but that the value of B depends on the distortion function. We also note that the Gaussianity of the input signal x is only required for η to be uncorrelated to any other jointly Gaussian random variable y. The same decomposition can be used for the non-Gaussian x, which follows from utilizing the linear MMSE estimator, but η in that case will only be uncorrelated to x.

Extension to MIMO systems

In recent years, it has become popular to analyze MIMO systems that are subject to hardware impairments, in particular, in MIMO communications [11], [12]. In this part, we extend the Bussgang results to be applicable to such cases.

Consider two jointly circularly symmetric Gaussian random vectors $\mathbf{x} \sim \mathcal{N}_{\mathbb{C}}(\mathbf{0}, \mathbf{C}_x)$ and $\mathbf{y} \sim \mathcal{N}_{\mathbb{C}}(\mathbf{0}, \mathbf{C}_y)$, which both have length M. The correlation matrices are denoted as $\mathbf{C}_x = \mathbb{E}\{\mathbf{x}\mathbf{x}^{\mathrm{H}}\}$ and $\mathbf{C}_y = \mathbb{E}\{\mathbf{y}\mathbf{y}^{\mathrm{H}}\}$ and are assumed to have full rank. The cross-correlation matrix is denoted as $\mathbf{C}_{xy} = \mathbb{E}\{\mathbf{x}\mathbf{y}^{\mathrm{H}}\}$. Using this notation, we can generalize the Bussgang theorem as follows.

Theorem 3: The Bussgang theorem for MIMO distortions

Consider the jointly circularly symmetric Gaussian random vectors $\mathbf{x}$ and $\mathbf{y}$. Let $\mathbf{U}: \mathbb{C}^M \to \mathbb{C}^M$ denote a distortion function and $\mathbf{z} = \mathbf{U}(\mathbf{x})$ is the distorted signal when using $\mathbf{x}$ as input. The cross-correlation matrix $\mathbf{C}_{zy} = \mathbb{E}\{\mathbf{z}\mathbf{y}^{\mathrm{H}}\}$ of $\mathbf{z}$ and $\mathbf{y}$ is a linear transformation of the cross-correlation matrix $\mathbf{C}_{xy}$ of $\mathbf{x}$ and $\mathbf{y}$:

$$\mathbf{C}_{zy} = \mathbf{C}_{zx}\mathbf{C}_x^{-1}\mathbf{C}_{xy}. \tag{16}$$

Proof

The proof is a matrix extension of the proof of Theorem 2. Let us express $\mathbf{y}$ as a summation of the MMSE estimate of it given $\mathbf{x}$ and the estimation error $\boldsymbol{\epsilon} \in \mathbb{C}^M$:

$$\mathbf{y} = \mathbf{C}_{yx}\mathbf{C}_x^{-1}\mathbf{x} + \boldsymbol{\epsilon}, \tag{17}$$

where $\boldsymbol{\epsilon}$ is defined as $\boldsymbol{\epsilon} = \mathbf{y} - \mathbf{C}_{yx}\mathbf{C}_x^{-1}\mathbf{x}$. If we multiply both sides of (17) by $\mathbf{x}^{\mathrm{H}}$ from the right and take the expectation, we obtain

$$\begin{aligned} \mathbf{C}_{yx} &= \mathbf{C}_{yx}\mathbf{C}_x^{-1}\mathbf{C}_x + \mathbb{E}\{\boldsymbol{\epsilon}\mathbf{x}^{\mathrm{H}}\} \\ &= \mathbf{C}_{yx} + \mathbb{E}\{\boldsymbol{\epsilon}\mathbf{x}^{\mathrm{H}}\}, \end{aligned} \tag{18}$$

from which it follows that $\mathbb{E}\{\boldsymbol{\epsilon}\mathbf{x}^{\mathrm{H}}\} = \mathbf{0}$. Hence, $\boldsymbol{\epsilon}$ and $\mathbf{x}$ are uncorrelated, which implies that they are also independent since these are jointly Gaussian variables. Finally, we obtain (16) as

$$\begin{aligned} \mathbf{C}_{zy} &= \mathbb{E}\{\mathbf{z}\mathbf{y}^{\mathrm{H}}\} \\ &= \mathbb{E}\{\mathbf{z}\mathbf{x}^{\mathrm{H}}\}\mathbf{C}_x^{-1}\mathbf{C}_{yx}^{\mathrm{H}} + \mathbb{E}\{\mathbf{z}\boldsymbol{\epsilon}^{\mathrm{H}}\} \\ &= \mathbf{C}_{zx}\mathbf{C}_x^{-1}\mathbf{C}_{xy} \end{aligned} \tag{19}$$

by utilizing that $\mathbf{C}_{yx}^{\mathrm{H}} = \mathbf{C}_{xy}$ and that $\mathbb{E}\{\mathbf{z}\boldsymbol{\epsilon}^{\mathrm{H}}\} = \mathbf{0}$ since $\mathbf{z}$ and $\boldsymbol{\epsilon}$ are independent. ■

From this theorem we notice that the Bussgang gain is represented by the matrix

$$\mathbf{B} = \mathbf{C}_{zx}\mathbf{C}_x^{-1}, \tag{20}$$

and we call it a MIMO extension since the distortion function takes multiple inputs and provide multiple outputs. It is possible to extend the result to the case where $\mathbf{C}_x$ is rank deficient, in which case the inverse in (20) is replaced by a pseudo-inverse; see [4, Sec. II.A] for details.

A consequence of Theorem 3 is the Bussgang decomposition for MIMO functions:

$$\mathbf{z} = \mathbf{U}(\mathbf{x}) = \mathbf{B}\mathbf{x} + \boldsymbol{\eta}, \tag{21}$$

where the additive distortion term $\boldsymbol{\eta}$ is uncorrelated both with $\mathbf{x}$ and any other Gaussian random vector $\mathbf{y}$ that is correlated with $\mathbf{x}$. This result is illustrated in Figure 2(a).

Element-wise distortion for MIMO systems

The Bussgang decomposition for MIMO functions has been widely used to model the hardware impairments in multiple-antenna communication systems [11]. In this case, M is the number of receive antennas and the distortion function represents

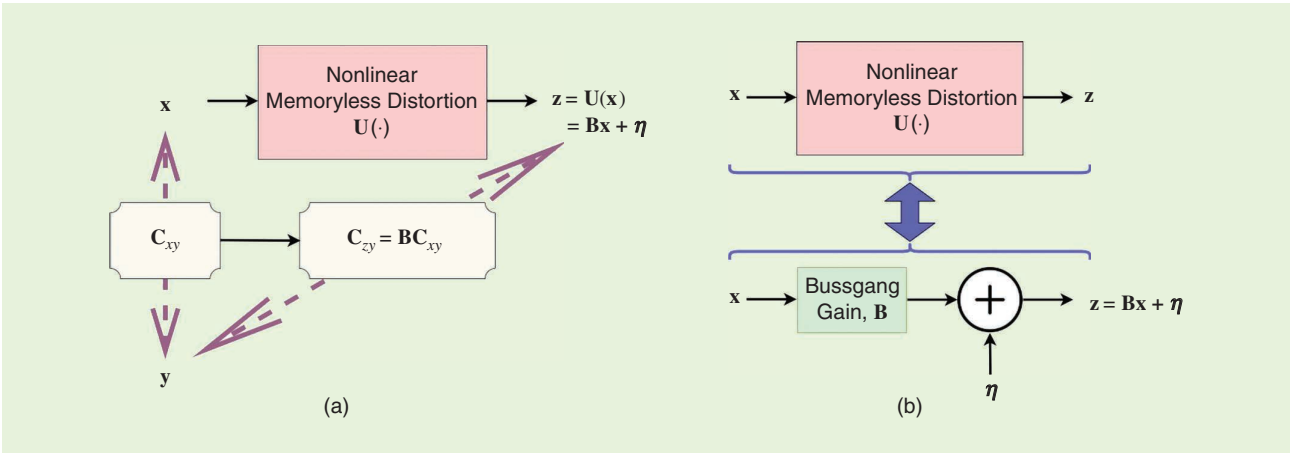


FIGURE 2. The Bussgang decomposition for a nonlinear memoryless distortion function $\mathbf{U}(\cdot)$. (a) the Bussgang decomposition for jointly Gaussian random vectors $\mathbf{x}$ and $\mathbf{y}$ and (b) the generalized Bussgang decomposition for a non-Gaussian random vector $\mathbf{x}$.

impairments in the antenna branches. A common assumption is that there is no crosstalk between the branches, so that each one can be separately modeled in the way shown in Figure 1. The distortion function then has the form

$$\mathbf{z} = \mathbf{U}(\mathbf{x}) = \begin{bmatrix} U_1(x_1) \\ \vdots \\ U_M(x_M) \end{bmatrix}, \tag{22}$$

where x_m denotes the mth element of $\mathbf{x}$. Hence, each output is a distorted version of only the input having the same index. We can then simplify the Bussgang matrix. We note that all elements of $\mathbf{x}$ are jointly Gaussian and the distortion noise for each element of $\mathbf{z}$ is thus uncorrelated to $\mathbf{x}$ by Theorem 2. Then, it follows that $\mathbf{C}_{zx} = \mathbf{D}\mathbf{C}_x$, where $\mathbf{D} = \text{diag}(d_1, \ldots, d_M)$ is a diagonal matrix and $d_m = \mathbb{E}\{U_m(x_m)x_m^*\}/\mathbb{E}\{|x_m|^2\}$ is the Bussgang gain corresponding to the mth component of the distortion function, i.e., $z_m = U_m(x_m)$. Hence, the Bussgang gain matrix of the overall MIMO distortion becomes $\mathbf{B} = \mathbf{C}_{zx}\mathbf{C}_x^{-1} = \mathbf{D}$ from (20), and we obtain the simplified Bussgang decomposition

$$\mathbf{z} = \mathbf{D}\mathbf{x} + \boldsymbol{\eta} = \begin{bmatrix} d_1 x_1 \\ \vdots \\ d_M x_M \end{bmatrix} + \boldsymbol{\eta}. \tag{23}$$

Hence, when an element-wise distortion function affects the Gaussian signal $\mathbf{x}$, the output $\mathbf{z}$ is an element-wise scaled version of $\mathbf{x}$ plus a distortion vector $\boldsymbol{\eta}$ that is uncorrelated with $\mathbf{x}$. We reiterate that the distortion vector $\boldsymbol{\eta}$ is uncorrelated to any other vector $\mathbf{y}$ that is jointly Gaussian with $\mathbf{x}$, which is the part of this result that critically requires $\mathbf{x}$ and $\mathbf{y}$ to be Gaussian distributed.

Are the elements of the distortion noise $\boldsymbol{\eta}$ uncorrelated?

Since the Bussgang gain matrix is diagonal when having element-wise distortions, one may tend to think that the elements of the distortion $\boldsymbol{\eta}$ will also be uncorrelated, so that we effectively get one separate Bussgang decomposition per received signal. However, this is generally not the case, as we show next. Let $\mathbf{C}_\eta = \mathbb{E}\{\boldsymbol{\eta}\boldsymbol{\eta}^{\text{H}}\} \in \mathbb{C}^{M\times M}$ denote the correlation matrix of the distortion vector $\boldsymbol{\eta}$. Using the fact that $\boldsymbol{\eta}$ is uncorrelated with $\mathbf{x}$, it can be computed as

$$\mathbf{C}_\eta = \mathbf{C}_z - \mathbf{B}\mathbf{C}_x\mathbf{B}^{\text{H}}. \tag{24}$$

Whenever the input signal $\mathbf{x}$ contains correlated elements, such that $\mathbf{C}_x$ is nondiagonal, the correlation matrix will likely also be nondiagonal. This is intuitively quite clear: If two (almost) identical signals are sent through identical hardware components, then the distortion should also be (almost) identical. This type of correlation typically appears in wireless communications since each receive antenna observes a different linear combination of the same transmitted information signals. Some conditions for when the correlation can be neglected, so that $\mathbf{C}_\eta$ is approximately diagonal, are derived in [4]. However, it is rather common that the correlation is neglected without motivation (cf. [12]), which might lead to substantial approximation errors.

As an example, we consider a setup where a four-antenna receiver quantizes the real and imaginary parts of each entry in the received signal $\mathbf{x}$ using identical b-bit ADCs. The input signal is generated as $\mathbf{x} = \mathbf{H}\mathbf{s}$, where $\mathbf{H} \in \mathbb{C}^{4\times 4}$ is the MIMO channel matrix from a four-antenna transmitter. We consider Rayleigh fading where $\mathbf{H}$ has independent $\mathcal{N}_{\mathbb{C}}(0,1)$-distributed entries. For each channel realization, $\mathbf{H}$ is assumed perfectly known and the transmitted signal is $\mathbf{s} \sim \mathcal{N}_{\mathbb{C}}(\mathbf{0}, \mathbf{I}_4)$, so $\mathbf{x}$ is conditionally complex Gaussian distributed. The Bussgang decomposition then says that the ADC output can be written as $\mathbf{z} = \mathbf{D}\mathbf{x} + \boldsymbol{\eta}$. To demonstrate that the elements of $\boldsymbol{\eta}$ are correlated, Figure 3 shows the cumulative distribution function (CDF) of the normalized off-diagonal elements of $\mathbf{C}_\eta$ (i.e., the correlation coefficients) for different number of ADC bits. When the ADC resolution is low, most of the correlation coefficients are nonzero and some are rather large. However, when the ADC resolution is high, the off-diagonal elements are almost zero and can potentially be approximated as zero when quantifying communication rates.

The generalized Bussgang decomposition for non-Gaussian input signals

In the Bussgang theorem, we are utilizing that $\mathbf{x}$ and $\mathbf{y}$ are Gaussian signals. The main result cannot be generalized to non-Gaussian signals. However, we can always decompose the distorted signal according to (21) using the Bussgang gain matrix $\mathbf{B} = \mathbf{C}_{zx}\mathbf{C}_x^{-1}$, but it generally will not be a diagonal matrix, even if an element-wise distortion of the type in (22) is used. As mentioned previously, the intuition is that $\mathbf{B}\mathbf{x}$ is the linear MMSE estimate of $\mathbf{z}$ given a non-Gaussian distributed observation $\mathbf{x}$. In this analogy, $\boldsymbol{\eta}$ is the estimation error which is uncorrelated with $\mathbf{x}$ since

$$\begin{aligned}\mathbb{E}\{\boldsymbol{\eta}\mathbf{x}^{\text{H}}\} &= \mathbb{E}\{(\mathbf{z} - \mathbf{C}_{zx}\mathbf{C}_x^{-1}\mathbf{x})\mathbf{x}^{\text{H}}\} \\ &= \mathbf{C}_{zx} - \mathbf{C}_{zx}\mathbf{C}_x^{-1}\mathbf{C}_x = \mathbf{0}.\end{aligned} \tag{25}$$

The generalized Bussgang decomposition for the non-Gaussian input $\mathbf{x}$ is illustrated in Figure 2(b). It is suitable both for quantifying the SDR and for

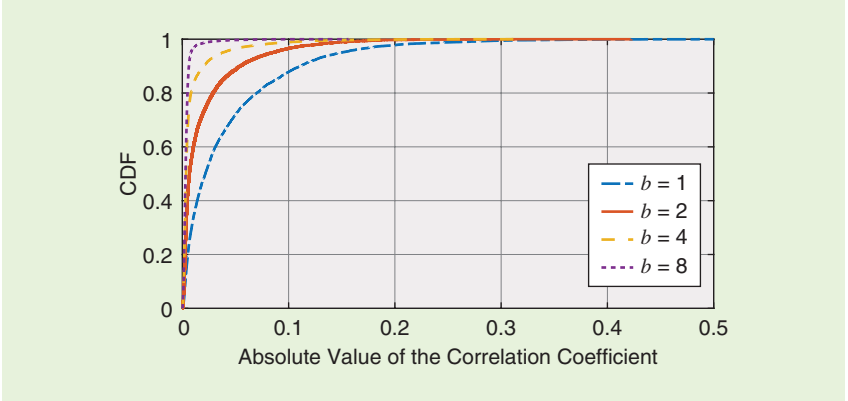


FIGURE 3. The cumulative distribution function (CDF) of the absolute value of the correlation coefficient between elements in $\boldsymbol{\eta}$.

analyzing the performance of nonlinear communication systems. For example, [10] did this using practically modulated data signals. That article also showed that although treating the uncorrelated distortion $\boldsymbol{\eta}$ as independent Gaussian noise is convenient, one can increase the performance by exploiting its information content.

Lessons learned

The Bussgang decomposition establishes that the output of a nonlinear function is a scaled version of the random input signal plus an uncorrelated distortion term. It is an exact and unique representation. The distortion is not independent and not Gaussian, but it can be treated as that to obtain a lower bound on the communication performance. The decomposition can be extended to MIMO systems, but then, the entries of the distortion vector are generally mutually correlated.

Acknowledgments

The authors were partially supported by the Excellence Center at Linköping–Lund in Information Technology (ELLIIT) and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Authors

Özlem Tuğfe Demir (ozlem.tugfe.demir@liu.se) received her Ph.D. degree in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 2018. She is currently a postdoctoral researcher at Linköping University, Linköping, Sweden. Her research interests focus on signal processing and optimization in wireless communications; massive multiple-input, multiple-output systems; deep learning; and green communications. She is a Member of IEEE.

Emil Björnson (emil.bjornson@liu.se) received his Ph.D. degree in telecommunications from the KTH Royal Institute of Technology, Sweden, in 2011. He is currently an associate professor at Linköping University, Linköping, Sweden. He has authored *Optimal Resource Allocation in Coordinated Multi-Cell Systems* (2013) and *Massive MIMO Networks: Spectral, Energy, and Hardware Efficiency* (2017). He received the 2018 IEEE Marconi Prize Paper Award in Wireless Communications, the 2019 European Association for Signal Processing Early CAREER Award, the 2019 IEEE Communications Society Fred W. Ellersick Prize, and the 2019 IEEE Signal Processing Magazine Best Column Award. In September 2020, he became a part-time visiting full professor at the KTH Royal Institute of Technology. He is a Senior Member of IEEE.

References

[1] J. J. Bussgang, "Crosscorrelation functions of amplitude-distorted Gaussian signals," Research Lab. Electronics, Massachusetts Inst. Technology, Cambridge, MA, Tech. Rep. 216, 1952. [Online]. Available: http://hdl.handle.net/1721.1/4847

[2] H. E. Rowe, "Memoryless nonlinearities with Gaussian inputs: Elementary results," *Bell Syst. Tech. J.*, vol. 61, no. 7, pp. 1519–1525, 1982. doi: 10.1002/j.1538-7305.1982.tb04356.x.

[3] J. Minkoff, "The role of AM-to-PM conversion in memoryless nonlinear systems," *IEEE Trans. Commun.*, vol. 33, no. 2, pp. 139–144, 1985. doi: 10.1109/TCOM.1985.1096262.

[4] E. Björnson, L. Sanguinetti, and J. Hoydis, "Hardware distortion correlation has negligible impact on UL massive MIMO spectral efficiency," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1085–1098, Feb. 2019. doi: 10.1109/TCOMM.2018.2877331.

[5] T. Schenk, *RF Imperfections in High-Rate Wireless Systems: Impact and Digital Compensation*. Berlin: Springer-Verlag, 2008.

[6] A. K. Fletcher, S. Rangan, V. K. Goyal, and K. Ramchandran, "Robust predictive quantization: Analysis and design via convex optimization," *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 4, pp. 618–632, 2007. doi: 10.1109/JSTSP.2007.910622.

[7] B. Hassibi and B. M. Hochwald, "How much training is needed in multiple-antenna wireless links?" *IEEE Trans. Inf. Theory*, vol. 49, no. 4, pp. 951–963, 2003. doi: 10.1109/TIT.2003.809594.

[8] W. Zhang, Y. Wang, C. Shen, and N. Liang, "A regression approach to certain information transmission problems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2517–2531, 2019. doi: 10.1109/JSAC.2019.2933964.

[9] W. McGee, "Circularly complex Gaussian noise—A price theorem and a Mehler expansion," *IEEE Trans. Inf. Theory*, vol. 15, no. 2, pp. 317–319, 1969. doi: 10.1109/TIT.1969.1054293.

[10] Ö. T. Demir and E. Björnson, "Channel estimation in massive MIMO under hardware non-linearities: Bayesian methods versus deep learning," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 109–124, 2020. doi: 10.1109/OJCOMS.2019.2959913.

[11] E. Björnson, J. Hoydis, M. Kountouris, and M. Debbah, "Massive MIMO systems with non-ideal hardware: Energy efficiency, estimation, and capacity limits," *IEEE Trans. Inf. Theory*, vol. 60, no. 11, pp. 7112–7139, 2014. doi: 10.1109/TIT.2014.2354403.

[12] L. Xu, X. Lu, S. Jin, F. Gao, and Y. Zhu, "On the uplink achievable rate of massive MIMO system with low-resolution ADC and RF impairments," *IEEE Commun. Lett.*, vol. 23, no. 3, pp. 502–505, 2019. doi: 10.1109/LCOMM.2019.2895823.

 (continued from page 139)

This close coupling of emerging application areas, processing devices, and theoretical advances demands expertise, which cuts across the three areas of interest of the three predecessor technical communities. The ASPS TC addresses this cross-cutting requirement.

A notable, unique, aspect of the ASPS community is its industrial links. This is as a result of its focus on identifying practical industrial applications of signal processing and applying known signal processing techniques and theories to their practical contexts. This is a natural consequence of the development of the ASPS as an extension of the IDSPT committee, but industrial interaction is a common theme across all of the progenitor communities. This focus will manifest in a number of ways; the ASPS sees a high percentage of industrial involvement via committee membership. But indeed, the linkages run deeper. The ASPS TC organizes the Industry Technology Track at ICASSP. This covers a huge range of areas of interest relevant to industry. For this track, we advocate immediately important applications, commercial-ready implementations, and key signal processing technologies that are ready for industry adoption. A good number of the papers in the track tend to come from industry, while others may come from academia but are capable of withstanding the review and selection process that involves numerous tough industry reviewers.

Generally, the ASPS committee provides a forum for industry people and academics to get together and share current industry-focused interest areas and the promising emerging signal processing technology that can apply. For example, with the trend toward the adoption of machine learning techniques to solve conventional signal processing applications, we are seeing increased industry scrutiny on how practical and implementable these new approaches can be. With the mix of industry and academic people in the ASPS, we are well positioned to recognize both the most impactful possible upcoming technologies as well as those than can be practically implemented and commercialized. More specifically, TC Member Ivan Tashev (Microsoft) chairs the Industry Technical Working Group (TWG), which includes membership from Intel, HIPCAM Global, Novartis, Yahoo, and NVIDIA, alongside academia. The remit of this TWG considers how the SPS can better serve the needs of its industrial members, where there is sometimes less of a premium placed on traditional academic activities, such as publishing papers. The ASPS TC is keen to involve the views of all corners of the Society in this process, so please feel free to contact the TC leadership should you wish to offering suggestions or get further involved.

Of course, the TC is heavily involved in cutting-edge research alongside its industrial considerations. Recent years have seen a near tripling in submissions to the DISPS track at ICASSP, and the TC also sponsors the annual IEEE Workshop on Signal Processing Systems (SiPS)—the leading international gathering on processing architectures, software, and design. The 2019 edition, held in Nanjing, China, saw 120 delegates enjoy a technical program with 42 papers, together with an excellent array of keynote speakers, including Prof. K. Parhi (the University of Minnesota, United States), Prof. Deming Chen (the University of Illinois at Urbana-Champaign, United States), and Sunny Zhang (Intel) (see Figure 1).

SiPS 2020 was due to be held in Coimbra, Portugal, in October 2020, but the on-location event had to be replaced by a virtual alternative due to uncertainty related to COVID-19. Chaired by Prof. Leonel Sousa (the Instituto de Engenharia de Sistemas e Computadores—Investigação e Desenvolvimento, Lisbon) and Prof. Joe Cavallaro (Rice University, Texas), and with a program led by Prof. Gabriel Falcao (the University of Coimbra) and Dr. Farhana Shiekh (Intel Corporation), SiPS 2020 welcomed more than 240 delegates who enjoyed another 43 papers as part of an event that was a tremendous success, particularly the dazzling assortment of keynote speakers. Luca Benini (ETH Zurich and the University of Bologna, Italy) spoke about low-power processors for edge-based signal processing and artificial intelligence, Vivienne Sze (the Massachusetts Institute of Technology, Cambridge) discussed deep neural network evaluation, and Shilpa Talwar (Intel Corporation) reviewed wireless intelligent systems.

FIGURE 1. Attendees at SiPS 2019 in Nanjing, China.

These combined foci on both academic and industrial impacts, and the strong mix of design; implementation; applications; and technologists; both academic and industrial, offer an extremely promising mix for the future of the ASPS TC.

Authors

John McAllister (jp.mcallister@ieee.org) received his Ph.D. degree in electronic engineering from Queen's University Belfast, Belfast, U.K., in 2004, where he is currently a member of the academic staff in the School of Electronics, Electrical Engineering and Computer Science. He was a chair of the IEEE Technical Committee (TC) on Design and Implementation of Signal Processing Systems (DISPS) in 2020 and is the founding chair of the IEEE Applied Signal Processing Systems TC. He is the senior area editor of *IEEE Transactions on Signal Processing*, the founding chief editor of the IEEE Signal Processing Society Resource Center, a cofounder of Analytics Engines Ltd., and a member of the editorial board of *Journal of Signal Processing Systems*. His research addresses resource-constrained and embedded signal processing and artificial intelligence.

Mike Polley (polley@alum.mit.edu) received his B.S., M.S., and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1989, 1990, and 1996, respectively. He is a senior vice president and the head of the Mobile Processor Innovation Lab at Samsung Research America, Plano, Texas, USA. He currently serves as chair of the Industry Digital Signal Processing Technology Standing Committee for 2019–2020. His research interests include computational imaging and artificial intelligence-based camera systems. He is a Senior Member of IEEE.

Roozbeh Jafari (rjafari@tamu.edu) received his Ph.D. degree in computer science from the University of California (UC), Los Angeles, and completed a postdoctoral fellowship at UC Berkeley. He is currently a professor of biomedical engineering, computer science and engineering, and electrical and computer engineering at Texas A&M University, College Station, Texas, USA. He is the recipient of the NSF CAREER Award (2012), the ACM Transactions on Embedded Computing Systems Best Paper Award (2019), and the Outstanding Engineering Contribution award from the College of Engineering at Texas A&M (2019), to name a few. He is an associate editor of *IEEE Sensors Journal* and *IEEE Internet of Things Journal*, among others. He serves on scientific panels for funding agencies frequently and is presently a standing member of the National Institutes of Health Biomedical Computing and Health Informatics Study Section. His research interests lie in the areas of wearable computer design and signal processing. He is a Senior Member of IEEE.

SP

John McAllister, Mike Polley, and Roozbeh Jafari

The Applied Signal Processing Systems Technical Committee

The Applied Signal Processing Systems (ASPS) Technical Committee (TC) is a brand-new member of the TC roster of the IEEE Signal Processing Society (SPS). It was formed at the start of 2021 as a merger of a number of previously independent groups: the Design and Implementation of Signal Processing Systems (DISPS) TC, the Internet of Things (IoT) Special Interest Group (SIG), and the Industry Digital Signal Processing Technology (IDSPT) Standing Committee. These groupings had similar outlooks on signal processing technology that were not focused on a particular class of signals or specific application areas, but rather sought to apply practical processing technologies such as embedded processors, programmable hardware and custom chips (the DISPS community), and distributed and networked IoT platforms (the IoT SIG) to new application areas (the IDSPT). This complementary outlook makes merging these into a single entity that combines the best aspects of all three an exciting opportunity for the new TC and the SPS itself.

The success of digital signal processing (DSP) technology, in particular, has been based on the continuous development of high-performance, efficient computing devices, processors, and software. The constant increases in computational capacity per unit cost afforded by Moore's law have worked hand in glove with the development of new algorithms to drive the development of wireless communications, image processing and computational imaging, and many other areas. This extended into diverse processing architectures, moving from DSP chips to programmable hardware, such as field-programmable gate array, graphics processing units, and complex systems on chip, further extending the reach of DSP capabilities. However, as advances in networking—both wired and wireless—have emerged, complex, multilayer signal processing systems have become established.

The IoT, in particular, is gaining significant traction in the consumer and industrial markets; the bold vision of unobtrusive, pervasive, and continuous sensing, computation and actuation are taking hold in everyday life. IoT platforms offer unique features, constraints, and requirements. Powerful and distributed sensing is enabling new applications, leveraging sensing modalities such as acoustic; motion; temperature; humidity; acceleration; or even physiological sensors for wearable computers. The limited availability of power—often battery operated—or energy-harvesting systems imposes additional constraints, requiring low power and duty cycling for sensing and reducing computation, storage, and communications.

Various data-transfer methods, including low power, and stateless communication paradigms such as Bluetooth Low Energy; storage; and burst communication as well as passive methods, such as the National Electrical Code and radio-frequency identification-assisted methods, are actively leveraged. Novel wireless communication paradigms provide additional services, such as localization. Actuation and feedback provide opportunities for health care, industrial and residential energy management, safety, disaster response, self-driving, and many more new paradigms all aimed at creating a safer, more efficient and more desirable smart environment, homes, cities, communities, and services. All in all, the technical challenges are introducing new opportunities for R&D to continually push the performance envelope of IoT platforms while enabling new applications with significant impacts on individuals and communities.

Alongside their companion cloud- and edge-processing contexts, the IoT is enabling major new areas of signal processing endeavors to emerge and become established, with increasing speed. New kinds of structure (such as sparsity, networked, or graph-structure), have emerged and been exploited in new application areas (such as body centric, neural, swarming, or the IoT). These are powered by rapidly advancing theoretical foundations and an accompanying raft of devices, which tailor their capability to both application and data structures, such as Google's Tensor Processing Unit and Coral, and brain-inspired or graph processing units.

(continued on page 137)

Digital Object Identifier 10.1109/MSP.2020.3032116
Date of current version: 24 December 2020

DATES AHEAD

Please send calendar submissions to:
Dates Ahead, Att: Samantha Walter, E-mail: walter.samantha@ieee.org

Editor's Note

Due to changing situations around the world because of the novel coronavirus (COVID-19) outbreak, please double-check each conference's website for the latest news and updates.

2021

JANUARY

Virtual: 28th European Signal Processing Conference (EUSIPCO)
18–22 January, Amsterdam, The Netherlands.
General Chairs: Richard Heusdens and Cédric Richard
URL: https://eusipco2020.org

Virtual: IEEE Spoken Language Technology Workshop (SLT)
19–22 January, Shenzhen, China.
General Chairs: Zhijian Ou and Lei Xie
URL: http://slt2020.org/index.html

MARCH

Data Compression Conference (DCC)
23–26 March, Snowbird, Utah, United States.
Conference Chairs: Michael W. Marcellin and James A. Storer
URL: https://www.cs.brandeis.edu/~dcc/

APRIL

IEEE International Symposium on Biomedical Imaging (ISBI)
13–16 April, Nice, France.
Conference Chairs: Laure Blanc-Féraud and Françoise Peyrin
URL: https://biomedicalimaging.org/2021/

Digital Object Identifier 10.1109/MSP.2020.3028254
Date of current version: 24 December 2020

©SHUTTERSTOCK.COM/R.M. NUNES

The 2021 IEEE International Conference on Autonomous Systems is scheduled to be held 11–13 August in Montréal, Québec, Canada.

MAY

International Conference on Information Processing in Sensor Networks (IPSN)
18–21 May, Nashville, Tennessee, United States.
General Chair: Ákos Lédeczi
URL: https://ipsn.acm.org/2021/

JUNE

IEEE Data Science and Learning Workshop (DSLW)
5–6 June, Toronto, Ontario, Canada.
General Chairs: Stark Draper and Z. Jane Wang
URL: https://conferences.ece.ubc.ca/dslw2021/#/

IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)
6–11 June, Toronto, Ontario, Canada.
General Chairs: Dimitri Androutsos, Kostas Plataniotis, and Xiao-Ping (Steven) Zhang
URL: https://2021.ieeeicassp.org/

JULY

IEEE International Conference on Multimedia and Expo (ICME)
5–9 July, Shenzhen, China.
General Chairs: Moncef Gabbouj, Houqiang Li, Guo-Jun Qi, and Yonghong Tian
URL: https://2021.ieeeicme.org/

IEEE Statistical Signal Processing Workshop (SSP)
11–14 July, Rio de Janeiro, Brazil.
General Chair: Rodrigo C. de Lamare
URL: http://ssp2020.cetuc.puc-rio.br

AUGUST

IEEE International Conference on Autonomous Systems (ICAS)
11–13 August, Montréal, Québec, Canada.
General Cochairs: Amir Asif and Arash Mohammadi
URL: https://2021.ieee-icas.org

SEPTEMBER

IEEE International Conference on Image Processing (ICIP)
19–22 September, Anchorage, Alaska, United States.
General Chair: Saif alZahir
URL: https://2021.ieeeicip.org

IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)
27–30 September, Lucca, Italy.
General Chair: Luca Sanguinetti
URL: https://www.spawc2021.com

SP

Call for Papers

Special Issue on Deep Learning in Biological Image and Signal Processing

Studies of the fundamental structural and functional properties of life, from molecules to cells, tissues, organs, and complete organisms including human life, nowadays critically depend on advanced imaging systems and measurement devices generating data of ever-increasing quantity and complexity. Deep learning of artificial neural networks has emerged as a powerful tool for extracting the relevant information from such data and helping researchers to detect patterns that may be unnoticeable to the human senses.

Many scientific and engineering challenges remain to improve the efficacy of deep learning methods and make them trustworthy enough for use in critical biological image and signal processing tasks. Deep learning needs to become more explainable and interpretable, more generalizable and transferable across applications, especially where little or only weakly annotated data is available, and optimal network design needs to become more automated. Signal processing theory, information theory, statistics and other fields could play a key role in filling the gaps.

This special issue of *IEEE Signal Processing Magazine* provides a venue for a wide and diverse audience to survey recent research advances in deep learning for applications in biological image and signal processing. Fostering cross-pollination between data-driven and model-driven approaches, the special issue aims to inspire researchers in developing novel solutions to current challenges of deep learning in biological applications.

Topics of interest include but are not limited to:

- Deep multimodal bioimage/biosignal processing
- Integrated data-driven and model-driven approaches
- Explainable deep learning for bioimage/biosignal analysis
- Unsupervised and weakly supervised deep learning
- Deep learning strategies in imaging genetics
- Graph neural networks for connectivity analysis
- Privacy-preserving distributed deep learning
- Deep learning for biomarker discovery
- Generative deep models for disease fingerprints
- Annotation-efficient deep learning strategies

Important Dates

White papers due:	1 February 2021
Invitation notification:	1 March 2021
Full manuscripts due:	1 May 2021
First review to authors:	1 July 2021
Revision due:	1 September 2021
Final decision:	1 November 2021
Final materials due:	1 December 2021
Publication:	1 March 2022

All topics are to be covered from the perspective of applications in biological research. Papers focusing entirely on clinical medical applications will not be considered.

White papers are required, and full articles will be invited based on the review of white papers. The white paper format is up to 4 pages in length, including the proposed title, motivation and significance of the topic, an outline of the proposed paper, and representative references. An author list with contact information and short bios should also be included. **Submitted articles must be of tutorial/overview/survey nature, in an accessible style to a broad audience, and have a significant relevance to the scope of the special issue.** Submissions must not have been published or be under review elsewhere, and must be made online at https://mc.manuscriptcentral.com/sps-ieee. For submission guidelines, see the Information for Authors at https://signalprocessingsociety.org/publications-resources/ieee-signal-processing-magazine/information-authors-spm.

Guest Editors

Prof. Erik Meijering, University of New South Wales, Australia, meijering@imagescience.org
Prof. Vince D. Calhoun, TReNDS, Georgia State/Tech and Emory University, USA, vcalhoun@gatech.edu
Prof. Gloria Menegaz, University of Verona, Italy, gloria.menegaz@univr.it
Prof. David J. Miller, Pennsylvania State University, USA, djm25@psu.edu
Prof. Jong Chul Ye, Korea Advanced Institute of Science and Technology (KAIST), Korea, jong.ye@kaist.ac.kr

Digital Object Identifier 10.1109/MSP.2020.3040489

MATLAB SPEAKS MACHINE LEARNING

With MATLAB® you can use clustering, regression, classification, and deep learning to build predictive models and put them into production.

mathworks.com/machinelearning

MathWorks®