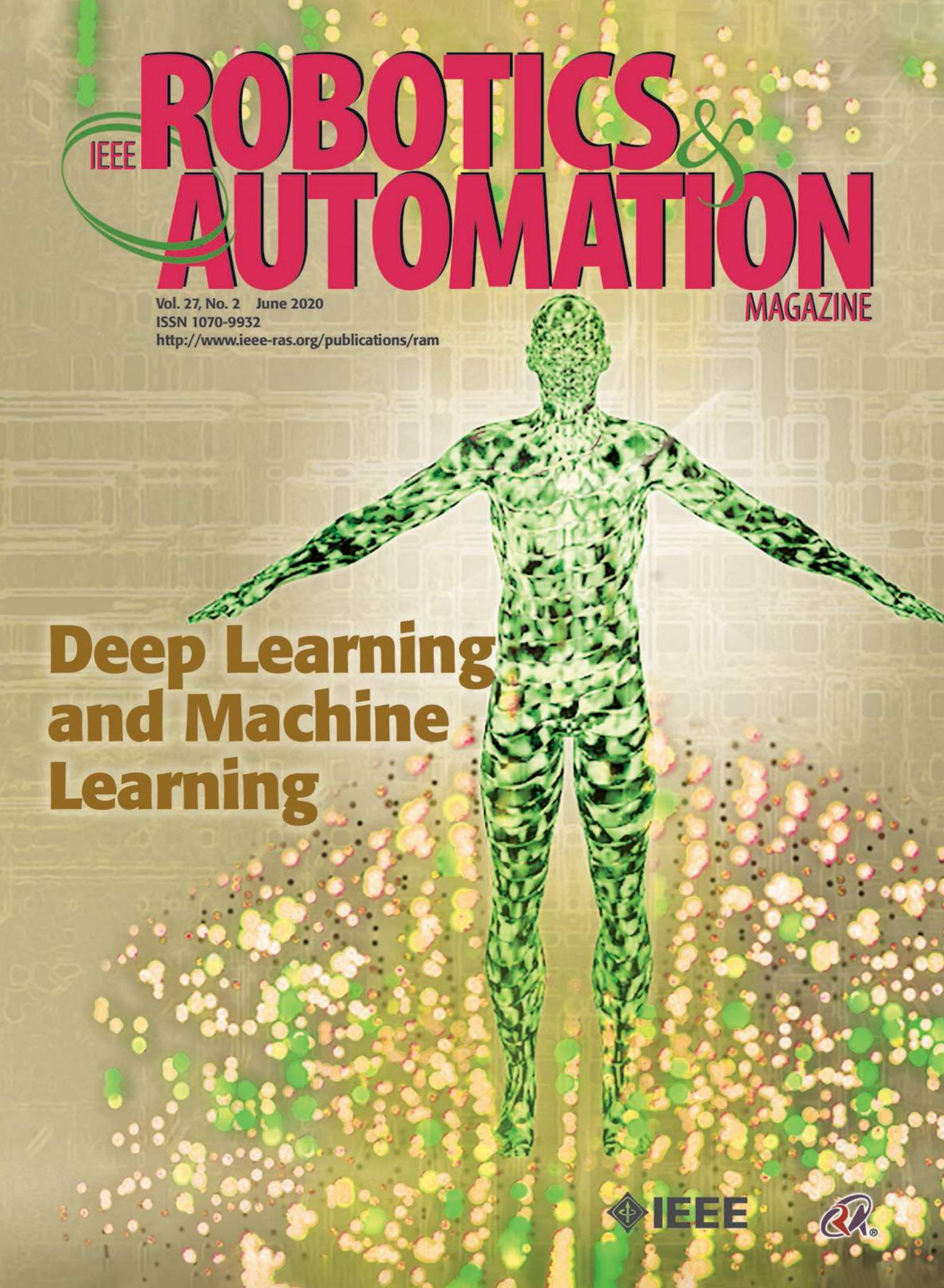
IEEE ROBOTICS & AUTOMATION MAGAZINE

Vol. 27, No. 2 June 2020
ISSN 1070-9932
http://www.ieee-ras.org/publications/ram



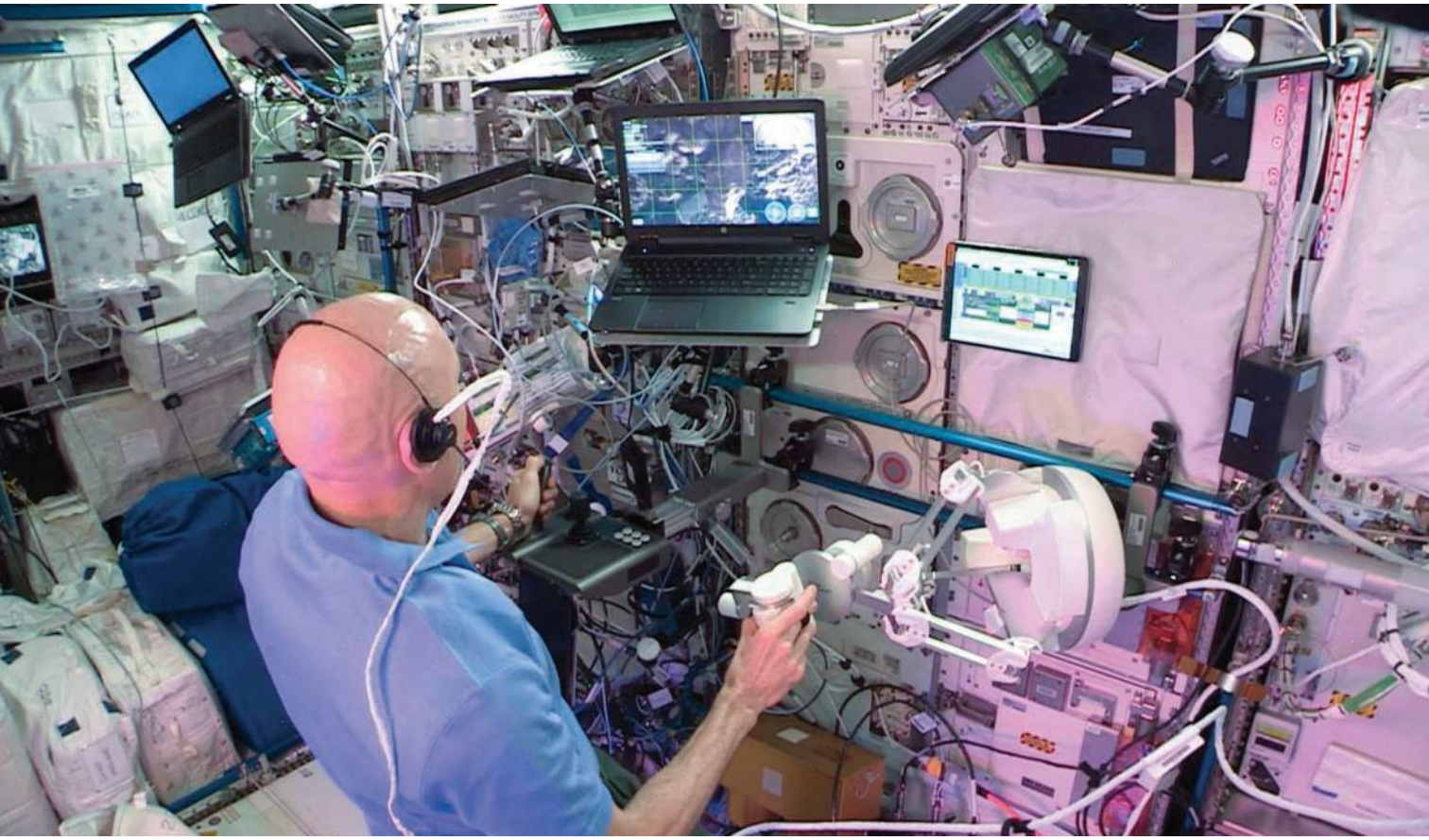
Deep Learning and Machine Learning

IEEE

force dimension

Displaying the world in your hands. Inventing new ways to interact.

Force Dimension designs and manufactures the finest **master haptic devices** for leading-edge applications in research, medical, industry, and human exploration.



From space astronaut Luca Parmitano uses the haptic plateform to control a robot manipulator ©2019 ESA

In collaboration with the **Human Robot Interaction Lab** at the **European Space Agency**, our most advanced haptic device, the **sigma.7**, was launched to the International Space Station (ISS). The haptic device is part of the ESA METERON experiment which explores new ways to operate robots from space using the sense of touch.

Designed and manufactured in Switzerland, the **sigma.7** is the first 3D haptic device certified for use in space.

Force Dimension
Switzerland

www.forcedimension.com
info@forcedimension.com

IEEE ROBOTICS & AUTOMATION MAGAZINE

Vol. 27, No. 2 JUNE 2020
ISSN 1070-9932
http://www.ieee-ras.org/publications/ram

FEATURES

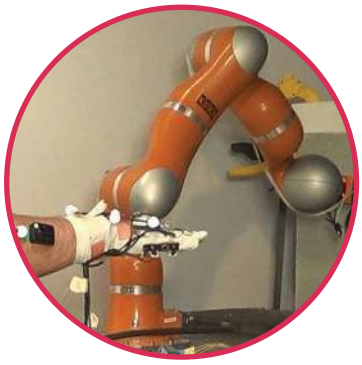
ON THE COVER
This special issue focuses on deep learning and machine learning approaches that have been validated on real-world robots, scenarios, and automation problems.

Digital Object Identifier 10.1109/MRA.2020.2984467

WINNER 2013 APEX® AWARDS FOR PUBLICATION EXCELLENCE

If you like an article, click this icon to record your opinion. This capability is available for online Web browsers and offline PDF reading on a connected device.

FEATURES *(continued)*



COLUMNS & DEPARTMENTS

IEEE ROBOTICS & AUTOMATION MAGAZINE

A Publication of the IEEE ROBOTICS AND AUTOMATION SOCIETY
Vol. 27, No. 2 June 2020 ISSN 1070-9932 http://www.ieee-ras.org/publications/ram

Digital Object Identifier 10.1109/MRA.2020.2984468

IEEE prohibits discrimination, harassment, and bullying. For more information, visit http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html.

IEEE Robotics & Automation Magazine (ISSN 1070-9932) (IRAMEB) is published quarterly by the Institute of Electrical and Electronics Engineers, Inc. Headquarters: 3 Park Avenue, 17th Floor, New York, NY 10016-5997 USA, Telephone: +1 212 419 7900. Responsibility for the content rests upon the authors and not upon the IEEE, the Society or its members. IEEE Service Center (for orders, subscriptions, address changes): 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855 USA. Telephone: +1 732 981 0060. Individual copies: IEEE Members US$20.00 (first copy only), non-Members US$135 per copy. Subscription rates: Annual subscription rates included in IEEE Robotics and Automation Society member dues. Subscription rates available on request.

Periodicals postage paid at New York and additional mailing offices. Postmaster: Send address changes to IEEE Robotics & Automation Magazine, IEEE, 445 Hoes Lane, Piscataway, NJ 08854 USA. Canadian GST #125634188 PRINTED IN THE U.S.A.

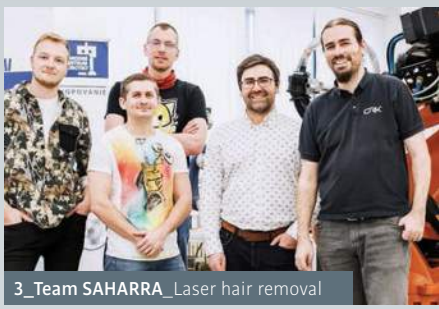
KUKA

Say Hello_to the next robotic innovator

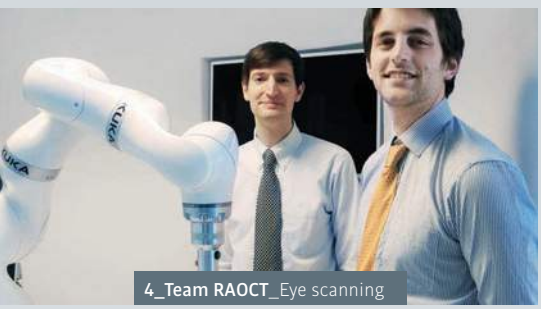
Innovators and entrepreneurs from around the globe were asked to submit their applications in response to the Medical Robotics Challenge of the KUKA Innovation Award 2020. Five teams made it to the finals and will be presenting their projects live at MEDICA – where the world of medical technology meets. Already in its 7th year, the award comes with much prestige and a 20,000 euro prize. Meet our five finalists!

1_Team SpheriObot_Bone cutting at the hip

2_Team HIFUSK_Focused ultrasound



3_Team SAHARRA_Laser hair removal



4_Team RAOCT_Eye scanning

5_Team CONEEbot_Haptic needle placement

MEDICA

Visit us at MEDICA from 16–19 November 2020, and watch the finals live at the KUKA booth.

1_Team SpheriObot, Shanghai Jiaotong University Affiliated Sixth People's Hospital, China: "We are developing a system that can make spherical cuts at the hip. Our method effectively reduces the interference on the bone structure to allow for better alignment of the bone segments and significantly reduce surgical trauma."

2_Team HIFUSK, The BioRobotics Institute, Scuola Superiore Sant'Anna, Italy: "We are developing a robotic platform that can change the way of treating cancer by enabling a totally non-invasive (no incision) surgical technique based on focused ultrasound."

3_Team SAHARRA, National Centre of Robotics, Slovakia: "Our project is focused on the development of a robotic system which will increase accuracy and speed, and will allow relief from monotonous work in permanent laser hair removal procedure in aesthetics and dermatology."

4_Team RAOCT, Duke University, USA: "Our robotically-aligned eye scanner promises to enhance the accessibility and quality of eye examinations in non-specialist settings by using a combination of robotic and optical tracking without needing the usual chinrests or forehead braces for stabilization."

5_Team CONEEbot, Institute of Medical Technology, Hamburg University of Technology, Germany: "Our objective is precise and safe needle placement for minimally invasive procedures. We equip the LBR Med with smart sensors embedded in a needle for collaborative robotic needle driving to put the physician's finger virtually at the tip of the needle."

KUKA Innovation Award 2020

www.kuka.com/innovationaward

Robotics and Artificial Intelligence

By Bram Vanderborght

One hundred years ago, the Czech writer Karel Čapek gave us the word "robot," which first appeared in his play *R.U.R.* in 1920 (although, later, he identified his brother Josef as the actual source of the term). Long before science fiction became recognized as a separate genre, Čapek wrote plays and novels in which he discussed ethical aspects of industrial inventions and processes, including mass production and robots. His robots were not exactly robots as we think of them today; they were living flesh-and-blood creatures rather than assemblages of mechanical components and electronics. Last year, I was invited for a debate following a performance of the famous play. A century after its creation, the theme of robots, in whatever form, is more topical than ever. While man increasingly plays God, his creation threatens to take over more and more. *R.U.R.* is as visionary as it is relevant, as tragic as it is comical, and as unlikely as it is frightening.

***R.U.R.* is as visionary as it is relevant, as tragic as it is comical, and as unlikely as it is frightening.**

Currently, the media talk about an explosion of major breakthroughs in the field of artificial intelligence (AI). In 1956, American computer scientist John McCarthy organized the Dartmouth Conference, at which the term *artificial intelligence* was adopted. But ideas that led to the concept of an electronic brain came much earlier, with, e.g., Claude Shannon's work on digital signals or Alan Turing's theory of computation. Despite well-funded global efforts early on, governments and corporations began to lose faith in AI. Computers were not well developed enough to process such a large magnitude of data. The so-called AI winter happened from the mid-1970s to the mid-1990s, when computer scientists dealt with an acute shortage of funding for AI research.

The exponential gain in processing power, storage, and network capabilities offered companies such as Facebook, Amazon, Google, Baidu, and others opportunities to leverage AI to their huge commercial advantage for many (often invisible to the general public) applications in daily life. This did not go unnoticed by governments. In 2019, the White House launched the American AI Initiative, which articulates a comprehensive vision for U.S. leadership in the technology. China is spending growing sums on AI research and development. Europe is also increasing its investments and in February published a white paper aiming to foster a European ecosystem of excellence and trust in AI.

Among the general public, a robot is often regarded as being the same as AI and vice versa. We often define a robot as a physical machine with sensors and actuators that may or may not require intelligence to perform specific tasks, whereas AI is a program, so it doesn't need to be physical. But developments in robotics bring additional perspectives to the integration of AI because robots' physical nature enables them to become increasingly capable of perceiving, acting upon, and shaping their environment and engaging in social interaction. On the other hand, AI enables robots to evolve and learn. No wonder that robotics and AI become more and more intertwined and that, for example, there are many sessions on deep learning at our IEEE Robotics and Automation Society conferences.

In its Coordinated Plan, the European Commission defines AI as "systems that display intelligent behavior by analyzing their environment and taking action—with some degree of autonomy—to achieve specific goals." That sounds like a possible definition of a robot, no? For the European Commission, robotics is regarded as part of AI, and AI is used as an umbrella term for deep learning, neural networks, prediction, machine translation, natural language processing, computer vision, artificial agents, and robotics.

Therefore, this special issue on deep learning and machine learning in robotics comes at a timely moment. We received a whopping 37 submissions. Special thanks go to the guest editors, Fabio Bonsignorio (Heron Robots, Genoa, Italy), David Hsu (National University of Singapore), Matthew

Digital Object Identifier 10.1109/MRA.2020.2984469
Date of current version: 10 June 2020

2020
NAGAMORI AWARDS
永守賞

Winners of the 6th Nagamori Awards Selected

Since motors appeared in the early 19th century, they have been used in all types of electrical appliances and are now an indispensable part of our daily lives. Today, a huge number of motors are used in a wide range of applications, and it is claimed motors account for more than 55% of the world's power consumption.

Therefore, motor research is extremely important if we are to maintain our affluent lives while also perpetually conserving the global environment.

We created these Nagamori Awards to bring vitality to technological research of motors and related fields, such as generators and actuators, and also to support the researchers and development engineers who strive each day to fulfill their dreams.

The 6th Nagamori Awards

Nagamori Foundation decided six winners of the 6th Nagamori Awards on May 26th. From the six, the Grand Nagamori Award Winner will be selected. Each Nagamori Awards winner receives 2 Million JPY and the Grand Nagamori Award winner 5 Million JPY.

For more information, please visit: http://www.nagamori-f.org/en

The 6 Award Winners are:

S. Nagamori

Chris Gerada
Professor of Electrical Machines and Associate Pro-Vice-Chancellor,
Department of Electrical and Electronic Engineering,
Faculty of Engineering, University of Nottingham
For contributions to advancements in high performance electrical machines and their industrial application and uptake

Yasuhisa Hirata
Professor, Department of Robotics, Graduate School of Engineering,
Tohoku University
Development of motion control technology for passive robot using servo brake actuators for high safety and low power consumption

Alireza Khaligh
Professor, Department of Electrical and Computer Engineering &
The Institute for Systems Research,
University of Maryland at College Park
Pioneering research and development on design and control of high-efficiency and high-power-density electric-motor-integrated wide bandgap power electronics

Shihua Li
Vice Dean, Professor, School of Automation, Southeast University
For contributions to the nonlinear modeling, analysis and multi-disturbance rejection control solution for precise motion control systems

Annette Muetze
Professor, Vice Dean, Head of Institute,
Electric Drives and Machines Institute,
Department of Electric Engineering and Information Technology,
Graz University of Technology
Increasing the reliability, efficiency, and utilization of variable speed drive systems

Jin Wang
Professor, Center for High Performance Power Electronics,
The Ohio State University
Leading-edge research and development of wide bandgap power device based electric machine drives

Contact information: Nagamori Foundation
Address: 338 Kuzetonoshiro-cho, Minami-ku, Kyoto 601-8205, JAPAN
Tel: +81-75-935-7731 E-mail: n.awards@nidec.com

This special issue on deep learning and machine learning in robotics comes at a timely moment.

Johnson-Roberson (University of Michigan, Ann Arbor), and Jens Kober (Delft University of Technology, The Netherlands), for handling all the articles. This special issue focuses on approaches that have been validated on real-world robots, scenarios, and automation problems. Transferring AI algorithms from the digital realm to the real world brings additional challenges: data are abundant, but labeling is sparse and expensive; reinforcement learning can require more iterations than are feasible on real systems; and mistakes might be costly or generate unsafe behaviors. Therefore, safe learning with small amounts of data becomes paramount. Enjoy the issue!

Responding to a Pandemic

By Seth Hutchinson

As I write this, sitting in my study, where I now pass large blocks of time in the new work-from-home rhythm, the street below is strangely quiet for a warm, mid-April evening in Atlanta. Looking forward in time, it seems impossible to predict how things will be when this column goes to press in early June, a mere two months from now, yet well beyond any reliable prediction horizon. As an engineer, I know that prediction is a difficult game. Good prediction relies on good models, accurate characterizations of uncertainty, and informative observations of state as things evolve, none of which is readily available with respect to the COVID-19 pandemic that now dictates lifestyle for much of the world. And yet, even in the absence of reliable forecasts, decisions must be made—in the case of the IEEE Robotics and Automation Society, these include determinations about conferences (whether to cancel, reschedule, or move to a virtual format), financial matters, new initiatives, and ongoing programs.

Happily, none of these decisions rest with a single individual. Each conference has a team of organizers, chosen based on their expertise and experience. Our Society's administrative and executive committees are populated by wise, rational leaders in our community, supported by an excellent group of full-time IEEE staff. In addition, the IEEE has a team specifically trained to deal with emergency situations who are already working with us to plan a way forward in these times of uncertainty.

However, even with this outstanding team, optimal outcomes cannot be guaranteed. We, as engineers and scientists, know that under significant uncertainty, optimality in the expected sense is the best one can hope for, and this provides little comfort when dealing with specific outcomes. It is certainly possible that some of the decisions being made now will, in retrospect, seem too conservative or too optimistic, cavalier, or short-sighted. This cannot be avoided. However, I can say with absolute confidence that the people making these decisions are acting in good faith, taking their responsibilities seriously, and considering all of the available data, costs, and benefits as well as the well-being of our Society—in short, that they are stepping up to this challenge, accepting responsibility, and doing their best in service to the research community. When this crisis has passed, we will owe these folks a debt of thanks for their thoughtfulness, their energy, their investment of time, and their commitment.

Some of the more consequential decisions are already known. It was clear some weeks ago that ICRA 2020 could not be held as an in-person meeting in Paris during the first week of June. It has now been decided that the 2020 conference will be a virtual meeting, although, as I write this, it is still not certain when the meeting will be held. Likewise, the IEEE Haptics Symposium, RoboSoft, and CASE have all decided to move to fully virtual events. Other conference organizers are now in the process of rescheduling their in-person meetings in the hope that a sufficient delay will give the world time to return to something more normal. These include ISMR, BioRob, COINS, and ICMA.

In all cases, these decisions have been driven by the question of how best to serve our membership. Safety concerns have been preeminent. The quality of the conference experience has been the driving criterion for optimality. Financial consequences to the Society have been, at most, a secondary concern. It is quite possible that our Society will lose money this year; that is not surprising in a time of crisis, and years of careful budgeting have left us well prepared for such an event.

It is certainly possible that some of the decisions being made now will, in retrospect, seem too conservative or too optimistic, cavalier, or short-sighted.

However, as important as all of this may be for our Society, in the larger scheme of today's reality, how roboticists conduct their meetings is not overwhelmingly important. That our community is inconvenienced by travel

Digital Object Identifier 10.1109/MRA.2020.2987716
Date of current version: 10 June 2020

restrictions, might suffer financial losses, could endure research setbacks, or might need to rethink priorities for the coming year or two—none of this is likely to garner much sympathy from those whose livelihoods, and even lives, have been jeopardized in the face of this pandemic. As researchers, this is a time to elevate our thoughts and aspirations. In the robotics press, there have recently been a number of articles and editorials outlining the possible role of robotics in fighting this pandemic. I suggest that this is a moment for us, as a research community, to take an even broader view: not only to ask how we can contribute in the face of the current crisis but also to systematically reevaluate our priorities in light of the now-evident fragility of humanity and its supporting ecosystem. This is a moment to do good, improve the human condition, and think beyond ourselves. Let us not miss it.

This is a moment to do good, improve the human condition, and think beyond ourselves.

Status of IEEE RAS Standardization Efforts

By Craig Schlenoff

The IEEE Robotics and Automation Society's (RAS's) Standards study and working groups are strong and growing stronger. As of the time of this month's publication, we have two published standards, six working groups actively developing future standards, and one study group about to transition to a working group. More information about these groups can be found at https://www.ieee-ras.org/industry-government/standards. This article provides a brief overview of the activities in each of these groups as well as information for how to become involved.

Robot Task Representation Working Group

The Robot Task Representation Working Group (P1872.1) is developing a standard that defines an ontology allowing for the representation of, reasoning about, and communication of task knowledge in the robotics and automation domain. This ontology includes key terms as well as their definitions, attributes, types, structures, properties, constraints, and relationships. It will address aspects of task decomposition including concurrency among tasks and hierarchical task decomposition. This standard will provide a way for planning systems to represent task knowledge and allow them to communicate with both planning systems and humans. The working group is currently focused on structuring knowledge frames that partition the concepts and definitions as well as the concepts and definitions themselves. To become involved, please contact Stephen Balakirsky at stephen.balakirsky@gtri.gatech.edu.

Standards for the Autonomous Robots Working Group

The Standards for the Autonomous Robots Working Group (P1872.2) is developing a standard that is a logical extension to the existing Core Ontologies for Robotics and Automation (CORA) standard. This effort extends the CORA ontology by defining additional ontologies appropriate for autonomous robots (AuRs) related to

- the core design patterns specific to AuRs in common robotics and automation subdomains
- the general ontological concepts and domain-specific axioms for AuRs
- the general use cases and/or case studies for AuRs.

The Working Group has defined the vocabulary for AuRs and is currently developing a case study to validate the proposed concepts. Three milestones have been identified for the implementation of two cooperating robots: 1) modeling the robots and devices in the environment to represent the relations, instances, and functionalities, 2) modeling the goal of the robots and the behaviors, and 3) including CORA in the implementation. For more information, please contact Howard Li at howard@unb.ca.

3D Map Data Representation Working Group

The 3D Map Data Representation (3D-MDR) Working Group has the goal of compiling a standard to allow robots to exchange 3D maps in a common format, which is different from the private internal representations robots use to build, maintain, and use their maps. The 3D-MDR Working Group has completed the definition of the types of 3D maps that will compose the standard. These include grid-based representations, like dense and sparse grids and octrees, and non-grid-based representations, like point clouds and polygonal meshes. The 3D-MDR Working Group is currently working on the definition of data formats in JSON that can be used to represent the aforementioned types of 3D maps. To become involved, please contact Francesco Amigoni at francesco.amigoni@polimi.it.

We have two published standards, six working groups actively developing future standards, and one study group about to transition to a working group.

Verification of Autonomous Systems–Guidelines Working Group

The Verification of Autonomous Systems–Guidelines Working Group

Digital Object Identifier 10.1109/MRA.2020.2986565
Date of current version: 10 June 2020

(P2817) is developing a standard that enables users to define a customized process for verification of their autonomous system. The standard documents best practices across all levels of abstraction within a given system. This standard also describes a conceptual model that assists in the development of new verification processes for autonomous systems and provides both integration guidance for developing a verification process and the techniques, methodologies, and tool types that support verification process development. Recently, the group has been compiling existing verification tools and techniques and establishing the details of the boundaries between the to-be-developed verification of autonomous systems guide and the existing standards and standards under development. This will ensure that they're pointing guideline users to the most relevant information and enable the group to focus on the aspects that are not covered in current standards efforts. To become involved, please contact Signe Redfield at signe@ieee.org.

Currently, the group is finalizing a top-level ontology that will provide coherence and interoperability among the concepts with other ontological standards.

Ontological Standard for Ethically Driven Robotics and Automation Systems Working Group

The Ontological Standard for Ethically Driven Robotics and Automation Systems Working Group (P7007) aims to create an ontological standard for ethically driven robotics and automation systems. Over the past years, the group has been intensively pursing this goal and actively discussing several topics that comprise ethics, ethical principles, norms, data privacy protection, transparency, accountability, responsibility, and so forth. The draft standard currently includes the following sections: Norms and Ethical Principles Ontology, Data Privacy and Protection Ontology, Transparency and Accountability, and Ethical Violation Management Ontology. Each section has its respective ontology expressed using Unified Modeling Language and concepts and relationships axiomatized using Common Logics. Currently, the group is finalizing a top-level ontology that will provide coherence and interoperability among the concepts with other ontological standards. The Working Group expects to submit its standard for balloting before July 2020. For more information, please contact Edson Prestes at prestes@inf.ufrgs.br.

Standard for Ethically Driven Nudging for Robotic, Intelligent, and Autonomous Systems Working Group

The Standard for Ethically Driven Nudging for Robotic, Intelligent, and Autonomous Systems Working Group (P7008) will provide guidance for developers and ethicists involved in the design of autonomous, intelligent systems that seek to "nudge" humans, that is, to influence the choices made by humans through subtle means. These systems may select the individuals they nudge, customize the means used to influence behavior, and/or learn from the effectiveness of the nudges implemented using machine learning. The goal of the standard is to align these systems with ethical design principles and embody an ethical framework grounded in respecting human autonomy, in particular, the human dignity of informed, noncoerced choice. The Working Group began in October 2018 and holds biweekly sessions by teleconference. An overall outline for the working draft has been completed, and three subgroups are now developing three sections: Goals and Limitations of Ethical Design, Architecture and Taxonomy of Nudging Systems, and Required Elements for a Design Method to Comply With the Standard. For more information, please contact Sean Dougherty at seand@ieee.org.

Robot Agility Standard Study Group

The Robot Agility Standard Study Group is working toward a future standard that promotes agility for industrial robot systems, enables industrial robots on shop floors to be more productive and autonomous, and requires less support from shop floor workers. Agility comprises automatic failure identification and recovery, automated planning to minimize up-front robot programming, and plug-and-play swappable robots that need minimal reprogramming. The Study Group has been working to engage interested parties to ensure that there is a sufficient breadth of industries represented. To become an official Working Group, the group has been preparing a project authorization request to be submitted this summer. For more information, please contact Anthony Downs at anthony.downs@nist.gov.

There is also an active effort to include undergraduate college students in these groups to pique their interest in robot standards. This involves providing limited travel support for students to attend major robotics conferences where standards meetings are being held. More information about this effort can be found at https://www.ieee-ras.org/students/student-news-and-announcements/1572-call-students-participating-in-ieee-robotic-standardization-efforts-spirse.

IROS 2019 Lifelong Robotic Vision: Object Recognition Challenge

By Heechul Bae, Eoin Brophy, Rosa H.M. Chan, Baoquan Chen, Fan Feng, Gabriele Graffieti, Vidit Goel, Xinyue Hao, Hyonyoung Han, Sathursan Kanagarajah, Somesh Kumar, Siew-Kei Lam, Tin Lun Lam, Chuanlin Lan, Qi Liu, Vincenzo Lomonaco, Liang Ma, Davide Maltoni, German I. Parisi, Lorenzo Pellegrini, Duvindu Piyasena, Shiliang Pu, Qi She, Debdoot Sheet, Soonyong Song, Youngsung Son, Zhengwei Wang, Tomas E. Ward, Jianwen Wu, Meiqing Wu, Di Xie, Yangsheng Xu, Lin Yang, Qihan Yang, Qiaoyong Zhong, and Liguang Zhou

Humans have a remarkable ability to learn continuously from the external environment and inner experience. One of the grand goals of robots is to build an artificial "lifelong learning" agent that can shape a cultivated understanding of the world from the current scene and previous knowledge via an autonomous lifelong development. It is challenging for the robot learning process to retain earlier knowledge when robots encounter new tasks or information. Recent advances in computer vision and deep-learning methods have been impressive due to large-scale data sets, such as ImageNet [1] and COCO [2]. However, robotic vision poses unique new challenges for applying visual algorithms developed from these computer vision data sets because they implicitly assume a fixed set of categories and time-invariant task distributions [3].

Semantic concepts change dynamically over time [4]–[6]. For bridging the gap between robotic vision and stationary computer vision fields, we utilize a real robot mounted with multiple high-resolution sensors [e.g., monocular/red- green-blue-depth (RGB-D) from RealSense D435i, dual fisheye images from RealSense T265, and lidar; see Figure 1] to actively collect the data from the real-world objects in several kinds of typical scenarios, such as homes, offices, campuses, and malls.

Digital Object Identifier 10.1109/MRA.2020.2987186
Date of current version: 10 June 2020

Lifelong learning approaches can be divided into

1) regularization methods, e.g., Learning without Forgetting (LwF) [7], elastic weight consolidation (EWC) [8], and synaptic intelligence (SI) [9]
2) network expansion methods, e.g., context-dependent gating [10] and Dynamic Expandable Network [11]
3) rehearsal approaches with a sampling replay or generative mechanism to fit distribution from prior tasks [12]–[14], e.g., incremental classifier and representation learning [15], Deep Generative Replay (DGR) [16], and DGR with dual memory [17] and feedback [18].

This report summarizes the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2019 Lifelong Robotic Vision Competition (Lifelong Object Recognition Challenge) with the data set, rules, methods, and results from the top eight finalists (of over 150 teams) (Figure 2). Individual reports, data set information, rules, and released source codes can be found at the project home page [19].

Challenge Data Set and Rules

This challenge aimed to explore how to leverage the knowledge summarized from previous tasks for learning a new task efficiently as well as how previously learned tasks could be efficiently memorized in lifelong robotic vision. The goal of this competition was to test a model's capability to continuously learn objects

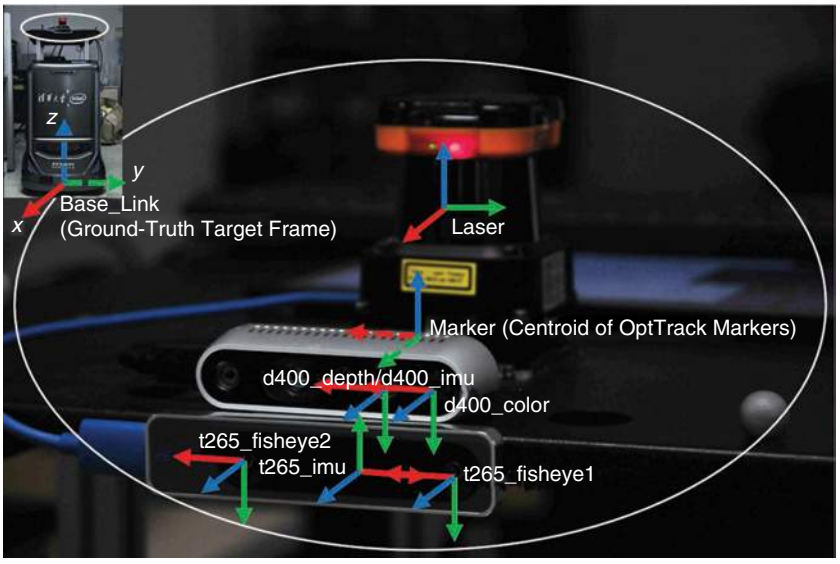


Figure 1. The OpenLORIS robotic platform (left) mounted with multiple sensors (right). In the OpenLORIS-Object data set, the RGB-D data are collected from the depth camera.

in a service robot scenario. Over 150 registered participants representing eight teams competed in the final testing phase. The work paved the way for robots to behave like humans in terms of knowledge transfer, association, and combination capabilities.

Challenge Data Set

The data set Lifelong Robotic Vision (OpenLORIS)–Object Recognition (OpenLORIS–Object) is designed to drive lifelong learning research and potential applications in the robotic vision domain, with everyday objects that exist in home, office, campus, and mall scenarios. The data set explicitly quantifies the variants of illumination, object occlusion, object size, camera-object distance/angles, and clutter information. The IROS 2019 competition organizers provided the first version of the OpenLORIS–Object data set for the participants. Note that our data set has been updated with twice the size in content available at the project home page [20], including data set visualization, download instructions, and more benchmarks on state-of-the-art lifelong learning methods [21].

The competition data set is a collection of 69 instances, including 19 categories of daily necessities objects under seven scenes (see Table 1). For each instance, a 17-s video (at 30 frames per second) was recorded with a depth camera delivering 260 distinguishable chosen RGB-D frames. Four environmental factors, each with three level changes, are considered explicitly (Table 1). The data were divided into 12 sequential tasks by randomly sampling from different factors and levels. The organizers also provided a more challenging bonus test set that was recorded under different context backgrounds with some deformation and extreme view angles.

Challenge Rules

Rules are designed to quantify the learning capability of the robotic vision system when faced with the objects appearing in the dynamic environments. Different from a standard computer vision challenge, not only was the overall accuracy on all tasks evaluated; the model efficiency, including model size, memory cost, and replay size (the number of old task samples used for learning new tasks; smaller is better), was also considered (Table 2). Meanwhile, instead of directly asking the participants to submit the prediction results on the test data set as in standard deep learning challenges [1], [2], the organizers received either source or binary codes to evaluate their whole lifelong learning process to make a fair comparison. The finalists' methods were tested by the organizers on an Intel Core i9 CPU and a Nvidia RTX 1080 Ti graphics processing unit.

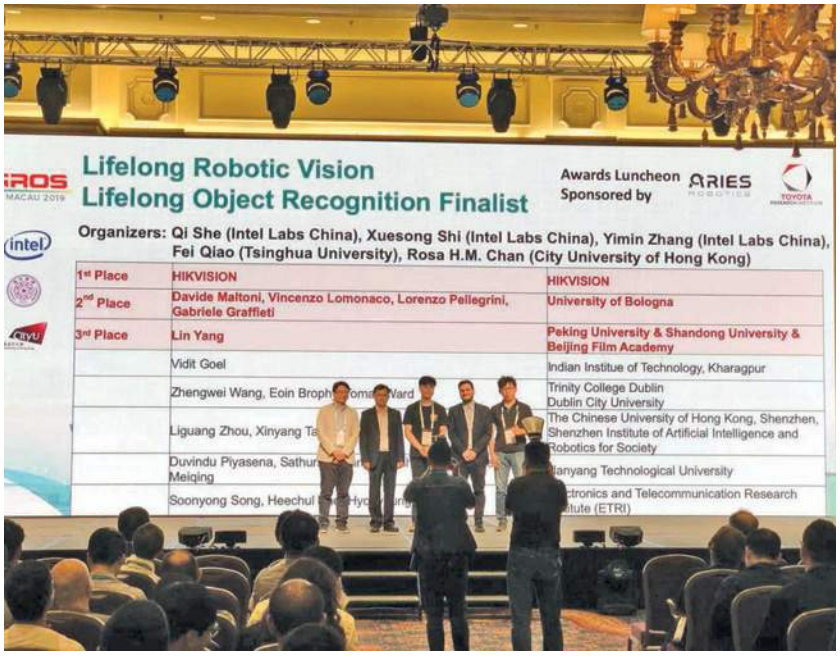


Figure 2. The Lifelong Robotic Vision Challenge finalists at IROS 2019.

Challenge Methods and Results

The finalists and their results are summarized in Table 3, with the top result(s) in each category designated in bold. Details such as the report, slide, and poster of each solution can be found on the project home page [19]. With excellent participants and the solutions they presented, we anticipate that the resulting solutions can help robots perform well under dynamic environments.

HIK_LIG Team (Champion)

- *Title*: Dynamic Neural Network for Incremental Learning
- *Members*: Liang Ma, Jianwen Wu, Qiaoyong Zhong, Di Xie, and Shiliang Pu
- *Affiliation*: Hikvision Research Institute, Hangzhou, China.

Table 1. The details for each of the three levels of four real-life robotic vision challenges.

Level	Illumination	Occlusion (%)	Object Pixel Size	Clutter	Context	Classes	Instances
1	Strong	0	> 200 × 200	Simple	Home/office/ campus/mall	19	69
2	Normal	25	30 × 30 – 200 × 200	Normal			
3	Weak	50	< 30 × 30	Complex			

Table 2. The metrics and grading criteria.

Metric	Accuracy	Model Size	Inference Time	Replay Size	Oral Presentation	Accuracy on the Bonus Data Set
Weight	50%	8%	8%	8%	10%	16%

Table 3. The IROS 2019 Lifelong Robotic Vision Challenge final results.

Teams	Final Accuracy (%)	Model Size (MB)	Inference Time (s)	Replay Size (Number of Samples)	Bonus-Set Accuracy (%)
HIK_ILG	96.86	16.3	25.42	**0**	**21.86**
Unibo	97.68	**5.9**	**22.41**	1,500	8.5
Guinness	72.9	9.4	346	**0**	10.96
Neverforget	92.93	342.9	467.1	**0**	1.52
SDU_BFA_PKU	**99.56**	171.4	2,444	28,500	19.54
Vidit98	96.16	9.4	112.2	1,300	1.39
HYDRA-DI-ETRI	10.42	13.4	1,323	21,312	7.1
NTU_LL	93.56	467.1	4,213	**0**	2.1

- *Method*: The team developed a dynamic neural network comprising two parts: dynamic network expansion for data across dissimilar domains and knowledge distillation for data in similar domains [Figure 3(a)]. The domain similarity was determined by the accuracy of the previous model before training on the current task.

Unibo Team (First Runners Up)

- *Title*: Efficient Continual Learning with Latent Rehearsal
- *Members*: Gabriele Graffieti, Lorenzo Pellegrini, Vincenzo Lomonaco, and Davide Maltoni
- *Affiliation*: University of Bologna, Italy
- *Method*: The team proposed a new lifelong learning approach based on latent rehearsal, namely, the replay of latent neural network activation instead of raw images at the input level [see the architecture and corresponding Android application in Figure 3(b)]. The algorithm can be deployed on the edge with low latency. Details can be found in [22].

Guinness Team

- *Title*: Learning Without Forgetting Approaches for Lifelong Robotic Vision

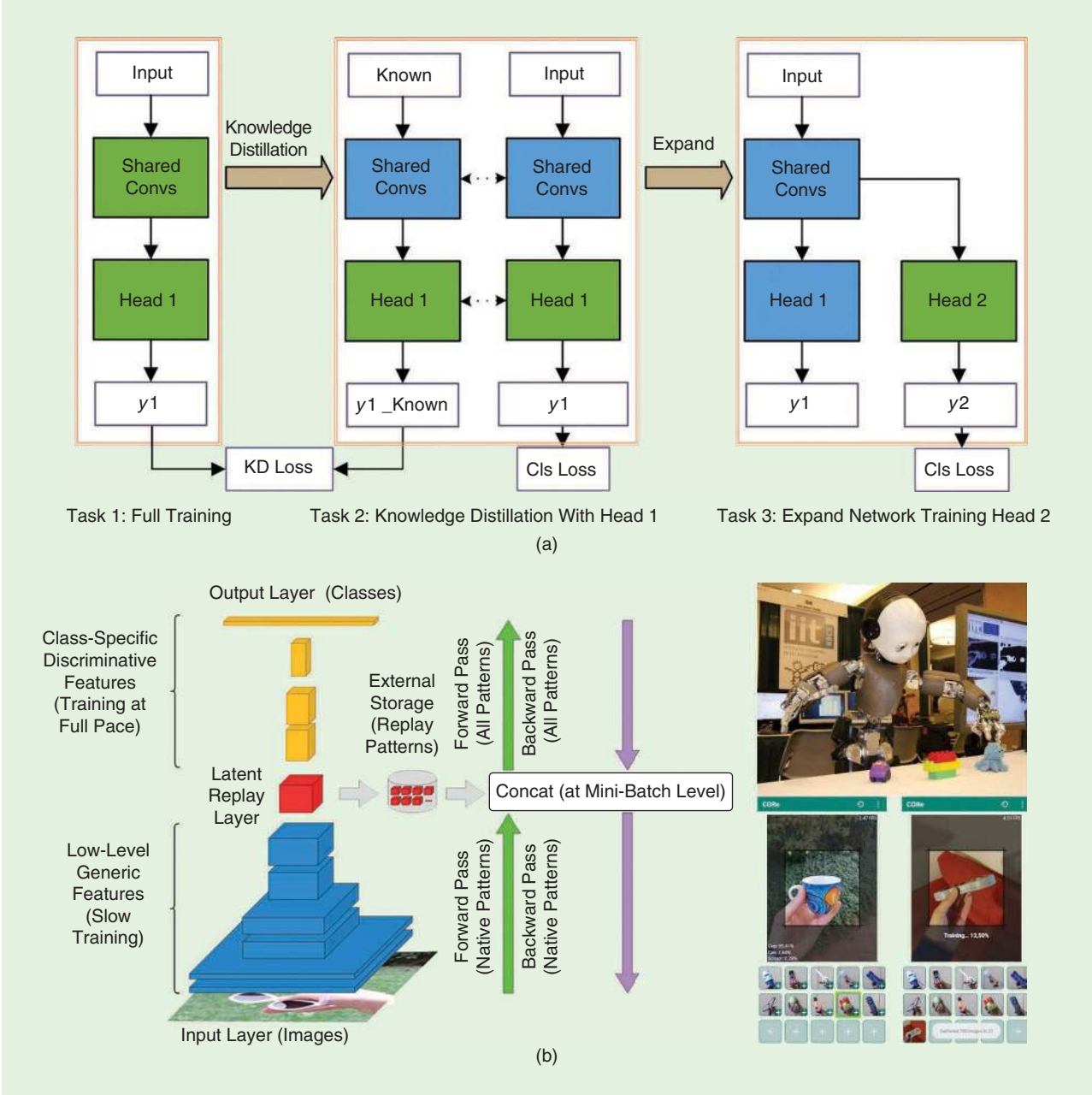


Figure 3. The challenge solutions. (a) HIK_LIG Team: dynamic network expansion for data across dissimilar domains and knowledge distillation for data in similar domains. (b) Unibo Team: replay of latent neural network activation instead of raw images at the input level (architecture and corresponding Android application). Cls: class; Convs: convolutions; KD: knowledge distillation. [(a) Source: HIK_LIG Team; used with permission; (b) source: Unibo Team; used with permission.]

- *Members*: Zhengwei Wang, Eoin Brophy, and Tomás E. Ward
- *Affiliations*: Wang: V-SENSE, School of Computer Science and Statistics, Trinity College, Dublin, Ireland; Brophy and Ward: Insight Center for Data Analytics, School of Computing, Dublin City University, Ireland
- *Method*: The core backend of the method was LwF [7]. There was no replay of previous task images in this structure.

Neverforget Team

- *Title*: A Small Step to Remember: Study of Single Model Versus Dynamic Model
- *Members*: Liguang Zhou, Tin Lun Lam, and Yangsheng Xu
- *Affiliation*: The Chinese University of Hong Kong, Shenzhen, China, and Shenzhen Institute of Artificial Intelligence and Robotics for Society, China
- *Method*: This approach was based on EWC [8] without a replay mechanism. The team also found the fact that the estimation of the Fisher information matrix might be biasedly estimated.

SDU_BFA_PKU Team

- *Title*: SDKD: Saliency Detection with Knowledge Distillation
- *Members*: Lin Yang and Baoquan Chen
- *Affiliation*: Peking University, Beijing, China; Shandong University, Qingdao, China; and Beijing Film Academy, Beijing, China
- *Method*: The approach disentangled this problem with two aspects: background removal problem and classification problem. The entrant used saliency maps to implement background removal and knowledge distillation to address catastrophic forgetting.

Vidit98 Team

- *Title*: Intelligent Replay Sampling for Lifelong Object Recognition
- *Members*: Vidit Goel, Debdoot Sheet, and Somesh Kumar
- *Affiliation*: Indian Institute of Technology, Kharagpur, India
- *Method*: This approach sampled validation data from the buffer and used them as replay data. It intelligently created the replay memory for a task. The replay memory was an efficient representation of previous task data, whose information was lost and sampled from the validation set.

HYDRA-DI-ETRI Team

- *Title*: Selective Feature Learning with Filtering Out Noisy Objects in Background Images
- *Members*: Soonyong Song, Heechul Bae, Hyonyoung Han, and Youngsung Son
- *Affiliation*: Electronics and Telecommunications Research Institute, Korea
- *Method*: The team proposed a selective feature learning method to eliminate irrelevant objects in target images. A single-shot multibox detection (SSD) algorithm selected the desired objects [23]. The SSD algorithm alleviated performance degradation by noisy objects. Then SSD weights were trained with annotated images in task 1 and the refined data were fed into a classification module.

NTU_LL Team

- *Title*: Lifelong Learning with Regularization and Data Augmentation
- *Members*: Duvindu Piyasena, Sathursan Kanagarajah, Siew-Kei Lam, and Meiqing Wu
- *Affiliation*: Nanyang Technological University, Singapore
- *Method*: The team utilized a combination of an SI-based regularization method [9] and data augmentation for each task.

Acknowledgments

This work was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China, under project CityU

11215618. The authors would like to thank Hong Pong Ho from the Intel RealSense Team for the technical support of RealSense cameras for recording the high-quality RGB-D data sequences. The author list is in alphabetical order: R.H.M. Chan, F. Feng, X. Hao, C. Lan, Q. Liu, V. Lomonaco, G.I. Parisi, Q. She, and Q. Yang prepared the report. The other co-authors were competition finalists. The corresponding author is Qi She (qi.she@intel.com).

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[2] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. European Conf. Computer Vision (ECCV)*, 2014, pp. 740–755.

[3] F. Feng, R. H. M. Chan, X. Shi, Y. Zhang, and Q. She, "Challenges in task incremental learning for assistive robotics," *IEEE Access*, vol. 8, pp. 3434–3441, Nov. 25, 2019. doi: 10.1109/ACCESS.2019.2955480.

[4] Q. She and A. Wu, "Neural dynamics discovery via Gaussian process recurrent neural networks," in *Proc 35th Conf. Uncertainty Artificial Intelligence (UAI)*, 2019, p. 159.

[5] Q. She, and R.H.M. Chan, "Stochastic dynamical systems based latent structure discovery in high-dimensional time series," in *Proc. 2018 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 886–890. doi: 10.1109/ICASSP.2018.8461755.

[6] Q. She, Y. Gao, X. Kai, and R. H. M. Chan, "Reduced-rank linear dynamical systems," in *Proc. 32nd AAAI Conf. Artificial Intelligence (AAAI)*, 2018, pp. 4050–4057.

[7] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, 2017. doi: 10.1109/TPAMI.2017.2773081.

[8] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *Proc. Natl. Acad. Sci. (PNAS)*, vol. 114, no. 13, pp. 3521–3526, 2017. doi: 10.1073/pnas.1611835114.

[9] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. 34th Int. Conf. Machine Learning (ICML)*, 2017, pp. 3987–3995.

[10] N. Y. Masse, G. D. Grant, and D. J. Freedman, "Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization," *Proc. Nat. Academy Sci. (PNAS)*, vol. 115, no. 44, pp. 467–475, 2018. doi: 10.1073/pnas.1803839115.

[11] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, Lifelong learning with dynamically expandable networks. 2017. [Online]. Available: arXiv:1708.01547

[12] Z. Wang, Q. She, and T. E. Ward, Generative adversarial networks: A survey and taxonomy. 2019. [Online]. Available: arXiv:1906.01529

[13] Z. Wang, Q. She, A. F. Smeaton, T. E. Ward, and G. Healy, A Neuro-AI interface for evaluating generative adversarial networks. 2020. [Online]. Available: arXiv:2003.03193

[14] Z. Wang, Q. She, A. F. Smeaton, T. E. Ward, and G. Healy, Neuroscore: A brain-inspired evaluation metric for generative adversarial networks. 2019. [Online]. Available: arXiv: 1905.04243

[15] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2001–2010.

[16] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Proc. Advances Neural Information Processing Systems (NIPS)*, 2017, pp. 2990–2999.

[17] N. Kamra, U. Gupta, and Y. Liu, Deep generative dual memory network for continual learning. 2017. [Online]. Available: arXiv:1710.10368

[18] G. M. van de Ven and A. S. Tolias, Generative replay with feedback connections as a general strategy for continual learning. 2018. [Online]. Available: arXiv:1809.10635

[19] Q. She et al., "IROS 2019 Lifelong Robotic Vision Challenge—Lifelong object recognition report," Nov. 2019. Accessed on: Apr. 2020. [Online]. Available: https://arxiv.org/abs/2004.14774

[20] Q. She, "OpenLORIS-object dataset and Benchmark," Nov. 2019. Accessed on: Apr. 2020. [Online]. Available: https://lifelong-robotic-vision.github.io/dataset/Data_Object-Recognition

[21] Q. She et al., OpenLORIS-Object: A robotic vision dataset and benchmark for lifelong deep learning. 2020. [Online]. Available: arXiv:1911.06487v2

[22] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, Latent replay for real-time continual learning. 2019. [Online]. Available: arXiv:1912.01100v2

[23] W. Liu et al., "SSD: Single shot multibox detector," in *Proc. European Conf. Computer Vision (ECCV)*, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

Exoskeleton Developed by Greek Students Helps People With Disabilities

By Dora Fourou

The IEEE Region 8 Student and Young Professionals (SYP) Conference has as its primary goal to enhance and foster engineering sense and capabilities through different activities, exchanging experience and knowledge among young professionals and students from different parts of the world. The latest IEEE Region 8 SYP meeting in Porto, Portugal, was an opportunity for the members of two IEEE Student Branches in Greece to meet and get motivated of what's coming next.

Sharing the same passion, Iordanis Kostelidis, Despoina Markoglou, Antonis Alexos, Tilemachos Tsiapras, Kokozidis Pavlos, Dimitrios Dallas, and Stella Konstanti have been working together since 2018 sharing the same vision: to create an exoskeleton to help people with disabilities, offering them the opportunity of a better and easier life.

After working dedicated for more than one year in this demanding project, the team is ready for the next big step, to participate at the Cybathlon 2020, with their pilot Modestos Kapinas.

In *IEEE Region 8 Today* (R8T), we spoke with the team, and we share the interesting conversation we had, describing the birth of the idea, the project, and their next big step.

R8T: What is an exoskeleton, and how does it work?

A powered exoskeleton is a wearable mobile device, powered by a group of systems in a combination of technologies, that allows helping limp movement with enhanced strength and endurance. It usually consists of electric motors, control systems, and power management and is getting powered from a power source (battery).

Powered exoskeletons are used for medical, military, civilian, or industrial use. Their main objective is to give mechanical benefits and movement to the users. A power source (usually a battery) provides energy to the electric motors. The motors are controlled by a system that tells them what to do, then to do it and how to do it. Sensors are placed in various positions to take measurements and communicate with the main system for adjustments and alerts and inform the user of the stability of the device.

R8T: What differentiates the Hermes exoskeleton from others?

The Hermes exoskeleton is a wearable robotic exoskeleton that can be built from simple materials (3D-printed parts) and known components (e.g., Arduino), be modular (for every user, age, height, weight), and be maintained by anyone. The main purpose of our device is to help people with paraplegia to stand, walk on straight or bumpy ground, and ascend/descend stairs and more generally to facilitate their daily activities. It works four main exoskeleton motor systems, sponsored by Maxon, controlled by a Raspberry Pi (a low-cost, credit card-sized computer), and the system can sense the movements of the user with various sensors,

The Hermes Team consists of IEEE Student Members from Greece who share a message of innovation and dedication, developing a powered exoskeleton for people with disabilities. This article first appeared in a slightly different form in *IEEE Region 8 Today*.

controlled by an Arduino board (Arduino is a simple open source motherboard with built-in microcontroller and inputs/outputs, which can be programmed in the Wiring language). The control of the exoskeleton movement is achieved through a smaller controller associated with the right crutch, where all control buttons and the screen are. Custom-made two-layer boards needed to be designed for the necessary control circuits, by Altium Designer. Necessary simulations have been made using MathWorks sponsored software, MATLAB. In addition, through sensors and control systems we ensure maximum performance and security for the user. All of this, of course, is developed with the potential of upgrading both hardware (boards, motors), software (improvement of the operating system specifically designed by us for this application), and functional level (more features like auto balance).

The Hermes exoskeleton is unique, due to its design, simplicity, elegance, and modularity. It's customizable, which means it can be adapted to each person's physical specifications (height and weight). It is lightweight, durable, easy to use, and comfortable during its use. We achieve the best possible

Digital Object Identifier 10.1109/MRA.2020.2986566
Date of current version: 10 June 2020

The members of the HERMES Team (from left): Dimitrios Dallas, Stella Konstanti, Antonios Alexos, Iordanis Kostelidis, Modestos Kapinas, Pavlos Kokozidis, Despoina Markoglou, and Tilemachos Tsiapras.

The women members of HERMES Team: Stella Konstanti (left) and Despoina Markoglou.

weight-to-strength ratio through the extensive use of composite materials such as carbon fiber and specially selected alloys (such as Al-7075). With the sponsorship of Dassault Systems of SolidWorks Suite, we made every mechanical component and stress test necessary for the safety of the device. The key difference is the combination of all those traits, in one device.

R8T: How was this idea born? What motivated you?

The whole idea began in July 2018, during the R8 SYP 2018 Conference, in Porto, between the IEEE University of Thessaly Student Branch, Greece, and the IEEE RAS Student Chapter–International Hellenic University–Serres (formerly Technological Educational Institute of Central Macedonia), Greece. We realized that, as young student engineers, there is a need of synergies inside the ecosystem of the IEEE, between organizational units and, more specifically, Branches and Student Chapters. And out of nothing the idea struck. We saw in *IEEE Robotics and Automation Magazine* an article dedicated to the Cybathlon and that was it. From that moment, we decided to build an exoskeleton for paraplegic persons in order to participate in the competition.

R8T: How does the life of a paraplegic person change by using an exoskeleton?

The life of a disabled person with the help of an exoskeleton is significantly improved. Daily tasks such as ascending and descending a series of stairs become routine. It changes dramatically. The daily routine of paraplegic people is upgraded, so they can serve themselves. It has an impact on the exercise and health of the user, as blood circulation returns to more normal rhythms and the risk of thrombosis is reduced. In addition, pressure sores are minimized, as they are the main danger of paraplegic individuals. It also reintroduces the user to society, providing the ability to socialize and start a normal life. The ability to work, moreover, is feasible, as mobility issues are minimized.

R8T: Is it difficult for someone to wear the exoskeleton?

Not at all. Each exoskeleton has a specific way that it can be worn. To be more precise, with our exoskeleton, the start position of the device is at the sitting position, so that each user, with a wheelchair, can be transferred without any particular help to the appropriate position to fit the device to the user's needs. With Velcro straps, the user can fasten the device to the knees, thighs, buttocks, and chest for better fit and balance in movement. Also, there are cuffs for better stability and user security. Once it is fastened, the user is ready to boot the device through a modified crutch that is attached to the device.

R8T: Seeing young people working with such enthusiasm and dedication on a demanding project is very motivational. However, the team also includes women, and I want to ask them. What's your influence being an engineer?

Hermes Team women engineers: Being a female engineer in a male-dominant industry is very challenging.

(continued on page 176)

When Machine Learning Implies Intelligence

By Patrick van der Smagt

In the early 1990s, we thought neural networks would be commonplace soon and used everywhere. It took 30 years longer, but that point has now been reached. Neural networks can learn any function from complex and high-dimensional data sets, image or pattern recognition is solved, and probability theory serves as a sound mathematical basis for it all.

Neural network-based solutions to supervised learning have provided us with many applications related to image recognition, human–machine interaction, and so on, where large numbers of human-annotated data are available. Where these scale, they scale fast. This gives sound business models for IT companies with large user bases.

However, only very few types of applications can be tackled by human annotation. Europe is centered around manufacturing, to a large part covered by small and medium enterprises (70% of the gross domestic product in Europe versus 40% in the United States). For applications to scale, we need more than supervised learning.

We now have to look into fortifying research on more complex machine learning problems: unsupervised learning on small data sets, incorporating previous knowledge. If we ever want to get close to biological intelligence, these are key problems that must be tackled.

Digital Object Identifier 10.1109/MRA.2020.2986567
Date of current version: 10 June 2020

Beyond Frequentist

Why are neural networks so data hungry? An important reason is a mathematical one. At the core of a neural network loss function lies the maximum likelihood estimation principle, which assumes a specific probability density in the distribution of the data. For regression, the standard candidate is the normal distribution: following the central limit theorem, this assumption is reasonable as long as we have enough data. The Bayesian neural network solves this, where uncertainty is represented in the weights of the network. This approach allows for learning with very few data, but these networks are hard to train and do not scale like their frequentist counterparts. Practical applications are not near.

Incremental Learning

At least as serious a problem is incremental learning. Biological systems do not learn everything from scratch but combine the previously learned with new data. Technical approaches should be able to do the same and also incorporate handcrafted physical models into learning systems. How do we put model knowledge in a neural network? Possible solutions exist—e.g., in latent-variable time-series models—but are not commonplace, nor do they scale. This is a problem that must be solved before we can proceed to the next state of system intelligence.

Learning Without Oracle

Learning in biology is never supervised, needing large numbers of precise examples. Instead, we learn from combining observation, previous experience, and experimentation. How does this work?

A neural network solution to supervised learning has been available since the early 1980s, when the autoencoder was proposed. Recently, these models have proven to be useful as efficient latent-variable models that combine probabilistic approaches with neural networks. Time-series extensions have been proposed in the last few years, but scaling to real applications is not near.

A neural network solution to supervised learning has been available since the early 1980s, when the autoencoder was proposed.

It may be as in the 1990s: we have some first working approaches, but these have to scale and extend to wide applicability. Even if investment in machine learning may wane, if we continue focusing research on unsupervised Bayesian learning and latent-variable models, there's a good chance that, next time around, we'll have the resurrection covered.

Deep Learning and Machine Learning in Robotics

By Fabio Bonsignorio, David Hsu, Matthew Johnson-Roberson, and Jens Kober

Deep learning has gone through massive growth in recent years. In many fields—computer vision, speech recognition, machine translation, game playing, and others—deep learning has brought unprecedented progress and become the method of choice. Will the same happen in robotics and automation?

In a sense, it is already happening. Today, deep learning is often the most common keyword for work presented at major robotics conferences. At the same time, robots, as physical systems, pose unique challenges for deep learning in terms of sample efficiency and safety in real-world robot applications. With robots, data are abundant, but labels are sparse and expensive to acquire. Reinforcement learning in principle does not require data labeling but does require a significant number of iterations on real robots. Transferring the capabilities learned in simulation to real robots and collecting sufficient data for practical robot applications both present major challenges. Further, mistakes by robot learning systems are often much more costly than those by their counterparts in the virtual world. These mistakes may cause irreversible damage to robot hardware or, even worse, loss of human lives. Safety is thus paramount for robot learning systems. A related issue is interpretability, i.e., the ability for humans to understand the learning process. We need shared supervision and control approaches where humans have the option to integrate with machine learning systems and overcome the systems' cognitive limitations. This requires that humans understand the decision processes of the machine in sufficient detail.

This special issue of *IEEE Robotics and Automation Magazine* focuses on approaches that have been validated on real-world robots, scenarios, and automation problems. It features an exceptionally large number of scientific articles, 11 in total, in keeping with the recent massive growth in the field of robot learning. The articles were carefully selected by the guest editors in two rounds of reviews and provide a good account of the depth and breadth of the research in this area. The methods range from algorithms based on probability theory to deep learning. The whole spectrum of robotics—from manipulators to legged robots to drones to autonomous vehicles—is covered, with applications ranging from rehabilitation to cargo transportation. Additionally, this issue contains two other loosely connected regular articles.

"Movement Primitive Learning and Generalization" by You Zhou et al. considers the representation of movements capable of generalizing according to variations in the environment and task, rather than simply reproducing a demonstration. The focus of this article is the ability to represent behaviors that have multiple modes (i.e., different strategies) in a joint representation. The final evaluation is a throwing task where the robot decides whether to bounce the ball off a wall or not depending on the target's location. "Gaussians on Riemannian Manifolds" by Sylvain Calinon provides an overview of the use of Riemannian geometry in robotics. The article then illustrates the use of Gaussians on Riemannian manifolds for movement generation in more detail. This is important as a possible way to reduce the computational burden of learning strategies by considering the manifold structure of the data coming from robot motions. Similarly, the following article proposes a structured approach to deal with temporal features, another characteristic aspect of robotic systems. "Interactive Learning of Temporal Features for Control" by Rodrigo Pérez-Dattari et al. explores how robots can be taught new skills interactively by human teachers. The authors focus on tasks where the learning agent needs to be equipped with memory to succeed.

"Multifingered Grasp Planning via Inference in Deep Neural Networks" by Qingkai Lu et al. employs deep learning to predict grasp success, using a voxel representation of the object and the grasp pose as inputs. This model, jointly with a prior over-grasp pose, is then used to infer the grasp with the highest chance of success on unseen, novel objects. "Optimal Deep Learning for Robot Touch" by Nathan Lepora and John Lloyd shows how deep learning methods can be employed to estimate object poses based on input from optical tactile sensors. The authors also demonstrate how the method can then be employed for 3D object exploration.

Digital Object Identifier 10.1109/MRA.2020.2984470
Date of current version: 10 June 2020

"Machine Learning for Active Gravity Compensation in Robotics" by Axier Ugartemendia et al. improves the performance of mechatronic systems used in rehabilitation, bringing their capabilities closer to those of human physiotherapists. The article compares various methods for learning to compensate for gravity. "Decoding Motor Skills of Artificial Intelligence and Human Policies" by Kai Yuan et al. offers the opposite perspective. It shows how using machine learning to discover how a humanoid robot's push recovery strategies can be used to better understand and study human balance control. Those discovered strategies can then be employed for control design.

"Assured Runtime Monitoring and Planning" by Esen Yel et al. addresses one of the biggest open questions in robot learning: safety and verification. This article proposes to speed up online reachability analysis with a pretrained neural network where the neural network itself has been verified. The approach is demonstrated experimentally with quadrotors.

"Multifidelity Reinforcement Learning With Gaussian Processes" by Varun Suryan et al. shows how data collected on less realistic simulations can be used to minimize the amount of data that need to be collected on a real robot. The approach is validated with a navigation task on a Pioneer robot. "Unsupervised Pedestrian Pose Prediction" by Xiaoxiao Du et al. addresses a problem that is highly relevant for autonomous driving, i.e., not only detecting pedestrians but also predicting their future behavior. In particular, the authors present a method that does not require labeled pedestrian data for training, which has been a major bottleneck in the practical application of such methods. "Deep Learning-Based Localization and Perception Systems" by Zhe Liu et al. presents a system, based on deep learning, that can very accurately localize itself. This is an essential building block toward enabling autonomous navigation, in this case in the Hong Kong International Airport, without requiring modifications to the environment.

As a whole, the 11 articles in this special issue cover a broad range of challenges, ranging from crucial theoretical problems (manifold structure of data, temporal features, unsupervised learning, and multifidelity reinforcement learning) to key transversal needs (assurance and verification of autonomous operations, gravity compensation, and human skill decoding) while covering a wide and diverse set of applications and robotic morphological structures. In addition, the article on the CUDA parallel computing platform and programming model by Enric Cervera suggests some interesting technical insights that can be useful for the practical implementation of machine/deep learning systems. Further, the article by David St-Onge et al. on planetary exploration provides an example of an application where those approaches could have huge benefits in the future.

We hope that, while providing a rich picture of the state of the art in the application of machine and deep learning in intelligent robotics, this special issue will inspire further research from both the theoretical and application standpoints.

Fabio Bonsignorio is with Heron Robots, Genoa, Italy. Email: fabio.bonsignorio@heronrobots.com.

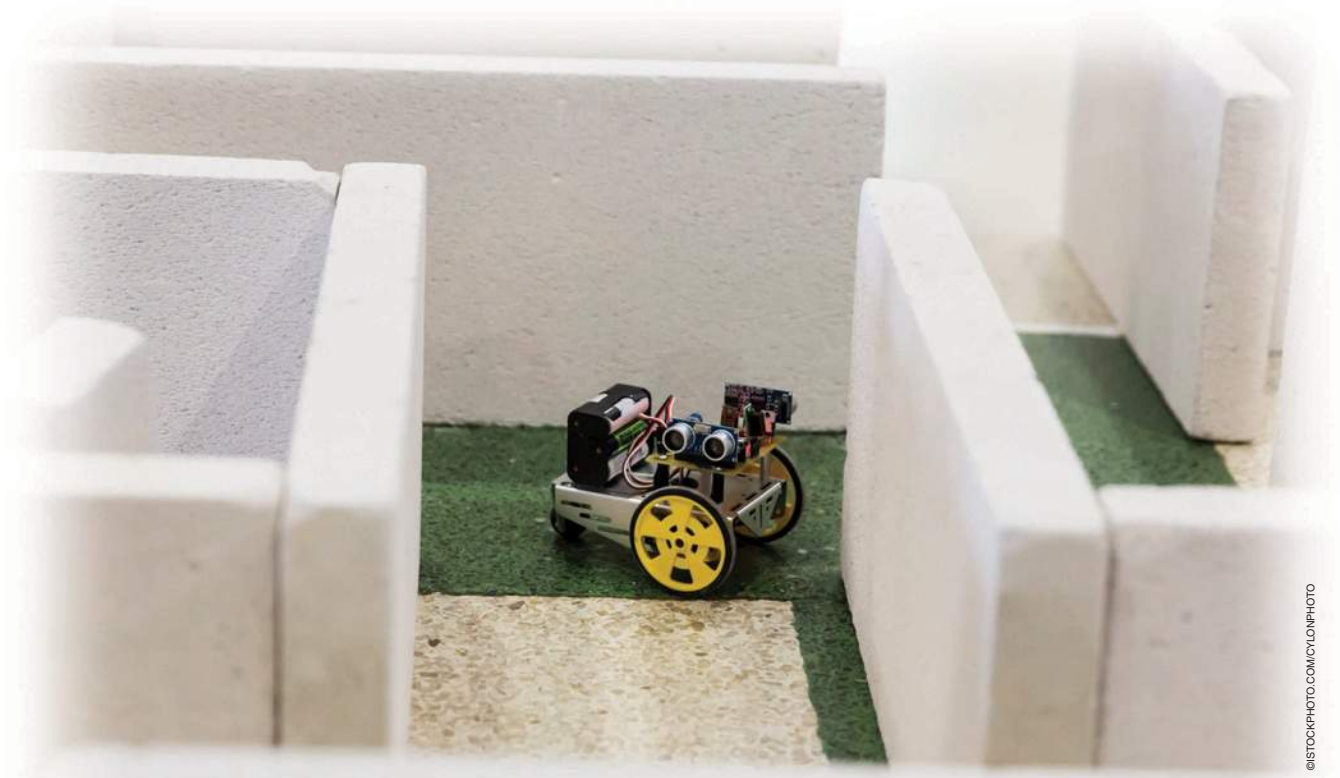
David Hsu is with the National University of Singapore. Email: dyhsu@comp.nus.edu.sg.

Matthew Johnson-Roberson is with the University of Michigan, Ann Arbor, United States. Email: mattjr@umich.edu.

Jens Kober is with the Delft University of Technology, The Netherlands. Email: j.kober@tudelft.nl.

Movement Primitive Learning and Generalization

Using Mixture Density Networks



©ISTOCKPHOTO.COM/CYLONPHOTO

By You Zhou, Jianfeng Gao, and Tamim Asfour

Representing robot skills as movement primitives (MPs) that can be learned from human demonstration and adapted to new tasks and situations is a promising approach toward intuitive robot programming. To allow such adaptation, mapping between task parameters and MP parameters is needed, and different approaches have been proposed in the literature to learn such mapping. In human demonstrations, however, multiple modes and models exist, and these should be taken into account when learning these mappings and generalized MP representations.

Here, a challenging problem is mode or model collapse. To solve this problem, we propose using a mixture density network (MDN) that takes task parameters as input and provides a Gaussian mixture model (GMM) of the MP parameters. To avoid mode and model collapse during MDN training, we introduce an entropy cost to achieve a more balanced association of demonstrations to GMM components. Since it is often easier to collect failed examples using an underfitted MDN model instead of additional human demonstrations, we introduce a failure cost to reduce the occurrence of failures in future executions. We evaluated our approach in simulation and real robot experiments and showed that the method outperforms previous approaches.

Digital Object Identifier 10.1109/MRA.2020.2980591
Date of current version: 6 April 2020

Challenges of Imitation Learning

Over the past decades, robotics researchers have developed different approaches that enable robots to learn from humans, imitate human behavior, and autonomously improve their

1070-9932/20©2020IEEE

movements. As a replacement for manual robot programming, learning from human demonstrations, also called *imitation learning*, has been proven a promising and powerful technique for intuitive robot programming [1]. Inspired by neuropsychological findings [2], we use MPs as essential building blocks for describing robot motions. In this context, the question of how a generalizable and compact representation of MPs can be learned from demonstrations and adapted to new situations and changing task parameters is an active research area in robotics.

Different MP representations have been proposed, such as dynamic MP (DMP) [3], probabilistic MP (ProMP) [4], and task-parameterized GMM (TP-GMM) [5]. These representations can adapt to task parameters in the space in which we define these primitives. For example, a DMP adapts to a new start or goal; a ProMP adapts to intermediate via points; and TP-GMM adapts to changes of predefined local frames, where we describe the demonstrated trajectories. However, for different tasks, the parameters could have different meanings and are not directly associated with spatial or temporal requirements for motion trajectories. For example, a robot throwing a ball to a target specified in the Cartesian space by executing a motion learned and represented in the robot's joint space should be able to adapt the learned motion to new targets. None of these approaches can adapt a learned MP to new targets specified in a different space.

For such generalization problems, methods have been proposed in which task-specific mapping between the task and MP parameters is learned through regression models, such as locally weighted regression (LWR) [6], [7], Gaussian process regression (GPR) [8], or deep neural networks [9]. However, such methods cannot deal with the problems of mode and model collapse. Given the task of reaching a target while avoiding an obstacle (see Figure 1), the goal can be reached using two different modes, i.e., by trajectories passing the obstacle from the left or right. Thus, learning an MP for such a task should take multiple modes in human demonstrations into account to increase the diversity of motions, which is beneficial in the case of changing task constraints.

Even though there are no multiple modes for each individual task parameter query in human demonstrations, there might exist multiple models for different types of task parameters. As shown in Figure 1(b), the goals on the left and right sides of the obstacle are reached separately by two types of trajectories, resulting from two different models, to allow the obstacle to be passed from the left or right.

In many applications, human demonstrations might use multiple modes and models at the same time. To avoid both mode and model collapse, we propose learning a mapping from the task parameter query $\boldsymbol{q}$ to a GMM of the MP parameters $\boldsymbol{w}$ with MDNs. Each mixture component of the GMM represents either a mode for one specific task parameter query or a model for multiple task parameter queries. However, training an MDN with only the negative log-likelihood (NLL) cost, as is usually done in the original MDN, might still lead to mode and model collapse, especially when the set of demonstrations is relatively small, as shown in Figure 1. To further reduce the occurrence of both collapses, we introduce an entropy cost function for training the MDN (entropy MDN).

Figure 1(a) shows the human demonstrations (dashed curves) for obstacle avoidance with an imbalanced distribution of examples associated with the two different modes as well as the results obtained with different approaches for a new task parameter query (the green dot). As can be seen, our approach (entropy MDN) generates multiple solutions (here, five solutions are shown in Figure 1) that cover both demonstration modes (blue and green) for the same task parameter query. We achieve this result with the entropy cost function, which ensures a more balanced probability associated with

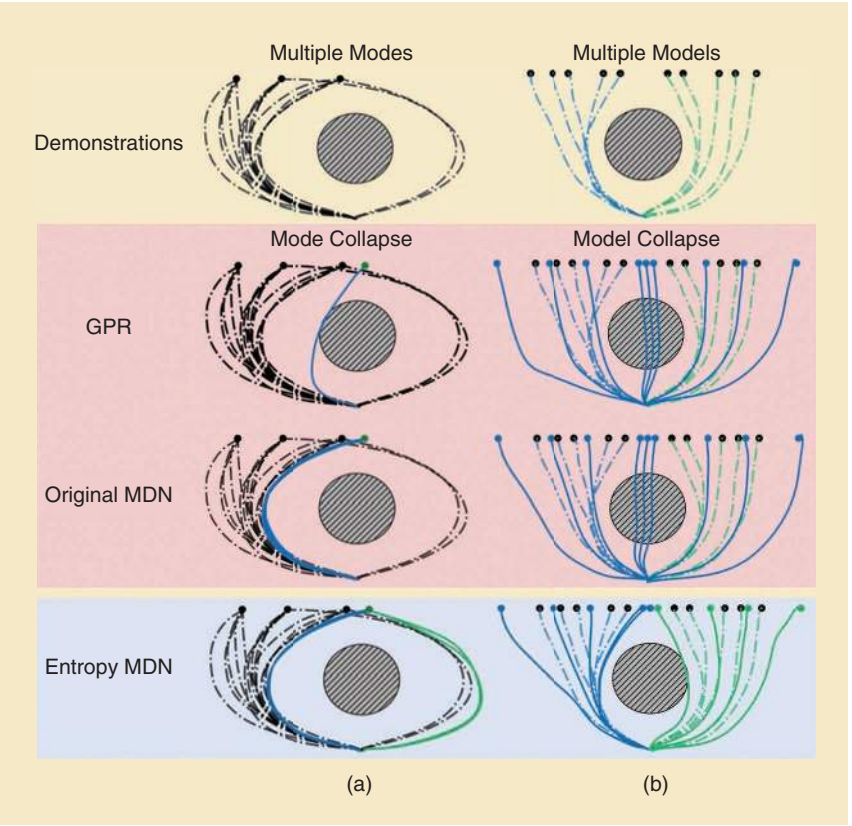


Figure 1. The (a) mode and (b) model collapse issues in the obstacle-avoidance problem are solved using the entropy MDNs. The dashed curves denote the demonstrations, and the solid curves show the generated trajectories.

the different modes. The original MDN generates multiple solutions, but these solutions (also five) reflect only one mode in the human demonstrations since it is associated with a high probability for the dominant mode. The GPR generates only one single solution that is close to the examples for the dominant mode.

Figure 1(b) shows the demonstrations (blue and green dashed curves) associated with two different models and the trajectories generated by different approaches (solid curves) for several task parameter queries (the colored dots). As shown at the bottom of Figure(b), our approach (entropy MDN) successfully learns the two models and generates the solution for the task parameter query correspondingly. We also achieve this result with the entropy cost function, which ensures that each model is associated with some human demonstrations. The original MDN has a higher chance of being stuck in the local minima of the NLL, where only one model is trained and overfitted for all task parameter queries. This fact leads to the failure of the task execution, i.e., collision with the obstacle, especially when the task parameter queries are on the boundary of two models. The GPR can learn only one model; thus, it suffers from the same problem as the original MDN.

For many tasks, it is often easier to collect failed samples with an underfitted MDN model instead of requesting additional human demonstrations. To further improve the MDN performance, we introduce a failure cost function that reduces the occurrence of the same failures for a given task parameter query.

Related Works

DMP is one of the popular approaches to representing robot motions. A DMP consists of a damped spring system and a nonlinear force term $f(x)$ such that

$$\begin{aligned} \tau\dot{v} &= K(g-y) - Dv + (g-y_0)f(x)x, \\ \tau\dot{y} &= v, \\ f(x) &= \frac{\sum_{i=1}^{N} \boldsymbol{\psi}_i(x)\boldsymbol{w}_i}{\sum_{i=1}^{N} \boldsymbol{\psi}_i(x)}, \end{aligned} \tag{1}$$

where v and y are the scaled velocity and position of the trajectory point, respectively; g, y_0, and τ, as the hyperparameters, represent the goal, start, and temporal factor separately; and x is the canonical variable, which goes from 1 to 0. The force term $f(x)$ is a linear regression model with N squared exponential kernels (SEKs) $\boldsymbol{\psi}_i(x) = \exp(-h_i(x-c_i)^2)$, where h_i, and c_i are fixed constants. $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)^T$ is a vector representing the DMP parameters.

DMP generalization means replacing $\boldsymbol{w}$ with a parameterized function $\omega(\boldsymbol{q})$. As mentioned previously, this function can be represented and learned with different approaches, such as GPR [8], LWR [6], [7], support-vector regression (SVR) [10], or deep neural networks [9]. To train those models, the training data set is collected as M pairs of task parameter queries and MP parameters $\{(\boldsymbol{q}_i, \boldsymbol{w}_i)\}_{i=1}^{M}$, where the ith MP parameter $\boldsymbol{w}_i$ is learned from the ith demonstration and corresponds to the ith task parameter $\boldsymbol{q}_i$.

Those methods require two parameterized functions, $f(x)$ and $\omega(\boldsymbol{q})$. The output of $\omega(\boldsymbol{q})$ is the parameter $\boldsymbol{w}$ of $f(x)$; hence, they are called *two-step methods*. In [11], Stulp et al. proposed combining two functions into one single function $f(x, \boldsymbol{q})$, which was learned with LWR or GPR and extended to the GMM in [12]. Since these methods use only one single function, they are called *one-step methods*, and they are more compact than the two-step methods.

The idea of the TP-GMM, suggested in [5], is to observe human demonstrations from multiple perspectives (local frames). A global GMM represents the trajectory points in a global frame and maximizes the likelihood of demonstrations from different perspectives. With the transformation of the local frames, the TP-GMM achieves a better extrapolation performance than other methods. In [13], Pignat and Calinon extended the TP-GMM and learned the sensory data together with the motion trajectories. As mentioned before, however, the task parameters considered by the TP-GMM are limited.

The ProMP uses a linear regression model with kernel functions $\boldsymbol{\psi}(\cdot)$ to directly represent the motion trajectory $y(x) = \boldsymbol{\psi}(x)^T\boldsymbol{w}$. In [4], Parachos et al. assumed that $\boldsymbol{w}$ follows a Gaussian distribution. In [14], the Gaussian distribution was extended to a GMM. For human–robot interactions, the ProMP parameter $(\boldsymbol{w} = \{\boldsymbol{w}_o, \boldsymbol{w}_c\})$ is separated to encode both human $(\boldsymbol{w}_o)$ and robot motions $(\boldsymbol{w}_c)$. With the conditional probability, the robot MP parameter $\boldsymbol{w}_c$ is inferred based on the human MP parameter $\boldsymbol{w}_o$. Both methods in [13] and [14] learn a generative model and use the conditional probability to infer unknown variables based on known ones.

In this article, we use a via-points MP (VMP), an MP formulation presented in our previous work [15] and described in the "The VMP" section. Instead of learning a generative model, we learn an MDN to directly map from a task parameter $\boldsymbol{q}$ to a GMM of the MP parameters $\boldsymbol{w}$. As a GPR or SVR for DMP, our technique belongs to the two-step methods.

MP Generalization

The VMP

We use the VMP (see [15]) to represent robot motions, which consists of an elementary trajectory h and shape modulation f as follows:

$$y(x) = h(x) + f(x) = g + x(y_0 - g) + \boldsymbol{\psi}(x)^T\boldsymbol{w}, \tag{2}$$

where x is the canonical variable, which goes from 1 to 0 with a linear-decay canonical system. The elementary trajectory takes the form of a polynomial; here, it is a first-order polynomial, i.e., a line connecting the start and the goal. By changing y_0 and g as the hyperparameters, the VMP adapts to the new start and goal. As the nonlinear force term in DMP, the shape modulation is a linear regression model with the SEKs $\boldsymbol{\psi}$. $\boldsymbol{w}$ is the parameter vector of the VMP.

In [15], we assumed that the parameter $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ follows a Gaussian distribution and that the maximum likelihood estimation of $\boldsymbol{\mu}$ is the empirical mean of all $\boldsymbol{w}$s, each of which corresponds to one demonstration and is obtained by solving a least-square problem.

Here, we extend the Gaussian distribution to a GMM and consider task parameter queries $\boldsymbol{q}$ as inputs:

$$\boldsymbol{w} = \omega(\boldsymbol{q}) \sim \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_k(\boldsymbol{q}), \Sigma_k(\boldsymbol{q}))^{z_k}, \tag{3}$$

where z_k is an element of a K-dimensional binary variable $\boldsymbol{z} = (z_1, z_2, \ldots, z_K)^T$, with only one particular element being equal to one and all other elements being zero. The probability of the kth element of $\boldsymbol{z}$ being equal to one is

$$p(z_k = 1) = \pi_k(\boldsymbol{q}).$$

The probability of the result $\boldsymbol{w}$ given the task parameter $\boldsymbol{q}$ is

$$p(\boldsymbol{w}|\boldsymbol{q}) = \sum_{k=1}^{K} \pi_k(\boldsymbol{q})\, \mathcal{N}(\boldsymbol{\mu}_k(\boldsymbol{q}), \boldsymbol{\Sigma}_k(\boldsymbol{q})). \tag{4}$$

We assume that the number of the mixture components K of the GMM is known and that $\{\pi_k(\cdot), \boldsymbol{\mu}_k(\cdot), \boldsymbol{\Sigma}_k(\cdot)\}_{k=1}^{K}$ are the functions to be learned.

The advantage of the VMP over DMP is its ability to adapt to intermediate via points by modifying the elementary trajectory $h(x)$ (see [15]). Apart from the via-points adaptation, extending the force term in a DMP to a GMM makes VMPs and DMPs exchangeable. Since the ProMP lacks the elementary trajectory and has no hyperparameters y_0 and g, the ProMP parameter function $\boldsymbol{w} = \omega(\boldsymbol{q})$ determines both the motion trajectory shape and its start and goal. In many tasks, the start and goal are a part of the task parameter queries. With a VMP, we reduce the learning complexity, because one requirement of the task, i.e., reaching a new goal, is directly satisfied by the hyperparameter g.

For example, in Figure 2, in contrast to a VMP, not all of the trajectories generated by the ProMP reach the goal if we use the entropy MDN and select the most probable parameter $\boldsymbol{w}$ from the output distribution. In some tasks, however, the goal is not a part of the task parameters and is necessary for the task execution. For these tasks, the VMP requires learning an additional mapping from the task parameters to the goals, whereas the ProMP provides a more compact solution. As shown in Figure 2, the entropy MDN suggested for VMP generalization also works for ProMP generalization.

In general, for MP generalization, M human demonstrations for different task parameter queries are collected. The purpose is to learn an MDN $[\omega(\cdot)]$ mapping from the task parameter query $\boldsymbol{q}$ to the parameter distribution of the MP parameter $\boldsymbol{w}$. For MDN, we have an assumption that the number of mixture components K of the output GMM is known.

The MDN

Using neural networks to learn the functions in (4) results in an MDN (see [16]). The mixing coefficients $\pi_k(\cdot)$, mean $\boldsymbol{\mu}(\cdot)$, and covariance $\boldsymbol{\Sigma}(\cdot)$ are represented by the network branches, as shown in Figure 3. These network branches

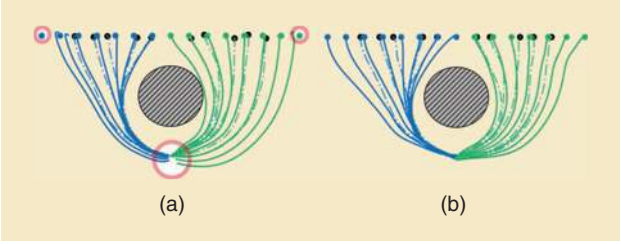


Figure 2. A comparison of (a) a ProMP and (b) a VMP. The dashed curves are demonstrations, and the solid curves are generated trajectories for different goals. The red circles indicate the starts and goals missed by the ProMP.

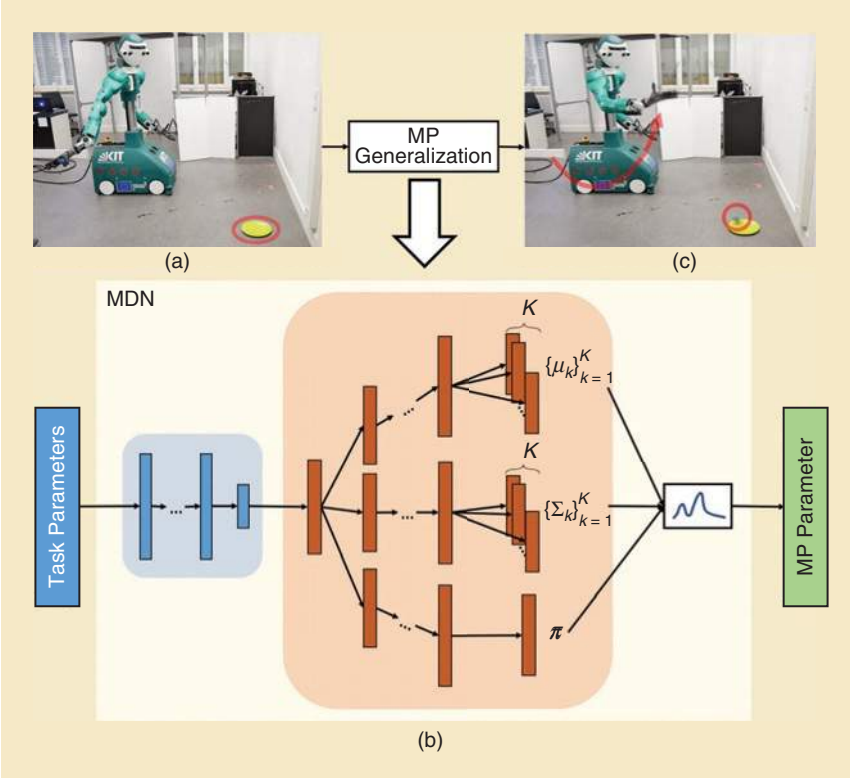


Figure 3. The MDN proposed for MP generalization. As an example, (a) with the target as the task parameter indicated by the red circle, the system (b) generates an MP parameter corresponding to (c) the motion of throwing the ball (see the red curves) on the target.

share a common network part (the blue box), which allows extracting common latent features.

We assume that the covariance is a diagonal matrix $\boldsymbol{\Sigma}$. According to [17], a GMM with a diagonal variance matrix approximates any given density function to arbitrary accuracy. The diagonal covariance output has the same dimension as the mean output. For K components, an MDN has K pairs of mean and variance outputs. Each pair corresponds to a mixture component of a GMM.

The dimension of the MP parameters, as the output of the MDN, determines the accuracy of the trajectory representation. With more SEKs, the MP represents the motion in a more accurate way. In the following experiments, we use 10 SEKs for each dimension. As an example, the output mean vector has 30 dimensions for a 3D motion trajectory, and there are a total of $K + K \times 30 \times 2$ values in the MDN output.

For one single demonstration, we calculate $\boldsymbol{w}$ by solving the least-square problem for (2). With M demonstrations, a training data set $\{(\boldsymbol{q}_i, \boldsymbol{w}_i)\}_{i=1}^{M}$ is collected. The NLL is written as

$$l_{\mathrm{NLL}}(\boldsymbol{\Theta}) = -\sum_{i=1}^{M} \log\left(\sum_{k=1}^{K} \pi_k(\boldsymbol{q}_i;\boldsymbol{\Theta})\, \mathcal{N}(\boldsymbol{w}_i;\boldsymbol{\mu}(\boldsymbol{q}_i;\boldsymbol{\Theta}), \boldsymbol{\Sigma}(\boldsymbol{q}_i;\boldsymbol{\Theta}))\right), \tag{5}$$

where $\boldsymbol{\Theta}$ is the parameters of the network, and

$$\mathcal{N}(\boldsymbol{w}_i;\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{d/2}\,|\boldsymbol{\Sigma}_i|^{1/2}} \cdot \exp\left(-\frac{1}{2}\sum_{j=1}^{d} \frac{(w_{i,j}-\mu_{i,j})^2}{\sigma_{i,j}^2}\right), \tag{6}$$

where d is the dimension of the output, $\boldsymbol{\mu}_i = \boldsymbol{\mu}(\boldsymbol{q}_i;\boldsymbol{\Theta})$, and $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}(\boldsymbol{q}_i;\boldsymbol{\Theta})$. A stochastic gradient descent method is used to minimize the NLL.

According to previous works [18], [19], training an MDN with the NLL suffers from mode collapse. In [18], to avoid the mode collapse and reduce learning complexity, Hjorth and Nabney suggest fixing the mean and variance on a grid defined in the output space and training only the model that outputs the mixing coefficients to reduce the NLL. If there are enough components regularly distributed in the output space, fixed means and variances do not reduce the representation capability. However, for large-dimensional outputs, such as MP parameters, this method leads to a sizeable intractable grid.

In [19], Makansi et al. used an MDN to predict the distribution of future car positions based on a current car position. To avoid mode collapse, they separated the MDN into two parts: a sampling and inference network. The sampling network takes the current car position as input and outputs a fixed number of hypotheses for future car positions. The authors trained the sampling network to place hypotheses to cover all of the observed outputs diversely. Based on these hypotheses, an inference network infers the parameters of the GMM. The MDN is a combination of the sampling and inference networks. The proposed method avoids mode collapse for the car-position prediction. However, for a high-dimensional output, such as MP parameters, the sampling network requires a large output dimension. Hence, it is difficult to apply both methods to our problem.

Entropy Costs for the MDN

Before introducing the entropy cost function, we first inspect the reasons that mode and model collapses occur when learning an MDN from demonstrations. Mode collapse occurs if the demonstrations associated with different modes for a task parameter query are imbalanced. For example, in Figure 1(a), only a small number of demonstrations take the path from the right side. By maximizing the likelihood of all demonstrations, the MDN tends to output a small mixing coefficient for the mixture component, which corresponds to the mode with fewer associated training data. In theory, it is correct to associate a small probability with an event that rarely happens in the observations. However, the reasons for the imbalance of the demonstrations in different modes, such as the habit of the demonstrator, can be meaningless for correct motion generation. It is often the case that we cannot collect enough demonstrations to cover all modes. Even if there are only a few demonstrations, where the human accomplishes the task with particular types of motions, the robot should learn these motions to increase motion diversity.

Model collapse occurs in particular when relatively few demonstrations are available. Several mixture components of the MDN, which are represented by neural networks, are powerful enough to overfit all demonstrations. After training of the MDN, instead of all K mixture components, it uses only a subset of them, which corresponds to the local minima of the NLL and results in poor performance of the MDN for some task parameter queries. As shown in Figure 1(b), one of the two models disappears with the original MDN, and the MDN performs similarly to the GPR. Compared to mode collapse, model collapse is more severe because it can lead to the failure of task execution.

To reduce the occurrence of mode and model collapses, we introduce the negative model entropy cost function as follows:

$$l_{\mathrm{model}}(\boldsymbol{\Theta}) = \sum_{k=1}^{K} p(m=k\,|\,\boldsymbol{D};\boldsymbol{\Theta}) \log p(m=k\,|\,\boldsymbol{D};\boldsymbol{\Theta}), \tag{7}$$

where

$$p(m=k\,|\,\boldsymbol{D};\boldsymbol{\Theta}) = \sum_{i=1}^{M} \pi_k(\boldsymbol{q}_i;\boldsymbol{\Theta})\, p(\boldsymbol{q}_i), \tag{8}$$

where m is the component index, and $p(\boldsymbol{q}_i) \propto M^{-1}$. By minimizing the cost, we increase the uncertainty of the model labels when considering all demonstrations $\boldsymbol{D}$. A high uncertainty of the model labels is equivalent to either equally distributed mixing coefficients for each task parameter query or equally distributed demonstrations to different models. In the former case, if all mixing coefficients for one specific task

parameter query are almost equal and close to $1/K$, each mode has the same probability of being selected to generate motions. Hence, the mode collapse does not occur. In the latter case, if each model is associated with some demonstrations, the corresponding mixture component, i.e., the network branch, is well trained. Hence, model collapse does not occur.

The objective function for training the entropy MDN is a weighted sum of the NLL and the entropy cost function: $w_{\text{NLL}} l_{\text{NLL}} + w_{\text{model}} l_{\text{model}}$. In the following experiments, the weights are empirically determined: $w_{\text{NLL}} = 1$, $w_{\text{model}} = 50$.

Improving the MDN With Failures

In many applications, the failed samples are easily collected with an underfitted MDN model. To reduce the occurrence of these failed MP parameters for similar task parameter queries, we introduce the failure cost function as follows:

$$l_{\text{neg}}(\boldsymbol{\Theta}) = \sum_{i=1}^{M^*} \log\left(\sum_{k=1}^{K} \pi_k(\boldsymbol{q}_i; \boldsymbol{\Theta})\, \mathcal{N}(\boldsymbol{w}_i^*; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_{\text{neg}}) \right), \tag{9}$$

where the normal distribution has the same form as (6), but $\boldsymbol{\Sigma}_{\text{neg}} = \sigma_{\text{neg}} \boldsymbol{I}$. By minimizing this cost, the output mean vector $\boldsymbol{\mu}_i$ for a specific task parameter $\boldsymbol{q}_i$ is kept away from the failed MP parameters $\{\boldsymbol{w}_i^*\}_{i=1}^{M^*}$.

If σ_{neg} is too small, the failure cost does not affect the results; on the other hand, a σ_{neg} that is too large can result in trajectories that are significantly different from demonstrations. Here, we determined σ_{neg} empirically with the smallest variance of all MP parameter components.

To train an MDN with the failure cost, we prepare an evaluation data set. After a certain number of training steps, we run the MDN on this evaluation data set and collect the failed samples in a failures data set $\{\boldsymbol{w}_i^*\}_{i=1}^{M^*}$. In the next training steps, we calculate the failure cost function based on the failures data set. To avoid increasing the computational cost, we use a fixed data set size M^* and remove the earliest failed samples when new samples are collected.

To evaluate the MP generalization methods, we check whether the generated MPs accomplish the tasks with different task parameters. In learning from demonstrations, a successful task execution means that the generated motions are similar to the demonstrations and can accomplish the task with specific task parameters. In the proposed method, we meet this requirement by training the MDN with the NLL, which is related to the similarity between the collected and generated MP parameters. To check whether the trained model meets the latter requirement, we evaluate it with the success rate of task execution.

For task execution, we can only execute one motion after another. Hence, the MP parameter for the task must be determined based on the MDN output distribution in a subsequent step.

Generating Motion With the MDN

The purpose is to generate single motions for some task parameter queries. In the following experiments, we consider two strategies: selecting the most probable mode or choosing the best one from multiple samples. The most probable mode is the output mean vector of the mixture component that has the most significant mixing coefficient. If the Gaussian components of the output GMM have separated means, the most probable mode corresponds to the mode of the GMM.

For one specific task parameter query $\boldsymbol{q}$, the MDN outputs K Gaussian mixture components with their mixing coefficients $\{\pi_k\}_{k=1}^{K}$. The K modes of these Gaussian mixture components correspond to the K most probable motions of different types. However, not all of these K modes can accomplish the task. The most significant mixing coefficient indicates the mode that most likely succeeds. With this strategy, the MDN serves as a deterministic model; hence, we can compare it with previous deterministic methods. Selecting the most probable mode is the simplest way to generate motion from the MDN output. Moreover, this strategy works quite well in many tasks.

However, with the most probable mode, we ignore the information provided by the output variance $\boldsymbol{\Sigma}$ of the MDN. Each of its diagonal elements indicates how various the generated trajectories can be at the corresponding time for successful task execution. When we draw samples from the output GMM for a specific task parameter query, the variance matrix ensures that the samples have a high probability of success. In some tasks, the most probable mode does not work very well, such as in the experiment described in the "The Hit-the-Ball Experiment in MuJoCo" section. To improve the performance, we draw several MP parameters from the output distribution and execute one after another for the task until success. In this case, the success rate is also dependent on the number of samples.

Moreover, with the former strategy, the MDN always generates the same motion for one specific task parameter query. To demonstrate motion diversity and the fact that the MDN learns multiple modes, we also must draw multiple samples from the output distribution (see the "The Throw-the-Ball Experiment" section).

In the next section, we evaluate the proposed method with four experiments. In the first two investigations, we select the most probable mode and compare our method with previous deterministic methods. In the other two, we draw multiple samples from the MDN output to either improve the performance or show the motion diversity.

Experiments and Evaluations

Fitting Polynomials

In this experiment, we consider a fifth-order polynomial $y(x) = \sum_{k=1}^{5} a_k x^k$. The purpose is to learn a mapping from the coefficients a_k to the MP parameter $\boldsymbol{w}$ in (2). The error is the distance between the true fifth-order polynomials and the generated trajectories by the output VMP parameters. We evaluate different methods separately for the inputs with one dimension a_5 to all dimensions $(a_5, a_4, \ldots, a_0)^T$. Figure 4 shows the results for 60 experiments. In each experiment, 30 random coefficients are for training, and 20 are for testing.

The one-step approaches presented in [11] with SVR and GPR are denoted as "–1," while two-step methods are indicated with "–2." Notice that the dot-product kernel is the perfect assumption for this task because the polynomial value is, indeed, a dot product of the coefficients and bases. However, GPR-1 with dot-product kernels is worse than other methods when learning the mapping from $(a_5, a_4)^T$ to $\boldsymbol{w}$; this results because the time-dependent variable x is a part of the input, which loses the advantage of the correct dot-product assumption. In contrast, the two-step method GPR-2 with dot-product kernels perfectly reproduces the polynomial. On the other hand, however, SVR-1 performs better than SVR-2.

For the MDN, we select the most probable VMP parameter as the output. Except for the GPR with dot-product kernels, the MDN outperforms all other methods.

Random-Obstacles Avoidance

To show whether the methods can scale to a more complex task than the one shown in Figure 1, we randomly placed three obstacles in 2D space and asked for the collision-free trajectories with random starts and goals. To collect demonstrations, a person drew curves connecting random starts and goals without collisions with three randomly generated 2D balls on a tablet. Without any instructions, the human demonstrations show multiple modes and models.

The success of the task execution requires that the generated trajectory connects the start and goal without any collision

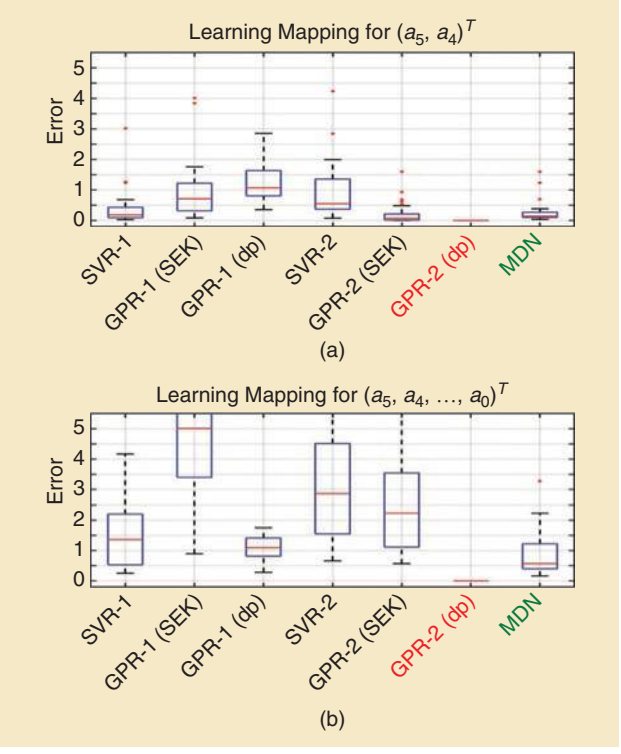


Figure 4. The models learned to fit a fifth-order polynomial, showing the results for learning a mapping from (a) a part of the coefficients to the MP parameters and (b) all coefficients to the MP parameters. dp: dot-product kernel.

with randomly placed obstacles. Due to the task complexity and existence of multiple modes and models, previous approaches cannot achieve acceptable results. With a data set with 100 demonstrations, the TP-GMM has a success rate of only approximately 45% with five local frames (three for the obstacles and two for the start and goal). Both one-step and two-step methods with SVR and GPR perform worse, with a success rate of less than 30%.

For the MDN, we assume three mixture components: $K = 3$. To extract the latent feature (see the blue box in Figure 3), we introduce three separate network branches for three obstacles. Each branch takes the position of one obstacle and the start and goal of the trajectory as the input and outputs a hidden feature vector. The three hidden feature vectors are then concatenated for the rest of the MDN. The entire MDN is trained in an end-to-end manner.

During testing, we select the most probable mode, as mentioned previously. For each number of training data, 30 experiments are conducted for 30 different training data sets randomly chosen from the collected demonstrations. To utilize the failure cost function, after each 100 training steps, the MDN is evaluated on an evaluation data set to produce failed samples. To avoid increasing data, we consider only the most recent 3,000 failed samples.

As shown in Figure 5, with 100 demonstrations, the MDN with both the entropy and failure cost functions ($l_{\text{NLL}} + l_{\text{model}} + l_{\text{neg}}$) achieves a success rate of approximately 82%. The performance is improved further to 85% with 300 demonstrations. With 100 demonstrations, the entropy MDN with the failure cost function ($l_{\text{NLL}} + l_{\text{model}} + l_{\text{neg}}$) achieves the best result, and the entropy MDN ($l_{\text{NLL}} + l_{\text{model}}$) is better than the original MDN (l_{NLL}). Their difference decreases with an increasing number of demonstrations.

In Figure 6 [16], testing samples are shown. The colorized solid curves are generated by the most probable mode (MP parameter) given by the MDN, and the transparent dashed curves correspond to the other two modes, which are not selected by the MDN, with relatively small output mixing coefficients. Three different colors indicate three different modes: the green curves bend toward the top, red curves bend toward the bottom, and blue curves connect the start and goal directly. As can be seen, the MDN accomplishes the task with two steps. One step is to separately update each mixture component branch to increase the success rate of their modes. The other step is to adjust the mixing-coefficient output to select the mode that has the highest chance of accomplishing the task.

The Hit-the-Ball Experiment in MuJoCo

In this experiment, the robot hits the ball with its fist. After being hit, the ball slides on the table and stops at some locations. The final location of the ball on the table is the task parameter query. The purpose is to generate an appropriate robot motion to hit the ball and let the ball stop at a specific location. We conducted this experiment in the MuJoCo simulator [20] with the model of the humanoid robot ARMAR-6 [21].

For the demonstrations, we used a random trajectory generator based on a fifth-order polynomial with which the position and velocity at the end of the trajectory can be specified. The initial ball location on the table is fixed. The end velocities of the trajectories are randomly sampled from a uniform distribution. With different hitting velocities, the ball stops at different locations. We collected the final locations of the ball $\boldsymbol{q}$ and MP parameters $\boldsymbol{w}$ in a training data set.

As shown in Figure 7, the ball can bounce off the borders of the table, which realizes multiple modes for one specific target location in the collected demonstrations. The table measures 260×200 cm, and the ball has a radius of 5 cm. Successful task execution means that the ball is no more than 10 cm from the target location.

We use $K = 3$ mixture components and train the MDN with 50 random demonstrations and test it on 100 new ball locations. When selecting the most probable mode, we get an average success rate of approximately 15%. This poor performance might be because the task requires an accurate trajectory. With a randomly small perturbation of the MP parameters that correspond to the original 50 demonstrations, the success rate can drop rapidly. As mentioned previously to improve the task performance, we draw several samples from the output distribution. In this case, a successful MP generalization means that there exists at least one of these samples that leads to

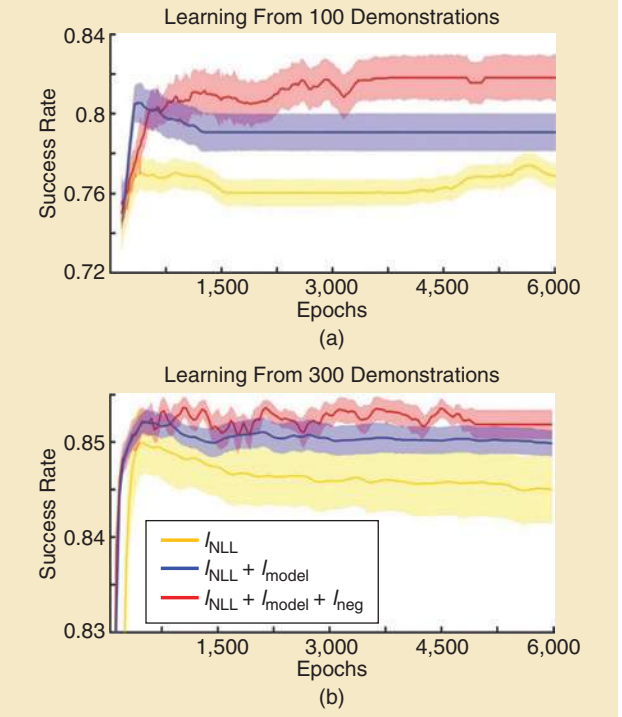


Figure 5. A comparison between the original and entropy MDNs, showing the results with (a) 100 and (b) 300 demonstrations.

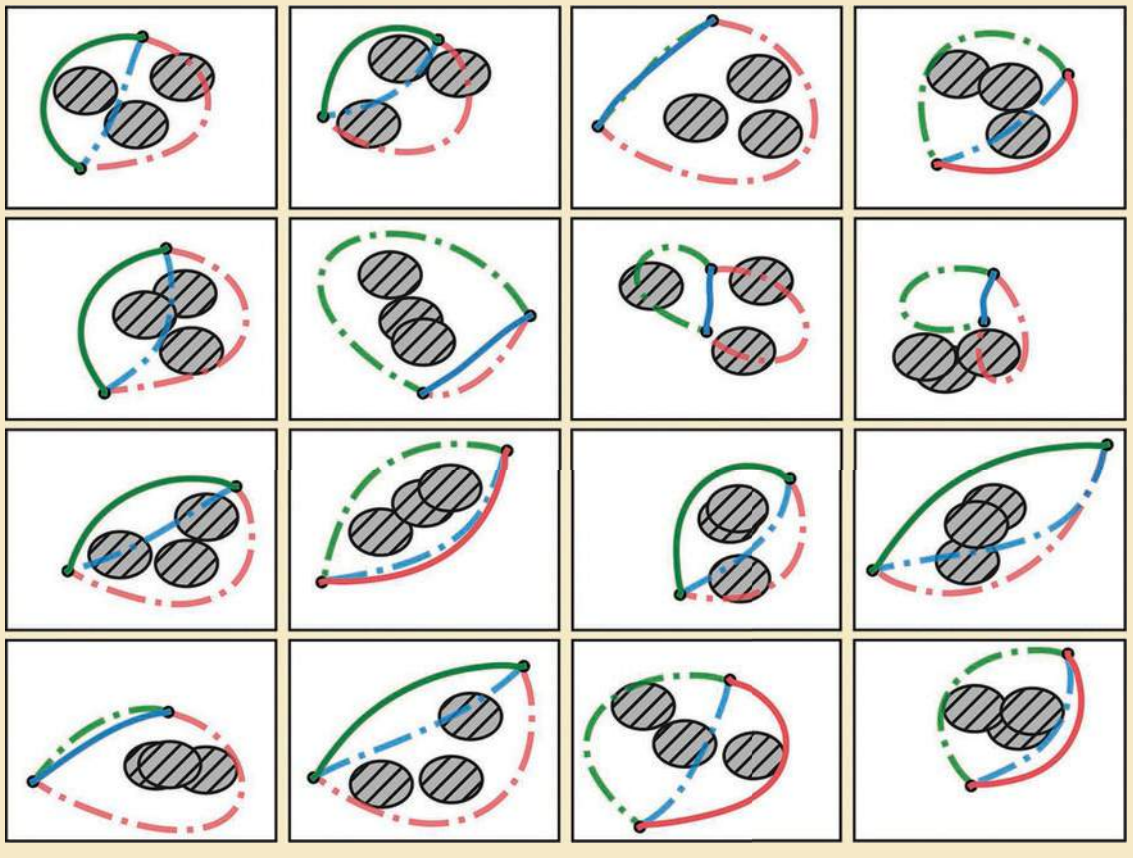


Figure 6. Three obstacles are randomly placed in the 2D space. The solid curves correspond to the most probable mode, and the dashed curves are generated by the two other modes that are not selected by the MDN.

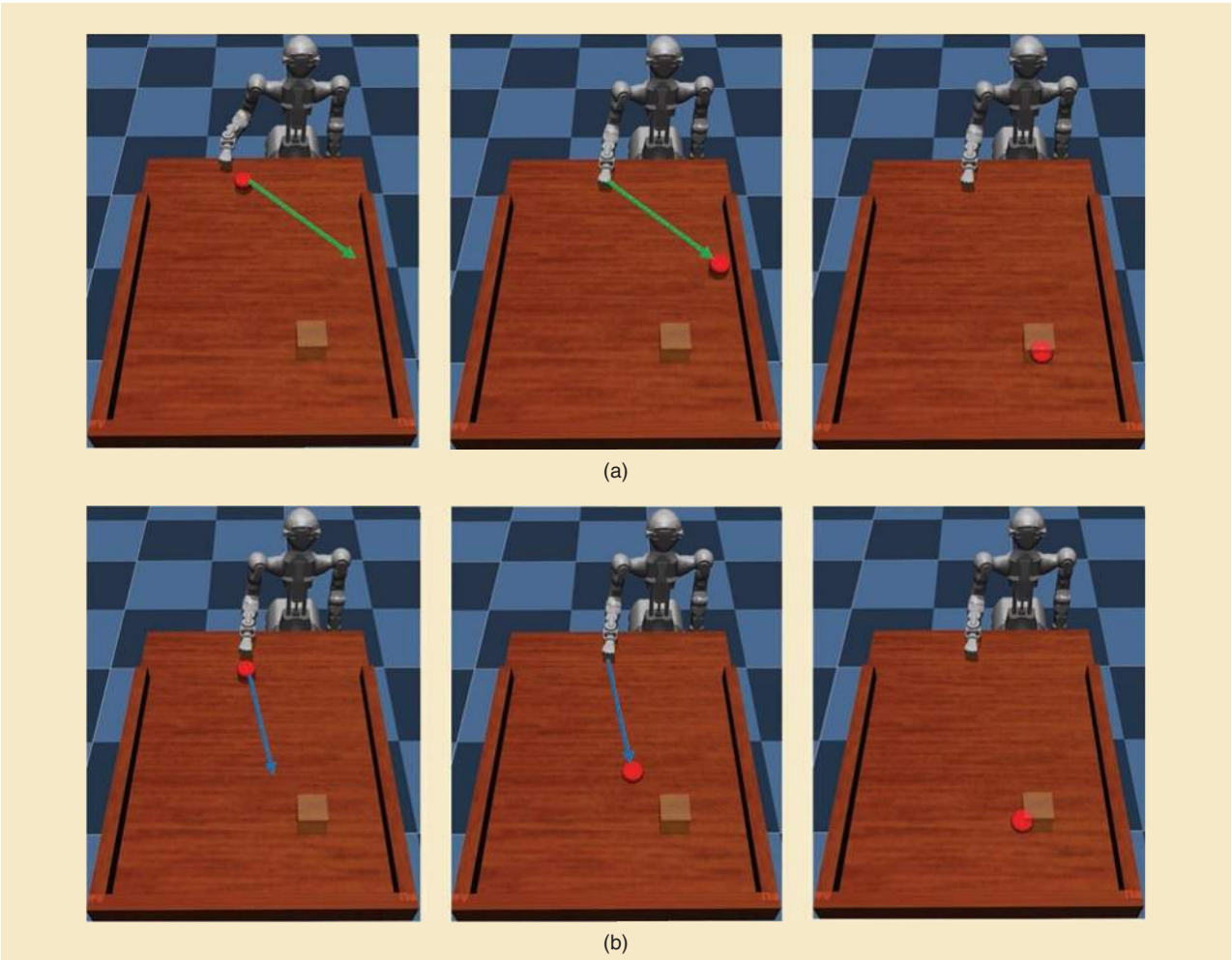


Figure 7. In the hit-the-ball experiment, the desired final ball location is the input of the MDN, which is denoted by a transparent box. (a) The robot hits the ball from its right side, and the ball bounces off the border and stops at the target. (b) The robot hits the ball directly toward the target.

successful task execution. As shown in Figure 8, increasing the number of samples improves the success rate.

In this specific task, the number of samples helps for two reasons. One trivial reason is the setup of the task, which allows successful task executions by chance: the VMP guarantees that the robot hits the ball, and the table borders limit the final ball locations. The other reason is that the MDN learns the correct distribution, which gives a high probability to the correct MP parameter, which is, unfortunately, not precisely the mode. Sampling from the correct distribution has a greater chance of finding the correct solution than directly selecting the most probable mode.

To prove that the latter exists with the MDN for this task, we consider a uniform distribution of the MP parameters as the baseline, whose interval is determined by the minimal and maximal components of the MP parameters, which correspond to the 50 demonstrations. In addition to the baseline, we construct Gaussian distributions by considering the GPR and SVR outputs as mean vectors and with a fixed-variance matrix ($\Sigma = 0.01\boldsymbol{I}$).

As shown in Figure 8, the MDN outperforms the others for all sample numbers. For the baseline, it coincides with the intuition that its success rate is almost proportional to the sample number because it does not learn from the demonstrations. GPR and SVR have better performances than the baseline because they draw the samples close to their output MP parameters. However, the samples drawn around their outputs are totally by chance because of the fixed-variance matrix. In contrast, the MDN learns a relatively correct distribution output. Hence, it already achieves a high success rate with a smaller sample number.

The Throw-the-Ball Experiment

To further evaluate our methods, we let the humanoid robot ARMAR-6 throw a ball at a specific target. The arms of ARMAR-6 have eight degrees of freedom (DoF) each. To simplify the task, we used only four of them without loss of generality (see the 4 DoF in Figure 9). The demonstrations were conducted by a human using kinesthetic teaching. After learning the corresponding MP parameter for each

demonstration, we speed up the motion to 1 s and set a fixed joint goal. The robot hand always opens at 0.55 s. Then, we record the location of the ball when it drops to the ground. By fixing the goals and speed of the motions, these hit-ground locations are dependent only on the shapes of the joint trajectories. In the experiment, we let the robot face the wall, and it can bounce the ball off the wall to the target.

We let the robot throw 50 times with different human demonstrations and randomly split the collected data into 30 for training and 20 for testing. We train an MDN ($K = 2$) on 30 demonstrations. For the testing, we use only the hit-ground locations of the other 20 demonstrations as task parameter queries, which guarantees that all of the hit-ground locations are reachable. During the testing, we place a plate on the ground to indicate the current query. Successful task execution is to throw the ball onto the plate either directly or by bouncing it off the wall. In the experiment, with 10 samples, the robot missed only two out of 20 target hit-ground locations. In Figure 9, for one specific task parameter query, we show how two of 10 MP parameters, which correspond to two different modes, result in different paths of the ball.

Discussion and Conclusions

This work addressed the problem of MP generalization to different tasks and was concerned with two aspects. First, to take the multiple modes and models of human demonstrations into account, we propose using an MDN for the mapping from the task parameter query to the MP parameter distribution. The experiments show that the MDN-based approach outperforms techniques used in previous works. Second, to further reduce the occurrence of mode and model collapse during training of the MDN, we propose the entropy cost function. Moreover, for some tasks, we introduce the failure cost to improve the performance of the MDN further. The comparison of different MDNs shows that the new cost functions perform better than the original one, especially when the set of demonstrations is relatively small.

What we did not consider here is the extrapolation of the method to areas outside the demonstration range. Since the MDN is learned fully from demonstrations, its extrapolation capability is limited. Current methods dealing with the extrapolation problem focus only on a specific set of task parameters, such as the TP-GMM and via-points adaptation of the VMP, described in [5] or our previous work [15]. The extrapolation of MP generalization to arbitrary task parameter queries is still unsolved.

Recent approaches, such as those in [13] and [14], also take task-relevant sensory inputs into account and learn them together with the robot motions. For human–robot

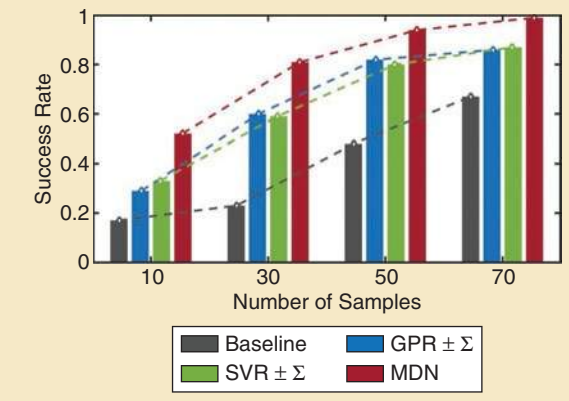


Figure 8. The results of the hit-the-ball experiment show that the MDN (red) outperforms the baseline (gray), GPR ± Σ (blue), and SVR ± Σ (green).

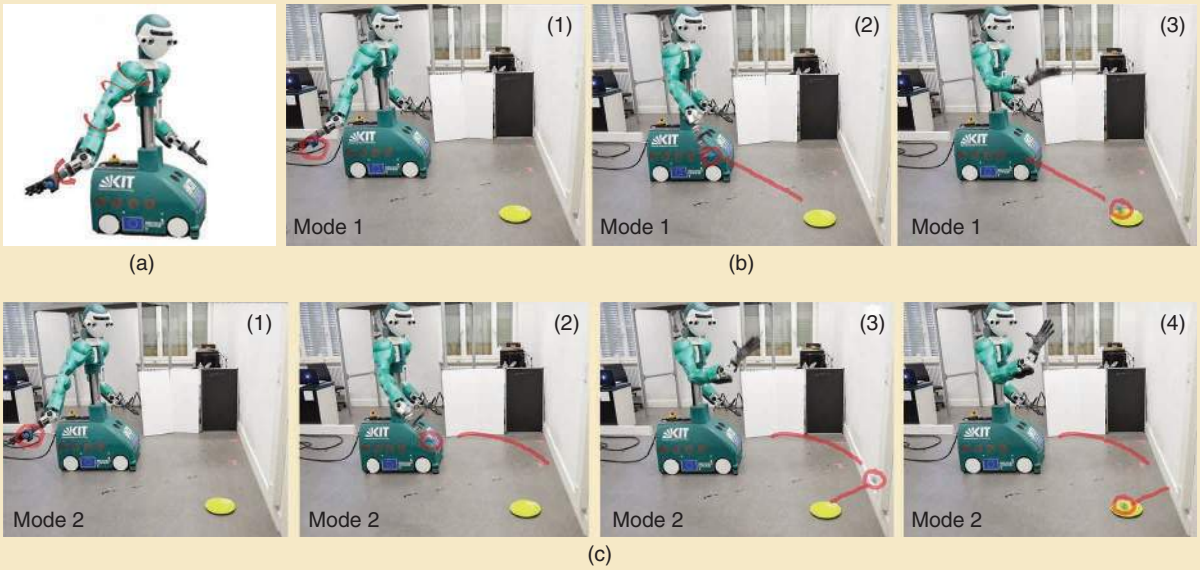


Figure 9. (a) There are 4 DoF used for throwing the ball. (b) The robot throws the ball directly to the target. (c) The robot bounces the ball to the target off the wall.

interactions, the robot motion is generated based on the observed human activity. With the human activity considered as a task parameter query, these methods also learn a mapping from the task to MP parameters. However, unlike the MDN, which directly learns this mapping, these approaches learn a generative model. In the future, we will explore the use of our proposed method for human–robot interaction tasks and compare these methods.

For the sampling strategy, selecting the most probable mode already solves many tasks. However, for some other tasks, the suggested method needs multiple samples to achieve better performance, such as that described in the "The Hit-the-Ball Experiment in MuJoCo" section. This fact requires multiple robot trials for each task parameter query. To solve this problem, we consider either using reinforcement learning to refine the mean vector given by the trained MDN with a small number of trials, as in [22], or training a discriminator that can predict success or failure based on both task and MP parameters.

Acknowledgments

The research leading to these results received funding from the German Federal Ministry of Education and Research under the project Organic Machine Learning (01IS18040A) and the European Union Horizon 2020 Research and Innovation program under grant agreement 643950 (SecondHands). We would like to thank Jonas Rothfuss for the fruitful discussions.

References

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, *Robot Programming by Demonstration*. New York: Springer-Verlag, 2008, pp. 1371–1394.

[2] S. F. Giszter, F. A. Mussa-Ivaldi, and E. Bizzi, "Convergent force fields organized in the frog's spinal cord," *J. Neurosci., Official J. Soc. Neurosci.*, vol. 13, no. 2, pp. 467–491, 1993. doi: 10.1523/JNEUROSCI.13-02-00467.1993.

[3] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, Feb. 2013. doi: 10.1162/NECO_a_00393.

[4] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Proc. Advances Neural Information Processing Systems 26*, 2013, pp. 2616–2624.

[5] S. Calinon, T. Alizadeh, and D. G. Caldwell, "On improving the extrapolation capability of task-parameterized movement models," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Nov. 2013, pp. 610–616. doi: 10.1109/IROS.2013.6696414.

[6] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Trans. Robot.*, vol. 26, no. 5, pp. 800–815, Oct. 2010. doi: 10.1109/TRO.2010.2065430.

[7] Y. Zhou and T. Asfour, "Task-oriented generalization of dynamic movement primitive," in *Proc. 2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2017, pp. 3202–3209. doi: 10.1109/IROS.2017.8206153.

[8] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robot. Auton. Syst.*, vol. 60, no. 10, pp. 1327–1339, 10 2012. doi: 10.1016/j.robot.2012.05.004.

[9] R. Pahic, A. Gams, A. Ude, and J. Morimoto, "Deep encoder-decoder networks for mapping raw images to dynamic movement primitives," in *Proc. 2018 IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 1–6. doi: 10.1109/ICRA.2018.8460954.

[10] B. da Silva, G. Konidaris, and A. Barto, "Learning parameterized skills," in *Proc. 29th Int. Conf. Machine Learning*, June 2012, pp. 1443–1450. doi: 10.5555/3042573.3042758.

[11] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, and O. Sigaud, "Learning compact parameterized skills with a single regression," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, Oct. 2013, pp. 417–422. doi: 10.1109/HUMANOIDS.2013.7030008.

[12] A. Pervez and D. Lee, "Learning task-parameterized dynamic movement primitives using mixture of GMMs," *Intell. Service Robot.*, vol. 11, no. 1, pp. 61–78, Jan. 2018. doi: 10.1007/s11370-017-0235-8.

[13] E. Pignat and S. Calinon, "Learning adaptive dressing assistance from human demonstration," *Robot. Auton. Syst.*, vol. 93, no. C, pp. 61–75, July 2017. doi: 10.1016/j.robot.2017.03.017.

[14] M. Ewerton, G. Neumann, R. Lioutikov, H. Ben Amor, J. Peters, and G. Maeda, "Learning multiple collaborative tasks with a mixture of interaction primitives," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2015, pp. 1535–1542. doi: 10.1109/ICRA.2015.7139393.

[15] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter- and extrapolation capabilities," in *Proc 2019 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 4301–4308. doi: 10.1109/IROS40897.2019.8968586.

[16] C. M. Bishop, "Mixture density networks," Dept. of Computer Science and Applied Mathematics, Aston Univ., Birmingham, NCRG/94/004, 1994.

[17] G. Mclachlan and K. Basford, *Mixture Models: Inference and Applications to Clustering* (Statistics, Textbooks and Monographs Series 84). New York: Marcel Dekker, 1988.

[18] L. U. Hjorth and I. T. Nabney, "Regularisation of mixture density networks," in *Proc. 9th Int. Conf. Artificial Neural Networks ICANN 99*, Sept. 1999, vol. 2, pp. 521–526. doi: 10.1049/cp:19991162.

[19] O. Makansi, E. Ilg, Ö. Çiçek, and T. Brox, "Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7137–7146. doi: 10.1109/CVPR.2019.00731.

[20] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Proc 2012 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 5026–5033. doi: 10.1109/IROS.2012.6386109.

[21] T. Asfour et al., "Armar-6: A collaborative humanoid robot for industrial environments," in *Proc 2018 IEEE/RAS Int. Conf. Humanoid Robots (Humanoids)*, pp. 447–454. doi: 10.1109/HUMANOIDS.2018.8624966.

[22] J. Kober and J. Peters, "Reinforcement learning in robotics: A survey," in *Learning Motor Skills: From Algorithms Robot Experiments*. Cham, Switzerland: Springer-Verlag, 2014, pp. 9–67.

You Zhou, Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Germany. E-mail: you.zhou@kit.edu.

Jianfeng Gao, Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Germany. E-mail: jianfeng.gao@kit.edu.

Tamim Asfour, Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Germany. E-mail: asfour@kit.edu.

Gaussians on Riemannian Manifolds

By Sylvain Calinon

This article presents an overview of robot learning and adaptive control applications that can benefit from a joint use of Riemannian geometry and probabilistic representations. The roles of Riemannian manifolds, geodesics, and parallel transport in robotics are discussed, and several forms of manifolds already employed in robotics are explained. A varied range of techniques employing Gaussian distributions on Riemannian manifolds is then introduced, and two example applications are presented, involving the control of a prosthetic hand from surface electromyography (sEMG) data and the teleoperation of a bimanual underwater robot.

Riemannian Geometry in Robotics

Data encountered in robotics are characterized by simple but varied geometries that are sometimes underexploited in robot learning and adaptive control algorithms. Such data include joint angles in revolving articulations [1], rigid body motions [2], [3], unit quaternions to represent orientations

Applications for Robot Learning and Adaptive Control

©ISTOCKPHOTO.COM/VCHAL

Digital Object Identifier 10.1109/MRA.2020.2980548
Date of current version: 6 April 2020

[4], and symmetric positive definite (SPD) matrices, which can represent sensory data processed as spatial covariances [5], inertia [6], [7], and stiffness/manipulability ellipsoids [8] as well as metrics used in the context of similarity measures.

Moreover, many applications require all these heterogeneous data to be handled together. Several robotics techniques employ components from the framework of Riemannian geometry. But, unfortunately, they are often implemented without providing an explicit link to this framework, which can either weaken the links to other techniques or limit the potential extensions. This can, for example, be the case when computing orientation errors with a logarithmic map in the context of inverse kinematics and when interpolating between two unit quaternions with spherical linear interpolation. This article aims to highlight the links among existing techniques and catalog the missing links that could be explored in further research. These points are discussed in the context of varied robot learning and adaptive control challenges.

Riemannian manifolds are related to a wide range of problems in machine learning and robotics. This article mainly focuses on robot learning applications based on Gaussian distributions. This includes techniques requiring that uncertainty and statistical modeling be computed on structured non-Euclidean data. The article presents an overview of existing work and further perspectives on jointly exploiting statistics and Riemannian geometry. One appealing use of Riemannian geometry in robotics is that it provides a principled and simple way to extend algorithms initially developed for Euclidean data to other manifolds by efficiently taking into account prior geometric knowledge about those manifolds.

By using Riemannian manifolds, data of various forms can be treated in a unified manner with the advantage that existing models and algorithms initially developed for Euclidean data can be extended to a wider range of data structures. They can, for example, be used to revisit constrained optimization problems formulated in the Euclidean space by treating them as unconstrained problems, inherently taking into account the geometry of the data.

Figure 1 shows various common problems in robotics that can directly leverage Riemannian geometry. In Figure 1(a), a set of data points is clustered as two distributions represented in red and blue, trained as a Gaussian mixture model (GMM), where each point is displayed in the color of the most likely cluster to which it belongs. In Figure 1(b), the intersection of two distributions (in red and blue) results in a distribution (in black), which is the equivalent of a product of Gaussians. In Figure 1(c), a controller is computed by solving a linear quadratic tracking (LQT) problem, with the goal of reaching a

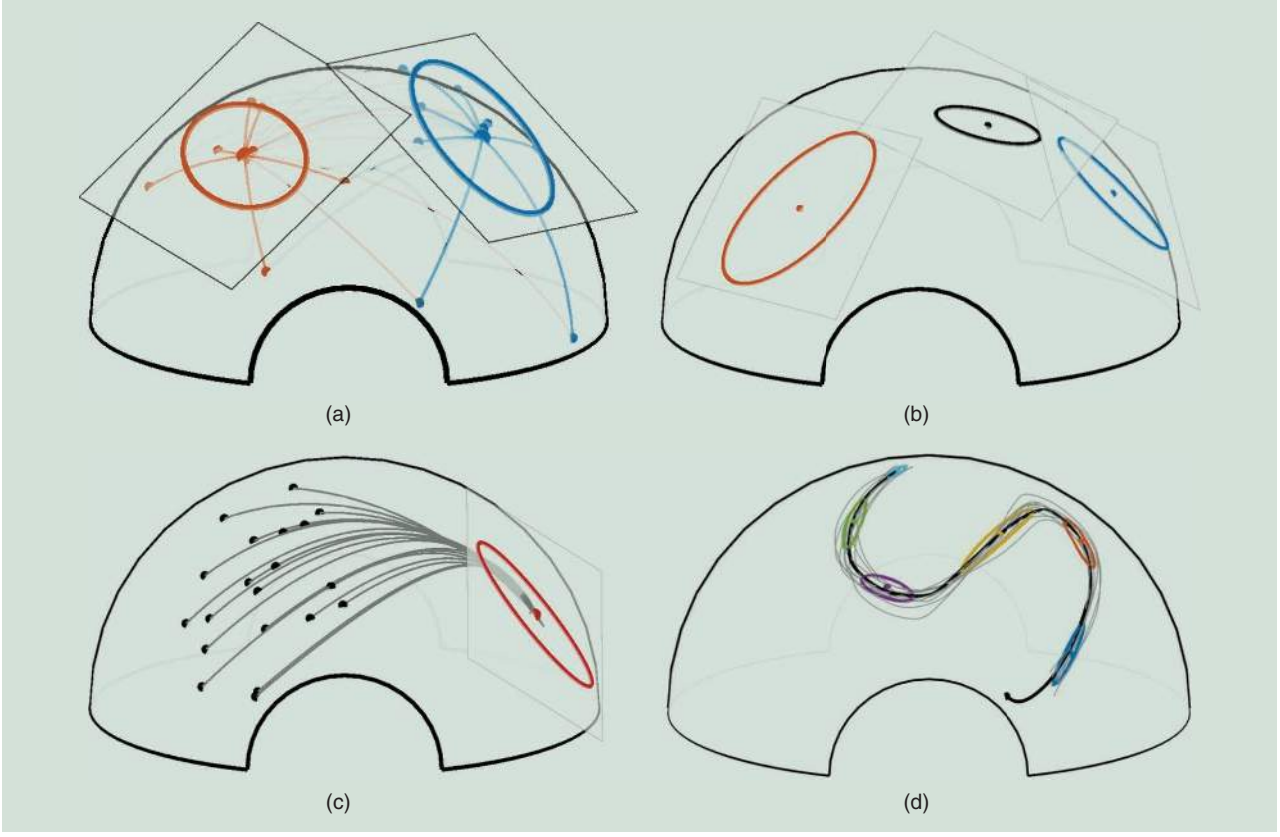


Figure 1. Several problems in robotics that can leverage the proposed Gaussian-based representation on Riemannian manifolds: (a) clustering, (b) information fusion, (c) LQT, and (d) MPC. Such an approach can be used to extend clustering, regression, fusion, control, and planning problems to non-Euclidean data. In these examples, Gaussians are defined with centers on the manifolds and covariances in the tangent spaces of the centers.

target point on the manifold within a desired precision matrix, represented here as a covariance matrix (the inverse of precision matrix) in the tangent space of the target point. In the model predictive control (MPC) example of Figure 1(d), a reference path is defined as a set of Gaussians acting as via points to pass through (i.e., within desired covariances). This GMM is first learned from a set of demonstrated reference paths (gray lines). The resulting controller computes a series of commands anticipating the next points to reach, resulting in a path on the manifold (black lines).

The problems depicted in Figure 1 require data to be handled in a probabilistic manner. For Euclidean data, multivariate Gaussian distributions are typically considered to encode either the (co)variations of the data or the uncertainty of the estimates. This article discusses how these approaches can be extended to other manifolds by exploiting a Riemannian extension of Gaussian distributions. A practitioner perspective is adopted, with the goal of conveying the main intuitions behind the presented algorithms, sometimes at the expense of a more rigorous treatment of each topic. Didactic source codes accompany the article, available as part of PbDlib [9], a collection of source codes for robot programming by demonstration (learning from demonstration), including various functionalities for statistical learning, dynamical systems, optimal control, and Riemannian geometry. Two distinct versions are maintained, and they can be used independently in MATLAB (with full GNU Octave compatibility) or C++. Scalars are denoted by lowercase letters (x), vectors by boldface lowercase letters ($\boldsymbol{x}$), and matrices by boldface uppercase letters ($\boldsymbol{X}$), where $\boldsymbol{X}^\top$ is the transpose of $\boldsymbol{X}$. Manifolds and tangent spaces are designated by the calligraphic letters $\mathcal{M}$ and $\mathcal{TM}$, respectively.

Riemannian Manifolds

A smooth d-dimensional manifold $\mathcal{M}$ is a topological space that locally behaves like the Euclidean space $\mathbb{R}^d$. A Riemannian manifold is a smooth and differentiable manifold equipped with a PD metric tensor. For each point $\boldsymbol{p} \in \mathcal{M}$, there exists a tangent space $\mathcal{T}_p\mathcal{M}$ that locally linearizes the manifold. On a Riemannian manifold, the metric tensor induces a PD inner product on each tangent space $\mathcal{T}_p\mathcal{M}$, which enables vector lengths and the angles between vectors to be measured. The affine connection, computed from the metric, is a differential operator that provides, among other functionalities, a way to compute geodesics and transport vectors on tangent spaces along any smooth curves on the manifold [10], [11]. It also fully characterizes the intrinsic curvature and torsion of the manifold. The Cartesian product of two Riemannian manifolds is also a Riemannian manifold (often called *manifold bundles* or *manifold composites*), which enables joint distributions to be constructed on any combination of Riemannian manifolds. Two basic notions of Riemannian geometry are crucial for robot learning and adaptive control applications. They are illustrated in Figure 2 and described as follows:

- *Geodesics*: The minimum-length curves between two points on a Riemannian manifold are called *geodesics*. Similar to straight lines in the Euclidean space, the second derivative is zero everywhere along a geodesic. The exponential map $\text{Exp}_{x_0}: \mathcal{T}_{x_0}\mathcal{M} \to \mathcal{M}$ charts a point $\boldsymbol{u}$ in the tangent space of $\boldsymbol{x}_0$ to a point $\boldsymbol{x}$ on the manifold so that $\boldsymbol{x}$ lies on the geodesic starting at $\boldsymbol{x}_0$ in the direction $\boldsymbol{u}$. The norm of $\boldsymbol{u}$ is equal to the geodesic distance between $\boldsymbol{x}_0$ and $\boldsymbol{x}$. The inverse map is called the *logarithmic map*, $\text{Log}_{x_0}: \mathcal{M} \to \mathcal{T}_{x_0}\mathcal{M}$. Figure 2(a) depicts these mapping functions.
- *Parallel transport*: Parallel transport $\Gamma_{g\to h}: \mathcal{T}_g\mathcal{M} \to \mathcal{T}_h\mathcal{M}$ moves vectors between tangent spaces such that the inner product between two vectors in a tangent space is conserved. It employs the notion of connection, defining how to associate vectors between infinitesimally close tangent spaces. This connection enables the smooth transport of a vector from one tangent space to another by sliding it (with

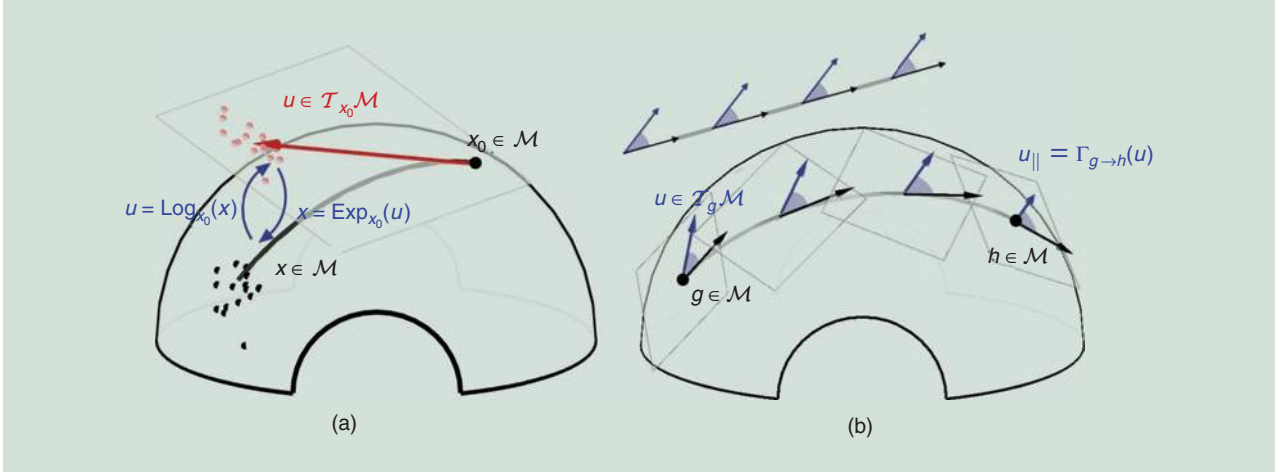


Figure 2. Applications in robotics using Riemannian manifolds rely on two well-known principles of Riemannian geometry: exponential/logarithmic mapping and parallel transport, which are depicted on an $\mathcal{S}^2$ manifold embedded in $\mathbb{R}^3$. (a) The bidirectional mappings between the tangent space and manifold. (b) The parallel transport of a vector along a geodesic.

infinitesimal moves) along a curve. Figure 2(b) depicts this operation.

In Figure 2(b), the flat surfaces show the coordinate systems of several tangent spaces along the geodesic. The black vectors represent the directions of the geodesic in the tangent spaces. The blue vectors are transported from $\boldsymbol{g}$ to $\boldsymbol{h}$. Parallel transport enables a vector $\boldsymbol{u}$ in the tangent space of $\boldsymbol{g}$ to be transported to the tangent space of $\boldsymbol{h}$ by ensuring that the angle (i.e., inner product) between $\boldsymbol{u}$ and the direction of the geodesic (represented as black vectors) is conserved. At point $\boldsymbol{g}$, this direction is expressed as $\mathrm{Log}_{\boldsymbol{g}}(\boldsymbol{h})$. This operation is crucial to combine information available at $\boldsymbol{g}$ with information available at $\boldsymbol{h}$ by taking into account the rotation of the coordinate systems along the geodesic [notice the rotation of the tangent spaces in Figure 2(b)]. In the Euclidean space [Figure 2(a)], the parallel transport is simply the identity operator (a vector operation can be applied to any point without additional transformation).

By extension, a covariance matrix Σ can be transported with $\Sigma_{\parallel} = \sum_{i=1}^{d} \Gamma_{\boldsymbol{g}\to\boldsymbol{h}}(\boldsymbol{v}_i)\,\Gamma_{\boldsymbol{g}\to\boldsymbol{h}}^{\top}(\boldsymbol{v}_i)$, using the eigen decomposition $\Sigma = \sum_{i=1}^{d} \boldsymbol{v}_i \boldsymbol{v}_i^{\top}$. For many manifolds in robotics, this transport operation can equivalently be expressed as a linear mapping $\Sigma_{\parallel} = \boldsymbol{A}_{\boldsymbol{g}\to\boldsymbol{h}} \Sigma \boldsymbol{A}_{\boldsymbol{g}\to\boldsymbol{h}}^{\top}$ (see the supplementary material for this article in IEEE *Xplore* for the computation of $\boldsymbol{A}_{\boldsymbol{g}\to\boldsymbol{h}}$).

Manifolds in Robot Applications

The most common manifolds in robotics are homogeneous, providing simple analytic expressions for exponential/logarithmic mapping and parallel transport. Some of the most important representations are listed in the following (see the supplementary material for this article in IEEE *Xplore* for the corresponding mapping and transport operations).

Figure 3 provides four examples of Riemannian manifolds that can be employed in robot manipulation tasks. For these four manifolds, the bottom graphs depict $\mathcal{S}^2$, $\mathcal{S}^2_{++}$, $\mathcal{H}^2$, and $\mathcal{G}^{3,2}$, with a clustering problem in which the data points (black dots/planes) are segmented in two classes, each represented by a center (red and blue dots/planes). The geodesics depicted in Figure 3 show the specificities of each manifold. The sphere manifold $\mathcal{S}^d$ is characterized by constant positive curvature. The elements of $\mathcal{S}^d_{++}$ can be represented as the interior of a convex cone embedded in its tangent space Sym^d. Here, the three axes correspond to A_{11}, A_{12}, and A_{22} in the SPD matrix $\left(\begin{smallmatrix} A_{11} & A_{12} \\ A_{12} & A_{22} \end{smallmatrix}\right)$. The hyperbolic manifold $\mathcal{H}^d$ is characterized by constant negative curvature. Several representations exist: $\mathcal{H}^2$ can, for example, be represented as the interior of a unit disk in the Euclidean space, with the boundary of the disk representing an infinitely remote point (the Poincaré disk model, as depicted here). In this model, geodesic paths are arcs of circles intersecting the boundary perpendicularly, and $\mathcal{G}^{d,p}$ is the Grassmann manifold of all p-dimensional subspaces of $\mathbb{R}^d$.

Two Lie groups widely used in robotics are also presented, namely special orthogonal (SO) group SO(3) and special Euclidean (SE) group SE(3). Similarities and differences between Riemannian manifolds and Lie groups are discussed in the following section. Examples of applications for the aforementioned Riemannian manifolds are presented as follows:

- *The sphere manifold*: $\mathcal{S}^d$ can be used in robotics to encode directions/orientations. Unit quaternions $\mathcal{S}^3$ can be used

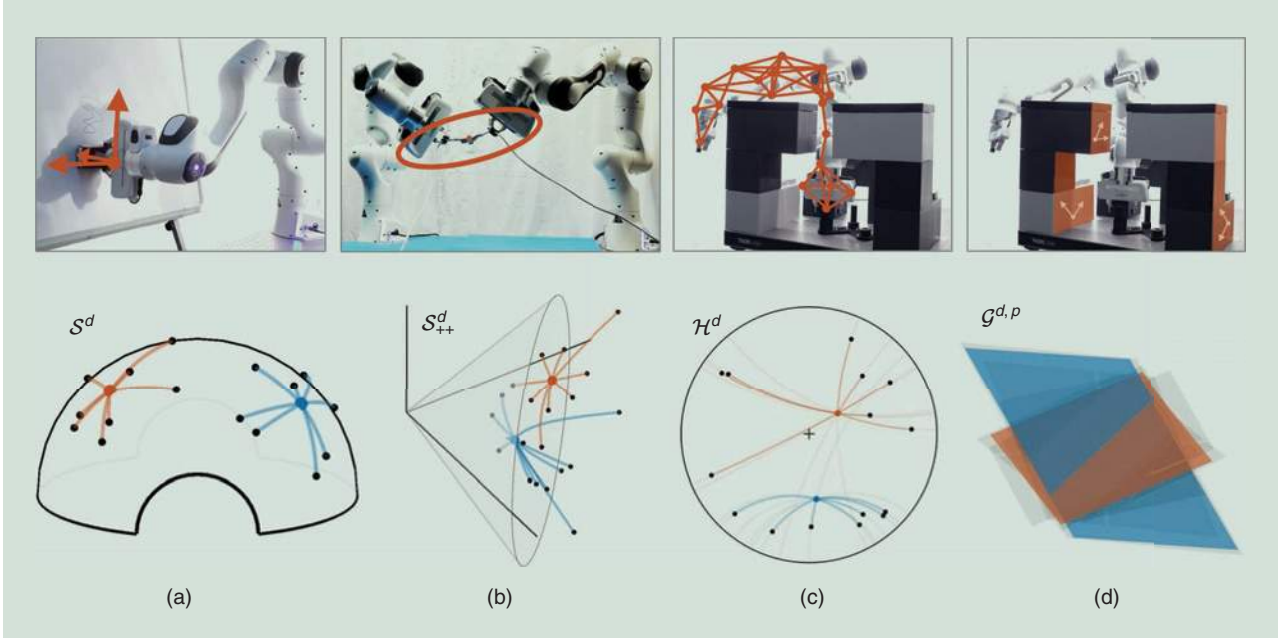


Figure 3. Structured manifolds in robotics. (a) $\mathcal{S}^3$ can be used to represent the orientation of a robot end effectors (unit quaternions), and (b) $\mathcal{S}^6_{++}$ can be employed to represent the manipulability ellipsoids (manipulability capability in translation and rotation) corresponding to an SPD matrix manifold. (c) $\mathcal{H}^d$ can be used to represent trees, graphs, and roadmaps, while (d) $\mathcal{G}^{d,p}$ can be employed to represent subspaces (planes, null spaces, and projection operators).

to represent end-effector (tool-tip) orientations [4], while $\mathcal{S}^2$ can be employed to signify a unit directional vector that is perpendicular to surfaces (e.g., for contact planning). Articulatory joints can be represented on the torus $\mathcal{S}^1 \times \mathcal{S}^1 \times \ldots \times \mathcal{S}^1$ [12], [13]. The Kendall shape space used to encode 3D skeletal motion-capture data also relies on unit spheres [14].

- *The SO group*: $\mathrm{SO}(d)$ is the group of rotations around the origin in a d-dimensional space. SO(2) and SO(3) are widely used in robotics. For example, in [15], the manifold structure of the rotation group is exploited for preintegration and uncertainty propagation in SO(3). This is used for state estimation in visual–inertial odometry with mobile robots. In [16], Kalman filtering adapted to data in SO(3) is used for estimating the attitude of robots that can rotate in space. The optimization problem in [17] employs sequential quadratic programming working directly on the manifold $\mathrm{SO}(3) \times \mathbb{R}^3$.
- *The SE(3)*: SE(3) is the group of rigid body transformations. A rigid body transformation is composed of a rotation and a translation. The geometry of SE(3) can be used to describe the kinematics and Jacobian of robots [2]. Therefore, it is widely used to describe robot motion and pose estimation [18]. For example, in [3], exponential maps are exploited to associate uncertainty with SE(3) data points in robot pose estimation problems.
- *The manifold of SPD matrices*: $\mathcal{S}^d_{++}$ can be employed in various ways in robotics. For example, human–robot collaboration applications require the use of various sensors. These sensory data can be preprocessed with sliding windows to analyze at each time step the evolution of the signals within a short time window (e.g., to analyze data flows). Often, such analysis takes the form of spatial covariances, which are SPD matrices [5]. In robot control, tracking gains can be defined in the form of SPD matrices. The use of tracking gains as full SPD matrices instead of scalars has the advantage of enabling the controller to take into account the coordination of different control variables (e.g., motor commands). For articulatory joints, these coordinations often relate to characteristic synergies in human movements. Manipulability ellipsoids are representations used to analyze and control robot dexterity as a function of the articulatory joints configuration. This descriptor can be designed according to different task requirements, such as tracking a desired position and applying a specific force [8], [19]. Manipulator inertia matrices also belong to $\mathcal{S}^d_{++}$ and can, for example, be exploited in human-like trajectory planning [13]. SPD matrices are also used in problems related to metric interpolation/extrapolation and metric learning [20]. In covariant Hamiltonian motion planning [21], a precision matrix (metric tensor) is used to prefer perturbations resulting in small accelerations in the overall trajectory. In [22], Riemannian manifold policies are employed to generate natural obstacle-avoidance reaching motion through traveling along geodesics of curved spaces defined by the presence of obstacles.
- *Hyperbolic manifolds*: $\mathcal{H}^d$ are the analogs of spheres with constant negative curvature instead of constant positive curvature. They are currently underexploited in robotics despite their interesting potential in a wide range of representations, including dynamical systems, Toeplitz/Hankel matrices, and autoregressive models [23]. Notably, hyperbolic geometry could be used to encode and visualize heterogeneous topology data, including graphs and trees structures, such as rapidly exploring random trees [24], designed to efficiently search nonconvex, high-dimensional spaces in motion planning by randomly building a space-filling tree. The interesting property of hyperbolic manifolds is that the circumference of a circle grows exponentially with its radius, which means that exponentially more space is available with increasing distance. It provides a convenient representation for hierarchies, which tend to expand exponentially with depth.
- *The Grassmannian*: $\mathcal{G}^{d,p}$ is the manifold of all p-dimensional subspaces of $\mathbb{R}^d$. It can, for example, be used to extract and cluster planar surfaces in the robot's 3D environment. This manifold is largely underrepresented in robotics despite the fact that such a structure can be used in various approaches, such as system identification [25], spatiotemporal modeling of human gestures [26], and the encoding of nullspaces and projection operators in a probabilistic way.
- *Manifolds with nonconstant curvature*: These are also employed in robotics, such as spaces endowed with the Fisher information metric [27], [28] or kinetic energy metric [1], [12], [13], [18]. As described in the "Riemannian Manifolds" section, the curvature of a Riemannian manifold depends on the selected metric tensor. Consequently, a varying metric will result in a changing curvature. Many problems in robotics can be formulated with a smoothly varying matrix $\boldsymbol{M}$ (Riemannian metric) that measures the distance between two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as a quadratic error term $(\boldsymbol{x}_1 - \boldsymbol{x}_2)^\top \boldsymbol{M}(\boldsymbol{x}_1 - \boldsymbol{x}_2)$. In this context, the Riemannian formulation has the advantage of being coordinate independent (i.e., geodesic paths are invariant to the choice of local coordinates) [1], [12], [13]. In robot dynamics problems, this is typically useful for taking into account the inertia in the robot motion [1]. In policy-learning problems, if the conditional density of the action given the state is Gaussian, the natural policy gradient is given by the Fisher information matrix [28], which can, for example, be used in deep reinforcement learning [29].

The use of a Riemannian metric is also relevant for deep generative models, such as variational autoencoders (VAEs) and generative adversarial networks, as it provides a geometric interpretation of these models. For example, VAEs learn nonlinear data distributions through a set of latent variables

and a nonlinear generator function that maps latent points into the input space. The nonlinearity of the generator implies that the latent space gives a distorted view of the input space. The latent space provides a low-dimensional representation of the data manifold, of which it can reveal the underlying geometrical structure [30].

In neural networks such as VAEs, by using activation functions that are C^2 differentiable, an SPD matrix $\boldsymbol{M} = \boldsymbol{J}^\top \boldsymbol{J}$ can be employed as a smoothly changing inner product structure acting as a local Mahalanobis distance measure, where $\boldsymbol{J}$ is the Jacobian characterizing the decoder function. The method yields a distance measure that can, for example, be used to replace linear interpolations in the latent space by geodesics. In [30], Arvanitidis et al. show that in the latent space learned by a VAE, distances and interpolants can be improved significantly under this metric, which, in turn, improves probability distributions, sampling algorithms, and clustering in the latent space.

One downside of manifolds with nonconstant curvature is that the geodesic optimization problem most often corresponds to a nonconvex problem (a system of ordinary differential equations) that can be computationally heavy to solve. Several research directions can be explored to address this issue. In [31], the problem is circumvented by spanning the latent space with a discrete finite graph (a k-dimensional tree data structure with edge weights based on Riemannian distances) used in conjunction with a classic A^* search algorithm. Recent approaches in discrete differential geometry also address similar problems by extending continuous Riemannian manifolds to discrete formulations with fast computations [32]. Currently, these developments principally target computer graphics applications, but they have great potential in robotics. They could, for example, provide an approach to link discrete robot planning problems, such as probabilistic roadmaps, to their continuous counterparts in Riemannian geometry.

Riemannian Geometry and Lie Theory

A Lie group is a smooth and differentiable manifold that possesses a group structure, therefore satisfying the group axioms. There are strong links between Riemannian geometry and Lie theory. In particular, some Lie groups, such as SO(3), can be endowed with a bi-invariant Riemannian metric, which gives them the structure of a Riemannian manifold. In robotics, Lie theory is mainly exploited for applications involving SO(3) and SE(3) groups.

In the literature, distinctive vocabulary and notation are often employed, which hinders some of the links between the applications from exploiting Riemannian geometry and Lie theory. Among these differences, the Lie algebra is the tangent space at the origin of the manifold, acting as a global reference, while $\boldsymbol{u}^\wedge$ (hat) and $\boldsymbol{u}^\vee$ (vee) are used to transform elements from the Lie algebra (which can have nontrivial structures, such as complex numbers and skew-symmetric matrices) to vectors in $\mathbb{R}^d$, which are easier to manipulate. They are the operations corresponding to the exponential and logarithm maps in Riemannian geometry. In Lie theory, $\oplus$ and $\ominus$ are operators used to facilitate compositions with exponential/logarithmic mapping operations. For further reading, an excellent introduction to Lie theory for robot applications can be found in [33].

Gaussian Distributions on Riemannian Manifolds

Several approaches have been proposed to extend Gaussian distributions in Euclidean space to Riemannian manifolds [34]. Here, we focus on a simple approach that consists of estimating the mean of the Gaussian as a centroid on the manifold (also called the *Karcher/Fréchet mean*) and representing the dispersion of the data as a covariance expressed in the tangent space of the mean [4], [10], [35]. Distortions arise when points are too far from the mean, but this distortion is negligible in most robotics applications. In particular, this effect is strongly attenuated when a mixture of Gaussians is

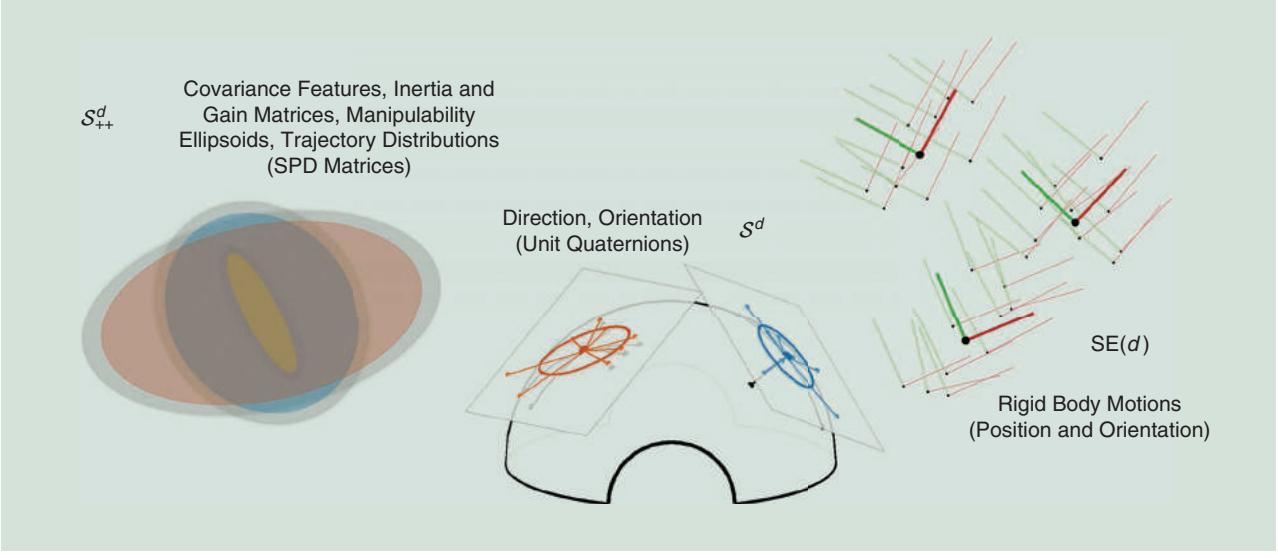


Figure 4. Clustering on various manifolds with GMMs.

considered, as each Gaussian is employed to model a limited region of the manifold. In the general case of a manifold $\mathcal{M}$, such a model is a distribution maximizing the entropy in the tangent space. It is defined as

$$\mathcal{N}_{\mathcal{M}}(\boldsymbol{x}\,|\,\boldsymbol{\mu},\Sigma)=((2\pi)^{d}\,|\,\Sigma\,|)^{-\frac{1}{2}}\,e^{-\frac{1}{2}\mathrm{Log}_{\mu}(\boldsymbol{x})\Sigma^{-1}\mathrm{Log}_{\mu}(\boldsymbol{x})},$$

where $\boldsymbol{x}\in\mathcal{M}$ is a point of the manifold, $\boldsymbol{\mu}\in\mathcal{M}$ is the mean of the distribution (origin of the tangent space), and $\Sigma\in\mathcal{T}_{\mu}\mathcal{M}$ is the covariance defined in this tangent space.

For a set of N data points, this geometric mean corresponds to the minimization,

$$\min_{\mu}\sum_{n=1}^{N}\mathrm{Log}_{\mu}(\boldsymbol{x}_n)^{\top}\,\mathrm{Log}_{\mu}(\boldsymbol{x}_n),$$

which can be solved by a simple and fast Gauss–Newton iterative algorithm. The algorithm starts from an initial estimate on the manifold and an associated tangent space. The data points $\{\boldsymbol{x}_n\}_{n=1}^{N}$ are projected in this tangent space to compute a direction vector, which provides an updated estimate of the mean. This process is repeated by iterating

$$\boldsymbol{u}=\frac{1}{N}\sum_{n=1}^{N}\mathrm{Log}_{\mu}(\boldsymbol{x}_n),\quad\boldsymbol{\mu}\leftarrow\mathrm{Exp}_{\mu}(\boldsymbol{u}),$$

until convergence. In practice, such an algorithm converges very fast in only a couple of iterations (typically fewer than 10 for the accuracy required by the applications presented here). After convergence, a covariance is computed in the tangent space as $\Sigma=1/N\,\Sigma_{n=1}^{N}\mathrm{Log}_{\mu}(\boldsymbol{x}_n)\,\mathrm{Log}_{\mu}^{\top}(\boldsymbol{x}_n)$; see Figure 1. This distribution can, for example, be used in a control problem to represent a reference to track with an associated required precision (e.g., learned from a set of demonstrations). Such a learning and control problem results in the LQT solution depicted in Figure 1 and described in detail in [36]. Importantly, this geometric mean can be directly extended to weighted distances, which will be exploited in the next sections for mixture modeling, fusion (product of Gaussians), regression (Gaussian conditioning) and planning problems.

Gaussian Mixture Model

The GMM is a ubiquitous representation in robotics, including clustering and modeling of distributions as a superposition of Gaussians (see Figure 4). Similar to a GMM in the Euclidean space, a GMM on a manifold $\mathcal{M}$ is defined by $p(\boldsymbol{x})=\Sigma_{k=1}^{K}\pi_k\mathcal{N}_{\mathcal{M}}(\boldsymbol{x}\,|\,\boldsymbol{\mu}_k,\Sigma_k)$, with K being the number of components and π_k the mixing coefficients (priors) such that $\Sigma_k\pi_k=1$ and $\pi_k\geq 0,\forall k\in\{1,\ldots,K\}$. The parameters of this GMM can be estimated by expectation maximization

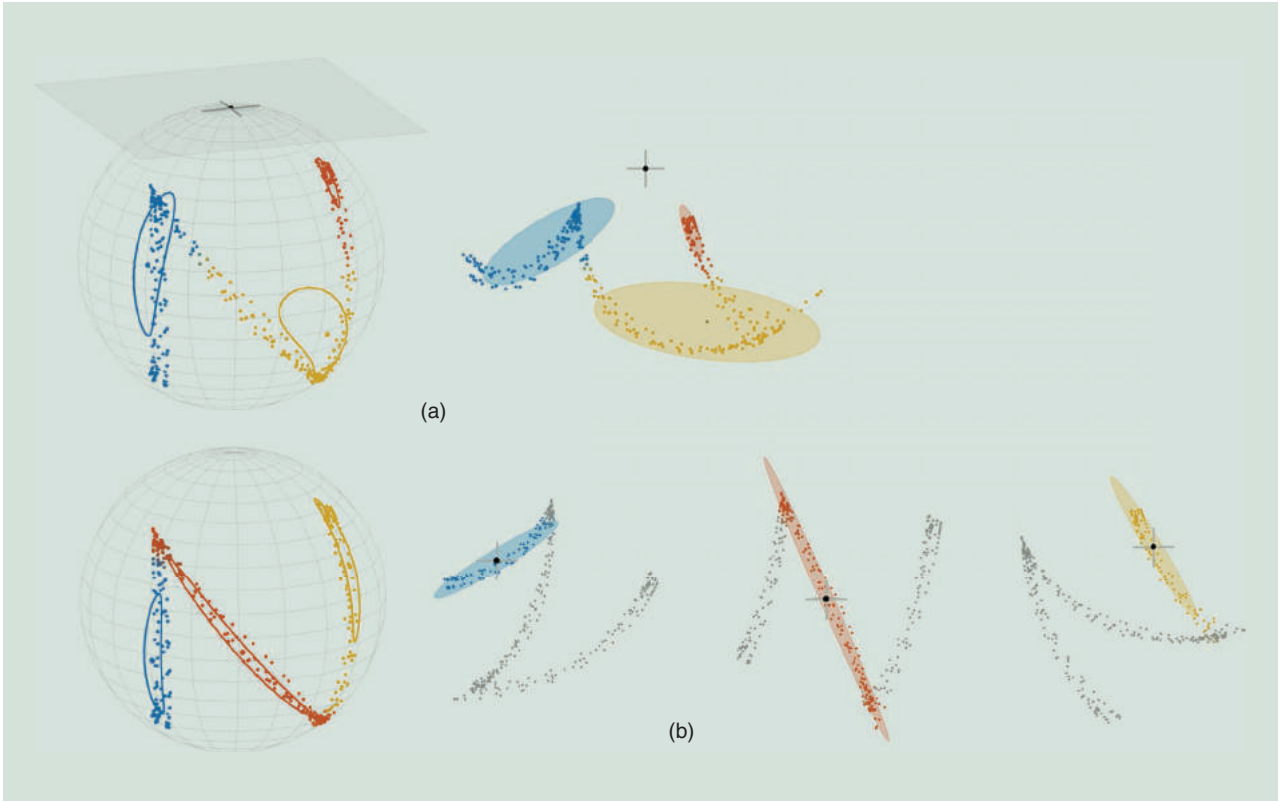


Figure 5. The GMM trained with the EM algorithm on an $\mathcal{S}^2$ manifold. (a) GMM encoding in a single tangent space (at the origin). (b) The proposed GMM, where the covariances are computed in the tangent spaces of the means. On the left-hand figures, the contours of the covariances are projected on the manifold. The right-hand figures show the projections of the data into the tangent spaces considered in the computation of the GMM. We can see that representing the local dispersion of the data as covariances in the tangent spaces of the means [in (b)] results in a much better fit than representing the GMM in a single tangent space [in (a)].

(EM) [37], where the Gauss–Newton procedure presented previously is performed in the M step.

Figure 5(a) shows that a GMM computed in a single tangent space (here, at the origin of the manifold) introduces distortions resulting in poor modeling of the data. Figure 5(b) illustrates that the proposed representation limits the distortions by encoding the local spread of the data in covariance matrices expressed in different tangent spaces (i.e., at the centers of the Gaussians). An example application with links to robotics is [35], where human poses are modeled using a GMM on $\mathcal{S}^d$. MATLAB examples `demo_Riemannian_Sd_GMM*.m` can be found in [9].

Gaussian Conditioning

As detailed in [4], we consider input and output data jointly encoded as a multivariate Gaussian $\mathcal{N}_M(\boldsymbol{\mu}, \Sigma)$ partitioned with symbols ^{I} and ^{O} (input and output). Given an input data point x^I, the conditional distribution $\boldsymbol{x}^O|\boldsymbol{x}^I \sim \mathcal{N}_M(\hat{\boldsymbol{\mu}}^O, \hat{\Sigma}^O)$ can be locally evaluated by iterating

$$\boldsymbol{u} = \mathrm{Log}_{\hat{\mu}^O}(\boldsymbol{\mu}^O) - \Sigma_{\parallel}^{OI}\,\Sigma_{\parallel}^{I-1}\,\mathrm{Log}_{x^I}(\boldsymbol{\mu}^I),\ \hat{\boldsymbol{\mu}}^O \leftarrow \mathrm{Exp}_{\hat{\mu}^O}(\boldsymbol{u}),$$

with $\Sigma_{\parallel}$ being a covariance matrix transported from $[\boldsymbol{\mu}^{I\top}, \boldsymbol{\mu}^{O\top}]^\top$ to $[\boldsymbol{x}^{I\top}, \hat{\boldsymbol{\mu}}^{O\top}]^\top$ (see the "Riemannian Geometry in Robotics" section for a description of parallel transport). After convergence, the covariance is computed in the tangent space as $\hat{\Sigma}^O = \Sigma_{\parallel}^{O} - \Sigma_{\parallel}^{OI}\,\Sigma_{\parallel}^{I-1}\Sigma_{\parallel}^{IO}$. MATLAB examples `demo_Riemannian_Sd_GMR*.m` can be found in [9].

Fusion With Products of Gaussians

As shown in [36] and [38], the product of K Gaussians on a Riemannian manifold can be locally evaluated by iterating

$$\boldsymbol{u} = \left(\sum_{k=1}^{K} \boldsymbol{\Sigma}_{\parallel k}^{-1}\right)^{-1} \mathrm{Log}_{\mu}(\boldsymbol{\mu}_k),\quad \boldsymbol{\mu} \leftarrow \mathrm{Exp}_{\mu}(\boldsymbol{u}),$$

with covariance matrix $\Sigma_{\parallel k}$ transported from $\boldsymbol{\mu}_k$ to $\boldsymbol{\mu}$ (see the "Riemannian Geometry in Robotics" section for the description of parallel transport). After convergence, the covariance is computed in the tangent space as $\Sigma = (\Sigma_{k=1}^{K}\Sigma_{\parallel k}^{-1})^{-1}$. An example of the product of Gaussians on $\mathcal{S}^2$ is depicted in Figure 1(b). A MATLAB example `demo_Riemannian_Sd_GaussProd01.m` can be found in [9].

Model Predictive Control

MPC is widely employed in robotics as an adaptive control strategy with anticipation capability. It consists of estimating a series of control commands $\boldsymbol{u}$ across a moving time window of size $T-1$. The problem is described here as an LQT problem with velocity commands $\boldsymbol{u}_t \in \mathbb{R}^d$ and an evolution of the state $\boldsymbol{x}_t \in \mathbb{R}^d$ described by a linear system $\boldsymbol{x}_{t+1} = \boldsymbol{A}_t\boldsymbol{x}_t + \boldsymbol{B}_t\boldsymbol{u}_t$. However, the approach can be generalized to other controllers. The resulting controller is

$$\begin{aligned}\hat{\boldsymbol{u}} &= \underset{\boldsymbol{u}}{\mathrm{argmin}} \left\| \boldsymbol{x} - \boldsymbol{\mu} \right\|_{\boldsymbol{Q}}^2 + \left\| \boldsymbol{u} \right\|_{\boldsymbol{R}}^2 \\ &= (\boldsymbol{S}_u^\top \boldsymbol{Q}\boldsymbol{S}_u + \boldsymbol{R})^{-1}\boldsymbol{S}_u^\top \boldsymbol{Q}(\boldsymbol{\mu} - \boldsymbol{S}_x\boldsymbol{x}_1),\end{aligned} \tag{1}$$

with $\boldsymbol{x} = [\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top, \ldots, \boldsymbol{x}_T^\top]^\top \in \mathbb{R}^{dT}$ being the evolution of the state variable, $\boldsymbol{u} = [\boldsymbol{u}_1^\top, \boldsymbol{u}_2^\top, \ldots, \boldsymbol{u}_{T-1}^\top]^\top \in \mathbb{R}^{d(T-1)}$ the evolution of the control variable, and d the dimension of the state space. Here, $\boldsymbol{\mu} = [\boldsymbol{\mu}_1^\top, \boldsymbol{\mu}_2^\top, \ldots, \boldsymbol{\mu}_T^\top]^\top \in \mathbb{R}^{dT}$ represents the

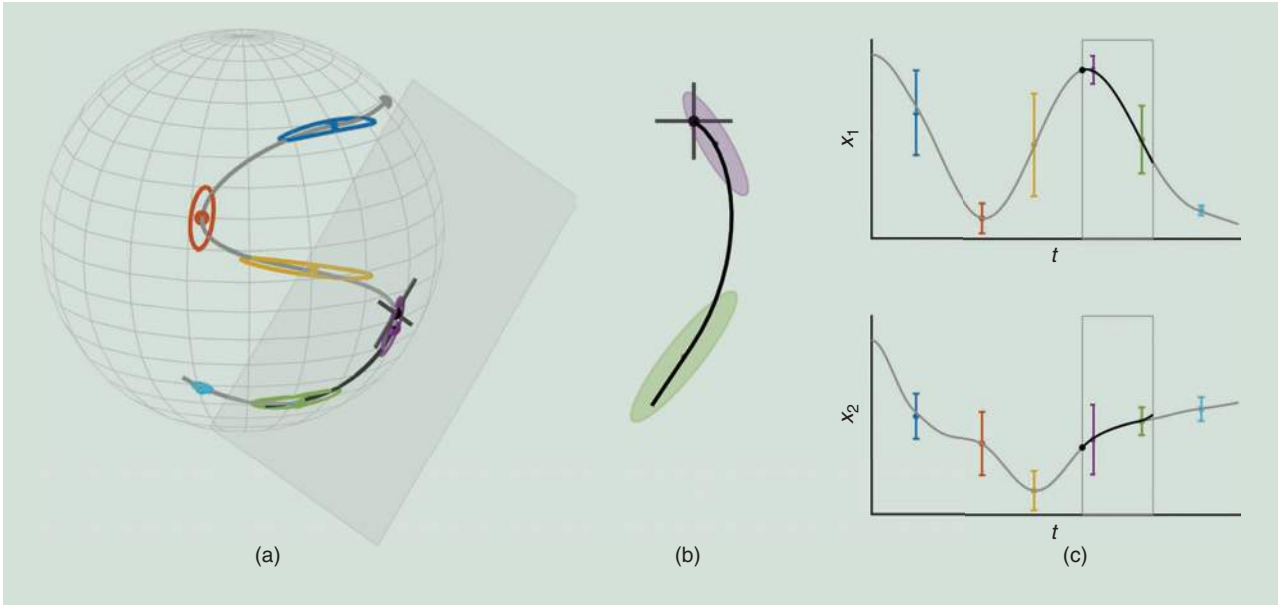


Figure 6. MPC on an $\mathcal{S}^2$ manifold, with a set of via points defined by a GMM. (a) The final movement generated by MPC (in gray) superposed with the partial movement (in black) for the given time horizon (1/5 of the total duration) predicted at 3/5 of the trajectory. (b) The visualization in the tangent space of $\boldsymbol{x}$, where only two Gaussians appear within the current time horizon. (c) The timeline plot showing the evolution of the first two variables of the state space, where the time horizon is depicted as a gray box. The reference to track is represented as a set of colored via points with desired error margins (represented as standard deviations).

evolution of the reference to track, $\boldsymbol{Q} = \text{blockdiag}(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_T) \in \mathbb{R}^{dT \times dT}$ signifies the evolution of the required tracking precision, and $\boldsymbol{R} = \text{blockdiag}(\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_{T-1}) \in \mathbb{R}^{d(T-1) \times d(T-1)}$ indicates the evolution of the cost on the control inputs. In (1), $\boldsymbol{S_u}$ and $\boldsymbol{S_x}$ are transfer matrices; see the supplementary material for this article in IEEE *Xplore* for details of the computation. This formulation corresponds to a basic form of MPC in the Euclidean space by considering quadratic objective functions and linear systems with velocity commands and position states. We showed in [39] that the reference signal to be tracked can be represented by a GMM to form a stepwise trajectory.

Equation (1) is typically used to compute a series of control commands that are reevaluated at each iteration across a time window. Thus, only the first (few) commands are used in practice. In the previous formulation, the first time step of this moving time window corresponds to the current time step in which the problem is solved [see Figure 6(c) for an illustration of this moving time window and the computed control commands within this time window].

Such an MPC/LQT problem can be extended to Riemannian manifolds by exploiting the tangent space of the state x_1 (the point that will introduce the least distortions). By extension of (1), we can solve at each iteration

$$\begin{aligned}\hat{\boldsymbol{u}} &= \underset{\boldsymbol{u}}{\text{argmin}} \left\| \text{Log}_{x_1}(\boldsymbol{x}) - \text{Log}_{x_1}(\boldsymbol{\mu}) \right\|^2_{\boldsymbol{Q}_{\|}} + \left\| \boldsymbol{u} \right\|^2_{\boldsymbol{R}} \\ &= (\boldsymbol{S}_{\boldsymbol{u}}^{\top} \boldsymbol{Q}_{\|} \boldsymbol{S}_{\boldsymbol{u}} + \boldsymbol{R})^{-1} \boldsymbol{S}_{\boldsymbol{u}}^{\top} \boldsymbol{Q}_{\|} \text{Log}_{x_1}(\boldsymbol{\mu}), \end{aligned} \tag{2}$$

where the vector $\hat{\boldsymbol{u}}$ is composed of T–1 commands expressed in the tangent space of $\boldsymbol{x}_1$; $\text{Log}_{x_1}(\boldsymbol{x})$ and $\text{Log}_{x_1}(\boldsymbol{\mu})$ are vectors respectively composed of T elements $\text{Log}_{x_1}(\boldsymbol{x}_t)$ and $\text{Log}_{x_1}(\boldsymbol{\mu}_{s_t})$, with $\{s_t\}_{t=1}^{T}$ the sequence of Gaussian identifiers used to build the stepwise reference trajectory from the GMM; and $\boldsymbol{Q}_{\|}$ is a matrix composed of the block-diagonal elements $\boldsymbol{Q}_{\|t} = \Sigma_{i=1}^{d} \Gamma_{\boldsymbol{\mu}_{s_t} \to \boldsymbol{x}}(\boldsymbol{v}_i) \Gamma^{\top}_{\boldsymbol{\mu}_{s_t} \to \boldsymbol{x}}(\boldsymbol{v}_i)$, using the eigen decomposition $\boldsymbol{Q}_{s_t} = \Sigma_{i=1}^{d} \boldsymbol{v}_i \boldsymbol{v}_i^{\top}$. This transport operation can equivalently be expressed as a linear mapping $\boldsymbol{Q}_{\|t} = \boldsymbol{M}_{\|} \boldsymbol{Q}_{s_t} \boldsymbol{M}_{\|}^{\top}$. In the preceding formulation, $\boldsymbol{R}$ is assumed to be isotropic and thus does not need to be transported.

The first velocity command in (2) (denoted by $\hat{\boldsymbol{u}}_{1:d}$) is then used to update the state with

$$\boldsymbol{x}_1 \leftarrow \text{Log}_{x_1}(\boldsymbol{B}_1 \hat{\boldsymbol{u}}_{1:d}), \tag{3}$$

where $\boldsymbol{B}_1$ corresponds to the linear system $\boldsymbol{x}_2 = \boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{B}_1 \boldsymbol{u}_1$ at the first time step of the time window.

Figure 6 gives an example on $\mathcal{S}^2$, where the computations in (2) and (3) are repeated at each time step to reproduce a movement (with the reference to track encoded as a GMM). Extensions to more elaborated forms of MPC follow a similar principle. A MATLAB example `demo_Riemannian_Sd_MPC01.m` can be found in [9].

Examples of Applications

The operations presented in the previous sections (mixture modeling, conditioning, and fusion) can be combined in different ways, which is showcased in the following sections by two example applications.

Control of Prosthetic Hands With GM Regression

The Gaussian conditioning approach presented in the "Gaussian Conditioning" section can be extended to the GMM approach presented in the "Gaussian Mixture Model" section. The resulting approach is Gaussian mixture regression (GMR), a simple nonlinear regression technique that does not model the regression function directly but, instead, first models the joint probability density of input–output data in the form of a GMM [37], [39]. GMR provides a fast regression approach in which multivariate output distributions can be computed in an online manner, with a computation time independent of the number of data points used to train the model, by exploiting the learned joint density model. In GMR, the inputs and outputs can be multivariate, and, after learning, any subset of input–output dimensions can be selected for regression.

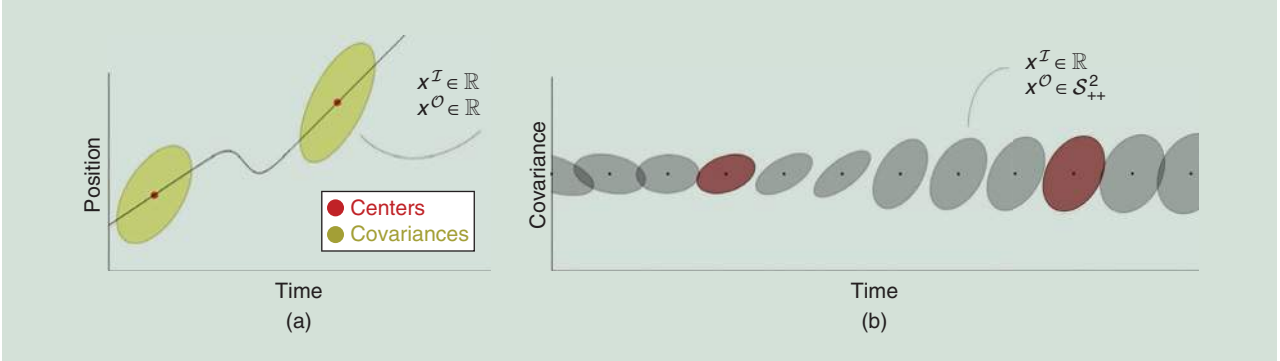


Figure 7. The GMR on the SPD manifold. (a) The classical use of GMR to encode trajectories, with time as input and the position as output (both in the Euclidean space). (b) The extension to Riemannian manifolds with outputs on the SPD manifold. This nonlinear regression approach provides a conditional estimate of the output expressed in the form of matrix–variate Gaussians. Covariances of SPD data points are not depicted here (fourth-order tensors).

This is exploited in robotics to handle different sources of missing data where expectations for the remaining dimensions can be computed as a multivariate distribution. These properties make GMR an attractive tool for robotics that can be used in a wide range of problems and combined fluently with other techniques [39].

The authors of [35] and [40] present methods for regression from a mixture of Gaussians on Riemannian manifolds, but they only partially exploit the manifold structure in Gaussian conditioning. In [35], each distribution is located on its own tangent space, with the covariances encoded separately, resulting in a block-diagonal structure in the joint distribution. In [40], a GMM is reformulated to handle the space of rotation in $\mathbb{R}^3$ using logarithms and exponential transformations on unit quaternions, with these operations formulated in a single tangent space (at the origin) instead of applying the transformations locally (see Figure 5). The link to Riemannian manifolds also is not discussed.

Here, it is proposed to extend GMR to input and/or output data on SPD manifolds; see Figure 7. As the covariance of SPD data points is a fourth-order tensor, a method is proposed in [5] for the parallel transport of high-order covariances on SPD manifolds by exploiting the supersymmetry properties of these fourth-order tensors. As an example application, GMR on an SPD manifold is applied to predict wrist movement from spatial covariances computed from sEMG data. In this application, the GMR input data are spatial covariances that belong to the SPD manifold. Compared to the Euclidean GMR, the GMR on an SPD

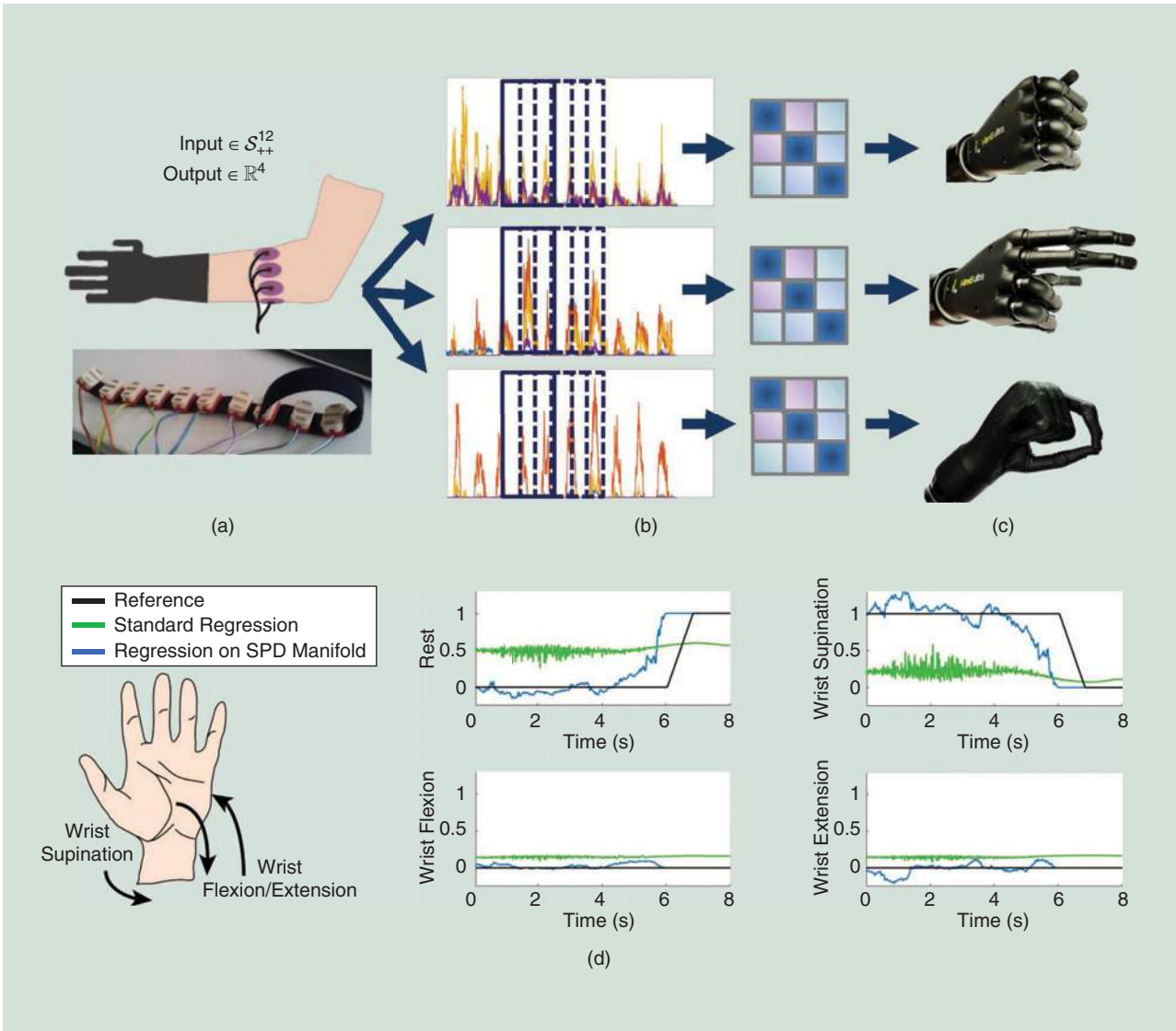


Figure 8. GMR for the control of prosthetic hands within the TACT-HAND project. The (a) sEMG measurements, (b) transformation in spatial covariances (SPD matrices), and (c) control of the corresponding hand pose. SPD signals are used as input in the form of spatial covariances computed from sEMG sensors on participants' forearm. Activation signals corresponding to different hand poses are used as outputs. (d) In this experiment (see [5] for details), taking the geometry of the data into account in GMR (in blue) results in better discrimination than treating the data as if they were in a Euclidean space (in green).

manifold improved the detection of wrist movement for most participants and proved to be efficient at locating transitions between movements; see Figure 8 and Table 1 for a summary of the results. This shows the importance and benefits of considering the underlying manifold structure of the data in this application. The details of this experiment can be found in [5].

Underwater Robot Teleoperation With Task-Parameterized GMM

The fusion approach presented in the "Fusion With Products of Gaussians" section can be extended to the GMM approach presented in the "Gaussian Mixture Model" section. This is particularly useful when mixtures of Gaussians are encoded in different coordinate systems that need to be fused at reproduction time to satisfy constraints in multiple frames of reference. Within the DexROV project [41], this task-parameterized GMM (TP-GMM) approach [4], [39] is used with the MPC approach presented in the "Model Predictive Control" section to teleoperate an underwater robot at a distance, with a teleoperator wearing an exoskeleton and visualizing a copy of the robot workspace in a virtual environment.

Figure 9 presents an overview of this application (also see [41] for a description of this teleoperation approach and [39] for a general description of TP-GMM). Because of the long communication delays between the teleoperator and the robot, the locations of the objects and tools of interest are not the same on the teleoperator and robot sides. Using a parameterization associated with the locations of objects and tools, we can cope with this discrepancy by locally adapting the movement representation to the position and orientation of the objects/tools, represented as coordinate systems. Figure 9 depicts an example with two coordinate systems (with models represented in orange and purple) corresponding, respectively, to the robot and a valve that needs to be turned. A motion relative to the valve and the robot is encoded as GMMs in the two respective coordinate systems. During teleoperation, each pair of GMMs is rotated and translated according to the current situations on the teleoperator and robot sides. Products of Gaussians are then computed at each side to fuse these representations.

Table 1. The wrist movement detection results.

	Rest	Wrist Supination	Wrist Extension	Wrist Flexion
$\mathcal{S}^D_{++}$	0.29 ± 0.00	0.18 ± 0.00	0.25 ± 0.00	0.27 ± 0.00
$\mathbb{R}$	0.47 ± 0.00	0.31 ± 0.00	0.33 ± 0.00	0.33 ± 0.00
$\mathcal{S}^D_{++}$	0.32 ± 0.02	0.29 ± 0.14	0.36 ± 0.07	0.43 ± 0.13
$\mathbb{R}$	0.46 ± 0.00	0.34 ± 0.00	0.35 ± 0.00	0.35 ± 0.00
$\mathcal{S}^D_{++}$	0.36 ± 0.02	0.22 ± 0.00	0.31 ± 0.00	0.29 ± 0.00
$\mathbb{R}$	0.42 ± 0.00	0.42 ± 0.00	0.43 ± 0.00	0.43 ± 0.00

The table compares the root-mean-square error obtained by GMR on the SPD manifold and the standard Euclidean GMR for wrist motion estimation from sEMG (see [5] for details). The results are presented for three participants.

Movements are encoded in this way with position $\mathbb{R}^3$ and orientation $\mathcal{S}^3$ data (in Figure 9, a representation with $\mathbb{R}^2$ and $\mathcal{S}^2$ is shown as an illustration). For the position, the retrieved path in black (and the associated covariances) correspond to a movement of the robot to the valve (represented as red U shapes) by taking into account how these different coordinate systems are oriented. We can see that the approaching phase is perpendicular to the coordinate system to properly reach the valve. For the orientation, the retrieved path shows how the orientation of the end effector changes with time. At the beginning of the motion, this orientation relates to that of the robot; at the end, the orientation of the end effector matches that of the valve.

In these graphs, the purple and orange ellipsoids depict two GMMs representing uncertain trajectories with respect to two different frames of reference (red U shapes for position data and red points on the spheres for orientation data). The black ellipsoids indicate the final trajectory and its uncertainty, obtained by fusing the trajectories of the two different frames of reference through products of Gaussians. Although the red U shapes and points are not the same on the teleoperator and robot sides, the retrieved paths on the two sides can quickly adapt to these different situations. Using a Riemannian manifold framework,

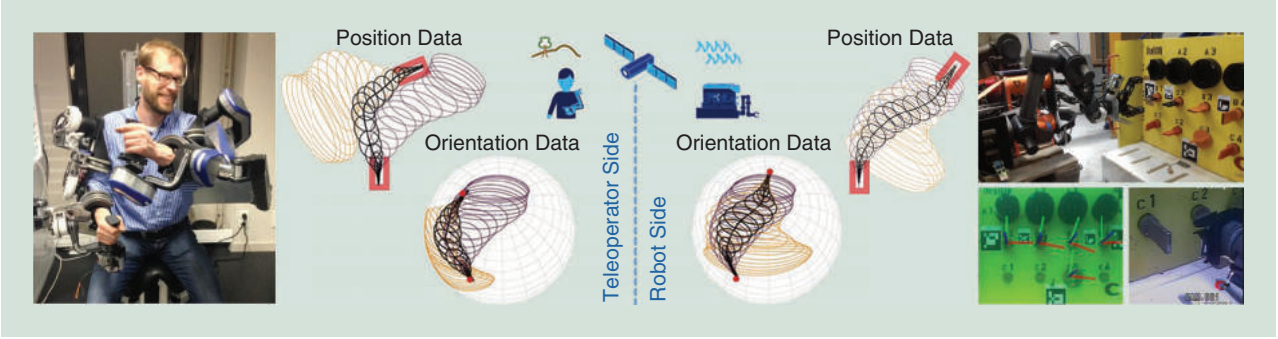


Figure 9. The TP-GMM extended to $\mathcal{S}^d$ manifolds in the DexROV project.

orientations are encoded uniquely in a representation that does not contain singularities. Such an approach is employed in this application on $\mathcal{S}^3$ to learn and retrieve the evolution of robot end-effector orientations by adapting them to the orientation of objects and tools in the robot workspace. This approach was successfully tested during field trials in the Mediterranean Sea, offshore of Marseille, France, where seven extended dives at four sites (8-, 30-, 48-, and 100-m water depths) were performed with the underwater robot while connected via satellite to the teleoperation center in Brussels, Belgium; see [41] for a general description of the experiment.

Further Perspectives and Conclusions

This article showed that a wide range of challenges in robot learning and adaptive control can be recast as statistical modeling and information fusion on Riemannian manifolds. Such an interpretation can avoid potential misuses of algorithms in robotics that might originate from Riemannian geometry and that are treated with a limited view. One such example is to perform all computations in a single tangent space (typically, at the origin of the manifold) instead of considering the closest tangent spaces to avoid distortions. Another example concerns domain adaptation and transfer learning, which require the realignment of data to cope with nonstationarities. For example, sensory data collected by different subjects or throughout several days should use the Riemannian notion of parallel transport instead of only recentering the data [42].

This article also showed that the combination of statistics and differential geometry offers many research opportunities and can contribute to recent challenges in robotics. Further work can be organized in two categories. First, the field of robotics is abundant with new techniques proposed by researchers because of its interdisciplinary aspect and the richness of the problems it involves. The common factor in many of these developments is that they rely on some form of statistics and/or propagation of uncertainty. These models and algorithms are typically developed for standard Euclidean spaces, where an extension to Riemannian manifolds has several benefits to offer.

Second, some Riemannian manifolds remain largely underexploited in robotics despite the fact that they are mathematically well understood and characterized by simple closed-form expressions. Grassmann manifolds seem particularly promising to handle problems in robotics with high-dimensional data points and only few training data, where subspaces are required in the computation to keep the most essential characteristics of the data. They are also promising in problems in which hierarchies are considered (such as inverse kinematics with kinematically redundant robots) because they provide a geometric interpretation of null-space structures. Other Riemannian manifolds, such as hyperbolic manifolds, also seem propitious for bringing a probabilistic treatment to dynamical systems, tree-based structures, graphs, Toeplitz/Hankel matrices, and autoregressive models. Finally, a wide range of metric learning problems in robotics could benefit from a Riemannian geometry treatment.

Acknowledgments

I would like to thank Noémie Jaquier, who provided relevant suggestions for the writing and organization of the article and carefully proofread the manuscript. This work was supported by the Swiss National Science Foundation/German Research Foundation project TACT-HAND and by the European Commission's Horizon 2020 program (Memory of Motion project, http://www.memmo-project.eu/, grant 780684; and DexROV project, http://www.dexrov.eu/, grant 635491).

References

[1] F. C. Park, J. E. Bobrow, and S. R. Ploen, "A lie group formulation of robot dynamics," *Int. J. Robotics Res.*, vol. 14, no. 6, pp. 609–618, 1995. doi: 10.1177/027836499501400606.

[2] J. M. Selig, *Geometric Fundamentals of Robotics*. Berlin: Springer-Verlag, 2005.

[3] T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems," *IEEE Trans. Robotics*, vol. 30, no. 3, pp. 679–693, June 2014. doi: 10.1109/TRO.2014.2298059.

[4] M. J. A. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on Riemannian manifolds," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1240–1247, June 2017. doi: 10.1109/LRA.2017.2657001.

[5] N. Jaquier and S. Calinon, "Gaussian mixture regression on symmetric positive definite matrices manifolds: Application to wrist motion estimation with sEMG," in *Proc. IEEE/RSJ Intl Conf. Intelligent Robots and Systems (IROS)*, Vancouver, Canada, Sept. 2017, pp. 59–64. doi: 10.1109/IROS.2017.8202138.

[6] T. Lee and F. C. Park, "A geometric algorithm for robust multibody inertial parameter identification," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2455–2462, 2018. doi: 10.1109/LRA.2018.2799426.

[7] S. Traversaro, S. Brossette, A. Escande, and F. Nori, "Identification of fully physical consistent inertial parameters using optimization on manifolds," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 5446–5451. doi: 10.1109/IROS.2016.7759801.

[8] T. Yoshikawa, "Manipulability of robotic mechanisms," *Int. J. Robotics Res.*, vol. 4, no. 2, pp. 3–9, 1985. doi: 10.1177/027836498500400201.

[9] "PbDlib robot programming by demonstration software library," Idiap Research Institute, Martigny, Switzerland, 2020. Accessed on: Mar. 10, 2020. [Online]. Available: http://www.idiap.ch/software/pbdlib/

[10] X. Pennec, "Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements," *J. Math. Imag. Vision*, vol. 25, no. 1, pp. 127–154, 2006. doi: 10.1007/s10851-006-6228-4.

[11] P. A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ: Princeton Univ. Press, 2007.

[12] S. Arimoto, M. Yoshida, M. Sekimoto, and K. Tahara, "A Riemannian-geometry approach for modeling and control of dynamics of object manipulation under constraints," *J. Robotics*, vol. 2009, pp. 1–16, Mar. 2009, Art. no. 892801. doi: 10.1155/2009/892801.

[13] A. Biess, T. Flash, and D. G. Liebermann, "Riemannian geometric approach to human arm dynamics, movement optimization, and invariance," *Phys. Rev. E*, vol. 85, no. 1, pp. 031927, Mar 2011. doi: 10.1103/PhysRevE.85.019907.

[14] N. Hosni, H. Drira, F. Chaieb, and B. Ben Amor, "3D gait recognition based on functional PCA on Kendall's shape space," in *Proc. Int. Conf. Pattern Recognition (ICPR)*, Aug. 2018, pp. 2130–2135. doi: 10.1109/ICPR.2018.8545040.

[15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 1–21, Feb. 2017. doi: 10.1109/TRO.2016.2597321.

[16] J. R. Forbes and D. E. Zlotnik, "Sigma point Kalman filtering on matrix Lie groups applied to the SLAM problem," in *Geometric Science of Information (GSI)*, F. Nielsen and F. Barbaresco, Eds. Cham, Switzerland: Springer-Verlag, 2017, pp. 318–328.

[17] S. Brossette, A. Escande, and A. Kheddar, "Multicontact postures computation on manifolds," *IEEE Trans. Robotics*, vol. 34, no. 5, pp. 1252–1265, Oct. 2018. doi: 10.1109/TRO.2018.2830390.

[18] M. Zefran and V. Kumar, "Planning of smooth motions on SE(3)," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Apr. 1996, pp. 121–126. doi: 10.1109/ROBOT.1996.503583.

[19] N. Jaquier, L. Rozo, D. G. Caldwell, and S. Calinon, Geometry-aware manipulability learning, tracking and transfer. 2018. [Online]. Available: *arXiv:1811.11050*

[20] S. Hauberg, O. Freifeld, and M. J. Black, "A geometric take on metric learning," in *Proc. Advances Neural Information Processing Systems (NIPS)*, 2012, pp. 2024–2032.

[21] M. Zucker et al., "CHOMP: Covariant Hamiltonian optimization for motion planning," *Int. J. Robotics Res.*, vol. 32, nos. 9–10, pp. 1164–1193, 2013. doi: 10.1177/0278364913488805.

[22] C.-A. Cheng et al., RMPflow: A computational graph for automatic motion policy generation. 2018. [Online]. Available: *arXiv:1811.07049*

[23] E. Chevallier, F. Barbaresco, and J. Angulo, "Probability density estimation on the hyperbolic space applied to radar processing," in *Geometric Science of Information (GSI),* F. Nielsen and F. Barbaresco, Eds. Springer-Verlag, 2015, pp. 753–761.

[24] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *Int. J. Robotics Res.*, vol. 20, no. 5, pp. 378–400, 2001. doi: 10.1177/02783640122067453.

[25] K. Usevich and I. Markovsky, "Optimization on a Grassmann manifold with application to system identification," *Automatica*, vol. 50, no. 6, pp. 1656–1662, 2014. doi: 10.1016/j.automatica.2014.04.010.

[26] R. Slama, H. Wannous, M. Daoudi, and A. Srivastava, "Accurate 3D action recognition using learning on the Grassmann manifold," *Pattern Recogn.*, vol. 48, no. 2, pp. 556–567, 2015. doi: 10.1016/j.patcog.2014.08.011.

[27] A. D. Wilson, J. A. Schultz, and T. D. Murphey, "Trajectory synthesis for Fisher information maximization," *IEEE Trans. Robotics*, vol. 30, no. 6, pp. 1358–1370, 2014. doi: 10.1109/TRO.2014.2345918.

[28] S. Amari, *Information Geometry and Its Applications*. Tokyo: Springer-Verlag, 2016.

[29] A. Rajeswaran et al., "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Proc. Robotics: Science and Systems (RSS)*, Pittsburgh, PA, June 2018.

[30] G. Arvanitidis, L. K. Hansen, and S. Hauberg, "Latent space oddity: On the curvature of deep generative models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2018, pp. 1–15.

[31] N. Chen et al., "Fast approximate geodesics for deep generative models," in *Proc. Int. Conf. Artificial Neural Networks (ICANN)*, 2019, pp. 554–566. doi: 10.1007/978-3-030-30484-3_45.

[32] N. Sharp, Y. Soliman, and K. Crane, "The vector heat method," *ACM Trans. Graph.*, vol. 38, no. 3, pp. 24:1–24:19, 2019. doi: 10.1145/3243651.

[33] J. Solà, J. Deray, and D. Atchuthan, A micro Lie theory for state estimation in robotics. 2019. [Online]. Available: *arXiv:1812.01537*

[34] S. Said, L. Bombrun, Y. Berthoumieu, and J. H. Manton, "Riemannian Gaussian distributions on the space of symmetric positive definite matrices," *IEEE Trans. Inform. Theory*, vol. 63, no. 4, pp. 2153–2170, 2017. doi: 10.1109/TIT.2017.2653803.

[35] E. Simo-Serra, C. Torras, and F. Moreno-Noguer, "3D human pose tracking priors using geodesic mixture models," *Int. J. Comput. Vis.*, vol. 122, no. 2, pp. 388–408, 2017. doi: 10.1007/s11263-016-0941-2.

[36] M. J. A. Zeestraten, I. Havoutis, S. Calinon, and D. G. Caldwell, "Learning task-space synergies using Riemannian geometry," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Vancouver, Canada, Sept. 2017, pp. 73–78. doi: 10.1109/IROS.2017.8202140.

[37] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an EM approach," in *Advances in Neural Information Processing Systems (NIPS)*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. San Francisco: Morgan Kaufmann, 1994, pp. 120–127.

[38] M. J. A. Zeestraten, I. Havoutis, and S. Calinon, "Programming by demonstration for shared control with an application in teleoperation," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1848–1855, July 2018. doi: 10.1109/LRA.2018.2805105.

[39] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intell. Serv. Robot.*, vol. 9, no. 1, pp. 1–29, Jan. 2016. doi: 10.1007/s11370-015-0187-9.

[40] S. Kim, R. Haschke, and H. Ritter, "Gaussian mixture model for 3-DoF orientations," *Robotics Autonomous Syst.*, vol. 87, pp. 28–37, 2017. doi: 10.1016/j.robot.2016.10.002.

[41] A. Birk et al., "Dexterous underwater manipulation from onshore locations: Streamlining efficiencies for remotely operated underwater vehicles," *IEEE Robot. Autom. Mag.*, vol. 25, no. 4, pp. 24–33, 2018. doi: 10.1109/MRA.2018.2869523.

[42] O. Yair, M. Ben-Chen, and R. Talmon, "Parallel transport on the cone manifold of SPD matrices for domain adaptation," *IEEE Trans. Signal Process.*, vol. 67, no. 7, pp. 1797–1811, Apr 2019. doi: 10.1109/TSP.2019.2894801.

Sylvain Calinon, Idiap Research Institute, Martigny, Switzerland. Email: sylvain.calinon@idiap.ch.

Interactive Learning of Temporal Features for Control

By Rodrigo Pérez-Dattari, Carlos Celemin, Giovanni Franzese,
Javier Ruiz-del-Solar, and Jens Kober

Current ongoing industry revolution demands more flexible products, including robots in household environments and medium-scale factories. Such robots should be able to adapt to new conditions and environments and be programmed with ease. As an example, let us suppose that there are robot manipulators working on an industrial production line and that they need to perform a new task. If these robots were hard coded, it could take days to adapt them to the new settings, which would stop production at the factory. Robots that non-expert humans could easily program would speed up the process considerably.

In this regard, we present a framework in which robots are capable of quickly learning new control policies and state representations (SRs) by using occasional corrective human

Digital Object Identifier 10.1109/MRA.2020.2983649
Date of current version: 22 April 2020

1070-9932/20©2020IEEE

feedback. To achieve this, we focus on robots interactively learning these policies from non-expert humans who act as teachers. We present a neural network (NN) architecture along with an interactive imitation learning (IIL) method that efficiently learns spatiotemporal features and policies from raw high-dimensional observations (raw pixels from an image) for tasks in environments that are not fully temporally observable.

We denominate IIL as a branch of IL, where human teachers provide different kinds of feedback to robots, such as new demonstrations triggered by robot queries [1], corrections [2], preferences [3], reinforcements [4], and so forth. Most IL methods work under the assumption of learning from perfect demonstrations; therefore, they fail when teachers have only partial insights about the task execution. Non-expert teachers could include all users who are neither machine learning/control experts nor skilled enough to fully show the desired behavior of the policy.

Interactive approaches, such as COACH (which is the short form of Corrective Advice Communicated by Humans) [5], and some interactive reinforcement learning (IRL) approaches [4], [6] are intended for non-expert teachers but are not completely deployable for end users. Sequential decision-making learning methods (IL, IIL, IRL, and so forth) rely on good SRs, which simplify the shaping of the policy landscape and provide suitable generalization properties. However, this requirement means that experts on feature engineering must preprocess the states properly before running the learning algorithms.

The inclusion of deep learning (DL) in IL (given the popularity DL has gained in the field of RL [7]) enables practitioners to skip preprocessing modules for inputting policies since some NN architectures endow the agents with intrinsic feature-extraction capabilities. This has been exhaustively tested in end-to-end settings [7]. In this regard, DL enables non-expert humans to shape policies based only on their feedback.

Nevertheless, in real-world problems, we commonly face tasks wherein the observations do not explain the complete state of the agent, due to the lack of temporal information (e.g., rates of change) or because the agent–environment interaction is non-Markovian (e.g., dealing with occlusions). For these cases, it is necessary to provide memory to the learning policy. Recurrent NNs (RNNs) can learn to model dependencies from past observations and map them to the current outputs. This recurrency has been used in RL and IL, mostly through long short-term memory (LSTM) networks [8].

Therefore, LSTMs are included in our NN architecture to learn temporal features, which contain relevant information from the past. However, DL algorithms require large amounts of data; as a way to tackle this shortcoming, SR learning (SRL) has been used to learn features more efficiently [9], [10]. Considering that real robots and human users have time limitations, as an SRL strategy, a model of the world is learned to obtain SRs that make the policy convergence possible within feasible training time intervals (see Figure 1). The combination of SRL and the teacher's feedback is a powerful strategy for efficient learning of temporal features from raw observations in non-Markovian environments.

The experiments presented in this article show the impact of the proposed architecture in terms of data efficiency and the policy's final performance within the deep COACH (D-COACH) IIL framework [11]. Additionally, the experimental procedure demonstrates that the proposed architecture could even be used with other IL methods, such as data aggregation (DAgger) [12]. The code used in this paper can be found at https://github.com/rperezdattari/Interactive-Learning-of-Temporal-Features-for-Control.

Background and Related Work

Our method combines elements from SRL, IL, and memory in NN models to build a framework that enables non-expert teachers to interactively shape policies in tasks with non-Markovian environments. These elements are introduced in the following.

Dealing With Non-Markovian Environments

There are different reasons why a process could be partially observable. One is when the state describes time-dependent phenomena, but the observation contains only partial information about them. For instance, velocities cannot be estimated from camera images unless observations from different time steps are combined. Other examples of time-dependent phenomena are temporary occlusions and corrupted communication systems between the sensors and the agent.

For these environments, the temporal information needs to be implicitly obtained within the policy model. There are two well-known approaches for adding memory to agents in sequential decision-making problems when using NNs as function approximators:

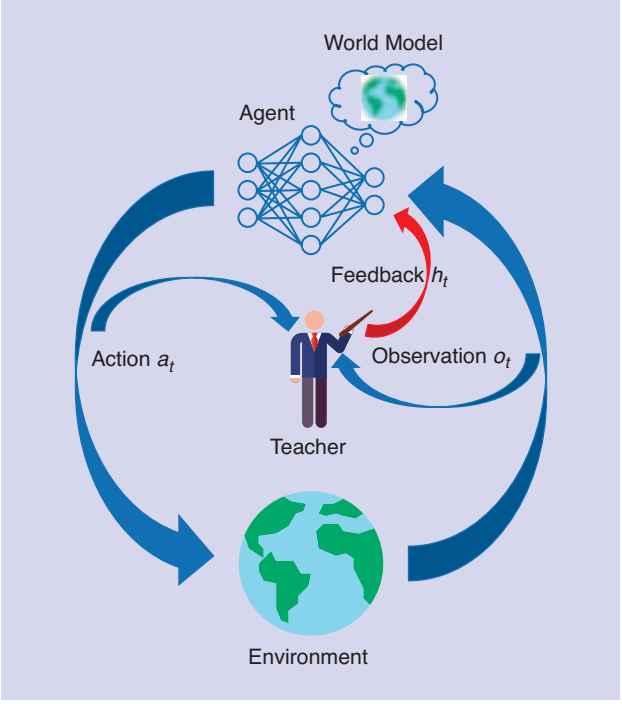


Figure 1. Interactively shaping policies with agents that model the world. (Source: turkkub and Freepik from Flaticon.)

1) *Observation stacking policies* [7]: stacking the last N observations $(o_t, o_{t-1}, \ldots o_N)$ and using this stack as the input of the network
2) *Recurrent policies* [13]: including RNN layers in the policy architecture.

One of the main issues with observation stacking is that the memory of these models is determined by the number of stacked observations. The overhead increases rapidly for larger sequences in high-dimensional observation problems.

In contrast, RNNs can model information for an arbitrarily long period of time [14]. Also, they do not add input-related overheads because, when these models are evaluated, they use only the last observation. Therefore, RNNs have a lower computational cost than observation stacking. Given the more practical usage of recurrent models and their capability of representing arbitrarily long sequences, in this article, we use RNN-based policies (with LSTM layers) in the proposed NN architecture. Nevertheless, the use of LSTMs has a critical disadvantage since their training is more complex and requires more data, something very problematic when considering human teachers and real systems. We now introduce SRL, which helps to accelerate LSTM convergence.

SRL

In most of the problems faced in robotics, the state s_t, which fully describes the situation of the environment at time step t, is not fully accessible from the robot's observation o_t. As mentioned, in several problems these observations lack the temporal information required in the state description. Moreover, these observations tend to be raw sensor measurements that can be high-dimensional, highly redundant, and ambiguous. A portion of this data may even be irrelevant.

As a consequence, to successfully solve these problems a policy needs to 1) find temporal correlations between several consecutive observations and 2) extract relevant features from observations that are hard to interpret. However, finding relations among these large data structures with the underlying phenomena of the environment while also learning controllers can be extremely inefficient. Therefore, efficiently building controllers on top of raw observations requires learning-informative, low-dimensional SRs [15]. The objective of SRL is to obtain an observer capable of generating such representations.

Algorithm 1: (HG-)DAgger

1: **Require**: demonstrations database $\mathcal{D}$ with initial demonstrations, policy update frequency b
2: **for** t = 1, 2, … **do**
3: **if** mod(t, b) is 0 **then**
4: **update** π_θ from $\mathcal{D}$
5: **observe** state o_t
6: **select** action from agent or expert
7: **execute** action
8: **feedback** provide label a_t^* for o_t, if necessary
9: **aggregate** (o_t, a_t^*) to $\mathcal{D}$

A compact representation of a state is considered suitable for control if the resulting SR

- is Markovian
- has good generalization to unseen states
- is defined in low-dimensional space (considerably lower than the actual observation dimensionality) [9].

Along with the control objective function (e.g., reward function and imitation cost function), other objective functions can be used for SRL [10]:

- observation reconstruction
- forward model or next observation prediction
- the inverse model
- the reward function
- the value function.

Interactive Learning Methods

This section briefly introduces two approaches for interactively learning from human teachers while agents are executing a task.

Data Aggregation: DAgger and Human-Gated DAgger

DAgger [12] is an IIL algorithm that aims to collect data through online sampling. To achieve this, trajectories are generated by combining the agent's policy π_θ and the expert's policy. The observations o_t and the demonstrator's corresponding actions a_t^* are paired and added to a database $\mathcal{D}$, which is used for training the policy's parameters θ iteratively in a supervised learning manner to asymptotically approach the expert's policy. At the beginning of the learning process, the demonstrator has all the influence over the trajectory made by the agent; then the probability of following the demonstrator's actions decays exponentially.

For working in real-world systems with humans as demonstrators, a variation of DAgger, human-gated DAgger (HG-DAgger) [2], was introduced. In this approach, the demonstrator is not expected to give labels across every action of the agent but only in places where she/he considers that the agent's policy needs improvement. Only these labels are aggregated to the database and used for updating the policy. Additionally, every time feedback is given by the human, the policy will follow the provided action. As a safety measure, in HG-Dagger, the uncertainty of the policy across the observation space is estimated; that element is omitted in this article. Algorithm 1 shows the general structure of DAgger and HG-DAgger.

D-COACH

In this framework, humans shape policies, giving occasional corrective feedback for actions executed by the agents [11]. The human indicates agent actions that she/he considers to be erroneous through a binary signal h_t, the direction in which the action should be modified. This feedback is used to generate an error signal for updating the policy parameters θ. It is performed in a supervised learning manner, with the cost function J and using the mean squared error and stochastic gradient descent. Hence, the update rule is

$$\theta \leftarrow \theta - \alpha \cdot \nabla_\theta J(\theta). \quad (1)$$

The feedback given by the human indicates only the sign of the policy error. Its magnitude is supposed to be unknown since the algorithm works under the assumption that the user is non-expert; therefore, she/he does not know the magnitude of the proper action. Instead, the error magnitude is defined as the hyperparameter e that must be defined before starting the learning process. Thus, the policy $error_t$ is defined by $h_t \cdot e$.

To compute a gradient in the parameter space of the policy, the error needs to be a function of θ. This is achieved by observing that

$$error_t(\theta) = a_t^{\text{target}} - \pi_\theta(o_t), \quad (2)$$

where a_t^{target} is the incremental objective generated by the feedback of the human $a_t^{\text{target}} = a_t + error_t$, and a_t is the current output of the policy π_θ. From (1), (2), and the derivative of the mean squared error, we can get the COACH update step:

$$\theta \leftarrow \theta + \alpha \cdot error_t \cdot \nabla_\theta \pi_\theta. \quad (3)$$

To be more data efficient and avoid locally overfitting to the most recent corrections, D-COACH has a memory buffer that stores the tuple $(o_t, a_t^{\text{target}})$ and replays this information during learning. Additionally, when working in problems with high-dimensional observations, an autoencoding cost is incorporated in D-COACH as an observation reconstruction SRL strategy. In the D-COACH pseudocode (Algorithm 2), this SRL step is omitted. D-COACH learns everything from scratch through only one interactive phase, unlike other deep interactive RL approaches [4], [6], which split the learning process into two sequential learning phases: first, recording samples of the environment for training a dimensionality reduction model (e.g., an autoencoder) and, second, using that model for the input of the policy network during the actual interactive learning process.

Learning Temporal Features Based on Interactive Teaching and World Modeling

In this section, the SRL NN architecture is described along with the interactive algorithm for policy shaping.

Network Architecture for Extracting Temporal Features

When approaching problems that lack temporal information in the observations, the most common solution is to model the policy with RNNs, as discussed in the "Dealing With Non-Markovian Environments" section; therefore, we propose to shape policies that are built on top of RNNs, with occasional human feedback. In this article, we use the terms *world model* and *transition model* interchangeably.

IIL methods can take advantage of SRL for training with other objective functions by 1) making use of all of the experience collected in every time step and 2) boosting the process of finding compact Markovian embeddings. We propose a neural architecture separated into two parts: 1) the transition model and 2) the policy. The transition model is in charge of learning the dynamics of the environment in a supervised manner, using samples collected by the agent. The policy part is shaped using only corrective feedback. Figure 2 shows this architecture.

Learning to predict the next observation o_{t+1} forces a Markovian SR. This has been successfully applied in RL [16]. RNNs can encode information from past observations in their hidden state h_t^{LSTM}. Thus, the objective of the first part of the NN is to learn $\mathcal{M}(o_t, a_t, h_{t-1}^{\text{LSTM}}) = \tilde{o}_{t+1}$, which, as a consequence, learns to embed past observations in h_t^{LSTM}. Additionally, when the observations are high-dimensional (raw images), the agents also need to learn to compress spatial information. To achieve this, a common approach is to compress this information in the latent space of an autoencoder.

For the first part of the architecture, we propose a combination of an autoencoder with an LSTM to compute the

Algorithm 2: D-COACH

Require: error magnitude e, buffer update interval b
Init: $\mathcal{B} = []$ *# initialize memory buffer*
for t = 1, 2, … **do**
  observe state o_t
  execute action $a_t = \pi_\theta(o_t)$
  feedback human corrective advice h_t
  if h_t is not **0 then**
    $error_t = h_t \cdot e$
    $a_{\text{target}(t)} = a_t + error_t$
    update π using SGD with pair $(o_t, a_t^{\text{target}})$
    update π using SGD with a minibatch sampled from $\mathcal{B}$
    append $(o_t, a_t^{\text{target}})$ to $\mathcal{B}$
  if mod(t, b) is 0 **then**
    update π_θ using SGD with a minibatch sampled from $\mathcal{B}$

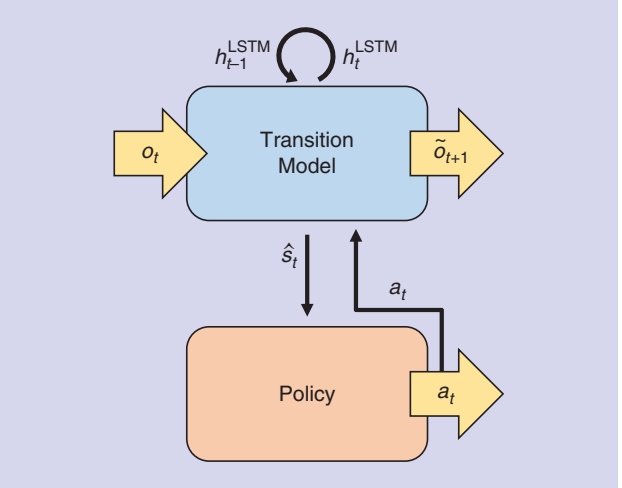


Figure 2. The general structure of the transition model and policy.

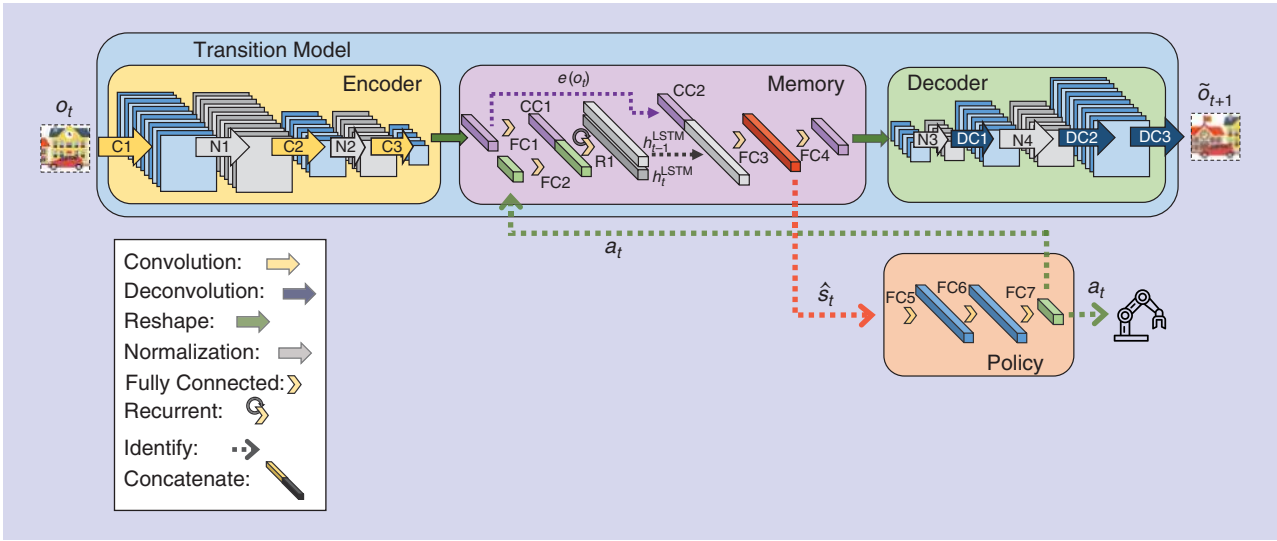


Figure 3. The proposed NN architecture. Convolutional and recurrent (LSTM) layers are included in the transition model for learning of spatiotemporal SRs. The estimated state $\hat{s}_t$ is used as input to the policy, which is a fully-connected NN. C: convolution; N: normalization; FC: fully connected; CC: concatenate; R: recurrent; DC: deconvolution. (Source: Vectors Market, Smashicons, Freepik, and Smalllikeart from Flaticon.)

Algorithm 3: Online temporal feature learning

1: **Require**: Policy update algorithm π^{update}, training sequence length τ, model update rate d
2: **Init**: $\mathcal{D} = []$
3: **for** t = 1, 2, … **do**
4: **observe** observation o_t
5: **append** $(o_{t-1}, \ldots, o_{t-\tau}, a_{t-1}, \ldots, a_{t-\tau}, o_t)$ to $\mathcal{D}$
6: **execute** action $a_t = \pi_\theta(o_t, h_{t-1}^{\text{LSTM}})$
7: **compute** h_t^{LSTM} from $\mathcal{M}(o_t, a_t, h_{t-1}^{\text{LSTM}})$
8: **feedback** human feedback h_t
9: **call** $\pi^{\text{update}}(o_t, a_t, h_t)$
10: **if** mod(t, d) is 0 **then**
11: **update** $\mathcal{M}$ using SGD with minibatches of sequences sampled from $\mathcal{D}$

transition function, i.e., predicting the next high-dimensional observation. A detailed diagram of this architecture can be seen in Figure 3. In the second part of the architecture, the policy takes as input a representation of the state $\hat{s}_t$, which is generated inside the transition model network. This representation is obtained at the output of a fully connected layer (FC3) that combines the information of h_{t-1}^{LSTM} with the encoder compression of the current observation $e(o_t)$. This is achieved by adding a skipping connection between the output of the encoder and that of the LSTM.

Interactive Algorithm for Policy and World-Model Learning

Algorithm 3 presents the pseudocode of the SRL strategy. The hidden state of the LSTM is denoted as h^{LSTM} and the human corrective feedback as h. In every time step, a buffer $\mathcal{D}$ stores the samples of the transitions with sequences of length τ (line 5). The agent executes an action based on its last observation and the current hidden state of the LSTM (line 6). This hidden state is updated using its previous value and the most recent observation and action (line 7). Line 8 captures the occasional feedback of the teacher, which could be a relative correction when using D-COACH or a corrective demonstration when using HG-DAgger. Also, depending on the learning algorithm, the policy is updated in different ways (line 9). $\mathcal{D}$ replays past transitions of the environment to update the transition function model (line 11). This is done following the bootstrapped random updates [13] strategy. This model is updated every d time steps.

Experiments and Results

In this section, we present experiments for validating the proposed NN architecture and the interactive training algorithm. To obtain a thorough evaluation, different experiments are carried out to compare and measure the performance (the return, i.e., the sum of the rewards) of the proposed components. Initially, the network architecture based on SRL is evaluated in an ablation study aiming to quantify the data efficiency improvement added by the network architecture's different components. Then, using the proposed architecture, D-COACH is compared with HG-DAgger using simulated tasks and simulated teachers (oracles). The third set of experiments is performed with human teachers in simulated environments, again comparing different learning methods. Finally, a fourth set of validation experiments is conducted in real systems with human teachers. Most of the results are presented in this article; however, some are in the supplementary material, which can be found in IEEE *Xplore* along with more detailed information about the experiments.

Two real and three simulated environments with different complexity levels are used, all employing raw images as observations. The simulated environments are mountain–car, swing-up pendulum, and car racing, the implementations of which are taken from OpenAI Gym [17]. These simulations provide rendered image frames as observations of the environment. The frames visually describe the position of the system but not its velocity, which is necessary to control the system. The experiments on the real physical systems consist of a swing-up pendulum and a setup for picking oranges on a conveyor belt with a three degrees of freedom (3 DoF) robot arm. The metrics used for the comparisons are the achieved final policy performance and the speed of convergence, which is very relevant when dealing with real systems and human teachers. A video showing most of these experiments can be found at https://youtu.be/4kWGfNdm21A.

Ablation Study

In this ablation study, the architecture of the network is the independent variable. Three independent comparisons were carried out using DAgger, HG-DAgger, and D-COACH. The training sessions were run using a simulated teacher to avoid any influence from human factors. Three different architectures were tested for learning the policy from an oracle. The structure of the networks is introduced in the following:

1) *Full network*: the proposed architecture
2) *Memoryless SRL (M-Less SRL)*: similar to the full network but without using recurrence between the encoder and decoder (the autoencoder is trained using the reconstruction error of the observation)
3) *Direct policy learning (DPL)*: the same architecture as in the full network but without using SRL, i.e., not training the transition model (the encoding, recurrent layers, and policy are trained using only the cost of the policy).

The ablation study is done on a modified version of the car racing environment. Normally, this environment provides an upper view of a car on a race track. In this case, we occluded the bottom half of the observation such that the agent was not able to precisely know its position on the track. This position can be estimated if past observations are taken into account. As a consequence, this is an appropriate setting for making a comparison of different NN architectures. Table 1 gives the various performances obtained by the learning algorithms when modifying the structure of the network. The results show a normalized averaged return through 10 repetitions for each experiment, in which five evaluations were carried out for each of the repetitions.

As expected, DAgger with the full architecture obtained the best performance, and, given that it received new samples every time step, it was robust against changes in the architecture, even when it did not have memory. On the other hand, D-COACH was very sensitive to changes in the architecture, especially the DPL architecture. This shows how the full model is able to enhance the performance of the agents in problems where temporal information is required. It even makes D-COACH perform almost as well as DAgger, despite the fact that the former does not require constant and perfect teacher feedback. Finally, HG-DAgger was more robust than D-COACH in the DPL case, but its performance with the full model was not as good.

Table 1. A comparison of the performance (return) of different learning methods in the car racing problem.

	Full	M-Less SRL	DPL
D-COACH	0.97	0.76	0.68
DAgger	1	0.87	0.96
HG-DAgger	0.89	0.69	0.9

The returns were normalized with respect to the best performance (DAgger full).

Simulated Tasks With Simulated Teachers

In the second set of experiments, we performed a comparison among the DAgger, HG-DAgger, and D-COACH algorithms using the proposed full network architecture. To keep the experiments free of human-factor effects, the teaching process was, once again, performed with simulated teachers. The methods were tested in the mountain–car (in the supplementary material) and swing-up pendulum simulated problems. A mean of the return obtained through 20 repetitions is presented for these experiments, along with the maximum and minimum values of these distributions.

Swing-Up Pendulum

In the case of the swing-up pendulum, the results are very different for both DAgger agents (see Figure 4), which have a higher rate of improvement than D-COACH during the first minutes, when the policy is learning the swinging behavior. Since the swinging part requires large actions, the improvement with D-COACH is slower. However, once the policy is able to swing the pendulum up, the second part of the task is to keep the balance in the upright position, which requires fine actions. It is at this point that learning becomes easier for the D-COACH agent, which obtains a constant and faster improvement than the HG-DAgger agent, even reaching a higher performance. In Figure 4, the expected performance upper bound is indicated by a black dashed line, which is the return obtained by the simulated teacher. The purple dashed line shows the performance of a random policy, which is the expected lower bound.

Simulated Tasks With Human Teachers

The previous experiments give insights into how the policy architectures and/or the learning methods perform when imitating an oracle. Most IL methods are intended for learning

from any source of expert demonstrations. The source does not necessarily have to be a human; it can be any type of agent. However, the focus of this article is on learning from non-expert human teachers, who are complex to model and simulate. Therefore, conclusions have to be based on results that also include validation with real users.

Experiments with the mountain–car (in the supplementary material) and swing-up pendulum were run with eight human teachers. In this case, the classical DAgger approach was not employed since, as discussed in the "Interactive Learning Methods" section, it is not specifically designed for human users. Instead, HG-DAgger was validated.

Swing-Up Pendulum

This task is relatively simple from a control theory point of view. Nevertheless, it is quite challenging for humans to teleoperate the pendulum due to its fast dynamics. Indeed, participants were not able to successfully teleoperate the agent; therefore, unlike the mountain–car task, we could consider the participants as non-experts in the undertaking.

Figure 5 displays the results of this experiment, which are similar to the ones presented in Figure 4. At the beginning, D-COACH has a slower improvement when learning to swing up; however, it learns faster than HG-DAgger when the policy needs to learn the accurate task of balancing the pendulum. For users, it is more intuitive and easier to improve the balancing with the relative corrections of D-COACH than with the perfect corrective demonstrations of HG-DAgger, as users do not need to know the right action but, rather, just the direction of the correction. Unlike the performance of the simulated teacher depicted in Figure 4, the plot in Figure 5 shows the performance of the best human teacher teleoperating the pendulum with the same interface used for the teaching process. It can be seen that using both agents facilitated obtaining policies that outperformed the non-expert human teachers. All policies trained with D-COACH were able to balance the pendulum, whereas, with HG-DAgger, the success rate was half as high. Additionally, after the experiment, the participants were queried about which learning strategy they preferred. Seven out of eight expressed a preference for D-COACH.

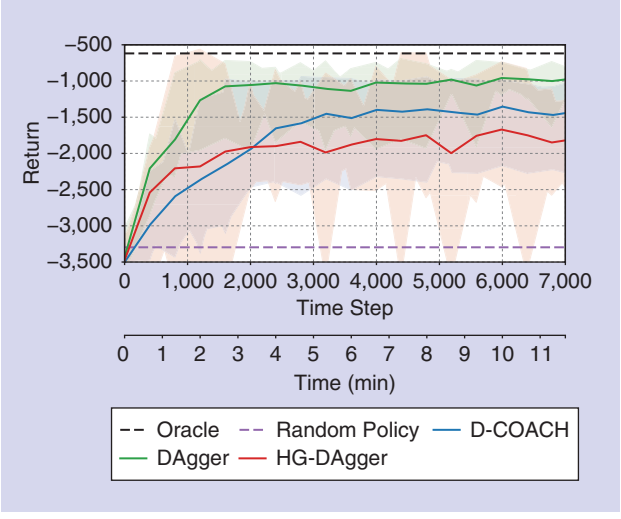


Figure 4. The D-COACH and HG-DAgger comparison in the swing-up pendulum problem using a simulated teacher.

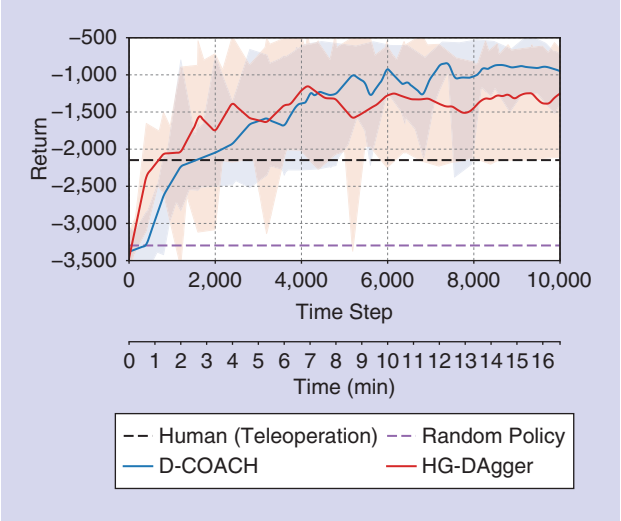


Figure 5. The simulated swing-up pendulum learning curve with human teachers.

Validation on Physical Systems With Human Teachers

The previous experiments performed comparison studies of the NN architectures and learning methods under controlled conditions in simulated environments. In this section, D-COACH is validated with human teachers and

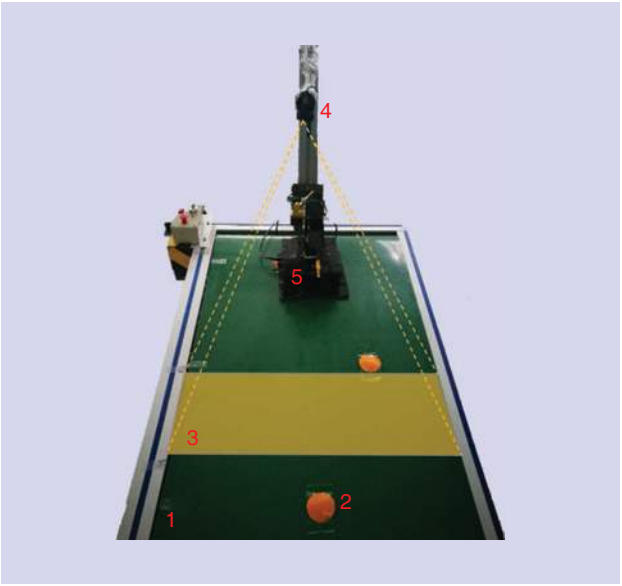


Figure 6. The orange-selector experimental setup. The 1: conveyor belt; 2: orange samples; 3: frame observed by the camera; 4: RGB camera; and 5: 3-DoF robot arm.

real systems through two different tasks: 1) a real swing-up pendulum and 2) a fruit-classifier robot arm. The real swing-up pendulum is a very complex system for a human to teleoperate. Its dynamics are faster than that of the OpenAI Gym simulated one used in the previous experiments. The supplementary material provides more details of this environment along with the learning curve of the agents trained by the participants of this validation experiment. Those results as well as the video show that non-expert teachers can manage to teach good policies.

Orange Selector With a Robot Arm

This setup consists of a conveyor belt transporting "pears" and "oranges," a 3-DoF robot arm located over the belt, and a red–green–blue (RGB) camera with a view of the belt from above (Figure 6). The image of the camera does not capture the robot arm. The robot has to select oranges with the end effector and avoid pears. The robot does not have any tool, such as a gripper or vacuum gripper, to pick up the oranges. Therefore, in this context, we consider a successful selection of an orange when the end effector intersects the object. The performance of the learning policy is measured using two indices: 1) the orange selection success rate and 2) the pear rejection success rate.

The observations obtained by the camera are from a different region of the conveyor belt than where the robot is acting. Therefore, observations cannot be used to compensate for the robot position in the current time step; rather, they are meaningful for future decisions. In other words, the current action must be based on past observations. Indeed, the delay between the observations and their influence on the actions is roughly 1.5 s. This delay is given by the difference between the time when the object leaves the camera range and the time when it reaches the robot's operating range, which is why this task requires learning temporal features for the policy.

The problem is solved by splitting it into two subtasks that are trained separately:

1) *Orange selection*: The robot must intercept the orange coordinate with the end effector exactly when the orange coordinate passes beneath the robot.
2) *Pear rejection*: The robot must classify between oranges and pears, so, when a pear is approaching, the end effector should lift far from the belt plane; otherwise, it should get close.

These two subtasks can be trained sequentially. The orange selection is initially trained through a procedure in which there are some oranges being transported at a fixed position on the belt while some others are placed randomly. This is to avoid overfitting the policy to specific sequences. When the robot is able to track the oranges within its reach, the pear rejection learning starts. For that, pears are placed randomly throughout the sequences of oranges, and the human teacher provides corrections to the robot movement to make the end effector move away from the pears when they are in the operation region of the robot.

Figure 7 depicts the average learning curves for this task after five runs of the teaching process. It is possible to see that the pear rejection subtask is learned within 20 min with 100% success, while the orange selection is a harder subtask that only reaches roughly 80% success after 50 min. Effectively, combining the two subtasks, the performance of the learned policies is given only by the success of the orange selection since the pear rejection was perfectly attained in all runs executed for this experiment.

Conclusions

This article introduced and validated an SRL strategy for interactively learning policies from human teachers in environments that are not fully temporally observable. Results show that, when meaningful spatiotemporal features are extracted, it is possible to teach complex end-to-end policies to agents using just occasional relative and binary corrective signals. Moreover, these policies can be learned from teachers who are not skilled at executing the task.

The evaluations with the DAgger approaches and D-COACH depict the potential of this kind of architecture to work with different IIL methods, especially those based on occasional feedback, which are intended to reduce the human workload. The comparative results between HG-DAgger and D-COACH with non-expert teachers showed that, with the former, the policy remains biased, with mistaken samples even if the teacher makes sure not to provide more wrong corrections (given that HG-DAgger works with the assumption of expert demonstrations), thus making the policy harder to refine. On the other hand, D-COACH proved to be more robust against mistaken corrections since all non-expert users were able to teach tasks they were not able to demonstrate.

The previously discussed shortcoming of DAgger algorithms opens possibilities for future works intended to study how to deal with databases that have mistaken examples. Another field of study is data-efficient movement generation in animation [19], which, combined with our method, would make it possible to learn (non)periodic

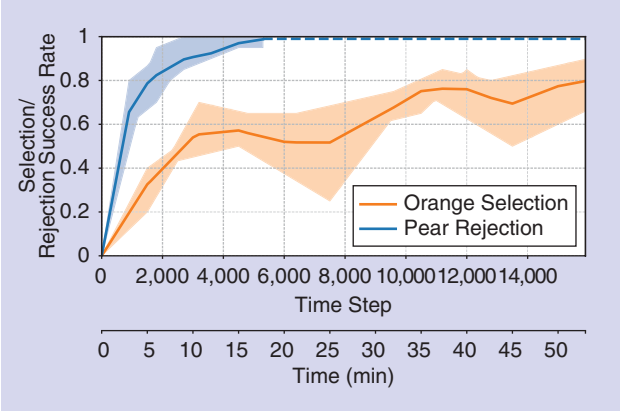


Figure 7. The orange selection/pear rejection learning curve.

movements using spatiotemporal features and IIL. Challenges such as the generation of smooth, precise, and stylistic movements (i.e., dealing with high-frequency details [20]) could be also addressed.

Acknowledgments

This research is funded by the Netherlands Organization for Scientific Research project Cognitive Robots for Flexible Agro-Food Technology, grant P17-01; European Research Council Starting Grant Teaching Robots Interactively, project reference 804907; Chile's National Fund for Scientific and Technological Development project (FONDECYT) 1201170; and Chile's Associative Research Program of the National Research and Development Agency (ANID/PIA) project AFB180004.

References

[1] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *J. Artificial Intell. Res.*, vol. 34, pp. 1–25, Jan. 2009. doi: 10.1613/jair.2584.

[2] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. Kochenderfer, "HG-DAgger: Interactive imitation learning with human experts," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 8077–8083. doi: 10.1109/ICRA.2019.8793698.

[3] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Proc. Advances Neural Information Processing Systems*, 2017, pp. 4299–4307.

[4] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The TAMER framework," in *Proc. 5th ACM Int. Conf. Knowledge Capture*, 2009, pp. 9–16. doi: 10.1145/1597735.1597738.

[5] C. Celemin and J. Ruiz-del Solar, "An interactive framework for learning continuous actions policies based on corrective feedback," *J. Intell. Robotic Syst.*, vol. 95, pp. 77–97, July 2019. doi: 10.1007/s10846-018-0839-z.

[6] J. MacGlashan et al., "Interactive learning from policy-dependent human feedback," in *Proc. 34th Int. Conf. Machine Learning*, 2017, vol. 70, pp. 2285–2294.

[7] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. doi: 10.1038/nature14236.

[8] 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.

[9] W. Böhmer, J. T. Springenberg, J. Boedecker, M. Riedmiller, and K. Obermayer, "Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations," *KI-Künstliche Intelligenz*, vol. 29, no. 4, pp. 353–362, 2015. doi: 10.1007/s13218-015-0356-1.

[10] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat, "State representation learning for control: An overview," *Neural Netw.*, vol. 108, pp. 379–392, Dec. 2018. doi: 10.1016/j.neunet.2018.07.006.

[11] R. Pérez-Dattari, C. Celemin, J. Ruiz-del Solar, and J. Kober, "Continuous control for high-dimensional state spaces: An interactive learning approach," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 7611–7617. doi: 10.1109/ICRA.2019.8793675.

[12] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14th Int. Conf. Artificial Intelligence and Statistics*, 2011, pp. 627–635.

[13] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *Proc. AAAI Fall Symp. Sequential Decision Making for Intelligent Agents (AAAI-SDMIA15)* Arlington, VA, Nov. 2015.

[14] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," in *Proc. 31st AAAI Conf. Artificial Intelligence*, 2017, pp. 2140–2146.

[15] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Proc. Advances Neural Information Processing Systems*, 2018, pp. 2450–2462.

[16] A. Zhang, H. Satija, and J. Pineau, Decoupling dynamics and reward for transfer learning. 2018. [Online]. Available: arXiv:1804.10689

[17] G. Brockman et al., OpenAI gym. 2016. [Online]. Available: arXiv:1606.01540

[18] I. Mason, S. Starke, H. Zhang, H. Bilen, and T. Komura, "Few-shot learning of homogeneous human locomotion styles," *Comput. Graph. Forum*, vol. 37, no. 7, pp. 143–153, 2018. doi: 10.1111/cgf.13555.

[19] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Trans. Graph. (TOG)*, vol. 37, no. 4, pp. 1–11, 2018. doi: 10.1145/3197517.3201366.

Rodrigo Pérez-Dattari, Department of Cognitive Robotics, Delft University of Technology, The Netherlands. Email: r.j.perezdattari@tudelft.nl.

Carlos Celemin, Department of Cognitive Robotics, Delft University of Technology, The Netherlands. Email: c.e.celeminpaez@tudelft.nl.

Giovanni Franzese, Department of Cognitive Robotics, Delft University of Technology, The Netherlands. Email: g.franzese@tudelft.nl.

Javier Ruiz-del-Solar, Department of Electrical Engineering and the Advanced Mining Technology Center, Universidad de Chile, Santiago. Email: jruizd@ing.uchile.cl.

Jens Kober, Department of Cognitive Robotics, Delft University of Technology, The Netherlands. Email: j.kober@tudelft.nl.

By Qingkai Lu, Mark Van der Merwe,
Balakumar Sundaralingam, and Tucker Hermans

We propose a novel approach to multifingered grasp planning that leverages learned deep neural network (DNN) models. We trained a voxel-based 3D convolutional neural network (CNN) to predict grasp-success probability as a function of both visual information of an object and grasp configuration. From this, we formulated grasp planning as inferring the grasp configuration that maximizes the probability of grasp success. In addition, we learned a prior over grasp configurations as a mixture-density network (MDN) conditioned on our voxel-based object representation. We show that this object-conditional prior improves grasp inference when used with the learned grasp success-prediction network compared to a learned, object-agnostic prior or an uninformed uniform prior. Our work is the first to directly plan high-quality multifingered grasps in configuration space using a DNN without the need of an external planner. We validated our inference method by

Multifingered Grasp Planning via Inference in Deep Neural Networks

Outperforming Sampling by Learning Differentiable Models

Digital Object Identifier 10.1109/MRA.2020.2976322
Date of current version: 27 March 2020

performing multifinger grasping on a physical robot. Our experimental results show that our planning method outperforms existing grasp-planning methods for neural networks (NNs).

Overview and Motivation

Learning-based approaches to grasping [1]–[8] have become a popular alternative to geometric [9]–[12] and model-based planning [13], [14] over the past decade. This is largely because grasp learning generalizes well to previously unseen objects where only partial-view visual information is available. Moreover, by having a robot attempt and validate its own grasps in a self-supervised manner [4], we remove the need for humans to make guesses about what grasps will be successful [1] as well as the need for the performance of complex mappings from human grasps to robot grasps.

Recently, researchers have looked to capitalize on the success of DNNs to improve grasp learning. Broadly speaking, DNN methods for grasp learning can be split into two approaches: 1) predicting grasp success for the object visual information represented by an image patch or a pointcloud associated with a gripper configuration [3], [4], [6], [15]–[19] and 2) directly predicting a grasp configuration from the object visual information represented by an image or image patch using regression [8], [20]–[22]. While these deep-learning approaches have shown impressive performance for parallel jaw grippers (e.g., [4]), little work has focused on the more difficult problem of multifingered grasping [7], [8], [19], [22], [23]. We believe two primary difficulties restrict the use of deep learning for multifingered grasping:

1) the grasping input representation in NNs
2) the reliance on external planners for generating candidate grasps.

The first difficulty arises from

1) the great variability in object shapes a robot may encounter during deployment
2) the high dimensionality of multifingered grasp configurations.

As such, grasp learning requires representations that can efficiently, in terms of data and computation, encode both the geometry of the object and the grasp configuration necessary to predict grasp success. The second issue arises from the increase in search complexity for planning multifingered grasps compared to grasping with parallel jaw grippers. As such, we desire an efficient inference procedure that can accomplish tasks faster than typical model-based planners and that does not suffer from the limiting assumptions and bias present in such human-designed planners.

To combat these two problems, we propose an alternative approach to grasp planning with DNNs. With this approach, we directly use the learned network for planning. In our work, we trained a network to predict grasp success, given an object and grasp-configuration representation. However, unlike currently employed sampling methods, we performed a continuous optimization over the grasp configuration, guiding the updates with the network.

In addition to giving the NN the object visual information represented by a red, green, blue, and depth (RGB-D) image or a voxel grid as input, we provided the grasp-configuration parameters in the form of finger-preshape joint angles and palm pose. However, we formulated the learning and inference problem in a general form that would allow other grasp representations to easily be used instead (e.g., joint angles of in-contact fingers, desired finger contact locations on the object surface, and others).

Once trained, given the object visual representation, we performed inference over the grasp-configuration parameters to maximize the probability of grasp success learned by our CNN. We performed this probabilistic inference as a direct optimization over the grasp configuration, which leverages the efficient computation of gradients in NNs, while ensuring joint angles remain within their limits. Thus, our approach can quickly plan reliable multifingered grasps given an image of an object and an initial grasp configuration.

This article makes the following contributions over our previous work [24]:

- We propose a new grasp model for grasp learning and inference, including a voxel-based 3D CNN for grasp-success probability prediction that encodes the object 3D geometry well.
- We present a novel MDN to model a grasp-configuration prior conditioned on the observed object, which removes the need for an external grasp planner to initialize our grasp optimization.
- Our grasp planner generates both successful side and overhead grasps on a real robot, while our grasp planner, described in Lu et al. [24], only generates successful side grasps on the real robot.
- We train our grasp model using more than seven times the data of Lu et al. [24].
- We perform real-robot grasp experiments on more objects than in Lu et al. [24].
- In total, this creates a grasp planner that achieves a higher success rate than our previous planner, described in Lu et al. [24].

Our planner offers a number of benefits over previous deep-learning approaches to multifingered grasping. Kappler and colleagues [23] learned to predict whether a given palm pose will be successful for multifingered grasps using a fixed preshape and performed planning by evaluating a number of sampled grasp poses. Varley et al. [19] presented a deep-learning approach to effectively predict a grasp-quality metric for multifingered grasps, but they relied on an external grasp planner to provide candidate grasps. In contrast, our method learns to predict grasp success as a function of both the palm location and preshape configuration and plans grasps directly using the learned network. Saxena et al. [2] also performed grasp planning as inference using learned probabilistic models; however, they used separate classifiers for both the image and range data, and they used hand-selected models instead of a unified deep model. Zhou and Hauser [25] concurrently proposed a similar optimization-based grasp-planning

approach to ours using similar CNN architecture. In contrast to our work, they did not interpret planning as probabilistic inference; they optimized only for hand pose, ignoring hand-joint configurations, and they validated only in simulation.

Veres et al. [22] trained a conditional variational autoencoder deep network to predict the contact locations and normals for a multifingered grasp given an RGB-D image of an object. To perform grasping, an external inverse kinematics solver must be used for the hand to try to reach the desired contact poses as best as possible. Liu et al. [8] trained a 3D voxel CNN to directly predict the multifinger grasp configuration. Implicit in such regression methods, as proposed in Liu et al. [8] and in Veres et al. [22], is the assumption that there is a unique best grasp for a given object view. In contrast, our method can plan multiple successful grasps for a given object using different initial configurations with associated high confidence prior to execution. This offers the robot the option of selecting a grasp best suited for its current task. Additionally, we show that our classification-based network can effectively learn with a smaller data set compared to a regression network, which cannot leverage negative grasp examples.

We formulated multifingered grasp planning as probabilistic inference in a learned DNN without a prior over grasp configuration in our previous work [24]. Our multichannel DNN in Lu et al. [24] took a grasp configuration and an RGB-D image grasp patch as inputs and predicted as output the probability of grasp success. Our planning algorithm generally achieved higher grasp-success rates compared with sampling-based and regression approaches currently used for grasping with NNs.

We explored a probabilistic graphical model for grasp learning and planning over grasp type and configuration for a given object in our previous work [26]. We used a data-driven Gaussian mixture model (GMM) prior independent of the object to constrain the inference not to stray into areas far from grasp configurations observed at training time, where we have little evidence to support grasp-success predictions. The grasping experiment results in Lu and Hermans [26] demonstrated the benefit of a data-driven prior for grasp inference. In this article, we propose an object-conditional prior modeled as an MDN [27]. Our MDN prior models the grasp-configuration distribution based on the geometry of the object of interest. Our real-robot experiments show that the grasp inference with the MDN object-conditional prior outperforms grasp inference with the GMM object-independent prior. We trained a logistic regression classifier to predict the grasp-success probability on a small data set with 120 grasps in Lu and Hermans [26]. In this article, we trained a voxel-based 3D CNN to predict the grasp-success probability on a larger data set containing 10,811 grasp attempts.

Grasp Planning as Probabilistic Inference

Following Ciocarlie et al. [11], we defined the grasp-planning problem as finding a grasp preshape configuration. In our case, the grasp-configuration vector is composed of the palm pose in the object-reference frame and the hand's preshape joint angles, which define the shape of the hand prior to closing the hand. To make the grasp inference agnostic to object poses, we put the palm pose in the object-reference frame for learning and inference. After finding the grasp preshape configuration, the robot moved to this preshape and ran a controller to close the hand forming the grasp on the object. We explain the specific joints used for defining the preshape and how the grasp controller works for our experiments in the section "Grasp Data Collection." We focus on scenarios where a single, isolated object of interest is present in the scene. Importantly, we assume no explicit knowledge of the object beyond a single camera sensor reading of it in its current pose. The problem we address is stated as follows: given such a grasp scenario, plan a grasp preshape configuration that allows the robot to successfully grasp and lift the object without dropping it.

Given the learned model parameters, $\boldsymbol{W}$ and $\boldsymbol{\Phi}$, along with the visual representation, z, associated with an observed object of interest, our goal is to infer the grasp-configuration parameters, $\boldsymbol{\theta}$, that maximize the posterior probability of grasp success $Y = 1$. Here Y defines a random Boolean variable, with 0 meaning failure and 1 meaning success. We can thus formalize grasp planning as a maximum a posteriori (MAP) inference problem:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} - \log p(\boldsymbol{\theta} \mid Y = 1, z, \boldsymbol{W}, \boldsymbol{\Phi}) \tag{1}$$

$$\text{subject to } \boldsymbol{\theta}_{\min} \preceq \boldsymbol{\theta} \preceq \boldsymbol{\theta}_{\max}. \tag{2}$$

We constrain the grasp-configuration parameters to obey the joint limits of the robot hand in (2).

We define the grasp-success likelihood $p(Y = 1 \mid \boldsymbol{\theta}, z, \mathbf{W})$ to be a DNN. $\mathbf{W}$ represents the NN parameters. The DNN predicts the probability of grasp success, Y, as a function of the visual representation of the object of interest, z, and hand configuration, θ. (We previously represented the object of interest as an RGB-D image patch in Lu et al. [24]. In this article, we use a voxel grid to represent the object of interest.) We describe the details of our NN classifier in the section "Voxel-Based Grasp-Likelihood Classification."

We present three different ways to model the prior over the grasp configuration θ. In each case, $\boldsymbol{\Phi}$ represents the associated parameters of the prior distribution. In the first approach, we assume a uniform prior over valid grasp configurations, resulting in the grasp-success posterior probability being proportional to the likelihood:

$$p(\boldsymbol{\theta} \mid Y = 1, z, \boldsymbol{W}, \boldsymbol{\Phi}) \propto p(Y = 1 \mid \boldsymbol{\theta}, z, \boldsymbol{W}). \tag{3}$$

This prior requires all grasp parameters to be bounded to prevent the inference straying from the training evidence. This approach was used in our initial work [24]. It is trivial to bound the preshape joint angles using the robot hand joint limits. However, it requires heuristic bounds to be manually designed for the hand palm pose in Cartesian space.

The second prior we examine defines a prior over grasp configurations to encode preferred grasp configurations

independent of the observed object, which gives the following posterior:

$$p(\boldsymbol{\theta}|\, Y = 1, z, \boldsymbol{W}, \boldsymbol{\Phi}) \propto p(Y = 1|\, \boldsymbol{\theta}, z, \boldsymbol{W}) p(\boldsymbol{\theta}|\, \boldsymbol{\Phi}). \quad (4)$$

In practice. one could choose from many different functions to implement this prior; however, in this article, we focus on using a GMM, building on our previous use of GMM grasp priors in Lu and Hermans [26].

As a third prior, we propose an object-conditional prior:

$$p(\boldsymbol{\theta}|\, Y = 1, z, \boldsymbol{W}, \boldsymbol{\Phi}) \propto p(Y = 1|\, \boldsymbol{\theta}, z, \boldsymbol{W}) p(\boldsymbol{\theta}|\, z, \boldsymbol{\Phi}). \quad (5)$$

Here, the preferred grasp configurations are conditioned on the observed sensory information. For both the GMM of (4) and the object-conditional prior of (5), we examine data-driven priors to encode knowledge of what data were observed by the learner during training. This prior thus prefers grasps similar to those seen during training. Such priors can be viewed as an approximation of the epistemic uncertainty of the learned classifier [28], encoding the belief that the classifier's confidence should decrease for grasp or objects far from those observed during training. We define the details of the object-conditional prior as a DNN in the section "Voxel-Based Grasp Prior Networks." In the section "Robotic Grasp Inference Experiments," we describe experiments we performed comparing grasp inference using variants of these three priors.

We solved the inference problem for all three grasp models in the log-probability space and regularized the log-prior with a multiplicative gain of 0.5 to prevent the prior from dominating the inference. We used the popular limited memory Broyden–Fletcher–Goldfarb–Shanno optimization algorithm with bound constraints [29], [30] to efficiently solve the inference problem. We used the scikit-learn library (http://scikit-learn.org/stable/index.html) to perform the optimization. We initialized the inference by randomly sampling from the learned priors. We initialized the uniform prior using a heuristic described in the section "Grasp Data Collection" as previously done [24].

Voxel-Based Deep Networks for Multifingered Grasp Learning

Voxel-Based Grasp-Likelihood Classification

Our voxel-based neural network classifier predicts grasp success for multifingered hands. Figure 1(a) shows the architecture of our grasp-success prediction network. Our voxel-based classifier takes three inputs: a $32 \times 32 \times 32$ object voxel grid; a vector defining the width, height, and depth of the voxel grid in the 3D scene; and a 14D vector encoding the grasp preshape configuration, which we define in more detail in the section "Grasp Data Collection." This preshape configuration could be replaced with other grasp representations in a straightforward manner. The network processes the object voxel grid with a subnetwork composed of four 3D convolutional layers and one fully connected layer, which we named the *voxel encoder*. We pretrain this voxel encoder on a 3D object-reconstruction task described in the section "Grasp Model Training."

We concatenate the voxel features processed by the voxel encoder with the object-size vector, and we pass the concatenated features through two fully connected layers to generate the final object-feature representation. The grasp-configuration input is processed by two fully connected layers to generate the grasp-configuration features. Then we concatenate the grasp-configuration features and the final object features and pass them through two fully connected layers followed by a sigmoid output layer to generate the grasp-success probability. We apply batch normalization for all convolutional and fully connected layers except the output layer. We train our voxel-based classifier using the cross-entropy loss.

To generate the voxel grid, we first segment the object from the 3D pointcloud by fitting a plane to the table using random sample consensus [31], [32] and extracting the points above the table. We then estimate the first and second principle axes of the segmented object to create a right-handed object-reference frame aligned relative to the world frame. We compute the object size along the three coordinates of the object-reference frame to construct the object-size vector. We then generate a $32 \times 32 \times 32$ voxel grid oriented about this reference frame. We define the center of the voxel grid to be the centroid of the points in the object segmentation. More details of the pointcloud voxelization can be seen in the section "Grasp Model Training."

Voxel-Based Grasp Prior Networks

To model the grasp-configuration distribution based on the geometry of the object of interest, we construct an MDN as our object-conditional prior. Given its input, an MDN predicts the parameters (means, covariance, and mixing weights) of a GMM as output. Our MDN takes the object voxel grid and object-size vector as inputs and predicts the parameters of a GMM modeling a probability distribution over grasp configurations. Thus, the MDN learns to model the conditional probability distribution $p(\boldsymbol{\theta}|\, z, \boldsymbol{\Phi})$, where $\boldsymbol{\Phi}$ defined the learned weights of the MDN. We train the MDN over all grasp attempts from the training set, i.e., the learned distribution models the probability that the specific grasp $\boldsymbol{\theta}$ being evaluated was observed at training, given the current object.

The MDN generates its object-feature representation using the same subnetwork structure as the voxel-based classifier. The MDN then passes the final object-feature representation through two fully connected layers with rectified linear unit activations. These two fully connected layers have 128 and 32 neurons respectively. Finally, the fully connected output layer predicts the weights, mean, and diagonal covariance of the mixture distribution over grasp configuration. Figure 1(b) shows the architecture of our MDN. We apply batch normalization for all layers of the MDN except the output layer. We train our voxel-based MDN using the negative log likelihood loss. Figure 2 shows the mean grasp configuration of each

mixture component predicted by the MDN for several different objects.

Grasp Data Collection

We conducted all training and experiments using the four-fingered, 16-degrees-of-freedom (DoF) Allegro hand-mounted on a Kuka LBR4 7-DoF arm. We used a Kinect2 camera to generate the pointcloud of the object on the table. We collected simulated grasp data using our robot hand–arm setup inside the Gazebo simulator with the Dynamic Animation and Robotics Toolkit physics engine (https://dartsim.github.io/). We used the built-in Gazebo Kinect camera to generate pointclouds simulating the Kinect2 RGB-D camera we used in real-world experiments. All data and software used in this article are available online (https://robot-learning.cs.utah.edu/project/grasp_voxel_inference).

We collected training data using a heuristic, geometry-based grasp planner adapted from our previous planner in Lu et al. [24], which is quite similar to the geometric primitive planner of Miller et al. [33] for boxes or cylinders. We collected both side and overhead multifingered grasps. For side grasps, we had the thumb pointing toward the top as in Lu et al.

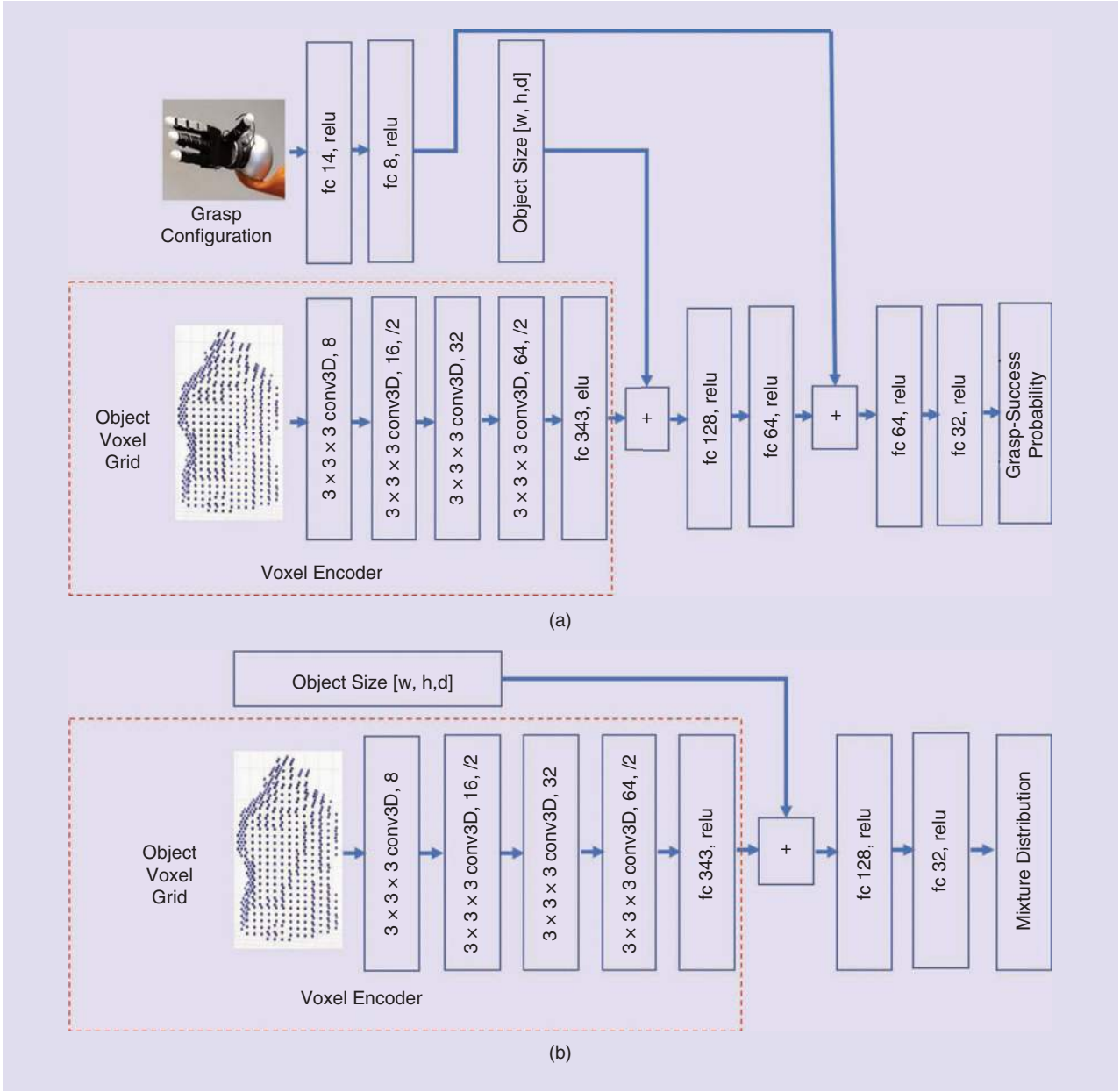


Figure 1. The voxel-config-net and MDN architectures. (a) The architecture of our voxel-config-net for grasp success probability prediction. (b) The architecture of our MDN modeling the grasp conditional prior. The image labeled "Object Voxel Grid" shows the voxel grid for the mustard object. All convolutional layers use $3 \times 3 \times 3$ 3D convolutional filters with exponential linear unit (elu) activations. We annotated the number of filters and the stride (/2 means a stride of 2) for the convolutional layers. We annotated the number of neurons and the activation function for fully connected layers. relu: rectified linear activation unit; fc: fully connected; conv3D: convolutional 3D.

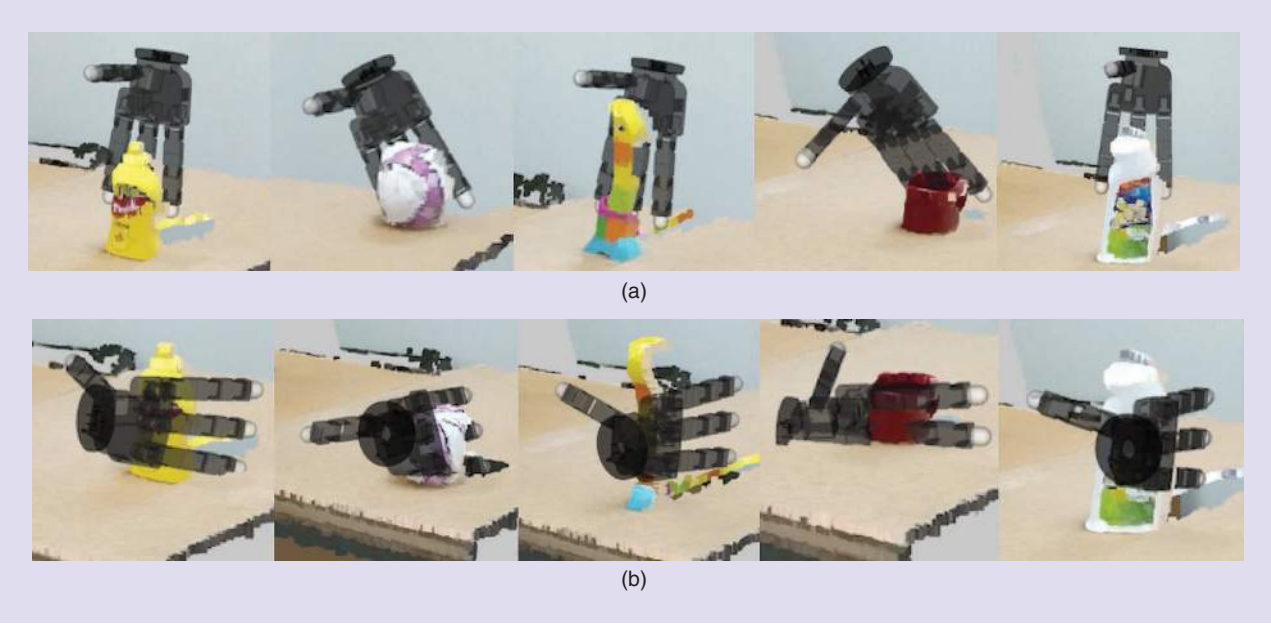


Figure 2. The mean visualizations of two MDN mixture components for five different objects. (a) Mean of the overhead component. (b) Mean of the side component.

Table 1. Number of successful and failed side and overhead grasps in the simulation-based training and offline-testing sets generated using the heuristic planner.

	Training		Offline Testing	
Grasp Type	**Success**	**Failure**	**Success**	**Failure**
Side	1,559	2,616	339	515
Overhead	749	4,064	157	810

[24]. In that work, overhead grasps were randomly selected to align with either the major or minor axis of the bounding box top face, but this fails to generate overhead grasps robustly. In this article, we improve the overhead heuristic planner by aligning the thumb to point in the direction of the mean vector related to the major and minor axes of the bounding box top face, with the palm parallel to the top face. This modification boosts the overhead grasp success rates in data collection.

We generated a preshape by randomly sampling joint angles for the first two joints of all fingers within a reasonable range, fixing the last two joints of each finger to be zero. There are 14 parameters for the Allegro hand preshape, six for the palm pose, and eight relating to the first two joint angles of each finger proximal to the palm. Given a desired pose and preshape, we use the rapidly exploring random tree-connect motion planner in MoveIt! to plan a path for the arm. We executed all feasible plans moving the robot to the sampled preshape.

After moving the hand to the desired preshape, the robot ran a grasp controller to close the hand. The grasp controller closed the fingers at a constant velocity, stopping each finger independently when contact was detected by the measured joint velocities being close to zero. The grasp controller closed the second and third joints of the nonthumb fingers and the two distal joints of the thumb. Note that the proximal joint of all nonthumb fingers rotated the finger about its major axis, causing it to change the direction of closing. As such, we maintained the angle provided by the grasp planner for these joints.

Upon closing, the robot attempted to lift the object to a height of 15 cm. If the robot succeeds in reaching this height without the object falling, the simulator automatically labels the grasp as successful. We collected 10,809 grasp attempts in total, of which 2,804 resulted in successful grasps. The data set covers more than 100 objects of the Bigbird [34] data set, which contains more than seven times the number of grasp attempts as in the data set from our previous work [24], including 1,898 successful side grasps and 906 successful overhead grasps. We used 8,988 grasps for training of our grasp model and 1,811 grasps for testing. Table 1 shows the number of successful and failed grasp attempts of side and overhead grasps separately in our training and offline testing sets.

Grasp Model Training

We trained our networks using the Adam optimizer with minibatches of size 64 for 90 epochs. The learning rate started at 0.001 and decreased by 10 × every 30 epochs. The MDN was trained with the same specifications. The training of both models requires fewer than 25 min on a computer with an Intel-i74790k processor, 64-GB RAM, and an Nvidia GeForce GTX 970 graphics card. We implemented all our deep network models in TensorFlow. We fit the GMM prior parameters using the expectation-maximization algorithm over all grasp configurations attempted in the training set.

We pretrained the voxel encoder for our classifier and MDN on a voxel-based 3D object-reconstruction task. We froze the voxel encoder parameters to the values learned on

this reconstruction task during grasp training of the classifier and MDN. We found that freezing the encoder parameters achieves better testing performance than fine-tuning the encoder parameters for both models.

Our voxel reconstruction autoencoder has the same output and decoder structure as the reconstruction variational autoencoder in Brock et al. [35], except we dropped the variational constraint, i.e., we do not predict a variance associated with the output of the encoder. We found this achieves better reconstruction results than the variational autoencoder in Brock et al. [35] on our data set.

To train our voxel reconstruction autoencoder, we synthetically rendered 590 meshes from the Grasp Database [36] at 200 random orientations each, adding noise to the depth images to reflect sensor noise. We backprojected these points into a 3D pointcloud, which we voxelized to a $26 \times 26 \times 26$ voxel grid, centered in a $32 \times 32 \times 32$ total voxel grid. We scaled the bounding box extracted from the segmented object pointcloud to have $26 \times 26 \times 26$ voxels. We were here concerned only with capturing object-shape information, as object size and pose were handled by the grasp network independently. As such, we learned the mapping from the partial pointcloud voxelization to a centered and independently scaled full-mesh voxelization. We trained with a sigmoid cross-entropy loss on the true and predicted reconstructed voxel representation. The voxel reconstruction autoencoder was trained using the momentum optimizer with minibatches of size 64 for 100 epochs. We used a starting learning rate of 0.001 and decreased the rate by $10\times$ after 20 and 80 epochs and set momentum to 0.9. Our network trained in about 4 h on a single Nvidia Tesla V100 graphics card. Our pretrained, voxel reconstruction network achieved 95.56% accuracy and an F1-score of 0.5166 on a test set comprising left out models from the Grasp Database [36] and all models from the Yale–Carnegie Mellon University–Berkley (YCB) data set [37].

Offline Grasp-Learning Validation

We validated the performance of our voxel-based grasp-success classifier on an offline prediction task using the held-out test set collected in simulation. We show the prediction accuracy and F1 score in Table 2. We compared our approach (voxel-config-net) with our previous RGB-D-based classifier from Lu et al. [24] (rgbd-config-net). We retrained the RGB-D network on the training data of this article with the same specifications as stated, except we used minibatches of size 8. The RGB-D network takes more than 24 h to finish 90 epochs of training.

Our voxel-based network significantly outperformed the classifier using the RGB-D object representation on this task in terms of F1 and accuracy. This result holds for both side and overhead grasps. We believe this implies the voxel-based approach encodes the object geometry better for grasping than using an RGB-D image directly.

We treated predictions with grasp probability above 0.5 as positive for the voxel-based approach. As done in Lu et al. [24], we thresholded the RGB-D network predictions with a value of 0.4, as setting the threshold to 0.5 predicts all grasps as failures. This necessary modification highlights another advantage of our novel voxel-based classifier: the predicted grasp probabilities better reflect the true success probabilities, making them more useful for planning as inference.

For completeness, we also show the MDN negative log likelihood loss on the testing set, where smaller MDN loss reflects higher conditional probability density.

Robotic Grasp Inference Experiments

We evaluated grasp planning as inference on the physical robot system using our learned voxel-based grasp classifier with all three grasp-configuration prior probability models described in the section "Grasp Planning as Probabilistic Inference" (i.e., uniform, GMM, and object-conditional). We performed experiments on eight YCB [37] objects covering different textures, shapes, and sizes. We show the experimental setup and objects used in Figure 3. All experimental objects are unseen in training except for the Pringles object. We attempted both overhead and side grasps at five different poses per object, for a total of 80 grasp attempts per method. We used the same set of locations across different methods, but each object had its own set of random poses. In total, we

Table 2. Grasp model evaluation on testing set for all, side, and overhead grasps.

Grasp Type	voxel-config-net		RGB-D-config-net		MDN Loss
	Accuracy	F1	Accuracy	F1	
Both	0.786	0.592	0.573	0.476	−10.73
Side	0.724	0.649	0.661	0.401	−14.71
Overhead	0.842	0.456	0.474	0.52	−7.2

We show the accuracy and F1 score of voxel-config-net and RGB-D-config-net. We also show the negative log likelihood loss of MDN.

Figure 3. The experimental setup with the objects used. Objects are (from left) a Pringles container, Lego construction, soccer ball, mug, mustard bottle, sugar box, soft scrub bottle, and pitcher. Objects range in size from $8 \times 9 \times 11$ cm (mug) to $13 \times 17 \times 24$ cm (pitcher).

performed 240 grasp attempts for three different methods across eight objects in this article.

Our evaluation protocol on the physical robot mirrors that used in simulation. Namely, we labeled a grasp attempt that lifted the object to a height of 0.15 m without dropping it as successful. We used the same motion planner and grasp controller that we used in data collection to plan paths for the arm and close the hand. If the motion planner failed to generate a plan for a grasp due to either inverse kinematics or collision avoidance, we generated a new grasp using the same grasp planner with a different initialization. If the grasp planner could not generate a grasp with a motion plan in five attempts, we treated the grasp attempt as a failure.

The MDN object-conditional and GMM priors both have two mixture components. For both learned prior models, we found the grasp-configuration mean of one mixture component is a side grasp (we term this component the *side-grasp component*) and the grasp-configuration mean of the other component is an overhead grasp (we term this component the *overhead-grasp component*). We randomly sampled a grasp configuration from the side-grasp component to initialize the grasp inference to generate a side grasp. Similarly, we initialized the grasp inference with a random sample from the overhead-grasp component to plan an overhead grasp. We initialized the grasp inference of the uniform prior with side and overhead grasps generated by the heuristic geometry-based planner used for data collection to plan side and overhead grasps respectively.

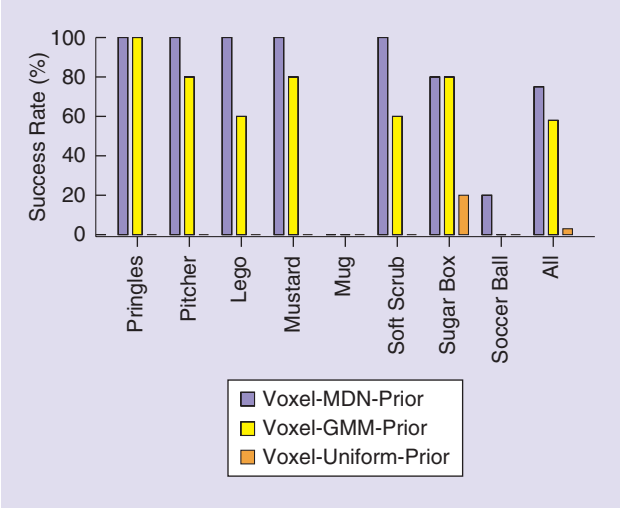


Figure 4. The multifingered side-grasping success rates of three different prior distribution methods on the real robot. The Pringles object was seen in training. The other seven objects were previously unseen.

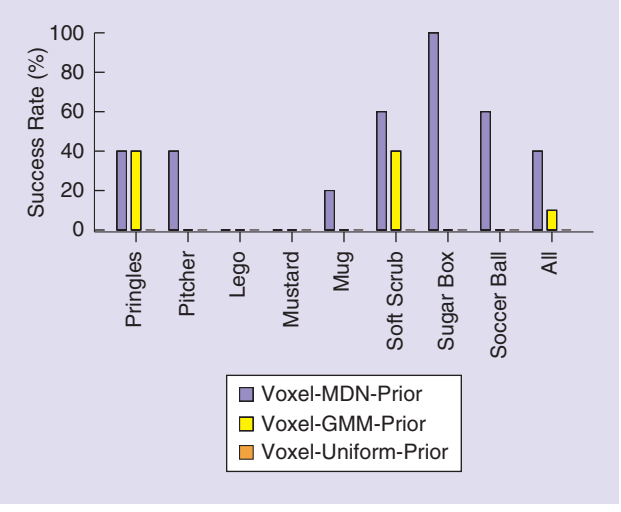


Figure 5. The multifingered overhead-grasping success rates of three different methods on the real robot. The Pringles object was seen in training. The other seven objects were previously unseen.

The side-grasp success rates for all three methods are summarized in Figure 4. Figure 5 shows the overhead-grasp success rates for all three methods. It takes 5–10 s for each method to generate a grasp. The grasp planners using the MDN, the GMM, and the uniform priors achieved success rates of 75, 58, and 3%, respectively, on side grasps for the eight objects. The grasp planner was not able to plan any successful side grasps for the mug object. Since the mug object is relatively short, the motion planner could not find paths that would not collide with the table during side grasps for all three methods. The grasp planners using the MDN, GMM, and uniform priors achieved success rates of 40, 10, and 0%, respectively, on overhead grasps of the eight objects. The grasp planner never generated overhead grasps successfully for the mustard and Lego objects. The mustard and Lego objects have relatively smaller contact areas available for overhead grasps, and our grasp controller would push them away when closing the hand as it had no feedback from vision or haptic sensors to know that the object was moving.

The grasp planner using the MDN prior achieved higher success rates than the two other methods for both side and overhead grasps; this demonstrates the benefit of modeling the grasp prior conditionally on the observed object when performing inference. The uniform prior achieved the lowest success rates for both side and overhead grasps. This shows that data-driven priors, even if not conditioned on the observed object, outperform the use of a heuristic planner for initialization and locally constraining the grasp configuration, as these heuristics cannot reliably generate successful grasps.

In Figure 6, we show example grasps for different objects generated by our inference approach with the voxel-based classifier and MDN prior. Grasps in the top two rows [(a)–(n)] are side grasps, which provide strong stability. The bottom two rows [(o)–(bb)] show overhead grasps, which provide access to objects in clutter and often demonstrate improved dexterity between the robot and object.

Discussion and Conclusions

In this article, we presented a novel approach for multifingered grasp planning formulated as probabilistic inference in learned DNNs. We proposed a voxel-based 3D CNN to predict the probability of grasp success as a function of both the object voxel grid and grasp configuration. We showed that our novel, voxel-based grasp-success classifier for

multifingered grasping outperforms our previous RGB-D image patch-based NN presented in Lu et al. [24] in terms of both predictive accuracy and training time.

In our previous work, we showed that planning as inference with the RGB-D structure outperforms several alternatives for grasp planning, such as sampling and regression. As our results here indicate improvement over the RGB-D network structure, we can infer that our planning similarly outperforms alternative multifingered grasp-planning approaches, while still running fast enough for use in a deployed robotic system. Additionally, our CNN classification approach to learning grasp success allows for more data-efficient learning compared to directly predicting grasps using regression. In the regression-based formulation, the NN takes the visual information (e.g., RGB-D or voxel grid) as input and directly predicts a grasp configuration as output. These regression models can learn only from successful grasps, while our success classifier learns from both successful and failed grasp attempts.

Using our learned, voxel-based classifier, we examined the role of different prior probability distributions over grasp configuration in the planning process. Our real-robot grasp experiments for the three different prior models, defined in the section "Grasp Planning as Probabilistic Inference," show that using a learned object-conditional prior over grasp configurations benefits grasp inference when combined with the learned grasp-success prediction network. This learned MDN benefits from the same voxel encoder to represent the observed object of interest. Furthermore, the data-driven GMM also provides benefits over the bounded uniform prior. This provides further evidence that learned priors provide better planning performance compared with heuristic, weak priors.

Learned priors provide additional benefits for multifingered grasp planning. First, they enable the robot to directly sample initial configurations for use in the resulting optimization problem. In our previous work, we showed how randomly generating numerous samples from the uniform prior and selecting the one with highest predicted success fails to reliably generate successful grasps. This demonstrates the benefit of the learned prior, which focuses the search space to promising configurations, something the uniform prior cannot provide. To overcome this problem, we previously relied on an external grasp planner to initialize the optimization.

The second additional benefit of the learned prior comes from its elimination of the need for an external planner or heuristic for initialization. In addition to generally being computationally more efficient, the removal of this external planner reduces the bias present in human-designed planners, which limit the space of grasps under consideration. Model-based planners tend to prefer only a single class of grasps,

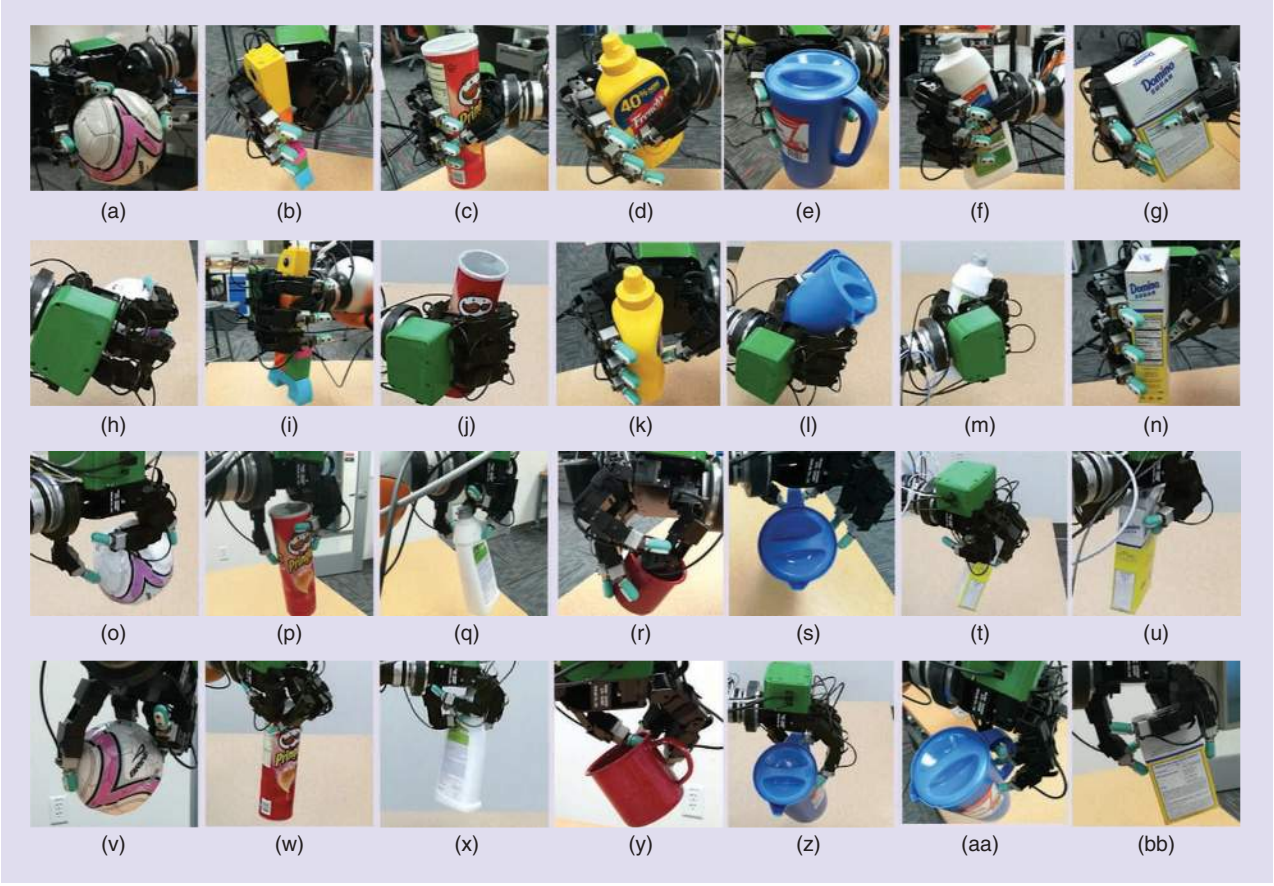


Figure 6. Examples of successful grasps generated by grasp inference using our voxel-based classifier with the MDN object-conditional prior. (a)–(n) Side grasps and (o)–(bb) overhead grasps.

such as power or precision but not both [26]. By leveraging data-driven priors, the robot is not restricted to grasps similar to those provided by the planner; instead, it can leverage any grasp its learned model predicts will be successful. Indeed, our resulting grasp planner reliably generates successful side and overhead grasps on the real robot across several different objects used for testing.

We can directly attribute this ability to generate both overhead and side grasps to the new data set we generated for this article, which contains substantially more successful overhead grasps than our previous data set (Lu et al. [24]). However, this improvement highlights that we have simply shifted the burden of the external planner from inference initialization to grasp exploration in generating training data. While we overcame the bias of the planner somewhat by adding random perturbations to the output of our heuristic planner, we still limited the space of grasps explored during training.

To overcome this issue in the future, we wish to explore active learning where the robot selects what grasps to attempt for learning based on the previous attempted grasps and the currently learned grasp model. This should improve the data efficiency of our learning algorithm while also learning a wider variety of grasps. However, new issues arise concerning how to correctly update the learned model in an online fashion, as the standard independent identically distributed data assumption used for batch NN training will no longer hold.

A more obvious shortcoming of our MDN prior stems from the need to explicitly select the number of mixture components in the model. An open question remains as to how we can expand the capacity of the mixture network to encode a greater variety of grasps as the robot collects more data for training.

As we noted in the section "Grasp Planning for Probabilistic Inference," our learned priors can be viewed as an approximation of the model uncertainty (i.e., epistemic [28]) of the learned classifier. In future work, we wish to compare prior learning with explicitly learning priors over the NN weights w, which would hopefully provide better-calibrated predictions of the probability of grasp success. Accurate models of the probability of grasp success would enable more reliable task-level planning, where the robot could reason over the probability of a sequence of events, producing the desired outcome under uncertainty of the manipulated object's shape and physical properties. However, it is unclear how one could use such Bayesian NNs to efficiently perform MAP inference for grasp planning, as a single evaluation of the NN uncertainty typically requires several forward-pass evaluations of the NN model [28].

A final weakness of our results as presented stems from our planner achieving better performance in attempted side grasps than overhead grasps. This presents significant issues in attempting to perform grasping in clutter or grasping of low-profile objects where the hand must be close to the table. Indeed, this problem arose in attempting to grasp the mug in this article. We think learning or designing a more complex feedback controller for overhead grasps using tactile feedback would boost the overhead-grasp performance, especially for objects with less contact areas on the top.

In conclusion, our article shows that we can improve grasp planning as MAP inference by incorporating three particular benefits. First, using a voxel-based object representation instead of an RGB-D improves learning performance. Second, learning MDN priors represents an improvement over uniform or object-independent learned priors. Three, unsurprisingly, more data representing grasps of increased variability improve grasp planning.

Nevertheless, several issues and open questions still remain with our planning framework as presented. Clearly learning-based approaches are becoming more and more prevalent if not the norm for manipulation. We hope the manipulation community takes hold of these questions and finds more to further our understanding of grasp planning as probabilistic inference.

Acknowledgments

Qingkai Lu and Balakumar Sundaralingam were supported in part by National Science Foundation award 1846341.

References

[1] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *Int. J. Robot. Res.*, vol. 27, no. 2, pp. 157–173, 2008. doi: 10.1177/0278364907087172.

[2] A. Saxena, L. L. S. Wong, and A. Y. Ng, "Learning grasp strategies with partial shape information," in *Proc. AAAI Nat. Conf. Artificial Intelligence*, 2008, pp. 1491–1494. doi: 10.1609/aaai.v33i01.33013542.

[3] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Robot. Res.*, vol. 34, nos. 4–5, pp. 705–724, 2015. doi: 10.1177/0278364914549607.

[4] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 3406–3413. doi: 10.1109/ICRA.2016.7487517.

[5] M. Kopicki, R. Detry, M. Adjigble, R. Stolkin, A. Leonardis, and J. L. Wyatt, "One-shot learning and generation of dexterous grasps for novel objects," *Int. J. Robot. Res.*, vol. 35, no. 8, pp. 959–976, 2016. doi: 10.1177/0278364915594244.

[6] A. Mousavian, C. Eppner, and D. Fox, 6-DOF GraspNet: Variational grasp generation for object manipulation. 2019. [Online]. Available: arXiv:1905.10520

[7] B. Wu, I. Akinola, and P. K. Allen, Pixel-attentive policy gradient for multi-fingered grasping in cluttered scenes. 2019. [Online]. Available: arXiv:1903.03227

[8] M. Liu, Z. Pan, K. Xu, K. Ganguly, and D. Manocha, Generating grasp poses for a high-DoF gripper using neural networks. 2019. [Online]. Available: https://arxiv.org/abs/1903.00425

[9] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Robot. Auton. Syst.*, vol. 60, no. 3, pp. 326–336, 2012. doi: 10.1016/j.robot.2011.07.016.

[10] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis-a survey," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 289–309, 2014. doi: 10.1109/TRO.2013.2289018.

[11] M. Ciocarlie, C. Goldfeder, and P. Allen, "Dimensionality reduction for hand-independent dexterous robotic grasping," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2007, pp. 3270–3275. doi: 10.1109/IROS.2007.4399227.

[12] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011, pp. 2845–2850. doi: 10.1109/ICRA.2011.5980395.

[13] R. A. Grupen, "Planning grasp strategies for multifingered robot hands," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 1991, pp. 646–651. doi: 10.1109/ROBOT.1991.131656.

[14] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC, 1994.

[15] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2016, pp. 598–605. doi: 10.1109/IROS.2016.7759114.

[16] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot. Res.*, vol. 37, nos. 4–5, p. 0278364917710318, 2016. doi: 10.1177/0278364917710318.

[17] J. Mahler et al., "Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Proc. Robotics Science and Systems (RSS)*, 2017. doi: 10.15607/RSS.2017.XIII.058.

[18] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2016, pp. 4461–4468. doi: 10.1109/IROS.2016.7759657.

[19] J. Varley, J. Weisz, J. Weiss, and P. Allen, "Generating multi-fingered robotic grasps via deep learning," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2015, pp. 4415–4420. doi: 10.1109/IROS.2015.7354004.

[20] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 1316–1322. doi: 10.1109/ICRA.2015.7139361.

[21] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2017, pp. 769–776. doi: 10.1109/IROS.2017.8202237.

[22] M. Veres, M. Moussa, and G. W. Taylor, "Modeling grasp motor imagery through deep conditional generative models," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 757–764, 2017. doi: 10.1109/LRA.2017.2651945.

[23] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 4304–4311. doi: 10.1109/ICRA.2015.7139793.

[24] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans, "Planning multi-fingered grasps as probabilistic inference in a learned deep network," in *Proc. Int. Symp. Robotics Research*, 2017, pp. 455–472.

[25] Y. Zhou and K. Hauser, "6-DoF grasp planning by optimizing a deep learning scoring function," in *Proc. Robotics: Science and Systems (RSS) Workshop on Revisiting Contact–Turning Problem Solution*, vol. 2, p. 6, 2017.

[26] Q. Lu and T. Hermans, "Modeling grasp type improves learning-based grasp planning," *IEEE Robot. Autom. Lett.,* vol. 4, no. 2, pp. 784–791, 2019. doi: 10.1109/LRA.2019.2893410.

[27] C. M. Bishop, "Mixture density networks," Neural Computing Research Group Report, Aston Univ., Birmingham, Tech. Rep. NCRG/94/004, 1994.

[28] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Proc. Advances Neural Information Processing Systems*, 2017, pp. 5575–5585.

[29] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM J. Sci. Comput.*, vol. 16, no. 5, pp. 1190–1208, 1995. doi: 10.1137/0916069.

[30] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, 1997. doi: 10.1145/279232.279236.

[31] 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.

[32] T. Hermans, J. M. Rehg, and A. F. Bobick, "Decoupling behavior, perception, and control for autonomous learning of affordances," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2013, pp. 4989–4996. doi: 10.1109/ICRA.2013.6631290.

[33] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2003, vol. 2, pp. 1824–1829. doi: 10.1109/ROBOT.2003.1241860.

[34] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3D database of object instances," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014, pp. 509–516. doi: 10.1109/ICRA.2014.6906903.

[35] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, Generative and discriminative voxel modeling with convolutional neural networks. 2016. [Online]. Available: arXiv:1608.04236

[36] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *Proc. IEEE Intl. Conf. Robotics and Automation (ICRA)*, 2015, pp. 4304–4311. doi: 10.1109/ICRA.2015.7139793. [Online]. Available: Http://ieeexplore.ieee.org/document/7139793/

[37] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *Proc. Int. Conf. Advanced Robotics (ICAR)*, 2015, pp. 510–517. doi: 10.1109/ICAR.2015.7251504.

Qingkai Lu, School of Computing and Robotics Center, University of Utah, Salt Lake City. Email: qklu@cs.utah.edu.

Mark Van der Merwe, School of Computing and Robotics Center, University of Utah, Salt Lake City. Email: mark.vandermerwe@utah.edu.

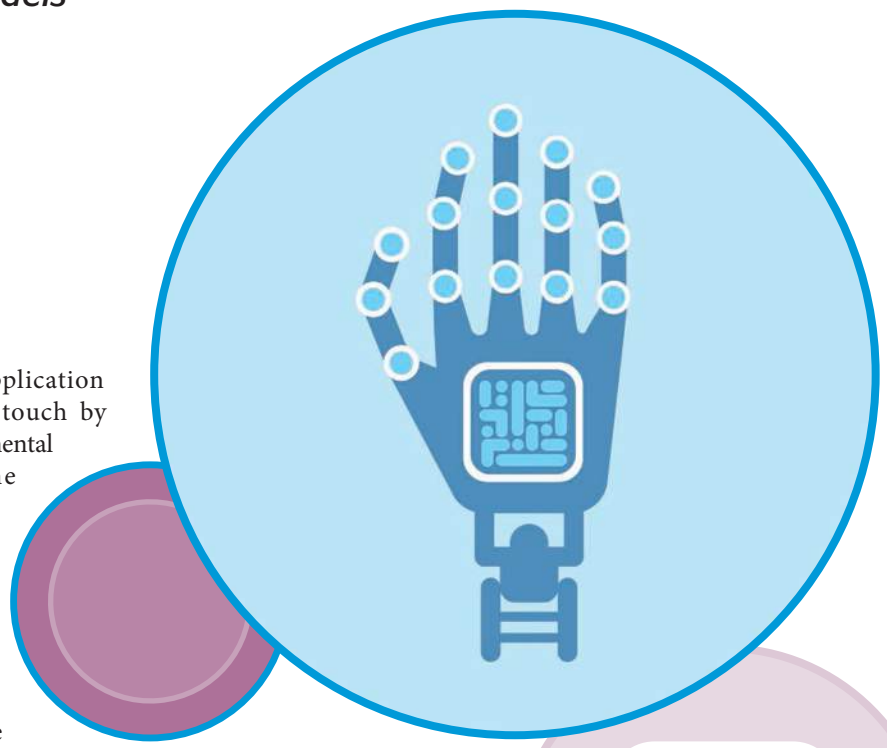
Balakumar Sundaralingam, School of Computing and Robotics Center, University of Utah, Salt Lake City. Email: bala@cs.utah.edu.

Tucker Hermans, School of Computing and Robotics Center, University of Utah, Salt Lake City. Email: thermans@cs.utah.edu.

Optimal Deep Learning for Robot Touch

Training Accurate Pose Models of 3D Surfaces and Edges

By Nathan F. Lepora and John Lloyd



This article illustrates the application of deep learning to robot touch by considering a basic yet fundamental capability: estimating the relative pose of part of an object in contact with a tactile sensor. We begin by surveying deep learning applied to tactile robotics, focusing on optical tactile sensors, which help to link touch and deep learning for vision. We then show how deep learning can be used to train accurate pose models of 3D surfaces and edges that are insensitive to nuisance variables, such as motion-dependent shear. This involves including representative motions as unlabeled perturbations of the training data and using Bayesian optimization of the network and training hyperparameters to find the most accurate models. Accurate estimation of the pose from touch will enable robots to safely and precisely control their physical interactions, facilitating a wide range of object exploration and manipulation tasks.

The Importance of Touch

Our primary human senses of vision, audition, and touch enable us to interact with a complex and ever-changing environment. Vision and audition are distal senses with which we reason about and plan our interactions. In contrast, touch is a proximal sense that enables us to interact directly with our surroundings, either to avoid harm or to explore and manipulate nearby objects. It has become something of a cliché to remark that if we want robots to interact in a useful way with our world, they will need versions of these three senses and the intelligence to use them effectively.

However, there is a huge disparity between the research effort invested in applying modern artificial intelligence (AI) to distal senses compared with proximal senses. For vision, there has been an explosion of interest, accounting for many tens of thousands of research papers. This has not been the case for touch, where there has been a more gradual increase to roughly 50 published studies. Only a handful of labs worldwide routinely apply deep learning to tactile sensing, and fewer still apply that expertise to control physically interactive robots.

One potential explanation for this disparity is the degree of synergy between artificial vision and deep learning. Originally, convolutional neural networks (CNNs) were inspired by the neuroanatomy and layered hierarchy from the retina

Digital Object Identifier 10.1109/MRA.2020.2979658
Date of current version: 7 April 2020

1070-9932/20©2020IEEE

through to the subcortical and cortical visual-processing structures of the brain. In 2012, AlexNet won at the ImageNet competition, sparking a revolution in computer vision. However, deep learning has spread quickly to other applications, such as speech recognition, so this initial impetus to vision does not explain the later disparity with touch.

In our view, the main barriers to applying deep learning to touch are 1) a lack of cheap, robust, and easy-to-use artificial tactile sensors, contrasting with modern cameras for computer vision; 2) the difficulty of obtaining high-quality tactile data due to a lack of public repositories and because a robot is usually needed to investigate the most interesting research questions; and 3) a lack of interest in the AI community for applying deep learning to touch. The latter barrier seems almost paradoxical, when most reports on AI aimed at policy makers and the public are filled with pictures of humanoid robots that will be useless in practice without functional hands.

In this article, we illustrate the application of deep learning to robot touch by considering a basic yet fundamental capability: estimating the relative pose of part of an object in contact with a fingertip. Tactile sensors can estimate the pose of the region of the object being contacted by inverting the tactile image into geometric features of the contact. However, finding the relation between high-dimensional tactile images and the low-dimensional pose is a challenge: Tactile sensors such as our fingertips are soft and curved, so physical interaction deforms the sensor in complex ways depending on the object shape, contact forces, and contact history. In our view, this difficulty has confined the use of robot touch to very primitive tasks compared with the fine motor capabilities of humans.

Accurate estimation of pose from touch will enable robots to safely and precisely control their physical interactions. For example, pose information can enable precise control of a fingertip sliding across complex objects, analogous to how humans trace their fingers across novel objects to explore shape. Previous work [1] demonstrated contour-following around planar objects in 2D using deep learning applied directly to the tactile images but required that the network be carefully hand-tuned; otherwise, the pose estimate would fail because the sensor sheared while sliding across the object. Here, we adopt a different approach by collecting training data that simulate the effect of shear and then using a black-box Bayesian optimizer to select the network architecture and other associated hyperparameters. In consequence, we demonstrate controlled sliding motion across complex 3D objects (Figure 1).

The main contributions of this research are to

1) show how deep learning can be used to train accurate models to estimate 3D pose from tactile images
2) develop pose-estimation models that are insensitive to nuisance variables, such as motion-dependent shear, by directly incorporating the variables into the data collection
3) introduce a systematic approach to model selection, which is needed for the most accurate models yet has not been used before for touch.

We also take the opportunity to survey deep learning applied to tactile robotics. In particular, we focus on optical tactile sensors (Figure 2) that use an internal camera to image skin deformation, since these sensors help bridge advances in deep learning for vision and the new domain of touch.

Background

Deep Learning for Tactile Sensing

Initial applications of deep learning to artificial tactile sensing were with taxel-based sensors, composed of discrete tactile elements embedded in a skin. The first study, in 2014, focused on tactile object recognition based on deep learning and dropout [3] and using a four-fingered robot hand with pressure-sensitive capacitive tactile arrays on the palm and finger joints/tips. Overall, the hands could recognize 20 objects held in a variety of grasps at a success rate of roughly 90%, using a

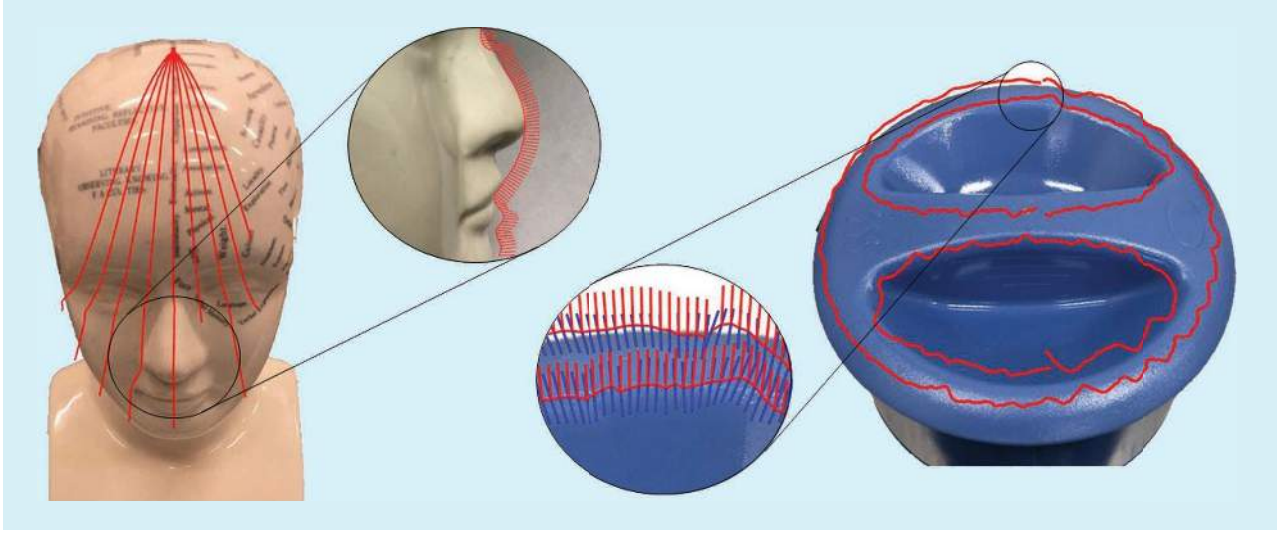


Figure 1. The robot trajectories on a complex 3D surface and edge (a porcelain bust and a container top) using pose estimation while sliding across the object.

CNN with 241 tactile inputs and 71 motor angles, currents, and force/torque readings. The authors also observed that they were likely the first to produce research in this area because of the difficulty of gathering many samples of high-dimensional data with tactile sensors.

During the next couple of years (2015–2016), a handful of studies followed. These applied deep learning to pressure-sensitive tactile arrays on the fingertips of robot hands using CNNs for tactile shape recognition [4], material texture classification [5], and slip identification [6] as well as a recurrent long short-term memory network for classifying held objects [7]. These studies used spatial and temporal data from their tactile arrays, which in combination give a high dimensional input suitable for deep learning. Related approaches have continued since, with a variety of taxel-based sensors and hands.

In 2016, a related area that combined tactile and visual data branched off from this early work. This was first applied to haptic adjectives from visual images and a single BioTac sensor [8], and the authors observed that NNs serve as a natural unifying framework for multimodal signal fusion. Soon after, visuo-tactile sensing was applied to object classification and grasp planning with a red–green–blue (RGB) depth camera and tactile arrays on a robot hand [9]. This area has continued to develop since, with multiple studies connecting look and feel [2], [10], [11], progressing more recently to deep-learning methods that can transform between [12] or match [13] visual and tactile data.

Optical Tactile Sensing for Deep Learning

During the past few years (2017–2019), the adoption of optical tactile sensors by several research groups has found a natural synergy with deep learning (Figures 2 and 3). These optical tactile sensors use an internal camera to image the deformation of a compliant sensing surface in contact with a physical stimulus. Although optical methods have been considered promising for tactile sensing since the 1980s [16], in practice, the field had been dominated by various means of electromechanical transduction. More recently, however, a coherent body of research on deep learning for robot touch has used optical transduction to make progress on previously intractable problems, as typified by most of the recent progress in visuo-tactile sensing [2], [10]–[13].

The first application of deep learning to an optical tactile sensor was for shape-independent hardness estimation using the GelSight [17] [Figure 2(b) and (d)]. Tactile images (960 × 720 pixels) were fed through a pretrained deep CNN into a recurrent (LTSM) network to predict the Shore hardness of the contacted object; training was with 7,000 video sequences (five frames each) across various object shapes and levels of hardness. A major benefit was that the hardness estimation was insensitive to nuisance variables, such as the object shape and loading of the contact, that have a complicated influence on the sensor output and would be very difficult to model otherwise. Since that first study, the GelSight with deep learning has been used for various tactile robotic problems, including most of the

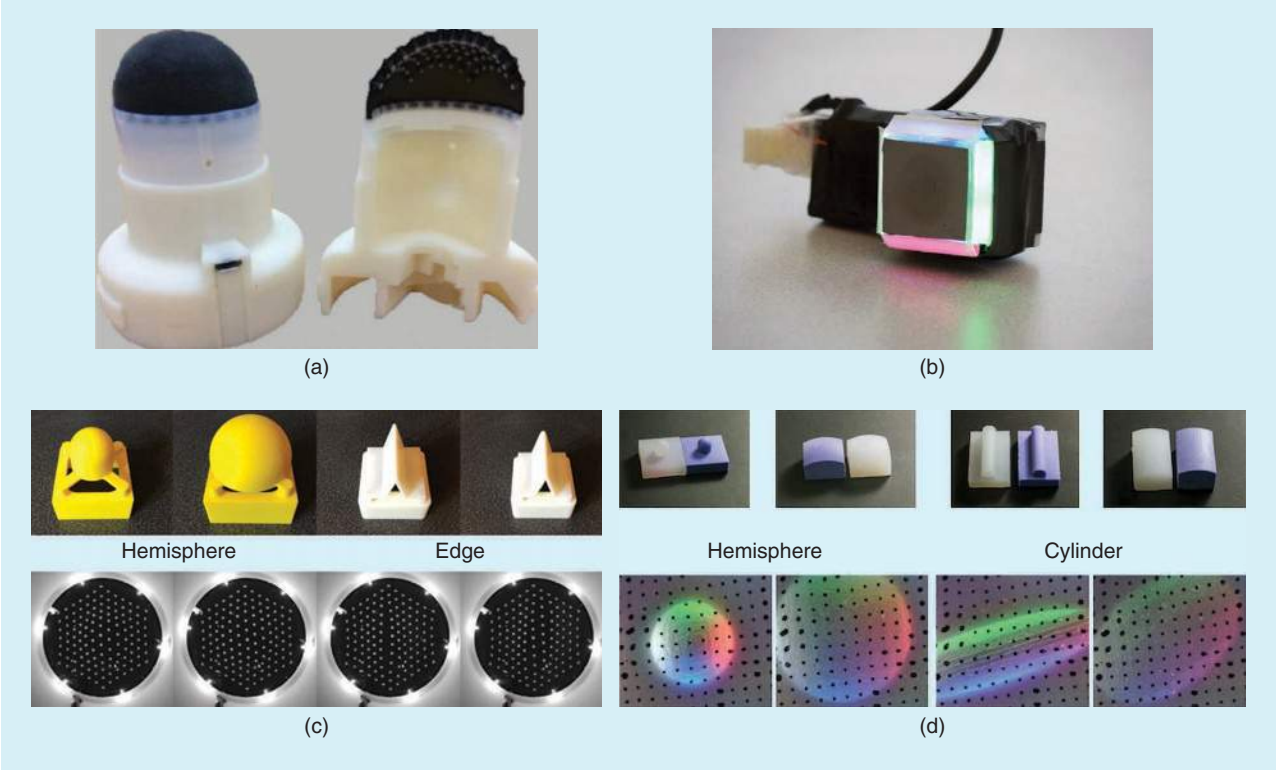


Figure 2. The (a) BRL TacTip biomimetic optical tactile fingertip and (b) GelSight optical tactile sensor. The bottom panels show (c) TacTip and (d) GelSight tactile images from manual contact against a few test stimuli. The GelSight tactile images are from [2], and the TacTip tactile images were generated for this article.

visuo-tactile studies mentioned previously [2], [10]–[13]; control of rolling motion [18]; and grasp stability [19] and grip readjustment [15] on two-digit grippers (Figure 3).

Recently, deep learning has been found highly suited to another type of optical tactile sensor: the BRL tactile fingertip, known as the *TacTip* (Figure 2). The first study considered robust edge perception for exploring objects with contour, following [1]. Tactile images (128 × 128 pixels) were fed into a CNN to predict the 2D edge pose; training was with 2,000 videos (five frames each) across a range of edge angles and positions. The challenge was to make the pose estimation insensitive to motion-dependent shear during sliding, which was addressed by hand-tuning the network architecture to have a stack of input convolutional layers that learned broader features across the tactile image. The TacTip has since been used with deep learning to identify benchmark objects and predict the grasp success of a three-finger robot hand [14] [Figure 3(a)] as well as to extract features for predicting the shape, 2D edge pose, and force with a convolutional autoencoder [20]. These studies used standard CNN architectures to make predictions that are insensitive to nuisance variables, such as the unknown object pose after automated grasping.

Why Optical Tactile Sensing?

Given our opening comments about the barriers to applying deep learning to touch, why has a large proportion of research in that area used optical tactile sensing? In our view, the GelSight and TacTip tactile sensors share some commonalities that suit deep learning; for example, both use an internal camera and internal light sources to image the skin deformation. That said, there are also significant differences that have implications for future development and use. Historically, both sensors were invented before the present revolution in AI. The first paper on the GelSight was in 2009, on retrographic sensing for the measurement of surface texture and shape [21]; the first paper on the TacTip was also in 2009, on the development of a tactile fingertip based on biologically inspired edge encoding [22]. Since then, both sensors have progressed in their designs and uses, but their operating principles have remained the same [23], [24].

The GelSight operates on the principle of measuring the normal indentation of a surface based on the shading of light reflected off the underside of that surface [21], [23] so that, with multiple light sources, a depth map can be reconstructed using a photometric stereo algorithm. The most common design uses internal RGB lights arranged symmetrically inside the sensor to give three sources of shading [Figure 2(b) and (d)]. Although the original intention was to measure indentation directly, when used with deep learning, the unprocessed RGB images are fed into the input layer of a CNN to extract tactile image features.

The TacTip operates by measuring the deformation of a surface from the shear displacement of markers on the tips of 3D pins protruding beneath that surface [22], [24] [Figure 2(b) and (d)]. This design is biomimetic because human fingertips have an analogous structure in which mechanoreceptors lie on papillae protruding between the epidermal and dermal skin layers. Originally, the pin tips were detected and tracked to give tactile features, most commonly as a time series of (x, y) displacements of the markers. However, when used with deep learning, the images are fed directly as inputs to a CNN.

In our view, the GelSight and TacTip sensors are well suited for deep learning because they both produce tactile images where every pixel gives information about surface deformation. For the GelSight, every pixel has RGB intensity readings that relate to indentation. For the TacTip, every pixel can signal whether or not a pin has moved to that location. Moreover, in recent years, the GelSight design has been modified to have markers inside the skin to indicate shear [23], which is similar to the operation of the TacTip (but without its 3D pin structure). Meanwhile, depth maps can be inferred indirectly from TacTip pin positions (for example, by using a Voronoi transformation).

PoseNet: 3D-Pose Estimation From Touch

To illustrate the application of deep learning to robot touch, we considered local pose estimation of a 3D surface in contact with the TacTip optical tactile sensor. Pose estimation is a fundamental capability of tactile sensing because knowledge of the relative pose enables a tactile robot to control its interactions. In the following, we refer to this NN as a *PoseNet* because it estimates the 3D pose from a tactile image.

Following previous work with optical tactile sensors [1], we used a standard CNN architecture to generate and process tactile features to predict the 3D pose (Figure 4). The tactile image feeds forward through a sequence of N_{conv} convolutional and max-pooling hidden layers, using N_{filters} filters in each layer and 3 × 3 kernels with unit stride (no padding). The resulting tactile features then pass through further N_{dense} fully connected hidden layers, each with N_{units} processing units that compute a dense matrix multiplication plus bias.

Figure 3. The (a) TacTip optical tactile sensor integrated with a robot hand and (b) GelSight optical tactile sensor integrated with a gripper. The TacTip-enabled hand is the BRL Tactile Model O [14]; the GelSight is integrated as the GelSlim sensor [15].

These processed features are combined linearly at the output layer to give N_{out} pose components. Unlike previous work in deep learning for tactile sensing, we did not hand-tune these network parameters or rely on values from other studies. Instead, we treated them as unknown hyperparameters to be optimized in the training process. This approach has been used widely elsewhere in deep learning, but, to the best of our knowledge, this is the first time it was applied to touch.

Other hyperparameters of the PoseNet architecture and training process that were optimized include the activation function used in the hidden layers [rectified linear unit (ReLU) or exponential linear unit (ELU)] and regularization parameters to avoid overfitting the training data, encompassing batch normalization, dropout, and L1/L2 regularization. Where batch normalization was used, it was inserted between the convolution operation and activation function in all convolutional hidden layers. Where dropout was employed, it was applied to the inputs of all dense hidden layers and the output layer, before the dense matrix multiplication. Where incorporated, L1 or L2 regularization was applied to the weights in all dense hidden and output layers. The reason for including more than one type of regularization approach is that the methods tend to work in complementary ways: Batch normalization and dropout inject noise into the training process, helping to prevent the complex coadaptation of features, whereas L2 regularization encourages smooth mappings, and L1 regularization encourages sparse models.

In this article, we considered two distinct types of PoseNet: one for a 3D surface and another for a 3D edge (Figure 4). In addition to differences in their trained parameters, the networks have different outputs. The network for the 3D surface has three outputs: the depth, roll, and pitch of the contact; the network for the 3D edge has five outputs: horizontal distance, depth, roll, pitch, and yaw. These parameters are sufficient to describe the pose of a tactile sensor in contact with a surface or contour by defining the local pose relative to a plane or line on a tangent to those objects.

Training Data and Network Optimization

One contribution of this article is to identify some subtleties in the data collection and hyperparameter optimization process when developing deep NNs for robot touch. Robot touch senses via contact, so the manner of contact affects the tactile data. To predict the pose accurately, the trained network should be insensitive to how the sensor has reached its current pose; for example, the motion of the sensor sliding across a surface or along an edge (Figure 1). Here, we included representative examples of these motions as unlabeled perturbations of the training data. However, these perturbations make it more difficult to predict the pose, to the extent that improperly tuned hyperparameters gave extremely suboptimal results. For a systematic approach, we used automatic-tuning methods, specifically Bayesian optimization.

Training Data Collection

Three data sets were collected for each of the two objects considered here (a flat surface and straight edge; Figure 5), giving independent training, validation, and testing data. Each set had more than 2,000 contacts, with the poses sampled randomly from uniform distributions within their allowed ranges (Table 1). Separate training and validation sets were used to mitigate the potential data set shift, rather than using a randomized split. The data consisted of single tactile images cropped and subsampled to a 128 × 128-pixel region that viewed all tactile elements of the sensor surface. These images were then preprocessed with an adaptive threshold to produce binary (black/white) inputs to the NN (see the examples in Figure 6), reducing the sensitivity to changes of the lighting inside the sensor.

We found it crucial to mimic the effect of motion on the tactile data while collecting the training data because motion-dependent shear affects measurements from optical tactile sensors, such as the TacTip. Each sample of data had a random labeled pose and a random unlabeled shear perturbation (the ranges are given in Table 1). First, the sensor was brought into contact at the labeled target pose minus the unlabeled perturbation, then moved to the target pose; at that point, the

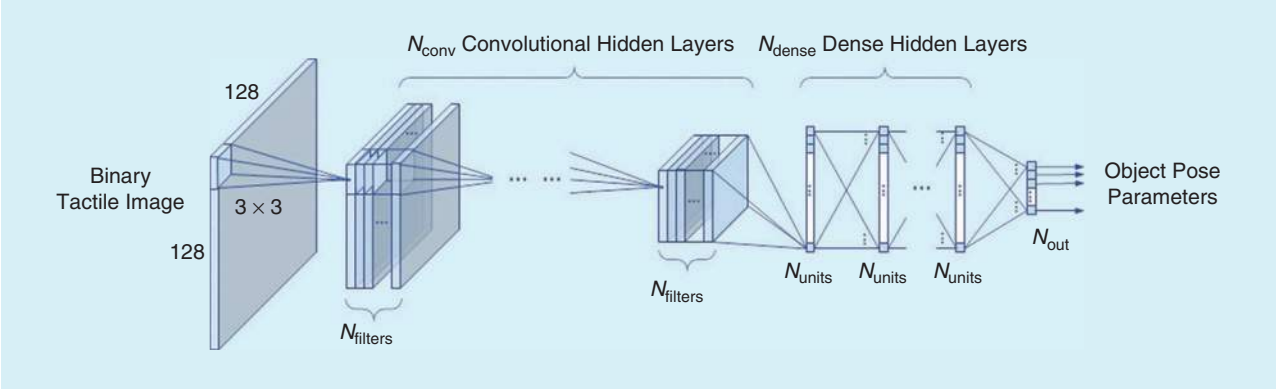


Figure 4. The PoseNet CNN. A binary sensor image is processed by a sequence of N_{conv} convolutional layers, each containing $N_{filters}$ filters. The resulting convolutional features then pass through a sequence of N_{dense} fully connected hidden layers, each containing N_{units} processing units, to produce a high-level feature vector. The high-level features are combined by a linear output layer to give N_{out} continuous-valued object pose parameters.

tactile image was collected. The perturbations took uniform random values within ranges typical of movements that may advance the sensor without damaging it, in all pose components except the vertical. Ranges for the target poses and the unlabeled perturbations were chosen according to how much the sensor could move to remain safely in contact without causing damage.

Model Training

As is common practice in NN regression problems, the network weights and biases were optimized during training by minimizing the mean square error (MSE) between the predicted outputs and target labels. The loss components were weighted by $1/\text{maximum}^2$ for the parameters in Table 1 (e.g., depth: $1/5^2$; roll: $1/15^2$; and yaw: $1/45^2$) to give peak losses of order one. Several different regularization approaches were used to avoid overfitting the training data, including constraining the number and size of the hidden layers and using early stopping during training. These choices were part of the hyperparameter optimization process described in the following.

All PoseNet models were trained using the Adam adaptive learning rate optimizer (initial learning rate: 10^{-4}; decay coefficient: 10^{-6}). This optimizer consistently gave good solutions and seemed to converge more quickly than other types. Prior to training, all weights were initialized to small random values. Early stopping was incorporated to terminate the optimization process, where the MSE loss was computed for a separate validation data set and the process halted when there was no further improvement through 10 epochs. Training and optimization of the deep NNs were implemented in the Keras library on a Titan Xp graphics processing unit (GPU) (12-Gb memory) hosted on a Windows 10 PC. A training run typically takes 5–20 min, depending on the size of the network.

Hyperparameter Optimization

Bayesian optimization was employed to give a systematic approach to setting the network and training hyperparameters. The PoseNet models were optimized using 300 cost function evaluations, incorporating 50 random startup evaluations, from the MSE loss on the validation data set after early stopping. Bayesian optimization is a sequential, derivative-free, black-box method that is well suited to noisy and expensive-to-evaluate cost functions. It incrementally constructs an acquisition function that directs cost function sampling to areas where improvement is likely. We used a Python implementation of Bayesian optimization [HyperOpt (http://hyperopt.github.io/hyperopt/], which constructs a generative, nonparametric model of an acquisition function, the Tree-Structured Parzen Estimator. Overall, the optimization and training processes took 1–2 days on our hardware.

The model configurations were constrained to lie within a search space bounded by parameter ranges that took account of the available computing resources and reasonable values of the hyperparameters (Table 2). For example, given the 128 × 128 dimension of the input tactile image and the 2 × 2 max-pooling architecture of the network, the number of convolutional layers $N_{\text{conv}} \leq 5$, and computing resources constrained the number of filters per layer $N_{\text{filter}} \leq 512$ and batch size ≤ 256. We emphasize that it was not obvious a priori where

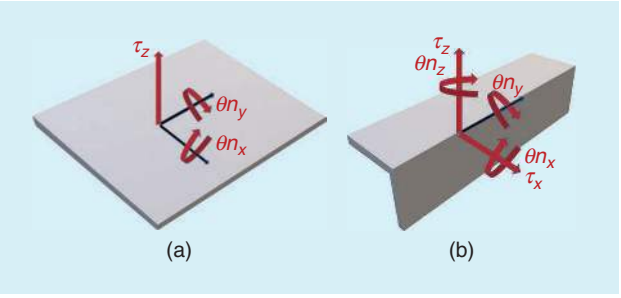


Figure 5. The pose parameters for (a) the 3D surface and (b) the 3D edge. The surface has three parameters: depth, roll, and pitch; the edge has five parameters: *x*-horizontal, depth, roll, pitch, and yaw. The labels are in axis–angle coordinates ($\boldsymbol{\tau}$, $\theta\boldsymbol{n}$).

Table 1. The training data pose parameter ranges.

	Labeled		Unlabeled
Parameter	**3D Surface**	**3D Edge**	**Perturbation**
x-horizontal	–	[–5, 5] mm	[–5, 5] mm
y-horizontal	–	–	[–5, 5] mm
Depth	[–5, –1] mm	[–5, –1] mm	0 mm
Roll	[–15, 15]°	[–15, 15]°	[–5, 5]°
Pitch	[–15, 15]°	[–15, 15]°	[–5, 5]°
Yaw	–	[–45, 45]°	[–5, 5]°

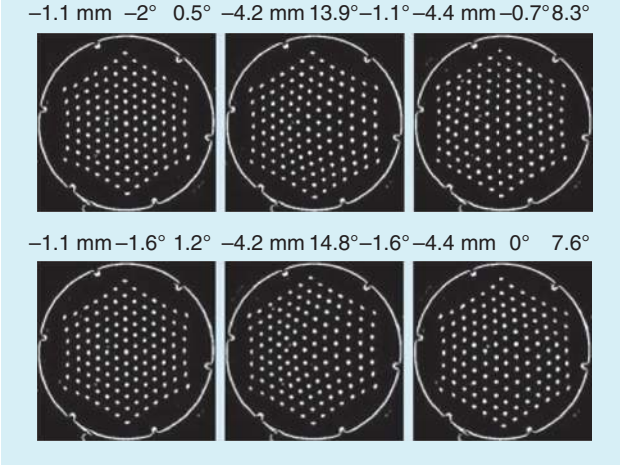


Figure 6. The example preprocessed tactile images for the 3D surface and their pose labels (depth, roll, and pitch). The tactile images were selected to have similar labels for each column (equal depth; roll and pitch within 1°). Differences between rows are due to the unlabeled random perturbations in the pose, and are most apparent for the strongest/deepest contacts (right column).

the optimal parameter values would be, so these ranges are the broadest that were reasonable for the problem; other network and learning hyperparameters could be considered, but this set of 10 seemed reasonable based on past experience with deep learning using this sensor.

Tactile Pose Estimation for a 3D Surface

3D Surface Model Results

The performance of the trained PoseNet was assessed on a third test data set gathered for this purpose, consisting of 2,000 tactile images labeled with depth, roll, and pitch pose parameters. The data were subject to random unlabeled perturbations in the target pose prior to collecting the tactile image, using the procedure described in the "Training Data Collection" section. All labeled poses and unlabeled perturbations were sampled randomly within the same ranges as the training and validation data (Table 1).

Table 2. The search ranges used to optimize the PoseNet hyperparameters.

Parameter	Range	Distribution
Number of convolutional hidden layers, N_{conv}	{1, 2, 3, 4, 5}	Uniform
Number of convolutional filters, $N_{filters}$	{2, 4, 8, …, 512}	Uniform
Number of dense hidden layers, N_{dense}	{1, 2, 3, 4, 5}	Uniform
Number of dense hidden layer units, N_{unit}	{2, 4, 8, …, 512}	Uniform
Hidden-layer activation function	{ReLU, ELU}	Uniform
L1-regularization coefficient	$[10^{-4}, 10^{-1}]$	Log uniform
L2-regularization coefficient	$[10^{-4}, 10^{-1}]$	Log uniform
Dropout coefficient	[0, 0.5]	Uniform
Batch size	{16, 32, 64, 128}	Uniform

Highly accurate performance was obtained for the optimized PoseNet, with a close match between the predicted and labeled pose parameters (Figure 7). The mean average error (MAE) between predictions and labels was used to summarize the model performance, giving values of 0.1 mm for the depth and 0.3° for the roll and pitch. These results are precise compared with the size of the tactile sensor (40-mm diameter tip) and pin spacing (4 mm); also, the TacTip was not designed for this task but was a standard version used in our lab [24]. We emphasize that the precise accuracies hold even though the data were randomly perturbed by an unknown shearing motion that significantly affected the tactile image (Figure 6).

Analysis of 3D Surface Model Optimization

The benefit of optimizing the hyperparameters is evident from the 1,000-fold decrease in the validation loss through the 300 trials of training [Figure 8(a)]. The first 50 trials featured a period of high scattered loss during the startup evaluations ($0.05 \lesssim \text{MSE} \lesssim 2$). Then, within another 20 trials, the optimizer quickly found a good model ($\text{MSE} \simeq 0.01$). Another 150 trials were needed before the optimizer improved substantially to give the best loss ($\text{MSE} \simeq 0.005$), by which time it consistently gave models with low losses. The convergence of the optimization process can also be seen by plotting the hyperparameters against their corresponding losses, visualized as scatter plots ordered by the loss (Figure 8). As the loss becomes small (to the left), the

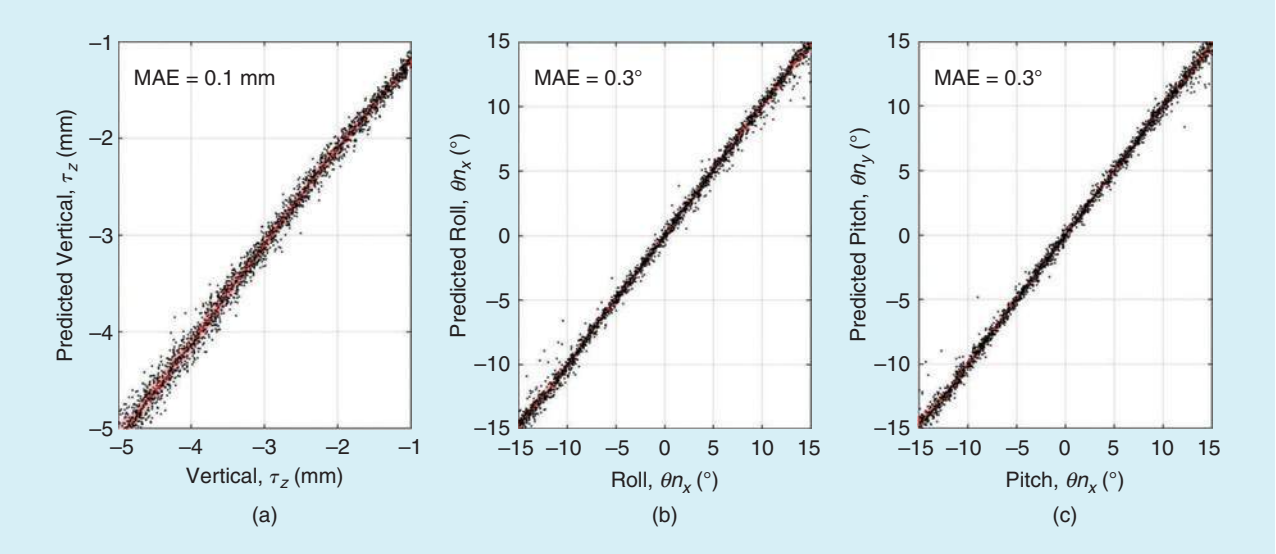


Figure 7. The optimal PoseNet performance for the 3D surface, with the predicted (a) vertical, (b) roll, and (c) pitch results. The smoothed predictions (red: 100-sample moving average), region within the smoothed absolute error (pink), and MAEs are shown.

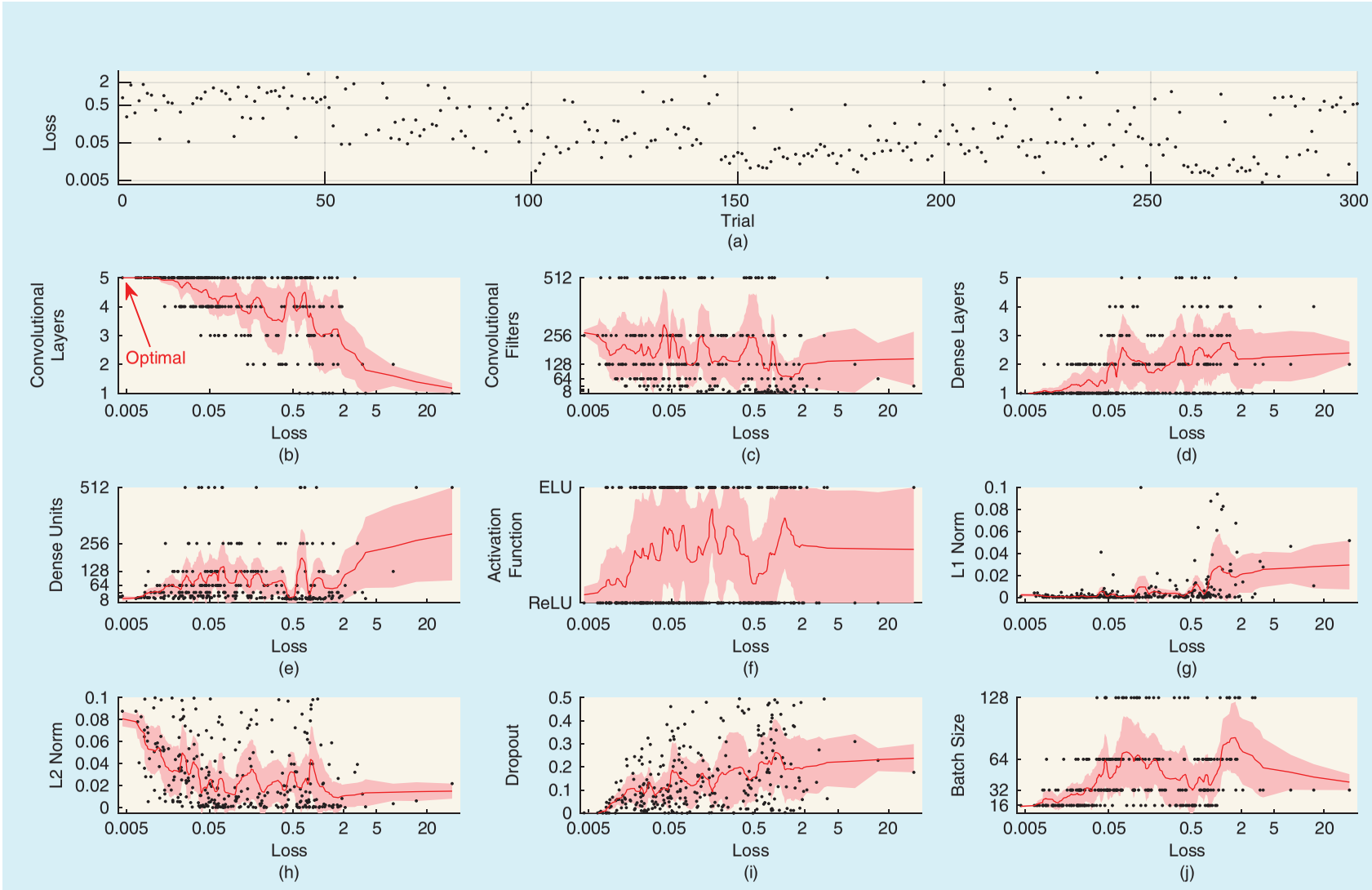


Figure 8. The hyperparameter optimization for the 3D surface. (a) The loss against the trial number. (b)–(j) The hyperparameter dependence on the loss, for each of the parameters in Table 2. The optimal network is to the left of the scatter plot.

hyperparameter values become more consistent near the optimum, rather than being scattered.

Overall, the optimization favored network architectures (Table 3) that had a deep convolutional stage (five hidden layers); a shallow, fully connected stage (one hidden layer); the most convolutional filters allowed (512 per layer); a modest number of dense units (16); and ReLU activation functions. For the training hyperparameters, small batch sizes (16) were preferred, likely because these data are regular enough to not require many samples to find a reasonable stochastic gradient. Also, larger batch sizes can sometimes exceed the GPU's memory, resulting in a failed run. Dropout was hardly used (0.001), which may be because the convolutional layers were already heavily regularized due to weight sharing and the number of dense layers and their sizes were already constrained. There was little L1 regularization but some L2 regularization (0.001 and 0.064), which again might be because it was better to constrain the number and sizes of the hidden layers.

Tactile Pose Estimation for a 3D Edge

3D Edge Model Results

The optimized PoseNet performance was assessed for the 3D edge on a third data set, consisting of 2,000 tactile images labeled with the horizontal pose, depth, roll, pitch, and yaw. The data were again subject to unlabeled pose perturbations, with the poses and perturbations sampled randomly (Table 2). Accurate performance was obtained (Figure 9), with MAEs between the predicted and labeled pose parameters of 0.6 and 0.2 mm and 1.4, 1.6, and 6.4°. Clearly, these are less accurate than the surface, which reflects the fact that pose prediction for the edge is a far more difficult regression problem. For example, the edge has two more pose parameters than the surface, but the same amount of training data was used. Also, by the nature of its geometry, the edge can feel similar across distinct poses: One example is that contacts at the horizontal extremes are ambiguous under the yaw. These difficulties emphasize the importance of finding the best possible model fit using hyperparameter optimization.

Table 3. The optimal PoseNet hyperparameters.

Parameter	3D Surface	3D Edge
Number of convolutional hidden layers, N_{conv}	5	5
Number of convolutional kernels, $N_{filters}$	512	256
Number of dense hidden layers, N_{dense}	1	2
Number of dense hidden layer units, N_{unit}	16	512
Hidden-layer activation function	ReLU	ReLU
Dropout coefficient	0.001	0.203
L1-regularization coefficient	0.001	0.0001
L2-regularization coefficient	0.064	0.0003
Batch size	32	16

Analysis of 3D Edge Model Optimization

For the 3D edge model, the hyperparameter optimization resulted in a 10-fold reduction of the loss across the 300 training trials, with the lowest losses after trial [Figure 10(a) and (b)]. This improvement in the loss was substantial but much less than for the 3D surface, which again reflects the fact that estimating the pose of the edge is a more difficult problem. The losses were in the range 0.1–1 MSE, with the lowest losses occurring only near the end of the optimization. The convergence of the optimization can also be seen in scatter plots of the hyperparameters (Figure 10), with the lowest losses bunched at the left of the plots.

Overall, the optimization favored network architectures (Table 3) that had a deep convolutional stage (five hidden layers), a fairly shallow fully connected stage (two hidden layers), many convolutional filters (256 per layer), a moderate number of dense units (64), and ReLU activation functions. For the training hyperparameters, dropout was strongly preferred (0.2), and L1 and L2 regularization were rarely used ($\lesssim 0.001$). Modest batch sizes were again preferred. The network and training hyperparameters were similar to those for the surface, apart from a greater use of dropout with the ReLU activation function. This appears to be a strategy for coping with the more difficult pose estimation problem for the edge. The different configuration emphasizes the benefit of an automated process for model optimization.

Demonstration of 3D Object Exploration

Pose estimation is a fundamental capability of tactile sensing because knowledge of the relative pose enables a robot to control its interactions. The local pose gives information about how to reposition the tactile sensor to maintain contact while moving safely across the object. To show pose estimation in action, Figure 1 displays trajectories generated by the PoseNet models in this article applied to controlling a robot moving across a complex 3D surface and edge.

We used a robot system consisting of a TacTip mounted on a 6-degrees-of-freedom robot arm (IRB 120, ABB robotics). This robot was previously used to study 2D contour-following [1]; we refer to that paper for details of the robot, software infrastructure, and literature on tactile servoing. To demonstrate 3D surface following, we extended the previous 2D control policy to a 3D surface (three pose variables) and a 3D edge (five pose variables). This policy has two aims: 1) move the sensor to remain normal to the object surface and 2) move the sensor tangentially along the surface (by 1 mm per time step t). Here, we implemented a proportional-integral controller in discrete time with, as output, a change in the 3D pose of the sensor in its reference frame:

$$\Delta s(t) = K_p \boldsymbol{e}(t) + K_i \sum_{t=0}^{t} \boldsymbol{e}(t').$$

The gain matrices K_p, K_i were diagonal with proportional gains of 0.5 and integral gains of 0.3 and 0.1 for translations and rotations, respectively. The error $\boldsymbol{e}$ was evaluated between the predicted pose and a reference normal to the surface or edge.

Using the PoseNet models (see the "Optical Tactile Sensing for Deep Learning" section) for tactile pose estimation of a 3D surface and 3D edge (discussed in the "Tactile Pose Estimation for a 3D Surface" section) resulted in successful object exploration across two complex 3D objects: surface exploration across a porcelain bust and edge-following around a container top (Figure 1). The exploration encompassed a region bounded by the robot's reach and a joint-space singularity across the bust (on the right cheek). The estimated pose is shown in the insets, with the surface normals in red and edge normals in blue along the trajectories.

Conclusions

In this article, we showed how optimal deep learning can give highly accurate models for robot touch using tactile images from optical tactile sensors. To illustrate this application, we considered local pose estimation of a 3D surface or edge in contact with the BRL tactile fingertip (TacTip), a biomimetic optical tactile sensor. To predict the pose accurately, the trained network should be insensitive to how the sensor has reached its current pose. We developed models that were insensitive to motion-dependent shear by including representative examples of these motions as unlabeled perturbations of the training data. However, these made it difficult to predict the pose, to the extent that improperly tuned models gave extremely suboptimal results. For a systematic approach, we introduced Bayesian optimization of the network and training hyperparameters to find the most accurate models. Consequently, the models were highly accurate (e.g., a 0.1-mm depth and 0.3° surface orientation) even though the data were randomly perturbed by an unknown shearing motion. The models were also robust to the surface shape and texture, as demonstrated by using the predicted poses to control a robot sliding the sensor across complex 3D objects (Figure 1).

Overall, the approach for robot touch introduced here offers the potential for safe and precise physical interaction with complex environments, encompassing tasks from exploring natural objects to closed-loop dexterous manipulation. Even though we used the TacTip optical tactile sensor, a similar approach should apply to other high-accuracy tactile sensors, such as the GelSight, provided that they can slide repeatedly across objects without damage. This work aims to bring artificial tactile sensing one step closer to human performance and so raises the question of whether humans use similar strategies during their own tactile interactions. In our view, soft tactile sensors such as human fingertips cannot function usefully in natural environments unless they have a

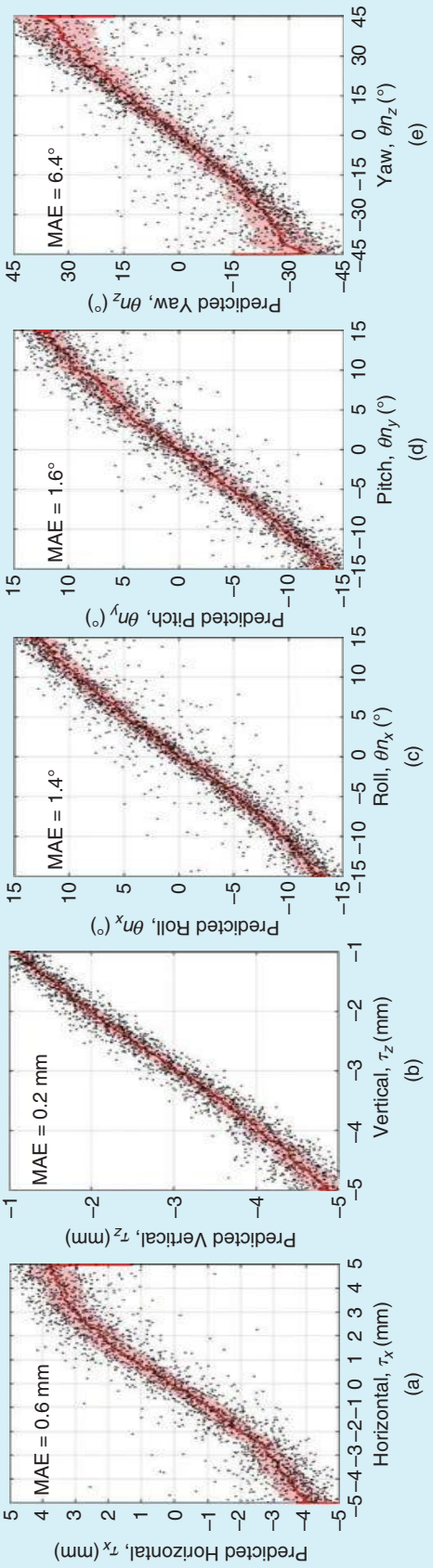


Figure 9. The optimal PoseNet performance for the 3D edge. The smoothed predictions (red: 100-sample moving average), region within the smoothed absolute error (pink), and MAEs are shown. The (a) horizontal, (b) vertical, (c) roll, (d) pitch, and (e) yaw results.

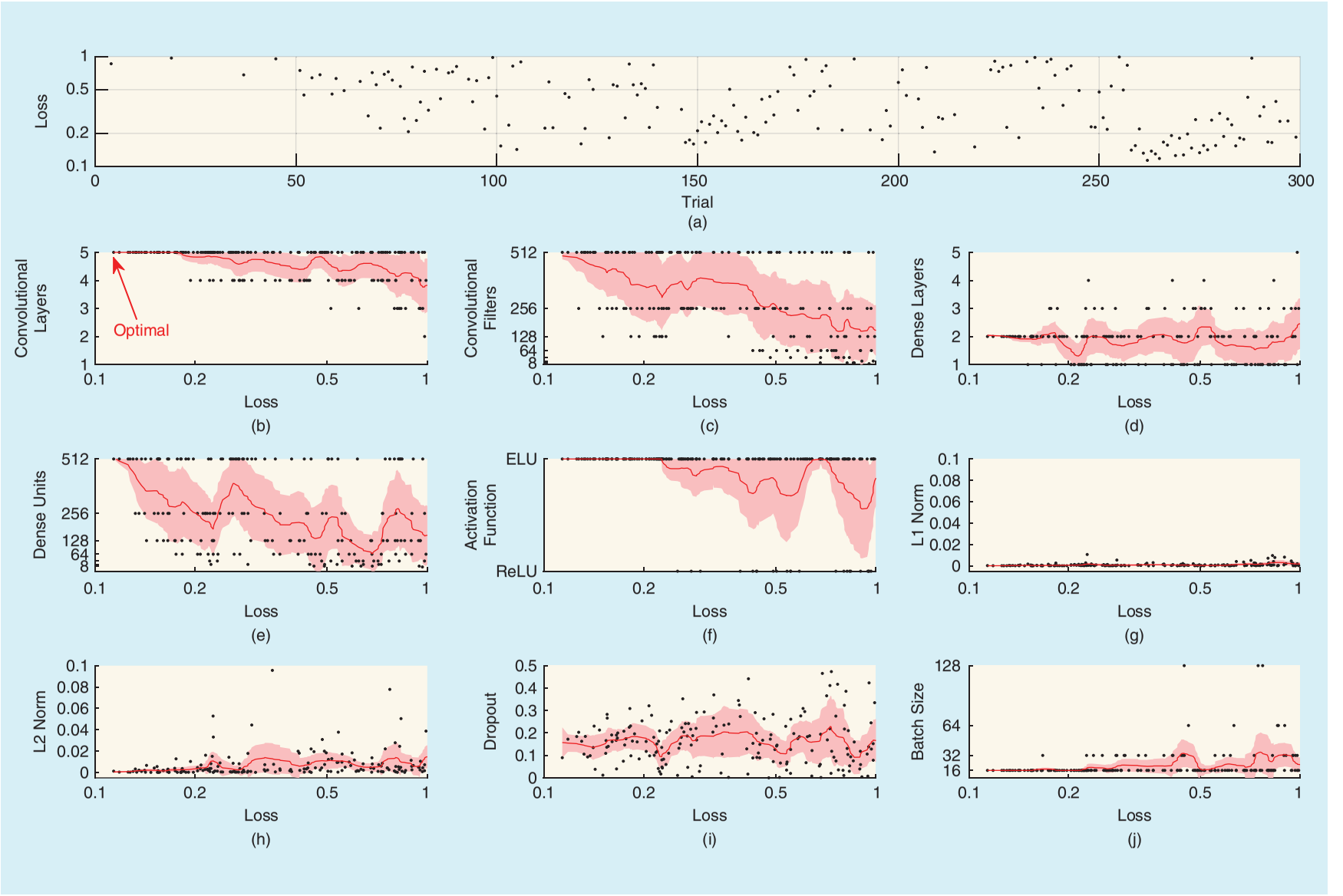


Figure 10. The hyperparameter optimization for the 3D edge. (a) The loss against trial number. (b)–(j) The hyperparameter dependence on the loss, for each of the parameters in Table 2.

perceptual system with invariance to contact motion. As demonstrated here, appropriately trained deep NNs can solve that problem.

Acknowledgments

We thank NVIDIA for the donation of the Titan Xp GPU used for this research. This work was supported by an award from the Leverhulme Trust for "A Biomimetic Forebrain for Robot Touch" (RL-2016-39). Nathan F. Lepora and John Lloyd contributed equally to this work.

References

[1] N. Lepora, A. Church, C. de Kerckhove, R. Hadsell, and J. Lloyd, "From pixels to percepts: Highly robust edge perception and contour following using deep learning and an optical biomimetic tactile sensor," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2101–2107, 2019. doi: 10.1109/LRA.2019.2899192.

[2] W. Yuan, S. Wang, S. Dong, and E. Adelson, "Connecting look and feel: Associating the visual and tactile properties of physical materials," *in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4494–4502. doi: 10.1109/CVPR.2017.478.

[3] A. Schmitz, Y. Bansho, K. Noda, H. Iwata, T. Ogata, and S. Sugano, "Tactile object recognition using deep learning and dropout," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 1044–1050. doi: 10.1109/HUMANOIDS.2014.7041493.

[4] L. Cao et al., "Efficient spatio-temporal tactile object recognition with randomized tiling convolutional networks in a hierarchical fusion strategy," in *Proc. AAAI Conf. Artificial Intelligence*, 2015, pp. 3337–3345.

[5] S. Baishya and B. Bäuml, "Robust material classification with a tactile skin using deep learning," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2016, pp. 8–15. doi: 10.1109/IROS.2016.7758088.

[6] M. Meier, F. Patzelt, R. Haschke, and H. Ritter, "Tactile convolutional networks for online slip and rotation detection," in *Proc. Int. Conf. Artificial Neural Networks (ICANN)*, 2016, pp 12–19. doi: 10.1007/978-3-319-44781-0_2.

[7] G. Buscher, M. Meier, G. Walck, R. Haschke, and H. Ritter, "Augmenting curved robot surfaces with soft tactile skin," in *Proc. 2015 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 1514–1519. doi: 10.1109/IROS.2015.7353568.

[8] Y. Gao, L. Hendricks, K. Kuchenbecker, and T. Darrell, "Deep learning for tactile understanding from visual and haptic data," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2016, pp. 536–543. doi: 10.1109/ICRA.2016.7487176.

[9] F. Sun, C. Liu, W. Huang, and J. Zhang, "Object classification and grasp planning using visual and tactile sensing," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 7, pp. 969–979, 2016. doi: 10.1109/TSMC.2016.2524059.

[10] R. Calandra et al., "More than a feeling: Learning to grasp and regrasp using vision and touch," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3300–3307, 2018. doi: 10.1109/LRA.2018.2852779.

[11] S. Luo, W. Yuan, E. Adelson, A. Cohn, and R. Fuentes, "ViTac: Feature sharing between vision and tactile sensing for cloth texture recognition," in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 2722–2727. 2018. doi: 10.1109/ICRA.2018.8460494.

[12] J. Lee, D. Bollegala, and S. Luo, "Touching to see" and "seeing to feel": Robotic cross-modal sensory data generation for visual-tactile perception," in *Proc. 2019 Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 4276–4282. doi: 10.1109/ICRA.2019.8793763.

[13] J. Lin, R. Calandra, and S. Levine, "Learning to identify object instances by touch: Tactile recognition via multimodal matching," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 3644–3650. doi: 10.1109/ICRA.2019.879388.

[14] J. James, A. Church, L. Cramphorn, and N. Lepora, Tactile model O: Fabrication and testing of a 3d-printed, three-fingered tactile robot hand. 2019. [Online]. Available: https:arXiv:1907.07535

[15] F. Hogan, M. Bauza, O. Canal, E. Donlon, and A. Rodriguez, "Tactile regrasp: Grasp adjustments via simulated tactile transformations," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018, pp. 2963–2970. doi: 10.1109/IROS.2018.8593528.

[16] P. Dario and D. De Rossi, "Tactile sensors and the gripping challenge," *IEEE Spectr.*, vol. 22, no. 8, pp. 46–53, 1985. doi: 10.1109/MSPEC.1985.6370785.

[17] W. Yuan, C. Zhu, A. Owens, M. Srinivasan, and E. Adelson. "Shape-independent hardness estimation using deep learning and a GelSight tactile sensor," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2017, pp. 951–958. doi: 10.1109/ICRA.2017.7989116.

[18] S. Tian et al., "Manipulation by feel: Touch-based control with deep predictive models," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 818–824. doi: 10.1109/ICRA.2019.8794219.

[19] R Calandra et al., "The feeling of success: Does touch sensing help predict grasp outcomes?" in *Proc. Conf. Robot Learning (CoRL)*, 2017, pp. 1–10.

[20] M. Polic, I. Krajacic, N. Lepora, and M. Orsag, "Convolutional autoencoder for feature extraction in tactile sensing," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, p. 1, 2019. doi: 10.1109/LRA.2019.2927950.

[21] M. Johnson and E. Adelson, "Retrographic sensing for the measurement of surface texture and shape," in *Proc. 2009 IEEE Computer Society Conf. Computer Vision and Pattern Recognition Workshops, CVPR Workshops,* pp. 1070–1077. doi: 10.1109/CVPR.2009.5206534.

[22] C. Chorley, C. Melhuish, T. Pipe, and J. Rossiter, "Development of a tactile sensor based on biologically inspired edge encoding," in *Proc. Int. Conf. Advanced Robotics*, 2009, pp. 1–6.

[23] W. Yuan, S. Dong, and E. Adelson, "GelSight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, p. 2762, 2017. doi: 10.3390/s17122762.

[24] B. Ward-Cherrier et al., "The TacTip family: Soft optical tactile sensors with 3d-printed biomimetic morphologies," *Soft Robot.*, vol. 5, no. 2, pp. 216–227, 2018. doi: 10.1089/soro.2017.0052.

Nathan F. Lepora, Department of Engineering Mathematics, Faculty of Engineering, University of Bristol and Bristol Robotics Laboratory, United Kingdom. Email: n.lepora@bristol.ac.uk.

John Lloyd, Department of Engineering Mathematics, Faculty of Engineering, University of Bristol and Bristol Robotics Laboratory, United Kingdom. Email: jl15313@bristol.ac.uk.

By Axier Ugartemendia, Daniel Rosquete,
Jorge Juan Gil, Iñaki Díaz, and Diego Borro

Robotic rehabilitation for poststroke therapies is an emerging new domain of application for robotics with proven success stories and clinical studies. New robotic devices and software applications are hitting the market, with the aim of assisting specialists carrying out physical therapies and even patients exercising at home. Rehabilitation robots are designed to assist patients performing repetitive movements with their hemiparetic limbs to regain motion. A successful robotic device for rehabilitation demands high workspace and force

Machine Learning for Active Gravity Compensation in Robotics

©ISTOCKPHOTO.COM/ANDRESR

Application to Neurological Rehabilitation Systems

Digital Object Identifier 10.1109/MRA.2020.2978484
Date of current version: 3 April 2020

1070-9932/20©2020IEEE

feedback capabilities similar to a human physiotherapist. These desired features are usually achieved at the expense of other important requirements, such as transparency and backdrivability, degrading the overall human–machine interaction experience.

We present an active gravity compensation method that can significantly improve the performance of mechatronic systems used for rehabilitation and many other domains of robotic applications. Traditional algorithms to obtain active gravity compensation usually require the system's static equilibrium equations. However, for complex mechatronic configurations, solving these equations is not straightforward. The use of machine learning (ML) methods can achieve gravity compensation without the need to solve the equilibrium equations. To validate the performance of the proposed approach, the HomeRehab robotic rehabilitation system is used to obtain experimental results.

Robotic Rehabilitation

Stroke is currently the second most frequent cause of death, after coronary artery disease, and its prevalence is increasing at an alarming rate. Hemiparesis resulting in movement disabilities is the most common outcome of stroke. Fortunately, rehabilitation can help hemiparetic patients learn new ways of using and moving their weak arms and legs. It is also possible with immediate therapy that people who suffer from hemiparesis may eventually regain movement. Although there are several approaches, extensive task-specific repetitive movement is one of the safest and most effective methods to regain the lost mobility of the affected limbs. This therapy requires continual medical care and intensive rehabilitation, which often necessitates one-on-one manual interaction with the physical therapist.

Robotic rehabilitation is an emerging field that enables a patient to perform exercises with the assistance of a robotic device [1]. These systems can be used to provide therapy (even at the patient's home) for a long period of time, irrespective of skills and fatigue, compared to manual therapy. Besides the cost-effective aspect, robotic devices introduce higher accuracy and repeatability in performing exercises. Precise measurements of quantitative parameters by means of robotic instrumentation also improve the objective monitoring of patients' recovery. Furthermore, rehabilitation robots can be combined with virtual-reality environments to engage and push patients to keep training, as the patients' motivation and cognitive involvement have a great impact on the outcome of rehabilitation [2].

Currently, several devices on the market provide a robotic solution to these repetitive movements, and these have been installed in many hospitals around the world. Some examples are InMotion ARM (Bionik Labs), ReoGo (Motorika), and Armeo Power (Hocoma). For a successful rehabilitation system, the robot must be able to deliver physical forces similar to manual therapy. Mechanically, this implies developing robots with significant workspace and force feedback features. Such systems have, in turn, the drawback of being bulky and heavy, degrading the final interaction experience with the patient. Among the different technical challenges of these systems, gravity compensation is key for high rehabilitation performance.

In previous work [3], we developed the HomeRehab robotic system, which is capable of restoring haptic effects at its handle for upper limb rehabilitation (Figure 1). This mechanism has three degrees of freedom (3 DoF) and consists of a pantograph that pivots on a horizontal axis, but it is not perfectly gravity-balanced. Thus, an active compensation strategy is needed. In this article, *gravity compensation* means removing the gravity components of the mechanical device to make it transparent or imperceptible to the patient, in the sense that, during the rehabilitation exercises, the patient has to overcome only the weight of his or her own arm.

Related Work

Gravity compensation is a basic requirement in robotics, and more specifically in haptics, to ensure system usability and transparency. It is also a key specification of mechatronic rehabilitation devices [4], so that users can move their arms freely without feeling (and holding) the weight of the robot during the therapy. In addition, friction may be a limiting factor of these mechanisms, and some strategies try to compensate for both the unknown static friction and the gravity forces [5]. A comparison of different algorithms for gravity compensation in parallel mechanisms can be found in [6].

In the field of rehabilitation devices, several gravity compensation strategies have been applied that are also valid for any mechatronic system. In some cases, the mechanism is designed to be gravity balanced, that is, to be in neutral equilibrium without requiring joint actuator torques [7]. This feature can also be achieved by adding a passive mechanism to the robotic arm [8]. These solutions are intrinsically safe, but they are difficult to design.

Analytical Approaches

Analytical active compensation methods balance the system, acting on each joint according to a gravity model of the

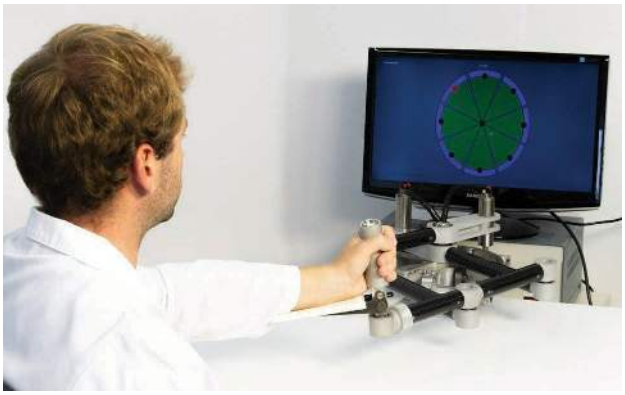


Figure 1. The HomeRehab robotic rehabilitation system developed at Ceit, a member of the Basque Research and Technology Alliance (BRTA) and associated with Tecnun, Universidad de Navarra, School of Engineering at San Sebastián.

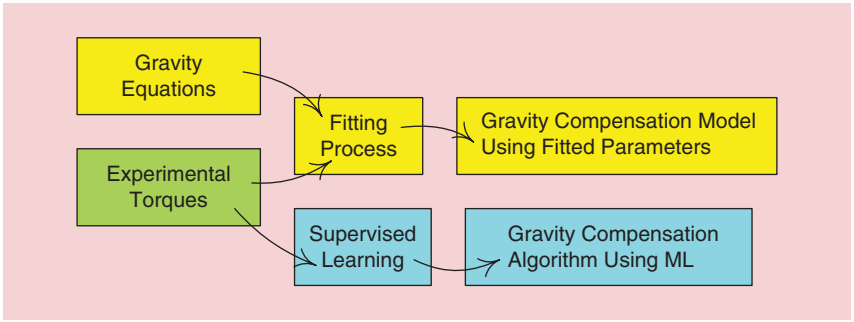


Figure 2. The gravity compensation strategies.

mechanism. In this model, the torques depend on the pose and weight distribution of the links. The positions of the centers of mass and the weights of the links may be estimated from the CAD model. However, these parameters must be experimentally adjusted to take into account manufacturing uncertainties and unmodeled components.

In the case of the rehabilitation robot presented in [9], the authors focused on offsetting the gravity of the motors. The mass of the rest of the mechanism is neglected compared to the mass of the motors. This assumption is due to the fact that the designed rehabilitation robot for the forearm and wrist is relatively small. Instead of using the CAD parameters or the known mass of the motors, some authors prefer to measure the torques in a set of positions to estimate the parameters of the gravity model without the need to identify the masses [10]. Note that this approach also uses the system's static equilibrium equations.

Other methods do not use any system equation. For example, in [4], the authors define a working area where the compensation for the weight of the exoskeleton is to be applied, and this volume is discretized into small cubes. In each cube, the force is calculated at each vertex so that the gravity is compensated for and the robot is immobile. Once the force database is completed (and knowing in which cube the exoskeleton is located during the rehabilitation exercise), a weighted mean of the eight vertices of the cube gives the gravity compensation force value.

ML-Based Approaches

ML is a set of algorithms based on two main ideas: the acquisition of new knowledge from external sources and the improvement of knowledge representations and structures so that existing knowledge may be better exploited [11]. In ML, there are many possible techniques and approaches to achieve the same goal; some of them are more appropriate than others for a specific problem. Among the existing approaches are neural networks, Bayesian classifiers, nearest neighbor classifiers, support vector machines, and decision trees.

ML algorithms use computational methods to get information directly from data without relying on a predetermined model. Thus, once a gravity force database is available, ML-based techniques can compute an estimation of the gravity components. In fact, ML has already been used to solve some mechatronic problems, such as the design of smart laser-welding controllers [12] and adaptive exoskeleton controllers for optimal rehabilitation [13].

Materials and Methods

Based on the strategies found in the literature, this article analyzes and compares two different solutions to achieve gravity compensation (Figure 2) and tests them on the HomeRehab system. The first one, the analytical method, uses the device's gravity equations with experimentally fitted parameters. The second solution implements a novel strategy based on ML to compensate for gravity without using the device's static equations. The HomeRehab robotic system (Figure 3) is used to train and test the proposed methods. Table 1 shows its main technical specifications.

HomeRehab has the option to work in both the 2D and 3D workspaces. When working in 2D, the patient exercising with HomeRehab sits in a chair and trains with 2D movements in a planar workspace. Once the patient is able to control and hold the weight of its arm, our device lets him or her exercise on virtual daily life activities in a 3D workspace by standing up and holding the end effector as a traditional haptic device. In this case, the proposed active gravity compensation methods aim to overcome the weight of the device so that the exercises are more realistic and less tiring.

To derive the static-equilibrium equations of HomeRehab, it is assumed that the links' centers of gravity are placed along their axis, but not necessarily at their geometric centers (unknown distances l_b, l_c, l_d, and l_e), and also at different heights with respect to the plane of rotation of the pantograph (l'_b, l'_c, l'_d, and l'_e). The lengths of the mechanism are $l_1 = 0.2$ m, $l_3 = 0.3$ m, and $l_3 = 0.4$ m. Angle θ_1 is the rotation of the mechanism with the horizontal plane, while angles θ_2 and θ_3 define the position of the pantograph links. The handle is modeled as a punctual mass (point A), and the gravity centers of the driving links (points D and E) are located closer to the ends where the motors are anchored. Note that the pulley of the mechanism rotates in solidarity with the pantograph. Its center of gravity (point F) is not over axis x, and it is also outside the pivoting plane of the pantograph.

Using the scheme in Figure 3 and operating the static-equilibrium equations, the gravity components of the device are

$$\begin{aligned}\tau_1 &= -p_1 s_1 + p_2 c_1 c_2 + p_3 c_1 s_3 - p_4 c_1 + p_5 c_1 c_3,\\ \tau_2 &= -p_2 s_1 s_2,\\ \tau_3 &= p_3 s_1 c_3 - p_5 s_1 s_3,\end{aligned} \tag{1}$$

where $s_i = \sin\theta_i$, and $c_i = \cos\theta_i$. These components depend on five parameters, p_1, p_2, p_3, p_4, and p_5, which, in turn,

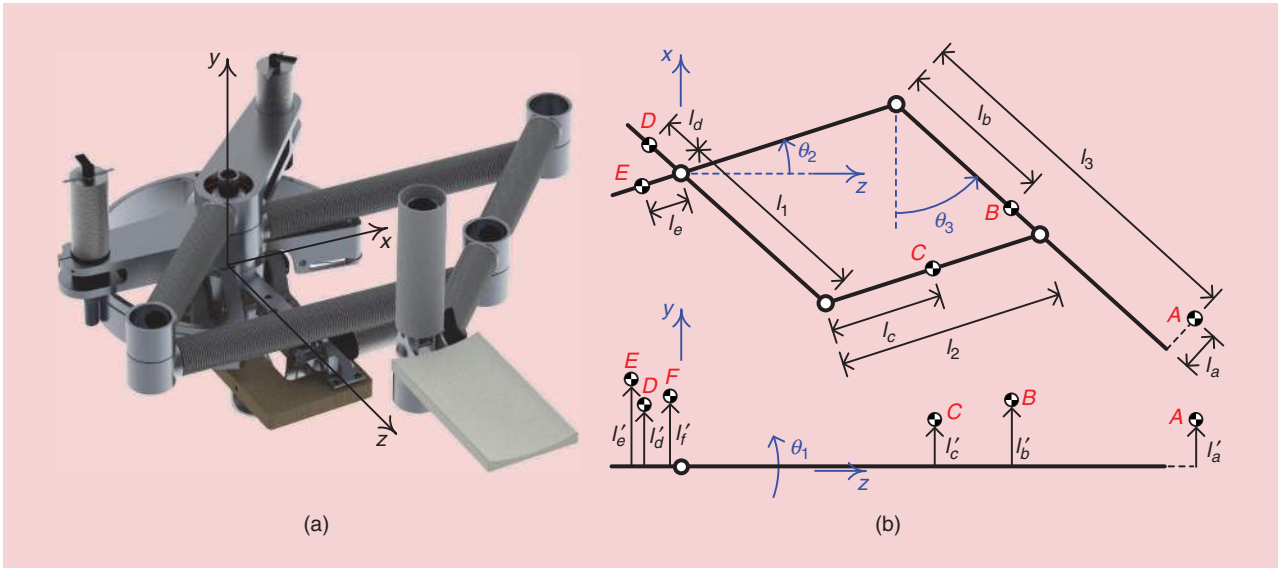


Figure 3. The (a) pantograph scheme and (b) positions of the centers of gravity.

depend on the masses and positions of the centers of gravity of the links:

$$
\begin{aligned}
p_1 &= (m_a l'_a + m_b l'_b + m_c l'_c + m_d l'_d + m_e l'_e + m_f l'_f)g,\\
p_2 &= (m_a l_2 + m_b l_2 + m_c l_c - m_e l_e)g,\\
p_3 &= (m_a l_3 + m_b l_b + m_c l_1 - m_d l_d)g,\\
p_4 &= m_f l_f g,\\
p_5 &= m_a l_a g.
\end{aligned} \tag{2}
$$

A rough estimation of the five parameters p_i could be derived using a CAD model of the mechanism. However, to obtain reliable values for p_i, an experimental fit is required. ML-based methods do not require the resolution of analytical equations. While HomeRehab is a mechanism whose equations may be derived with relative ease, for some parallel mechanisms and commercial devices whose CAD models and geometrical data are not available, the process to develop the equations may be complex.

Among the different ML techniques and approaches, in this article, a decision tree technique is used, since it enables fast responses and accurate results, as many researchers have already tested [14], [15]. A decision tree is an algorithm and data structure oriented for supervised learning, where each node represents an attribute or feature (in our case, 3D coordinates). For each node, the children are classified according to a criterion until obtaining a leaf node. These leaves represent the final decision [16].

Usually, several decision trees are used because more accurate results are obtained (each tree may give a different solution, and a vote scheme is performed to determine the final decision). The algorithm that achieves this process is called *random forest* [17]. This algorithm can be used to classify or perform a regression prediction where each tree in the ensemble is trained on a subset of the entire training data set. Then, each split is performed on a random subset of features (one for each tree) [16].

Table 1. The HomeRehab specifications.

Workspace	800 × 400 × 400 mm
Maximum force	14 N (continuous), 28 N (peak)
Actuators	Maxon DCX32L-GPX32, 128 μN·m, 4:1
Transmission	20:1 cable transmission
Encoders	Maxon ENX16, 1,024 pulses/revolution
Position resolution	6 μm
Weight	8.2 kg

The extra trees is an extension of the random forest regression model proposed by Geurts [18]. Random forest and extra trees are important algorithms within this class and have reported state-of-the-art performance on many regression tasks with high-dimensional inputs and outputs [19]. The differences between extra trees and the original random forest are as follows:

1) Unlike random forest, extra trees does not use the tree-bagging step to generate the training subset for each tree.
2) Extra trees randomly selects the best feature along with the corresponding value to split the node [20].

These two differences result in extra trees being less susceptible to overfitting and able to report better performance [18].

It is important to note that the extra trees regression is not a classification tree. In a regression tree, as the target variable does not have classes, we fit a regression model to the target variable using each of the independent variables. Then, for each independent variable, the data are split at several points. At each split point, the error between the predicted value and the actual values is squared to get a sum of squared errors

(SSE). The split-point errors across the variables are compared, and the variable/point yielding the lowest SSE is chosen as the root node/split point. This process is recursively continued.

Both methods, analytical and ML-based, need experimental data to fit the model or train the algorithm. The inputs to the methods are the angles measured at the three active joints of the device in any Cartesian position of its workspace, while the outputs are the three torques that must be applied to each joint to hold the device still in each Cartesian position. For other devices, the number of inputs and outputs should be equal to the number of active DoF of the device. To generate such a force and position database, the workspace of the device is divided into a finite number of points, and an experiment is designed to compute the torque values necessary to hold the system still at each point.

Experiments

This section first describes how the force database is collected and the procedure to fit the analytical equations for gravity compensation and to train the ML-based method. A final experiment is carried out on a set of points of HomeRehab's workspace, different from the points used for the training, to compare the performance of each method.

Experimental Setup

The HomeRehab workspace is divided into 4,920 points covering 79.4% of the workspace used for rehabilitation applications. The tested positions (Figure 4) form a rectangular parallelepiped grid in the Cartesian axes, 0.8 m × 0.22 m × 0.18 m. The distance between points is 2 cm.

A proportional–integral–derivative (PID) controller forces the system to move automatically from one position to another. When the mechanism reaches the steady state at each tested position (with a position error lower than 0.1 mm), the torques provided by the controller are precisely the torques that compensate for the gravity. These torques τ_1, τ_2, and τ_3 are recorded in a database together with the angles θ_1, θ_2, and θ_3 measured at each active joint of the device. The generation of this database takes 2.5 h.

Analytical Model

A least-squares optimization method is carried out to obtain the values of p_i that best fit model (1) with the training database. The following values are obtained:

$$\begin{aligned} p_1 &= 3.33\ \text{N} \cdot \text{m}, \\ p_2 &= 3.97\ \text{N} \cdot \text{m}, \\ p_3 &= 3.98\ \text{N} \cdot \text{m}, \\ p_4 &= 2.05\ \text{N} \cdot \text{m}, \\ p_5 &= 0.77\ \text{N} \cdot \text{m}. \end{aligned} \tag{3}$$

The experimental torques and those that are obtained from the gravity model (1) with the proposed parameters (3) are shown in Figure 5 for 2,000 consecutive points of the training database. It is worth noting that the experimentally fitted gravity model is robust to some assumptions of the scheme

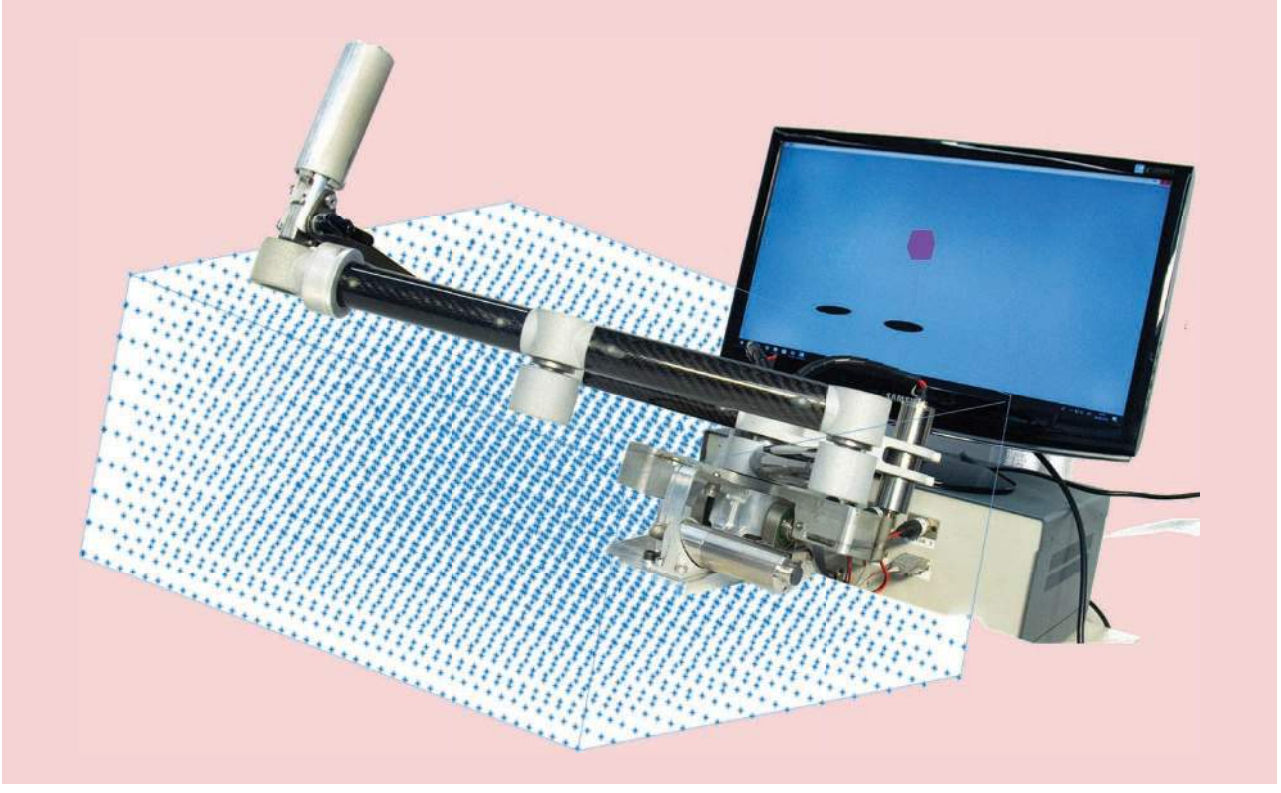


Figure 4. The tested positions.

depicted in Figure 3 because the estimated parameters p_i contain the contribution of several distributed masses. It is not especially relevant for the validity of the model to find the exact value of each length and mass of the model.

Taking into account that the dc motors of the mechanism can exert up to 10.24 N·m after the transmission, a nonnegligible amount of torque is used to compensate for the gravity of the device. In the case of torque τ_1, which is the worst case of the three motors (Figure 5), the mean value within all the positions of the workspace is 2.27 N·m, which represents 22% of the maximum continuous torque. The maximum gravity torque τ_1 is 3.91 N·m, 38% of the available torque.

ML-Based Method Development

The input data for the ML algorithm is the same as for the analytical method (4,920 training points). These data are arranged in six columns (x, y, z, τ_1, τ_2, and τ_3). No editing, cleaning, or any other technique was used on the data set. The ML method development consisted of evaluating several scenarios for the extra trees algorithm, changing the percentage of the data for training and testing. The ML algorithm is implemented in Python 3.6 using the Anaconda 5.0.1 64-b environment and scikit-learn 0.19 library. Table 2 shows the results of applying the extra trees algorithm to different scenarios.

The tests were performed using a fixed seed for NumPy 12345 to be reproducible. The accuracy was evaluated using other, different seeds, but there was no significant difference in the results. The base function used was *ExtraTreesRegressor* with 50 estimators. It is relevant to consider that increasing this parameter adds complexity to the calculation by adding trees to the forest. We tried to increase it, but found no significant improvement. However, we noticed a high penalty to the point calculation. The maximum features are three, since the data set has only three inputs. Finally, the random state is set to zero.

Table 2 shows that, by using 80% of the data, the best results are obtained. Other approaches, such as the random forest regressor, decision tree regressor, and others, were also tested with those data sets to check our initial hypothesis of using the extra trees regression method. The results were not improved in terms of accuracy (the coefficient of determination R^2 in Table 3). However, the MultiO/P GBR method achieves good performance, with a lower prediction time and memory usage. Thus, if real-time implementation specifications are very relevant (less than 1 ms), the final method would be the preferred choice.

Validation

A validation experiment is carried out with HomeRehab to test the performance of both methods. Analytical gravity

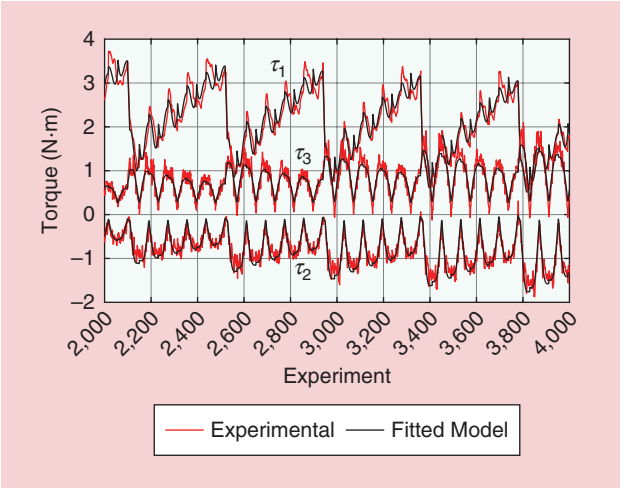


Figure 5. The experimental and theoretical torques of the fitted model.

Table 2. The results for different scenarios of the extra trees algorithm.

Scenario	1	2	3	4
Training	80%	40%	20%	5%
Testing	20%	60%	80%	95%
R^2	0.958	0.943	0.931	0.899

Table 3. The results for different ML algorithms.

Name	Mean Square Error	R^2	Training Time (Mean)	Prediction Time (Mean)	Memory (MB)
Extra trees	0.0156	0.9489	1.05 s	2.64 ms	278.61
k-nearest neighbors	0.0213	0.928	1.87 ms	353 μs	278.81
Linear regression	0.0655	0.7516	465 μs	45.3 μs	279.15
Ridge regression	0.0655	0.7509	1.37 ms	43.9 μs	279.57
Lasso	0.3305	—	598 μs	50.6 μs	279.61
Random forest regression	0.0176	0.9419	667 ms	5.32 ms	279.61
Decision trees	0.0363	0.8879	9.85 ms	50.1 μs	271.8
MultiO/P GBR	0.0199	0.9342	1.39 s	747 μs	271.92

MultiO/P GBR: multioutput gradient-boosting regressor.

compensation equations are directly programmed in HomeRehab's National Instruments (NI) MyRIO controller, which runs at a sampling time of 1 ms. A local host Python server is created as middleware between the PC that runs the ML algorithm and the HomeRehab controller. The input message for the server is the actual position of the end effector (three angles), and the output consists of the torques needed in the three motors to compensate for the gravity force of the mechatronic device. Communication between the computer and the NI MyRIO controller is achieved by the User Datagram Protocol (UDP), sending end-effector positions from the controller to the computer and receiving compensation torques computed by the ML algorithm. The average computation time for ML prediction and UDP communication is approximately 2–3 ms.

The validation experiment consists of moving the HomeRehab system to 120 points (84 points inside the limits of the training cube and 36 points outside the training cube). None of these points was previously used for the training phase. These 120 points result from the combination of $x = [-0.53, -0.43, -0.33, -0.23, -0.13, -0.03, 0.03, 0.13, 0.23, 0.43]$, $y = [0.07, 0.11, 0.15]$ and $z = [0.29, 0.33, 0.37, 0.41]$ (m). Thirty-six of the 120 points are outside the training cube (they do not satisfy $-0.4 < x < 0.4$ m).

The validation test is carried out as follows: First, a PID controller holds the device still at the selected point with a position error lower than 0.1 mm. The torque values computed by the PID to hold the device still at each one of the points are considered the ground-truth data for validation. Once the system reaches each point, we replace the torques computed by the PID controller with the torques derived from the proposed two methods, and we evaluate whether the device remains immobile in the same position or if it moves (either falling or moving in other directions). The outcomes of this experiment show that both methods behave properly, holding the device at all 84 points that lie within the training workspace. However, the ML-based method fails to hold the device still at 25 points outside the training workspace, while the analytical method succeeds at all 36 points outside the training workspace.

Figure 6 shows an example of the torque values computed by both methods along 10 validation points: $x = [-0.53, -0.43, -0.33, -0.23, -0.13, -0.03, 0.03, 0.13, 0.23, 0.43]$, $y = 0.11$, and $z = 0.37$ (m). It can be seen from Figure 6 that the torque values for both methods, compared with the PID torque values, are slightly different at each point. However, HomeRehab remains still at most of them, which is considered a good behavior. This fact is due to the friction components of the mechanism that need to be overcome to start moving. In this particular set the analytical equations are able to hold the system still at all 10 points. However, the ML-based method fails at points 1 and 10, which are outside the limits of the training cube (point 2 is also outside the limits, but the method performs well). At these two points, the robot does not hold still with the ML-based method.

Figure 7 illustrates the box plots of the error at all 120 points between the PID values and the values given by both methods. For each joint and method, two sets of box plot figures are shown, one for the 84 points inside the limits of the

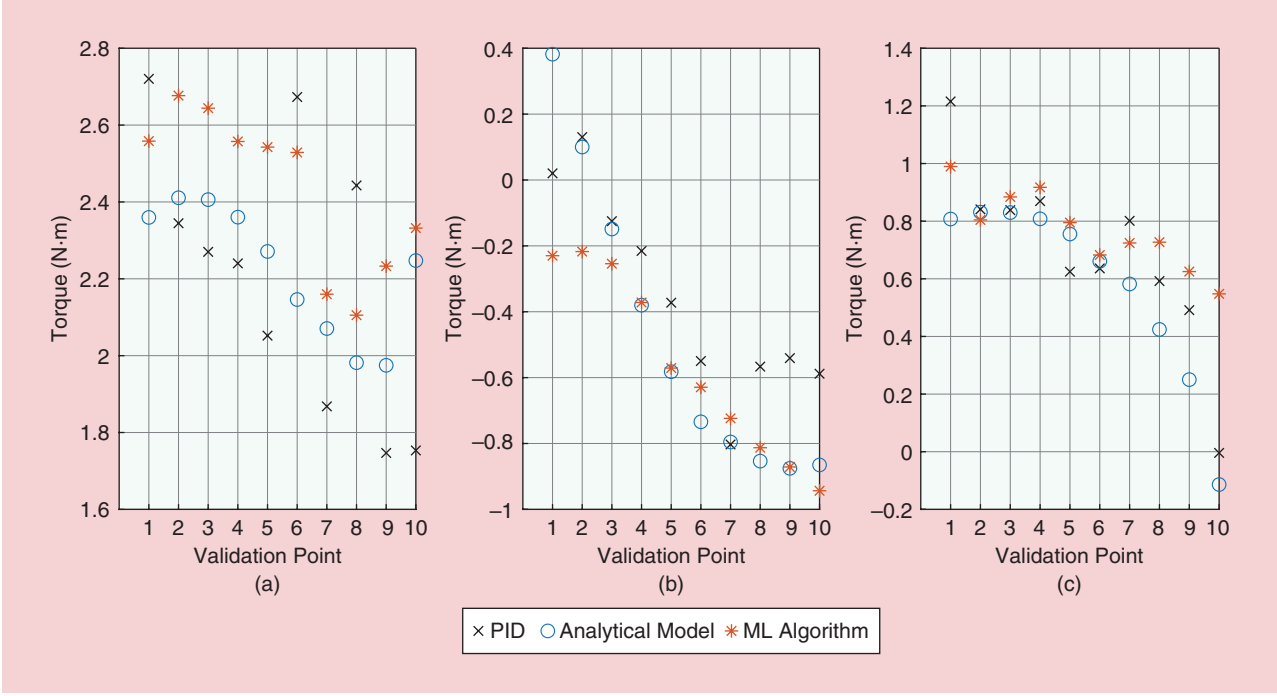


Figure 6. The torque values computed by the PID controller and both methods (analytical model and ML algorithm) in a set of 10 validation points. (a) Torque τ_1. (b) Torque τ_2. (c) Torque τ_3.

training cube (in black) and the other for the remaining points outside the limits (in blue). It can be seen that, while the ML-method computes similar values to the analytical method inside the training cube (except for joint 1), outside the cube the joint 2 and 3 torque values are different.

Discussion

From the outcomes of the validation experiment, it is difficult to determine which of the two methods provides better gravity compensation torque values. There is a range of torques where the results can be considered valid; that is, the system holds still. This range of torques directly depends on the friction of the system. Even the PID torque values, used as ground truth, are affected by the friction. Nevertheless, the qualitative result (whether the system holds still or not) is a valid outcome when the aim is to compensate for the device's gravity forces.

Training data are obtained relatively fast (2.5 h) for a medium-size workspace used in upper-limb rehabilitation. The analytical method seems to behave better, as it can give proper results even outside the workspace covered by the training data. However, deriving the gravity-equilibrium equations may not always be possible or easy (e.g., for some parallel mechanisms). The ML-based method enables the implementation of an algorithm that overcomes this drawback, as it does not require the analytical equations of equilibrium, but it does not perform very well outside the workspace considered in the training data.

The drawback of the ML method may not be relevant if the training data cover the entire workspace of the mechanism. However, this circumstance should be considered for large and complex workspaces. The fact that the ML method performs poorly outside the area of the training data sounds like overfitting. We performed several experiments with different parameters (estimators, seeds, the train/test split, and so forth) to further analyze the issue, and we discovered that it was not overfitting but a lack of precision of the method. Therefore, we think that using deep-learning methods may lead to better results. The goal of this article was to focus specifically on traditional ML methods, but future work will evaluate the performance of deep-learning methods as a solution to the poor performance of the ML method outside the workspace considered with the training data. A video showing the behavior of the system using the compensation methods is available as a supplement to this article. The ML-based method code and training data set are also available to the public.

Conclusions

Gravity compensation is a mandatory feature of mechatronic devices used for rehabilitation. People with limited mobility cannot manipulate bulky apparatus and should be able to perform rehabilitation tasks without extra impediments, as if they were moving their limbs freely.

In this article, we describe the use of ML methods to ease the complex task of developing proper active gravity compensation control algorithms for robotic rehabilitation. Gravity compensation is of paramount importance to achieve transparent haptic interactions, especially for medium- and large-size mechatronic devices where the inertia of the system is not negligible in free motion. Traditional control methods for active gravity compensation require deriving the

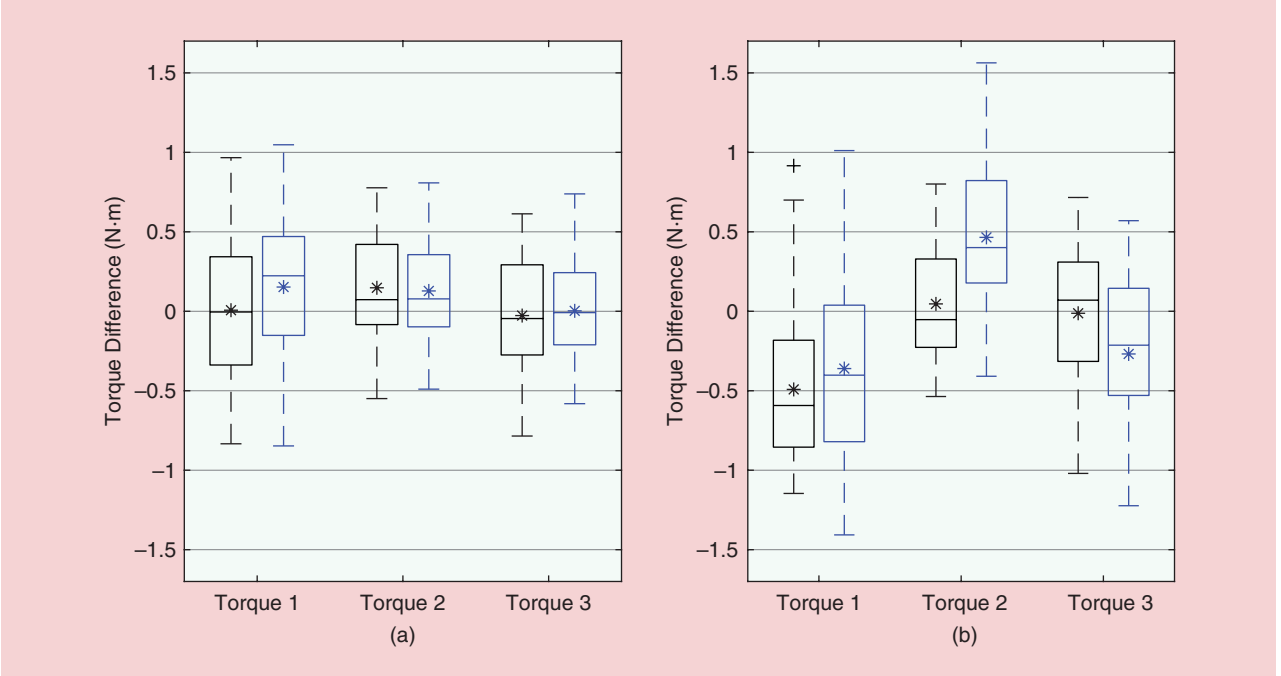


Figure 7. The box plots of the torque difference between (a) the PID analytical-model values and (b) the values given by the PID-ML methods (points within the training cube are in black, and points outside the training cube are in blue), with mean values (asterisks).

gravity-equilibrium equations of the system, which may not be straightforward for complex kinematic configurations.

This work proposes a gravity compensation strategy based on ML methods as an easy and fast approach to the problem. It describes the strategy's implementation and compares the method with the traditional approach, which is also described in detail. In general, the results show that the traditional analytical solution performs better than the ML approach. However, in situations where it is difficult to derive an analytical solution and obtaining training data for ML approaches is considerably easier, the results show that ML models offer a promising alternative with certain limitations regarding workspace coverage. The results can also be extended to other robotic and haptic domains. Furthermore, we believe that ML can be applied to other rehabilitation tasks, such as monitoring patients' progress by gathering and analyzing multiple robot sensor measures during exercise and personalizing the assistance control algorithms for each individual patient and exercise.

References

[1] A. M. Okamura, M. J. Mataric, and H. I. Christensen, "Medical and health-care robotics," *IEEE Robot. Autom. Mag.*, vol. 17, no. 3, pp. 26–37, Sept. 2010. doi: 10.1109/MRA.2010.937861.

[2] J. C. Pulido et al., "A socially assistive robotic platform for upper-limb rehabilitation: A longitudinal study with pediatric patients," *IEEE Robot. Autom. Mag.*, vol. 26, no. 2, pp. 24–39, June 2019. doi: 10.1109/MRA.2019.2905231.

[3] I. Díaz et al., "Development of a robotic device for post-stroke home tele-rehabilitation," *Adv. Mech. Eng.*, vol. 10, no. 1, pp. 1–8, 2018. doi: 10.1177/1687814017752302.

[4] R. Ott, M. Gutierrez, D. Thalmann, and F. Vexo, "Improving user comfort in haptic virtual environments through gravity compensation," in *Proc. World Haptics Conf.*, 2005, pp. 401–409. doi: 10.1109/WHC.2005.78.

[5] M. Liu and N. H. Quach, "Estimation and compensation of gravity and friction forces for robot arms: Theory and experiments," *J. Intell. Robot. Syst.*, vol. 31, no. 4, pp. 339–354, 2001. doi: 10.1023/A:1012089607929.

[6] D. Checcacci, E. Sotgiu, A. Frisoli, C. A. Avizzano, and M. Bergamasco, "Gravity compensation algorithms for parallel haptic interface," in *Proc. 11th IEEE Int. Workshop on Robot and Human Interactive Communication*, 2002, pp. 140–145. doi: 10.1109/ROMAN.2002.1045612.

[7] G. Spagnuolo, M. Malosio, A. Scano, M. Caimmi, G. Legnani, and L. M. Tosatti, "Passive and active gravity-compensation of lightarm, an exoskeleton for the upper-limb rehabilitation," in *Proc. IEEE Int. Conf. Rehabilitation Robotics*, 2015, pp. 440–445. doi: 10.1109/ICORR.2015.7281239.

[8] D. G. Chung, M. Hwang, J. Won, and D.-S. Kwon, "Gravity compensation mechanism for roll-pitch rotation of a robotic arm," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2016, pp. 338–343. doi: 10.1109/IROS.2016.7759076.

[9] L. Luo, L. Peng, Z. Hou, and W. Wang, "Design and control of a 3-dof rehabilitation robot for forearm and wrist," in *Proc. 39th Annu. Int. Conf. IEEE Engineering Medicine and Biology Society*, 2017, pp. 4127–4130. doi: 10.1109/EMBC.2017.8037764.

[10] S. Moubarak, M. T. Pham, R. Moreau, and T. Redarce, "Gravity compensation of an upper extremity exoskeleton," in *Proc. Annu. Int. Conf. IEEE Engineering Medicine and Biology Society*, 2010, pp. 4489–4493. doi: 10.1109/IEMBS.2010.5626036.

[11] G. Briscoe and T. Caelli, *A Compendium of Machine Learning*, vol. I. Norwood, NJ: Ablex, 1996.

[12] J. Günther, P. M. Pilarski, G. Helfrich, H. Shen, and K. Diepold, "Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning," *Mechatronics*, vol. 34, pp. 1–11, Mar. 2016. doi: 10.1016/j.mechatronics.2015.09.004.

[13] N. Ajjanaromvat and M. Parnichkun, "Trajectory tracking using online learning LQR with adaptive learning control of a leg-exoskeleton for disorder gait rehabilitation," *Mechatronics*, vol. 51, pp. 85–96, May 2018. doi: 10.1016/j.mechatronics.2018.03.003.

[14] M. N. Adnan and M. Z. Islam, "A comprehensive method for attribute space extension for random forest," in *Proc. 17th Int. Conf. Computer and Information Technology*, 2014, pp. 25–29. doi: 10.1109/ICCITechn.2014.7073129.

[15] L. Zhang and P. N. Suganthan, "Random forests with ensemble of feature spaces," *Pattern Recognit.*, vol. 47, no. 10, pp. 3429–3437, 2014. doi: 10.1016/j.patcog.2014.04.001.

[16] A. Heemann, D. Velinova, and M. Gastegger, "Text Classification—A Contest," (in German), Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät, Institut für Informatik, summer term 2017.

[17] M. N. Adnan and M. Z. Islam, "Effects of dynamic subspacing in random forest," in *Proc. Int. Conf. Advanced Data Mining and Applications*, 2017, pp. 303–312. doi: 10.1007/978-3-319-69179-4_21.

[18] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006. doi: 10.1007/s10994-006-6226-1.

[19] J. Gall, A. Yao, N. Ravazi, L. Van Gool, and V. Lempitsky, "Hough forests for object detection, tracking, and action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, pp. 2188–2202, 2011. doi: 10.1109/TPAMI.2011.70.

[20] V. John, Z. Liu, C. Guo, S. Mita, and K. Kidono, "Real-time lane estimation using deep features and extra trees regression," in *Proc. Pacific Rim Symp. Image and Video Technology*, 2015, pp. 721–733. doi: 10.1007/978-3-319-29451-3_57.

Axier Ugartemendia, Ceit and Tecnun, University of Navarra, Donostia–San Sebastián, Spain. Email: augartemendia@ceit.es.

Daniel Rosquete, Ceit and Tecnun, University of Navarra, Donostia–San Sebastián, Spain. Email: drosquete@ceit.es.

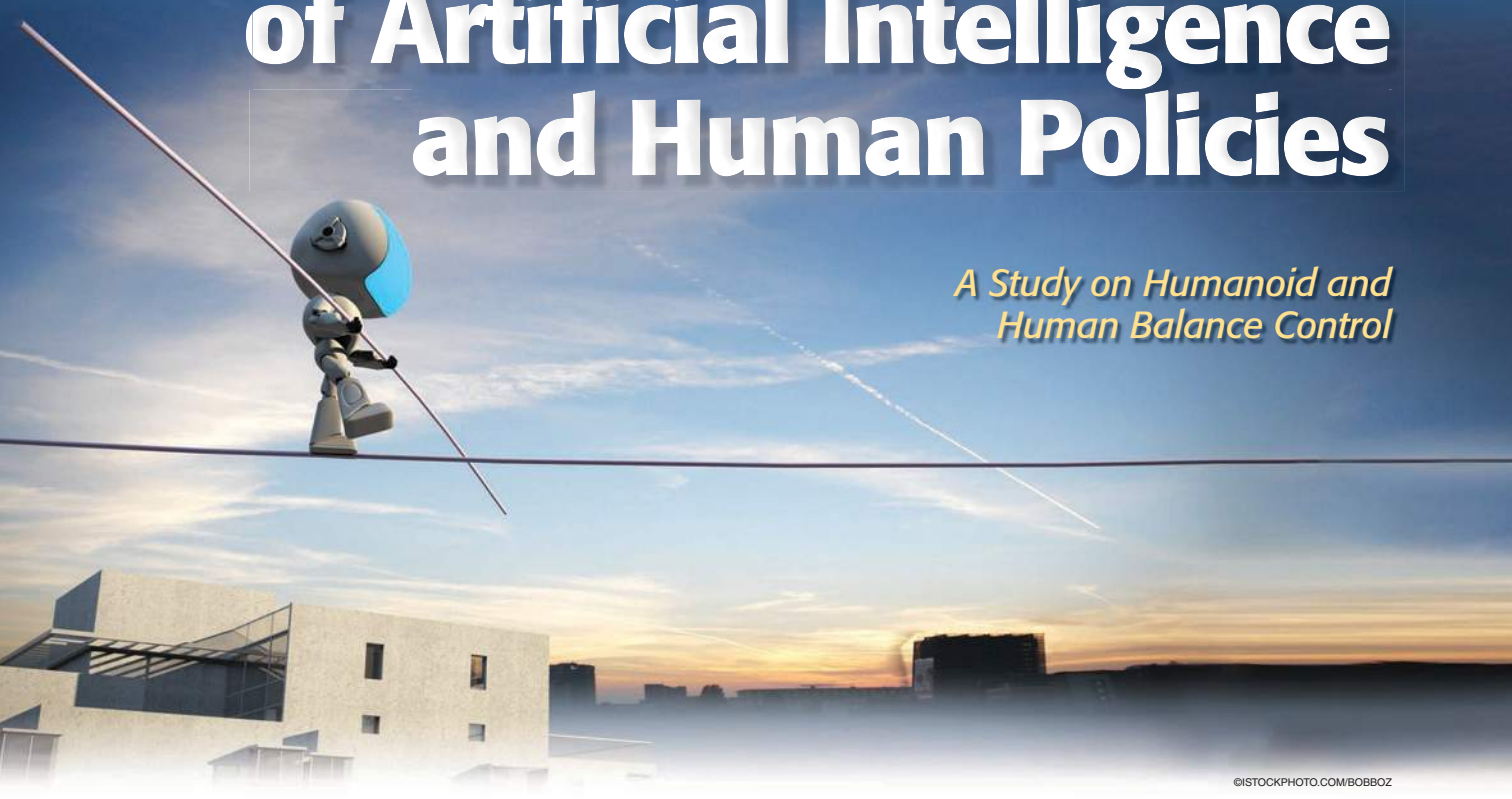
Jorge Juan Gil, Ceit and Tecnun, University of Navarra, Donostia–San Sebastián, Spain. Email: jjgil@tecnun.es.

Iñaki Díaz, Ceit and Tecnun, University of Navarra, Donostia–San Sebastián, Spain. Email: idiaz@ceit.es.

Diego Borro, Ceit and Tecnun, University of Navarra, Donostia–San Sebastián, Spain. Email: dborro@ceit.es.

Decoding Motor Skills of Artificial Intelligence and Human Policies

A Study on Humanoid and Human Balance Control



©ISTOCKPHOTO.COM/BOBBOZ

By Kai Yuan, Christopher McGreavy, Chuanyu Yang, Wouter Wolfslag, and Zhibin Li

From the advancement in computers, computer-aided design has emerged for mechanical and electronic engineering, architecture, and many other engineering fields. Foreseeing a similar development curve and technology wave, we forecast a new emerging discipline in the near future that uses learning-aided approaches to catalyze control development, alongside other similar applications such as medicine discovery. In this article, we propose a new paradigm for using machine learning to facilitate quicker, more efficient, and more effective control development, as an alternative way of leveraging the power of machine learning in addition to other options that intend to use learning directly in real-world applications.

Machine learning and deep reinforcement learning (DRL), in particular, have reached an advanced stage and can now produce powerful policies with better autonomous performance than many state-of-the-art control and planning approaches in robot locomotion [1], robotic manipulation [2], and even the control of complex morphological machines [3]. DRL's ability to solve complex problems in a relatively short development time is especially attractive. This is empowered by training the policies that maximize the cumulative reward through the exploration of the action and state space rather than using prior knowledge of the models about the robot, the world, and their interactions.

To leverage the capabilities of DRL, we first develop a DRL-based control framework to learn the rich motor skills of push recovery for humanoid robots. The complexity in whole-body balancing arises in challenges such as multicontact coordination based on multisensory inputs, state transitions between fully and underactuated situations, switching policies, and generalizing to external disturbances on any body parts and accounts for all the edge cases that a designer has difficulty considering beforehand. In such a setting, manually designing individual control strategies and finding a reliable switching mechanism require substantial development time, mathematical rigor, and code implementation. On the other hand, through a well-designed DRL framework and task-specific training procedures, a robust policy can be learned automatically by interacting with the environment,

Digital Object Identifier 10.1109/MRA.2020.2980547
Date of current version: 10 April 2020

requiring only computational power. In particular, as shown in Figure 1, our AI policy exhibits human-like behaviors with four typical push recovery modes emerging naturally: ankle, hip, toe, and stepping strategies.

Although the learned control policy could possibly be deployed on a real robotic system, the lack of explainability and analytical reasoning of the neural network (NN) makes such a policy unsuitable for safety-critical applications in the real world. Furthermore, due to the demand of large data and the sample-inefficient nature of DRL algorithms, complex policies are typically trained in simulation, which cannot guarantee the same performance when transferred directly to the real system [1], and the challenge of the reality gap raises concerns about safety and performance.

To benefit from both the safety and interpretability of a control policy and the versatility and adaptability gained from learning, we propose to take advantage of DRL to quickly discover versatile, deployable policies and solutions for very difficult problems and then study, analyze, and extract the principles of those policies as guidelines for developing engineered controllers in a reliable manner. By doing so, we utilize artificial intelligence (AI) solutions for rapid control development (Figure 2) to design safe and certifiable controllers, which can be verified and deployed on real-world robots (Figure 3).

Although classical control development is based on gradually building knowledge that increases performance incrementally, using a template policy provides disruptive, innovative solutions that escalate performance (green line, Figure 2). DRL is able to achieve good performance by a number of iterations in the DRL learning framework. However, the achieved performance is still comparatively low compared to what tuning in control can do. Combining both approaches to kick-start the iteration process helps in designing good controllers. After determining the system and controller, it is a straightforward approach to improve upon these because we are able to understand why the performance is lower than optimum; in the case of DRL, on the other hand, there is little influence from human engineers to improve performance, aside from reshaping the reward and/or altering the learning framework and relying on the exploration being sufficiently large to achieve high performance.

In this article, we study a viable approach to infer the underlying principles of an AI policy by studying its perception–action relation, i.e., to some extent, reverse engineering an equivalent controller in terms of functionality based on a black box policy. This methodology is applicable not only to AI policies but also to any black box policies, such as a human policy. Without knowing exactly how push recovery policies are realized by artificial NNs or biological human NNs, we can still analyze their behavior at the functionality level by studying their input–output relationship.

Based on evidence of optimality in human manipulation tasks [4], we hypothesize that policies for push recovery in humans and humanoids are optimal control processes that follow certain optimal criteria, which can be quantified. Following this hypothesis, we analyze and utilize input–output data collected from both humanoid and human policies and propose a minimum-jerk model-predictive control (MJMPC) framework able to quantitatively reflect both AI and human push recovery policies. The engineered controller has high similarity (a coefficient of determination more than 90%) with the collected data and also exhibits the same human-like push

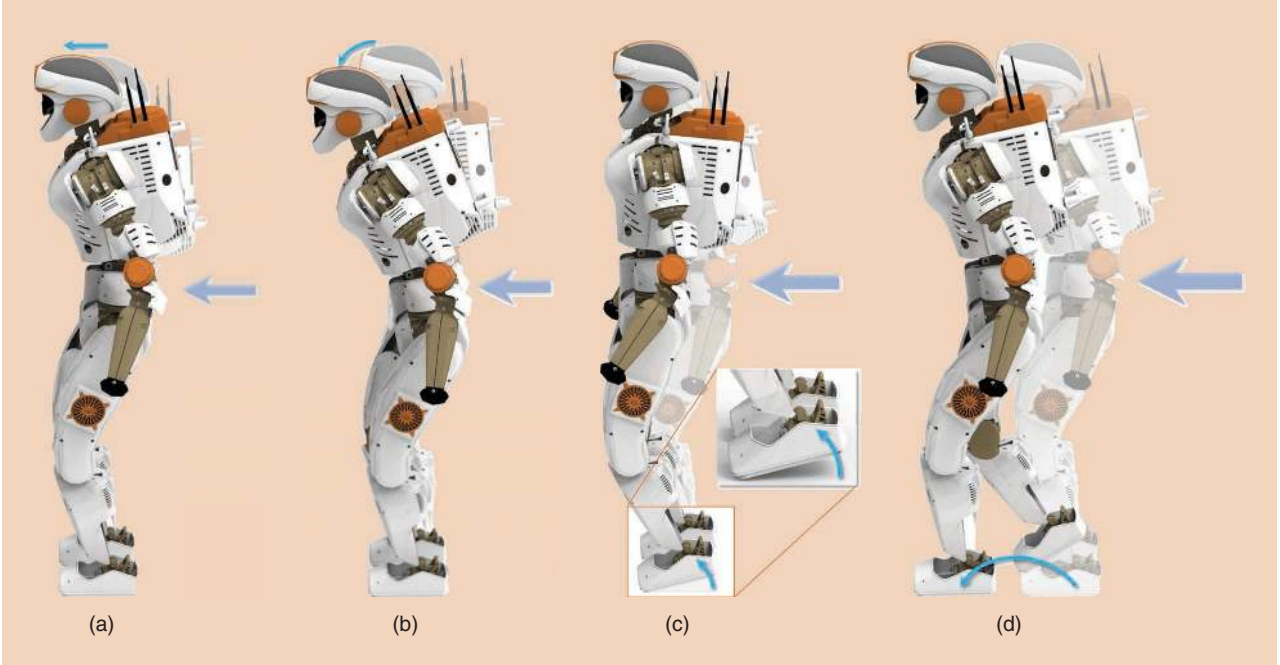


Figure 1. The human-like push recovery strategies emerging from DRL. The discovered behaviors serve as a guideline for the design of certifiable and safe controllers that replicate advantageous strategies from artificial intelligence (AI) policies. The (a) ankle, (b) hip, (c) toe, and (d) stepping strategies.

recovery strategies, which emerge from the proposed MJMPC without the need for manual switching between strategies.

Furthermore, a comparison between humanoid and human balancing is conducted to show the characteristics of the learned humanoid behavior. This comparison shows that DRL algorithms are very powerful when used to learn a policy (e.g., balancing) within a short development and training time that may require humans years to learn. In contrast, to design an engineered controller from scratch having similar performance, months or even years are needed for developmental iterations, mainly because of the high redundancy and diversity of control actions, which are yet challenging to resolve in terms of physical optimality on a high degree-of-freedom (DoF) robot. In this regard, the learning approach is very attractive because of the significant reduction of manual effort, and the learning architecture requires only the design of input–output and rewards. This article sheds some light on a new paradigm: the recent high-profile successes in DRL suggest new high-quality value in learning methods for which discovered policies can be used as a basis to speed up development of robotic controllers (Figure 3). As an outcome of this push recovery

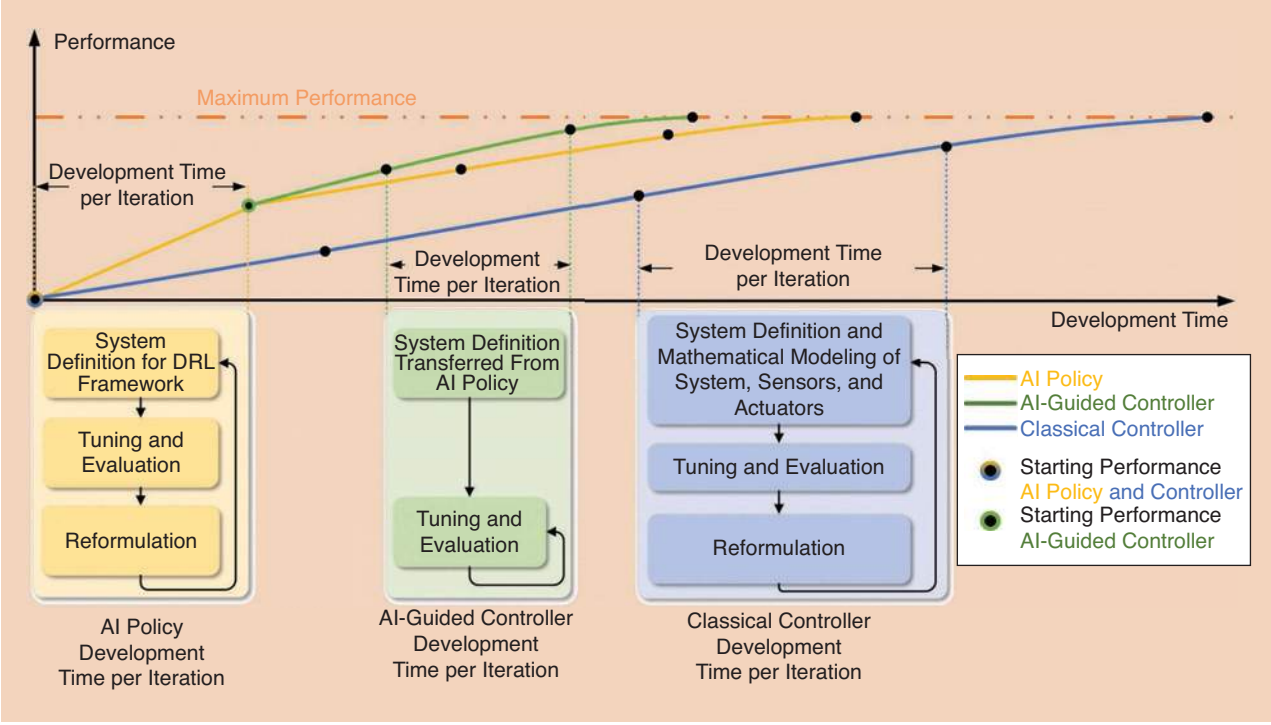


Figure 2. A qualitative depiction of the performance of controllers over iterations: leveraging the solution provided by an AI policy decreases the development time of controllers compared to AI policy and traditional controller development, while starting at a higher initial performance. The width between every black dot indicates the relative development time of the individual control design approaches, with the shortest development time per iteration for the AI-guided controller.

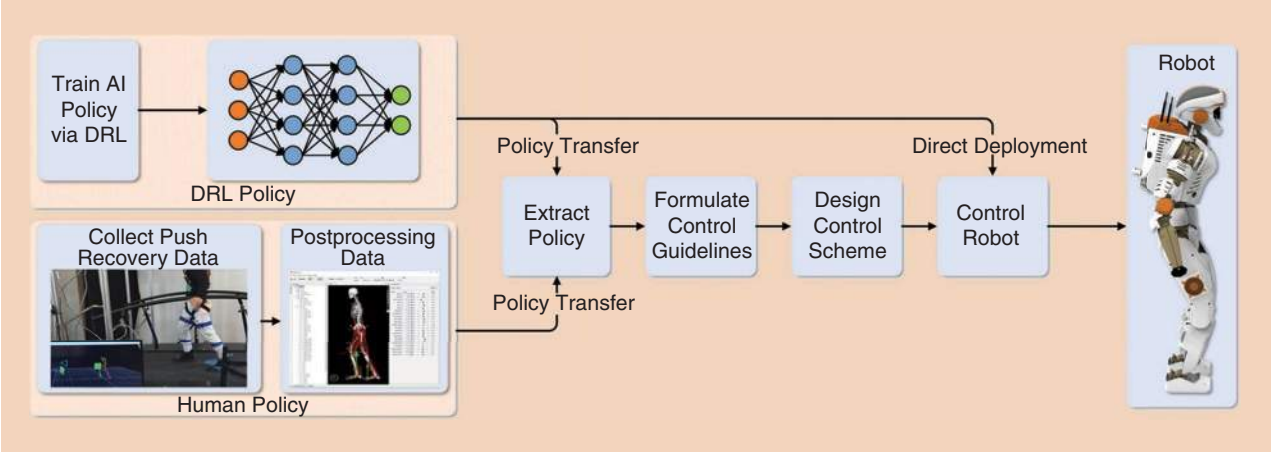


Figure 3. The proposed control design approach of policy transfer from AI policies to real robotic platforms. By understanding the underlying concepts of the learned policies, we can reverse engineer a controller that closely resembles the behavior of the AI policy while being transparent and safe.

study, we obtain a certifiable, analyzable optimal controller that does not require any state machine or switching mechanism, while exhibiting human-like push recovery strategies, such as ankle, hip, toe, and stepping, all in a coherent optimization process.

Generating Complex Motions for Humanoid Robots Through DRL

To use DRL policies as a basis for analysis, these policies must reach a certain performance threshold that ideally surpasses traditional control approaches in both the types of motions it can generate and the amount of disturbances it can withstand. DRL is capable of learning locomotion and fall recovery policies that surpass traditional control approaches for quadruped robots in terms of power efficiency and versatility of motion [1].

In this section, we present a hierarchical learning framework for achieving versatile behaviors during push recovery for humanoid robots, as proposed in [5]. The learned policy exhibits a wide range of balancing strategies comparable to human push recovery. In particular, the learned policy is able to withstand external disturbances by modulating the center of pressure (CoP) (ankle strategy), angular momentum (hip), center of mass (CoM), height (toe), and support polygon (stepping), and the policy surpasses traditional control methods with respect to the disturbances it can withstand.

All motions are trained for deployment on NASA's humanoid Valkyrie [6], a 44-DoF bipedal humanoid robot designed for operating in disaster-response scenarios and extraterrestrial planetary space missions such as unmanned predeployments on Mars. Valkyrie is built to operate in human-engineered environments, standing 1.87-m tall and weighing 129 kg with ranges of motion similar to humans. The locomotion and manipulation of Valkyrie is enabled through 25 series-elastic actuators in its arms, torso, and legs; its respective joint limits are described in Table 1. For sensing, Valkyrie has proprioceptive and exteroceptive sensors consisting of multiple gyroscopes, accelerometers, load cells, pressure sensors, sonar, lidar, depth cameras, and stereo sensors.

For training the policy, the physics simulator PyBullet [7] was used. PyBullet is able to simulate physics faster than real-time expediting of the training process and also to simulate collisions and soft and rigid dynamics by loading objects defined in the unified robot description format (URDF). The Valkyrie URDF model closely replicates the real robot and is provided by NASA; it includes physical quantities such as inertia, mass distribution, sensor noise, friction, and damping [6].

Learning Framework

Our DRL framework (Figure 4) has an actor–critic architecture, in which both the actor $\pi_\theta(a \mid s)$ and the critic $V_\phi(s)$ are NNs parametrized by the weights θ and ϕ, respectively.

The AI policy $\pi_\theta(a \mid s)$ is trained through a policy gradient method called *trust-region policy optimization* (*TRPO*) [8]. Policy gradient methods directly model and optimize the policy that will yield the highest reward $J(\theta)$:

$$\begin{aligned} J(\theta) &= \sum_{s\in\mathcal{S}} d^\pi(s)\, V^\pi(s) \\ &= \sum_{s\in\mathcal{S}} d^\pi(s) \sum_{a\in\mathcal{A}} \pi_\theta(a \mid s)\, Q^\pi(s,a), \end{aligned} \tag{1}$$

with stationary distribution $d^\pi(s)$, state value function $V^\pi(s)$, and action value function $Q^\pi(s,a)$ under policy π_θ. Using the gradient $\nabla_\theta J(\theta)$, gradient ascent is used to find an optimal parameter set θ that maximizes the reward.

Because the true value functions $V^\pi(s), Q^\pi(s,a)$ are not known, a critic $V_\phi(s)$ estimating the true value function $V^\pi(s)$ is trained. The critic $V_\phi(s)$ is updated by minimizing the expected loss $L_V(\phi)$ via gradient descent:

$$\min_\phi L_V(\phi) = \min_\phi \mathbb{E}\left[(V_\phi(s_t) - G_t)^2\right], \tag{2}$$

with value function estimation $V_\phi(s_t)$ and return G_t. The return $G_t = \Sigma_{k=t}^{\infty} \gamma^{k-t} r_k$ is the discounted cumulative reward using the discount factor $\gamma \leq 1$ and reward r_t.

Using the value function estimation $V_\phi(s_t)$, the actor's parameters θ are updated by maximizing the reward (1) via stochastic gradient ascent. Additionally, TRPO improves the training stability through trust-region optimization—constraining the Kullback–Leibler (KL) divergence during a policy update, thus preventing large policy changes that could cause instability—and reduces the variance of the policy gradient estimate using an advantage estimation A_t:

Table 1. The mechanical specifications of Valkyrie.

	Arm				Torso			Leg					
	Shoulder Roll	Shoulder Pitch	Shoulder Yaw	Elbow Pitch	Torso Roll	Torso Pitch	Torso Yaw	Hip Roll	Hip Pitch	Hip Yaw	Knee Pitch	Ankle Pitch	Ankle Roll
Lower joint position [*rad*]	−1.3	−2.9	−3.1	−2.2	−0.2	−0.1	−1.3	−0.6	−2.4	-0.4	−0.1	−0.9	−0.4
Upper joint position [*rad*]	1.5	2	2.2	0.1	0.3	0.7	1.2	0.5	1.6	1.1	2.1	0.7	0.4
Joint velocity [*rad/s*]	5.9	5.9	11.6	11.5	9	9	5.9	7	6.1	5.9	6.1	11	11
Joint torque [*Nm*]	190	190	65	65	150	150	190	350	350	190	350	205	205

$$\max_{\theta} J(\theta) = \max_{\theta} \mathbb{E}\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)} A_t(s_t)\right],$$
$$\text{subject to} \quad \mathbb{E}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s))\right] \leq \delta, \qquad (3)$$

with updated policy $\pi_\theta(a \mid s)$, old policy $\pi_\theta(a \mid s)$, advantage estimate $A_t = \Sigma_{l=0}^{\infty}(\gamma\lambda)^l(r_t + V(s_{t+1}) - V(s_t))$, discount factor γ, variance/bias tradeoff parameter $\lambda \in [0, 1]$, KL divergence $D_{\text{KL}}(\cdot \| \cdot)$, and trust region δ.

During the training process, by sampling the action a_t from the stochastic policy $\pi_\theta(a_t \mid s_t)$ and performing this action, the environment returns the resulting reward r_t, and the corresponding state s_{t+1}. During every update step during training, so-called state-action-reward-state (SARS) tuples $\{s_t, a_t, r_t, s_{t+1}\}$ are collected in a batch, $\mathcal{D}$. Upon reaching a certain batch size N—the size of the batch is a hyperparameter trading off the variance of the data against the speed of training—the actor and critic are then updated through gradient ascent and descent, respectively. Although a stochastic policy provides exploration, applying random forces to the pelvis increases the policy's ability to withstand large pushes due to a larger coverage of the state space in the stored experience data. Using the additional push experience during training, the policy will be able to generalize better to push disturbances during runtime.

Inferring Actions From the AI Polvicy

Training in the presented learning framework and interacting with the environment allow the AI policy to experience which reward will be yielded to certain actions in different states. Using these experiences in the form of SARS-tuple trajectories $\tau = \{s_{1:N}, a_{1:N}, r_{1:N}, s'_{1:N}\}$ and reward maximization through gradient ascent, the policy is incentivized to perform actions that will lead to high-value states while avoiding low-value states. Consequently, the AI policy $\pi(a \mid s)$ learns how to optimally act in state s_t by performing actions a_t to maximize a human-defined reward function r.

The reward function is designed as

$$r = r_{\text{pose}} + r_{\text{CoM pos}} + r_{\text{CoM vel}} + r_{\text{GRF}} + r_{\text{contact}} + r_{\text{power}}, \qquad (4)$$

such that high rewards are given if the torso pose r_{pose}, CoM position $r_{\text{CoM pos}}$, CoM velocity $r_{\text{CoM vel}}$, and ground contact force r_{GRF} (equally distributed ground reaction forces) remain close to the nominal values. A radial basis function $r_i = \exp(-\alpha_i(x_{\text{target}} - x)^2)$ is used for the torso pose, CoM

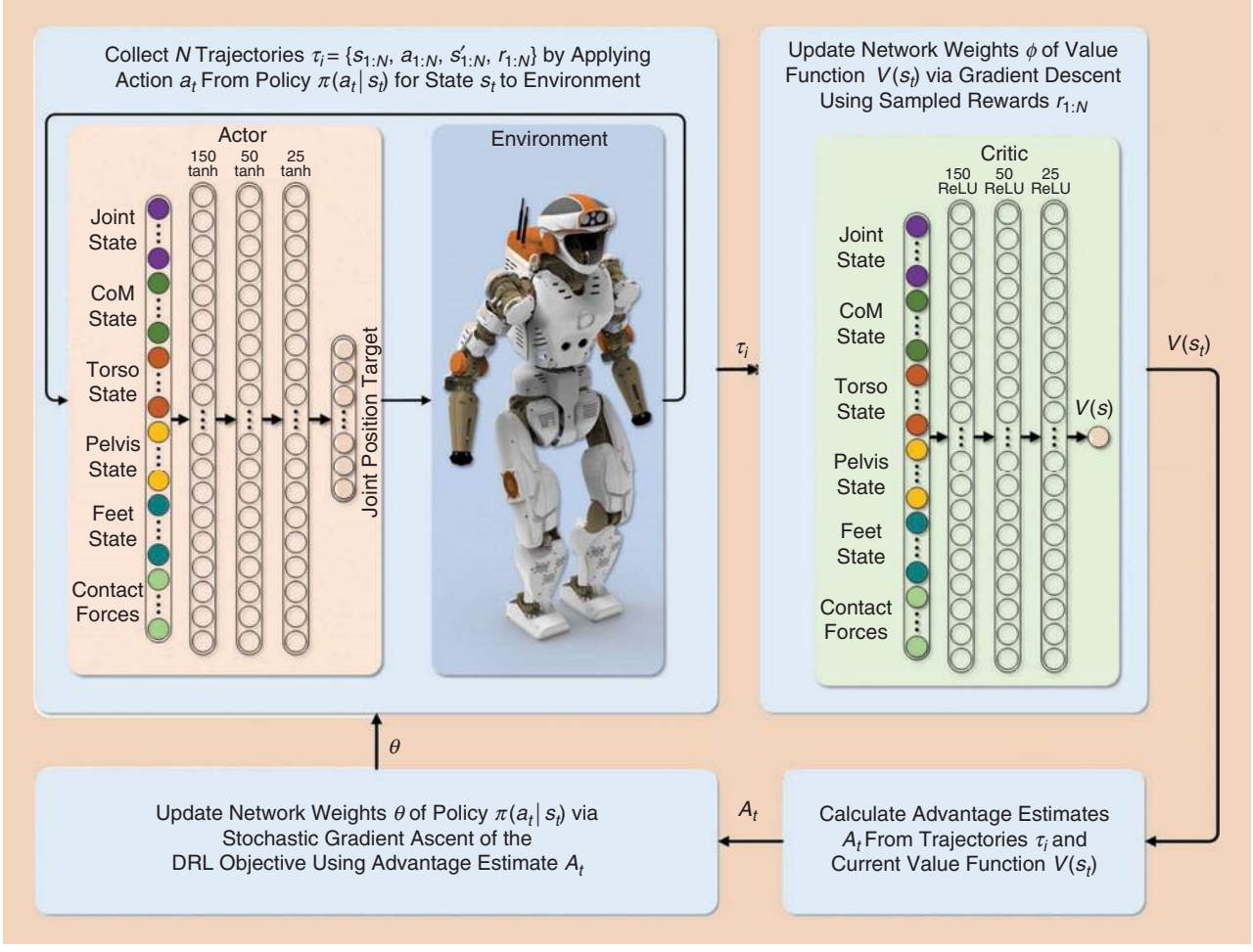


Figure 4. A learning framework for the DRL of a push recovery policy. ReLU: rectified linear unit.

position and velocity, and ground contact force to encourage the robot to be close to a target position. Furthermore, a penalty (negative value) is given proportional to the power consumption r_{power} and if the upper body is in contact with the ground or the foot is not in contact with the ground r_{contact}. For the contact penalty, a constant value is subtracted if there was no ground contact of the foot or upper body contact with the ground.

The high-level AI policy generates actions in the form of target joint-angle references at a frequency of 25 Hz by forward propagating through the actor network using the current state as input. The robot performs its motions with upper body joints locked in their nominal position. The action space $\mathcal{A} \in \mathbb{R}^{11}$ thus consists of joint positions for torso pitch, hip roll, hip pitch, knee pitch, ankle roll, and ankle pitch. The target joint angles are given to a low-level proportional–derivative (PD) controller operating at 500 Hz to generate the joint torques that are ultimately applied to control the robot [Figure 5(a)].

The state space $\mathcal{S} \in \mathbb{R}^{47}$ consists of the joint position and velocity of the actuated joints, pelvis states (translational and angular velocity, and orientation), CoM states (translational velocity and position in local frame), ground contact force, torso position (in local frame), and foot position (in local frame). The state is sampled at a frequency of 500 Hz and filtered by a first-order Butterworth filter with a cutoff frequency of 10 Hz. More details regarding the formulations of the learning framework can be found in [5].

Behaviors of the AI Policy

Training a robust push recovery policy in the presented learning framework requires 6–8 h of simulation time on a commercial desktop PC (Intel i7 6700 K, Nvidia Titan X, and TensorFlow) and equates to 1–2 days in real time. The learned push recovery policy demonstrates human-like push recovery strategies such as the ankle, hip, toe, and stepping strategies, that emerge at different levels of disturbance (Figure 1). The policy performs well in the presence of external disturbances and large sensor noises due to the filtering and sufficient amount of exploration.

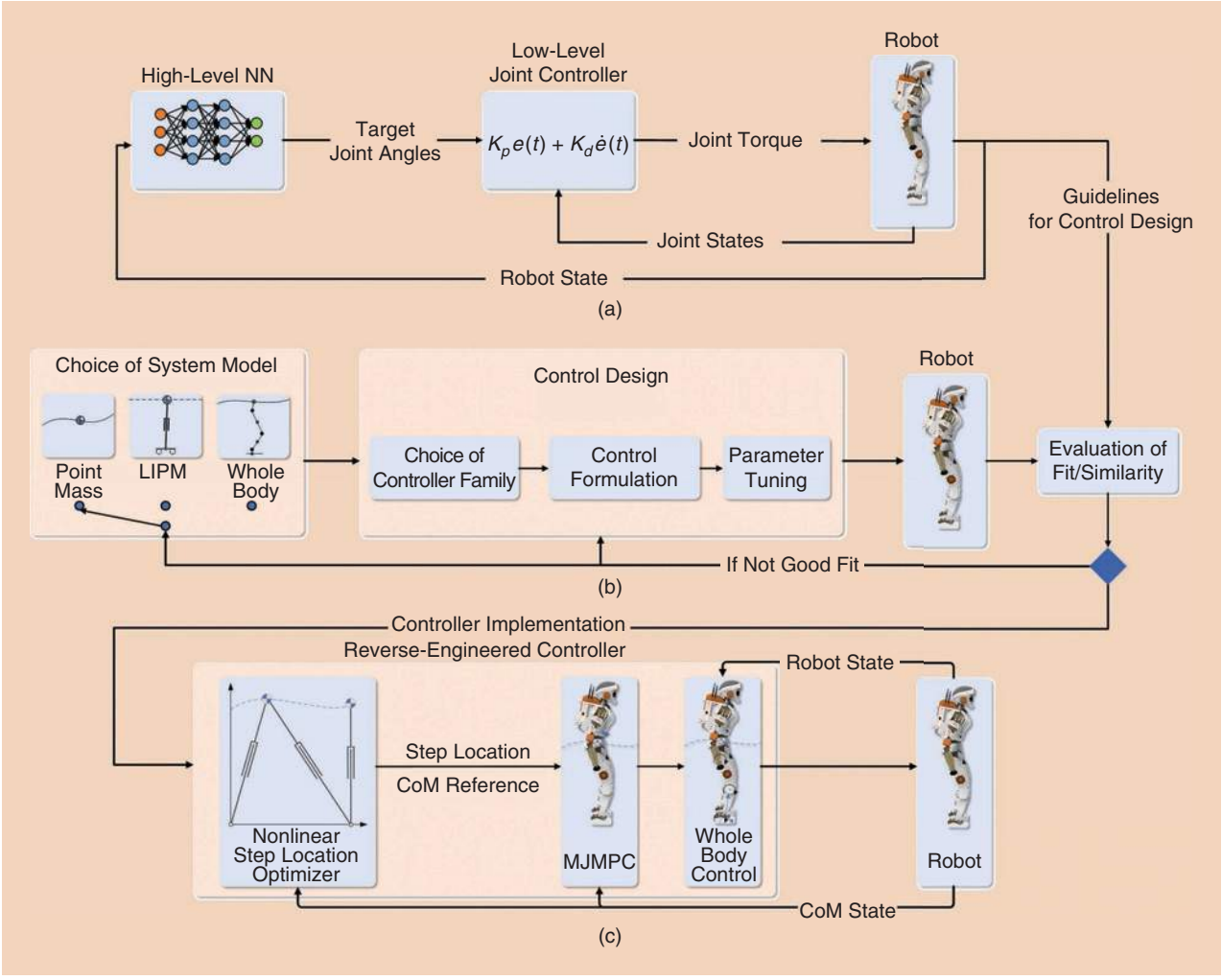


Figure 5. The process for AI-aided control design. (a) The hierarchical control system for push recovery. The high-level NN is learned as depicted in Figure 4. (b) The design process of the engineered controller. (c) The implementation of the designed controller into a whole-body control framework. LIPM: linear inverted pendulum.

Notably, the AI is able to generalize to external disturbances it was not trained for. Figure 6 shows snapshots of Valkyrie in which an impulse push on the shin is performed. The policy is able to generalize to this untrained case by generating a stepping behavior. This generalization ability further extends to the ability to recover from an impact during landing from a 0.55-m height.

The performance of the policy learned by the AI is compared with push recovery controllers in terms of maximum-impulse disturbance. The AI demonstrates the ability, comparable to state-of-the-art push recovery controllers, to withstand a wide range of impulse disturbances. The details of the comparison are presented in [5]. In the "Benchmarking Between Human and AI Policies" section, we further show that the motions are human-comparable, both quantitatively and qualitatively.

Understanding and Learning the Fundamental Principles From the AI Policy

The idea of using AI-generated policies as inspiration for solving hard problems has taken flight in other fields, most notably in games such as Go, Chess, and Shogi [9] for which the AI policy achieved superhuman-level performance. Analyzing the concepts and solution process of the AI allows

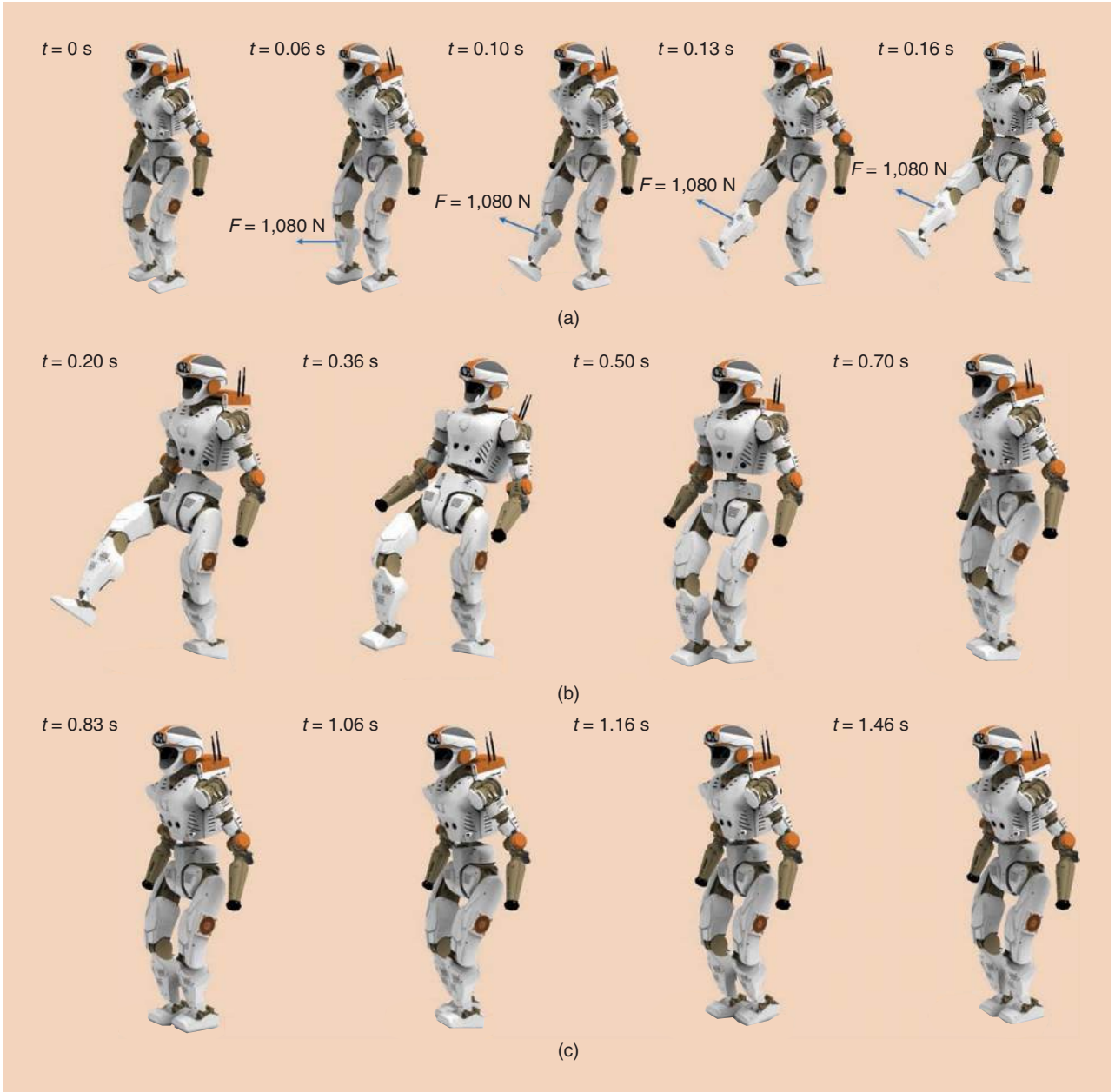


Figure 6. A Valkyrie recovering from an unexpected disturbance in a test scenario never encountered during training (impulse at the shin of 108 Ns). The learned policy naturally evolves and generates a stepping behavior. (a) A nominal starting pose (left) and the motion during the disturbance (being pulled at the shin). (b) and (c) The recovery reactions where Valkyrie takes three steps backwards and successfully recovers balance.

humans to construct better solutions and improve the way humans approach these problems: AlphaGo, a superhuman policy that beat 18-time world champion Lee Seedol four games to one, has subsequently been analyzed to provide humans with insights on why and how AlphaGo won. The release of AlphaGo Teach enabled Go players to analyze strategies and changed the way in which humans played the game: AlphaGo Teach helped to quantify the value of starting the game (suggesting that the current compensation of 6.5 points for not starting puts the nonstarting player in an advantage), redefined conventional openings, and utilized unconventional moves that went against conventional human wisdom and were previously deemed disadvantageous.

Inspired by the performance of the AI policy, this section aims to present an approach to reverse engineer the policy to obtain a unified controller exhibiting the same push recovery strategies as that of the AI policy. Reverse engineering the AI policy maintains its versatility while enabling us to analyze and modify the controller with respect to stability and safety.

We aim to find a controller that is able to stabilize the system and has a close similarity and fit to that of the DRL policy. To this end, the choice of system model and controller type needs to be made. In this section, we demonstrate that the DRL policy can be accurately replicated using point-mass dynamics, controlled by a minimum-jerk controller. This indicates that the NN may use a point-mass template internally and try to optimize the jerk.

The design process (Figure 5) involves three steps: acquiring state-action data from the AI policy, reverse engineering the policy, and deploying the controller in a whole-body control framework. First, various state-action pairs are acquired through simulation for different external disturbances. During the reverse-engineering process and based on the criterion of least square error fit, both a suitable system dynamics representation and a controller using these system dynamics will be determined. Lastly, in the deployment phase, the engineered controller will be used to generate step locations and the CoM trajectory, which will then be tracked by a whole-body controller.

Analyzing the AI Policy

The actor NN can be analyzed thoroughly with the goal of finding guidelines and quantities that enable general push recovery. The insights gathered from analyzing the actor can be used to improve controller design. Methods for analyzing and interpreting the NN are mainly of a visual nature and originate from the field of computer vision.

In this article, we visualize the NN's activation using t-distributed stochastic neighbor embedding (t-SNE) to project the activation of NNs onto a 2D plane while preserving neighborhoods and clusters in projections [10]. t-SNE is performed on the neuron activations to project the high-dimensional NN activation onto a 2D space to investigate the generalization behavior of the policy. In the final projection, data points that represent a similar NN activation are grouped nearby with high probability, whereas nonsimilar data points are distant from each other. We treat all neuron activations (compare Figure 4) during one time step as one high-dimensional data point for the t-SNE analysis. For every time step of a trial, we collect all neuron activations across disturbances ranging from 0 to 210 Ns. During these disturbances, all four push recovery strategies (Figure 1) occurred and are labeled by color in Figure 7.

As shown in Figure 7, the distinct strategies cover separate regions and thus partly explain the ability of the AI policy to generalize across different disturbances, as the activation is similar for the same strategy. NN representations labeled as the same strategy are perceptually similar and projected close to each other. The points that represent the ankle strategy cluster well with few outliers and are visually distinct from other strategies. The toe and stepping strategies are also quite distinct from each other. Furthermore, the lack of activations corresponding to the hip strategy (blue points in Figure 7) indicates that the AI policy does not utilize the angular momentum as much as the other strategies and could suggest that the hip strategy should be used less than other strategies, such as the toe strategy. The hip strategy does not cluster as clearly as the other strategies; thus, we hypothesize that it is an artifact of other motions rather than an intentional motion. The effectiveness of the hip strategy is further questioned in the literature [11], which should be kept in mind when reverse engineering the AI policy.

Incorporating AI Policy-Inspired Principles in the Control Design

The close resemblance between the DRL [dashed lines in Figure 8(a) and (b)] and human data [dashed lines in Figure 9(a) and (b)], for which strong evidence of being minimum jerk exists [4], [12], along with the typical smoothness characteristics of minimum-jerk trajectories motivates the design of a minimum-jerk control scheme. Inspired by this, reverse engineering the AI policy involved investigating whether it is also minimum jerk. For the control design process of reverse engineering the AI policy, we followed the same processes as those in Figure 5. Our study eventually found that applying the controller proposed in our previous work [12] leads to the best-fitting results and a strong resemblance between the AI policy and the reconstructed controller. In contrast, the subsequently tested different models [point-mass, inverted pendulum (IP), and whole-body models] and controllers (PD control, linear quadratic regulators, and MPC) exhibit a worse fitting to the data. We designed an MPC controller that uses point-mass dynamics as in Figure 10(a) to minimize the jerk of the CoM state for motion generation and feedback control. To prevent generating arbitrarily large motions that violate the actuation constraints of the robot, the CoM state is further constrained in the MPC scheme.

The idea of MPC lies in solving an optimization problem at every time step. Using the feedback of the current state, a closed-loop control behavior can be achieved. The objective function $J = (1/2)\int_0^{tf} (\mathrm{d}^3 x(t)/\mathrm{d}t^3)^2 \,\mathrm{d}t = (1/2)\int_0^{tf} u(t)^2 \,\mathrm{d}t$ is designed to minimize jerk $\dddot{x}$ (the input u of the system)

with final time t_f. The MPC solves the following constrained optimization problem:

$$\begin{aligned} \min_{u(t)} \quad & \frac{1}{2}\int_0^{t_f} u(t)^2 \mathrm{d}t, \\ \text{subject to} \quad & \frac{\mathrm{d}^3 x(t)}{\mathrm{d}t^3} = u. \\ & [x(0), \dot{x}(0), \ddot{x}(0)] = [x_0, \dot{x}_0, \ddot{x}_0], \\ & [x(t_f), \dot{x}(t_f), \ddot{x}(t_f)] = [x_f, \dot{x}_f, \ddot{x}_f], \\ & [x_{\min}, \dot{x}_{\min}, \ddot{x}_{\min}] \leq [x, \dot{x}, \ddot{x},] \leq [x_{\max}, \dot{x}_{\max}, \ddot{x}_{\max}], \end{aligned} \tag{5}$$

with initial condition $[x_0, \dot{x}_0, \ddot{x}_0]$ and terminal condition $[x_f, \dot{x}_f, \ddot{x}_f]$. The upper and lower inequality constraints of the optimization problem prevent the MJMPC scheme from outputting trajectories the subsequent whole-body controller cannot track. We constrain the maximum vertical displacement, velocity, and acceleration of the CoM as well as the maximum jerk the point mass can be exposed to. For the final state, two quantities need to be determined: the final time t_f at which the system should reach state $\mathbf{x}_f$ and the final CoM state itself, which is provided by a nonlinear step optimizer [13] and thus evokes stepping behavior if the push gets too large.

Determining Final Time via Data Fitting

Although the final time t_f at which the robot should come to a rest could also be optimized, this would introduce nonlinearity to the optimization problem. Thus, we treat the final time t_f as an open parameter: too short and it violates the physical capabilities; too long and it may get unstable or use too much energy. We determine an appropriate value for t_f via least-square-error fitting between collected DRL data and the MJMPC. Figure 11, shows that both a piecewise linear and quadratic approximation are able to fit the data.

Nonlinear Step Optimization

In an outer loop, a nonlinear step optimizer [13] provides the MJMPC scheme with a reference point. The position x_f in the final state $\boldsymbol{x}_f = [x_f, 0, 0]$ is determined by the nonlinear step-location optimizer, and the velocity $\dot{x}_f$ and acceleration $\ddot{x}_f$ are set to zero. The step optimizer is able find a step location and timing within 3 ms due to the specific formulation of the optimization problem. Consequently, we are able to run the optimizer at real time at 100 Hz. Furthermore, the nonlinear optimizer considers the kinematic constraints of the robot to determine the maximal step

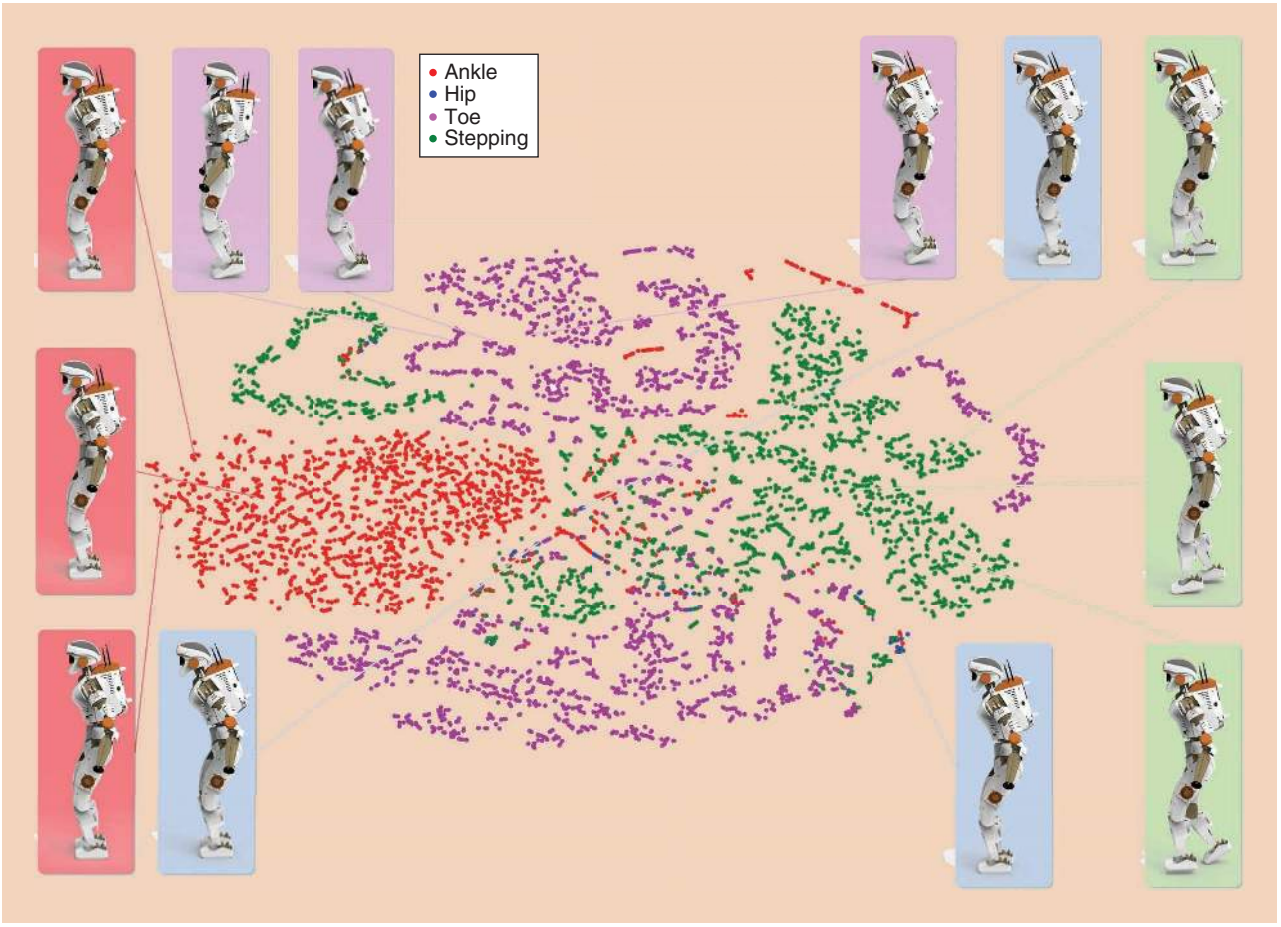


Figure 7. T-SNE, with every dot indicating the activation of all 175 neurons during one time step. The classification of every dot is determined by whether dots are nonzero (exceed a threshold) for foot velocity (step), angular momentum (hip), or vertical CoM velocity (toe) and is an ankle strategy otherwise.

length, while also constraining the robot's minimal and maximal step velocity.

Emerging Strategies and Quality of Fitting

From the x and z CoM position trajectories in Figure 8(a) and (b), it can be seen that MJMPC generates trajectories from which the push recovery strategies (Figure 1) emerge naturally. From the data of the DRL policy, we found that very little angular momentum was generated (see the "Analyzing the AI Policy" section) and, consequently do not regulate the angular momentum of the humanoid. Typical for the ankle strategy [blue solid line, Figure 8(a) and (b)], the CoP is moved within the support polygon to move the CoM position. The CoM height modulation emerges from regulating the CoM height to its nominal position. For large pushes, the step optimizer sets a new reference position, and, thus, stepping behavior emerges.

To show the quality of fit, we identify the open parameters by optimizing these via least square error between the controller and DRL policy. We apply k-fold cross validation across 2,000 trials of the robot and show the mean and standard deviation in Table 2 with an average coefficient of determination of 0.95 and 0.91 for the x and z directions, respectively.

As presented in Figure 8(a) and (b), using the methodology described in the "Determining Final Time via Data Fitting" section, a final time t_f can be chosen such that the engineered policy fits the DRL, indicating that MJMPC is a suitable controller for resembling the AI policy. The fit for the vertical component is slightly worse due to the fact that the approximation of t_f for the CoM height was not as good as it was for the sagittal component.

Realizability of MJMPC on Real Systems

In this section, we propose a framework for the real-world deployment of the generated push recovery motions and demonstrate the feasibility of the motions generated from MJMPC. To deploy the push recovery motions on a real robot while guaranteeing stability and implementability, a whole-body controller is included in the control framework (Figure 12). To this end, quadratic programming (QP)-based whole-body controllers can be leveraged to track reference motions while providing stability; these were successfully deployed on humanoids such as Valkyrie [6], Atlas [14], and HRP-2 [15]. Aside from their ability to incorporate stability and feasibility, thus guaranteeing constraints in the optimization problem formulation, off-the-shelf QP solvers can solve QP problems extremely quickly and hence enable loop

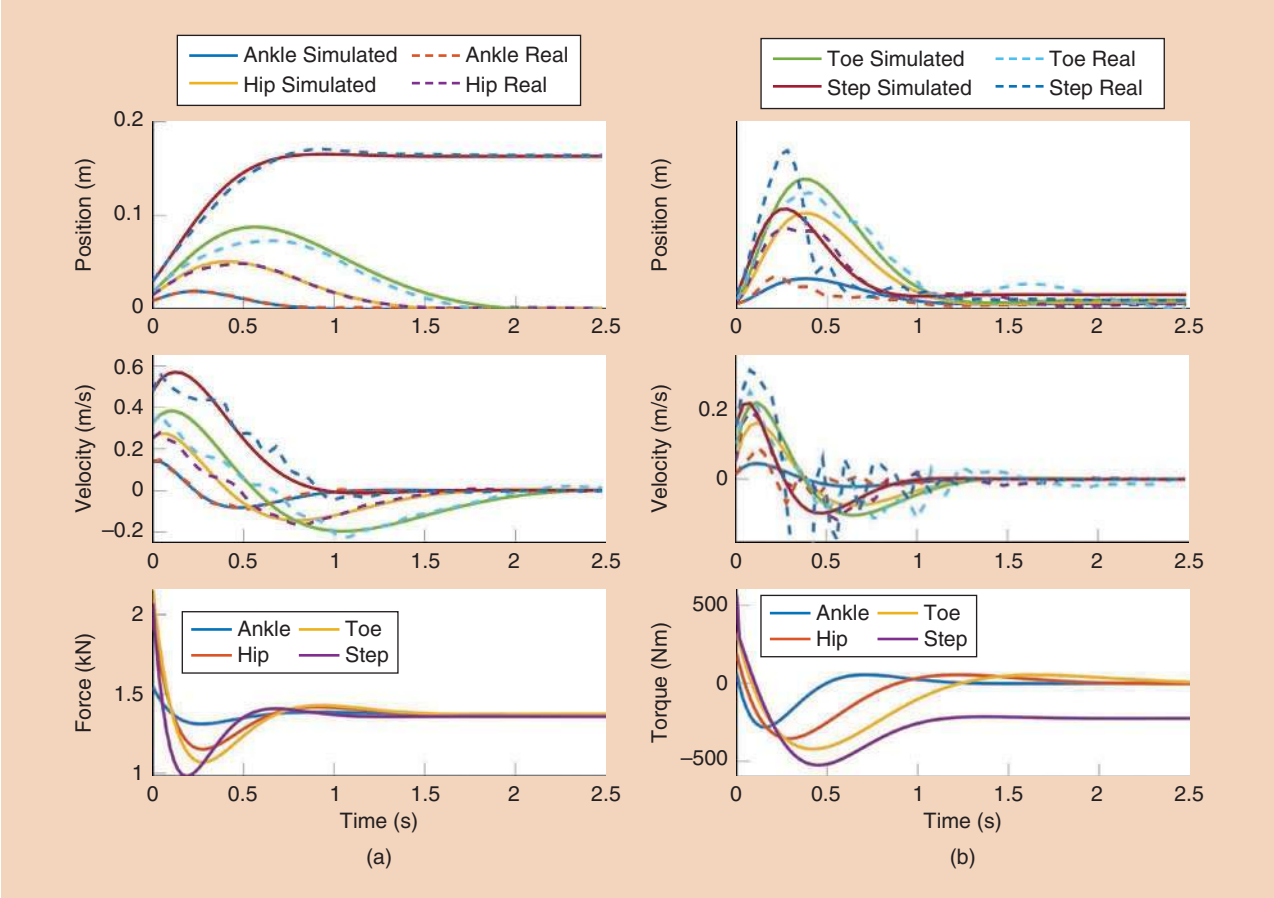


Figure 8. The fitting between robot data and the reverse-engineered controller. (a) The x position, velocity, and required force. (b) The z position, velocity, and required torque.

closure at more than 500 Hz. Furthermore, in practice, whole-body QP controllers exhibit robustness to model uncertainties due to fast, frequent loop closures and work reliably on the real Valkyrie platform.

After the MJMPC generates a reference CoM trajectory y_{ref}, a whole-body controller minimizes the tracking error $J_{\text{task}} = (1/2) \| y - y_{\text{ref}} \|$ while guaranteeing physically feasible torques that realize the push recovery motion. The reformulation of the tracking tasks of $J_{\text{task}} = (1/2) \| y - y_{\text{ref}} \|^2 = (1/2) \| AX - b \|^2$ with state $X = [\ddot{q}, \tau, \lambda]^T$ consisting of torque commands τ, joint accelerations $\ddot{q}$, and contact wrench λ yields the matrices $H = A^T A$, $f = -A^T b$. The QP problem is formulated as

$$\begin{aligned} &\min_X X^T H X + f^T X, \\ &\text{s.t.} \quad A_{\text{eq}} X + B_{\text{eq}} = 0, \\ &A_{\text{ineq}} X + B_{\text{ineq}} \geq 0. \end{aligned} \tag{6}$$

Further tasks, e.g., tracking feet trajectories from the nonlinear step-location optimizer, regularization terms, or other tasks benefiting the stability of the controller, can be stacked in $A = [w_0 A_0, w_1 A_1, ..., w_n A_n]^T$, $b = [w_0 b_0, w_1 b_1, ..., w_n b_n]^T$, and task priorities are represented by the weights w_i, where $i = 0, ..., n$ is the ith task (for more details, compare [14]).

The equations of motion-guaranteeing physical coherence between the states form the equality constraints of the QP problem (6):

$$\begin{bmatrix} M(q) & -S & -J^T(q) \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \tau \\ \lambda \end{bmatrix} + h(q, \dot{q}) = 0, \tag{7}$$

with inertia matrix $M(q)$, selection matrix S, Jacobian matrices $J^T(q)$ of the contact links, and nonlinear effects $h(q, \dot{q})$.

In addition to the physical coherence of the solution, locomotion-specific constraints are formulated, ensuring that the robot will not fall over. This is achieved by imposing inequality constraints in the QP problem on the contact wrench $\lambda = [f_x, f_y, f_z, \tau_x, \tau_y, \tau_z]$ between its feet and the ground. Slippage in the x, y direction is prevented by constraining the force in the respective direction $|f_x| \leq \mu f_z$, $|f_y| \leq \mu f_z$ for a given friction coefficient μ. Unilateral forces (no suction of the feet to the ground) are guaranteed by $f_z > 0$. A stability constraint is achieved by constraining the CoP within the support polygon $|\tau_x| \leq \mathcal{Y} f_z$, $|\tau_y| \leq \mathcal{X} f_z$ with dimensions $\mathcal{X}, \mathcal{Y}$ of the support polygon. Lastly, yaw slippage is prevented by $\tau_{\min} \leq \tau_z \leq \tau_{\max}$ with $\tau_{\min} = -\mu(X+Y) f_z + |\mathcal{Y} f_x - \mu \tau_x| + |\mathcal{X} f_y - \mu \tau_y|$, $\tau_{\max} = +\mu(X+Y) f_z - |\mathcal{Y} f_x + \mu \tau_x| - |\mathcal{X} f_y + \mu \tau_y|$.

The physical feasibility of the motions on the actuators during push recovery motions can be further shown using an IP model [16] as in Figure 10(b), which simplifies the dynamics of a robot into an IP whose length can be extended via a "kick force" and a torque applied to the pivot point corresponding to the ground reaction force and torque of the real robot. Using the torque limits of the actuators, the robot can

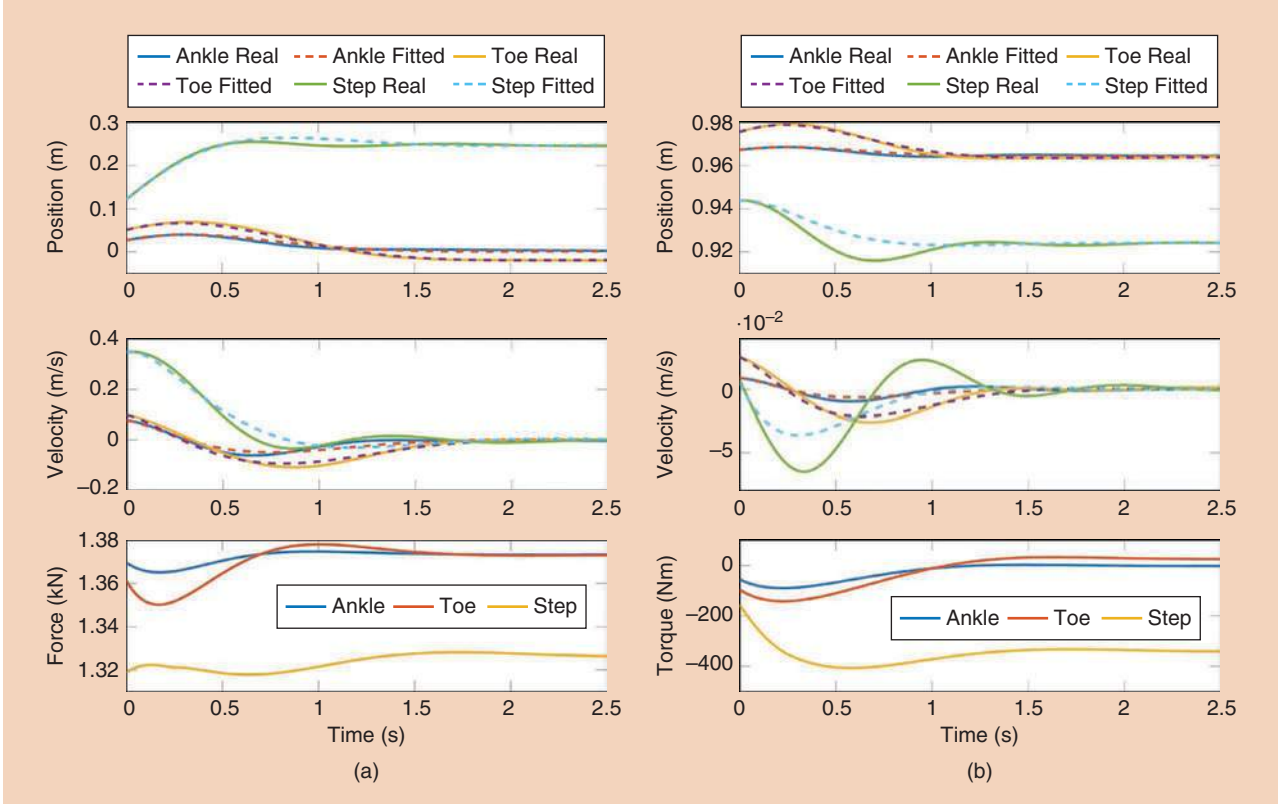


Figure 9. The fit between human data and the reverse-engineered controller. (a) The x position, velocity, and required force. (b) The z position, velocity, and required torque.

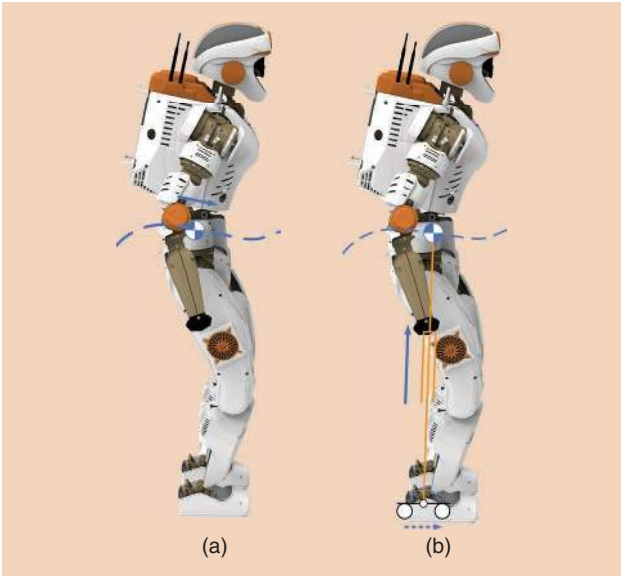


Figure 10. The system models used for control design. (a) A point-mass model with a free-moving CoM independent of actuation. This model is used for our proposed MJMPC. (b) The IP model with actuated ankle torque and kick force and free-moving CoM according to the IP motions. This model is used to validate and verify that the controller generated a feasible and implementable trajectory.

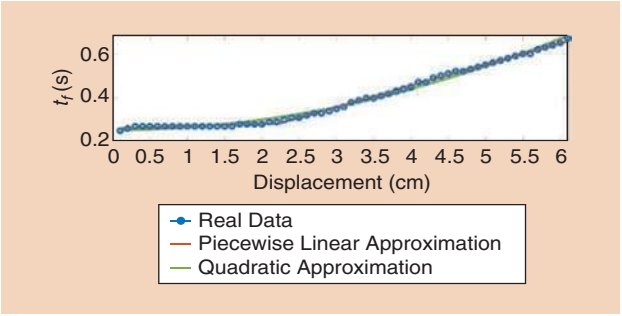


Figure 11. The CoM displacement from a nominal pose over t_f for the robot.

produce a ground contact force of $f_{\max} = 2{,}500$ N and torque of $\tau_{\max} = 1{,}000$ Nm. Using the IP model, the required ground contact forces and torques for tracking the CoM reference can be calculated and are shown in Figure 8(a) and (b). For all four push recovery scenarios, the required force and torques are well below the maximal generatable ground contact force and torque of $f_{\max} = 2{,}500$ N and $\tau_{\max} = 1{,}000$ Nm, respectively.

In summary, feasible and stable motions are enabled using this proposed control framework to first generate push recovery motions via MJMPC and then track the motions using a whole-body controller. The inequality constraint (5) of the MJMPC optimization problem explicitly considers at which speed the whole-body controller can track a reference motion; therefore, the trackability of the reference motion is guaranteed. Furthermore, for the whole-body controller, the CoM height modulation does not necessarily come from a toe lift but can be any leg extension that increases the CoM height. This is possible if the gait is not performed with stretched knees as is mostly the case in robot control tasks that aim to prevent singularities in a stretched knee pose.

Table 2. The average coefficient of determination (R^2) over 2,000 trials between DRL and robot trajectories for the ankle, hip, toe, and stepping strategies.

	Mean R^2				
Axis	**Ankle**	**Hip**	**Toe**	**Step**	**Total**
X	0.97	0.94	0.93	0.96	0.95
Z	0.9	0.91	0.96	0.88	0.91

Benchmarking Between Human and AI Policies

To this point, we have shown that methods of policy extraction can be used to estimate and reconstruct control policies from DRL. Interestingly, when these methods are applied to human movements, similar strategies are revealed [12]. In this section, we compare the findings from our previous work with those based on the AI policy to highlight the similarities between policies extracted from humans and DRL. This will demonstrate that, even though training in DRL can take just hours, the emergent behaviors are similar to human behaviors, which are highly successful only after being refined over months. These similarities show that DRL can be a high-quality source of inspiration for control design because it can closely match human-level behavior.

Data Collection

To compare human and DRL push recovery policies, human data was recorded in 60 trials while short impulses were applied close to their CoM [Figure 13(a)], analogous to those applied to robots during DRL testing. After the push ends [Figure 13(b)], the subject takes a recovery action [e.g., stepping in Figure 13(c)]. Different impulse magnitudes were applied to obtain a wide range of push recovery behavior. Impulses were measured by a force/torque sensor. Movement was measured via a Vicon motion-tracking system, and the data were postprocessed using OpenSim and gait analysis tools, as discussed in [17].

Push Recovery Strategies in Humans and AI Policies

An analysis of the human data [12] shows that various strategies are used for push recovery, as depicted in Figure 1. Each strategy is associated with a control action such as the modulation of CoP, angular momentum, CoM height, or support polygon. Interestingly, a very similar structure also emerges in the AI policy, which demonstrates the similarities between DRL and human policies.

Control Strategy

We find that a single controller implementing the MJMPC scheme in the "Understanding and Learning the Fundamental

Principles From the AI-Policy" section can explain human data across all strategies [see Figure 9(a) and (b)], as was the case for DRL. When comparing these to the DRL results [see Figure 8(a) and (b)] and observing the high coefficient of determination (Table 3) it becomes apparent that the trajectories are similar for each strategy. In summary, applying policy-extraction methods to humans and DRL revealed that they are generalizable to different sources and showed that the respective policies are very similar, as evidenced by the MJMPC controller.

Why Learn From DRL Policies Instead of From Humans?

This section shows that the policies of DRL and humans with their derived reversed-engineered controllers are similar, which raises the question: Why use an AI policy instead of learning from a human policy? The answer lies in the logistics and applicability of both methods for control design. Although humans could be used as template to gain insights on how to improve the target system (humanoid robots), learning from an AI policy yields three main advantages:

1) *Access to all internal states*: By deploying DRL on the same system as the target system, the state space is identical. Thus, data can be collected from DRL-learned policies as soon as the learning process is complete and be made immediately available for analysis. In the case of human policy analysis, data postprocessing is highly labor intensive and requires expensive data collection equipment to infer accurate joint angle, joint torques, and CoM positions [17]. Data collection

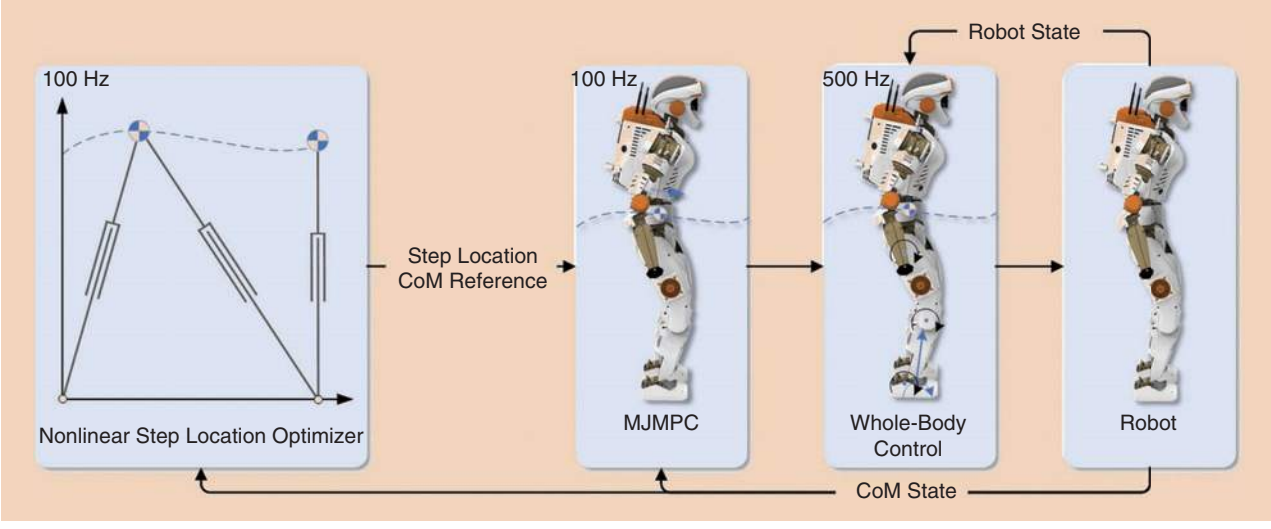


Figure 12. The control diagram for the robot.

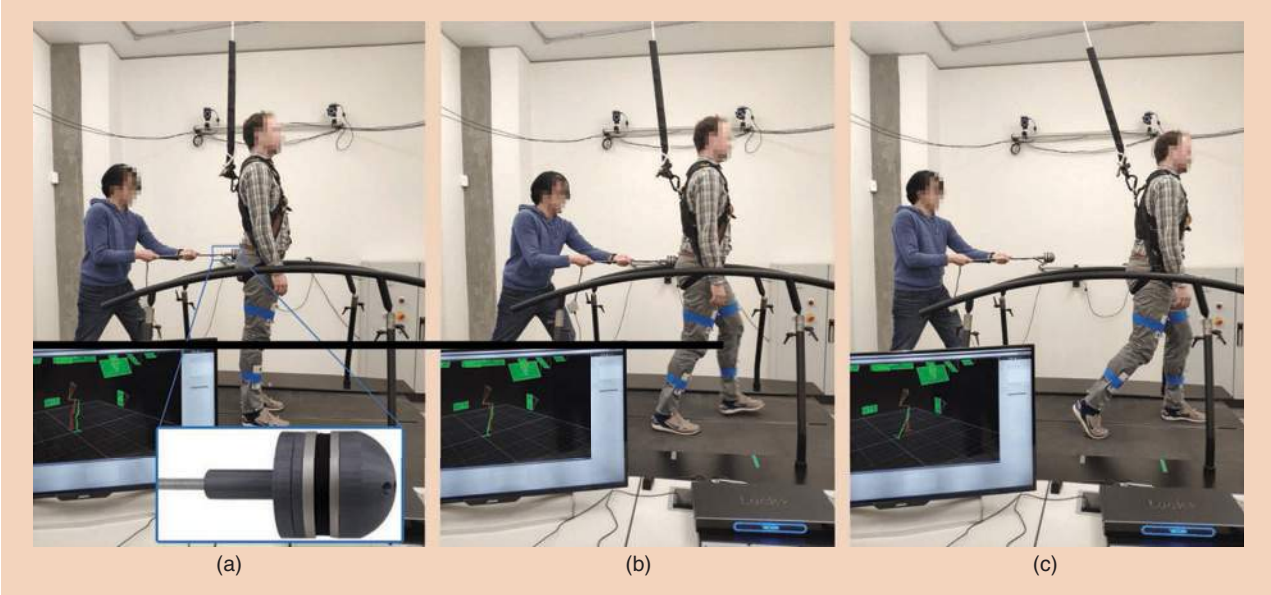


Figure 13. An experimental trial. (a) The subject stands upright during the beginning of the push. (b) The subject transitions to stepping strategy after being pushed. (c) The subject comes to a halt after the stepping action.

Table 3. The coefficient of determination (R^2) between human and robot trajectories for the ankle, hip, toe, and stepping strategies.

	Mean R^2				
Axis	**Ankle**	**Hip**	**Toe Lift**	**Step**	**Total**
X	0.76	0.84	0.95	0.89	0.86
Z	0.95	0.92	0.86	0.59	0.83

required approximately two weeks of work, including setting up experiments, collecting subjects, carrying out experiments, and processing data. These labor-intensive logistics further motivate the use of DRL to transfer policies over the collection of human data because a lot more data can be extracted in a shorter time from DRL trials.

2) *Policy transfer to nonhumanoid robots*: Using humans as a template policy allows a policy transfer only to humanoids. In contrast, DRL policies can be applied to systems that are nonhumanoid, such as quadrupeds or multilegged robots, or where the template is not available, e.g., extinct vertebrates [18].
3) *Analysis of the internal mechanisms of the policy*: For an analysis of the policy beyond its input–output relations, e.g., t-SNE analysis (see the "Analyzing the AI Policy" section), the AI policy is more accessible than that of the human policy. The AI policy is represented as an NN, and the analysis tools outlined in the "Analyzing the AI Policy" section can be leveraged to gain insights into the mechanisms of the AI policy. Analyzing the mechanisms of the human policy, on the other hand, requires a neuroscientific understanding of the brain and other involved components of humans. The summarized strengths and limitations of DRL in a number of different areas with respect to classical control and human control can be found in Table 4. Note that the limitations of DRL in optimality, robustness, and safety are canceled out by the strengths of classical control in these areas, and vice versa. The "Control + DRL" paradigm can overcome the difficulties encountered during human-inspired control (the right-hand column).

Discussion and Conclusions

In this article, we presented an alternative application of DRL: instead of directly deploying the AI policy, we aimed to formulate control guidelines from the AI policy. As a result, we bypass the major drawbacks of learned policies—unsafety and stability issues—and obtain a certifiably, optimal controller that demonstrates human-like behaviors (Figure 1). These results were obtained for the challenging task of push recovery.

Results

We showed that DRL is powerful enough to learn complex motions that resemble those of humans. The learned policy demonstrated the same push recovery strategies that can be observed in humans and exhibited similar robustness as state-of-the-art control algorithms. After analyzing the learned AI policy, we used it as a guideline and template for control design.

The engineered controller is able to reproduce the same strategies as those of human and AI policies with a close quantitative fit to the collected data. As observed in humans [12], the policy minimizes jerk and implements feedback control via MPC. Furthermore, an analysis of the required torques and forces on the system shows that the trajectories provided by the engineered policy are realizable on the real system.

We further compared the decoded DRL and human push recovery policies and found, surprisingly, that the AI policy has a strong similarity to human policies. We hypothesized that both the AI and humans are able to identify the key features in the problem, as required for high-quality performance. This finding is interesting due to the time required for the DRL agents to acquire human-comparable push recovery abilities: learning a good AI policy requires 6–8 h, and human infants require 10–18 months to learn the ability of locomotion [19].

Even though MJMPC was able to reproduce both policies and both human and AI policy can be used as a template model for control design, we found the usage of AI policies for template models to be more advantageous. This was due to the DRL data being immediately available—in contrast to the human data, which required time-intensive postprocessing—and

Table 4. A comparison of various control paradigms.

Attribute	Control	DRL	Humans	Control + DRL	Control + Human
Optimality	Optimal	Suboptimal	Near optimal	Optimal	Optimal
Robustness	High	Low	High	High	High
Behavior-emergence time	Weeks	Weeks	Months	Weeks	Weeks
Generalizability	High	High	Low	High	Low
Data collection time	N/A	Low	High	Low	High
Prior knowledge required	High	Low	High	Low	High
Safety/accountability	High	Low	High	High	High

N/A: not applicable.

having direct access to the policy in the form of an NN allowing analysis of the NN in addition to merely the input–output behavior. We performed a t-SNE analysis on the AI policy (Figure 7) and found that the DRL agent—through its interaction with the environment—decided not to use the angular momentum modulation due to its minimal effectiveness.

Challenges and Outlook

In this article, we showed that, through an analysis of template models from human and AI policies for push recovery, a certifiably safe controller for humanoid robots can be engineered. Currently, all analyses and implementations were conducted in a high-fidelity physics simulator that, despite our best efforts, differs from reality. To mitigate this problem, we proposed a control framework for real-world deployment that combines the proposed control law with whole-body controllers in a cascaded manner. Due to the shown feasibility of the motions and the reliable stability of whole-body controllers in real-world applications, we anticipate a seamless deployment of the controller onto a real system.

Our future work will investigate approaches in directly bridging this reality gap and use interactions with the real environment to close the gap in the DRL process. Moreover, we aim to apply this principled approach of designing controllers from template models to other systems (e.g., quadrupeds) and tasks (locomotion and manipulation).

Acknowledgments

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) Centres for Doctoral Training in Robotics and Autonomous Systems (EP/L016834/1) and the EPSRC Future AI and Robotics for Space (EP/R026092/1).

References

[1] J. Hwangbo et al., "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, p. eaau5872, 2019. doi: 10.1126/scirobotics.aau5872.

[2] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2017, pp. 3389–3396. doi: 10.1109/ICRA.2017.7989385.

[3] M. Zhang et al., "Deep reinforcement learning for tensegrity robot locomotion," in *Proc. 2017 IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 634–641. doi: 10.1109/ICRA.2017.7989079.

[4] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *J. Neurosci.*, vol. 5, no. 7, pp. 1688–1703, 1985.

[5] C. Yang, K. Yuan, W. Merkt, T. Komura, S. Vijayakumar, and Z. Li, "Learning whole-body motor skills for humanoids," in *Proc. 2018 IEEE-RAS 18th Int. Conf. Humanoid Robots (Humanoids)*, pp. 270–276. doi: 10.1109/HUMANOIDS.2018.8625045.

[6] N. A. Radford et al., "Valkyrie: NASA's first bipedal humanoid robot," *J. Field Robot.*, vol. 32, no. 3, pp. 397–419, 2015. doi: 10.1002/rob.21560.

[7] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation in robotics, games and machine learning," 2017. [Online]. Available: https://pybullet.org/wordpress

[8] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Machine Learning*, 2015, pp. 1889–1897.

[9] D. Silver et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018. doi: 10.1126/science.aar6404.

[10] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[11] P. Zaytsev, W. Wolfslag, and A. Ruina, "The boundaries of walking stability: Viability and controllability of simple models," *IEEE Trans. Robot.*, vol. 34, no. 2, pp. 336–352, 2018. doi: 10.1109/TRO.2017.2782818.

[12] C. McGreavy et al., "Unified push recovery fundamentals: Inspiration from human study," in *Proc. 2020 IEEE Int. Conf. Robotics and Automation (ICRA)*, to be published.

[13] W. Hu, I. Chatzinikolaidis, K. Yuan, and Z. Li, "Comparison study of nonlinear optimization of step durations and foot placement for dynamic walking," in *Proc. 2018 IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 433–439. doi: 10.1109/ICRA.2018.8461101.

[14] S. Feng, X. Xinjilefu, W. Huang, and C. G. Atkeson, "3d walking based on online optimization," in *Proc. 2013 13th IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, pp. 21–27. doi: 10.1109/HUMANOIDS.2013.7029950.

[15] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Int. J. Robot. Res.*, vol. 33, no. 7, pp. 1006–1028, 2014. doi: 10.1177/0278364914521306.

[16] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to Humanoid Robotics,* vol. 101. New York: Springer-Verlag, 2014.

[17] D. Gordon, G. Henderson, and S. Vijayakumar, "Effectively quantifying the performance of lower-limb exoskeletons over a range of walking conditions," *Front. Robot. AI*, vol. 5, p. 61, June 2018. doi: 10.3389/frobt.2018.00061.

[18] J. A. Nyakatura et al., "Reverse-engineering the locomotion of a stem amniote," *Nature*, vol. 565, no. 7739, pp. 351–355, 2019. doi: 10.1038/s41586-018-0851-2.

[19] H. Forssberg, "Ontogeny of human locomotor control. I. Infant stepping, supported locomotion and transition to independent locomotion," *Exp. Brain Res.*, vol. 57, no. 3, pp. 480–493, 1985. doi: 10.1007/BF00237835.

Kai Yuan, School of Informatics, The University of Edinburgh, United Kingdom. Email: kai.yuan@ed.ac.uk.

Christopher McGreavy, School of Informatics, The University of Edinburgh, United Kingdom. Email: c.mcgreavy@ed.ac.uk.

Chuanyu Yang, School of Informatics, The University of Edinburgh, United Kingdom. Email: chuanyu.yang@ed.ac.uk.

Wouter Wolfslag, School of Informatics, The University of Edinburgh, United Kingdom. Email: wouter.wolfslag@ed.ac.uk.

Zhibin Li, School of Informatics, The University of Edinburgh, United Kingdom. Email: alexrobotics@gmail.com.

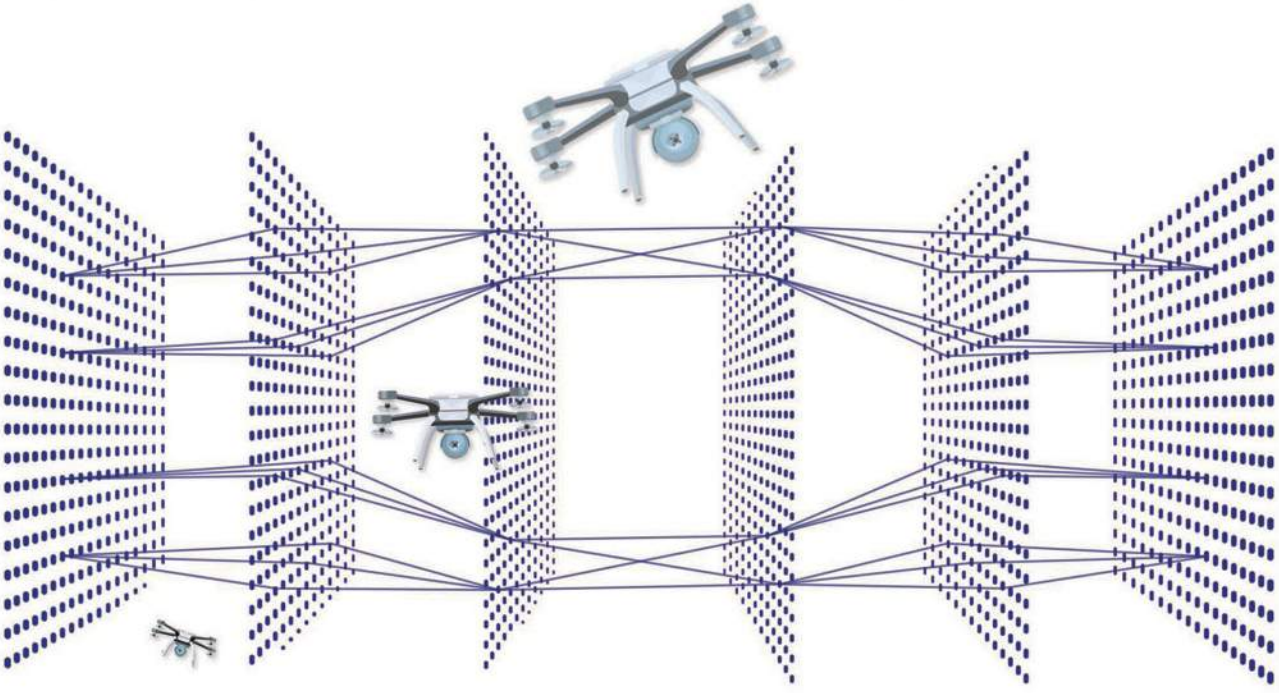
By Esen Yel, Taylor J. Carpenter, Carmelo Di Franco, Radoslav Ivanov, Yiannis Kantaros, Insup Lee, James Weimer, and Nicola Bezzo

Autonomous systems operating in uncertain environments under the effects of disturbances and noises can reach unsafe states even while using fine-tuned controllers and precise sensors and actuators. To provide safety guarantees on such systems during motion planning operations, reachability analysis (RA) has been demonstrated to be a powerful tool. RA, however, suffers from computational complexity, especially when dealing with intricate systems characterized by high-order dynamics, making it hard to deploy for runtime monitoring.

To deal with this issue, in this article, a neural network (NN)-based framework is proposed to perform fast online monitoring for safety, and an approach for the verification of NNs is presented. Training is performed offline using precise RA tools, while the trained NN is harnessed online as a fast safety checker for motion planning. In this way, at runtime, a planned trajectory can be quickly predicted to be safe or unsafe. When unsafe, a replanning procedure is triggered until a safe trajectory is obtained. The results of the trained network are tested for verification using our

Assured Runtime Monitoring and Planning

Toward Verification of Neural Networks for Safe Autonomous Operations



Digital Object Identifier 10.1109/MRA.2020.2981114

Date of current version: 14 April 2020

1070-9932/20©2020IEEE

recent tool Verisig, in which the NN is transformed into a hybrid system to provide guarantees before deployment. In the case of unverified NNs, the outputs of the verification are used to retrain the network until verification is achieved. Two illustrative case studies on a quadrotor unmanned aerial vehicle (UAV) (a pickup/drop-off operation and navigation in a cluttered environment) are presented to validate the proposed framework in simulations and experiments.

Safety Predictions

As autonomous vehicles find their way into our society, it becomes critical to guarantee safety against unpredictable uncertainties and disturbances during their operations at runtime. In fact, while model-driven motion planning and control techniques can be robust against noises and disturbances, they cannot prevent deviations from the desired behavior during system operation, potentially leading to unsafe states (e.g., crashing into an obstacle in the environment due to excessive wind disturbance). To assure safety during autonomous operations, it is necessary to take the effect of noises and disturbances into account during planning. Traditional RA tools, such as Hamilton–Jacobi reachability [1], and hybrid system RA techniques [2], [3] have proved to be very effective in providing safety guarantees by leveraging knowledge about the model of the system. However, their computational complexity makes them difficult to use at runtime, which is the subject of this article.

On the other hand, contrary to traditional RA tools, recent developments in machine learning have considerable potential to enable fast RA thanks to their efficiency and accuracy. Unfortunately, since the performance of such learning enabled components (LECs) depends significantly on the properties of the training data, it is challenging to provide guarantees in safety-critical operations. To deal with these challenges, this article presents a framework to verify LECs for fast, safe monitoring and planning of autonomous operations.

An NN trained to predict whether an operation will be safe or unsafe is verified if its output decision (safe or unsafe) always concurs with the results obtained by running the operation under the worst case conditions in which it was trained. Since it is unlikely that an NN with this strong property exists, in this article, we define one that is conservatively verified; in short, it is verified if, at the least, its safe decision output always concurs with the result obtained by running the actual operation. In other words, if the NN output is safe, the system can never reach an unsafe state; however, the NN is allowed to output unsafe when the system will be safe, since this is a conservative decision. This definition also holds in the extreme case that an NN outputs only unsafe decisions, in which a vehicle will be always safe, although it will not be able to move anywhere.

In the proposed scheme, an NN is trained to recognize safe and unsafe trajectories for an autonomous vehicle in an obstacle-populated environment. Specifically, training is performed, creating a library of trajectories and computing their reachable sets to make safety decisions; hence, the computational burden is limited to the offline stage. A trajectory is labeled safe if its reachable sets do not overlap with any obstacle in the environment; otherwise, it is marked unsafe.

Verification of the obtained NN is achieved using our recent technique Verisig [4], in which the NN is transformed into a hybrid system and the verification problem is cast as a hybrid system reachability problem. If the NN is not verified, meaning that it is not safe to be used, the output of the verification is employed as feedback to retrain the NN more conservatively until it is verified. Once the NN is verified, it is used at runtime as a safety checker for the planned trajectories from an untrained initial state under the effect of unknown online disturbances. If unsafe, a new trajectory is planned and tested until a safety-guaranteed course is obtained, if possible.

Contributions of the article include the following: with the proposed framework, we 1) develop a fast safety checking and replanning approach for autonomous vehicles' operations in cluttered environments under unknown runtime disturbances and 2) leverage a novel NN verification tool, Verisig, to verify and retrain the used NN as a fast safety monitor, before its deployment.

As a result, with this framework, it is possible to eliminate the need for computationally expensive RA tools for planning safe trajectories at runtime, and our verification method, Verisig, is capable of assessing the validity of the proposed NN. To better illustrate our proposed framework, two case studies on a quadrotor UAV are presented with simulations and experiments under the presence of unknown disturbances during runtime: 1) a pickup/drop-off mission and 2) safe navigation in a cluttered environment.

Verified Safe Motion Planning

Our verified safe motion planning framework consists of offline and online stages, as depicted in Figure 1. During the offline phase, a library of trajectories is generated. We parametrize the trajectory generation based on the initial state of the UAV, the desired goal position, the location of the obstacles on the way to the goal site, and the distance that must be maintained from these obstacles. These trajectory parameters are labeled safe or unsafe using RA, and an NN is trained with this set of parameters. To be able to use the NN in autonomous operations, it should be guaranteed that the trained network never outputs safe when the trajectory is actually unsafe. To provide this guarantee, we verify the trained NN using our recent tool Verisig, as outlined in the "Verification" section. In case the NN is not verified, the Verisig output is utilized to retrain the NN more conservatively (i.e., it outputs more unsafe decisions) until it is verified. Once the NN is verified, it is used to make decisions about the safety of a new set of trajectory parameters at runtime. If the NN decides that the trajectory is safe, the UAV

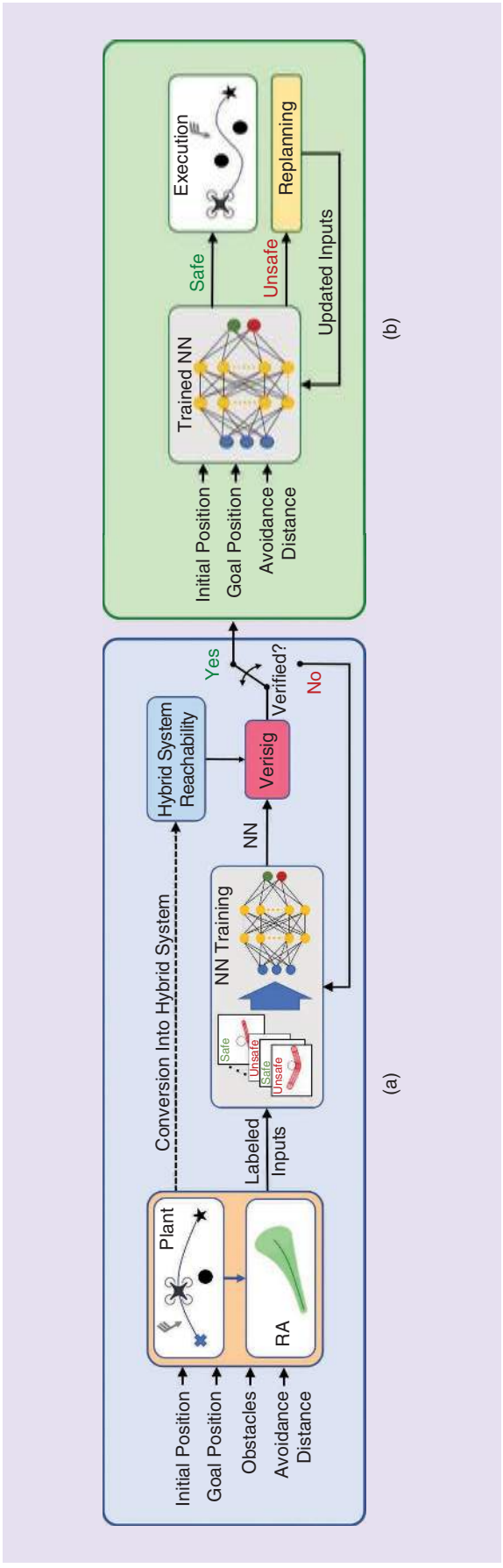


Figure 1. The architecture of the proposed framework for verifying NNs for runtime monitoring and planning autonomous operations. (a) During the offline stage, an NN is trained and verified, followed by its deployment at runtime for monitoring and replanning purposes. (b) The online stage.

executes the course. If the decision is unsafe, the trajectory is replanned by changing the parameters until a safe choice is obtained.

Reachability Analysis

As mentioned previously, RA is a very powerful method for computing the sets that a system could reach starting from an initial set. A hybrid system RA tool, Flow* [2], uses Taylor models to compute flowpipe overapproximations of the dynamics, while dReach [3] encodes the reachability problem as first-order formulas across real numbers and solves the problem using δ-decision procedures. Hamilton–Jacobi RA is also a widely used approach to provide guarantees for the safety of safety-critical systems' optimal system trajectories [5]. It performs well in terms of the generality of system dynamics, flexibility in the representation of sets, and control policy computation; however, it suffers from computational scalability [6]. A significant effort has been made to overcome the scalability problem for high-dimensional systems, such as decomposing the system dynamics [7] and using NNs to approximate the reachable sets [6], [8]. Other types of reachable sets, such as robust control invariant tubes [9], have also been proposed for safety-guaranteed UAV planning. All these traditional RA tools are very effective in providing safety assurances, although their computational complexity makes them difficult to use in making runtime safety decisions.

In the literature, there have been some efforts to make RA more usable for runtime applications. For example, in [10], the authors precomputed a library of trajectories and funnels (analogous to reachable sets) offline and combined these trajectories online to navigate in a priori unknown environments under disturbance effects. However, with this approach, the system is restricted to a discrete set of motion primitives. In [11], using Hamilton–Jacobi reachability, a lookup table is computed offline to find the bounds on the planned trajectory that are used to augment the obstacles to guarantee collision-free behavior under bounded disturbances in unknown environments. Similarly, in [12], forward reachable sets are computed offline for parameterized trajectories, and at runtime, safe trajectory parameters are picked to avoid the sensed obstacles in unknown environments with model uncertainties. Different from these works, our framework solves this problem of safe navigation in known and unknown environments under the presence of external disturbances by using verified NNs to perform safety decisions at runtime, leaving the RA computation offline. Our framework is also general and modular, meaning that any type of control and planning method can be considered. It is also independent from the choice of reachability analysis tool. Specifically, during the offline stage, reachable sets are generated for a given trajectory, and if they do not intersect with obstacles, the corresponding trajectory parameters are labeled as safe:

$$\begin{cases} s_\tau = 1 & \text{if } \boldsymbol{R}(\boldsymbol{p}_\tau, t) \cap \boldsymbol{p}_{o,j} = \varnothing \\ & \forall t \in [0, T], \forall j \in \{1, \ldots, N_o\}, \\ s_\tau = 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\boldsymbol{p}_\tau$ is the desired positions along the trajectory τ, $\boldsymbol{R}(\boldsymbol{p}_\tau, t)$ is the corresponding reachable set, s_τ is the safety label, $\boldsymbol{p}_{o,j}$ is the jth obstacle position, and N_o is the number of obstacles in the environment. An NN is trained using these safety-labeled parameters to make decisions for trajectories with different conditions.

NN Training for Safety Decisions

Following the diagram in Figure 1, after collecting a rich library of labeled trajectory parameters, we train an NN to provide safety decisions without having to run computationally expensive RA operations at runtime. The inputs to the NN are the trajectory parameters, which are the initial and final positions of the system, and the obstacle-avoidance distance. The output of the NN is a binary safe/unsafe decision. Since these conclusions are used by a safety-critical system, it is required that the NN never outputs a decision of safe if the trajectory is unsafe, as it can lead to dangerous outcomes (e.g., colliding with an obstacle). Therefore, we are interested in training an NN with zero false positives (FPs). The drawback of reducing the number of FPs is that the number of false negatives (FNs) (safe trajectories marked unsafe) may increase. Even a conservatively trained NN can output a safe decision for a set of untrained, unsafe trajectory parameters. Therefore, the absence of such FPs needs to be verified prior to its deployment.

Verification

In this article, we use Verisig to verify that the trained NN does not output safe if the robot plans an unsafe trajectory. As described in our prior work [4], Verisig was developed to verify safety properties of closed-loop systems with NN components. Verisig focuses specifically on sigmoid-based NNs and works by transforming the NN into an equivalent hybrid system. The NN's hybrid system is then composed with the plant, resulting in a new hybrid system that describes the entire closed-loop system, as depicted in Figure 1. This enables us to cast the verification problem as a hybrid system reachability problem, which is solved by an optimized hybrid system verification tool, such as Flow* [2]. Specifically, Verisig transforms the NN S into a hybrid system H_S by noting that the sigmoid derivative can be expressed in terms of the sigmoid itself, i.e.,

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)),$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid. With this observation in mind, we introduce a proxy function $\sigma_p(t, x) = \sigma(xt)$ such that $\sigma_p(1, x) = \sigma(x)$ and

$$\dot{\sigma}_p(x) = x\sigma_p(t, x)(1 - \sigma_p(t, x))$$

using the chain rule. In other words, σ_p can be considered as a state in a dynamical system that starts at $\sigma_p(0, x) = 0.5$ and is equal to $\sigma(x)$ at time 1. Thus, each neuron in the NN can be mapped to a state in a hybrid system, and each layer can be mapped to a mode. Transitions between modes occur when $t = 1$. Note that this time is local to the NN; global time does not progress during the NN's execution.

Although Verisig was originally applied to closed-loop systems with NN controllers, it applies to the setup considered in this article, as well. In particular, in the verification problem, the NN's hybrid system does not interact with the plant directly; rather, for a given set of initial conditions, we compute the reachable set for the NN's output and check whether it is possible for the robot to crash (when started from that initial set) while the NN outputs safe.

The composed hybrid system $H = H_q \| H_S$ is shown in Figure 2, where H_q is the hybrid system describing the robot's

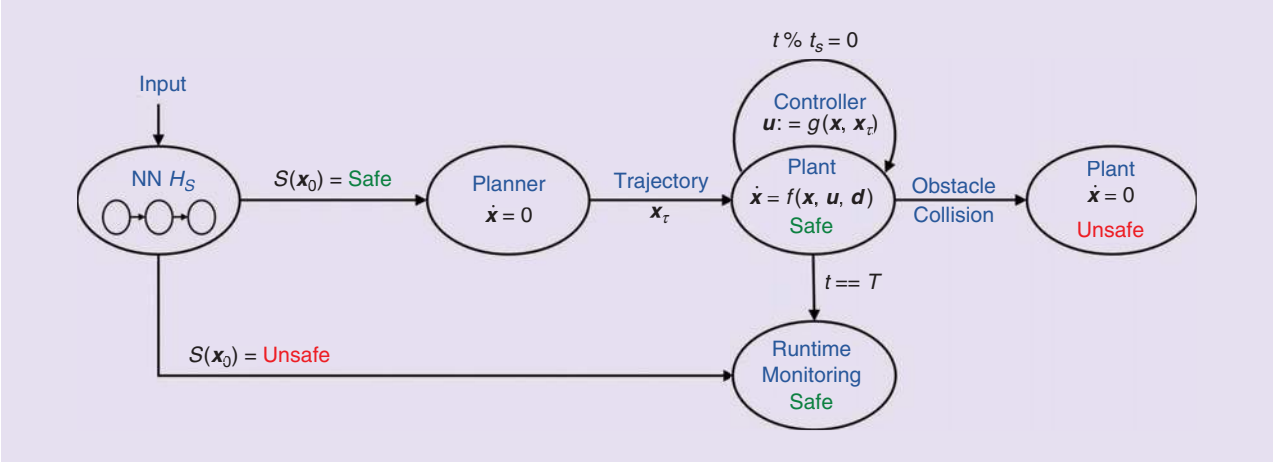


Figure 2. The composed hybrid system considered for verifying NNs for runtime monitoring. The plant dynamics evolve as a function of the state, input, and disturbance $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d})$. A controller is designed to follow the desired trajectory $\boldsymbol{x}_\tau$ by generating a control input with sampling time t_s, $\boldsymbol{u} = g(\boldsymbol{x}, \boldsymbol{x}_\tau)$. The goal is to verify that the plant (Unsafe) mode is never reached during the duration T of the mission, i.e., that the NN never outputs safe when the plant is unsafe.

dynamics. H contains the union of modes and states of H_q and H_S. H starts in the initial mode of H_S and adds a transition from the last mode of H_S to the initial mode of H_q, at which point the H_q execution begins. The goal is to verify that H does not enter the plant (Unsafe) mode when $S(\boldsymbol{x}_0) = \text{Safe}$.

Given a hybrid system description of the closed-loop system, one could use a tool such as Flow* to verify the system's safety. Note that, since hybrid system verification is undecidable in general, the typical approach used in these tools is to overapproximate the reachable sets. If the overapproximation does not contain any unsafe states, the system is safe. If the overapproximation contains safe and unsafe states, the outcome is unknown, since the unsafe states could be spurious; i.e., they do not exist in the true reachable set but only in the overapproximated one. Finally, if all states are unsafe, the system is unsafe. Various shapes have been explored to overapproximate the reachable sets, including polytopes, ellipsoids, and hyperrectangles. Flow* uses a Taylor model approximation, which is a Taylor series approximation with worst case error bounds. Taylor models scale well when used with interval analysis and are shown to have a low approximation error for a large class of nonlinear systems [2].

NN Retraining

At the end of the verification, Verisig specifies the regions in which 1) the NN is safe to be used (i.e., the plant is not unsafe when the NN output is safe), 2) the NN is not safe to be used (i.e., the plant is unsafe when the NN output is safe), and 3) the system is not able to make a decision due to the approximation errors introduced in the hybrid system RA during verification (i.e., the NN output is "unknown"). In case there are regions in which the NN is not safe to be used or Verisig cannot decide, the NN needs to be retrained. The output of Verisig can be leveraged to retrain the NN in several ways. One is to collect more data around those regions where Verisig is not able to make a decision, followed by NN retraining. An increased density of data around previously untrained regions may help with the verification.

However, how much data needs to be collected in those regions is not known a priori and is hard to predict, so the process could require multiple iterations of data collection and retraining. In addition, collecting new data to improve the training set may not always be possible. Instead, we propose adding points from the unsafe/unknown regions obtained from the Verisig output to the existing training set, marking them with unsafe labels, and finally retraining the NN. By retraining the NN with more unsafe points, a more conservative version is obtained in which unsafe regions are inflated, helping with the verification process. This retraining process is repeated until the NN is verified.

Case Studies

As a proof of concept, our verified safe monitoring and planning approach is applied to two case studies of quadrotor motion planning: 1) a pickup/drop-off mission and 2) a navigation operation in a cluttered environment. Both case studies use similar NNs to predict whether the vehicle trajectory will be safe or unsafe (and thus require offline training and verification). At runtime, the verified NN is used for different purposes. The pickup/drop-off task requires the UAV to go from one side of a static environment to the other, resembling operations that could happen inside a warehouse or factory. The UAV makes decisions about the safety of the planned trajectory based on the NN results and replans by adjusting the obstacle-avoidance distance until it finds a longer but safe course to its goal position. In the latter case study, the UAV is tasked with navigating a previously unknown cluttered area. Training is executed in a smaller environment with only one obstacle, acting as a primitive scenario that can appear and be composed multiple times at runtime. Training in a primitive environment enables the NN and verification to be generalized to different settings with the same type of obstacles located in previously unknown positions. Replanning here is executed by querying different waypoints along the path to the goal until the NN outputs a safe decision. In both case studies, we use the same vehicle, controller, planner, and disturbances, whose models are briefly summarized in the following section.

System Models

Quadrotor UAV and Controller Model

A quadrotor can be modeled using the following simplified sixth-order state vector $\boldsymbol{x} = [x \;\; y \;\; z \;\; v_x \;\; v_y \;\; v_z]^\top$, where x, y, and z are the world frame positions and v_x, v_y, and v_z are the world frame velocities. The quadrotor dynamics can be defined as $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d})$, where $\boldsymbol{u} = [F \;\; \phi \;\; \theta]^\top$ is the input vector with thrust, roll, and pitch commands and $\boldsymbol{d} = [d_x \;\; d_y \;\; d_z]^\top$ is the external disturbance vector. The dynamics can be described as

$$[\dot{x} \;\; \dot{y} \;\; \dot{z}]^\top = [v_x \;\; v_y \;\; v_z]^\top$$

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} g\theta \\ -g\phi \\ \frac{F}{m} - g \end{bmatrix} + k_d \begin{bmatrix} d_x - v_x \\ d_y - v_y \\ d_z - v_z \end{bmatrix},$$

where m is the mass of the quadrotor, g is gravity, and k_d is the drag coefficient. It should be noted that, in this article, a simplified quadrotor UAV model is used to alleviate the verification problem. Validating a high-fidelity model [13] is left for future work. To generate the necessary roll, pitch, and thrust inputs to follow the desired trajectory, a cascaded set of PID controllers is used [14].

Disturbance Model

The external disturbance considered in this article is bounded in magnitude: $\|\boldsymbol{d}\| \leq D_{\max}$, $\forall \boldsymbol{d} \in \mathcal{D}$, where $D_{\max}$ is the upper bound to the disturbance magnitude and $\mathcal{D}$ is the set of all possible disturbances. Here, we assume that the online disturbance is unknown but constant through time. This is a reasonable assumption, as wind disturbance generally follows a Brownian motion and does not change erratically during short periods of time [15], [16].

Trajectory Planning

Obstacle-avoidance trajectories are computed using a simple geometric approach. Specifically, if an obstruction is present along the way to the goal position, a waypoint is added to the path at a specified avoidance distance r_a away from the obstacle center. A course is finally generated to visit all waypoints on the path by using a minimum jerk trajectory generation [17]. It should be noted that we use this path-planning method due to its simplicity in implementation in simulations and experiments; however, the overall proposed framework is independent from the choice of path-planning approach.

Pickup/Drop-Off Task

The first case study that we present in this article is a pickup/drop-off task, an operation that is commonly used in factory applications where a vehicle moves back and forth between a warehouse and a workstation. The environment has a designated pickup area (warehouse) and drop-off position (workstation), with obstacles at known locations in between. The vehicle is tasked to move from a point inside the pickup area to the drop-off location. Once it reaches the drop-off site, it can move back to a new point in the pickup area. To complete its mission safely, the UAV needs to decide whether the planned trajectory, parametrized by the initial position $\boldsymbol{p}_0$, final goal $\boldsymbol{p}_g$, and avoidance distance r_a, is safe and if not, replanning is required. In this case, replanning is executed by adapting r_a.

To train an NN to make safety decisions in this scenario, two sets of trajectories with different avoidance distances are generated and labeled using RA: one set links a rich set of initial positions in the pickup area to the drop-off position, and the other connects the drop-off position to a rich set of final positions in the pickup area. The NN queries the initial and final positions and the avoidance distances; if they are unsafe, it checks a larger avoidance distance until it outputs a safe decision.

Simulation-Based Reachability

In this case study, we use a simulation-based RA. During the offline stage, we run each training trajectory under the worst-case scenario which, in our example, is the largest possible disturbance attainable in the environment. Under this condition, for a given trajectory $\boldsymbol{p}_\tau$, the maximum deviation d_m is calculated as follows:

$$d_m = \max_{\boldsymbol{d}\in\mathcal{D}} \max_{t\in[0,T]} \min_{\xi\in[0,T]} \|\boldsymbol{p}_{\boldsymbol{d}}(t) - \boldsymbol{p}_\tau(\xi)\|, \tag{2}$$

where $\boldsymbol{p}_{\boldsymbol{d}}$ is the position of the vehicle under disturbance $\boldsymbol{d}$. Here, d_m is used as an upper bound for the actual deviation from the trajectory, and it is conservative since it is the maximum deviation measured through the entire trajectory. The position-reachable sets are then generated as follows:

$$\boldsymbol{R}(\boldsymbol{p}_\tau, t) = \{\boldsymbol{p}(t) : \|\boldsymbol{p}(t) - \boldsymbol{p}_\tau(t)\| \leq d_m\}. \tag{3}$$

After generating the reachable sets, the trajectory is labeled safe or unsafe according to (1). In Figure 3, we show the reachable sets of two sample trajectories.

Offline Training

The environment has a designated rectangular pickup area that is limited between [0.0, 1.3] m in the x-axis and [−1.0, 1.0] m in the y-axis. The drop-off point is located at $\boldsymbol{p}_g = [4.0, 0.0]$ m. There are two obstacles between the pickup area and drop-off location, positioned at $\boldsymbol{p}_{o1} = [2.0, 0.1]$ m and $\boldsymbol{p}_{o2} = [3.0, -0.1]$ m. For training, 294 points are uniformly distributed in the pickup area and used as the initial and final positions. For trajectory generation, seven different avoid distances are considered: $r_a \in \{0.3,\ 0.35, 0.4, 0.45, 0.5, 0.6, 0.7\}$ m.

Two NNs were trained: one for the drop-off operation and the other for the pickup task. To implement the NNs, we chose Keras (https://keras.io), a deep-learning library capable of running on top of Tensorflow (https://tensorflow.org) through a set of application programming interfaces written in Python. For all layers (input, hidden, and output), we use a sigmoid activation function. The NN is composed of three input nodes (the x-y initial position and avoidance distance pair), one hidden layer of 40 nodes, and one output that determines whether the label is safe or unsafe. We trained two different NNs, one

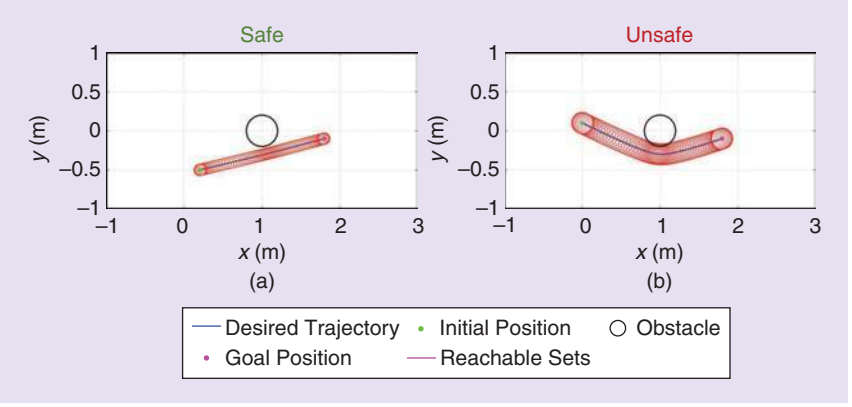


Figure 3. The reachable sets for two sample trajectories. (a) A safe trajectory. (b) An unsafe trajectory.

for each subtask (drop-off and pickup). The training results showed 0% FPs and roughly 5.2% FNs for the first NN and 0% FPs and approximately 1.2% FNs for the second one.

In Figure 4, the initial positions in the training set for the drop-off and pickup operations are presented with their respective labels and the NN results inside the proposed workspace. Each subfigure denotes a different avoidance

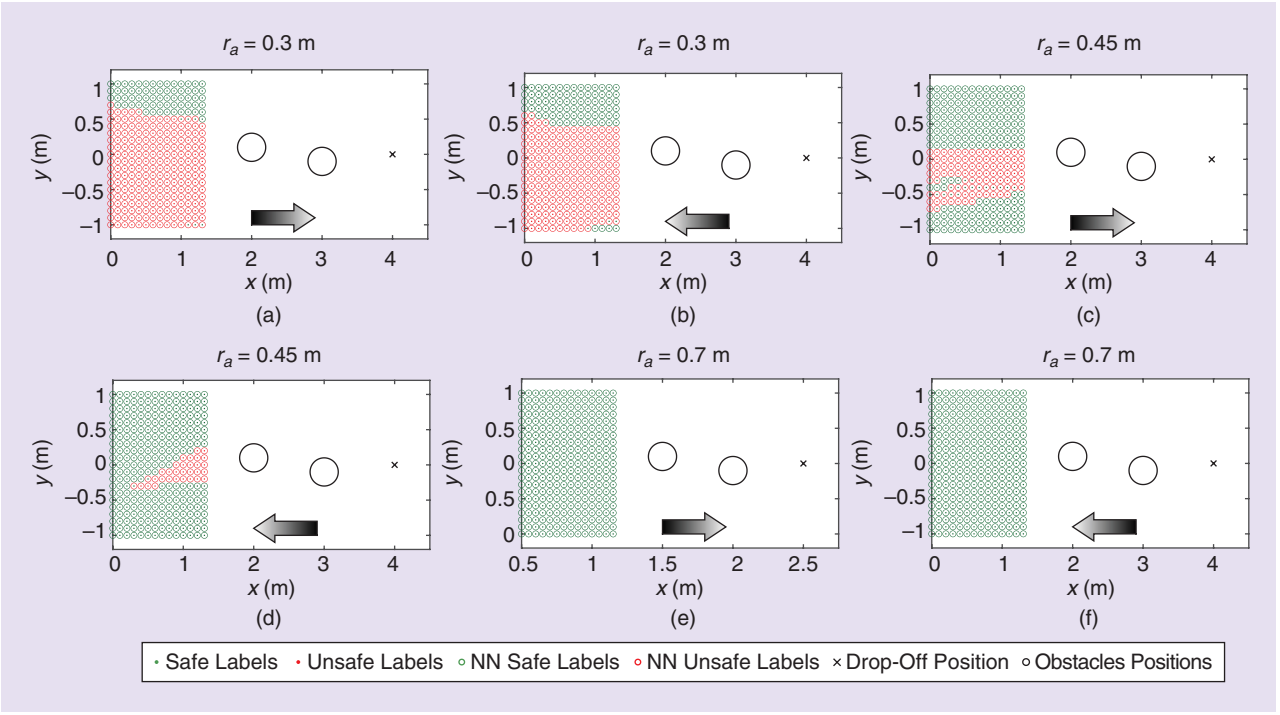


Figure 4. Safety maps for the initial and final positions in training sets with different avoidance distances. The (a) drop-off task and (b) pickup mission for $r_a = 0.3$ m. The (c) drop-off task and (d) pickup mission for $r_a = 0.45$ m. The (e) drop-off task and (f) pickup mission for $r_a = 0.7$ m.

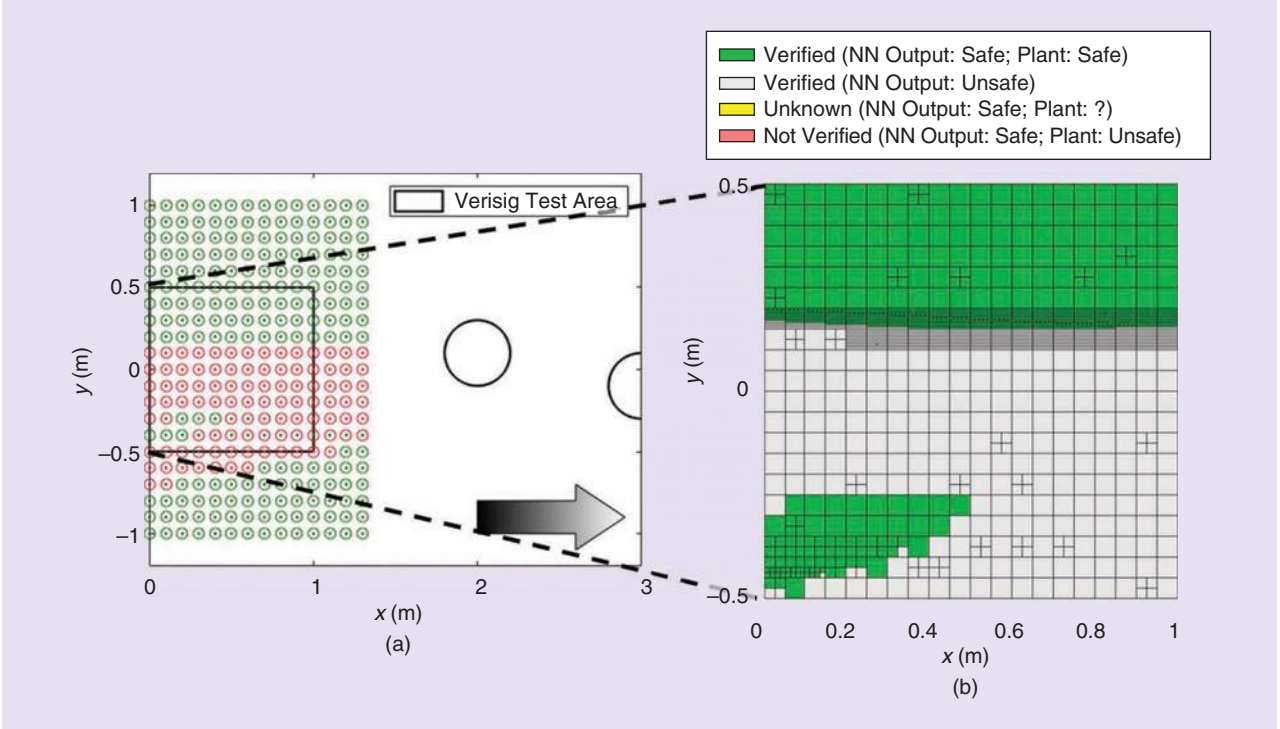


Figure 5. (a) The pickup task with avoid distance $r_a = 0.45$ m in Figure 4(c). (b) The initial set was divided into small subsets and verified. No unsafe sets were obtained.

distance marked on top of the figure. Due to space constraints, we show examples of data for only three avoidance distance values: $r_a = 0.3$ m, $r_a = 0.45$ m, and $r_a = 0.7$ m. Arrows inside the workspace indicate the direction of motion of the vehicle. Inside each subfigure, the green (red) dots represent initial positions from which the trajectories to the goal are labeled safe (unsafe) using reachability analysis. The green (red) circles around the dots denote the decisions of the NN on the same training points. As can be noticed and as expected, when the avoidance distance increases, the number of safe initial positions also increases in both missions because the distance between the desired trajectories and the obstacles becomes larger. Therefore, increasing the avoidance distance improves safety; however, the routes become longer, which generally is not desirable due to energy concerns.

Verification Results

Verification was performed with the methods in the "Verification" section. Here, we present the results for the second drop-off task shown in Figure 4, namely, the case in which the UAV starts in the set $x_0 \in [0, 1]$, $y_0 \in [-0.5, 0.5]$ and aims to reach the goal at [4, 0] m, with $r_a = 0.45$ m and disturbances d_x, $d_y \in \{-0.1, 0.1\}$ m/s.

The verification results are presented in Figure 5. We divided the initial set into smaller subsets and verified each one separately to keep the approximation error in Flow* small enough. The size of these subsets [i.e., the 5-cm boxes in Figure 5(b)] was chosen after some preliminary testing; these subsets were large enough to verify the majority of the initial set with a small approximation error. Some subsets were further refined when an instance resulted in an error that was too large. Refinements were necessary at the NN's decision boundary as well as for sets that triggered multiple if-cases in the planner (e.g., when planning around multiple obstacles). The verification was performed using Amazon

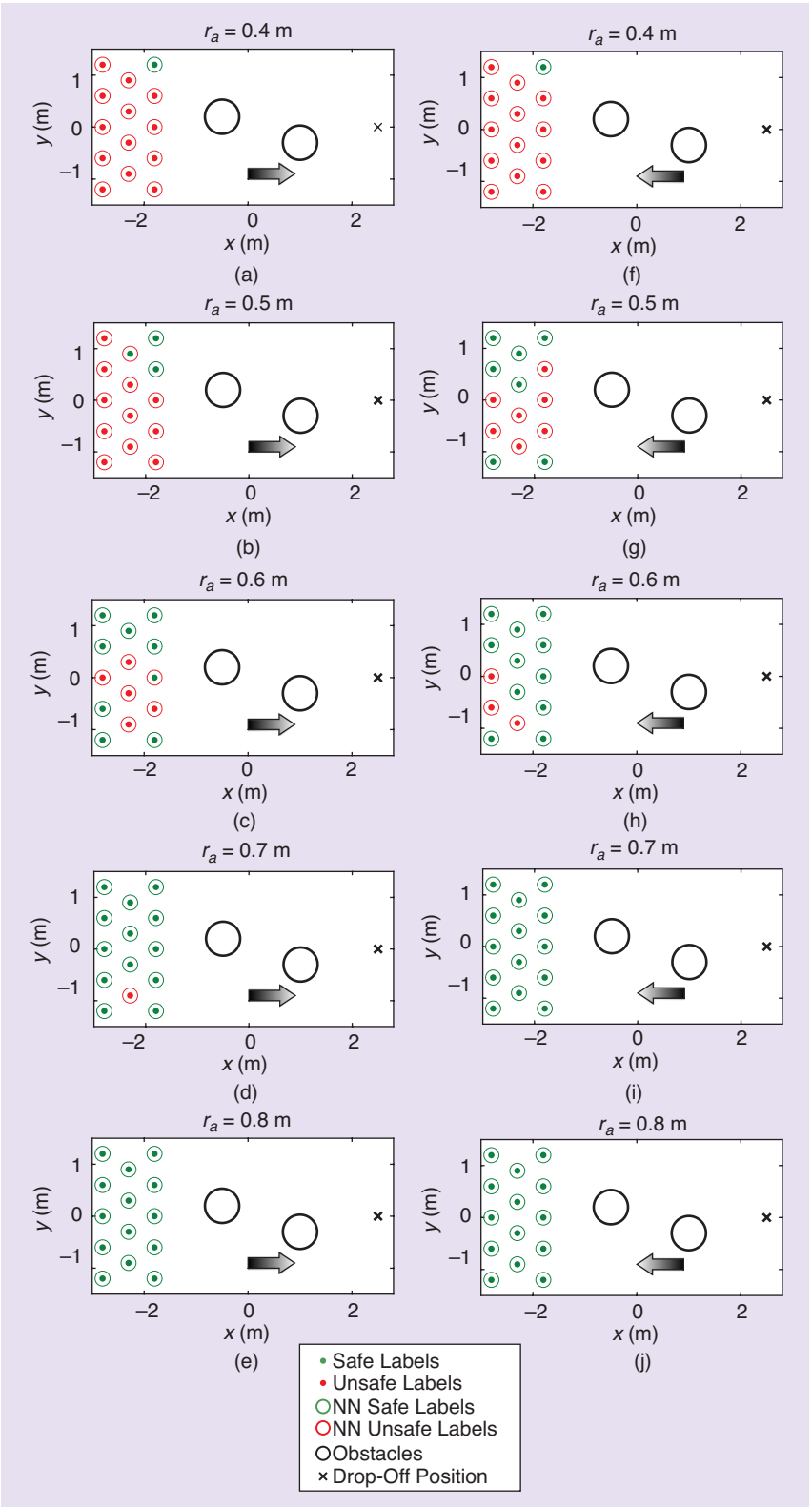


Figure 6. The safe and unsafe training and NN results for the pickup and drop-off tasks in the area where experiments were performed. An FP (red dot inside a green circle) corresponds to an unsafe label (red dot) with a safe NN decision (green circle), while an FN (green dot inside a red circle) is a safe label (green dot) marked as unsafe (red circle). The number of FPs is zero for both NNs, and there are two and zero FNs for the first and second NN, respectively. The drop-off tasks where (a) $r_a = 0.4$ m, (b) $r_a = 0.5$ m, (c) $r_a = 0.6$ m, (d) $r_a = 0.7$ m, and (e) $r_a = 0.8$ m. The pickup tasks where (f) $r_a = 0.4$ m, (g) $r_a = 0.5$ m, (h) $r_a = 0.6$ m, (i) $r_a = 0.7$ m, and (j) $r_a = 0.8$ m.

Web Services (https://aws.amazon.com). Each subset took roughly an hour to verify, although some took longer due to branching introduced by if-cases in the planner.

Figure 5 closely matches the corresponding graph in Figure 4(c). We confirmed that the NN was, indeed, conservative so that no unsafe events occurred when its output was safe. The same procedure can be used for all other cases. Note that we verified the safety for only the first NN as a proof of concept; the procedure for the second NN would be exactly the same.

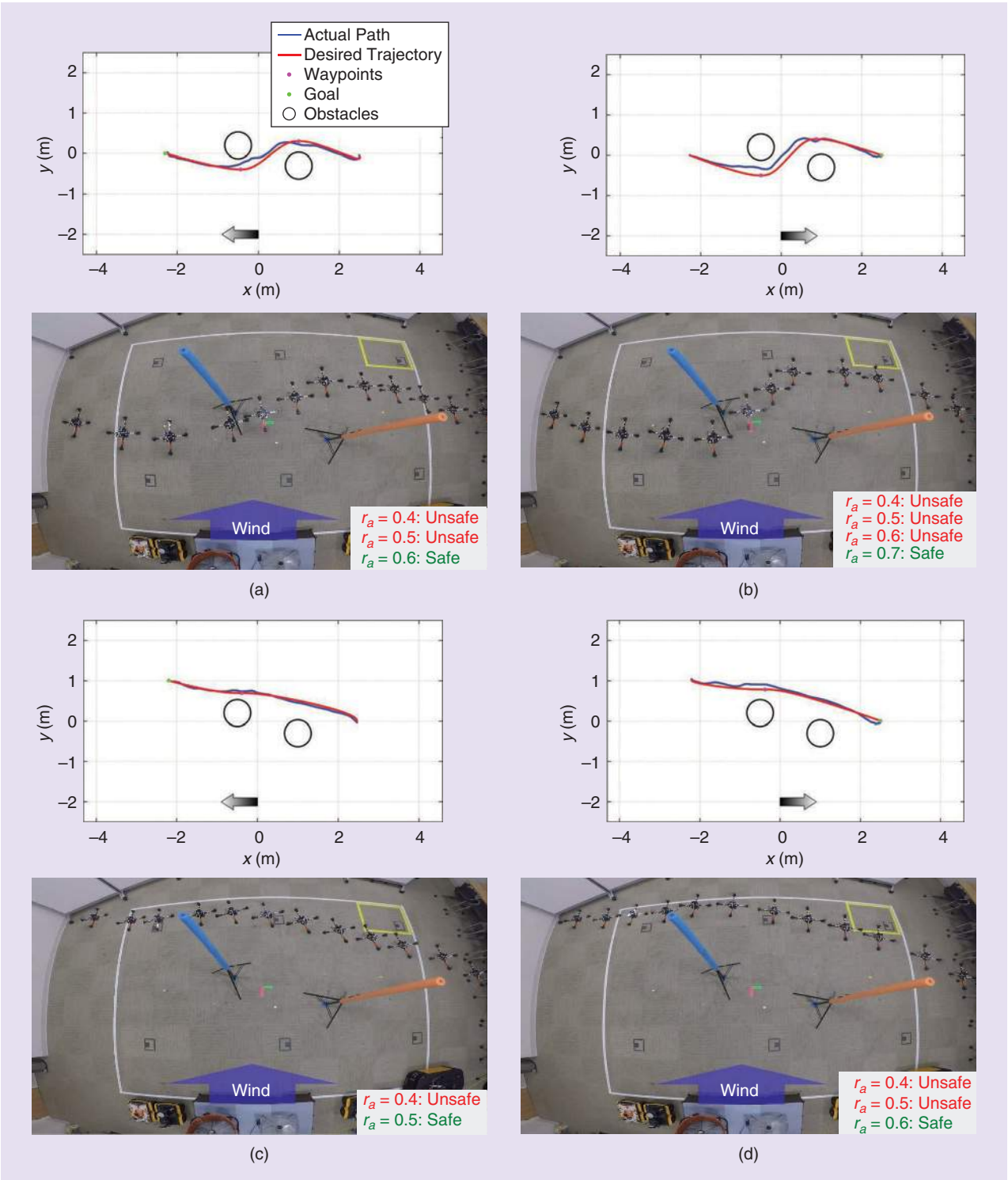


Figure 7. The experimental results in which the trained NN was used to make safety decisions and replan accordingly. The desired versus the actual trajectories are presented in the top row of subfigures, with the relative snapshots from the experiments in the bottom row. (a) The round 1 pickup task. (b) The round 1 drop-off task. (c) The round 2 pickup task. (d) The round 2 drop-off task.

Experimental Results

To test our framework, we designed a similar pickup/drop-off scenario (see Figure 6) in which the quadrotor was tasked to visit different points in the pickup area and return to the drop-off location every round while avoiding two obstacles (Figure 7). Real flights were performed with an AscTec Hummingbird quadrotor controlled through the Robot Operating System. A Vicon motion capture system was used to track the position of the quadrotor and provide ground truth position information. Two industrial fans blew wind in the middle of the area, creating a disturbance toward the obstacles. To generate safe and unsafe labels for the real scenario, 14 positions equidistant from one another in the pickup area were considered. For every avoidance distance $r_a \in \{0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8\}$ m, trajectories were generated from each starting position to the drop-off goal, and vice-versa, and safety decisions about these routes were made using a similar approach explained in the "Reachability Analysis" section. A trajectory was labeled unsafe if the maximum deviation from the desired course became larger than the distance between the obstacle and the desired route at any point along the path. We trained two NNs, one for each subtask. In Figure 6, the safety decisions are shown when the training set is provided as input to the NN.

During testing, we exploited the trained NN by executing multiple passes back and forth between different points in the pickup area through an unknown disturbance, generated in the middle of the area, that could push the quadrotor toward the obstacles. Throughout the experiment, the vehicle was tasked to navigate using the smallest avoid distance $r_a = 0.4$ m, if possible. As expected, the NN generated unsafe outputs for some of the points. Consequently, r_a was increased by 0.1-m increments until a safe decision was returned by the NN before sending the vehicle to the goal. The computation time for the NN to make a safety decision was on the order of milliseconds and constant for all trajectories, making it suitable for online replanning operations. Conversely, the simulation-based reachability used during training is not suitable at runtime, as its computation time increases linearly with the duration of the trajectory. Figure 7 shows the sequence of snapshots, the decisions of the NN, and the comparison between the desired and actual trajectories (upper rows) for two rounds of the operation.

Finally, we ran an experiment in which we generated trajectories with the minimum avoid distance $r_a = 0.4$ m, using the wind disturbance from the previous experiment and disregarding the decision from the NN. During the first round, which was predicted to be unsafe with $r_a = 0.4$ m, the quadrotor collided with an obstacle (Figure 8), confirming the unsafe decision predicted by the NN.

Navigation in Cluttered Environments

The other case study that we demonstrate in this article is a navigation operation in a cluttered environment, such as a heavily forested area. A UAV was tasked to reach a goal position in an area in which obstacles were scattered in a priori unknown locations discoverable at runtime as the system moved toward the final objective. To plan safety-guaranteed trajectories, we considered a smaller environment with only one obstacle in the center and trained and verified an NN to predict the safety of the routes within this smaller region. The trained and verified primitive space could then be fitted and composed multiple times at runtime to assess safety in larger regions with more obstacles.

Throughout a mission, every time the UAV encounters an obstacle along its path, it selects an intermediate goal point around the obstruction and queries the trained NN about the safety of the trajectory to the selected target. If unsafe, a new intermediate goal is queried until the output of the NN is safe. This procedure is repeated multiple times for each obstacle along the path until the vehicle reaches the final target position.

Offline Training

To train the NN for this operation, we used a small, box-shaped workspace with one obstacle in the center, as shown in Figure 9. We picked 44 initial and 44 final positions uniformly distributed across the start (to the left of the obstacle) and goal (to the right of the obstacle) regions. The safety of the trajectories generated from those 1,936 start–goal position pairs

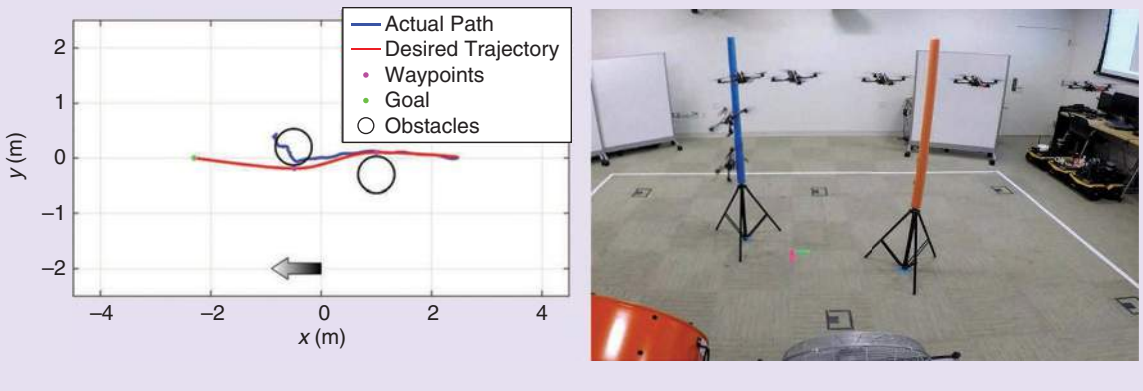


Figure 8. The trajectory followed by the quadrotor for the first pickup task in Figure 7(a), with $r_a = 0.4$ m and marked unsafe, resulting in a crash.

was decided offline using Flow* RA [2]. On average, it took roughly 5 min for Flow* to make a safety decision for one start–goal position pair under bounded disturbance conditions $d_x, d_y \in [-0.4, 0.4]$ m/s, reinforcing the fact that RA is expensive to perform at runtime.

Using this set of initial–final position pairs, an NN was trained to predict the safety of an untrained pair of initial–final goals. The NN was composed of four input nodes (the *x*-*y* initial position and *x*-*y* goal position pair), one hidden layer of 40 nodes, and one output, which determined whether the label was safe or unsafe. The NN performed with 0% FPs and roughly 0.2% FNs. In Figure 9, we present examples of labels and NN decisions for trajectories starting from three different initial positions to all of the final goals in the training set. Similar to the previous case, a green (red) dot represents a final position in which the trajectory was labeled safe (unsafe) from a given starting point, and a green (red) circle represents the NN decision for the same point.

NN Verification

Similar to the previous case study, since the results of an NN could be erroneous, we use Verisig to verify the safety predictions obtained by the trained NN. As a proof of concept, to demonstrate the procedures explained in the "Verification" and "NN Retraining" sections, the results of the NN from a single initial position to all goal positions in the primitive environment are verified. However, the same NN verification and retraining procedure can be performed for all possible initial and final regions. The training data for this case are presented in Figure 9(a), while Figure 10(a) displays the results of the verification. The gray shaded regions in Figure 10(a) represent areas where the NN outputs unsafe and where Verisig concurs without performing the whole verification, since the NN output is already unsafe and thus, in the worst-case scenario, conservative. Green regions represent areas where the NN outputs safe and where Verisig verifies that the plant is safe, too. In the yellow regions, the NN outputs safe, but Verisig cannot decide whether the plant is safe or not.

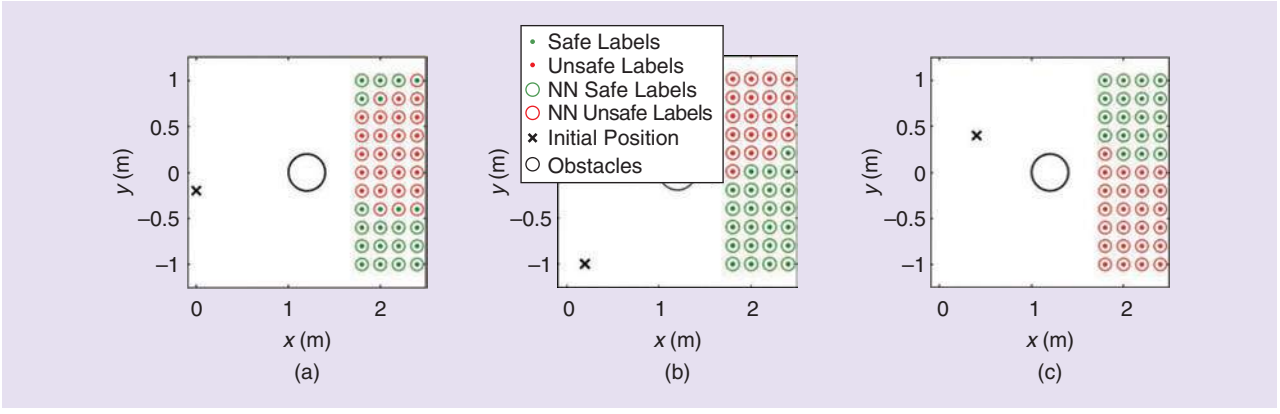


Figure 9. The safe and unsafe trained final goals and NN decisions from various initial positions. (a) Initial position [0.0, −0.2] m. (b) Initial position [0.2, −1.0] m. (c) Initial position [0.4, 0.4] m.

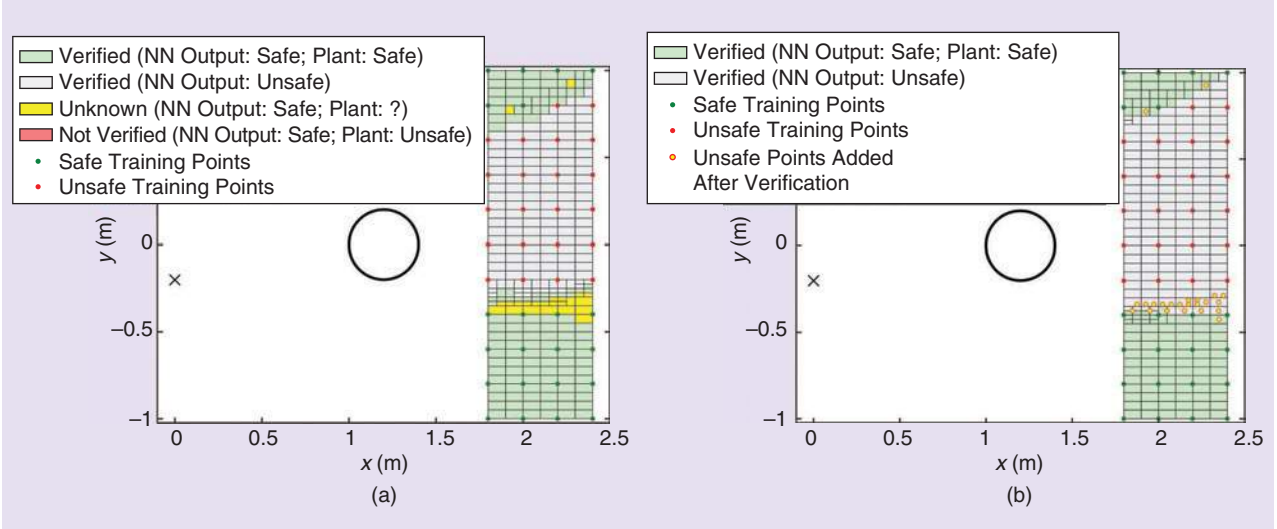


Figure 10. The NN verification results. (a) The verification of the NN trained with the original data set. (b) The verification of the retrained NN with the conservative data set.

Since the NN is not completely verified due to the undecided regions, points from these regions are added to the existing training set, and the NN is retrained as explained in the "NN Retraining" section. These points are shown by yellow dots in Figure 10(b). After retraining with the addition of these points, the NN performed with 0% FPs and roughly 1.9% FNs, which was expected since the NN was trained to be more conservative. Figure 10(b) gives the verification results with the retrained NN, and, as can be noted, the entire goal region was verified by Verisig.

Simulation Results

In this simulation, the trained NN is used to make decisions about the safety of a trajectory in the cluttered environment presented in Figure 11. First, the primitive box space used for training is superimposed around each obstacle (the square areas inside Figure 11) in such a way that there is only one obstacle in each primitive space; otherwise, the results of the NN may not be reliable due to the difference from the training conditions.

Once the mission is started, the quadrotor picks the closest point to the final goal inside the target region of the first primitive as an intermediate location. The NN makes a decision about the safety of this intermediate goal from the current position of the UAV. If the decision is deemed safe, the UAV moves to this intermediate position. If the NN decision is unsafe, it searches for a safe goal location in the target region of the current primitive. This search is performed by randomly querying points in the target area of the primitive, starting from a closer proximity of the initially selected goal and radially enlarging the search area if no safe goals are obtained immediately. Note that, to deploy such an approach, the NN must contain at least one safe point in the goal area of the primitive. This process continues until the UAV reaches its final destination.

Figure 11(a) shows the trajectory followed by the quadrotor in this environment. The queried intermediate goal positions found by the NN to be unsafe are shown by red dots in the goal regions in the primitives areas, while the safe intermediate goals traveled to by the UAV are shown by cyan dots. Wind disturbance is present throughout the entire mission, blowing in the northeast direction, as shown by the orange arrow inside the figures. The UAV is able to complete the mission without any collision. In Figure 11(b), we repeat the same case without using the NN decisions; here, the UAV moves to the intermediate goal positions even if the NN decision is unsafe. As expected, there are instances where the UAV crashes or gets very close to the obstacles. These results confirm that NNs can be used to monitor safety properties of motion planning operations using the composition of smaller verified regions into larger, more complex, and untrained environments.

Experimental Results

The same case study was also performed in experiments following a similar setup as the one presented in the previous example. NN training was done on a smaller primitive environment with one obstacle by performing 100 flights with our aerial testbed under a wind disturbance blowing in the $+y$ direction. An initial–final position pair was labeled unsafe if the reachable sets generated using the approach explained in the "Simulation-Based Reachability" section collided with

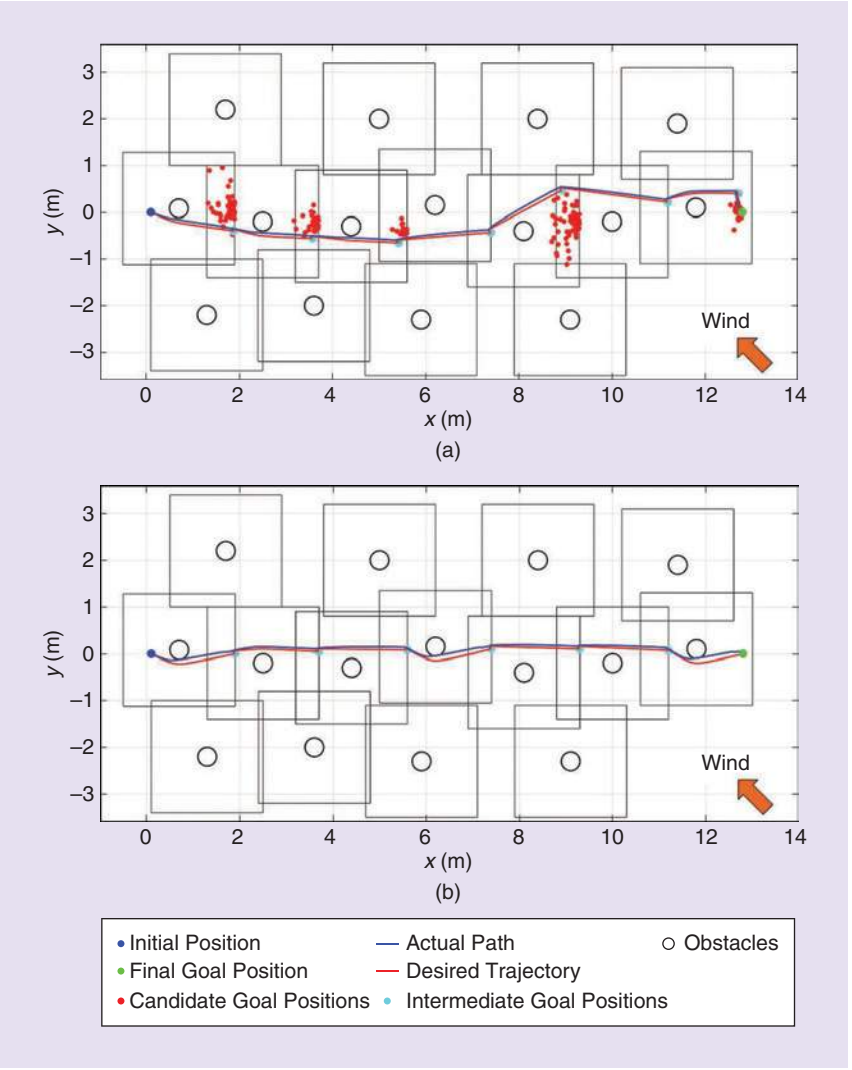


Figure 11. The navigation simulation in a cluttered environment. (a) The safe replanning using NN decisions. (b) The planning without using NN decisions.

the obstacle. Using these safe-/unsafe-labeled initial–final position pairs, a conservative NN was trained to make safety decisions about untrained initial–final position pairs at runtime. Figure 12 shows the safe and unsafe initial–final position pairs and corresponding NN decisions, using the same color coding as the previous cases.

The safe navigation approach was validated in an environment with three obstacles and two fans blowing air in the $+y$ direction, as seen in Figure 13. Obstacles in Figure 13 are represented as circles having a radius equal to the actual obstacle size plus the size of the UAV. In Figure 13(a), the intermediate goal positions queried by the NN are shown using the same color code as the simulation results. The UAV queried 20 points to avoid the first obstruction until it found a safe intermediate goal and repeated this operation until it reached its final destination. Using this NN-based framework, it takes a few milliseconds to search for a safe target. The experimental results of the proposed approach are compared with the ones in which the NN is not utilized to make safety decisions about intermediate goal positions. As expected, in Figure 13(b), the UAV fails to complete its mission safely, as it crashes and gets very close to the other obstacles. These experiments were executed without the obstructions, which are overlaid in the figure for reference.

Conclusions

The recent interest in LECs and their rapid introduction in our society, in particular in autonomous systems technologies, reveal considerable potential and benefits, especially in terms of computation overhead and decision-making applications. However, new challenges have emerged due to the lack of models and the complete reliance on data that do not provide the assurance and guarantees necessary for their deployment in safety-critical operations.

The framework for verification discussed in this article moves toward this assurance-driven design of LECs. However, many challenges remain that need to be addressed to fully integrate these technologies in safety-critical operations. In this section, we provide an overview of these challenges and offer some possible solutions and directions for future research on assured runtime monitoring.

Discussion

A first challenge typical of LECs centers around how to select the appropriate training set. The accuracy of any machine learning technique depends largely on the type, amount, and heterogeneity of the data used during the training phase. A poorly trained NN results in poor performance, leading to unsafe or overconservative behavior of autonomous systems. Similarly, overfitting can introduce overhead and poor prediction. To deal with these issues, it is possible to perform sensitivity analysis [18] and nonconformity analysis [19] on the system prior to training to better interpret and select data. It is also possible to leverage knowledge about the system dynamics to perform verification before the deployment of the LEC, as suggested in this article.

Verification provides safety guarantees for the outputs of such LECs; however, it does not provide any robustness guarantees against changes between training and testing conditions. Currently, state-of-the-art verification techniques, beyond machine learning applications [20], require knowledge about system model and bounded conditions. However, no guarantees can be provided if, for example, the disturbance acting on a system is above or below the bounds for which training was performed. This challenge becomes even more evident when dealing with real systems. In fact, typically (including in our setup in this article), verification considers a specific model that may and will change from the actual model of a real system. Even precise system-identification techniques are not able to

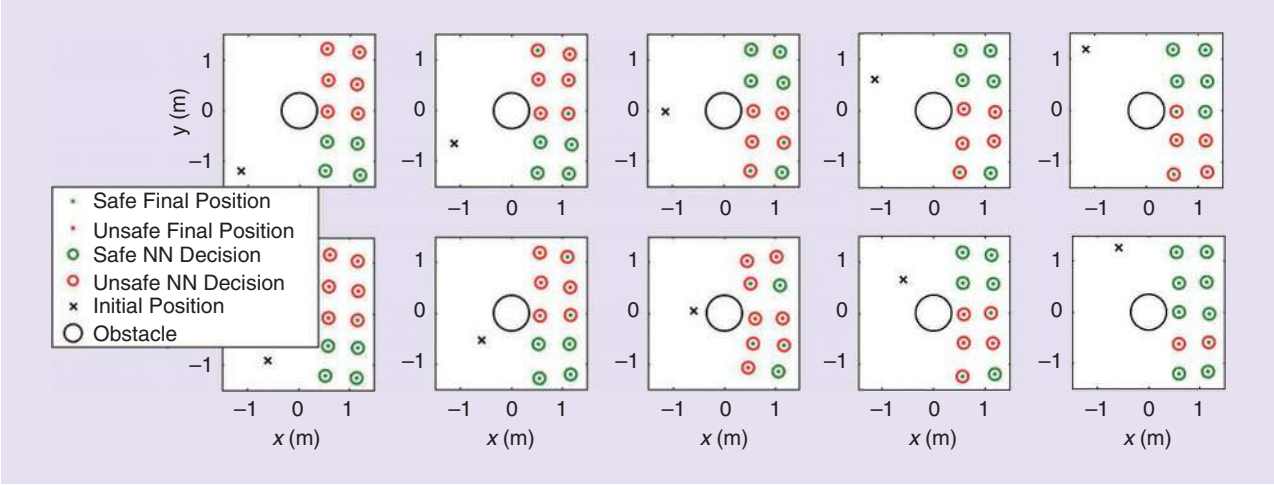


Figure 12. The safe and unsafe final positions and NN results from different initial locations in the experimental setting. The NN here is composed of four input nodes (an x-y initial position and x-y final position pair), one hidden layer of 40 nodes, and one output for the safety decision. The NN performed with 0% FPs and roughly 16% FNs.

provide fully accurate models and often rely on specific operating conditions.

One way to provide verification for systems with model uncertainties is to consider a conservative abstracted model and verify that the real system is equivalent to or safer than this abstraction. Building on the intuition in [21], if the system performance and behavior are verified to be closer to the desired behavior than the abstraction, verifying the abstraction also verifies the real system. Note that, since these conservative abstractions capture the worst-case behavior of the system, failing to verify the abstraction may lead to the belief that the actual system is not safe, which may not necessarily be true but safe.

Training and verification operations require heavy computation time; although this is a minor problem since these operations occur offline, it is still a concern, especially when dealing with high-order dynamical models and large unknowns in the system. Several services, such as Amazon Web Services (used in this article), Microsoft Azure (https://azure.microsoft.com), and Google Cloud Platform (https://cloud.google.com), are available and projected to become faster and more accessible in the future.

Lastly, we note that, similar to the approach presented in the second case study in this article, although verification is done for a static model, the approach that we presented could be generalized to other settings where verifying a subset of the state space can be sufficient for use compositionally to check safety properties about larger spaces.

Future Work

The framework proposed in this article enables fast and assured predictive and proactive monitoring of autonomous systems operations in cluttered and uncertain environments at runtime. NNs are leveraged to make decisions about the safety of planned trajectories at runtime and perform replanning accordingly. RA is used during the training phase and for verification purposes, bypassing its usage at runtime. In this way, most of the computation burden is limited to offline operations, leaving the fast decision-making and replanning tasks for the online application.

The applicability of the proposed framework was demonstrated in two case studies, and the designed NNs were verified using our recent Verisig tool to produce decisions that never lead to unsafe states. Safety was the main concern in this article. Possible future directions include adding energy constraints while designing safe operations, developing robust NNs to deal with changes in environments, and, as discussed in the previous section, diving deeper into the problem of verifying real systems.

Acknowledgment

This material is based upon work supported by the Air Force Research Laboratory (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under the Assured Autonomy program, Contract FA8750-18-C-0090. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not

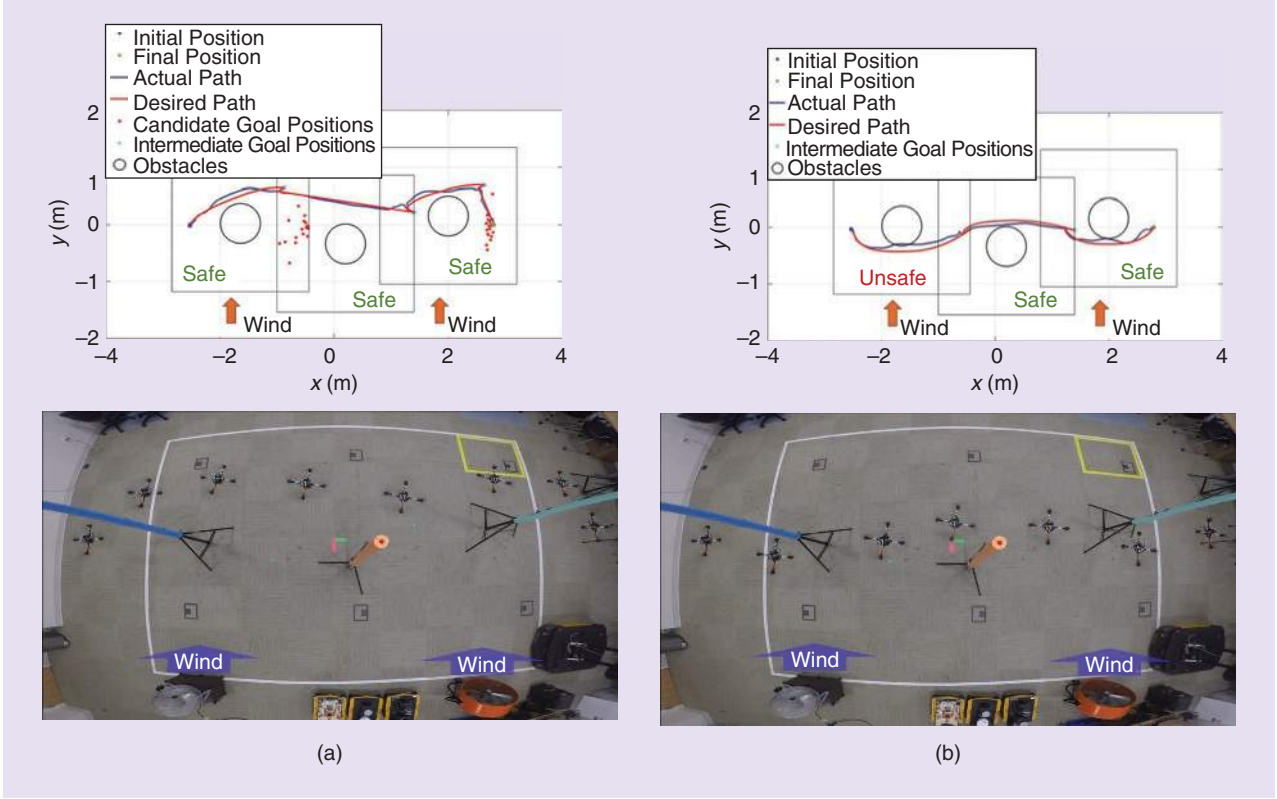


Figure 13. (a) The experimental results of the navigation of the quadrotor using the trained NN to make safety decisions and replan. (b) The unsafe navigation in which NN decisions were disregarded. The lower row shows the experiments relative to the upper row.

necessarily reflect the views of AFRL, DARPA, the Department of Defense, or the U.S. Government.

References

[1] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," in *Proc. IEEE 56th Annu. Conf. Decision and Control (CDC)*, Dec. 2017, pp. 2242–2253. doi: 10.1109/CDC.2017.8263977.

[2] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *Computer Aided Verification*, N. Sharygina and H. Veith, Eds. Berlin: Springer-Verlag, 2013, pp. 258–263.

[3] S. Kong, S. Gao, W. Chen, and E. Clarke, "dReach: δ-Reachability analysis for hybrid systems," in *Tools and Algorithms for the Construction and Analysis of Systems*, C. Baier and C. Tinelli, Eds. Berlin: Springer-Verlag, 2015, pp. 200–205.

[4] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: Verifying safety properties of hybrid systems with neural network controllers," *Proc. 22nd Int. Conf. Hybrid Systems: Computation and Control*, 2019, pp. 169–178. doi: 10.1145/3302504.3311806.

[5] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *Proc. ACM 18th Int. Conf. Hybrid Systems: Computation and Control*, 2015, pp. 11–20. doi: 10.1145/2728606.2728612.

[6] V. R. Royo, D. Fridovich-Keil, S. L. Herbert, and C. J. Tomlin, Classification-based approximate reachability with guarantees applied to safe trajectory tracking. 2018. [Online]. Available: http://arxiv.org/abs/1803.03237

[7] M. Chen, S. Herbert, and C. J. Tomlin, "Exact and efficient Hamilton-Jacobi guaranteed safety analysis via system decomposition," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2017, pp. 87–92. doi: 10.1109/ICRA.2017.7989015.

[8] B. Djeridane and J. Lygeros, "Neural approximation of PDE solutions: An application to reachability computations," in *Proc. 45th IEEE Conf. Decision and Control*, Dec. 2006, pp. 3034–3039. doi: 10.1109/CDC.2006.377184.

[9] S. Singh, A. Majumdar, J. J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2017, pp. 5883–5890. doi: 10.1109/ICRA.2017.7989693.

[10] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *Int. J. Robot. Res.*, vol. 36, no. 8, pp. 947–982, 2017. doi: 10.1177/0278364917712421.

[11] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *Proc. 2017 IEEE 56th Annu. Conf. Decision and Control (CDC)*, Dec, 2017, pp. 1517–1522.

[12] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots. 2018. [Online]. Available: arXiv:1809.06746.

[13] E. Yel, T. X. Lin, and N. Bezzo, "Self-triggered adaptive planning and scheduling of UAV operations," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Brisbane, Australia, May 21–25, 2018, pp. 7518–7524. doi: 10.1109/ICRA.2018.8463205.

[14] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro-UAV testbed," *IEEE Robot. Autom. Mag.*, vol. 17, no. 3, pp. 56–65, Sept. 2010. doi: 10.1109/MRA.2010.937855.

[15] E. van Doorn, B. Dhruva, K. R. Sreenivasan, and V. Cassella, "Statistics of wind direction and its increments," *Phys. Fluids*, vol. 12, no. 6, pp. 1529–1534, 2000. doi: 10.1063/1.870401.

[16] R. T. Palomaki, N. T. Rose, M. van den Bossche, T. J. Sherman, and S. F. De Wekker, "Wind estimation in the lower atmosphere using multirotor aircraft," *J. Atmos. Ocean. Technol.*, vol. 34, no. 5, pp. 1183–1191, 2017. doi: 10.1175/JTECH-D-16-0177.1.

[17] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 2011, pp. 2520–2525. doi: 10.1109/ICRA.2011.5980409.

[18] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digit. Signal Process.*, vol. 73, pp. 1–15, Feb. 2018. doi: 10.1016/j.dsp.2017.10.011.

[19] H. Papadopoulos, V. Vovk, and A. Gammerman, "Regression conformal prediction with nearest neighbours," *J. Artif. Intell. Res.*, vol. 40, pp. 815–840, Jan. 2011.

[20] W. Xiang et al., Verification for machine learning, autonomy, and neural networks survey. 2018. [Online]. Available: arXiv:1810.01989

[21] A. Girard and G. J. Pappas, "Approximate bisimulation relations for constrained linear systems," *Automatica*, vol. 43, no. 8, pp. 1307–1317, 2007. doi: 10.1016/j.automatica.2007.01.019.

Esen Yel, Department of Engineering Systems and Environment, University of Virginia, Charlottesville. Email: esenyel@virginia.edu.

Taylor J. Carpenter, Department of Computer and Information Science, University of Pennsylvania, Philadelphia. Email: carptj@seas.upenn.edu.

Carmelo Di Franco, Departments of Engineering Systems and Environment and Electrical and Computer Engineering, University of Virginia, Charlottesville. Email: cd8gm@virginia.edu.

Radoslav Ivanov, Department of Computer and Information Science, University of Pennsylvania, Philadelphia. Email: rivanov@seas.upenn.edu.

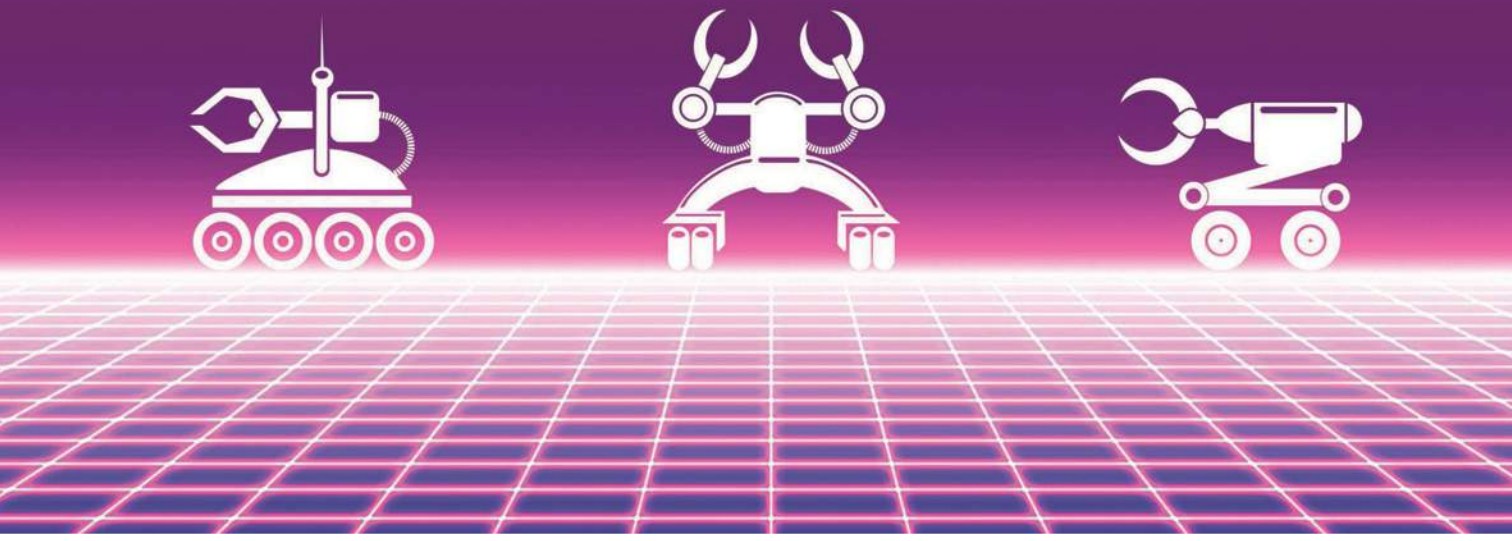
Yiannis Kantaros, Department of Computer and Information Science, University of Pennsylvania, Philadelphia. Email: kantaros@seas.upenn.edu.

Insup Lee, Department of Computer and Information Science, University of Pennsylvania, Philadelphia. Email: lee@seas.upenn.edu.

James Weimer, Department of Computer and Information Science, University of Pennsylvania, Philadelphia. Email: weimerj@seas.upenn.edu.

Nicola Bezzo, Departments of Engineering Systems and Environment and Electrical and Computer Engineering, University of Virginia, Charlottesville. Email: nbezzo@virginia.edu.

Multifidelity Reinforcement Learning With Gaussian Processes



GRID—©ISTOCKPHOTO.COM/ OLEKSANDR KHOMA,
ROBOT IMAGES—©ISTOCKPHOTO.COM/ VECTORTATU

Model-Based and Model-Free Algorithms

By Varun Suryan, Nahush Gondhalekar, and Pratap Tokekar

We study the problem of reinforcement learning (RL) using as few real-world samples as possible. A naive application of RL can be inefficient in large and continuous-state spaces. We present two versions of multifidelity RL (MFRL), model based and model free, that leverage Gaussian processes (GPs) to learn the optimal policy in a real-world environment. In the MFRL framework, an agent uses multiple simulators of the real environment to perform actions. With increasing fidelity in a simulator chain, the number of samples used in successively higher simulators can be reduced. By incorporating GPs in the MFRL framework, we empirically observe an up to 40% reduction in the number of samples for model-based RL and 60% reduction for the model-free version. We examine the performance of our algorithms through simulations and real-world experiments for navigation with a ground robot.

Digital Object Identifier 10.1109/MRA.2020.2977971
Date of current version: 16 April 2020

Recent Developments

Recently, there has been a significant development in RL for robotics. A major limitation of using RL for planning with robots is the need to obtain a large number of

real-world training samples, which can be expensive and, potentially, dangerous. In particular, obtaining exploratory samples—which are crucial to learning optimal policies—may require the robot to collide or fail, which is undesirable. Motivated by these scenarios, we focus on minimizing the number of real-world samples required for learning optimal policies.

One way to reduce the number of real-world samples is to leverage simulators [1]. Collecting learning samples in a robot simulator is often inexpensive and fast. One can use a simulator to learn an initial policy, which is then transferred to the real world—a technique usually referred to as *sim2real* [1]. This lets the robot avoid learning from scratch in the real world, hence reducing the number of physical interactions required. However, this comes with a tradeoff. Although collecting learning samples in simulators is inexpensive, they often fail to capture real-world environments perfectly, a phenomenon called the *reality gap*. Although simulators with increasing fidelity with respect to the real world are being developed, one would expect there to always remain some reality gap.

In this article, we leverage the concept of the MFRL algorithm [2], which uses multiple simulators with varying fidelity levels to minimize the number of real-world (i.e., highest-fidelity simulator) samples. The simulators, denoted by $\Sigma_1, \ldots, \Sigma_d$, have increasing levels of fidelity with respect to the real environment. For example, Σ_1 can be a simple simulator that models only the robot kinematics, Σ_2 can model the dynamics as well as kinematics, and the highest-fidelity simulator can be the real world (Figure 1).

MFRL differs from transfer learning [3], where a transfer of parameters is allowed only in one direction. The MFRL algorithm starts in Σ_1. Once it learns a sufficiently good policy in Σ_1, it switches to a higher-fidelity simulator. If it observes that the policy learned in the lower-fidelity simulator is no longer optimal in the higher-fidelity simulator, it switches back to the lower-fidelity simulator. Cutler et al. [2] showed that the resulting algorithm has polynomial sample complexity and minimizes the number of samples required for the highest-fidelity simulator.

The original MFRL algorithm learns the transition and reward functions at each level. The reward and transition for each state–action pair are learned independently of others. Although this is reasonable for general agents, when planning for physically grounded robots, we can exploit the spatial correlation between neighboring state–action pairs to speed up the learning.

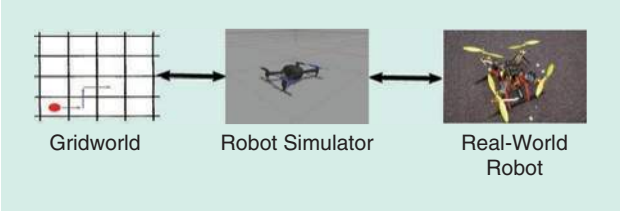


Figure 1. The MFRL framework: the first simulator captures only grid-world movements of a point robot, whereas the second simulator has more fidelity, modeling the physics as well. The control can switch back and forth between simulators and the real environment, which is the third simulator.

Our main contribution is to leverage the GP regression as a function approximator to speed up learning in the MFRL framework. GPs can predict the learned function value for any query point and not just for a discretized state–action pair. Furthermore, GPs can exploit the correlation among nearby state–action values by an appropriate choice of a kernel. GPs have been extensively used to obtain optimal policies in simulation-aided RL [4]. We take this further by using GPs in the MFRL setting.

Other function approximators have been used in RL previously. We chose to use GPs since they require fewer samples to learn a function when a good prior exists [5]. The priors can be imposed, in part, by using appropriate kernels, which make GPs flexible. A major limitation of learning with GPs is their computational complexity, which grows cubically with respect to the number of training samples. However, this issue can be mitigated by using sparse approximations for GPs [6]. In MFRL, the state space of Σ_i is a subset of the state space of Σ_j for all $j > i$. Therefore, when the MFRL algorithm switches from Σ_i to Σ_{i+1}, it already has an estimate for the transition function and Q values for states in Σ_{i+1}. Hence, GPs are particularly suited for MFRL, which we verify through our simulation results.

Our main contributions in this article include introducing

- a model-based MFRL algorithm, GP-VI-MFRL, which estimates the transition function and subsequently calculates the optimal policy using value iteration (VI)
- a model-free MFRL algorithm, GPQ-MFRL, which directly estimates the optimal Q values and, subsequently, the optimal policy.

We verify the performance of the algorithms presented through simulations and experiments with a ground robot. Our empirical evaluation shows that the GP-based MFRL algorithms learn the optimal policy faster than the original MFRL algorithm with even fewer real-world samples.

Related Work

Multifidelity methods are prominently used in various engineering applications to construct a reliable model of a phenomenon when obtaining direct observations of that phenomenon are expensive. The assumption is that we have access to cheaply obtained but possibly less accurate observations from an approximation of that phenomenon. Multifidelity methods can be used to combine those observations with expensive but accurate observations to construct a model of the underlying phenomenon [4]. For example, learning the dynamics of a robot using real-world observations may cause wear and tear of the hardware [7]. Instead, one can obtain observations from a simulator that uses a perhaps crude approximation of the true robot dynamics [8].

Let $f : \mathcal{X} \rightarrow \mathcal{Y}$ denote a function that maps the input $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$ to an output $\mathbf{y} \in \mathcal{Y} \subset \mathbb{R}^{d'}$, where $d, d' \in \mathbb{N}$.

Multiple approximations are available that estimate the same output with varying accuracy and costs. More generally, K different fidelity approximations, $f^{(1)}, \ldots, f^{(K)}$, are available representing the relationship between the input and output, $f^{(i)}: \mathcal{X} \to \mathcal{Y}$, $i = 1, \ldots, K$. Obtaining observations from the ith approximation incurs cost $c^{(i)}$, and typically $c^{(i)} < c^{(j)}$ for $i < j$.

There has been a significant surge in multifidelity methods research following seminal work on autoregressive schemes. Kennedy and O'Hagan [9] used GPs to explore ways in which runs from several levels of a computer code can be used to make inferences about the output of the complex computer code. A complex code approximates reality better, but, in extreme cases, a single run of a complex code may take many days. The GP framework is a natural candidate for estimating a phenomenon by combining data from various fidelity approximations [10], [11].

Multifidelity methods have been widely used in RL applications to automatically optimize the parameters of control policies based on data from simulations and experiments. Cutler et al. [2] introduced a framework that specifies the rules concerning when to collect observations from different fidelity simulators. Marco et al. [12] introduced a Bayesian optimization algorithm that uses entropy [13] as a metric to decide which simulator to collect the observations from. The authors employed their algorithm for a cart-pole setup with a Simulink model as the simulator.

Two closely related techniques addressing sim2real are domain randomization and domain adaptation. In both cases, the simulators can be controlled via a set of parameters. The underlying hypothesis is that some unknown set of parameters closely matches the real-world conditions. In domain randomization, the simulator parameters are randomly sampled, and the agent is trained across all of the parameter values. In domain adaptation, the parameter values are updated during learning.

However, the goal in these methods is to reduce the reality gap using a parameterized simulator [1], [14]. This restricts the use of such approaches to scenarios where altering the parameters for a simulator itself is trivial. Furthermore, only one type of simulator is used. In contrast, MFRL techniques leverage multiple simulators with varying fidelity levels and costs to operate. In addition, in MFRL, the policy learned in the highest-fidelity simulator (real-world) uses data from the same environment, unlike sim2real. As such, these approaches are beneficial when there is a significant reality gap, where maneuvers learned in simulators may not translate to the real world.

Our work is inspired by the work of Cutler et al. [2], which allows for bidirectional transfer of information between simulators. However, they used a tabular representation of the values function. We introduce two algorithms, in a similar spirit, that can decide from which simulator to collect the observations. We hypothesize that using a GP with an MFRL framework will lead to further improvements in the number of samples required from the real world.

Background

RL

RL problems can be formulated as a Markov decision process: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{P}(s_t, a_t, s_{t+1}) \mapsto [0, 1]$, reward function $\mathcal{R}(s_t, a_t) \mapsto \mathbb{R}$, and discount factor $\gamma \in [0, 1)$. A policy $\pi: \mathcal{S} \to \mathcal{A}$ maps states to actions. Together with the initial state s_0, a policy forms a trajectory $\zeta = \{[s_0, a_0, r_0], [s_1, a_1, r_1], \ldots\}$ where $a_t = \pi(s_t)$. r_t and s_{t+1} are sampled from the reward and transition functions, respectively.

We consider a scenario where the goal is to maximize the infinite-horizon, discounted reward starting from a state s_0. The value function for state s_0 is defined as $\mathcal{V}^{\pi}(s_0) = \mathbb{E}[\sum_{t=0}^{t=\infty} \gamma^t r_t(s_t, a_t) | a_t = \pi(s_t)]$. The state–action value function or Q value of each state–action pair under policy π is defined as $Q^{\pi}(s, a) = \mathbb{E}[\sum_{t=0}^{t=\infty} \gamma^t r_{t+1}(s_{t+1}, a_{t+1}) | s_0 = s, a_0 = a]$. which is the expected sum of discounted rewards obtained starting from state s, taking action a and following π thereafter. The optimal Q value function Q^* for a state–action pair (s, a) satisfies $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a) = \mathcal{V}^*(s)$ and can be written recursively as

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r(s_t, a_t) + \gamma \mathcal{V}^*(s_{t+1})]. \tag{1}$$

Our objective was to find the optimal policy $\pi^*(s) = \operatorname{argmax}_a Q^*(s, a)$ when $\mathcal{R}$ and $\mathcal{P}$ are not known to the agent. In model-based approaches, the agent learns $\mathcal{R}$ and $\mathcal{P}$ first and then finds an optimal policy by calculating optimal Q values from (1); the most commonly used model-based approach is VI [15], [16]. We can also directly estimate the optimal Q values, often known as *model-free approaches* [17], or directly calculate the optimal policy, often known as *policy-gradient approaches* [18]. The most commonly used model-free algorithm is Q-learning. For the GP-VI-MFRL implementation, we use GPs to estimate the transition function and VI to calculate the optimal policy. For our GPQ-MFRL implementation, we use Q-learning to perform the policy update using GP regression.

In this article, two versions (model based and model free) of GP-based MFRL are introduced. It should be remembered that, although model-based algorithms are generally more sample efficient in comparison to model-free algorithms, they are not memory efficient [16]. Hence, depending on the application, the model-free approach (GPQ-MFRL) may be a suitable alternative to the more sample-efficient model-based GP-VI-MFRL.

GPs

GPs are Bayesian nonparametric function approximators. They can be defined as a collection of infinitely many random variables, any finite subset $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ of which is jointly Gaussian, with mean vector $\mathbf{m} \in \mathbb{R}^k$ and covariance matrix $\mathbf{K} \in \mathbb{R}^{k \times k}$ [19].

Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ denote the set of training inputs, and let $\mathbf{y} = \{y_1, \ldots, y_k\}$ denote the corresponding training outputs. GPs can be used to predict the output value at a new test

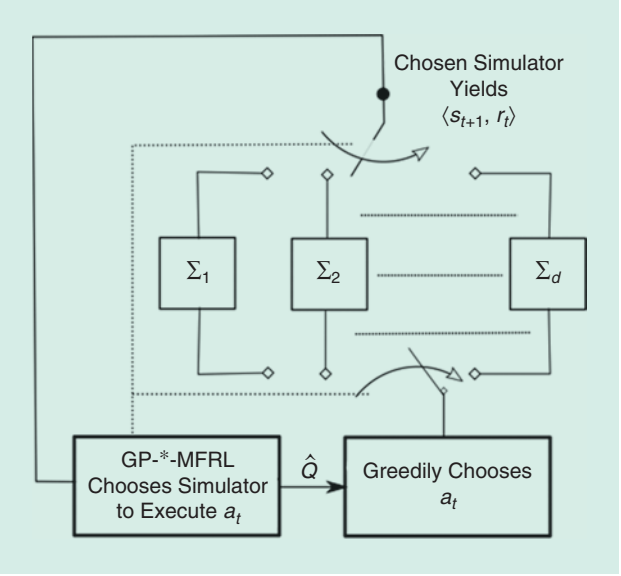


Figure 2. The simulators are represented by $\Sigma_1, \Sigma_2, \ldots, \Sigma_d$. The algorithms determine the simulator in which the current action is to be executed by the agent. Also, the action values in the chosen simulator are updated and used to select the best action, using the information from higher and lower simulators.

Algorithm 1: GP-VI-MFRL

procedure
Input: confidence parameters σ_{th} and σ_{th}^{sum}; simulator chain $\langle \boldsymbol{\Sigma}$, fidelity parameters $\boldsymbol{\beta}$, state mappings $\boldsymbol{\rho} \rangle$; $\mathcal{L}$.
Initialize: Transition functions $\mathcal{P}_{ss'}^{a}(i)$ and $\mathcal{D}_i$ for $i \in \{1, \ldots, d\}$; *change* = FALSE.
Initialize: $i \leftarrow 1$; $\hat{Q}_i \leftarrow$ PLANNER $(\mathcal{P}_{ss'}^{a}(i))$.
while terminal condition is not met
  $a_t \leftarrow \text{argmax}_a \hat{Q}_i(s, a)$
  if $\sigma_i(s_t, a_t) \le \sigma_{th}$: *change* = TRUE
  if $\sigma(\rho_i(s_t), a_t) \ge \sigma_{th}$ and *change* and $i > 1$
    $s_t \leftarrow \rho_i(s_t)$, $i \leftarrow i - 1$, continue
  $\langle s_{t+1}, r_{t+1} \rangle \leftarrow$ execute a_t in Σ_i
  append $\langle s_t, a_t, s_{t+1}, r_t \rangle$ to $\mathcal{D}_i$
  $\mathcal{P}_{ss'}^{a}(i) \leftarrow$ update GP$_i$ using $\mathcal{D}_i$
  $\hat{Q}_i \leftarrow$ call PLANNER with input $\mathcal{P}_{ss'}^{a}(i)$
  $t \leftarrow t + 1$
  if $\sum_{j=t-\mathcal{L}}^{j=t-1} \sigma_i(s_j, a_j) \le \sigma_{th}^{\text{sum}}$ and $t > \mathcal{L}$ and $i < d$
    $s_t \leftarrow \rho_{i+1}^{-1}(s_t)$, $i \leftarrow i + 1$;
    change = FALSE
end procedure

procedure PLANNER$(\mathcal{P}_{ss'}^{a}(i))$
Initialize: $Q(s, a) = 0$ for each (s, a), $\Delta = \infty$
while $\Delta > 0.1$
    $\Delta \leftarrow 0$
for every (s, a)
  $temp \leftarrow Q(s, a)$, $Q(s, a) = \hat{Q}_{i-1}(\rho_i(s), a) + \beta_i$
  for $k \in \{i, \ldots, d\}$
    $s_k = \rho_k^{-1} \ldots \rho_{i+2}^{-1} \rho_{i+1}^{-1}(s)$
    if $\sigma_k(s_k, a) \le \sigma^{th}$: $\mathcal{P}_{ss'}^{a}(i) = \mathcal{P}_{ss'}^{a}(k)$
  $Q(s,a) \leftarrow \sum_a \sum_{s'} \mathcal{P}_{ss'}^{a} [\mathcal{R}_{ss'}^{a} + \gamma \max_a Q(s', a)]$
  $\Delta \leftarrow \max(\Delta, |temp - Q(s, a)|)$
return $Q(s, a)$
end procedure

point, $\mathbf{x}$, conditioned on the training data. The predicted output value at $\mathbf{x}$ is normally distributed, with mean $\hat{\mu}(x)$ and variance $\hat{\sigma}^2(\mathbf{x})$ given by

$$\hat{\mu}(\mathbf{x}) = \mu(\mathbf{x}) + \mathbf{k}(\mathbf{x}, \mathbf{X}) [\mathbf{K}(\mathbf{X}, \mathbf{X}) + \omega^2 \mathbf{I}]^{-1} \mathbf{y}, \quad (2)$$

$$\hat{\sigma}^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x}, \mathbf{X}) [\mathbf{K}(\mathbf{X}, \mathbf{X}) + \omega^2 \mathbf{I}]^{-1} \mathbf{k}(\mathbf{X}, \mathbf{x}), \quad (3)$$

where $\mathbf{K}(\mathbf{X}, \mathbf{X})$ is the kernel. The entry $\mathbf{K}_{\mathbf{x}_l, \mathbf{x}_m}$ gives the covariance between two inputs $\mathbf{x}_l$ and $\mathbf{x}_m$, and $\mu(\mathbf{x})$ in (2) is the prior mean of the output value at $\mathbf{x}$.

We use a zero-mean prior and a squared-exponential kernel where $\mathbf{K}_{\mathbf{x}_l, \mathbf{x}_m}$ is given by

$$\mathbf{K}_{\mathbf{x}_l, \mathbf{x}_m} = \sigma^2 \exp\left(-\frac{1}{2} \sum_{d=1}^{d=D} \left(\frac{\mathbf{x}_{dl} - \mathbf{x}_{dm}}{l_d} \right)^2 \right) + \omega^2; \quad (4)$$

σ^2, l_d, and ω^2 are hyperparameters that can be either set by the user or learned online through the training data; and D is the dimension of the training inputs.

In the GPQ-MFRL algorithm, we use GPs to learn Q values. GPs are proven to be consistent function approximators in RL with convergence guarantees [19]. A set of state–action pairs is the input to the GP, and Q values are the output/observation values to be predicted. In the GP-VI-MFRL algorithm, we use GPs to learn the transition function. The input to the GPs is a set of observed state–action pairs, and we predict the next state as output for a newly observed state–action pair.

Algorithm Description

In this section, we first describe both versions of our algorithm. We compare the proposed algorithms with baseline strategies through simulations. A flowchart of our algorithms is shown in Figure 2. We make the following assumptions for both algorithms.

- The reward function is known to the agent; we make this assumption for ease of exposition. In general, one can use GPs to estimate the reward function as well. This assumption is required only for the GP-VI-MFRL algorithm.
- The state space in simulator Σ_{i-1} is a subset of the state space in simulator Σ_i. The many-to-one mapping ρ_i maps states from simulator Σ_i to states in simulator Σ_{i-1}. We give an example of such mapping in subsequent sections. Let ρ_i^{-1} denote the respective inverse mapping (which can be one-to-many) from states in Σ_{i-1} to states in Σ_i. The action space is discrete and the same for all of the simulators and the real world.

GP-VI-MFRL Algorithm

Algorithm 1 consists of a model learner and a planner. The model learner learns the transition functions using GP regression. We use VI [18] as our planner to calculate the optimal policy with learned transition functions. Let $\mathbf{s}_{t+1} = f(\mathbf{s}_t, a_t)$ be the (unknown) transition function that must be learned. We observe transitions $\mathcal{D} = \{\langle \mathbf{s}_t, a_t, \mathbf{s}_{t+1} \rangle\}$. Our goal is to learn an estimate $\hat{f}(\mathbf{s}, a)$ of $f(\mathbf{s}, a)$. We can then use this estimated $\hat{f}$ for unvisited state–action pairs (in place of f) during VI

to learn the optimal policy. For a given state–action pair (s, a), the estimated transition function is defined by a normal distribution, with the mean and variance given by (2) and (3). Algorithm 1 gives the details of the proposed framework.

Before executing an action, the agent checks (line 8) whether it has a sufficiently accurate estimate of the transition function for the current state–action pair in the previous simulator, Σ_{i-1}. Specifically, we check whether the variance of the current state–action pair in the previous simulator is less than σ_{th}. If not, and if the transition model in the current environment has changed, it switches to Σ_{i-1} and executes the action in the potentially less expensive environment. The agent lands in the state $\rho_i(s)$ in the lower-fidelity simulator.

We also keep track of the variance of the $\mathcal{L}$ most recently visited state–action pairs in the current simulator. If the running sum of the variances is below a threshold (line 15), this suggests that the robot is confident about its actions in the current simulator and can advance to the next one. In the original work [2], the agent switched to the higher-fidelity simulator after a certain number of known state–action pairs were encountered. In our implementation (line 7), the model of the current environment changes if the posterior variance for a state–action pair drops below a threshold value (i.e., the agent has a sufficiently accurate estimate of the transitions from that state). Lines 10–13 describe the main body of the algorithm, where the agent executes the greedily chosen action and records the observed transition in $\mathcal{D}_i$. The GP model for the transition function is updated after every step (line 12). New Q value estimates are computed every time after an update of the transition function (line 13). Note that we use a separate GP to estimate the transition function in each simulator.

One can use a number of termination conditions (line 5), e.g., maximum number of steps, changes in the value function, or maximum number of switches. In our implementation, Algorithm 1 terminates if the change in new estimates of value functions in the real-world environment is no more than a certain threshold (10%) compared with the previous estimates.

The planner utilizes knowledge of transitions from higher (lines 26–28) and lower simulators (line 25) to encourage exploration in the current simulator. For every state–action pair (s, a), the planner looks for the maximum-fidelity simulator in which a known estimate of transitions for (s, a) is available and uses them to plan in the current simulator. An estimate is termed *known* if the variance is below a threshold. If no such simulator is available, then it uses the Q values learned in the previous simulator plus a fidelity parameter β. This parameter models the maximum possible difference between the optimal Q values in consecutive simulators. The higher-fidelity simulator values will always be trusted over the lower-fidelity ones as long as we have low enough uncertainty in those estimates. We assume that, for two consecutive simulators, the maximum difference between the optimal action value of a state–action pair in Σ_i and the corresponding pair in Σ_{i-1} is not more than β_i. One must apply a state-space discretization to plan the actions. However, the learned transition function is continuous.

GPQ-MFRL Algorithm

The agent learns optimal Q values using GPs directly instead of learning the model first. The underlying assumption is that nearby state–action pairs produce similar Q values. This assumption can also be applied to problems where the states and actions are discrete but the transition function implies some sense of continuity. We chose the squared-exponential kernel because it models the spatial correlation we expect to see in a ground robot. However, any appropriate kernel can be chosen. We used a separate GP per simulator to estimate the Q values from data collected only in that simulator.

Algorithm 2 gives the details of the proposed framework. GPQ-MFRL continues to collect samples in the same simulator until the agent is confident about its optimal actions. If the running sum of the variances is below a threshold (line 17), this suggests that the robot has found a good policy with high confidence in the current simulator, and it must advance to the next one (line 18).

GPQ-MFRL uses similar thresholds (σ_{th} and σ_{th}^{sum}) as GP-VI-MFRL to determine when to switch to a lower- or higher-fidelity simulator. GP-VI-MFRL checks whether the agent has a sufficiently accurate estimate of the transition

Algorithm 2: GPQ-MFRL

1: **procedure**
2: **Input:** confidence parameters σ_{th} and σ_{th}^{sum}; simulator chain $\langle \Sigma$, fidelity parameters $\boldsymbol{\beta}$, state mappings $\rho\rangle$; $\mathcal{L}$.
3: **Initialize:** $\hat{Q}_i$ = initialize GP for $i \in \{1, \ldots, d\}$; state s_0 in simulator Σ_1; $i \leftarrow 1$; *change* = FALSE.
4: **Initialize:** $t \leftarrow 0$; $\mathcal{D}_i \leftarrow \{\}$ for $i \in \{1, \ldots, d\}$.
5: **while** terminal condition is not met
6: $a_t \leftarrow$ CHOOSEACTION(s_t, i)
7: **if** $\sigma_i(s_t, a_t) \leq \sigma_{th}$: *change* = TRUE
8: **if** $\sigma(\rho_i(s_t), a_t) > \sigma_{th}$ and *change* and $i > 1$
9: $s_t \leftarrow \rho_i(s_t)$, $i \leftarrow i - 1$, continue
10: $\langle r_t, s_{t+1}\rangle \leftarrow$ execute action a_t in Σ_i
11: append$\langle s_t, a_t, s_{t+1}, r_t\rangle$ to $\mathcal{D}_i$
12: $\mathcal{Y}_i \leftarrow \{\}$
13: **for** $\langle s_t, a_t, s_{t+1}, r_t\rangle \in \mathcal{D}_i$ //batch training//
14: $y_t \leftarrow r_t + \gamma \max_a \hat{Q}_i(s_{t+1}, a)$
15: append $\langle s_t, a_t, y_t\rangle$ to $\mathcal{Y}_i$
16: $\hat{Q}_i \leftarrow$ update GP$_i$ using $\mathcal{Y}_i$
17: **if** $\sum_{j=t-\mathcal{L}}^{j=t} \sigma_i(s_j, a_j) \leq \sigma_{th}^{\text{sum}}$ and $t > \mathcal{L}$ and $i < d$
18: $s_t \leftarrow \rho_{i+1}^{-1}(s_t)$, $i \leftarrow i + 1$
19: *change* = FALSE
20: **end procedure**
21:
22: **procedure** CHOOSEACTION(s, i)
23: **for** $a \in \mathcal{A}(s)$
24: $Q(s, a) = \hat{Q}_{i-1}(\rho_i(s), a) + \beta_i$
25: **for** $k \in \{i, \ldots, d\}$
26: $s_k = \rho_k^{-1} \ldots \rho_{i+2}^{-1} \rho_{i+1}^{-1}(s)$
27: **if** $\sigma_k(s_k, a) \leq \sigma^{th} : Q(s, a) = \hat{Q}_k(s_k, a)$
28: return $\arg\max_a Q(s, a)$
29: **end procedure**

function in the previous simulator, whereas GPQ-MFRL checks whether the agent has a sufficiently accurate estimate of optimal Q values in the previous simulator (line 8). Lines 10–15 describe the main body of the algorithm, where the agent records the observed transitions in $\mathcal{D}_i$. We update target values (line 14) for every transition, as more data are collected in $\mathcal{D}_i$ (line 13). The GP model is updated after every step (line 16).

The agent utilizes the experiences collected in higher simulators (lines 25–27) to choose the optimal action in the current simulator (line 6). Specifically, it checks for the maximum-fidelity simulator in which the posterior variance for (s, a) is less than a threshold σ_{th}. If one exists, it utilizes the Q values from the highest known simulator to choose the next action in the current simulator. If no such higher simulator exists, the Q values from the previous simulator (line 24) are considered to choose the next action in the current simulator with an additive fidelity parameter β.

GPQ-MFRL performs a batch retraining every time the robot collects new sample in a simulator (lines 13–15). During batch retraining, the algorithm updates the target values in previously collected training data using the knowledge gained by collecting new samples. Then, these updated target values are used to predict the Q values using GPs (line 16). As the amount of data grows, updating the GP can become computationally expensive; however, we can prune the data set using sparse GP techniques [6]. It is nontrivial to choose values for confidence bounds, but, for the current experiments, we chose the σ_{th}^{sum} to be 10% of the maximum Q value possible and σ_{th} to be one fifth of σ_{th}^{sum}.

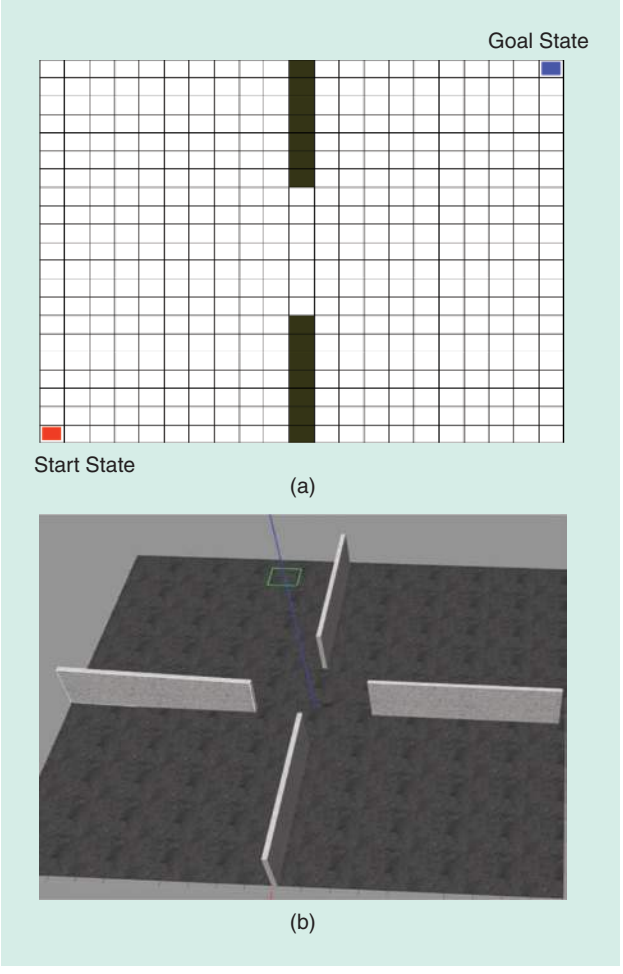


Figure 3. The environment setup for a multifidelity simulator chain. (a) The grid-world simulator (Σ_1) has two walls, while (b) the Gazebo simulator (Σ_2) has four walls.

Results

We use two environments to simulate GP-VI-MFRL and three environments for GPQ-MFRL. For GP-VI-MFRL, the goal was learning to navigate from one point to another while avoiding the obstacles. Σ_1 is a 21 × 21 grid world with a point robot, while Σ_2 is Gazebo (discretized in a 21 × 21 grid), which simulates the kinematics and dynamics of a quadrotor operating in 3D. For GPQ-MFRL, the goal was to learn to avoid the obstacles while navigating through the environment. Σ_1 is the Python-based simulator Pygame, Σ_2 is a Gazebo environment, and Σ_3 is the real world. We further used sparse GPs to speed up the computations required to perform GP inference. We report the improvements in computational time and a direct comparison between GP-VI-MFRL and GPQ-MFRL on an obstacle-avoidance task.

GP-VI-MFRL Algorithm

The task of the robot is to navigate from the start state to goal state. The start and goal states and the obstacles for the environment used are shown in Figure 3. The state of the robot is given by its x and y coordinates, and the action is a 2D velocity vector. Both simulators have the same state space; therefore, ρ_i is an identity mapping. The robot gets a reward of zero for all transitions except when it hits the obstacles (in which case it gets a reward of −50), and it gets a reward of 100 for landing in the goal state.

Since the state space $\mathcal{S} \in \mathbb{R}^2$ and the action space (velocity) $\mathcal{A} \in \mathbb{R}^2$, the true transition function is $\mathbb{R}^4 \rightarrow \mathbb{R}^2$.

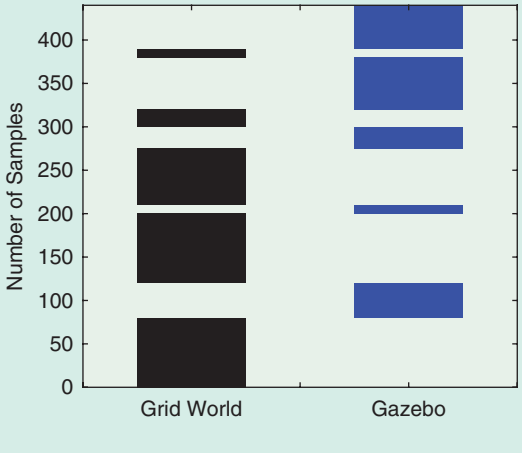


Figure 4. The samples collected at each level of the simulator for a 21 × 21 grid in the grid-world and Gazebo environments. The values of σ_{th}^{sum} and σ_{th} were kept at 0.4 and 0.1, respectively.

However, generally, GP regression allows for single-dimensional outputs only. Therefore, we assume independence between the two output dimensions and separately learn two components (along x and y) of the transition functions, $x_{i+1} = f_x(x_i, y_i, a_x)$ and $y_{i+1} = f_y(x_i, y_i, a_y)$, where (x_i, y_i) and (x_{i+1}, y_{i+1}) are the current and next states of the robot, and (a_x, a_y) is the velocity input. The GP prediction is used to determine the transitions, $(x_i, y_i, a_x) \rightarrow x_{i+1}$ and $(x_i, y_i, a_y) \rightarrow y_{i+1}$, where (x_{i+1}, y_{i+1}) is the predicted next state with variances σ_x^2 and σ_y^2, respectively.

Figure 4 shows the switching between the simulators pictured in Figure 3 for one run of the GP-VI-MFRL algorithm. Unlike unidirectional transfer-learning algorithms, the GP-VI-MFRL agent switches back and forth between the simulators, initially collecting most samples in the first simulator. Eventually, the robot starts to collect more samples in the higher-fidelity simulator. This is the case when the algorithm is near convergence and has accurate estimates for transitions in the lower-fidelity simulator as well.

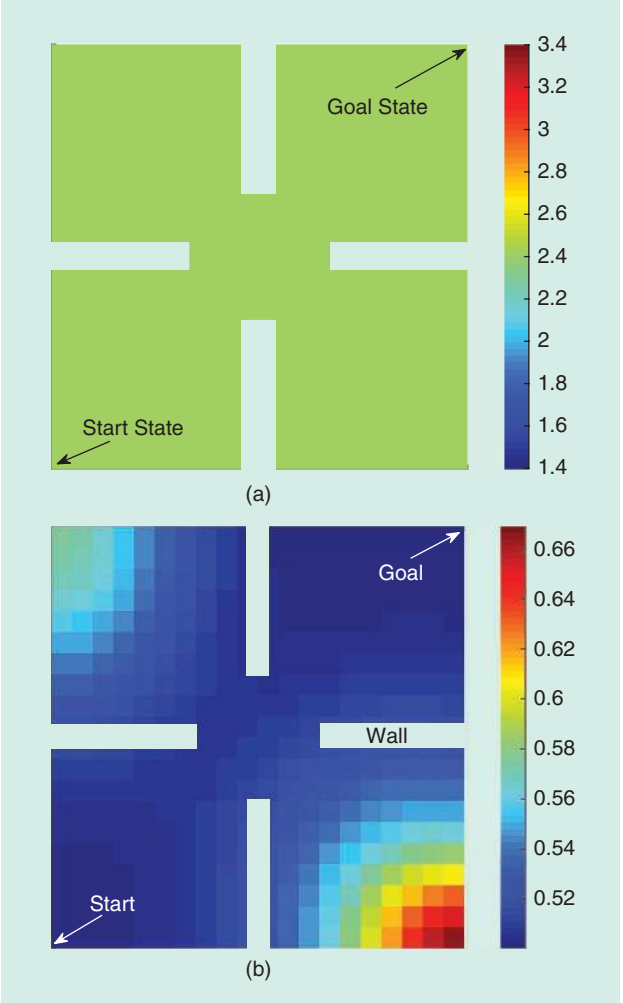


Figure 5. The variance plot for the Gazebo simulator after transition-function initialization and after the algorithm has converged. Colored regions show the respective $\sqrt{\sigma_x^2 + \sigma_y^2}$ values for the optimal action returned by the planner in each state at (a) initialization and (b) after convergence.

Next, we describe the effect of the parameters used in GP-VI-MFRL and the fidelity of the simulators on the number of samples until convergence.

Variance in Learned Transition Function

To demonstrate how the variance of the predicted transition function varies from the beginning of the experiment to convergence, we plotted heatmaps of the posterior variance for Gazebo environment transitions. The GP prediction for a state–action pair gives the variance σ_x^2 and σ_y^2, respectively, for the predicted state. After convergence (Figure 5), the variance along the optimal (i.e., likely) path is low, whereas the variance for states unlikely to be on the optimal path from start to goal remains high, since those states are explored less often in the Gazebo environment. Hence, utilizing the

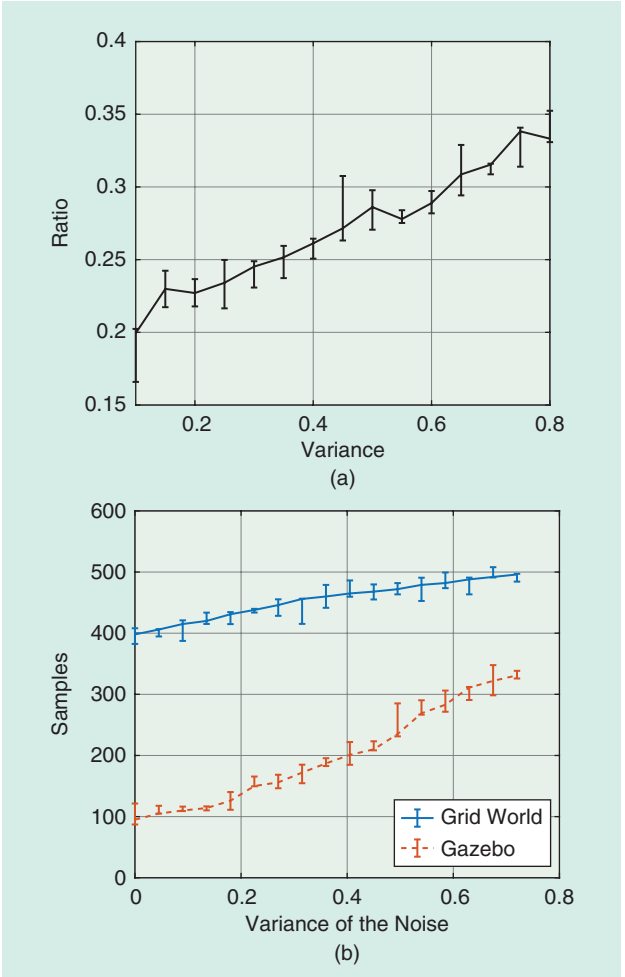


Figure 6. As we lower the fidelity of the grid world by adding more noise in grid-world transitions, the agent tends to spend more time in Gazebo. The plots show the average and minimum–maximum error bars of five trials. (a) The ratio of samples collected in Gazebo and total samples (y-axis) as a function of the fidelity of the grid world. We lower the fidelity of the grid world by increasing the variance of the simulated transition function. (b) The number of samples collected (y-axis) in Gazebo increases more rapidly (as demonstrated by the diminishing vertical separation between the two plots) than samples collected in the grid world.

experience from lower-fidelity simulator results in more directed exploration of the higher-fidelity simulators.

Effect of Fidelity on the Number of Samples

Next, we studied the effect of varying the fidelity on the total number of samples and the fraction of samples collected in the Gazebo simulator. Our hypothesis was that, as the fidelity of the first simulator decreases, the agent will need more samples in Gazebo. To validate this hypothesis, we varied the noise added to simulate the transitions in the grid world. The transition model in Gazebo remained the same. The total number of samples collected increases as we increase the noise in the grid world [Figure 6(b)]. As we do so, the agent learns less accurate transition functions, leading to more samples collected in Gazebo. Not only does the agent need more samples, but the ratio of the samples collected in Gazebo to the total number of samples also increases [Figure 6(a)].

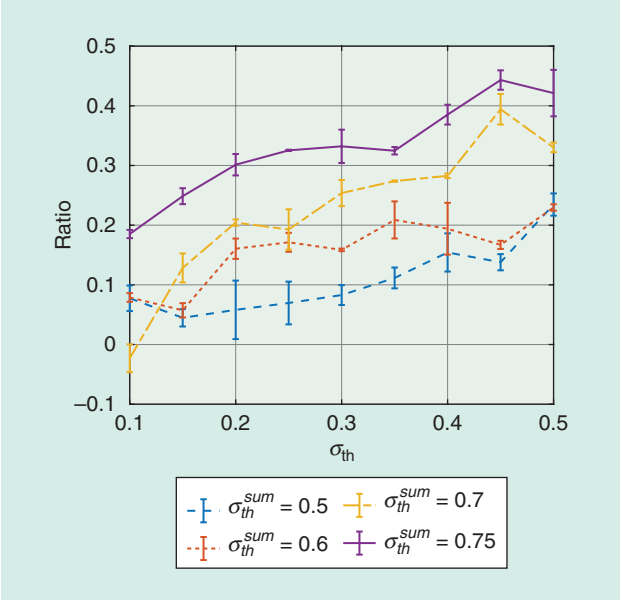


Figure 7. The ratio of samples collected in Gazebo to the total samples as a function of the confidence parameter σ_{th} for four different values of σ_{th}^{sum}. The figure shows the average and standard deviation of five trials.

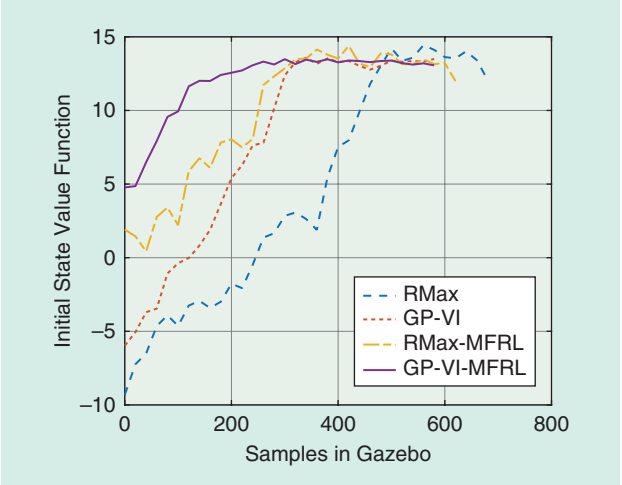


Figure 8. A comparison of GP-VI-MFRL with three baseline strategies. The *y*-axis shows the value function for the initial state $V(s_0)$ in Gazebo as a function of the number of samples collected in Gazebo. The value function estimation for GP-VI-MFRL converges most quickly.

Effect of the Confidence Parameters

The GP-VI-MFRL algorithm uses two confidence parameters, σ_{th} and σ_{th}^{sum}, which quantify the variances in the transition function to switch to a lower and higher simulator, respectively. Figure 7 shows the effect of varying the two parameters on the ratio of the number of samples collected in the Gazebo simulator to the total number of samples. Smaller σ_{th} and σ_{th}^{sum} result in the agent collecting more samples in the lower-fidelity simulator and may also result in slow convergence. Depending on user preference, one can choose the values of confidence bounds from Figure 7.

Comparison With RMax MFRL

Figure 8 compares GP-VI-MFRL with three other baseline algorithms:

- the RMax algorithm running only in Gazebo without grid world (RMax)
- the GP-MFRL algorithm running only in Gazebo with no grid world present (GP-VI)
- the original MFRL algorithm [2] (RMax-MFRL).

Specifically, we plot the value of the initial state, $V(s_0)$, as a function of the number of samples in Gazebo, i.e., Σ_2. We observe that GP-VI-MFRL uses fewer samples in Gazebo to converge to the optimal value than the other methods.

GP-VI-MFRL performs a GP update at each time step. This GP update grows cubically with the number of training samples, which makes GP-VI-MFRL computationally infeasible beyond a certain number of training samples. However, this issue can be addressed by using appropriate active-learning strategies, which select a subset of samples to retain, thereby keeping the size of the data set constant. The total computational time for GP-VI-MFRL to perform GP updates on collected samples accounts for approximately 10 min.

GPQ-MFRL Algorithm

We use three environments (Figure 9) to demonstrate the GPQ-MFRL algorithm. The task for the robot is to navigate through a given environment without crashing into the obstacles, assuming the robot has no prior information about the environments. There is no goal state.

The robot has a laser sensor that gives distances from obstacles along seven equally spaced directions. The angle between two consecutive measurement directions was set to be $\pi/8$ radians. The actual robot has a Hokuyo laser sensor that operates in the same configuration. Distance measurements along the seven directions serve as the state in the environments. Therefore, we have a 7D continuous state space: $\mathcal{S} \in (0, 5]^7$. The linear speed of the robot was held constant at 0.2 m/s. The robot can choose its angular velocity from 19

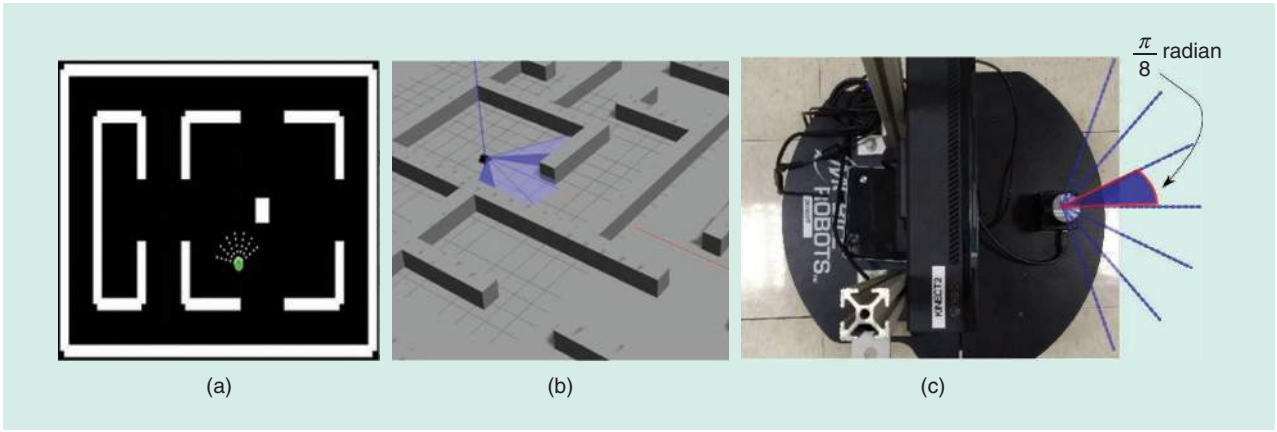


Figure 9. We used (a) the Python-based simulator Pygame as Σ_1, (b) Gazebo as Σ_2, and (c) the Pioneer P3-DX robot in the real world as Σ_3.

possible options: $\{-\pi/9, -\pi/8, \ldots, \pi/9\}$. The reward in each state was set to be the sum of laser readings from seven directions except when the robot hits the obstacle. In case of a collision, it gets a reward of –50.

We trained the GP regression: $Q(\mathbf{s}, a) : \mathbb{R}^8 \rightarrow \mathbb{R}$. Hyperparameters of the squared-exponential kernel were calculated offline by minimizing the negative log marginal likelihood of 2,000 training points, which were collected by letting the robot run in the real world directly. The parameter values for experiments in this section are given in Table 1.

Average Cumulative Reward in the Real World

In Figure 10, we compare the GPQ-MFRL algorithm with three other baseline strategies by plotting the average cumulative reward collected by the robot as a function of samples collected in the real world. The three baseline strategies are

- directly collecting samples in the real world without the simulators (direct policy)
- acquiring 100 samples in one simulator and transferring the policy to the Pioneer robot, with no further learning in the real world (frozen policy)
- attaining 100 samples in one simulator and transferring the policy to the robot while continuing to learn in the real world (transferred policy).

We observe that the direct policy performs worst in the beginning, which can be attributed to the fact that the robot started to learn from scratch. The frozen policy starts off better since it has already learned a policy in the simulator. However, it tends toward a lower value of average cumulative reward, which suggests that the optimal policy learned in the simulator is not the optimal policy in the real world. Although the transferred policy seems to perform better at the beginning than the frozen policy, it is difficult to state definitively that this will always be the case.

The direct policy has a large performance variance in the beginning. GPQ-MFRL outperforms the other strategies right from the beginning; we attribute this to the fact that the GPQ-MFRL collects more samples from the simulator in the beginning and, hence, starts better from the outset. If the transferred policy and frozen policy had been allowed to collect more samples from the simulator, they might have performed the same as the GPQ-MFRL. However, deciding how many samples one should allow is a nontrivial task and problem specific. GPQ-MFRL can decide the

Table 1. Parameters used in GPQ-MFRL.

Description	Type	Value
	σ	102.74
Hyperparameters	l	[2.1, 5.1, 14, 6.2, 15, 2, 2, 1]
	ω^2	20
	σ_{th}^{sum}	60
Confidence parameters	σ_{th}	15
Algorithm	$\mathcal{L}$	5

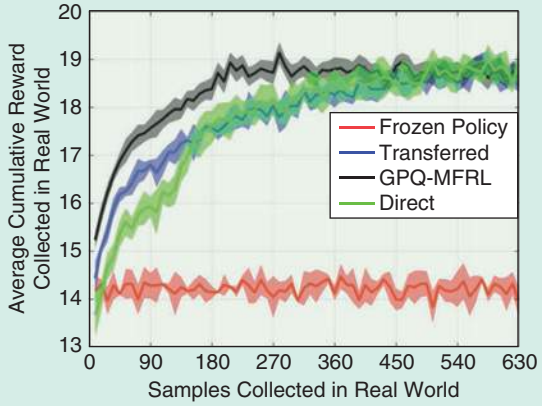


Figure 10. The average cumulative reward collected by the Pioneer robot in the real-world environment as a function of the samples collected in the real world. The plot shows the averages and standard deviations of five trials.

number of samples in each simulator by itself without the need for human intervention.

Policy Variation With Time

Figure 11 shows the absolute percent change in the sum of the value functions with respect to the last estimated sum of the value functions and average predictive variance for states $\{1, 3, 5\}^7$ in all three simulators. Initially, most of the samples are collected in the simulator, whereas, over time, the samples are collected mostly in the real world. The simulators help the robot make its value estimates converge quickly, as observed by a sharp dip in the first white region. Note that GP updates for the ith simulator ($\hat{Q}_i$) are made only when the robot is running in ith simulator.

Higher-Dimensional Spaces and Sparse GPs

One of the limitations of GPs is their computational complexity, which grows cubically with the number of training samples. However, we use sparse approximations to address this limitation. We also increase the dimensionality of the state space to verify whether the proposed algorithms scale to higher dimensions. Specifically, we increase the number of laser readings to 180 equally spaced directions. Therefore, GP regression is used to estimate $Q(\boldsymbol{s}, a) : \mathbb{R}^{181} \rightarrow \mathbb{R}$.

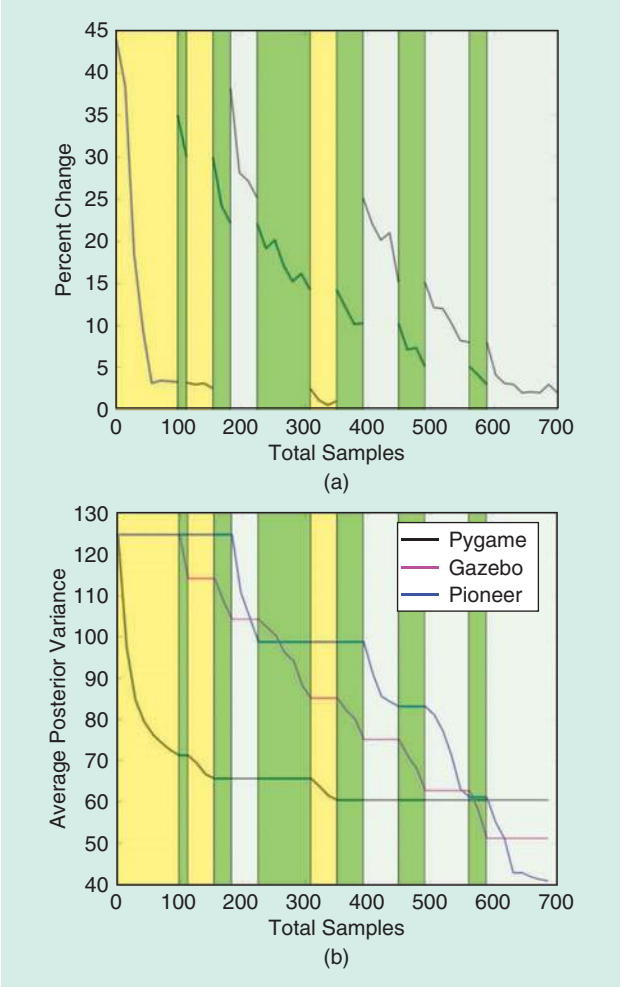


Figure 11. The yellow, green, and white regions correspond to the samples collected in the Pygame, Gazebo, and real-world environments, respectively. Plots are for state set $\{1, 3, 5\}^7$. (a) The sum of absolute change in the value functions. (b) The average variances in value-function estimations.

There are several methods for sparse GP approximations. We use the technique from [6] that finds a possibly smaller set of points (called *inducing points*) that best fit the data. The GP inference is conditioned on the smaller set of inducing points rather than the full set of training samples. Finding inducing points is closely related to finding low-rank approximations of the full GP covariance matrix. The inducing points may or may not belong to the actual training data.

We did several experiments with GPQ-MFRL to study the performance of the algorithm for a number of inducing points. We used Pyro [20] to implement sparse GPs. Figure 12 shows the average cumulative reward collected by the robot in the Gazebo environment when GP inference is done with the number of inducing points set to 5%, 15%, 25%, and 100% of the total training samples. We observe a significant increase in the cumulative reward collected when going from 5 to 15% but not much from 15 to 25% (y-axes in Figure 12).

A plot of the wall clock times to perform GP inference in Pygame is shown in Figure 13. The wall time to perform GP inference in Gazebo exhibits a similar trend, which we omit for the sake of brevity. The wall clock time represents the time to perform all GP operations, including the time to update the hyperparameters and find the inducing points of both GPs. We update these after every 10 new training samples in an individual simulator. The experiments were performed on a machine running Ubuntu 16.04 with an Intel Core i7-5600U CPU at 2.60 GHz, Intel HD Graphics 5500, and 16 GB of random-access memory.

The results suggest that a small number of inducing points is sufficient and yields diminishing marginal gains in the performance when the amount of inducing points increases. Inference on inducing points with 25% of the training data leads to performance almost as good as that achieved with full GP regression, in terms of the reward collected by the learned policy [Figure 12(d)], but it is significantly faster.

Comparison Between GP-VI-MFRL and GPQ-MFRL

We compare GP-VI-MFRL and GPQ-MFRL using the average cumulative reward collected by the robot in Gazebo as the metric in the obstacle-avoidance task. To do this, we used full GP regression to perform the inference. The laser obtains distance measurements from seven equally spaced directions, and we trained seven independent GPs to learn the transition function in GP-VI-MFRL (one GP corresponding to each direction). A performance comparison is shown in Figure 14. Although both algorithms seem to perform the same asymptotically, GP-VI-MFRL performs slightly better than GPQ-MFRL in the beginning.

Discussion and Future Work

We presented GP-based MFRL algorithms that leverage multiple simulators with varying fidelity and costs to learn a policy in the real world. We demonstrated empirically that the GP-based MFRL algorithms find the optimal policies using fewer samples than the baseline algorithms, including the original MFRL

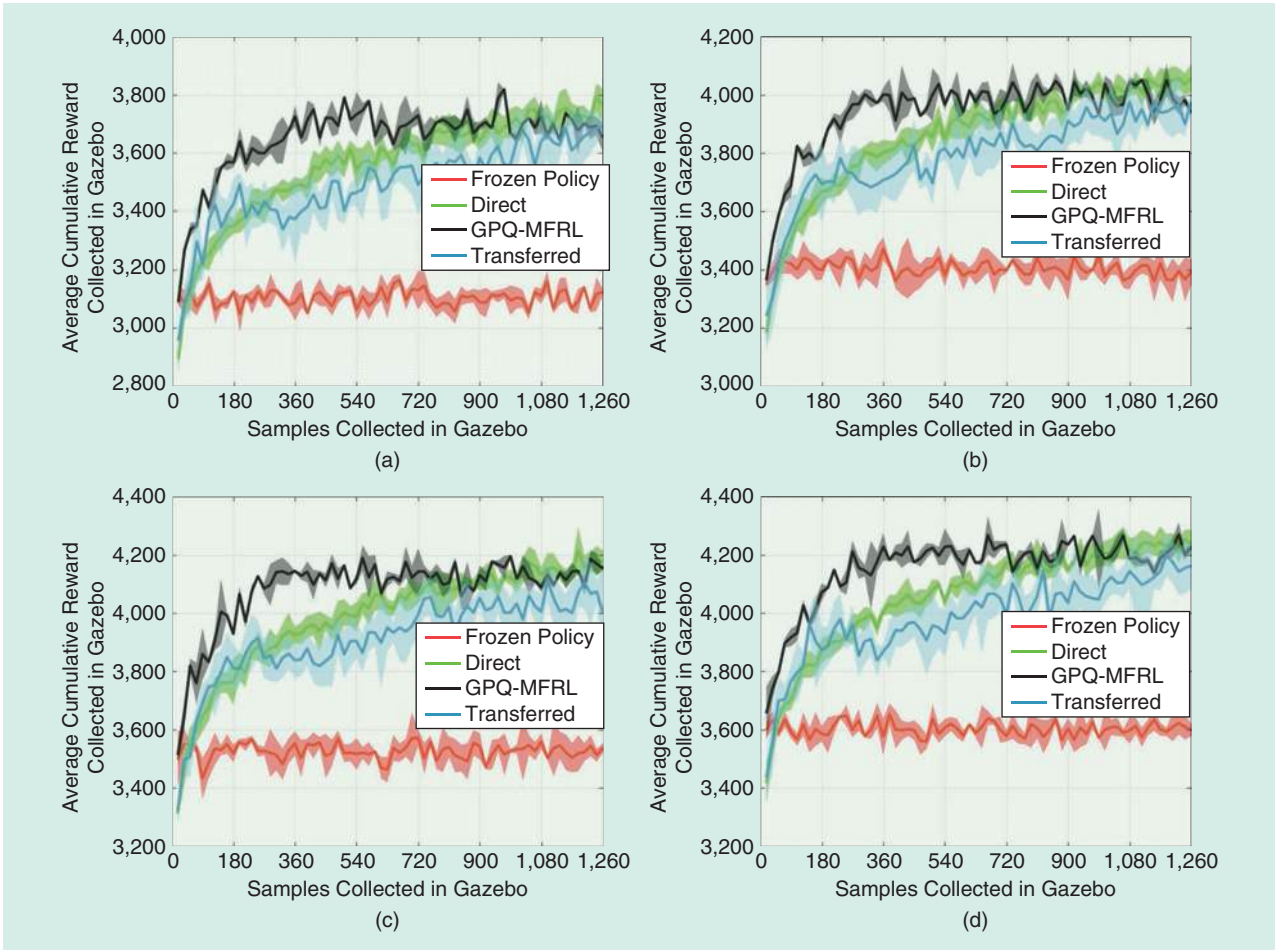


Figure 12. The average cumulative reward collected by the robot in the Gazebo environment as a function of the samples collected in Gazebo for different percentages of inducing points. The plots show the averages and standard deviations of five trials for the inducing points set to (a) 5%, (b) 15%, and (c) 25% of the training sample size as well as (d) full GP inference.

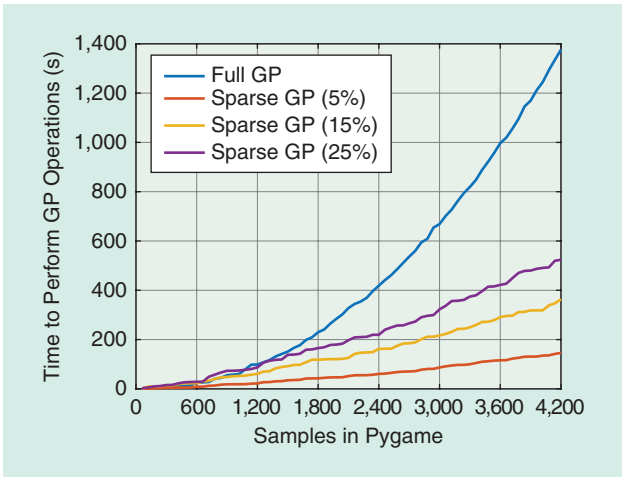


Figure 13. The wall clock time required to perform all GP-related operations in Pygame for various degrees of GP sparse approximations. The plot shows the mean time over five trials for each case.

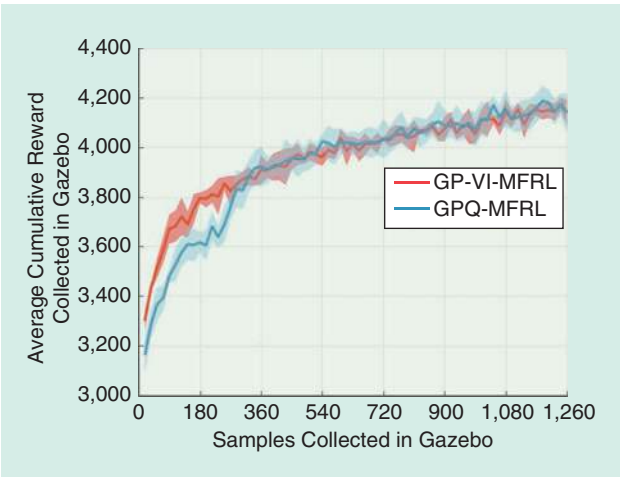


Figure 14. The average cumulative reward collected by the robot in the Gazebo environment as a function of the samples collected in Gazebo for GP-VI-MFRL and GP-Q-MFRL. The plots show the averages and standard deviations of 10 trials.

algorithm [2]. The computational limitations of sparse GPs can be mitigated, to an extent, by the use of sparse GP approximations. We also provided a head-to-head comparison between the two algorithms presented here. GP-VI-MFRL (the model-based version) performs better than GPQ-MFRL (the model-free version) in the beginning, which is consistent with the outcomes for traditional RL techniques, where model-based algorithms tend to perform better than model-free ones.

An immediate future work is to compare the MFRL technique with sim2real approaches [1]. Unlike sim2real, the presented MFRL techniques explicitly decide when to switch between simulators and use more than two levels of simulators. An interesting avenue of future work would be to combine the two ideas: use MFRL to model the fact that some simulators are cheaper/faster to operate than others and use parameterized simulators to bring in domain adaptation/randomization for better generalization. Finally, in the current approach, data from different simulators are not combined when GP regression is performed.

One possibility for improvement is to use multitask GPs that can simultaneously produce multiple outputs, one corresponding to each fidelity simulator. Multitask GPs [21] can produce multiple outputs simultaneously, one corresponding to each simulator. An alternative is to use deep GPs to combine data from various fidelities as part of the same network. In both cases, the goal is to learn the correlation between values in different environments directly.

Acknowledgments

This work has been funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation–sponsored industry/university cooperative research center under NSF award IIP-1161036, along with significant contributions from C-UAS industry members. This work was performed when the authors were with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg.

References

[1] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, and N. Ratliff, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, May 2019. doi: 10.1109/ICRA.2019.8793789.

[2] M. Cutler, T. J. Walsh, and J. P. How, "Real-world reinforcement learning via multifidelity simulators," *IEEE Trans. Robot.*, vol. 31, no. 3, pp. 655–671, June 2015. doi: 10.1109/TRO.2015.2419431.

[3] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via inter-task mappings for temporal difference learning," *J. Mach. Learn. Res.*, vol. 8, pp. 2125–2167, Sept. 2007.

[4] M. Cutler and J. P. How, "Efficient reinforcement learning for robots using informative simulated priors," in *Proc. 2015 IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 2605–2612. doi: 10.1109/ICRA.2015.7139550.

[5] R. M. Neal, *Bayesian Learning for Neural Networks*, vol. 118, Berlin: Springer-Verlag, 2012.

[6] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *J. Mach. Learn. Res.*, vol. 6, pp. 1939–1959, Dec. 2005. doi: 10.5555/1046920.1194909.

[7] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013. doi: 10.1177/0278364913495721.

[8] J. Tan et al., Sim-to-real: Learning agile locomotion for quadruped robots. 2018. [Online]. Available: https://arxiv.org/pdf/1804.10332.pdf

[9] M. C. Kennedy and A. O'Hagan, "Predicting the output from a complex computer code when fast approximations are available," *Biometrika*, vol. 87, no. 1, pp. 1–13, 2000. doi: 10.1093/biomet/87.1.1.

[10] P. Perdikaris, M. Raissi, A. Damianou, N. Lawrence, and G. E. Karniadakis, "Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling," *Proc. R. Soc. A, Math. Phys. Eng. Sci.*, vol. 473, no. 2198, p. 20160751, 2017. doi: 10.1098/rspa.2016.0751.

[11] A. Damianou, "Deep Gaussian processes and variational propagation of uncertainty," Ph.D. dissertation, Dept. Neurosci., Univ. Sheffield, 2015.

[12] A. Marco et al., "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization," in *Proc. 2017 IEEE Int. Conf. Robotics and Automation (ICRA)*. pp. 1557–1563. doi: 10.1109/ICRA.2017.7989186.

[13] P. Hennig and C. J. Schuler, "Entropy search for information-efficient global optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 1809–1837, June 2012.

[14] J. Tobin et al., "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. 2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30. doi: 10.1109/IROS.2017.8202133.

[15] T. Jung and P. Stone, "Gaussian processes for sample efficient reinforcement learning with RMAX-like exploration," in *Proc. European Conf. Machine Learning*, Sept. 2010, pp. 601–616. [Online]. Available: http://www.cs.utexas.edu/users/ai-lab/?ECML10-jung

[16] R. I. Brafman and M. Tennenholtz, "R-max-a general polynomial time algorithm for near-optimal reinforcement learning," *J. Mach. Learn. Res.*, vol. 3, pp. 213–231, Oct. 2002. doi: 10.1162/153244303765208377.

[17] R. Grande, T. Walsh, and J. How, "Sample efficient reinforcement learning with Gaussian processes," in *Proc. 31st Int. Conf. Machine Learning (ICML-14)*, 2014, pp. 1332–1340. doi: 10.5555/3044805.3045041.

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[19] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[20] E. Bingham et al., Pyro: Deep universal probabilistic programming. 2018. [Online]. Available: https://arxiv.org/pdf/1810.09538.pdf

[21] E. V. Bonilla, K. M. Chai, and C. Williams, "Multi-task Gaussian process prediction," in *Proc. Advances Neural Information Processing Systems*, 2008, pp. 153–160. doi: 10.5555/2981562.2981582.

Varun Suryan, Department of Computer Science, University of Maryland College Park. Email: suryan@umd.edu.

Nahush Gondhalekar, Qualcomm, Inc., San Diego, California. Email: nahushg@vt.edu.

Pratap Tokekar, Department of Computer Science, University of Maryland College Park. Email: tokekar@umd.edu.

By Xiaoxiao Du, Ram Vasudevan, and
Matthew Johnson-Roberson

Unsupervised Pedestrian Pose Prediction

A Deep Predictive Coding Network-Based Approach for Autonomous Vehicle Perception

Digital Object Identifier 10.1109/MRA.2020.2976313
Date of current version: 19 March 2020

Pedestrian pose prediction is an important topic, related closely to robotics and automation. Accurate predictions of human poses and motion can facilitate a more thorough understanding and analysis of human behavior, which benefits real-world applications such as human–robot interaction, humanoid and bipedal robot design, and safe navigation of mobile robots and autonomous vehicles. This article describes a deep predictive coding network (PredNet)-based approach for unsupervised pedestrian pose prediction from 2D camera imagery and provides experimental results of two real-world autonomous vehicle data sets. The article also discusses topics for future work in unsupervised and semisupervised pedestrian pose prediction and its potential applications in robotics and automation systems.

Background

Robots exist in an environment filled with moving people. A museum tour-guide robot such as in [1] and [2] is constantly surrounded by crowds of pedestrians, some who need

guidance regarding various exhibitions and some who may intentionally walk up to it or block its way to "test" the system. A self-driving car or a delivery robot can regularly find itself navigating a business district while trying to avoid a collision in the midst of heavy pedestrian traffic. In industrial settings, collaborative robots on assembly lines must allow humans to stand in their proximity and learn to recognize the intentions of human workers based on their movements and gestures to ensure safety and improve worker ergonomics and productivity [3], [4]. As these examples suggest, it is important for a robot or automation system to correctly understand and predict human poses to determine the intentions of humans and to make appropriate decisions and actions to avoid collisions and facilitate human–robot collaboration.

Our pose-detection system generates future frames based solely on a car-mounted camera and performs direct pose estimation per frame.

A human pose is commonly represented using a kinematic skeletal system of body joint locations, known as *keypoints* in the perception community. Given sequences of human poses extracted from collected data, such as images or videos in the past, the goal of pose prediction is to capture the underlying motion model of humans and make inferences about the pose and location of a person or multiple people at future time steps. There are a number of challenges associated with pedestrian pose estimation and prediction, including body part occlusion by other pedestrians or vehicles, human appearance and clothing variety, data collection range, camera viewpoint and lighting conditions, complex traffic scenes in the background, and stochasticity in human motion [5]–[7].

Various machine- and deep-learning (DL) methods have been proposed for human pose prediction based on sequences of motion data, including probabilistic dynamic models, physics-based models, supervised deep neural networks, and methods based on video generation. DL methods, such as recurrent neural networks (RNNs) and, in particular long short-term memory (LSTM) networks [8], have shown superior performance compared with traditional (nondeep) machine-learning and physics-based methods for pose prediction. However, supervised DL methods for pose prediction in the literature typically require accurate skeleton annotations of training sequences, which may be difficult or expensive to obtain. If prior annotations are imprecise or erroneous or if they contain missing frames, the performance of supervised DL methods may become subpar.

In this article, we developed an unsupervised method for pedestrian pose prediction based on car-mounted monocular camera videos collected by an autonomous vehicle. Our proposed system uses PredNet [9] as an unsupervised video-prediction method to generate future predicted frames and perform image-based pedestrian pose detection. This eliminates the need for accurate measurements and prior skeleton annotations, as are required by standard supervised pose-prediction methods.

PredNet-Based Unsupervised Pedestrian Pose Prediction

This section describes the foundation of our future frame-generation module, PredNet, and our proposed unsupervised pose-prediction pipeline.

PredNet

PredNet [9] is a deep neural network inspired by the concept of predictive coding from the neuroscience literature. The PredNet architecture consists of four basic layered units: recurrent representation (R), prediction ($\hat{A}$), the input convolutional layer (A), and error representation (E). Figure 1(b) shows the four-layer PredNet structure used in our system. Given a sequence of images (video data) as input, the target value of the lowest layer A_0 is set to be equal to the image sequence. PredNet goes through top-down and bottom-up passes to calculate the states of all units in each layer. First, at the top-most layer ($l = 3$) at time step t, the recurrent prediction state R_3^t is updated by passing a copy of the error signal at previous time step E_3^{t-1} and the recurrent prediction state at previous time step R_3^{t-1} through the convolutional LSTM units. For all subsequent layers $l = 0, 1, 2$, the recurrent prediction state R_l^t is updated according to E_l^{t-1}, R_l^{t-1}, and an upsampled copy of R_{l+1}^t.

At each layer, the prediction $\hat{A}_l^t$ is computed by a convolution of R_l^t followed by rectified linear unit (ReLU) for nonlinearity. The bottom layer $\hat{A}_0^t$ is additionally passed through a saturating linear unit (SatLU) to make sure the predictions of the next frame do not exceed the maximum pixel value. The error response at the bottom layer E_0^t is calculated by passing the difference between $\hat{A}_0^t$ and A_0^t (predicted image and actual image, respectively) through a ReLU and then dividing it into positive and negative errors and concatenation. For $l = 1, 2, 3$, the errors from the layer below E_{l-1}^t are passed through a convolution unit, followed by ReLU and max pooling, to become the input to the next layer A_l^t. PredNet can be trained end-to-end using gradient descent by minimizing the firing rates of the error neurons. The prediction state of the lowest layer $\hat{A}_0^t$ is returned as the prediction result for time step t. This process is repeated for all time steps $t = 1 \rightarrow T$ in the given image sequence. The pseudocode of the PredNet update process can be seen in algorithm 1 in [9].

Our pose-prediction network is based on, but not limited to, PredNet. In fact, the video-prediction module (shaded yellow in Figure 1) can be easily swapped with other video-prediction methods. We favor PredNet in our system as it has demonstrated superior performance for future frame prediction with moving backgrounds (as demonstrated in the original article on the KITTI benchmark suite [10]), which fits our goal of predicting pedestrian poses from data as typically observed by an autonomous vehicle-perception system. Also, PredNet is an unsupervised method, which has the advantage

of not requiring any annotation on the skeleton or mesh of people in the scene at prior sequences. In the following experiments, we used a four-layer PredNet structure with 3 × 3 convolutions and layer channel sizes of (3,48,96,192), the same parameter settings as described in [9] that produced effective results on natural image-sequence prediction.

Pose Prediction

Figure 1 provides an illustration of the full system for our proposed PredNet-based unsupervised pedestrian pose-prediction network. Given a sequence of red, green, and blue (RGB) images from a camera video as input, PredNet was used to predict pixel-level RGB values for the next frame. Then, based on predicted future frames, poses were extracted using pretrained human pose detectors. In our experiments, we used the OpenPose human keypoint detector [11] to identify 25 2D key points of pedestrians based on predicted future frames. OpenPose is available for image-based pose detection; see [22].

OpenPose is a deep convolutional neural network (CNN)-based human pose and keypoint detector that has shown state-of-the-art performance in real-time, skeleton-based pose detection based on videos and images containing multiple persons in various actions, including common pedestrian activities such as standing and walking. To further filter out false alarms caused by nonhuman objects, such as tree branches, utility poles, and billboards, another layer of human detection is applied to the OpenPose detection results. In our system, we applied a pretrained Mask R-CNN method [12] to estimate bounding box (bb) locations for humans and, thus, classify humans with nonhuman objects. Pre-trained weights on the Common Objects in Context (COCO) data set for Mask R-CNN human detection are available in [23]. The poses detected within the human bb are returned as the output of this system, i.e., the predicted skeleton pose for the generated future frame.

Our system offers a solution to unsupervised pose prediction inspired by human visual learning. As the car is in motion, human drivers are able to easily identify pedestrians and moving objects in the environment based on past observations. Additionally, human drivers do not necessarily require the exact metric of pedestrians in past sequences to know their pose (for example, walking in a direction that could intersect with the car). Our pose-detection system generates future frames based solely on a car-mounted camera and performs direct pose estimation per frame.

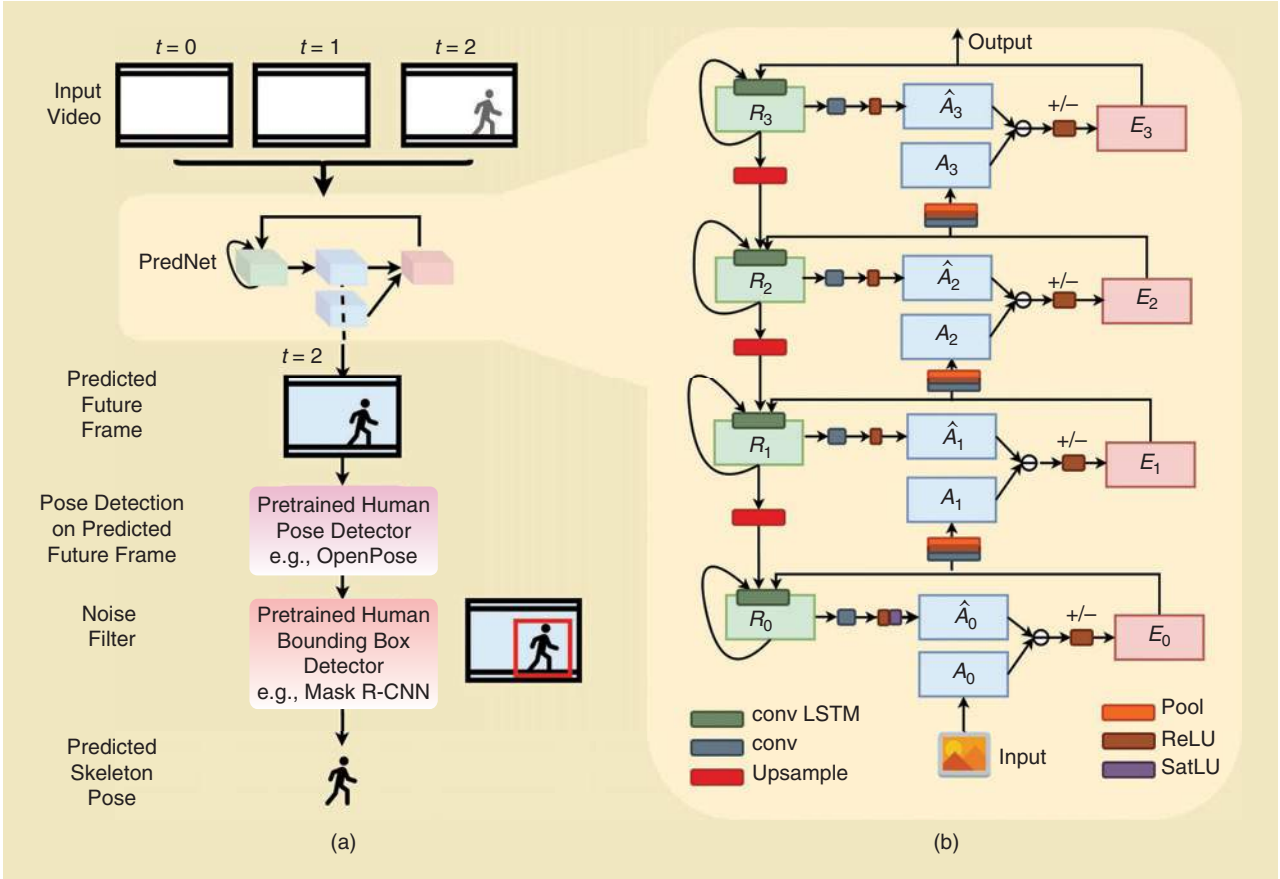


Figure 1. The system design for the proposed PredNet-based unsupervised pedestrian pose-prediction network. (a) The flowchart of the proposed system. For this illustration, the input video has three time steps ($t = 0, 1, 2$), but the input video can have more than three frames. (b) Details of the four-layer PredNet module where the RGB blocks represent the recurrent representation (R), prediction ($\hat{A}$) and input convolutional layer (A), and error representation (E). conv: convolutional; ReLU: rectified linear unit.

Figure 2. The PedX data-collection system on an autonomous vehicle parked at an intersection in downtown Ann Arbor, Michigan.

Table 1. The evaluation of next-frame predictions on JAAD and PedX data sets.

	MSE ($\times 10^{-3}$)	SSIM
PredNet on JAAD	**2.621 (0.587)**	**0.926 (0.017)**
Copy last frame on JAAD	6.602 (0.732)	0.904 (0.028)
PredNet on PedX	**1.244 (0.093)**	**0.962 (0.003)**
Copy last frame on PedX	1.759	0.955

These no longer require accurate measurements and annotations for historic skeleton sequences, which often further requires expensive data collection instruments such as 3D lidar, as required by prior work. (Lidar is an instrument commonly used for autonomous vehicle perception to measure precise ranges of objects. High-resolution lidars can be expensive.)

In addition, the PredNet module inherently incorporates implicit models of scene structures and movements of objects such as buildings, streets, and pedestrians as observed from a moving vehicle. Moreover, previous supervised pose-prediction methods often perform poorly, with imprecise measurements or occluded body parts in previous frames. However, our method naturally alleviates the missing-data problem since pose detection is performed per frame and does not rely on previous skeleton-detection results.

Experimental Results: Joint Attention for Autonomous Driving and PedX

We present the experimental results of our proposed pose-prediction approach on two real-world autonomous-vehicle-perception data sets, Joint Attention for Autonomous Driving (JAAD) and PedX. The JAAD dataset [13]–[16] contains 346 video clips collected from one of three different high-resolution monocular cameras in North America and Europe. The camera is positioned inside the car below the rear-view mirror. The frame rate is 30 frames/s (FPS). PedX [17] is a large-scale multimodal data set for pedestrians collected at intersections in downtown Ann Arbor, Michigan, in 2017. The data set provides high-resolution stereo images and lidar data with manual 2D ground-truth annotations. Data were captured using two pairs of stereo cameras and four Velodyne lidar sensors.

Figure 2 shows the data-collection configuration of the PedX data set. In the following experiments, we use data from only one of the forward-facing cameras with a 6-FPS frame rate. Both JAAD and PedX provide car-mount RGB videos in urban traffic scenes and observe pedestrian movements in the wild. The JAAD data set was collected from a moving car (thus, moving background and moving pedestrians), while the PedX data set was collected from a parked autonomous vehicle (stationary scene, moving pedestrians). The JAAD data set also includes a variety of traffic scenes, such as roads and parking lot, while PedX focuses on three urban intersections.

The JAAD data set contains 81,906 frames in total. We divided it into 30-frame sequences (1 s long) and used 85% of the sequences that were randomly shuffled for training, 5% for validation, and the remaining 10% for testing (8,280 frames). We used the same parameters of the PredNet model described in [9] and trained the model from scratch. The input images were downsampled to 256 × 456 to accommodate the computer memory while maintaining the aspect ratio. We also divided the PedX data set into 30-frame sequences (approximately 5 s long) and used the PredNet module trained from JAAD to test all 484 PedX sequences (14,520 frames).

Frame-Prediction Results

We first report the quantitative evaluation of the predicted future frames as compared to the actual collected video frames. Two metrics are used: the mean squared error (MSE) and the Structural Similarity Index Measure (SSIM) [18]. The MSE calculates the pixel difference between the predicted and actual frames (the lower the better), and the SSIM is correlated with perceptual similarity (the larger the better). We also report the results from the trivial solution of copying the last frame as baseline. As shown in Table 1, the PredNet module is effective in predicting the future frame and achieves better performance in both MSE and SSIM on both the JAAD and PedX data sets. The MSE is lower and the SSIM higher for the PedX data set overall since the background scene in PedX data is fixed while the background changes as the car moves in the JAAD data set. In the table, the best performance is in bold, and the standard deviation across three runs is presented in parentheses.

Figure 3 provides visual examples of the future frame-prediction results of PredNet on sample sequences from JAAD and PedX. We observed that the frames predicted by PredNet (rows 2 and 5) were very similar visually to those from the actual video clips (rows 1 and 4). The frame-prediction step was also capable of extrapolating the sky and textures of other

objects and making fairly accurate predictions, such as when cars move in or out of the frame.

Pose-Prediction Results

The output of PredNet frame prediction is of shape (N, T, 256, 456, 3), where N is the number of sequences and T is the sequence length (T = 30). To ensure optimal performance of the OpenPose keypoint detector, we used the *pyplot.savefig* function in *matplotlib* in Python to save the frame predictions with dots per inch equal to 1,000, resulting in 2,791 × 4,967 images for the JAAD data set and 2,687 × 3,645 for the PedX data set for pose prediction. We call this process *upsampling*. The OpenPose was then applied to the upsampled (saved high-resolution) images to detect pedestrian keypoints.

Since the JAAD data set did not provide skeleton annotations, we used OpenPose pose-detection results filtered by

Figure 3. An example of sample frames from the JAAD and PedX video sequences, the PredNet future frame-prediction results, and the pose-prediction results of our proposed system. The top six rows show next-frame-prediction results. The first three are from the JAAD data set, where two pedestrians are walking in a parking lot, and the middle three are from the PedX data set, where a pedestrian is walking across the crosswalk at an intersection. The bottom three rows show the GT poses from the JAAD data set, the multiple time-step frame-prediction results, and our multiple time-step pose-prediction results. The original video sequences are 30 frames long, which is equal to 1 s in JAAD and 5 s in PedX. Here, we plot $t = 1, 7, 13, 19, 25$, which corresponds to approximately 0.2-s spacing in the JAAD sequence and 1-s spacing in PedX. For more results containing consecutive frames, please see the accompanying video, which can be found as supplemental material for this article in IEEE *Xplore*.

Table 2. The evaluation of pose prediction on the JAAD data set. The equal sign means "corresponds to."

	RMSE-*x*	RMSE-*y*	RMSE-*xy*
Copy last frame	102.534 (165.329)	89.801 (137.015)	101.561 (148.418)
Frame difference	92.456 (151.035)	78.352 (117.988)	89.816 (132.828)
LSTM-3LR	72.366 (125.301)	57.324 (95.502)	68.374 (109.531)
Ours	**29.347 (39.818)**	**19.088 (21.685)**	**26.516 (30.623)**
Ours: bb	0.145 (0.144) ≙ 14.5 (14.4) cm	0.040 (0.042) ≙ 8.0 (8.2) cm	—

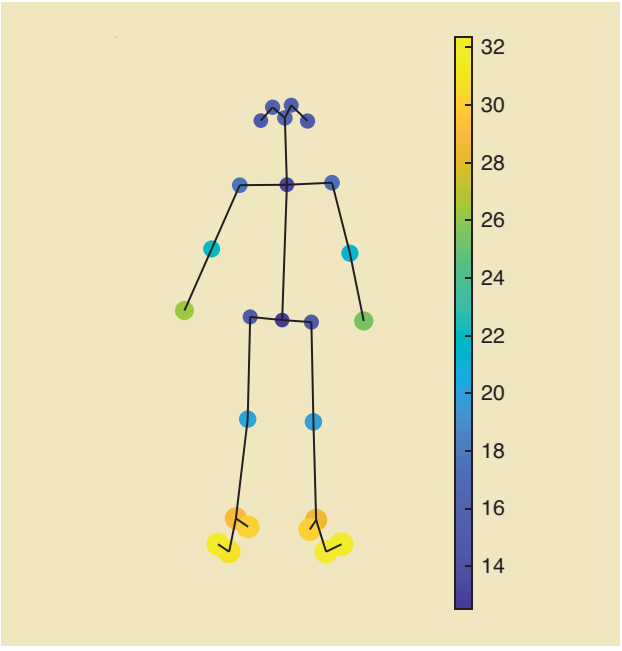


Figure 4. The predicted skeleton RMSE results of our proposed method in pixels. Each circle represents a keypoint (a joint) on the body. The 25 keypoints include left and right shoulders, elbows, wrists, hips, knees, ankles, eyes, ears, feet, toes, heels, nose, neck, and center of hip. The colors of the circle represent the mean RMSE results across three runs; yellow shows higher error and dark blue shows low error. The size of the circle represents the standard deviation value at each joint; the larger the circle, the higher the standard deviation.

Mask R-CNN, based on the actual video frames, as the ground truth (GT). We then compared the pose-detection results from our proposed approach (based on the predicted future frames) to the GT and calculated the root-mean-square-error (RMSE) results. In PedX, we also report the RMSE results between our pose-prediction results and the manual 2D annotations. We compared our approach with two previous supervised DL pose-prediction methods, LSTM-3LR [6] and a frame-difference method adapted from Bio-LSTM (Bio-LSTM-L_c) [19]. For LSTM-3LR, we used a stacked, three-layer LSTM with 64 units and 250 training epochs in our experiments. For the frame difference method, we used the most prominent gait feature, the gait periodicity loss (L_c) in the Bio-LSTM objective function, with a two-layer, stacked LSTM RNN for pose prediction. We also reported the RMSE results from the naïve baseline by copying poses from the last frame.

Table 2 presents the pose-prediction results of the JAAD data set. In the table, the best performance is in bold, and the standard deviation across three runs is presented in parentheses. The RMSE-*x*, RMSE-*y*, and RMSE-*xy* columns correspond to the RMSE results on the *x*-axis (width of the image), *y*-axis (height of the image), and average of both, respectively. The lower the RMSE results, the better the prediction performance. The first four rows show the RMSE results in pixels. The bottom row shows the percentage of pose-prediction error normalized by bb sizes, along with their corresponding error values in metric space in centimeters.

Our proposed approach yields smaller RMSE mean and standard deviation values than comparison methods in both the *x* and *y* (width and height) directions, indicating its effectiveness and consistency. The two supervised comparison methods, LSTM-3LR and Bio-LSTM, were both originally developed for 3D skeleton/mesh prediction and require accurate full-body annotations in previous frames. In this experiment, where 2D key points detected in previous frames are often imprecise or missing due to occlusion, supervised methods produced a higher error. The trivial solution of directly copying the last frame yielded the highest error in both height and width directions, which is as expected since this baseline method does not account for pedestrian motion between frames at all. Our proposed pose-prediction method, on the other hand, produces pose estimations based on the predicted future frame and does not require previous skeleton-detection results, which resulted in significantly fewer errors and more consistent pose-prediction results.

Figure 4 plots the skeleton RMSE results in pixels at each joint. As shown, the body center has the lowest error whereas the extremities (hands and feet/heels) show both larger mean and standard deviation error values. This is understandable as the hands and feet are the most flexible parts of the human body and also the most easily occluded; they are, therefore, the most difficult to predict. To further translate the image-level evaluation (in pixels) to metric space (in meters), we computed the pose-prediction errors normalized by the heights and widths of the corresponding bbs detected by Mask R-CNN. The bottom row in Table 2 shows the percentage error with respect to human bb sizes. Assuming that a person is approximately 2 m tall and 1 m wide, our average pose error in the physical space is approximately 14.5 cm in the lateral direction and 8 cm height-wise.

Table 3. The evaluation of pose prediction on the PedX data set.

	RMSE-*x*	RMSE-*y*	RMSE-*xy*
OpenPose actual versus GT	39.616 (49.049)	31.014 (45.166)	38.634 (52.428)
Ours versus OpenPose actual	58.235 (33.514)	33.486 (22.751)	49.689 (24.653)
Ours versus GT	68.781 (47.798)	43.546 (40.845)	62.000 (45.836)
Ours: bb	0.251 (0.148) ≙ 25.1 (14.8) cm	0.063 (0.055) ≙ 12.6 (11) cm	—

We used the PredNet module trained from JAAD to test on the PedX data set. Table 3 presents the pose-prediction results on PedX. The top row shows the RMSE results between pose detected by OpenPose on the actual video clip and the manual GT annotation. The second row shows the RMSE results between our proposed method and poses detected by OpenPose on the actual video clip. The third row shows the RMSE results between the OpenPose pose on the predicted video clip (i.e., our proposed method) and manual GT annotation. The RMSE-*x*, RMSE-*y*, and RMSE-*xy* columns correspond to the RMSE results on the *x*-axis (width of the image), *y*-axis (height of the image), and average of both, respectively. The lower the RMSE results, the better the prediction performance. The first three rows show the RMSE results in pixels. The bottom row shows the percentage of pose-prediction error normalized by bb sizes, along with their corresponding error values in metric space in centimeters.

Since the PedX data set contains manual GT annotations, we compared our pose-prediction results to both the manual GT and the poses detected by OpenPose on the actual video clip. We also evaluated the accuracy of the OpenPose keypoint detector compared with the manual GT. Compared with Table 2, the PedX prediction results have slightly higher RMSE than when tested on the JAAD data set due to 1) the frame rate being five times higher in JAAD than PedX and 2) the PedX scenes having never been observed in training. Nevertheless, using the model trained on JAAD can still make pose prediction on previously unseen PedX contexts with a relatively small error (25.1 cm in width and 12.6 cm in height).

Figure 3 shows visual examples of pose-prediction results for sample sequences from JAAD and PedX. The predicted skeleton poses were overlaid with the predicted RGB images in rows 3 and 6. Our pose-prediction system can correctly detect and predict human poses in the scene based on the predicted future frame from PredNet. Particularly, when a pedestrian was partially occluded because of the motion of the ego vehicle (the vehicle with the data-collection camera), as shown in the final column of row 3 in Figure 3, our pose-prediction process was still able to detect the correct pose and mark that the legs are occluded (not plotted on the image).

Multiple Time-Step Prediction

This section presents the experimental results of multiple time-step prediction (MTP) in the future. Given 10 actual observed frames in the JAAD data set, our PredNet-based, video-generation module extrapolates the next 20 time steps, in which the 11th frame prediction was fed back to the network as input to generate the 12th frame prediction and so on. Then, our pose-prediction module performed pose estimation on the predicted frames. Since R_l^0 and E_l^0 in PredNet were initialized to zero, the prediction at the initial time step was spatially uniform and therefore not considered in our analysis.

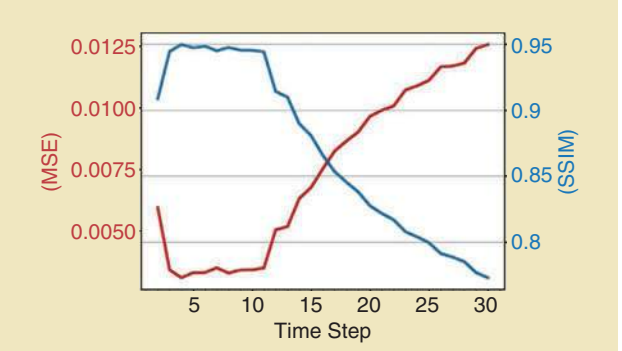


Figure 5. The multiple time-step frame-prediction results on the JAAD data set. The *x*-axis marks the time steps (the final 20 time steps were extrapolated from the MTP process), and the *y*-axes show the frame MSE (left, red) and SSIM (right, blue).

Figure 5 shows the multiple time-step frame-prediction results evaluated by the metrics MSE and SSIM. Recall that the MSE calculates the pixel difference between the predicted and actual frames (the lower the better) and the SSIM is correlated with perceptual similarity (the larger the better). The second time step produced high frame error due to the fact that there is no motion information available yet in the sequence, causing the frame reconstruction to be blurry. In the next few time steps, PredNet learned the underlying dynamics in the motion sequence, and the predicted frames better matched the input (actual) frames, producing relatively low MSE and high SSIM results between time steps 3 and 10. This observation is consistent with the description in [9]. Since previous predictions were used as

Our proposed approach does not rely on manually defined models and distributions but instead performs frame-based 2D pose prediction.

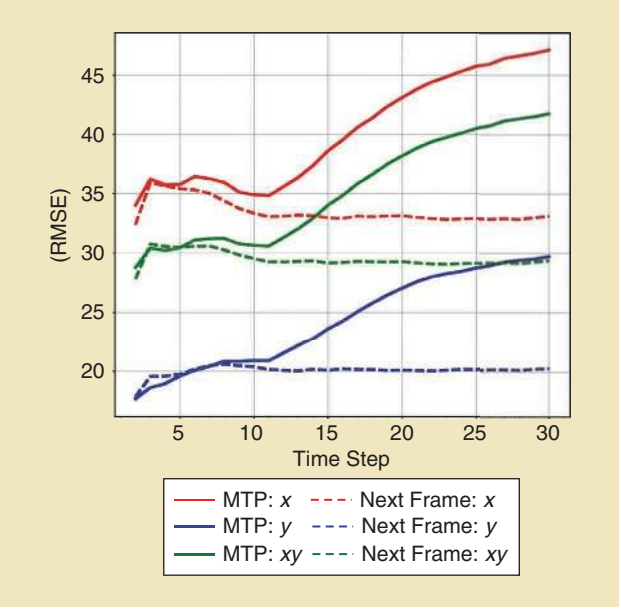


Figure 6. The multiple time-step pose-prediction results on the JAAD data set. The *x*-axis marks the time steps (the last 20 time steps were extrapolated from the MTP process), and the *y*-axis marks the OpenPose skeleton joint RMSE in pixels. The solid lines show the MTP results, and the dashed lines show the next-frame prediction results. The red, blue, and green colors represent the RMSE results on the *x*-axis (width of the image), *y*-axis (height of the image), and average of both, respectively.

input to extrapolate future frames, after time step 10 (in the MTP process), the frame errors were higher than next-frame-prediction results, and the MSE increased (and SSIM decreased) over time.

Figure 6 shows the pose-prediction results for time steps 2–30 based on the MTP frame predictions, with comparison to the next-frame prediction results. In time steps 2–10, where the actual frames were used as input, the MTP process is the same as the next-frame prediction and they yield comparable RMSE results. In time steps 11–30, where the previous prediction results were recursively iterated as inputs to generate future frames, the MTP error is higher than next-frame prediction and increased significantly over time as the noise overcame the system. Qualitatively, the bottom three rows in Figure 3 show the MTP frame- and pose-prediction results of a sample sequence from the JAAD data set. As time increases, the MTP model was still able to capture certain motions in the scene, such as the movements of clouds and surrounding cars in the parking lot. However, the frame predictions eventually became more and more blurry due to the accumulation of noise and uncertainty, thus causing inaccurate and missed pose-detection results (such as the person on the right-hand side) in its extrapolations after the first 10 time steps.

The PredNet training step can be omitted when a pretrained PredNet model is applied to other data sets, such as in our PedX experiments.

Discussion

The quantitative and qualitative results presented previously show that our proposed PredNet-based unsupervised pose-prediction approach can produce accurate results. Using JAAD and PedX, two real-world data sets in autonomous driving contexts, we show that our proposed approach is applicable to a variety of driving scenes and the frame-prediction models can be generalized to previously unseen environments.

Our PredNet-based frame-prediction step produces realistic future frames, accounting for both vehicle and pedestrian motions. We also observed that parts of the predicted frames can be blurry later in the sequences (when t increases) due to downsampling, particularly in small-textured regions, such as the pedestrian's facial features, or unfamiliar regions, such as when the red car entered the scene in the middle of the sequence in Figure 3. In PedX, the blur effect seemed more prominent due to the fact that we used a PredNet model pretrained on JAAD and the PedX environments were not learned in training; moreover, the spacing between frames was longer due to a low frame rate. However, such a blur effect did not have a significant impact on the pose-prediction step, and we observed that our pose-prediction module can still successfully detect the skeleton joint locations of pedestrians based on predicted future frames.

Our proposed approach does not rely on manually defined models and distributions but instead performs frame-based 2D pose prediction. Our approach transforms the standard supervised learning problem, which requires full-body pose annotations in prior sequences, into an image-data-driven, unsupervised framework. Our frame-prediction step realistically predicts scene dynamics as well as the relative motion between pedestrians (or moving objects) and background, and our pose-detection step naturally handles occlusion based on the OpenPose detector. Moreover, our approach produces both the RGB future frame as well as the future pose, which makes the pose visually interpretable within the frame.

One of the challenges associated with such frame-based pose-prediction methods is that the performance of the pose prediction depends significantly on the accuracy of the human detector, i.e., the OpenPose and Mask R-CNN algorithms used in our system. Although highly effective most of the time, human detectors can occasionally produce false alarms, such as mistaking a windshield wiper or a billboard painting as a pedestrian or misidentifying tree branches or dark shadows as a person, especially under difficult lighting conditions, such as sun glare (Figure 7). It is also challenging at times to accurately identify multiple people moving in a crowd at a distance. This can be solved by better positioning the camera, such as outside the windshield wiper or on top of the car to avoid potential interference of the wipers, using more vigorous noise-filtering approaches, and exploring

alternative sensor modalities to compensate for the lighting and other environmental effects on RGB cameras.

Analysis of Computation Time

Our proposed network was implemented in Python 3.6 using the Keras framework [20]. The PredNet module was trained on a desktop computer with an Intel Xeon 2.10-GHz CPU with four NVIDIA TITAN X graphics processing units and 128-GB memory. The memory requirement can be reduced if the input video is of lower resolution or if the sequence length is smaller. After PredNet was trained, the pose detection module was performed on a desktop computer with an Intel i7 3.60-GHz CPU with two NVIDIA TITAN X GPUs.

Figure 8 shows a breakdown of our end-to-end computation time for the JAAD data set experiments (next-frame prediction). PredNet training and upsampling are the two most time-consuming tasks. However, the PredNet training step can be omitted when a pretrained PredNet model is applied to other data sets, such as in our PedX experiments. The training time also varies if the input data size changes. The inference time for the PredNet testing stage is approximately 20 ms per frame. The OpenPose pose-detection step is also quite fast, taking approximately 418 ms/frame. In our current implementation, since both our pose detector and noise filter are image based, we saved all upsampled RGB images to disk. Therefore, the upsampling and Mask R-CNN filter steps include the long disk read and write time. Future work will include investigating non-image-based pose-prediction methods. Additionally, the pose-detection and filtering steps can be trivially parallelized.

Figure 7. Examples of OpenPose false alarms, where (a) a windshield wiper and a person on the billboard were misidentified as pedestrians and (b) tree branches were misidentified as pedestrians due to dark shadows under glare.

Future of Pose Prediction and Conclusions

This article presented an unsupervised pedestrian pose-prediction system based on PredNet for autonomous vehicle perception. Our system combines video-generation approaches, such as PredNet, and frame-based pose detectors, such as OpenPose and Mask R-CNN, and shows effective future pose-detection performance on real-world autonomous vehicle applications.

There are several new research directions related to this research. This article focused on autonomous driving-perception applications, but human pose prediction can be widely applied in various robotics and automation applications, including virtual reality, sports and artist posture analysis, and medical assistance. In many robotics and automation applications with abundant perception data, accurate annotations are expensive and difficult to obtain. Therefore, it is necessary to further develop unsupervised and semisupervised learning methods given sparse and imprecise labels for pedestrian pose prediction in various contexts. Particularly, with human pose analysis and sequence prediction, additional investigations can be conducted to incorporate more realistic spatial, temporal, textural, semantic, and biology-derived constraints in the learning model [21].

Furthermore, mobile robots and automation systems are interacting with the real world. In autonomous driving, pedestrians and vehicles are constantly making decisions based on their interactions and finding the balance between achieving certain goals and avoiding risk and collision. Interesting future work will include incorporating pedestrian–pedestrian and pedestrian–vehicle interactions in pose and trajectory prediction and using such prediction results for activity inference, low-level decision making (such as stop/go), and path planning.

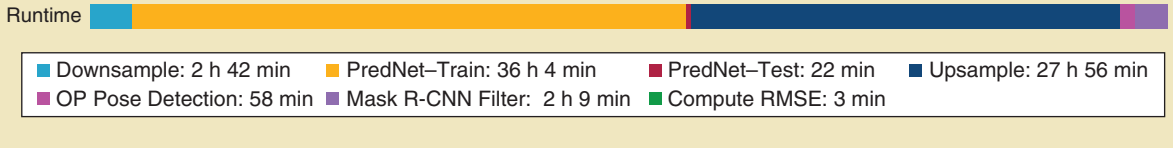


Figure 8. Analysis of the computation time for the JAAD data set experiments, trained and validated on 73,626 frames and tested on 8,280 frames. OP: OpenPose.

Our current pose-prediction performance depends on the accuracy of the frame prediction in RGB image space, which may become more challenging as weather and lighting conditions change. Future work will include investigating human pose prediction based on alternative or complementary sensors, such as depth and thermal imaging cameras. With the development of advanced computing units, network architecture design and automated parameter settings for improving the efficiency and scalability of DL approaches can be further studied as well.

Finally, it is essential to develop evaluation metrics for pose prediction, particularly when GT is not available or imprecise. An open problem remains as to how to evaluate whether the predicted pedestrian poses are natural and realistic and whether they are in accordance with traffic rules and regulations as well as ethical considerations.

Acknowledgments

This work was supported by a grant from the Ford Motor Company via the Ford–University of Michigan Alliance under award N022884.

References

[1] W. Burgard et al., "The interactive museum tour-guide robot," in *Proc. 15th Nat./10th Conf. Artificial Intelligence/Innovative Applications of Artificial Intelligence*, 1998, pp. 11–18. doi: 10.5555/295240.295249.

[2] S. Thrun et al., "MINERVA: A second-generation museum tour-guide robot," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 1999, vol. 3, pp. 1999–2005. doi: 10.1109/ROBOT.1999.770401.

[3] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, Nov. 2018. doi: 10.1016/j.mechatronics.2018.02.009.

[4] W. Kim et al., "Adaptable workstations for human-robot collaboration: A reconfigurable framework for improving worker ergonomics and productivity," *IEEE Robot. Autom. Mag*, vol. 26, no. 3, pp. 14–26, 2019. doi: 10.1109/MRA.2018.2890460.

[5] I. Petrov, V. Shakhuro, and A. Konushin, "Deep probabilistic human pose estimation," *IET Comput. Vis.*, vol. 12, no. 5, pp. 578–585, 2018. doi: 10.1049/iet-cvi.2017.0382.

[6] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2015, pp. 4346–4354. doi: 10.1109/ICCV.2015.494.

[7] J. Bütepage, H. Kjellström, and D. Kragic, "Anticipating many futures: Online human motion prediction and generation for human-robot interaction," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018, pp. 1–9. doi: 10.1109/ICRA.2018.8460651.

[8] 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.

[9] W. Lotter, G. Kreiman, and D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," in *Proc. Int. Conf. Learning Representations*, 2017. [Online]. Available: https://openreview.net/pdf?id=B1ewdt9xe

[10] 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.

[11] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published. doi: 10.1109/TPAMI.2019.2929257.

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

[13] I. Kotseruba, A. Rasouli, and J. K. Tsotsos, Joint attention in autonomous driving (JAAD). 2016. [Online]. Available: https://arxiv.org/abs/1609.04741

[14] A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "Agreeing to cross: How drivers and pedestrians communicate," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2017, pp. 264–269. doi: 10.1109/IVS.2017.7995730.

[15] A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "Are they going to cross? A benchmark dataset and baseline for pedestrian crosswalk behavior," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 206–213. doi: 10.1109/ICCVW.2017.33.

[16] A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "Understanding pedestrian behavior in complex traffic scenes," *IEEE Trans. Intell. Veh.*, vol. 3, no. 1, pp. 61–70, 2017. doi: 10.1109/TIV.2017.2788193.

[17] W. Kim et al., "PedX: Benchmark dataset for metric 3-D pose estimation of pedestrians in complex urban intersections," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1940–1947, Apr. 2019. doi: 10.1109/LRA.2019.2896705. [Online]. Available: http://pedx.io/

[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] X. Du, R. Vasudevan, and M. Johnson-Roberson, "Bio-LSTM: A biomechanically inspired recurrent neural network for 3-D pedestrian pose and gait prediction," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1501–1508, Apr. 2019. doi: 10.1109/LRA.2019.2895266.

[20] F. Chollet et al., "Keras: The Python deep learning library," Keras, 2015. [Online]. Available: https://keras.io

[21] F. Bonsignorio, E. Messina, A. P. del Pobil, and J. Hallam, Eds., "The road ahead-final remarks," in *Metrics of Sensory Motor Coordination and Integration in Robots and Animals: How to Measure the Success of Bioinspired Solutions with Respect to Their Natural Models, and Against More 'Artificial' Solutions?*, vol. 36, 2020, pp. 181–185. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-030-14126-4.pdf#page=201

[22] CMU Perceptual Computing Lab/OpenPose. Accessed on: Feb. 27, 2020. [Online]. Available: https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/quick_start.md#quick-start

[23] Mask_RCNN_Humanpose. Accessed on: Feb. 27, 2020. [Online]. Available: https://github.com/Superlee506/Mask_RCNN_Humanpose/releases

Xiaoxiao Du, Department of Naval Architecture and Marine Engineering, University of Michigan, Ann Arbor. Email: xiaodu@umich.edu.

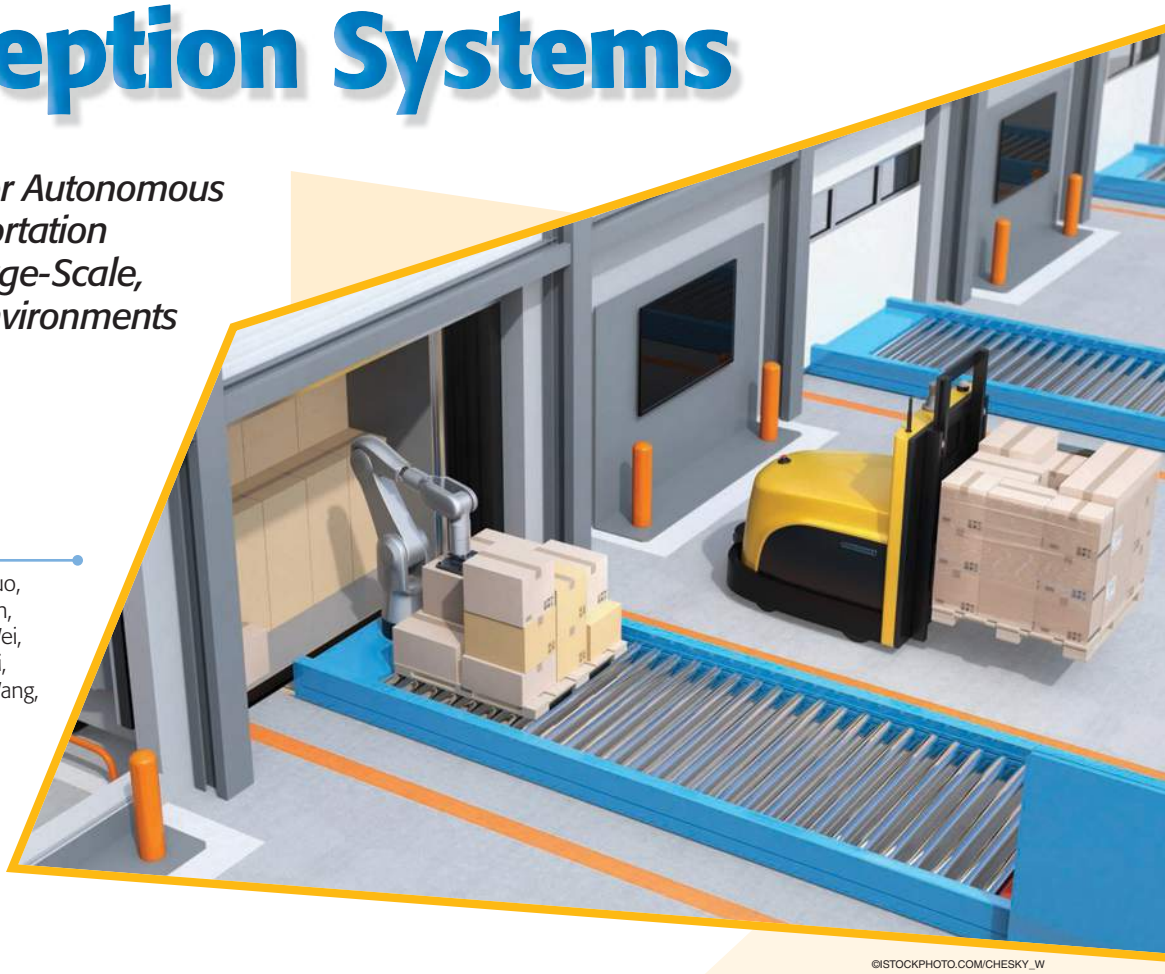
Ram Vasudevan, Department of Mechanical Engineering, University of Michigan, Ann Arbor. Email: ramv@umich.edu.

Matthew Johnson-Roberson, Department of Naval Architecture and Marine Engineering, University of Michigan, Ann Arbor. Email: mattjr@umich.edu.

Deep Learning-Based Localization and Perception Systems

Approaches for Autonomous Cargo Transportation Vehicles in Large-Scale, Semiclosed Environments

By Zhe Liu, Chuanzhe Suo, Yingtian Liu, Yueling Shen, Zhijian Qiao, Huanshu Wei, Shunbo Zhou, Haoang Li, Xinwu Liang, Hesheng Wang, and Yun-Hui Liu



Vehicles capable of delivering heavy cargoes are a major means for the promotion of social productivity and are widely applied in industry. Even though self-driving technologies have been studied for a few decades and several successful applications have been demonstrated, autonomous industry vehicles are currently applied only in some specific scenarios with very low speed and fixed routes and are typically implemented in the indoor closed area of small-scale warehouses. In this article, we introduce the main perception and localization approaches of autonomous cargo transportation vehicles for industrial applications. We also demonstrate how these technologies can enable autonomous cargo transportation in the Hong Kong International Airport.

Digital Object Identifier 10.1109/MRA.2020.2977290
Date of current version: 8 April 2020

Background

Currently, most vehicles used in industry to deliver heavy cargo are operated by human drivers. However, the cargo transportation companies at the Hong Kong International Airport, for example, currently have a 25% shortage of drivers. In addition, the salary for logistics vehicle drivers has increased 30% in the last few years, and the proportion of employees aged 40 and older in the logistics industry has increased from 51.50% in 2008 to 59% in 2018 (https://www.censtatd.gov.hk/home/index.jsp). The increasing need for but serious shortage of human drivers, their rapidly growing wages, and high-risk working environments necessitate that manual-driving industrial vehicles be replaced by autonomous ones, which will help greatly improve the efficiency and reduce the operational cost of cargo transportation in airports, railway stations, harbors, and industrial warehouses.

Even though self-driving technologies have been studied for a few decades [1] and several successful applications have been demonstrated in urban areas [2], rural roads [3], and agricultural environments [4], autonomous industry vehicles are currently applied only in some specific scenarios with very low speed and fixed routes and are typically implemented in an indoor closed area of small-scale warehouses [5]. Compared to on-road self-driving cars, autonomous industry vehicles have several important differences. First, in many tasks, the localization and navigation accuracy must be as high as 20–50 mm, which is much higher than the lane-level accuracy required for typical on-road self-driving cars. Second, the industrial environment is semiclosed but can be very large scale, highly dynamic, and feature sparse, which brings great difficulties to place-recognition, loop-closure, and mapping tasks. Third, vehicles may need to move in more unorganized environments. To meet stringent safety requirements, the sensing system should be more accurate and robust. These differences present new challenges in developing the mapping, localization, and perception systems of industrial vehicles.

In this article, we introduce the main perception and localization technologies of the proposed autonomous transportation vehicles used to achieve full autonomy in cargo transportation in large-scale airport terminals and tarmac environments. The main contribution of the article is to present a system-level overview of the deep learning-enhanced, large-scale localization and mapping approaches as well as multisensor fusion-based perception algorithms. We also demonstrate how these technologies can enable autonomous cargo transportation in large-scale airport environments.

Application Scenario and Related Work

As shown in Figure 1, in this article, we consider the industrial vehicle that delivers cargoes and materials in airport, harbor, and automated warehouse environments. Typically, these environments are semistructured and unorganized and have a very large-scale area with both indoor and outdoor routes. While GPS and inertial navigation systems are widely used in autonomous vehicles to present real-time location information, they still cannot fulfill the autonomous navigation requirement, especially in industrial applications. Firstly, the positioning error of the sensor system mentioned previously cannot, in general, fulfill the localization accuracy required by the high-precision navigation system in cargo transportation vehicles. Secondly, GPS often does not work because of the shelter of houses and overpasses and also suffers from various signal interference in industrial environments. Another important issue is the sparse static structured information. Since airport, harbor, and warehouse environments usually contain a number of dynamic objects (such as airplanes and vehicles) and temporarily stored entities (such as containers, pallets of goods, and industrial materials), autonomous navigation in such a highly dynamic environment requires higher localization accuracy than that of the current GPS-based inertial guidance systems for household vehicles. These factors bring great difficulties to the construction of the environment map and estimation of the vehicle pose.

In recent years, simultaneous localization and mapping (SLAM) is increasingly promoting the development of the fully autonomous navigation of mobile robots and vehicles. A typical SLAM system estimates the real-time vehicle pose and builds/updates the environment map simultaneously. In practice, for a semiclosed industry scenario, building a complete map beforehand is recognized as a more effective method due to its strong robustness to dynamic objects, high precision in the presence of off-line refinements, and better localization performance over a long period of time.

With a prior environment map, autonomous vehicles have the ability to plan global routes and local paths, estimate their locations with onboard sensor information, and achieve autonomous navigation. Current solutions include vision [6] and laser point cloud-based approaches [7]. Vision-based solutions are unreliable in industrial environments due to the limited field of view, motion blur, and nonrobustness to variable weather and illumination conditions. Advanced solutions based on laser point cloud are more popular in practical applications. A typical solution is to extract handcraft features (ground, edge, and planar points) from laser data and estimate the vehicle pose with odometry data and feature-matching results. Then parallel localization and mapping can be achieved by implementing the Iterative Closest Point (ICP)-based graph-optimization framework [7]. Recently, to enhance the generalization ability of extracted laser features and improve algorithm robustness, deep learning-based approaches have become more and more popular [2]. However, in large-scale, dynamic environments with sparse static features and large cumulative errors, mapping and localization tasks are still very challenging [8]. In this article, we present a point cloud description network to extract discriminative and generalizable laser features and achieve large-scale place recognition tasks. Based on this, we developed a deep learning-based integrated framework for mapping and localization tasks in large-scale dynamic environments.

In industrial environments, autonomous vehicles need to share the drivable area with human workers, forklifts, and other vehicles. The driving environment is usually crowded and narrow. Safety concern is the first priority, especially in airport and harbor environments. Vehicles need to be able to detect surrounding dynamic entities and track their poses. These help the autonomous vehicle have a deeper understanding of the surrounding scenario, predict short-term future information, and make sophisticated precedence decisions. For example, a detection and tracking system that estimates the state of surrounding pedestrians and vehicles provides the motion control system the capability of avoiding obstacles, thus ensuring safety. Computer vision-based approaches incorporated with deep learning techniques have been widely studied and have achieved remarkable performance in object detection and tracking tasks [15]. However, the illumination condition of industrial environments changes wildly, so we developed a multimodel information-fusion-based perception

system that integrates the vision information and laser data to fulfill the robust detection and accurate tracking requirements.

Advanced Sensing and Control System

As shown in Figure 1(a), we developed a sensor system to achieve robust perception and accurate localization functions. An Ouster OS1-64 laser (120-m range) mounted on the top allows for collecting surrounding 3D point cloud data, and two wide-angle, high-resolution cameras (1,920 × 1,200 resolution) are utilized to collect visual information in both the front- and rear-view images. Real-time kinematic GPS, inertial measurement units (IMUs), and external encoders are used to generate real-time odometry information (100-Hz frequency) using an extended Kalman filter (EKF)-based data fusion approach. In addition, a 2D safety laser is equipped on the front of the vehicle to eliminate sensing blindness and avoid obstacles on the ground.

The hardware architecture of the proposed vehicle is shown in Figure 2. A long-range-based wireless communication network is built to assign transportation tasks to vehicles, obtain on-site information, monitor vehicle status, and achieve remote control in any emergency. As shown in shown in Figure 3, the core control system includes the high-level system run in an industry computer with the Robot Operating System [host industrial PC (IPC), Nuvo-6108G, GTX-2080Ti GPU] and the low-level system run in the programmable logic controller (PLC) with the real-time operating system. The high-level system receives tasks from the task planning system/local human–machine interface (HMI) and generates global paths by searching the route map. In the meantime, it updates the

(a)

(b)

(c)

Figure 1. Application scenarios for autonomous transportation: (a) an autonomous industrial vehicle and its sensor system; (b) autonomous cargo transportation; and (c) large-scale industrial environments with sparse static structured information, a large number of dynamic objects, and varied weather and illumination conditions. IMUs: inertial measurement units.

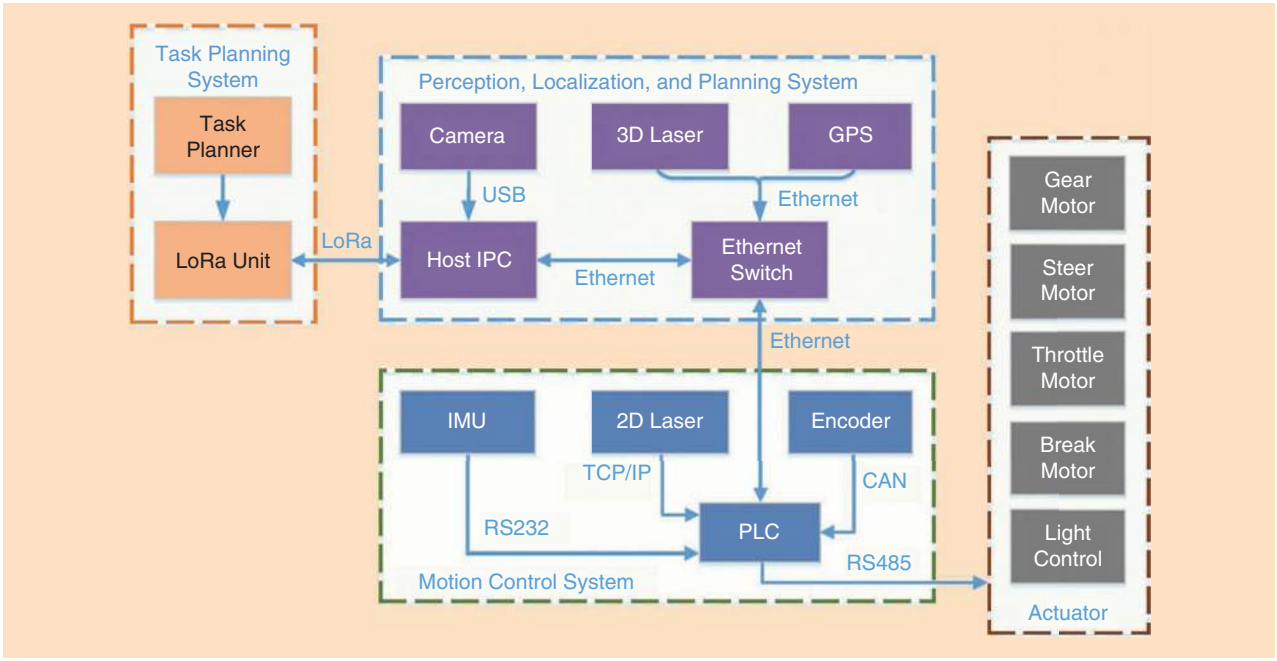


Figure 2. The hardware architecture of the proposed autonomous vehicle system. LoRa: long-range radio; CAN: controller area network.

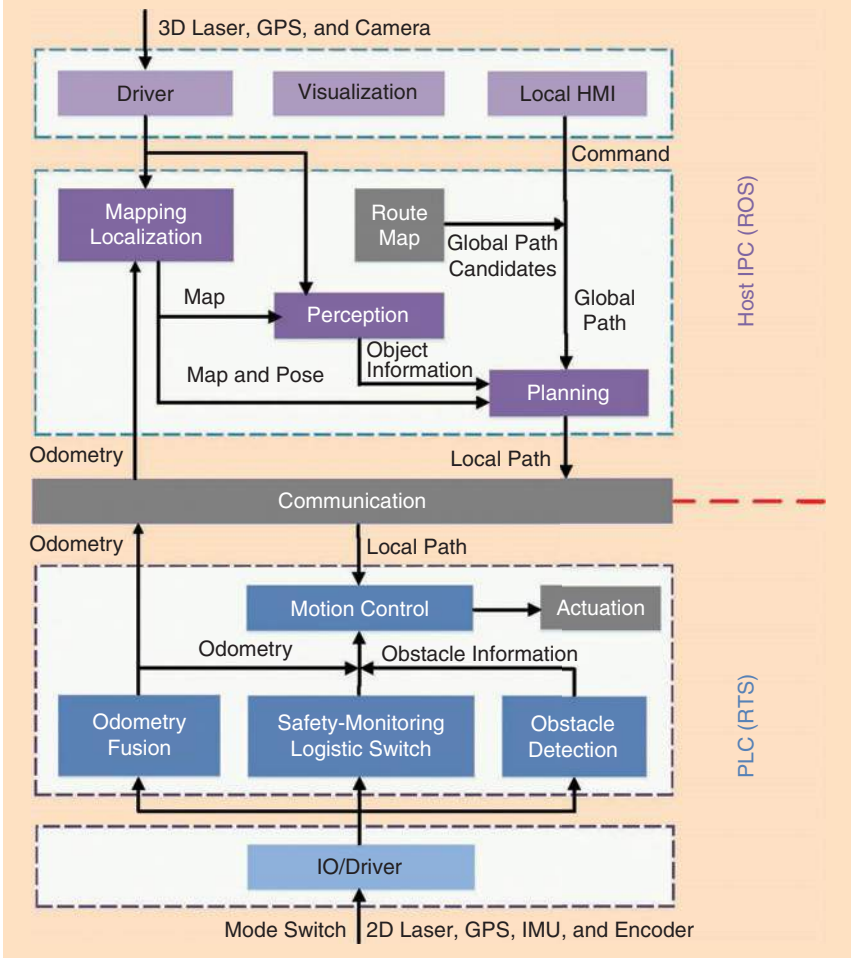


Figure 3. A schematic overview of the core system. Info: information; IO: input-output; HMI: human–machine interface; ROS: Robot Operating System; RTS: real-time operating system.

environment map, estimates vehicle pose, and detects surrounding objects by fusing the 3D laser data, visual information, and odometry data. Then the high-level system finally generates feasible local paths (10-Hz frequency) and sends them to the low-level system. In the low-level system, the local paths are fed into the velocity and steering control modules for proportional–integral–derivative control of the steering, throttle, and braking motors. The velocity control module provides an appropriate velocity value by comprehensively considering the local path curvature and detected obstacles. The steering control module accepts as input the local path, vehicle pose, and velocity value recommended by the velocity controller and then computes steering angle as output commands utilizing the Hoffmann controller [9].

Deep Learning-Enhanced, Large-Scale Mapping and Localization

Localization is one of the fundamental technical issues for autonomous vehicles because it is essential to establishing the feedback loop for path-tracking control and autonomous navigation. To achieve accurate

localization, it is necessary to build a prior map of the whole environment, and the mapping accuracy limits the errors incurred in estimating the pose of the vehicle. In this article, we present a deep learning-enhanced mapping and localization framework for large-scale industry environments. As shown in Figure 4, the proposed framework includes the following components:

- *Large-scale point cloud learning*: We first extract local features from the raw 3D laser data to reveal the local 3D structure around each point. An entropy minimization-based criterion is utilized to determine the neighborhood size, and three kinds of local features are calculated: 2D/3D eigenvalue-, height distribution-, and normal vector-based features. Using these local features as inputs, we then develop a deep learning-based place description network to generate a discriminative and generalizable global descriptor to uniquely encode the large-scale point cloud. More specifically, the graph neural network is introduced to reveal the intrinsic relations among different compositions in the point cloud and aggregate the spatial distribution characteristics of similar local structures; this contributes to aggregating the sparse static structured information in the large-scale environment and improve robustness to dynamic objects. More details of the proposed place description network can be found in [10].
- *Place descriptor mapping and environment analysis*: The generated descriptors with their associated positions compose a place descriptor map. Then we cluster the descriptors in the feature space (based on the L_2 distance) to find out the places corresponding to cluster centers, and we define these places as *typical places*.
- *Loop closure*: We introduce a coarse-to-fine strategy to detect loop-closure candidates. More specifically, after generating a new place descriptor, we first compare it with all the typical places to roughly estimate its location and then conduct a series of local place matchings around the typical place to find out the accurate location. A sequence-matching strategy is introduced to exclude interference brought by extremely similar places.

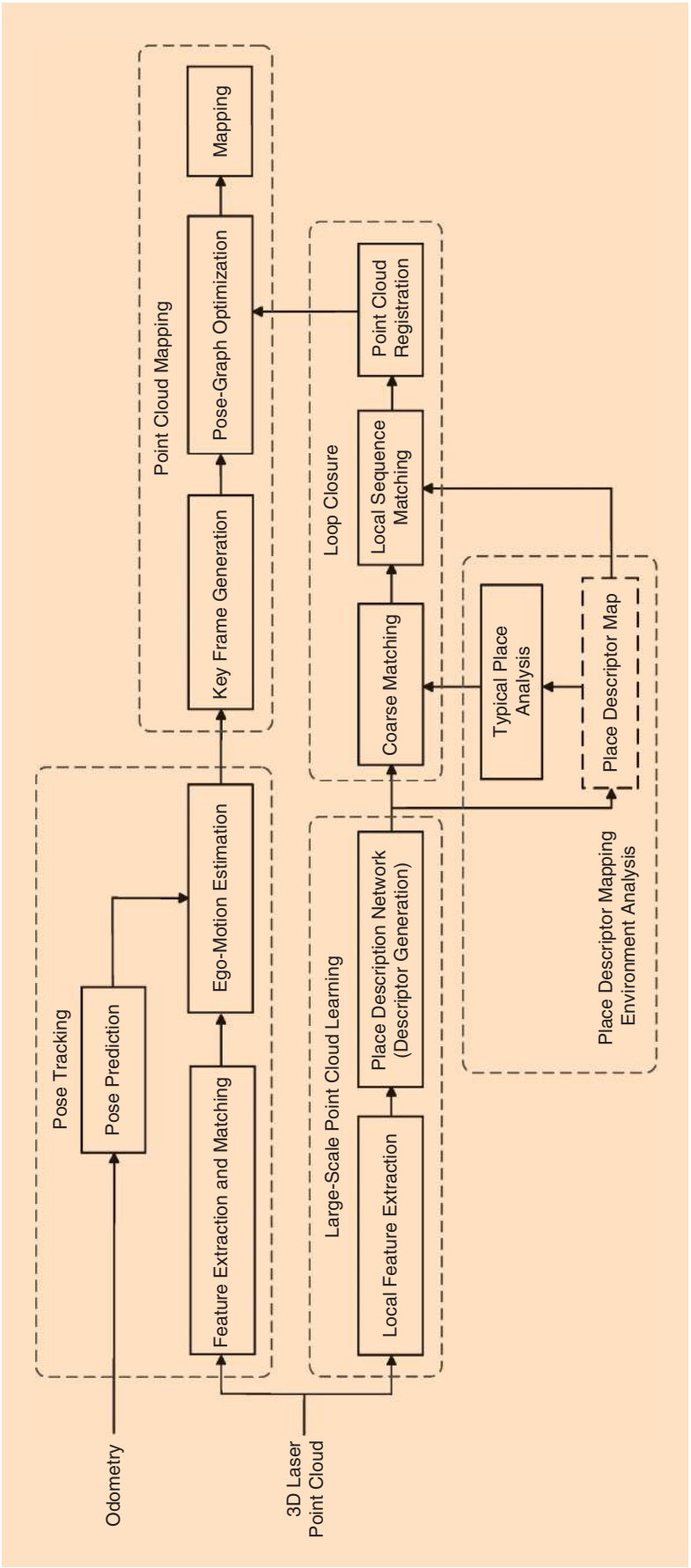


Figure 4. The system structure of the proposed deep learning-based, large-scale localization and mapping approach.

More details of the loop-closure detection approach can be found in [8]. Having a pair of loop-closure candidates (two point clouds correspond to the same place), the next step is to calculate their relative pose, i.e., point cloud registration. FlowNet3D [11], as the state of the art for point cloud-based scene flow estimation, is utilized in the loop-closure module to calculate the rigid motion between point clouds. More specifically, we first shift the target point cloud according to the scene flow predicted by FlowNet3D and perform singular value decomposition to estimate the rigid motion between the target point cloud and the shifted one. Then we conduct ICP to estimate the rigid motion between the shifted target point cloud and source point cloud. By multiplying these two rigid motions, we can calculate the final point cloud registration result, thus obtaining the loop-closure constraint.

For the mapping task in large-scale, dynamic environments, loop-closure detection and point cloud registration are the most challenging elements. As shown in [8], the state-of-the-art approach [7] may also fail in the presence of large driving distances. The sparse static features in the dynamic environment and the cumulative errors in the large-scale mapping process bring great difficulties. Therefore, we introduce deep learning techniques to generate more discriminative and generalizable place descriptors to improve the accuracy and robustness of loop-closure detection and present a point cloud registration approach by combining the deep learning technique and ICP method to achieve loop closure. Unlike existing approaches [2], [7], the proposed approach does not need an accurate initial pose estimation (which requires accurate odometry and robust differential GPS information and is difficult to obtain since the cumulative error could be very large in the long-distance mapping process).

Based on the previous three modules, the traditional front-end pose tracking and back-end pose graph optimization framework can be directly implemented to build a complete SLAM system:

- *Pose tracking*: In the front end, we directly utilize the state-of-the-art LeGO-LOAM [7] for pose tracking. The 3D laser point cloud is preprocessed to extract handcrafted features from the ground, edge, and planar points, and the ego-motion pose is estimated and updated iteratively by combining the odometry-based pose-prediction and feature-matching result.
- *Point cloud mapping*: In the back end, we utilize the factor graph model to estimate the pose graph that is constructed by a series of continuous key frames. The key frames are constructed considering the driving distance, steering angle, and time duration. Then the loop-closure constraints obtained from the loop-closure module are imported to optimize the pose graph and prevent the mapping error from quickly accumulating over long distance pose tracking.

For the real-time localization task, we use the similar pose-tracking and key frame construction methods. Through associating the constructed key frames with those in the prior map, the vehicle pose can be estimated in real time.

Multisensor Fusion Based Perception

The perception system needs to achieve the object detection, tracking, and motion estimation tasks. Currently, widely used approaches include visual- and laser-based ones. Visual sensor data can provide rich information and abundant features but suffers from a limited perception range and is sensitive to illumination changes. The laser-based perception approach can provide accurate spatial information of the detected object and is more robust at night or in bad weather, but it has local sparsity problems; in addition, it is difficult to obtain complete and accurate object segmentations. Considering their strong complementarity, as shown in Figure 5, we propose an integrated perception framework based on multisensor information fusions.

- *Joint calibration*: To carry out the integrated perception framework, joint calibration needs first to be conducted between the laser and camera. The calibration accuracy limits errors occurring in the integrated perception results. Most existing methods [12], [13] utilize the special calibration board and its spatial information to calculate the projection matrix. However, directly implementing these

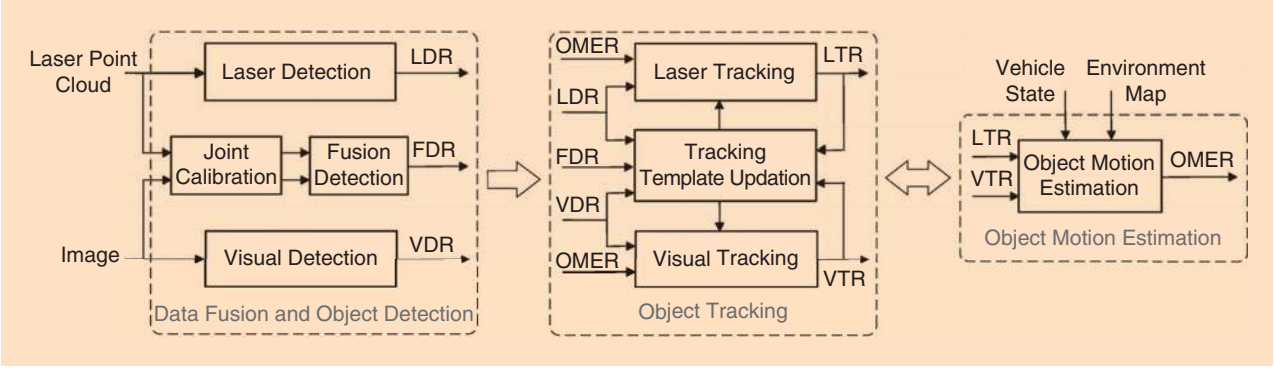


Figure 5. The overall structure of the proposed multisensor information-fusion-based perception system. LDR: laser detection result; FDR: fusion detection result; VDR: visual detection result; LTR: laser-tracking result; VTR: visual-tracking result; OMER: object motion estimation result.

methods in large-scale scenarios will lead to large calibration errors since the practical environment size is much larger than that of the calibration board. To obtain an accurate result, the spatial information in every part of the large-scale environment should be considered. In addition, the data pairs in traditional methods are usually not strict one-to-one correspondences. To overcome these shortages, we present a new method based on manually selected matching 3D/2D data pairs that dispersedly locate in the laser frame/image frame. Instead of using the corner points on the calibration board, the manually selected points can cover the whole calibration environment and thus ensure accuracy in large-scale application scenarios. Finally, we introduce the Efficient Perspective-n-Point (EPnP) method [14] to obtain the accurate 3D–2D projection matrix. More details are shown in Figure 6.

- *Object detection*: The objective of this module is to determine the human workers, vehicles, pallets of goods, and other dynamic entities moving around.
  - For the laser detection module, we first remove the point cloud belonging to the ground surface and then cluster the point cloud into segments, thus obtaining the laser detection results. The ground removal operation could reduce the data count and computation cost and make the object segment prominent without much noise.
  - For the visual detection module, we directly utilize the state-of-the-art YOLO-V3 [15] since it satisfactorily balances real-time performance and accuracy. We train the network on the BDD self-driving data set [16] that contains vehicles, pedestrians, and traffic signs under varied weather and illumination conditions. We also refine the network on our self-collected and labeled industrial data set.
  - With calibration results, the data mapping from 3D point cloud to image can be developed using the frustum projection model [17]; each frustum can be projected to a corresponding rectangle in the image plane. For the objects that can be detected in both the point cloud and the image, we present an integrated approach that benefits from both the object recognition ability of visual detection and the accurate measurement ability of laser detection. We first project the point cloud into the image plane and find out the laser points corresponding to the object area in the image. Then, we cluster those laser points to generate segments, remove foreground occluders and background clutter, and extract the object by the depth information. What's more, as shown in Figure 6, a scoring operation is used to evaluate the saliency of the objects (to define the priority) by considering segment locations and densities. More details of the scoring operation can be found in [18].
- *Object tracking*: The industrial environment is usually highly dynamic and narrow. Autonomous vehicles need to share the operation environment with a large number of moving vehicles and man-made objects. Since missing detection and false detection cannot be avoided in practical

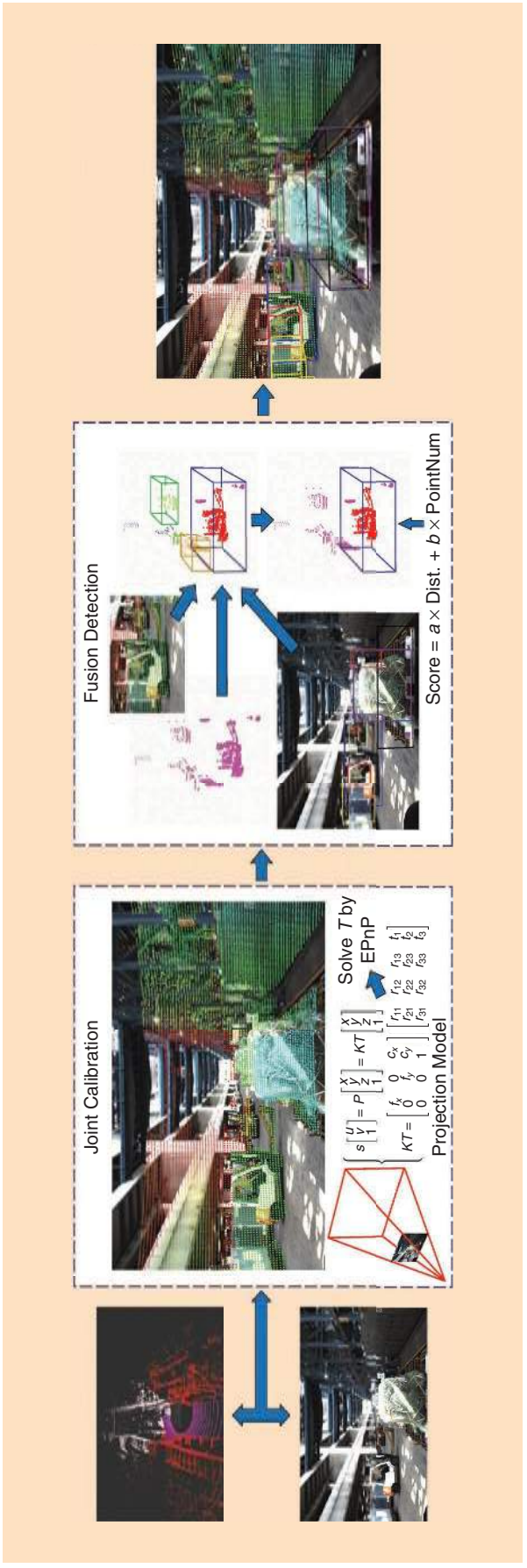


Figure 6. The system structure of the joint calibration and fusion detection modules. Dist.: distance; PointNum: number of points.

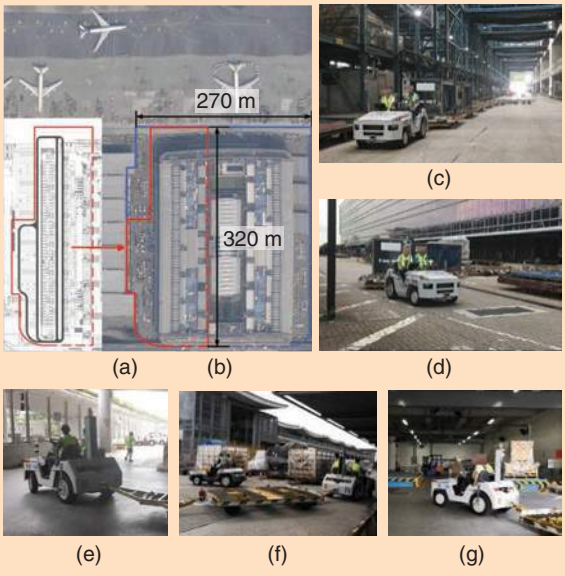


Figure 7. A real-word deployment in the Hong Kong International Airport. (a) and (b) Aerial views of the experiment environment in Google Maps. (c)–(g) Real autonomous transportation scenarios.

object detection results, the object-tracking system is of great importance to ensure safety. Tracking can also help to improve the real-time performance of the detection system and provide additional prediction information about dynamic objects.

- For the laser-tracking module, we match the current point cloud frame with the tracking template to determine corresponding objects. The laser-tracking template is generated considering the previous detection result, tracking result, and estimated object motion information. The laser-tracking results (LTRs) can also provide searching-area candidates for the visual-tracking module.
- For the visual-tracking module, the state-of-the-art approach [19] usually suffers from performance degradation since its tracking template is updated by considering only the previous tracking results. In industrial applications, tracking results may not always be accurate; once a fault occurs, the consequent tracking performance cannot be ensured. In addition, the tracking template should be adjusted dynamically during the whole task period, and a new template needs to be generated in the presence of new objects. To solve this, we modify the approach in [19] by presenting a new template-updating mechanism. More specifically, we update the tracking template with the latest detection result if the following criteria are satisfied simultaneously: 1) the detected and tracked objects are judged the same one by recognizing each object's category and computing the intersection over union, and 2) the confidence of the detection result reaches a sufficiently high level. Compared with [19], leveraging the feature from detection results leads to a high leap on both tracking effectiveness and detection robustness.
- *Object motion estimation*: Based on the previous tracking results, an object motion estimation module is presented using an EKF framework to integrate results from the laser- and visual-tracking modules and then to estimate the object's motion state. The vehicle motion state and environment map are also considered to obtain object motion estimations in the global coordinate.

The proposed detection and tracking modules can ensure acceptable performance in most cases; however, failures still cannot be avoided in extreme cases. To solve this, an ensemble detection module can be developed that acts as a high-level detector for exhaustively utilizing both the laser- and visual-based detection and tracking results. Currently, the consistency index is considered in the proposed perception system to eliminate false detection results, and the inconsistent results are regarded as potential objects that are considered only by the motion control system to ensure safety. In the future, the probabilistic model will be introduced to build the complete ensemble detection module.

Real-Word Deployment and Application

The proposed autonomous vehicle (the Toyota diesel tractor 52-2TD25 with proper self-driving retrofitting) has been successfully deployed in the cargo terminal of the Hong Kong International Airport to achieve automated cargo transportation tasks between the tarmac area and the warehouse. Figure 7(a) and (b) shows the experiment environment in Google Maps; we test the proposed localization, mapping, and perception approaches within the blue area (about 80,000 m^2) and conduct autonomous cargo transportation tasks within the red area. Transportation routes are depicted that include both indoor and outdoor routes. For a single round, the total route length is about 1,250 m. According to the requirement of administration departments, the upper bound of the vehicle velocity in executing automated transportation tasks is restricted to 15 km/h; in addition, two security guards stay ready on the vehicle to take it over in any emergency.

The proposed system has been tested for more than two months (the average operation time is about 3.5 h/day). During the long run, the proposed perception system shows robust performance in different weather and illumination conditions, and no collision has occurred. The results of the proposed perception system are shown in Figures 6 and 8. In Figure 8(a), different colors represent different classes of objects: green-vehicle, red-airplane/person, purple-container, and black-trailer. In Figure 8(b), we show three consecutive frames, where the white blocks are visual detection results and the colored blocks are visual-tracking results. One sees that the blue vehicle cannot be detected in frame 1 and the red vehicle cannot be detected in frames 2 and 3. These missing detection cases show the nonrobustness of the visual detection approach to small-scale objects in the image under

bad illumination conditions (these figures have been partially enlarged for better visualization). Thus, the object-tracking module is of great importance to ensure the performance of the perception system.

The loop-closure and mapping results are shown in Figure 9, which demonstrates the effectiveness of the proposed deep learning-based, large-scale mapping approach. We build a complete 3D point cloud map of the blue area found in Figure 7. Since we do not have a ground truth to evaluate the accuracy of the map, we compare it only with the CAD drawing to qualitatively demonstrate the mapping performance. We also show six typical places selected by clustering the place descriptors

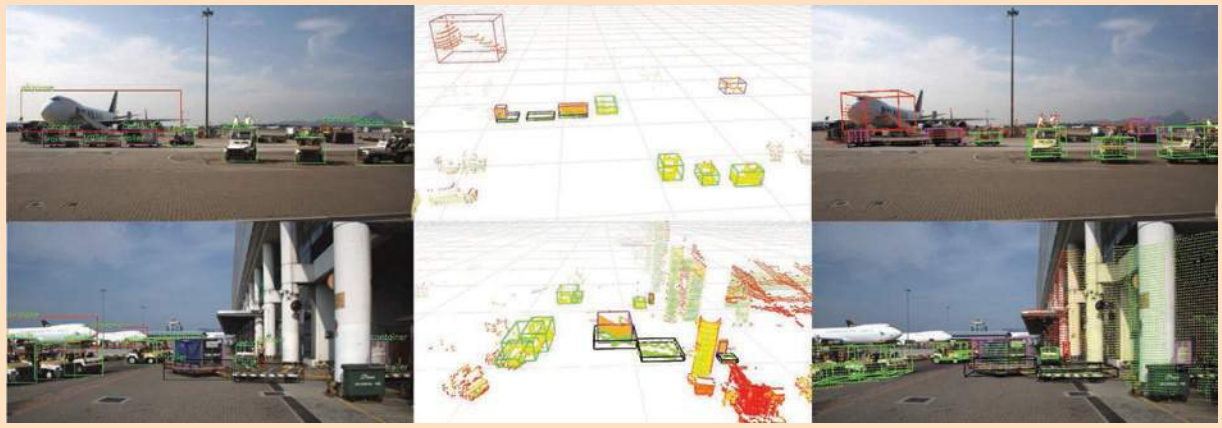


Figure 8. Perception results in the Hong Kong Airport: (a) detection results and (b) tracking results.

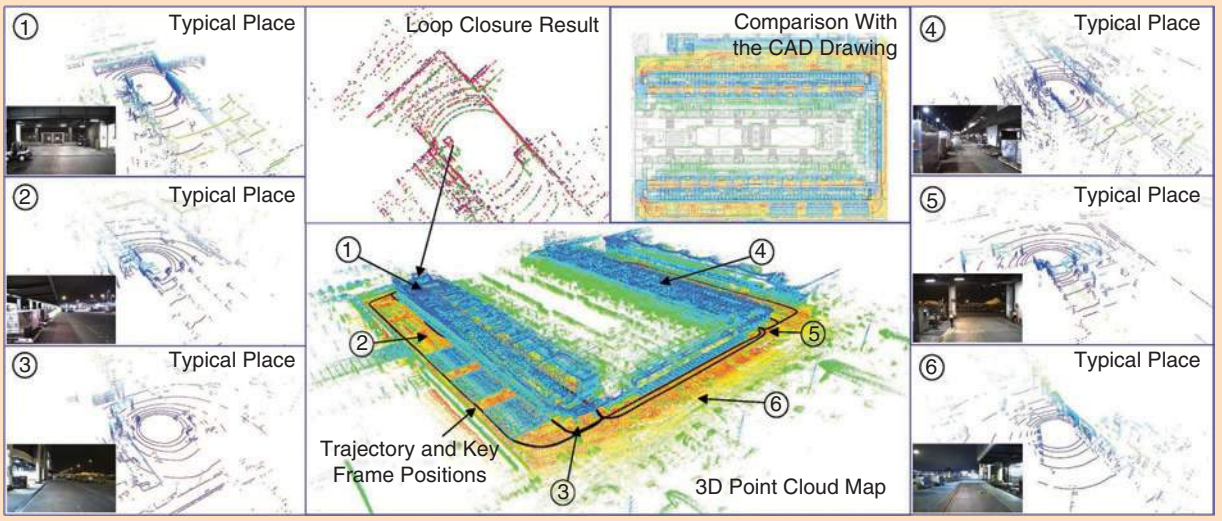


Figure 9. Large-scale loop-closure and mapping results in the Hong Kong Airport.

generated by the proposed place description network. Furthermore, in the loop-closure result, the red and green point clouds are selected as one of the loop-closure candidate pairs since these two point clouds have the smallest L_2 distance between their corresponding place descriptors. Then we calculate their relative pose by the proposed loop-closure method and shift the green point cloud according to the obtained relative pose to generate the blue one, which shows the final point cloud registration result.

The detailed system performance is summarized in Table 1, which validates that the proposed systems are able to provide sufficient performance for automated cargo transportation tasks. Figure 7(c)–(g) shows some real autonomous transportation scenarios to demonstrate the practical applicability of the proposed system. Detailed planning and control approaches can be found in [20].

In Table 2, we compare the proposed loop-closure detection approach presented in the section "Deep Learning-Enhanced Large-Scale Mapping and Localization" with the state-of-the-art laser-based SLAM system LeGO-LOAM [7] in the airport environment shown in Figure 7(a) and (b) (within the red area). To ensure fairness, we do not use local sequence matching in the proposed approach. Table 2 shows that the number of true-positive (TP), false-positive (FP), false-negative (FN), and true-negative (TN) results for each approach along with recall index (TP/(TP + FN)). From Table 2, we find that the accuracy (recall) of the proposed approach is much higher than LeGO-LOAM since the proposed approach can detect more correct loop-closure results (TP) and has a much lower omission rate (FN).

In Figure 10, we compare the mapping performance of the proposed approach presented in the section "Deep Learning-Enhanced Large-Scale Mapping and Localization" with the state-of-the-art laser-based approach LeGO-LOAM [7] and the visual-based approach ORB-SLAM2 [6]. In Figure 10(a), ORB-SLAM2 fails to perform visual odometry tasks due to the absence of enough feature points (caused by motion blur in turnings and large and rapid light condition variations in the process of indoor/outdoor switches). In Figure 10(b), LeGO-LOAM also fails to build a loop-closed map due to the nonrobustness of feature matchings and large accumulating errors over a long distance. However, in Figure 10(c) and (d), the proposed approach successfully builds the loop-closed map of the whole terminal environment, which shows the advanced performance of the proposed approach in large-scale dynamic environments.

In the future, we will implement a group of autonomous vehicles working 24 h/day. The successful application of the proposed automated transportation system will greatly improve the efficiency and quality of cargo transportation at the Hong Kong International Airport, thus leading to revolutionary changes.

Table 1. The performance of the proposed autonomous vehicle system.

Parameter	Value
Maximum velocity	20 km/h
Maximum payload	50 t
Maximum continuous working duration	24 h
Perception range	50 m
Maximum number of simultaneously tracked objects	10
Perception frequency	10 Hz
Loop-closure detection frequency in mapping	2 Hz
Repetitive localization accuracy	5 cm
Localization frequency	10 Hz
Maximum velocity control error	10%
Path-tracking accuracy (straight-line tracking)	8 cm
Path-tracking accuracy (turning)	12 cm
Storage-bin docking accuracy	5 cm

Conclusions

This article proposes a system-level overview of advanced localization and perception approaches for autonomous industrial vehicles. In particular, deep learning techniques are utilized to enhance system performance in a large-scale industrial environment. The proof-of-concept vehicle has been successfully applied in the Hong Kong International Airport to deliver cargo autonomously. This is the first time automated cargo transportation has been achieved at the Hong Kong International Airport, thus reaching a major milestone.

Enhanced by the proposed deep learning-based, large-scale place description network, the proposed localization system is able to achieve place recognition and loop-closure detection tasks in large-scale dynamic environments with sparse static information, Furthermore, it is the first time that we solve the point cloud registration problem by introducing the learning-based scene flow prediction method, thus making it possible to achieve the loop-closure task in the absence of initial pose estimation. In the industry environment,

Table 2. Comparison results of loop-closure detection in airport environments (with different distance bounds in defining correct loop-closure detection results: 5 m/2.5 m/1 m).

	TP	FP	FN	TN	Recall
Our results	255/224/171	0/0/0	10/9/7	754/786/841	0.962/0.961/0.961
LeGO-LOAM	97/97/97	0/0/0	168/136/81	754/786/841	0.366/0.416/0.545

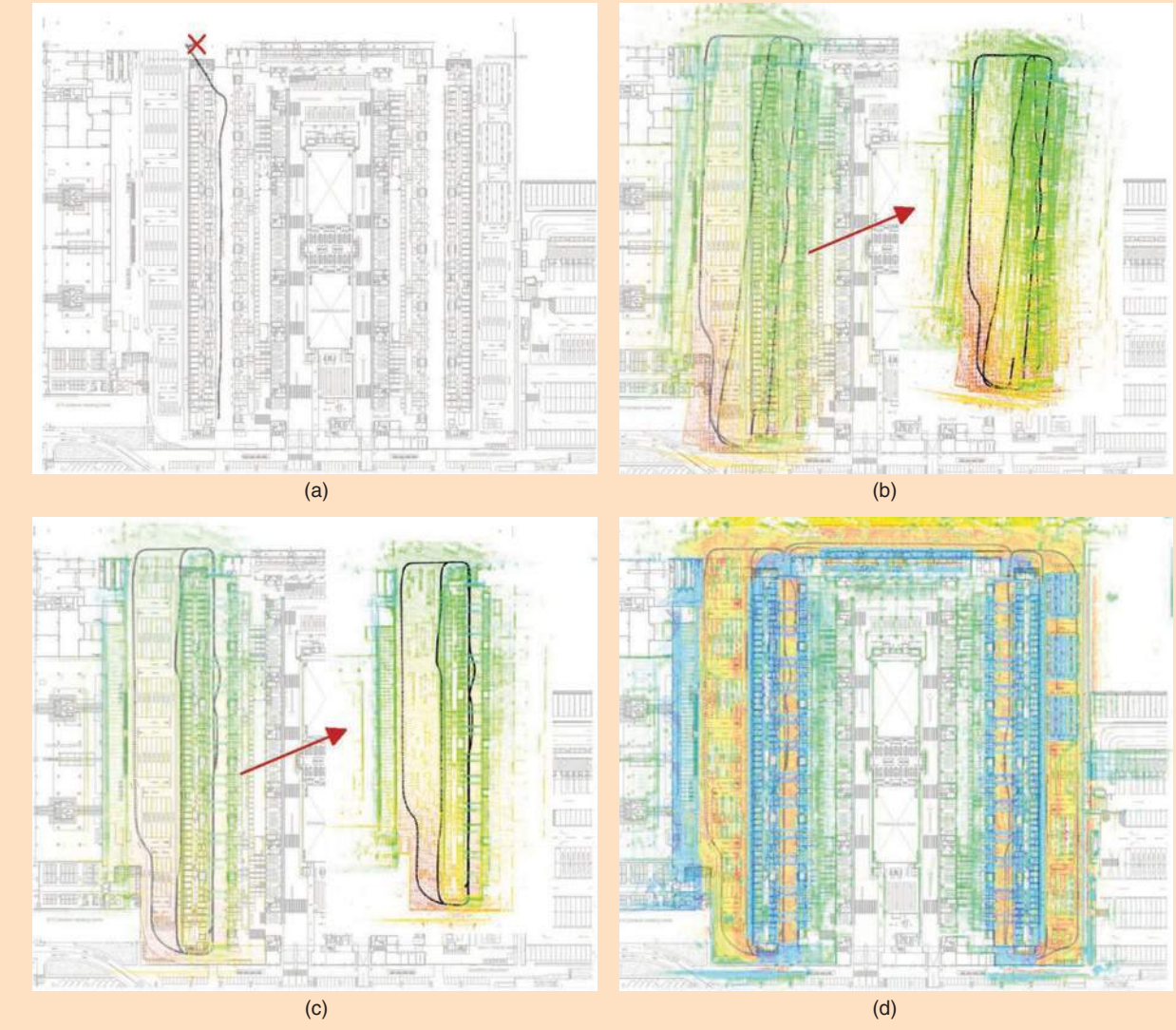


Figure 10. Comparison results of loop-closure and mapping tasks in airport terminal environments: (a) ORB-SLAM2, (b) LeGO-LOAM, (c) our result, and (d) our final mapping result. Since we do not have a ground truth to evaluate the accuracy of the map, we compare it only with the CAD drawing to qualitatively demonstrate the mapping performance.

sensor noises and cumulative errors cannot be avoided, so estimating an accurate initial pose is very hard. The proposed deep learning-enhanced localization system is a big step toward achieving robust localization and mapping performance in practical industrial applications.

In the proposed advanced perception system, a novel joint calibration method was presented by introducing the EPnP method. Compared with traditional methods, we do not need a calibration board and can ensure accuracy in whole large-scale environments. Based on this, a multisensor fusion-based object detection, tracking, and motion estimation system is developed by successfully modifying and integrating state-of-the-art deep learning-based perception networks. Several novel operations are designed for achieving a robust performance in dynamic environments with varied illumination and weather conditions. The proposed advanced perception system presents a systematic and comprehensive solution for practical industrial applications.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China under grant U1613218, Hong Kong ITC under grant ITS/448/16FP, Beijing Advanced Innovation Center for Intelligent Robots and Systems under grant 2019IRS01, and the Shenzhen Science and Technology Innovation Commission via KQTD20140630150243062. The corresponding authors for this article are Hesheng Wang and Yun-Hui Liu.

References

[1] Á. Takács, I. Rudas, D. Bösl, and T. Haidegger, "Highly automated vehicles and self-driving cars," *IEEE Robot. Autom. Mag.*, vol. 25, no. 4, pp. 106–112, 2018. doi: 10.1109/MRA.2018.2874301.

[2] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: 3D segment mapping using data-driven descriptors," in *Proc. Robotics: Science and Systems*, 2018. doi: 10.15607/RSS.2018.XIV.003.
[3] T. Ort, L. Paull, and D. Rus, "Autonomous vehicle navigation in rural environments without detailed prior maps," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2018, pp. 2040–2047. doi: 10.1109/ICRA.2018.8460519.
[4] M. Bergerman et al., "Robot farmers: Autonomous orchard vehicles help tree fruit production," *IEEE Robot. Autom. Mag.*, vol. 22, no. 1, pp. 54–63, 2015. doi: 10.1109/MRA.2014.2369292.
[5] L. Sabattini et al., "The PAN-robots project: Advanced automated guided vehicle systems for industrial logistics," *IEEE Robot. Autom. Mag.*, vol. 25, no. 1, pp. 55–64, 2018. doi: 10.1109/MRA.2017.2700325.
[6] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, 2017. doi: 10.1109/TRO.2017.2705103.
[7] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2018, pp. 4758–4765. doi: 10.1109/IROS.2018.8594299.
[8] Z. Liu et al., "SeqLPD: Sequence matching enhanced loop-closure detection based on large-scale point cloud description for self-driving vehicles," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2019, pp. 1218–1223. doi: 10.1109/IROS40897.2019.8967875.
[9] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *Proc. 26th American Control Conf.*, 2007, pp. 2296–2301. doi: 10.1109/ACC.2007.4282788.
[10] Z. Liu et al., "LPD-Net: 3D point cloud learning for large-scale place recognition and environment analysis," in *Proc. IEEE Int. Conf. Computer Vision*, 2019, pp. 2831–2840. doi: 10.1109/ICCV.2019.00292.
[11] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019, pp. 529–537. doi: 10.1109/CVPR.2019.00062.
[12] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna, LiDAR-camera calibration using 3D-3D point correspondences. 2017. [Online]. Available: arXiv:1705.09785
[13] H. Fan et al., Baidu Apollo EM motion planner. 2018. [Online]. Available: arXiv:1807.08048
[14] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EP*n*P: An accurate O(*n*) solution to the P*n*P problem," *Int. J. Comput. Vis.*, vol. 81, no. 2, p. 155, 2009. doi: 10.1007/s11263-008-0152-6.
[15] 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*, 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.
[16] F. Yu et al., BDD100K: A diverse driving video database with scalable annotation tooling. 2018. [Online]. Available: arXiv: 1805.04687
[17] 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 Conf. Computer Vision and Pattern Recognition*, 2018, pp. 918–927. doi: 10.1109/CVPR.2018.00102.
[18] Y. Liu, C. Suo, Z. Liu, and Y.-H. Liu, "A multi-sensor fusion based 2D-driven 3D object detection approach for large scene applications," in *Proc. IEEE Conf. Robotics and Biomimetics*, 2019, pp. 2180–2187. doi: 10.1109/ROBIO49542.2019.8961637.
[19] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware Siamese networks for visual object tracking," in *Proc. European Conf. Computer Vision*, 2018, pp. 103–119. doi: 10.1007/978-3-030-01240-3_7.
[20] H. Zhao et al., "Modelling and dynamic tracking control of industrial vehicles with tractor-trailer structure," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2019, pp. 2905–2910. doi: 10.1109/IROS40897.2019.8967986.

Zhe Liu, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. Email: zl457@cam.ac.uk.

Chuanzhe Suo, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. Email: suo_ivy@foxmail.com.

Yingtian Liu, School of Mechanical Engineering and Automation, Harbin Institute of Technology, Shenzhen, China. Email: liuyingtian@stu.hit.edu.cn.

Yueling Shen, Department of Automation, Shanghai Jiao Tong University, China. Email: shenyl@sjtu.edu.cn.

Zhijian Qiao, Department of Automation, Shanghai Jiao Tong University, China. Email: qiaozhijian@sjtu.edu.cn.

Huanshu Wei, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. Email: weihuanshu94@hotmail.com.

Shunbo Zhou, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. Email: shunbozhou@link.cuhk.edu.hk.

Haoang Li, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. Email: hali@mae.cuhk.edu.hk.

Xinwu Liang, School of Aeronautics and Astronautics, Shanghai Jiao Tong University, China. Email: xinwu113@163.com.

Hesheng Wang, Department of Automation, Key Laboratory of System Control and Information Processing of the Ministry of Education, Key Laboratory of Marine Intelligent Equipment and System of the Ministry of Education, Shanghai Jiao Tong University, Shanghai, China; Beijing Advanced Innovation Center for Intelligent Robots and Systems, Beijing Institute of Technology, China. Email:wanghesheng@sjtu.edu.cn.

Yun-Hui Liu, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. Email: yhliu@cuhk.edu.hk

GPU-Accelerated Vision for Robots

Improving System Throughput Using OpenCV and CUDA

By Enric Cervera

OpenCV is an open source computer vision and machine learning library for C/C++/Python available for Windows, Linux, macOS, and Android platforms. It contains low-level image processing functions as well as high-level algorithms such as object identification, face recognition, and action classification in videos. OpenCV has become very popular, with more than 47,000 people in its user community and 18 million downloads (see https://opencv.org/about/). Under a Berkeley Software Distribution (BSD) license, it can be used for both academic and commercial applications.

Digital Object Identifier 10.1109/MRA.2020.2977601
Date of current version: 25 March 2020

A significant part of computer vision is image processing, with massive parallel computations. Modern graphics processing units (GPUs) are highly parallel, multicore systems, powerful enough to perform general purpose computations on large blocks of data. So it is challenging, yet potentially very rewarding, to accelerate OpenCV on graphics processors.

Background

CUDA is a parallel computing architecture created by Nvidia that makes it possible to use the many computing cores in a GPU to perform general-purpose mathematical calculations [1]. However, it only works on Nvidia cards.

OpenCV and CUDA have been available for more than 10 years [2], and their use has increased significantly; however,

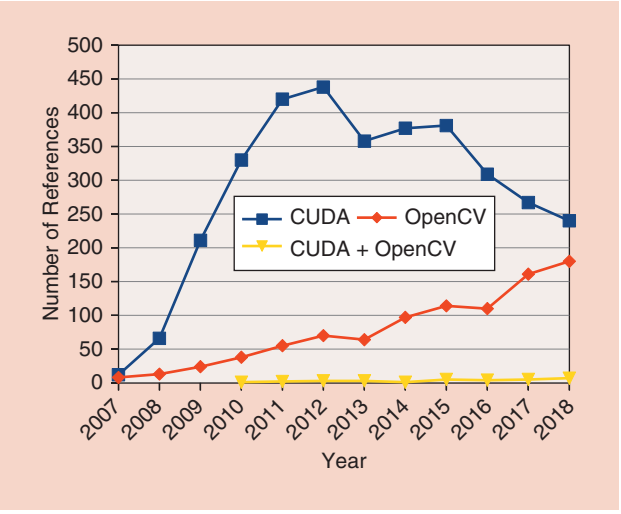


Figure 1. The number of references in IEEE *Xplore* for CUDA and OpenCV.

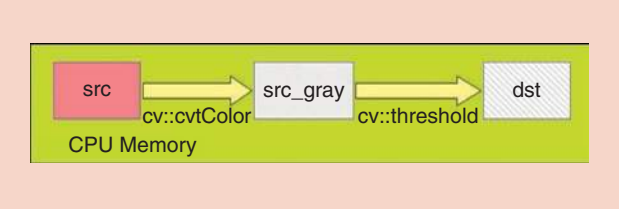


Figure 2. The image processing flow with a CPU only.

their combined application is not so widespread. Considering that a GPU/CUDA module for OpenCV has been available since 2010, the number of works using both libraries published in IEEE *Xplore* is relatively small and has grown very slowly. Figure 1 depicts the number of references in IEEE *Xplore* citing CUDA, OpenCV, or both. In 2018 only seven references for both are found, compared with 240 for CUDA and 180 for OpenCV.

The aim of this article is to describe how the CUDA module for OpenCV works, with some examples of well-known vision problems documented with source code, in order to encourage more robotics researchers to migrate their applications toward GPU computation.

A significant part of computer vision is image processing, with massive parallel computations.

The usefulness of CUDA in robotics and vision has been successfully demonstrated with significant speedups in many applications [3]–[6]. However, it introduces an overhead due to the need to transfer data between the CPU and GPU spaces because most GPU processors work in a dedicated memory, independent from the system memory of the CPU. Consequently, image data need to be moved back and forth between the different types of memory for processing in the GPU. The processing flow consists of the following steps:

1) upload data from main memory to GPU memory
2) initiate the GPU computing kernel
3) perform parallel computation in the GPU's cores
4) download the resulting data from GPU memory to main memory.

The OpenCV CUDA Module

In the following example, we assume that the reader is familiar with OpenCV and C++ programming (for novices, an introduction is provided in [7]). Unless otherwise stated, the code snippets are based on OpenCV 3.4.0, but they can be easily adapted to earlier (2.4) or later (4.x) versions.

In the OpenCV library, all the classes and functions are defined in the name space `cv`. The main object is the class `cv::Mat`, which is essentially a matrix holding pixel values of an image. The GPU modules in OpenCV define a class `cv::cuda::GpuMat`, which is a container for image data kept in GPU memory, with a very similar interface to its CPU counterpart.

Let's consider a quick example with a color image that is converted into gray and binarized with a fixed threshold. In the CPU version, the source image `src` is first converted to an intermediate gray image `src_gray`, which is then thresholded into the resulting image `dst`. We need to define the variables (line 1) and call the OpenCV functions `cv::cvtColor` and `cv::threshold` (lines 2–3) for executing the task:

```
1 cv::Mat src, src_gray, dst;
2 cv::cvtColor( src, src_gray, cv::COLOR_BGR2GRAY );
3 cv::threshold( src_gray, dst, 128, 255, cv::THRESH_BINARY );
```

This processing flow is depicted in Figure 2, where all the data are stored in the CPU memory, and all the operations are performed by the CPU.

In the GPU version, in addition to the variables for the initial and destination images (line 1), we need some new variables for processing the data in the GPU memory (line 2); the intermediate image `src_gray` is also stored in the GPU memory for minimizing data transfers:

```
1 cv::Mat src, dst;
2 cv::cuda::Mat gpu_src, gpu_dst, src_gray;
3 gpu_src.upload( src )
4 cv::cuda::cvtColor( gpu_src, src_gray, cv::COLOR_BGR2GRAY );
5 cv::cuda::threshold( src_gray, gpu_dst, 128, 255, cv::THRESH_
  BINARY );
6 gpu_dst.download( dst );
```

The processing task is performed by the equivalent functions of the OpenCV CUDA module `cv::cuda::cvtColor` and `cv::cuda::threshold`. First, the image is transferred from CPU to GPU memory (line 3); then, the

processing steps are executed (lines 4–5); and finally, the resulting image is transferred from the GPU back to CPU memory (line 6). The processing flow is depicted in Figure 3, where the CPU and GPU memory spaces and the different processing steps are represented.

There is an inherent overhead in the GPU processing flow due to the transfer of the images between the CPU and GPU memories. Such overhead can be minimized if all the processing operations are performed in the GPU and only the initial and final images are transferred:

$$T_{\text{overhead}} = T_{\text{upload}} + T_{\text{download}}.$$

Let's define T_{CPU} and T_{GPU} as the computation times of the image processing operations (`cvtColor, threshold`) at the CPU and GPU, respectively. A speed gain will be obtained if and only if

$$T_{\text{CPU}} > T_{\text{overhead}} + T_{\text{GPU}}.$$

These computation times depend mainly on two factors:

- *Hardware technology of the respective boards*: OpenCV is highly optimized for CPUs with multiple cores and vector instructions.
- *Degree of parallelization of the processing algorithms*: Some vision operations may benefit more than others from the use of multiple cores in the GPU.

Image Processing Applications

In the following, we elaborate on four examples of image processing applications [edge detection, feature extraction, optical flow, and object detection with deep neural networks (DNNs)] that use OpenCV with a CPU and GPU in different hardware configurations.

The first example is a simple edge detection application with the well-known Canny algorithm [8]. The CPU version of the application is as follows:

```
1 cv::Mat src, dst;
2 const int lowThreshold = 20;
3 const int ratio = 3;
4 const int kernel_size = 3;
5 cv::Mat src_gray, blurred, edges;

6 cv::cvtColor( src, src_gray, cv::COLOR_BGR2GRAY );
7 cv::blur( src_gray, blurred, cv::Size(3,3) );
8 cv::Canny( blurred, edges, lowThreshold,
    lowThreshold*ratio, kernel_size );
9 src.copyTo( dst, edges );
```

Besides the initial and final images defined in line 1, three more variables are created in line 5 for storing the intermediate images. The algorithm parameters are defined in lines 2–4, and the processing steps (converting to gray, blurring, and computing the edges) are executed in lines 6–8. Finally, the edges are used as a pixel mask for copying the original image to the destination image in line 9.

The CUDA version is very similar, yet there are some changes in the application programming interface of the processing functions:

```
1  cv::Mat src, dst;
2  const int lowThreshold = 20;
3  const int ratio = 3;
4  const int kernel_size = 3;
5  cv::cuda::GpuMat gpu_src, gpu_dst;
6  cv::cuda::GpuMat src_gray, blurred, edges;

7  gpu_src.upload( src ) ;
8  cv::Ptr<cv::cuda::Filter> blur =
     cv::cuda::createBoxFilter( CV_8UC1, CV_8UC1,
     cv::Size(3,3) );
9  cv::Ptr<cv::cuda::CannyEdgeDetector> canny =
     cv::cuda::createCannyEdgeDetector( lowThreshold,
     lowThreshold*ratio, kernel_size );

10 cv::cuda::cvtColor( gpu_src, src_gray, cv::COLOR_BGR-
     2GRAY );
11 blur->apply( src_gray, blurred );
12 canny->detect( blurred, edges );
13 gpu_src.copyTo( gpu_dst, edges );
14 gpu_dst.download( dst );
```

Now we need to define the original and destinations images both as Mat and GpuMat variables (lines 1 and 5). The parameters are defined in the same ways as in the previous version (lines 2–4), and the intermediate images are defined as GpuMat (line 6). The original image is uploaded to the GPU memory in line 7. Then, two new objects have to be defined for applying the blur filter and the Canny detector, respectively, in lines 8 and 9. Image processing is executed in lines 10–12, and the original image is masked with the detected edges and copied to the destination (line 13). Finally, in line 14 the result is downloaded to the

Modern GPUs are highly parallel, multicore systems, powerful enough to perform general purpose computations on large blocks of data.

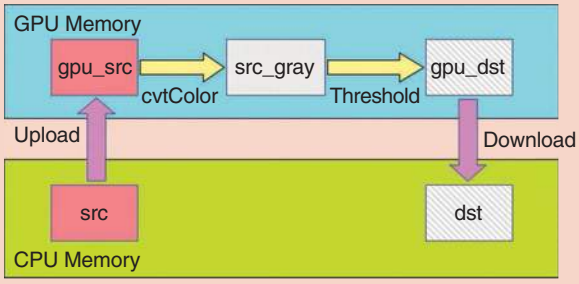


Figure 3. The image processing flow with a CPU and GPU.

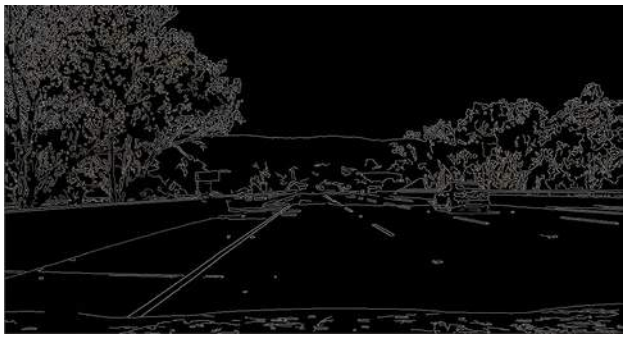


Figure 4. The output image of the edge detection example.

Table 1. The technical specifications of the systems used in the experiments.

GPU Features	Desktop PC	Laptop PC	Embedded PC
CUDA cores	2,560	640	128
Memory	8 GB	4 GB	4 GB
Memory interface	GDDR5	GDDR5	LPDDR4
Memory interface width	256 b	128 b	64 b
Memory bandwidth	320 GB/s	112 GB/s	25.6 GB/s
Power consumption	180 W	40–50 W	10 W

CPU memory. Figure 4 depicts the output for a frame of a video recorded during a car navigation task.

To measure the average computing times of the algorithm, we processed the frames of a benchmarking video on three different hardware configurations of a CPU and GPU:

- *Desktop PC*, with a CPU Intel Core i7-6700 at 3.4 GHz and a GPU GeForce GTX 1080
- *Laptop PC*, with a CPU Intel Core i7-8550U at 3.3 GHz and a GPU GeForce GTX 1050
- *Embedded PC*, an NVIDIA Jetson Nano with an ARM-A57 processor and an integrated GPU.

It is challenging, yet potentially very rewarding, to accelerate OpenCV on graphics processors.

The video consists of 50-s footage from a car in a highway available at Udacity's Advanced Line Finding Project (https://github.com/udacity/CarND-Advanced-Lane-Lines), recorded at 25 Hz with a resolution of 1,280 x 720 24-b red, green, and blue. The main specifications of the GPUs for the three systems are presented in Table 1. The desktop PC features the most powerful CPU both in terms of processing cores and transfer speed, but it also requires more energy compared to the laptop and embedded PCs, which are adequate for mounting on a small robotic platform.

Table 2. The computation times (in milliseconds) for the edge detection algorithm (lower, in bold, is better).

	Desktop PC	Laptop PC	Embedded PC
CPU	**3.514 ± 0.272**	**4.562 ± 0.331**	**12.985 ± 1.197**
GPU	4.422 ± 0.150	7.469 ± 0.106	54.064 ± 1.993

The source code with instructions for compilation and execution is publicly available (https://github.com/RobInLabUJI/opencv-cuda). For the sake of reproducibility, we use docker (https://www.docker.com), a Linux container technology that offers some advantages for an easy replication of code: encapsulation, isolation, portability, and control. In addition, containers have less overhead than virtual machines, and they can access the GPU transparently (usually the impact is on the order of less than 1% and hardly noticeable). As a downside, the GPU-enabled version of docker (Nvidia-docker) does not yet support Windows or macOS.

The code can also be compiled and executed natively in a Linux computer (as long as all the requirements are previously installed—basically, OpenCV and CUDA) with the typical building commands:

```
mkdir -p build
cd build
cmake . .
make
```

The results are shown in Table 2. They measure the mean and standard deviation of the execution time for 1,200 frames in the video (the initial 60 frames are skipped to avoid initialization delays). The execution time is measured starting from the first call to processing functions and continues until the final result is returned; this result is averaged with a moving window of 30 frames. For the GPU cases, the measured time includes the uploading of the initial image to GPU memory and the downloading of the result image back to CPU memory. Visual information [edges, object request broker (ORB) keypoints, and optical flow] is included in the measured code for clarity and debugging purposes, although in a real setup it could be removed to increase the throughput.

It is worth noting that, for this application, the CPUs are faster than the GPUs in all three systems; edge detection is a relatively simple computation, and the execution time is small compared with the overhead of transferring the images into the GPU memory.

An example of the benchmarking code for measuring the execution time is as follows:

```
1 double ticks = (double)cv::getTickCount();
2 if (use_gpu) {
3     gpu_processing(frame, dst);
4 } else {
5     cpu_processing(frame, dst);
6 }
7 ticks = ((double)cv::getTickCount() - ticks)/cv::getTick
  Frequency()*1000;
```

The OpenCV functions `cv::getTickCount()` and `cv::getTickFrequency` are used to obtain the number of ticks before and after the processing work, translated into seconds. A boolean variable indicates whether to use the CPU or GPU; the value of this variable can be toggled through a keyboard click.

In a second example, ORB features are detected and extracted from the image. Such features are very important in robotics applications, such as, for visual SLAM [9]. The source code for the CPU version is as follows:

```
1 cv::Mat src, dst;
2 cv::Mat src_gray, descriptors;
3 std::vector<cv::KeyPoint> keypoints;

4 cv::Ptr<cv::ORB> detector = cv::ORB::create();
5 cv::cvtColor( src, src_gray, cv::COLOR_BGR2GRAY );
6 detector->detect( src_gray, keypoints );
7 detector->compute( src_gray, keypoints, descriptors ) ;
8 cv::drawKeypoints( src, keypoints, dst,
    cv::Scalar::all(-1), cv::DrawMatchesFlags::DEFAULT );
```

First, we define the necessary variables for storing the original and final images, the intermediate gray image, and the structures for storing the keypoints and descriptors of the ORB features (lines 1–4). Second, the feature detector is initialized with default parameters in line 4. Finally, the processing steps are performed in lines 5–7; the original color image is converted into a gray image, the keypoints are detected, and their descriptors are computed. In line 8, the keypoints are drawn into the destination image for visualization.

The CUDA version of this example is straightforward:

```
1 cv::Mat src, dst;
2 cv::cuda::GpuMat gpu_src, gpu_dst;
3 cv::cuda::GpuMat src_gray, descriptors;
4 std::vector<cv::KeyPoint> keypoints;

5 gpu_src.upload(src);
6 cv::Ptr<cv::cuda::ORB> detector = cv::cuda::ORB::create();

7 cv::cuda::cvtColor( gpu_src, src_gray, cv::COLOR_BGR-
  2GRAY );
8 detector->detect( src_gray, keypoints ) ;
9 detector->compute( src_gray, keypoints, descriptors );
10 cv::drawKeypoints( src, keypoints, dst,
    cv::Scalar::all(-1), cv::DrawMatchesFlags::DEFAULT );
```

As in the previous example, we need to define two GpuMat variables for the original and destination images (line 2). The intermediate image is also stored in GPU memory, along with the descriptors (line 3), but the keypoints are stored in CPU memory (line 4). After uploading the image in line 5, the feature detector is created, and the processing steps are executed.

One should note that the processing code is basically similar to the previous version; lines 4–7 of the CPU code and lines 6–9 of the GPU code differ only in the use of the namespace `cv::cuda` instead of cv for the class ORB (line 4/6) and the functions `ORB::create` and `cvtColor` (lines 4/6 and 5/7).

The usefulness of CUDA in robotics and vision has been successfully demonstrated with significant speedups in many applications.

Finally, drawing the keypoints is done in exactly the same way (line 10 of the GPU code is the same as line 8 of the CPU code). The output of the ORB detector is shown in Figure 5.

For debugging purposes, the code examples include visualization, and the corresponding function calls have been included in the benchmarking. Since the visualization process uses the same function call in both the CPU and GPU versions, it should not affect the difference between them in terms of performance.

The results are shown in Table 3. In this case, the GPUs are faster than the CPUs due to the increased computational workload demanded by the ORB algorithm. For simplicity, this example has not computed the matching of ORB

Figure 5. The output image of the ORB feature detector.

Table 3. The computation times (in milliseconds) for the ORB feature extraction algorithm (lower, in bold, is better).

	Desktop PC	Laptop PC	Embedded PC
CPU	15.323 ± 0.788	19.281 ± 0.883	101.586 ± 4.112
GPU	**10.777 ± 1.246**	**11.588 ± 0.397**	**66.697 ± 2.936**

features, but it is possible to use either the CPU or the GPU for that purpose with the classes `cv::DescriptorMatcher` and `cv::cuda::DescriptorMatcher`, respectively.

In the third example, we compute the dense optical flow with the Farneback algorithm [10]. The source code for the CPU version is as follows:

```
1 cv::Mat src, dst;
2 cv::Mat prev, cv::Mat next;
3 cv::Mat flow(prev.size(), CV_32FC2);
4 cv::cvtColor(src, next, cv::COLOR_BGR2GRAY);
5 cv::calcOpticalFlowFarneback(prev, next, flow, 0.5, 3, 15, 3,
    5, 1.2, 0);
```

Since optical flow is computed with the difference between the current and previous frames, we need to define additional variables in line 2 to store the frames. We also define a matrix of float numbers `flow` for the result [in line 3, `CV_32FC2` means a 2-channel (complex) floating-point array]. This flow matrix contains the gradient of the movement between two frames; for each pixel location in the original frame, the channels contain dx and `dy`, so that `prev_x + dx = next_x`, and `prev_y + dy = next_y`.

CUDA and other computing frameworks have become programming standards for parallel computing.

The computation steps are quite simple: the color image is converted into a gray image (line 4), and the optical flow algorithm is executed (line 5). For the sake of simplicity, we have omitted additional instructions for displaying the result and storing the frames.

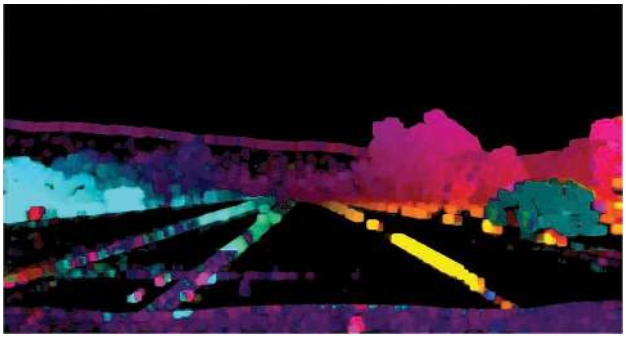


Figure 6. The output image of the dense optical flow algorithm; the hue represents the flow angle, and the intensity is proportional to the flow magnitude.

Table 4. The computation times (in milliseconds) for the dense optical flow algorithm (lower, in bold, is better).

	Desktop PC	Laptop PC	Embedded PC
CPU	196.382 ± 0.889	228.838 ± 6.736	983.943 ± 12.776
GPU	**33.970 ± 0.231**	**78.561 ± 0.498**	**686.960 ± 9.729**

The GPU version is not very different:

```
1 cv::Mat src, dst, flow;
2 cv::cuda::GpuMat gpu_src, gpu_flow;
3 cv::cuda::GpuMat prev, next;
4 gpu_src.upload(src);
5 cv::Ptr<cv::cuda::FarnebackOpticalFlow> fof =
    cv::cuda::FarnebackOpticalFlow::create();
6 cv::cuda::cvtColor( gpu_src, next, cv::COLOR_BGR2GRAY );
7 fof->calc( prev, next, gpu_flow );
8 gpu_flow.download( flow );
```

Besides defining all of the intermediate matrices in GPU memory (lines 2–3), the main difference is in the interface to the optical flow algorithm. In this version, the algorithm object is first defined in line 5, and then applied to the frames in line 7. Finally, the result is downloaded to CPU memory for visualization.

The output of the optical flow algorithm is displayed in Figure 6. The hue of each pixel block represents the orientation of the optical flow vector at that point, and the intensity is proportional to the magnitude of the flow. The results are shown in Table 4. As in the previous example, the execution times for the GPUs are lower than for the CPUs, since computing dense optical flow is a demanding operation.

Finally, we test the DNN module for OpenCV. Since version 3.1, there is a DNN module in the library that implements forward pass (inferencing) with networks pretrained using some popular deep-learning frameworks such as Caffe [11] or TensorFlow [12]. A backend for CUDA was added in OpenCV 4.2.0. In this example, we use the YOLO v3 network [13], a state-of-the-art, real-time object-detection system.

While the details of the OpenCV DNN module are beyond the scope of this article, its design is based on a unique interface that runs on different backends and computation devices (CPU, OpenCL, and CUDA). Consequently, the source code is exactly the same, no matter if the CPU or GPU is used, except for the parameters that select the appropriate backend and computation target. The values for using the CPU are as follows:

```
net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV);
net.setPreferableTarget (cv.dnn.DNN_TARGET_CPU);
```

The GPU can be selected with

```
net.setPreferableBackend(cv.dnn.DNN_BACKEND_CUDA);
net.setPreferableTarget (cv.dnn.DNN_TARGET_CUDA);
```

A typical output image from the DNN module is shown in Figure 7, where several cars are correctly identified in the input image. The frame rate for the CPU and GPU versions

Figure 7. A typical output image of the DNN module with YOLO v3.

Table 5. The frame rates (in hertz) for the YOLO v3 network in the DNN module (higher, in bold, is better).

	Desktop PC	Laptop PC	Embedded PC
CPU	3.51	2.63	0.23
GPU	**46.6**	**17.2**	**2.14**

running on the three types of computers used in the tests is shown in Table 5.

In addition to absolute timings, it is illustrative to calculate the speedup of the GPU with respect to the CPU: in other words, how much faster than the CPU is the GPU for a given application. The speedup results are shown in Figure 8, which displays the values for each application (edge detection, ORB features, optical flow, and DNNs) on each platform (desktop, laptop, and embedded PC).

The overhead penalty can be noticed for the edge detection application on every platform. On the other hand, the speedup is higher in the other applications, skyrocketing in the last example (DNN for object recognition). This result is not surprising, since CUDA is used intensively by the deep-learning community. But the benefits of using the GPU with other OpenCV functions cannot be overlooked, obtaining speedups of 580% and 290% for the computation of the optical flow in the desktop and laptop PCs, respectively.

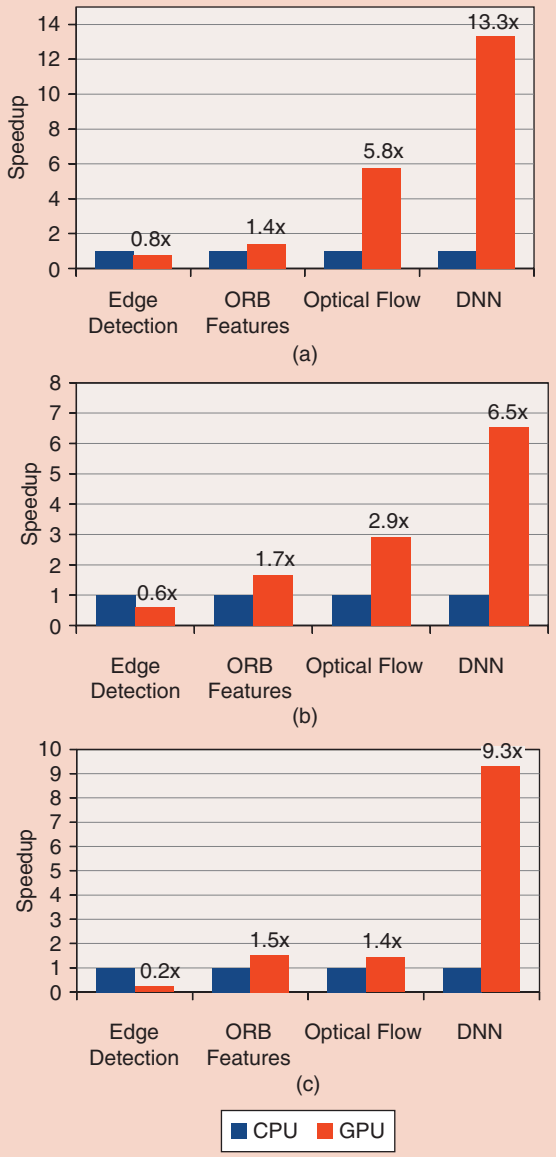


Figure 8. Performance comparisons of CPU versus GPU. (a) Desktop PC. (b) Laptop PC. (c) Embedded PC.

OpenCV in ROS Projects

OpenCV is a widely used library in robotics projects; consequently, there is a precompiled module for the Robot Operating System (ROS) [14] (http://wiki.ros.org/opencv3); this, unfortunately, does not include CUDA support. However, GPU acceleration can still be used by replacing the standard module with a CUDA-enabled version of the OpenCV library. This can be done by installing the library and setting the appropriate path in the file `CMakeLists.txt` of any ROS module using OpenCV:

```
find_package(OpenCV REQUIRED
  PATHS /usr/local
  NO_DEFAULT_PATH
)
```

In addition, all other ROS packages that are dependent upon OpenCV (`cv_bridge`, `image_pipeline`, `image_transport`, etc.) must be rebuilt; that is, their source code must be downloaded into an ROS workspace and compiled with `catkin_make`. A complete example of a simple subscriber is presented in the "ros" branch of the source code repository of this article at https://github.com/RobInLabUJI/opencv-cuda/tree/ros.

Converting an ROS topic image to a CUDA image is straightforward; the topic message is converted to an OpenCV image, and this image is uploaded to the GPU:

```
cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_
encodings::BGR8);
gpuInImage.upload(cv_ptr->image);
```

Once the image is uploaded, the processing can be done as usual, and the result can be downloaded and converted into an ROS message.

Conclusions

CUDA for OpenCV is an easy solution for accelerating vision applications in robotics on systems equipped with a CUDA-enabled GPU. The migration of the code from CPU-based to GPU-based is simple and relatively straightforward, even trivial in some cases. The speedup that can be achieved is system- and problem-dependent. For simple vision algorithms, modern CPUs can be faster; for complex problems involving a sequence of operations on the image, parallelization in the GPU leads to better performance; and for deep-learning applications, the improvement is significant.

An advantage of an open framework such as OpenCL over CUDA is that it is supported by both AMD and Nvidia cards.

We provided some examples with well-known algorithms that are widely used by the robotics community, with the aim of encouraging researchers to improve the throughput of their systems by squeezing all the computing power out of their hardware. CUDA and other computing frameworks (DirectCompute [15] and OpenCL [16]) have become programming standards for parallel computing, and their inclusion in popular libraries like OpenCV is an opportunity for developers to benefit from parallelization without a significant investment in learning specific parallel programming techniques.

An advantage of an open framework such as OpenCL over CUDA is that it is supported by both AMD and Nvidia cards. The interested reader can refer to [17] for details about using OpenCL in OpenCV.

References

[1] D. Luebke, "CUDA: Scalable parallel programming for high-performance scientific computing," in *Proc. 2008 5th IEEE Int. Symp. Biomedical Imaging: From Nano to Macro*, pp. 836–838. doi: 10.1109/ISBI.2008.4541126.

[2] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Real-time computer vision with OpenCV," *Commun. ACM*, vol. 55, no. 6, pp. 61–69, June 2012. doi: 10.1145/2184319.2184337.

[3] P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade, "GPU-accelerated real-time 3D tracking for humanoid locomotion and stair climbing," in *Proc. 2007 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 463–469. doi: 10.1109/IROS.2007.4399104.

[4] T. Xu, T. Pototschnig, K. Kuhnlenz, and M. Buss, "A high-speed multi-GPU implementation of bottom-up attention using CUDA," in *Proc. 2009 IEEE Int. Conf. Robotics and Automation*, pp. 41–47. doi: 10.1109/ROBOT.2009.5152357.

[5] J. Kim, E. Park, X. Cui, H. Kim, and W. A. Gruver, "A fast feature extraction in object recognition using parallel processing on CPU and GPU," in *Proc. 2009 IEEE Int. Conf. Systems, Man and Cybernetics*, pp. 3842–3847. doi: 10.1109/ICSMC.2009.5346612.

[6] N. Dalmedico, M. A. S. Teixeira, H. S. Barbosa, A. S. de Oliveira, L. V. R. de Arruda, and, and F. Neves Jr, "GPU and ROS: The use of general parallel processing architecture for robot perception," in *Robot Operating System (ROS) (Studies in Computational Intelligence)*, A. Koubaa Ed. pp. 407–448. Cham, Switzerland: Springer -Verlag, 2018.

[7] I. Culjak, D. Abram, T. Pribanic, H. Dzapo and M. Cifrek, "A brief introduction to OpenCV," in *Proc. 35th Int. Conv. MIPRO*, 2012, pp. 1725–1730.

[8] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986. doi: 10.1109/TPAMI.1986.4767851.

[9] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," in *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017. doi: 10.1109/TRO.2017.2705103.

[10] G. Farneback, "Very high accuracy velocity estimation using orientation tensors, parametric motion, and simultaneous segmentation of the motion field," in *Proc. 2011 IEEE Int. Conf. Computer Vision*, pp. 171–177. doi: 10.1109/ICCV.2001.937514.

[11] Y. Jia et al., "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, pp. 675–678. 2014, doi: 10.1145/2647868.2654889.

[12] M. Abadi et al., "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Systems Design and Implementation (OSDI)*, pp. 265–283. 2016.

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

[14] M. Quigley et al., "ROS: An open-source robot operating system," in *Proc. ICRA Workshop on Open Source Software*, 2009, vol. 3, no. 3.2, p. 5.

[15] T. Ni, "Direct Compute: Bring GPU computing to the mainstream," in *Proc. GPU Technology Conf.*, 2009, p. 23.

[16] P. Du, R. Weber, P. Luszczek, S. Tomov, G. Peterson, and J. Dongarra, "From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming," *Parallel Comput.*, vol. 38, no. 8, pp. 391–407, 2012. doi: 10.1016/j.parco.2011.10.002.

[17] H. Gasparakis, "Heterogeneous compute in computer vision: OpenCL in OpenCV," in *Proc. SPIE 9029, Visual Information Processing and Communication V*. 2014. vol. 9029, pp. 104–111. doi: 10.1117/12.2054961.

Enric Cervera, Department of Computer Science and Engineering, Jaume-I University, Castelló de la Plana, Castellón, Spain. Email: ecervera@uji.es.

©ISTOCKPHOTO.COM/NEMCHINOWA, SWARM OF DRONES—IMAGE LICENSED BY INGRAM PUBLISHING

Since the beginning of space exploration, Mars and the moon have been examined via orbiters, landers, and rovers. More than 40 missions have targeted Mars, and over 100 have been sent to the moon. Space agencies continue to focus on developing novel strategies and technologies for probing celestial bodies. Multirobot systems are particularly promising for planetary exploration, as they are more robust to individual failure and have the potential to examine larger areas; however, there are limits to how many robots an operator can control individually. We recently took part in the European Space Agency's (ESA's) interdisciplinary equipment test campaign (PANGAEA-X) at a lunar/Mars analog site in Lanzarote, Spain. We used a heterogeneous fleet of unmanned aerial vehicles (UAVs)—a swarm—to study the interplay of systems operations and human factors. Human operators directed the swarm via ad hoc networks and data-sharing protocols to explore unknown areas under two control modes: in one, the operator instructed

By David St-Onge, Marcel Kaufmann, Jacopo Panerati, Benjamin Ramtoula, Yanjun Cao, Emily B.J. Coffey, and Giovanni Beltrame

Planetary Exploration With Robot Teams

Implementing Higher Autonomy With Swarm Intelligence

Digital Object Identifier 10.1109/MRA.2019.2940413
Date of current version: 12 November 2019

each robot separately; in the other, the operator provided general guidance to the swarm, which self-organized via a combination of distributed decision making and consensus building. We assessed cognitive load via pupillometry for each condition and perceived task demand and intuitiveness via self-report. Our results show that implementing higher autonomy with swarm intelligence can reduce workload, freeing the operator for other tasks such as overseeing strategy and communication. Future work will further leverage advances in swarm intelligence for exploration missions.

Multirobot Teams

Extraterrestrial exploration missions are increasingly directed toward challenging landscapes and environments, such as mountains, craters, lava tubes, and oceans [1]–[4]. Rovers have been the most common vehicle choice for planetary exploration; however, they are designed to operate on relatively flat land [5]. As a result, other types of robots, such as UAVs and hydrobots, are being proposed for more challenging environments and geographic features, including the atmospheres and oceans of celestial bodies [6]. UAVs offer advantages over rovers for exploration where atmospheres are present: they provide higher resolution data than orbiters [7], have greater range and mobility [6], and can sample gases at different altitudes, thus also filling a planetary measurement gap [8].

Multirobot teams, perhaps with mixed capacities (that is, activators, sensors, and communication devices) could also be used to explore larger areas more effectively than single robots and could characterize and identify potential landing sites for manned missions as well as reveal hazardous areas. With an appropriate interface, a robotic team could conduct autonomous reconnaissance [9] and increase human situational awareness of mission-critical information. However, there are substantial technical and human challenges to organizing and controlling multirobot systems.

The ESA recently invited our team (Figure 1) to run an experiment on the use of a multirobot aerial system for planetary exploration as part of their PANGAEA-X exercise, a test campaign that brings together astronauts, scientists, engineers, and operations experts for advancing integrated human and robotics missions. Participants, including a European astronaut, controlled a heterogeneous fleet of four to six UAVs.

Our first objective was to demonstrate the physical deployment of the UAV team. Long distances generate communication latencies and impose low bandwidth, so we rely on decentralized control for our robots. All UAVs are replaceable by any others, improving the robustness of the overall system to individual robot failures. Our approach is suitable for gathering aerial images and providing operators with a fleet-wide communication link over kilometers in challenging real-world conditions. These characteristics are made possible by the combination of several of our core contributions to swarm robotics and to a novel approach of human–swarm interaction (HSI): our system sees the operator as just another robot.

The PANGAEA-X context presented a rare occasion to measure human behavior in an operational environment. Task performance and risk-taking behavior [10] differ in the field as compared with simulated robotic tasks, likely because real situations are more engaging and potentially stressful. Our second objective, therefore, was to study the human operator as he or she guides the swarm. We addressed two questions:

Figure 1. The PANGAEA-X field deployment team in Lanzarote, Spain: five engineers, a neuroscientist, five DJI Matrice 100 drones, and five Pleiades Spiri robots. The volcanic landscape of Lanzarote is similar to the surface of the moon. (From left): Marcel Kaufmann, Benjamin Ramtoula, David St-Onge, Giovanni Beltrame, Emily B.J. Coffey, and Yanjun Cao. (Source: ESA; used with permission.)

- How does the operator's perception of usability and workload change over levels of swarm autonomy?
- How do objective measures of user cognitive workload change over levels of autonomy?

To answer these questions, we created two control modes for our swarm that differ by degree of robotic autonomy. We monitored the operator's cognitive load via pupillometry during task performance and subsequently assessed the user's subjective experience. This article presents the main components and contributions of our experimental field setup and discusses the HSI results obtained at PANGAEA-X.

Decentralized Control

In safety and mission-critical multirobot applications, it is unacceptable to end a mission because of a single unit failure. For example, when a group of rovers explores a lava tube on the moon, if the leader robot—the one coordinating the mission and allocating the tasks—gets stuck, the team's exploration potential will be greatly reduced. In an emergency response scenario where a team of aircraft is searching for victims, losing the link to the control station responsible for trajectory planning could result in the loss of life. Decentralized paradigms, which use only local information for control and communication, can mitigate these issues while rendering a team more adaptable to dynamic environments. For instance, failure of a single robot or a broken communication link would not compromise the mission, as the remaining robots can collaboratively reorganize their activities to cover the search region.

For these reasons, we consider decentralized control to be an ideal solution for space applications, as previously shown for the formation control of multiple collaborating spacecraft [11] or to synchronize the actuators of a Martian ground robot [12].

Figure 2 shows the details of our implementation for these experiments. For interrobot communication, we rely on Xbee mesh modules and, for localization, on GPS. Because the robots' coordination relies solely on interrobot distance and bearing, localization is required only for user inputs. GPS can be substituted with other available means (for example, ultra-wide-band, landmarks, or camera-based mapping).

Swarm Behavior Design

Developing sophisticated, fully decentralized behaviors is very challenging, as they can rely only on limited information and local interactions. To simplify our implementation, we use Buzz, a domain-specific programming language for robot swarms [13]. Buzz provides special constructs for robots to share data with the swarm (a technique called *virtual*

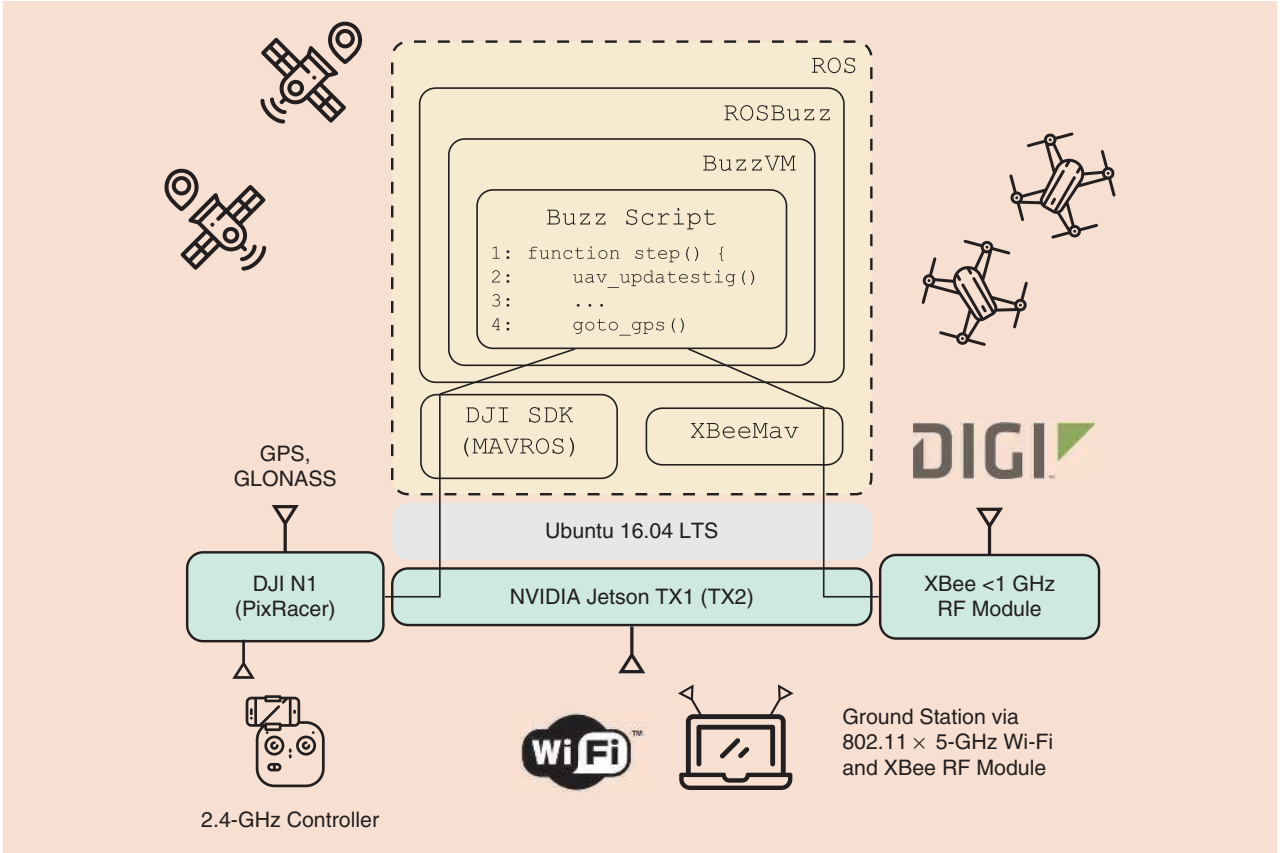


Figure 2. The control architecture (from left to right): the localization system and (backup) remote controllers interface with the UAVs' onboard computers and flight controllers (FCUs) to execute the decentralized behavioral Buzz script. The entire fleet executes the same script, interfacing with the FCU and communication device (Xbee) through the ROS and Mavlink protocol. RF: radio frequency. (Source: Freepik from Flaticon.)

stigmergy) and interact with their neighbors. Buzz is highly portable because it runs within a minimalist virtual machine that works on most computing systems. It merges bottom-up behavior development (i.e., assigning tasks to specific robots) with top-down programs controlling the whole swarm. The developer using Buzz can implement high-level coordination algorithms while still considering the specificity of each of the units in a heterogeneous team, such that the same code can be deployed on any autonomous robot. The algorithms implemented in Buzz for this experiment were tested in a robot operating system (ROS) environment with the Gazebo simulator before field deployment. The Buzz virtual machine [14] and its ROS integration illustrated in Figure 2—including the script files used in this article—are freely available online [15].

Communication With Neighbors

In critical application scenarios, it can be complicated to maintain a reliable connection to all of the robots in a team. The multiple challenges may include 1) large areas to cover, 2) limited one-hop communication ranges, and 3) the (human) command center being potentially located in a remote area for safety or operational reasons. Although a central control architecture requires a link to each robot, decentralized paradigms can support many other sparse network topologies. Decentralized control requires only one of the robots to be within communication range to the ground station to send commands and receive status updates (using Wi-Fi mesh, Xbee, Zigbee, and so on). The information can then be propagated, gossip-like, from robot to robot [as shown in Figure 3(d)].

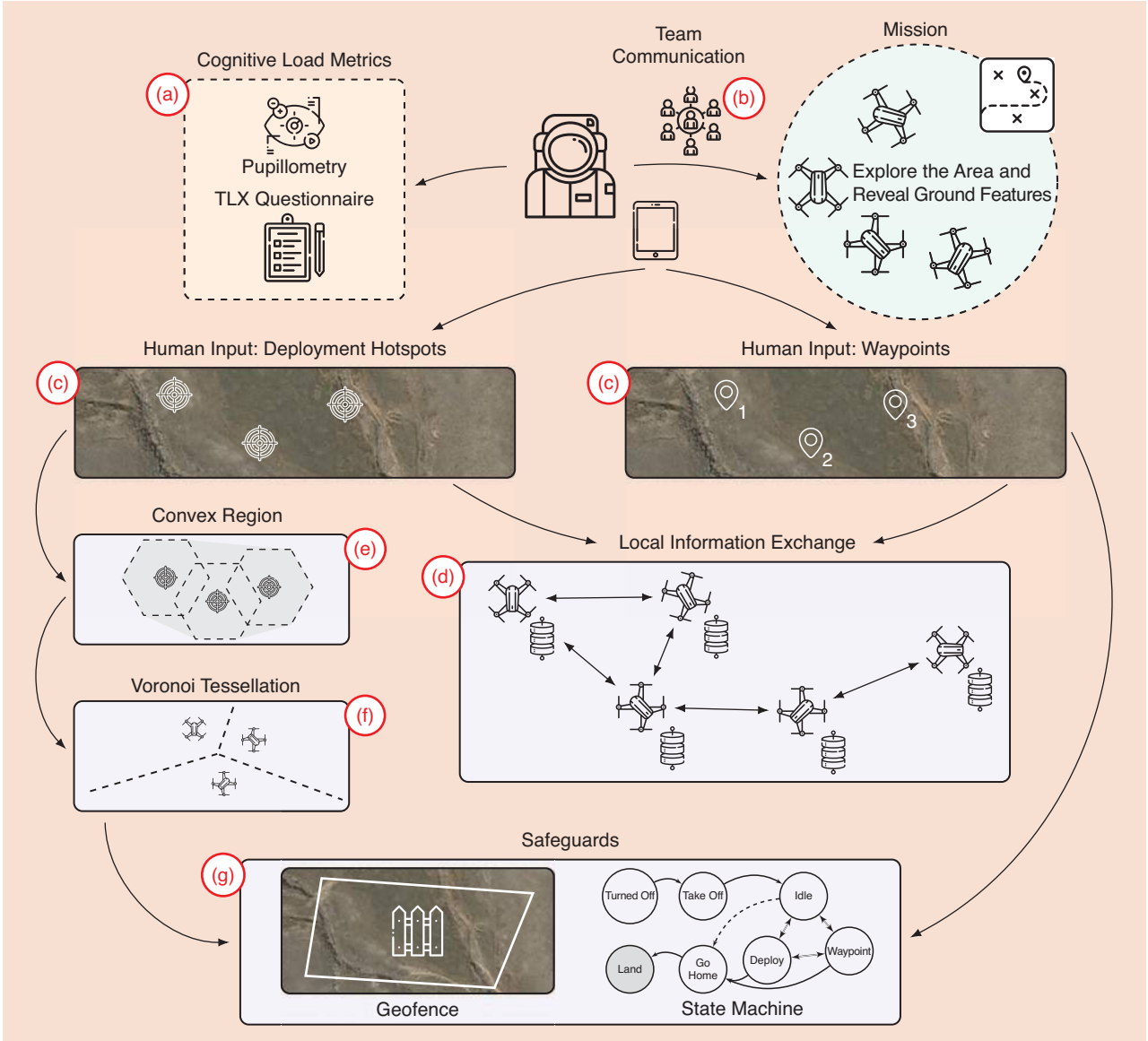


Figure 3. The experiment overview. (a) The objective and self-assessed metrics of the operator cognitive load. (b) The simulated radio chatter used to increase operator task load. (c) The operator inputs following the two modes (hotspots for self-deployment and waypoints for individual control). (d) The decentralized communication infrastructure. (e) The convex region computation on each robot from the user hotspots input. (f) The location goals computed from the tesselation of the region of interest. (g) The safety features implemented (geofence set before the mission and a state machine with consensus mechanism over transitions). (Source: Freepik from Flaticon.)

Each robot's memory contains two tables. The "neighbors" table includes the virtual machine state and sensor output, such as GPS (confidence and coordinates), network-device state (link quality), and battery level. The "mission" table includes mission-relevant information, such as the next goal(s) or the locations of landmarks of interest. Both tables are reliably shared across the robot team through Buzz's virtual stigmergy. In a nutshell, when a piece of data is required by an individual robot, the team agrees on who has its latest version, and the requester receives the updated value through the shortest network path available (that is, the one with fewest hops). This strategy optimizes the use of the bandwidth, allowing the fleet to rely on long-range-transmission devices with low data rates. In previous experiments [16], we stressed our communication architecture and demonstrated that our consensus strategy and information-sharing protocol were resilient with up to 80% packet loss. Field tests showed that we can rely on interrobot, one-hop links of more than 600 m using 800–900-MHz radios (Digimesh Xbee S3B Pro modules). The relations among the communication device, Buzz virtual machine, ROS, and localization system are illustrated in Figure 2.

Safety Features

In a remote-operation scenario for planetary exploration, sudden changes in the environment may arise too quickly for the operator to send each robot a timely command. To cope with bursts of wind, for instance, we use a virtual GPS fence—the geofence—according to limits set before takeoff [Figure 3(g)]. Additional layers of safety are required to cope with communication and human errors. For example, when setting a goal, the destination's distance is verified to be reachable in fewer than 30 s (for example, fewer than 300 m if the UAV maximum velocity is 10 m/s). This verification is done on input, before sending the command, and again by the receiving robot. As the robots generate their own trajectories, they can use the known locations of their neighbors to avoid each other by applying a decentralized collision-avoidance algorithm [17]. To minimize risk of collision, we also flew UAVs at different altitudes.

For each robot, the potential sources of failure increase with its mechanical, communication, and instrumentation complexity. If a robot self-detects an imminent failure, it can upload its current state and mission role to the fleet's shared memory using a mechanism derived from the virtual stigmergy for larger data [18]. Its identity will then be accessible so that another available unit can take over its mission. To achieve system-wide robustness, we wrapped a consensus strategy around the various critical steps of the mission. Before takeoff and before accepting user inputs, all members of the team verify that enough units are available and ready to execute the next task. If a UAV does not get acknowledgment from its expected number of neighbors, it hovers, waiting up to two minutes for the missing teammate(s) to arrive. If the issue is not resolved, the hovering drone will broadcast a message to let the team know that the next step cannot be processed in the current state. Then, the robot will either remain in its current state or proceed to the landing zone, according to the operator's preference. In this regard, the operator is also considered a member of the fleet, and if the link is broken, the same procedure will be triggered. The safety mechanisms discussed in this section are illustrated in Figure 3(g).

Operator Command and Control

A command center for exploration missions must ease the optimization of strategic allocation of field resources, situational awareness, and collaboration among team members. For deployed robotic systems, the command center design is adapted to the robots' level of autonomy. Most space missions fall between teleoperation and fully autonomous missions, having several scripted behaviors as well as high-level commands [19]. Specific to robot teams, novel interaction modalities have been studied for shared autonomy using gesture control [20], voice [21], and even full-body motion [22]; however, map-like interfaces are popular for critical scenarios as they are familiar and leverage operators' existing skills [23]. For UAVs, a map-based interface is commonly referred to as a *mission planner*: an application running on a ground station monitoring the fleet. Currently available commercial mission planners (such as DroneDeploy, DJI Go, and QGroundControl, among others) integrate maps and point-and-click waypoint selection to ease route design for operators.

Figure 4(a) provides a screenshot of the QGroundControl mission planning software. As shown in Figure 4(b), our interface is also based on a map service integrated in a browser control panel. This interface allows the operator to monitor detailed aspects of each robot (battery level, current state, and position) and send general instructions (for example, "Take off" or "Go home") or specific commands related to the fleet control mode (such as waypoint selection or fleet deployment). In all scenarios, the command center acts as a member of the swarm, showing the other members' states and sending updates to the virtual stigmergy data; there is no centralized

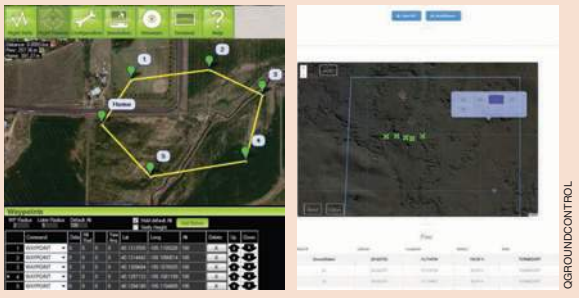


Figure 4. The mission planner screenshot. (a) A popular interface, QGroundControl, with five waypoints defined for a UAV route. (b) Our web-based minimal interface with a waypoint popup menu (individual waypoint control).

control. Path planning and collision avoidance are computed onboard each UAV (Figure 5).

Waypoint Selection Mode

When operating under the waypoint selection mode, our command center displays many of the features that are common to other mission planners, such as the one in Figure 4. In this mode, each robot is controlled individually: the operator clicks on the map and selects the desired robot from a pop-up menu, generating a target waypoint. This is repeated for every robot, resulting in individual goals [Figure 3(c)]. The waypoint command generated is sent to the UAV closest to the ground station and then propagated throughout the fleet. While the mission planning strategy is centralized around the user sending commands, communication is decentralized to ensure tolerance to link and robot failures [24]. With our infrastructure, each waypoint can also be attributed to any other UAV, since the list is shared among the fleet. This is a task allocation optimization problem, and there are several solutions in the literature, for instance [25], which ensures that a goal not reached by its associated robot will be put back in an active list of goals for another robot to pick.

At any time during the mission, the user may clear the waypoint list, triggering the fleet to switch to its hovering state until further instructions arrive. The user may also dynamically change the waypoints, even if the targeted UAV has not yet reached its goal. By doing so, a fast operator can act as if he or she is teleoperating all of the UAVs simultaneously. However, dynamic individual control requires constant monitoring and input. Therefore, cognitive resources are a limitation on swarm size, as each additional robot receives less individual attention, may be neglected, and may become ineffective for part of its exploration time [26]. Furthermore, additional operator tasks, such as communicating with human teammates, increase the operator's cognitive load [27], leading to slower deployment of robots.

Figure 5. German astronaut Matthias Maurer wearing eye-tracking glasses (Pupil Labs) while conducting an exploration mission. He holds a tablet running the mission planner and observes the UAVs' reactions to input. In the background, team members offer suggestions during familiarization training and stand by to assume manual control of the UAVs in case of an emergency. (Source: ESA; used with permission.)

Self-Deployment Mode

In the more autonomous mode, the operator identifies several hotspots that may be worth exploring. This control mode exploits concepts from computational geometry to implement enhanced autonomy within the fleet. The locations received from the operator are shared across the whole fleet, and each robot computes a minimum convex polygonal region of interest [Figure 3(e)]. This region is then split into cells to be distributed among the swarm members, a process known as *surface tessellation*. Some applications, such as search and rescue and sensor-network deployment, already use similar approaches [28]. To provide uniform coverage of the area to be explored, we select the Voronoi tessellation, for which cells contain all points that are geometrically closer to their center. For deploying robotic teams, the initial positions can be taken as seeds to the tessellation problem. A possible implementation consists of creating a frontier (line) halfway between each seed (that is, robot) and merging these lines into polygon edges. Distributed computation of the Voronoi tessellation was extensively studied for multirobot deployment [29]. We use the sweeping line algorithm (Fortune's algorithm), which is one of the most efficient ways to extract cell lines from a set of seeds [30]. We then cut the open cells using the user-defined convex polygonal boundary. From this point on, each robot has knowledge of its Voronoi cell's limits. To reach a uniform distribution of robots in the area, we use a simple gradient descent toward the centroid of each cell [28]. If one of the robots is not yet in the region of interest, it takes a random location goal inside the zone and becomes a seed for the uniform deployment as soon as it enters the region. Each robot recomputes the tessellation following updates on the relative position of its neighbors; an approach that is robust to both packet loss and dynamic inputs. The operator can change the shape and location of the region of interest at any time, and the robots adapt accordingly.

Operator Cognitive Load

Controlling larger numbers of robots is likely to increase the cognitive load of the operator [31]. Cumulative cognitive load might be a function of the attention required to supervise the robots [32], a robot's performance and autonomy when left unattended [33], and the need to manage dependencies among robots when they must coordinate to perform a task [34], in addition to other mission requirements, such as communication with teammates. The field of HSI specifically addresses the tension between a central element of control and a decentralized system [35]: a human operator issues commands to a swarm that may dynamically organize its configuration and the interdependencies between its robots. In the literature, most HSI studies are conducted in simulation. However, human behavior in real-world applications differs from that in simulations [36], suggesting that the operator's cognitive activities might also differ contextually. To avoid the issue, we conducted experiments with physical robots.

Our experiment would not be realistic if we ignored all of the other tasks that human operators would need to perform in parallel for planetary exploration. We created fictional team radio chatter, which is played over earphones [Figure 3(b)]. The audio files contain contextual information, such as "John is going out on EVA; keep an eye out," and occasional mission-specific questions prefaced with the call sign "Operator," such as "How far is robot one from the initial point?" The operator is instructed to quickly acknowledge the communication with a button press on the control screen and respond verbally. In this initial sample, the supplementary task serves to ensure that the speech is attended to, thus increasing workload; in larger sample sizes, it would be possible to use missed versus correct responses as an additional performance measure for statistically comparing conditions.

Cognitive load can be measured through subjective self-assessment metrics, such as questionnaires, and objective metrics, such as body motion, heart-rate variability, and measures of pupil dilation over time derived from pupillometry [37], [38]. Pupillometry has recently gained popularity in applied psychology as a reliable proxy for cognitive load [39]. For example, pupil dilation was used to study the effects of audiovisual interference on workload in piloting tasks [40].

We used both types of measurements: 1) a questionnaire, which included questions inspired by a survey originally designed to evaluate the perceived usability and acceptance of assistive devices [41] augmented with task-load-oriented questions from the NASA Task-Load Index, and 2) pupil dilation and variability [Figure 3(a)]. To provide equivalent psychological distance between the scores [42], all answers were on a seven-point Likert-like scale (range: 0–6).

Results: PANGAEA-X Field Test

Our field experiment was conducted in Lanzarote, Spain. The unique landscape of this volcanic island is one of the closest to a lunar landscape one can find on Earth, and the ESA uses it for geology training for astronauts.

Our sample was small (five participants) due to logistic and weather constraints: setting up the UAV fleet took approximately 30 min per run, each trial lasted about an hour, and UAVs have weather criteria for safe operation (visibility, light, precipitation, and wind speed). We first performed a series of experiments on our own team members (beta testers); then we conducted a core set of experiments on two members of the ESA crew and one journalist.

Informed consent was obtained. The eye tracker was calibrated [43]; then, the operators explored an area for hidden ground features by guiding the UAVs using each of the two control modes, in separate missions, while listening and responding to intermittent radio chatter. After each of the two missions, the participants' feedback on their user experience was collected.

Exploration Task Performance

The aim of each mission was to search for hidden ground features, a task best accomplished by widely covering the search area. A representative example of the UAVs' trajectories for the two control modes is shown in Figure 6. The area covered in the self-deployment mode is larger than in the waypoint mode, whereas, in the waypoint mode, there are duplicate trajectories, indicating lower overall search efficiency. This outcome is expected since the self-deployment algorithm aims at spreading the robots over the area of operator-defined interest. In the waypoint mode, human operators must place the waypoints to sweep the area while mentally keeping track of the area that each robot has already explored.

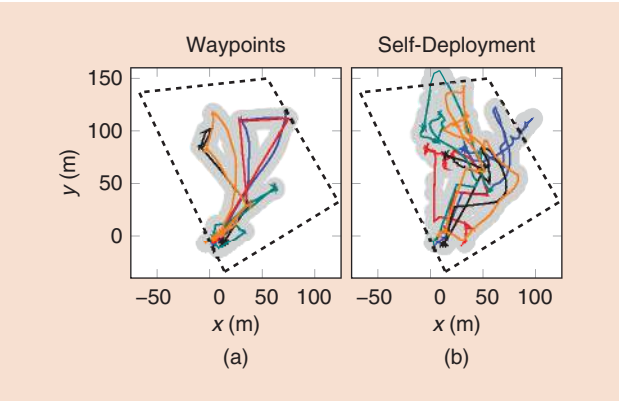


Figure 6. The top views of five UAV trajectories (different colors) comparing the two control modes from a representative participant. (a) The more manual waypoint condition shows duplicated paths for different UAVs. (b) The more autonomous self-deployment condition shows better area coverage. UAVs were twice pushed beyond the geofence by strong winds in the self-deployment condition (b).

On average, for both control modes, the participants discovered most (three of the five total) of the hidden ground features; testing a larger sample may reveal differences. Anecdotally, most participants expressed confusion about which areas had already been explored in the waypoint condition, and they also relied on their memory to assign new goals to the UAVs (instead of clicking on a UAV to get its ID), often sending an unintended UAV toward a new goal. Similar observations apply to the trajectories obtained by the other participants.

Perception of Usability and Workload Over Levels of Swarm Autonomy

Figure 7 presents the average results of the main survey elements for both control modes. For both, the interface was considered similarly easy to learn (>4.6), easy to use (>4.4), intuitive (>4.4), and effective (=4.4) without being cumbersome (<1.5). We attribute these promising initial results to the minimalist and clutter-free design of our mission planner [see Figure 4(b)] and to the good will of the participants. The results also show that neither task was considered highly demanding (<3.0) or hard to complete (<3.0).

Figure 7 shows that individual waypoint control gives the operator more confidence (less insecurity) about the outcome of his or her actions. Results also show that the self-deployment mode was less intuitive, less efficient, and harder to learn. We can also observe that this control mode was slightly more mentally demanding. We believe these perceptions are related to the

impression of not being fully in control; in the self-deployment mode, the relations between the operator's intentions and the robots' behavior is not explicit. Such self-organized and emerging behaviors are more challenging to visualize than deterministic control strategies. These observations suggest that further developments in mission planner data representation will be needed to best communicate from swarm to operator. Furthermore, results suggest that the users' impression of control may influence their perception of cognitive load.

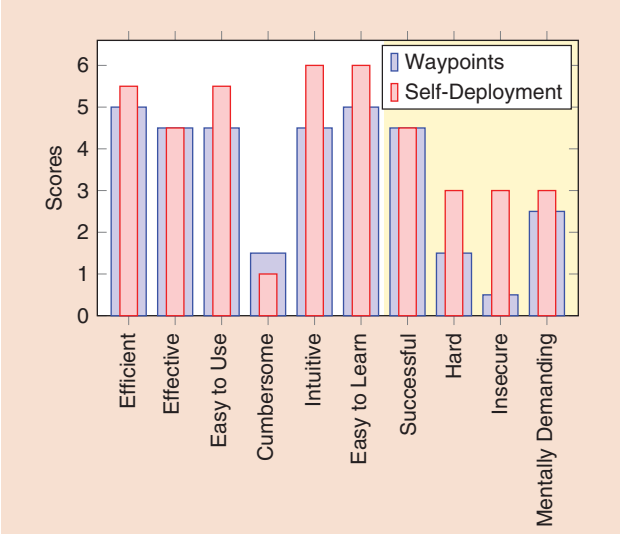


Figure 7. The average results from the questionnaire comparing the perceived usability (white background) and perceived task load (yellow background) of each control mode.

Objective Measures of Cognitive Workload Over Levels of Swarm Autonomy

An external objective measure is required to assess the real difference in cognitive load between the control modes. Pupillometry data were processed to remove outliers ($> \frac{\alpha}{2}$) and subtract the median diameter and then low-pass filtered. The resulting curves of the two missions for the same participant are shown in Figure 8. We overlaid on the image the periods with background radio chatter (orange), direct verbal questions to the operator (green), and operator responses (red).

The pupillometry metrics used here led to different conclusions than the user's self-assessment. The results show that the operator's pupil dilation range was, on average, higher for the waypoint control, suggesting a higher cognitive load for this control mode. The background radio chatter had a similar effect on both missions. Interestingly, both modes led to half of the acknowledgments of direct questions being missed, most likely because the operators were already overwhelmed with the mission.

Conclusions

The ESA PANGAEA-X field campaign provided our team with a unique opportunity to test a swarm fleet in a realistic planetary analog scenario. Leveraging our expertise on decentralized behavior design, simulation-to-field software workflow, and resilient robotic teams, we deployed a heterogeneous fleet of UAVs to demonstrate the robustness of the setup and the study protocol, which will support future development, and we compared the operator's experience of two different levels of embedded autonomy. We showed

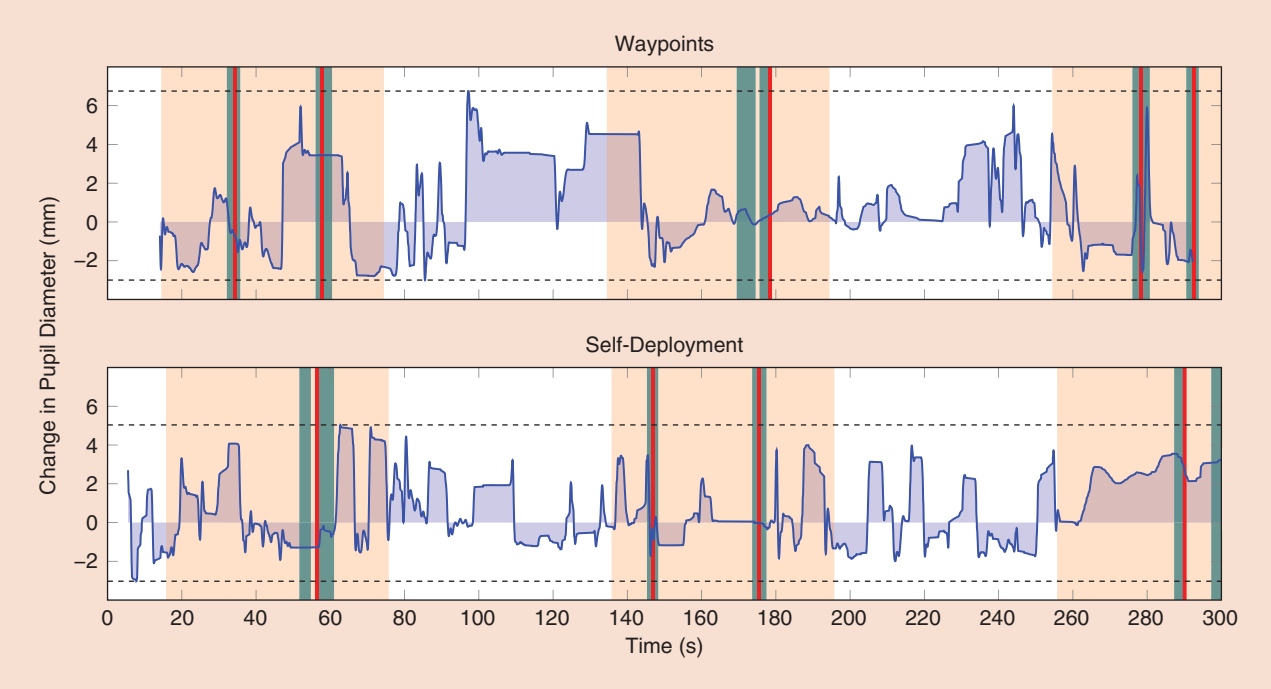


Figure 8. The pupil diameter variations (from the median value) for both control modes for a representative participant. Periods in which the operator was required to attend to background discussion are shown in orange. Green bars indicate direct questions to the operator, and red bars indicate the operator's acknowledgment. Overall, the self-deployment mode shows less pupil diameter variation, suggesting a lighter cognitive load [38].

that the technology of fully decentralized robotic systems is mature enough to provide a robust basis for human factor studies in the field—a first for robotics swarms. The small sample of operators involved does not guarantee the generalization of the human findings, but it does suggest that both objective and subjective measures will help us understand and improve HSIs. From the subjective measures, we learned that managing up to five UAVs simultaneously is still feasible for both the more autonomous and less autonomous control modes; greater benefits of autonomy are likely to be realized as systems are scaled up. We also observed that the uncertainty of the output from the swarm's self-organized behavior led to operator confusion, insecurity, and greater perceived cognitive load. It appears that workload can partly be disentangled from perceived control using objective physiological measures, such as pupillometry. The pupil dilation data seem to indicate that the mode with greater autonomy (self-deployment) is indeed less demanding. Nonetheless, the results point to a need for interfaces that communicate prospective swarm behavior more clearly if the full potential of autonomous swarms is to be realized. Our results motivate work on more intuitive command centers, such as new graphic overlays of prospective swarm behavior on the mission planner, or on a tangible table interface that replaces the mission planner entirely. We believe that swarm intelligence deployment in planetary exploration missions has a promising future and that the technology will also be applicable to many Earth-based mobile autonomous systems.

We are currently conducting a larger experiment to allow for statistical comparisons between conditions. While still using swarms of real robots, a miniature indoor setup will require less preparation time and ensure more stable environmental and luminance conditions, facilitating human measurements. We will also complement pupillometry with measures of skin conductivity and heart-rate variability, which have different physiological bases and will provide a more complete perspective on the operator's state. To firmly anchor our contributions to the development of technologies for space exploration, we are now adapting our tools for ground-to-air robotic teams in lava tubes. These regions are difficult to observe from space but may well be the most suitable environments for safe settlements on the moon and Mars [44].

Acknowledgments

We would like to acknowledge the invitation of the European Space Agency and the support of the PANGAEA-X logistics team. This research was partially funded by Natural Sciences and Engineering Research Council grants, and half of the fleet was designed and adapted to our needs by Pleiades Robotics.

References

[1] J. Bleacher, P. Richardson, W. Garry, J. Zimbelman, D. Williams, and T. Orr, "Identifying lava tubes and their products on olympus mons, mars and implications for planetary exploration," in *Proc. Lunar Planetary Sci. Conf.*, vol. 42, p. 1805, 2011.

[2] M. Cardinale et al., "Present-day aeolian activity in Herschel Crater, Mars," *Icarus*, vol. 265, pp. 139–148, Feb. 2016. doi: 10.1016/j.icarus.2015.10.022.

[3] R. W. Zurek, J. R. Barnes, R. M. Haberle, J. B. Pollack, J. E. Tillman, and C. B. Leovy, "Dynamics of the atmosphere of Mars," *Mars*, pp. 835–933, 1992.

[4] A. Fortes, "Exobiological implications of a possible ammonia–water ocean inside Titan," *Icarus*, vol. 146, no. 2, pp. 444–452, 2000. doi: 10.1006/icar.2000.6400.

[5] T. D. J. M. Sanguino, "50 years of rovers for planetary exploration: A retrospective review for future directions," *Rob. Auton. Syst.*, vol. 94, pp. 172–185, Aug. 2017.

[6] M. Hassanalian, D. Rice, and A. Abdelkefi, "Evolution of space drones for planetary exploration: A review," *Prog. Aerospace Sci.*, vol. 97, pp. 61–105, Feb. 2018. doi: 10.1016/j.paerosci.2018.01.003.

[7] A. Elfes, S. S. Bueno, M. Bergerman, E. C. de Paiva, J. G. Ramos, and J. R. Azinheira, "Robotic airships for exploration of planetary bodies with an atmosphere: Autonomy challenges," *Auton. Rob.*, vol. 14, no. 2/3, pp. 147–164, Mar. 2003. doi: 10.1023/A:1022227602153.

[8] A. A. Gonzales, C. J. Cropus, D. W. Hall, and R. W. Parks, "Development of a useful Mars airplane exploration concept at NASA/AMES Research Center," in *Proc. 6th Mars Society Conf.*, 2003. doi: 10.1.1.660.2009.

[9] NASA, "Mars airplane." Accessed on: March 2019. [Online]. Available: https://www.nasa.gov/centers/ames/research/technology-onepagers/mars-airplane.html

[10] J. W. Rudolph, D. B. Raemer, and R. Simon, "Establishing a safe container for learning in simulation: The role of the presimulation briefing," *Simul. Healthc.*, vol. 9, no. 6, pp. 339–349, 2014. doi: 10.1097/SIH.0000000000000047.

[11] M. Nazari and E. A. Butcher, "Decentralized consensus control of a rigid-body spacecraft formation with communication delay," *J. Guidance, Control, Dynamics*, vol. 39, no. 4, pp. 838–851, 2016.

[12] J. Bruce, "Design, building, testing, and control of SUPERball: A Tensegrity robot to enable new forms of planetary exploration," Ph.D. dissertation, Univ. California, Santa Cruz, 2016.

[13] C. Pinciroli and G. Beltrame, "Swarm-oriented programming of distributed robot networks," *IEEE Computer*, vol. 49, no. 12, pp. 32–41, 2016. doi: 10.1109/MC.2016.376.

[14] MIST Laboratory, "Code repository for the Buzz virtual machine." Accessed on: Jan. 2019. [Online]. Available: https://github.com/MISTLab/Buzz

[15] MIST Laboratory, "Code repository for ROSBuzz." Accessed on: Jan. 2019. [Online]. Available: https://github.com/MISTLab/ROSBuzz

[16] D. St-Onge, V. S. Varadharajan, G. Li, I. Svogor, and G. Beltrame, "ROS and buzz: Consensus-based behaviors for heterogeneous teams." 2017. [Online]. Available: http://arxiv.org/abs/1710.08843

[17] M. Shahriari, I. Svogor, D. St-Onge, and G. Beltrame, "Lightweight collision avoidance for resource-constrained robots," in *Proc. IEEE/RSJ Conf. Intelligent Robots (IROS)*, 2018, pp. 1–9. doi: 10.1109/IROS.2018.8593841.

[18] V. S. Varadharajan, D. St-Onge, B. Adams, and G. Beltrame, "Soul: Data sharing for robot swarms," *Auton. Rob.*, May 2019. doi: 10.1007/s10514-019-09855-2.

[19] V. Dimitrov and T. Padir, "A comparative study of teleoperated and autonomous task completion for sample return rover missions," in *Proc. IEEE Aerospace Conf.*, 2014, pp. 1–6. doi: 10.1109/AERO.2014.6836304.

[20] E. A. Cappo, A. Desai, and N. Michael, "Robust coordinated aerial deployments for theatrical applications given online user interaction via behavior composition," in *Proc. 13th Int. Symp. Distributed Autonomous Robotic Systems*, London, 2016, pp. 665–678.
[21] S. Pourmehr, V. M. Monajjemi, R. Vaughan, and G. Mori, "'You two! Take off!': Creating, modifying and commanding groups of robots using face engagement and indirect speech in voice commands," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, 2013, pp. 137–142.
[22] D. St-Onge, U. Cote-Allard, K. Glette, B. Gosselin, and G. Beltrame, "Engaging with robotic swarms: Commands from expressive motion," *Trans. Human-Rob. Interact.*, vol. 8, no. 2, p. 11, 2019. doi: 10.1145/3323213.
[23] F. Fink-Hooijer, "Civil protection and humanitarian aid in the Ebola response: Lessons for the humanitarian system from the EU experience," *Humanitarian Exchange*, vol. 64, pp. 3–5, June 2015.
[24] V. S. Varadharajan, B. Adams, and G. Beltrame, Failure-tolerant connectivity maintenance for robot swarms. 2019. [Online]. Available: http://arxiv.org/abs/1905.04771
[25] G. Li, D. St-Onge, C. Pinciroli, A. Gasparri, E. Garone, and G. Beltrame, "Decentralized progressive shape formation with robot swarms," *Auton. Rob.*, vol. 43, no. 6, pp. 1505–1521, Aug. 2019. doi: 10.1007/s10514-018-9807-5.
[26] B. Pendleton and M. Goodrich, "Scalable human interaction with robotic swarms," in *Proc. AIAA Infotech@Aerospace (I@A) Conf.*, 2013, pp. 1–13. doi: 10.2514/6.2013-4731.
[27] J.-P. Gagné, J. Besser, and U. Lemke, "Behavioral assessment of listening effort using a dual-task paradigm: A review," *Trends Hearing*, vol. 21, pp. 1–25, Jan. 2017. doi: 10.1177/2331216516687287.
[28] J. Cort et al., "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, 2004. doi: 10.1109/TRA.2004.824698.
[29] V. Alexandrov, K. Kirik, and A. Kobrin, "Multi-robot Voronoi tessellation based area partitioning algorithm study," *J. Behav. Rob.*, vol. 9, no. 1, pp. 214–220, 2018. doi: 10.1515/pjbr-2018-0014.
[30] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, p. 153, Nov. 1987. doi: 10.1007/BF01840357.
[31] G. Podevijn, R. O'Grady, N. Mathews, A. Gilles, C. Fantini-Hauwel, and M. Dorigo, "Investigating the effect of increasing robot group sizes on the human psychophysiological state in the context of human-swarm interaction," *Swarm Intell.*, vol. 10, no. 3, pp. 193–210, 2016. doi: 10.1007/s11721-016-0124-3.
[32] M. Lewis, "Human interaction with multiple remote robots," *Rev. Human Factors Ergonom.*, vol. 9, no. 1, pp. 131–174, 2013. doi: 10.1177/1557234X13506688.
[33] D. R. Olsen, Jr., and S. B. Wood, "Fan-out: Measuring human control of multiple robots," in *Proc. SIGCHI Conf. Human Factors in Computing Systems*, Vienna, Austria, 2004, pp. 231–238.
[34] J. Wang and M. Lewis, "Assessing coordination overhead in control of robot teams," *Syst. Man Cybern.*, pp. 2645–2649, Oct. 2007.
[35] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human interaction with robot swarms: A survey," *IEEE Trans. Human–Mach. Syst.*, vol. 46, no. 1, pp. 9–26, 2016. doi: 10.1109/THMS.2015.2480801.
[36] G. Podevijn, R. O'Grady, C. Fantini-Hauwel, and M. Dorigo, "Investigating the effect of the reality gap on the human psychophysiological state in the context of human-swarm interaction," *PeerJ Comput. Sci.*, vol. 2, Art. no. 82, 2016. doi: 10.7717/peerj-cs.82.
[37] J. Heard, C. E. Harriott, and J. A. Adams, "A survey of workload assessment algorithms," *IEEE Trans. Human–Mach. Syst.*, vol. 48, no. 5, pp. 434–451, 2018. doi: 10.1109/THMS.2017.2782483.
[38] R. Buettner, S. Sauer, C. Maier, and A. Eckhardt, "Real-time prediction of user performance based on pupillary assessment via eye-tracking," *AIS Trans. Human–Comput. Interaction*, vol. 10, no. 1, pp. 26–56, 2018. doi: 10.17705/1thci.00103.
[39] S. Mathot, "Pupillometry: Psychology, physiology, and function," *Journal of Cognition*, vol. 1, no. 1, p. 1–23, 2018.
[40] V. Peysakhovich, F. Dehais, and M. Causse, "Pupil diameter as a measure of cognitive load during auditory-visual interference in a simple piloting task," *Procedia Manuf.*, vol. 3, no. 5, pp. 5199–5205, 2015. doi: 10.1016/j.promfg.2015.07.583.
[41] J. Schmidtler, K. Bengler, F. Dimeas, and A. Campeau-Lecours, "A questionnaire for the evaluation of physical assistive devices (QUEAD)," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, Banff, AB, Canada, 2017, pp. 876–881.
[42] J. Carifio and R. J. Perla, "Ten common misunderstandings, misconceptions, persistent myths and urban legends about Likert scales and Likert response formats and their antidotes," *J. Social Sci.*, vol. 3, no. 3, pp. 106–116, 2007. doi: 10.3844/jssp.2007.106.116.
[43] T. Santini, W. Fuhl, and E. Kasneci, "Calibme: Fast and unsupervised eye tracker calibration for gaze-based pervasive human-computer interaction," in *Proc. 2017 CHI Conf. Human Factors in Computing Systems (CHI '17)*, New York, NY, 2017, pp. 2594–2605.
[44] J. Blamont, "A roadmap to cave dwelling on the moon and Mars," *Adv. Space Res.*, vol. 54, no. 10, pp. 2140–2149, 2014. doi: 10.1016/j.asr.2014.08.019.

David St-Onge, Department of Mechanical Engineering, École de Technologie Supérieure, Montréal, Canada. Email: david.st-onge@etsmtl.ca.

Marcel Kaufmann, Department of Computer Engineering and Software Engineering, Polytechnique Montreal, Canada. Email: marcel.kaufmann@polymtl.ca.

Jacopo Panerati, Department of Computer Engineering and Software Engineering, Polytechnique Montreal, Canada. Email: jacopo.panerati@polymtl.ca.

Benjamin Ramtoula, Department of Computer Engineering and Software Engineering, Polytechnique Montreal, Canada. Email: benjamin.ramtoula@polymtl.ca.

Yanjun Cao, Department of Computer Engineering and Software Engineering, Polytechnique Montreal, Canada. Email: yanjundream@outlook.com.

Emily B.J. Coffey, Department of Psychology, Concordia University, Montreal, Canada. Email: emily.coffey@concordia.ca.

Giovanni Beltrame, Department of Computer Engineering and Software Engineering, Polytechnique Montreal, Canada. Email: giovanni.beltrame@polymtl.ca.

Smart Collaborative Systems for Enabling Flexible and Ergonomic Work Practices

By Arash Ajoudani, Philipp Albrecht, Matteo Bianchi, Andrea Cherubini, Simona Del Ferraro, Philippe Fraisse, Lars Fritzsche, Manolo Garabini, Alberto Ranavolo, Patricia Helen Rosen, Massimo Sartori, Nikos Tsagarakis, Bram Vanderborght, and Sascha Wischniewski

The manufacturing industry is among the top wealth-generating sectors of the global economy and accounted for 15.3% and 10%, respectively, of the total European and American workforce in 2018 [1], [2]. Despite its crucial role, manufacturing is facing a critical challenge based on a reduction of skilled labor availability. This trend is imposing a bottleneck on growth due to the demands of an increasingly competitive market. The aging workforce is not helping this shortfall either, as the available workforce is less able to perform burdensome industrial tasks in an efficient and productive manner.

Efforts are being made to respond to such challenges in manufacturing scenarios and to promote higher quality and more efficient operations. Manufacturing automation by means of robots is widely known as a promising approach to combat such human-centric issues that have been brought about by inherited layouts and processes. Such automation efforts attempt to make complex operations easier for employees to comprehend and support the completion of physically demanding tasks. However, today's solutions require huge initial investments and are often bespoke for particular scenarios; hence, they are not flexible enough to cover all of the requirements of a dynamic, high-mix production environment, typical for small and medium enterprises. These solutions also demand and to a certain extent dictate very specific layouts and work-cell formats that are "robot friendly" [3].

Toward a unified perspective of boosting the competitiveness of the European manufacturing industry, in 2012, the European Commission set the goal of raising manufacturing's share of gross domestic product in Europe from 15% to 20% by 2020 [4]. In this direction, more companies have been encouraged to realize the potential of Industry 4.0 to achieve higher levels of automation, autonomous processes and machinery, and data exchange in manufacturing domains to respond to the customization needs of the future. Customized products are best produced close to the market to quickly respond to customer needs and reduce delivery times. This change implies challenges to the way work and resources are organized within a factory to guarantee cost-effective productivity and quality and reduce waste. In this vision, human–robot collaboration (HRC) frameworks have a high potential because they combine human beings' creativity and craftsmanship with the precision, repetition speed, and consistency of robots to perform complex, skill-demanding tasks while improving work ergonomics. In addition, they can be redeployed to other tasks much more easily. The quality of industrial production is improved, the yield of smaller lot sizes is increased, and the working conditions for humans are improved [3]. In fact, HRC has been a crucial enabler of the current industrial revolution and will be at the core of the upcoming fifth industrial revolution [6].

To create HRC systems that can radically increase the flexibility and productivity of manufacturing while improving the ergonomics of the workplace, four fundamental aspects, i.e., technology, flexibility, interaction quality, and standardization, must be covered comprehensively (see Figure 1). This has been the aim of the first wave of projects with the development of collaborative robots (cobots) (e.g., in Europe, AMARSI, SAPHARI, and PHRIENDS) and, subsequently, with the development of their control and high-level interfaces (e.g., in Europe, the SOPHIA project, whose motto is "human-centered and modular design"). Two distinct objectives of the new wave are to improve competitiveness [3] and reduce work-related musculoskeletal disorders, which represent the single largest category of industrial diseases in first-world counties [7].

HRC Technology

Several technologies must be in place to enable humans and robots to work together to achieve shared goals. In the next sections, we place our focus on human monitoring, sensing and feedback interfaces, and shared intelligence aspects of HRC technology, which have been less discussed in HRC literature in comparison to the traditional safety and control aspects of cobots.

Human Kinodynamic Monitoring

The traditional solutions used for the ergonomic monitoring of industrial

Digital Object Identifier 10.1109/MRA.2020.2985344
Date of current version: 10 June 2020

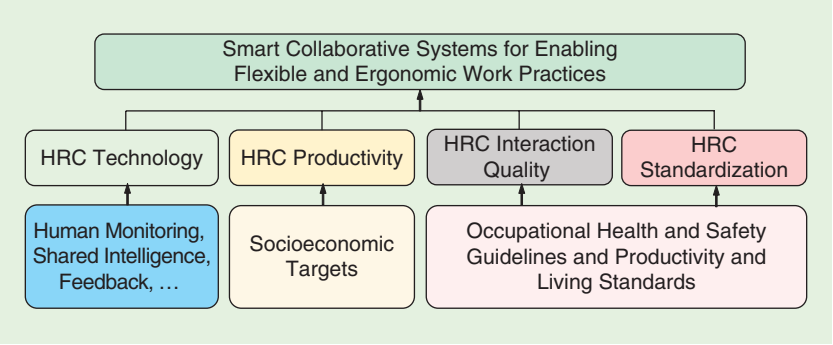


Figure 1. Several fundamental aspects, in addition to technology advancement, must be taken into account to create flexible, productive, and ergonomic HRC systems.

workers are mostly based on heuristic algorithms (e.g., using the ergonomics assessment worksheet) and do not have the required resolution for determining the function of individual muscles. This ultimately limits the design of effective technologies personalized to individual workers. Common monitoring techniques rely on simple measurements (e.g., limb accelerometry or kinematics) where the worker's ergonomics is determined based on whole-body postures. However, even kinematically correct postures may underlie negative muscle compensatory strategies that could increase mechanical tensions and loads on musculoskeletal tissues. Although complex and advanced biomechanical analyses exist, these are bounded to laboratory settings and are not viably transferable to factory settings because they often involve lengthy data acquisition and time-consuming offline analyses with quantitative results being generated only weeks after the initial subject's assessment. In factory settings, the accurate assessment of a worker's musculoskeletal function should be performed via noninvasive sensing technologies that require short preparation time as well as advanced yet rapid musculoskeletal function-probing techniques that can provide instant data on human musculoskeletal mechanics [8]–[10].

Computational musculoskeletal models can provide advanced analysis and an understanding of body function during complex dynamic tasks [11]. Inverse dynamics models have been proposed in which the contribution of individual muscles to joint actuation is resolved according to a priori defined optimization criteria (e.g., minimize the squared muscle-activation sum or metabolic cost of transport) and/or by enforcing preselected muscle-reflexive rules [12]. However, current approaches are dissociated from in vivo body function and therefore limited in describing workers' body function in real-world scenarios; that is, it is not possible to know a priori what the human body really optimizes during a working task, if anything at all. Moreover, even though one model can be tuned to reproduce experimental data [e.g., electromyography (EMG)] in one instance, synergies between muscles, or even between motor units, are highly variable across tasks, training, and fatigue levels [13] and are directly influenced by the environment, e.g., assistive devices or working settings. Therefore, an alternative solution is needed that can capture workers' true muscle-activation patterns and convert them into realistic estimates of musculoskeletal forces.

One way to do this is by developing a new class of data-driven musculoskeletal models. The idea is to combine multimodal body movement sensing with forward dynamics musculoskeletal modeling, i.e., as opposed to state-of-the-art inverse dynamics-based modeling techniques [14]. Data-driven modeling has recently been proposed for fusing 3D body kinematics and muscle EMG recordings and for simulating how muscles activate and how they contract and generate mechanical force with multiple skeletal degrees of freedom simultaneously, in both upper and lower extremities, without making any assumptions about muscle recruitment strategies [15]. More recently, these techniques were employed to connect robotic exoskeletons and bionic arms with the human neuromuscular system, thereby restoring lost motor function in neurologically impaired individuals as well as amputee subjects. The results showed that patients could achieve the voluntary control of wearable robots and that the performance of the established human–machine interface did not deteriorate across large time scales, i.e., days and weeks [16].

Sensing and Feedback Interfaces

To enable an effective interaction between humans and robots, it is fundamental to ensure a correct information exchange between the natural and the artificial side. This requires suitable interfaces that monitor human behavior—to properly plan the execution of the collaborative task—and strategies that increase the mutual awareness of the human–robot dyad. In the literature, different solutions have been proposed to sense human behavior in free motion or during interactions with the environment. Regarding kinematic sensing, both vision-based and wearable devices have been developed with special attention to the hand (e.g., a glove-based system or inertial measurement-unit-based solutions; see [17] and [18] and Figure 2). Recently, commercial and research solutions for whole-body kinematic tracking [see, e.g., Xsense (https://www.xsens.com)] as well as for accurate and wearable EMG acquisition (e.g. Trigno by Delsys, Inc. or FreeEMG by BTS Spa) and ground reaction forces (http://www.moticon.de) were introduced. In parallel, the use of inertial units and low-cost, wearable EMG devices [see, e.g., the Myo Gesture Control Armband (https://www.myo.com/)] has been successfully applied to implement body–machine interfaces for the control of and interaction with robotic and assistive systems (see, e.g., [19]). For warning feedback delivering information to humans on cobot status, wearable devices usually rely on vibrotactile stimuli, which are also applied for

guidance and human–robot-team cooperation [20].

Wearable haptic systems can also be an effective and unobtrusive solution to reproduce a wider range of haptic cues because they can be comfortably worn at different body locations and stimulate the skin locally by conveying to it different types of touch stimulations (see [21]). Regarding vision, in recent years, augmented reality (AR) has gained increased attention, and different commercial systems are now available for AR and virtual reality, e.g., Oculus Rift, Microsoft Hololens, and Google Tango. In AR, components of the digital world may be superimposed upon or composed with the real world and used in teleoperation [22]. The composite scene can be displayed to the user, e.g., through a head-mounted display, to improve situational awareness in HRCs.

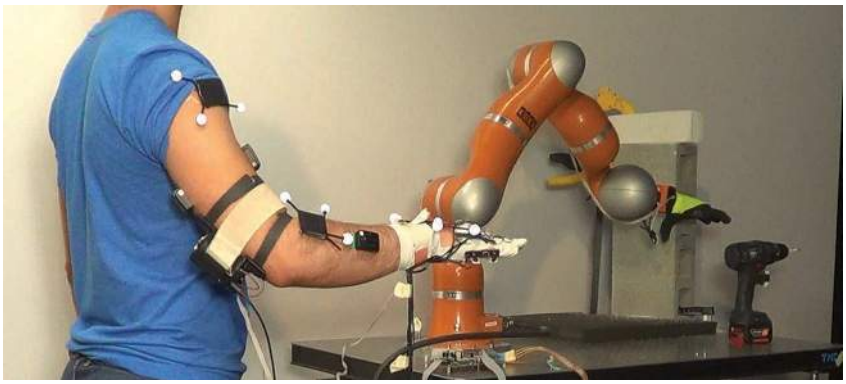


Figure 2. Wearable and lightweight sensing and feedback mechanisms can improve the usability and performance of HRC systems.

Shared Intelligence

Task-allocation schemes have been developed to share the work between human and robot depending on different factors such as capabilities, execution time, performance, and so on. A hierarchical framework for task allocation was developed that assigns a full sequence of atomic tasks based on the capabilities of each agent [23]. Similarly, in [24], complex tasks are split into basic subtasks and attributed depending on their skills. The decision-making algorithms are often based on a multiple-criteria approach using a cost function, such as in [25], and use Markov decisions to determine the best execution plan [26]. Other allocation algorithms have been proposed to extend the task-assignment scheme to the multihuman–multirobot context [27]. The authors in [26] and [27] also took ergonomics into consideration when developing a task-allocation scheme.

On the other hand, the principle of path-planning algorithms is to generate suitable trajectories for the robotic arm using, e.g., cubic interpolation functions, spline functions, or coverage path planning. Those solutions are often only valid for static environments, making them ill-suited for collaborative scenarios. Traditionally, the online method for constrained handling is to use closed-form laws such as potential field methods [30] or antiwindup strategies [31]. The former is able to avoid collisions by generating a field of repulsive forces around the obstacle but is typically unable to deal with actuator saturation. Model predictive control (MPC) is a general purpose control solution able used to handle both state and input constraints in real time. This method is based on the idea of solving at each time instant a constrained optimal control problem over a receding horizon. The main disadvantage of this method is its high computational load; although recent advancements in computational power have made it feasible to implement MPC on robots, MPC schemes are not commonly used in mechatronic applications due to their high computational cost and need for a precise model. Therefore, methods such as the explicit reference governor [32] can enforce both state and input constraints without having to solve an online optimization problem so it can be computed real time.

When applying task-allocation schemes, more criteria than the relative performance can be considered. These criteria may refer to reliability, the number of personnel, workload, or safety [33]. Regardless of which criteria are used to apply a function-allocation scheme, work designers have to be aware of the fact that the automation of functions may introduce new work tasks for the operator that are not directly related to any single function [34].

HRC Flexibility

Cox Jr. [35] defines manufacturing flexibility as "the quickness and ease with which plants can respond to changes in market conditions." Hence, HRC flexibility is needed at two layers: to adapt to the aforementioned manufacturing flexibility typical for Industry 4.0 (due, e.g., to the variety of part shapes and weights, each with a small batch size) and to adapt to worker intentions and commands (which may vary from one person to the other). Several specific manufacturing applications are reported in the research literature, where cobots have addressed the collaborative assembly of a homokinetic mechanical joint [36] and of cellular phones [37] among several others. As of yet, all the aforementioned research works target specific applications, and it is rare to see a cobot capable of addressing multiple and diverse factory tasks. Ideally, such cobots should be mobile, dexterous, bimanual, and easily reprogrammable. A platform developed with such flexibility in mind is the mobile cobot Bimanual

The principle of path-planning algorithms is to generate suitable trajectories for the robotic arm using, e.g., cubic interpolation functions, spline functions, or coverage path planning.

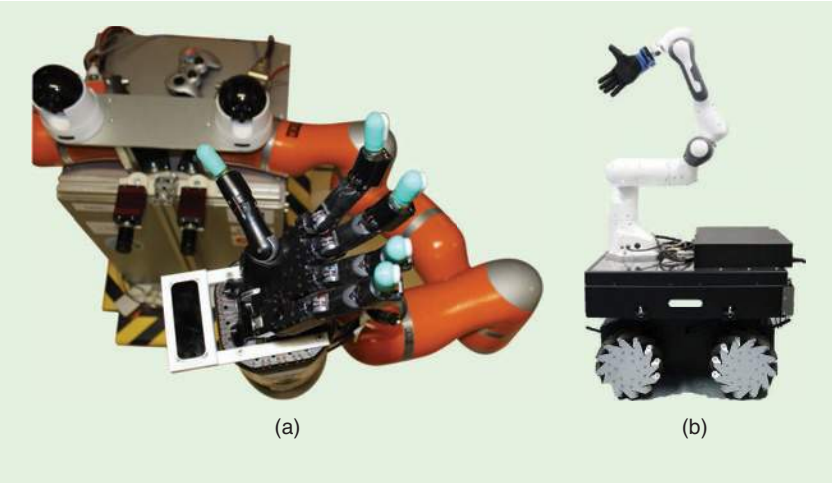


Figure 3. (a) The top view of BAZAR and (b) the side view of MOCA robots. Their locomanipulation capacities can add a certain level of flexibility in manufacturing scenarios.

Agile Zany Anthorpomorphic Robot (BAZAR) [38] [see Figure 3(a)]. Despite its open intuitive programming software OpenPHRI [39], BAZAR is still mainly a research prototype. BAZAR integrates a variety of hardware devices, an integration that today is eased by the diffusion of the Robot Operating System (ROS) (https://www.ros.org). Another example is the mobile collaborative robot assistant (MOCA) platform [40], whose advanced flexibility has been demonstrated in manufacturing [41], teleoperation [40], and so forth [Figure 3(b)].

ROS is a middleware that provides services designed for a heterogeneous computer cluster, similar to those generally present in robot applications.

Indeed, in our view, ROS has been disruptive in robotics, making modular programming available to everyone and facilitating the integration of software and hardware likewise. In this sense, ROS contributes to paving the way toward the sought-after "graal" of HRC flexibility. ROS is a middleware that provides services designed for a heterogeneous computer cluster, similar to those generally present in robot applications. These services include the following:

- hardware device abstraction and control
- the implementation of common robotics algorithms (i.e., for mapping and navigation, perception, localization, and so on) in C++, Python, and Lisp
- a graph representation of the architecture of running processes
- communication between the aforementioned processes (both synchronous and asynchronous)
- package management.

The ROS language-independent tools and most of its C++, Python, and Lisp libraries are open source software, free for both commercial and research use. Because of these open source software dependencies, the main ROS libraries are supported only on Unix-like systems—typically Ubuntu Linux. The fact that ROS was designed with open source in mind has spread it quickly throughout the robotics research community. More recently, ROS-Industrial (https://rosindustrial.org/) has succeeded in attracting the attention of the industrials sector toward the features of ROS. ROS-Industrial, which is also open source, extends the capabilities of ROS to manufacturing automation and robotics by including libraries, tools, and drivers for industrial hardware. Its focus is more on striving toward software robustness and reliability that meet the needs of industrial applications. Although open source robotic hardware seems still utopic, particularly in the industrial context, the breakthrough of ROS and ROS-Industrial as open source platforms may substantially contribute to the flexibility of future cobots worldwide.

HRC Interaction Quality and Acceptance

When designing and introducing human–robot interaction (HRI) to workplaces in a human-centered way, interaction quality plays a key role. Within dyads of humans interacting closely with robots, there are several aspects that contribute to the quality of the specific interaction. One group of characteristics can be summarized under the concept of user acceptance. The concept of technology user acceptance comprises a rich research history. A number of theories and models bring forward different factors influencing the overall user acceptance of a specific technology. These factors, for example, include aspects like the "subjective norms" one feels are associated with the use of a specific technology. However, as shown in the literature, the factors' "perceived usefulness" and "perceived ease of use" have the strongest effect on users' attitudes toward a technology and should therefore be considered carefully [42], [43].

Different robotic design aspects may weaken or enhance overall technology acceptance. There is a limited but growing amount of research focusing on the experiences of factory workers who collaborate with cobots outside the lab (e.g., in [44]). In a field study, a cobot named Walt was integrated on the manufacturing floor of Audi for a glue operation; due to noise in the factory, Walt was equipped with nonverbal communication cues to express its emotions and understand gestures [45]. In terms of social acceptance of the robot, the interviews performed at the end of the study with the operators who used Walt demonstrated that the robot had been accepted as part of the team. Furthermore, they mentioned that working with the latest technology actually gave them a sense of pride.

An additional fundamental aspect contributing to the overall interaction quality refers to the system's usability. ISO

Table 1. ISO Standard 9241-110:2019, *General HRI Principles for System Design*.

Suitability for the task	The meaningful use of the robot, which is adequate for the task.
Self-descriptiveness	The robot's mode of operation and its status are constantly given so that the human worker is aware of the interaction situation at any time.
Controllability	The possibility of the worker to intervene in the process at any time to maintain control over the robot.
Conformity to user expectations	The robot's functionality is always in accordance with the expectations of the worker and with the operational processes.
Error tolerance	False user input that can be corrected easily and the possibility for the worker to execute manual corrections in the task or process.
Suitability for individualization	The possibility of adapting the robot to the workers' needs and abilities.
Learnability	The system includes features that support or simplify learning how to operate the robot.

Standard 9241-110:2019 (see Table 1) provides general design principles for system design that should also be considered in terms of HRI: the principle of *suitability for the task* refers to a meaningful use of the robot that is appropriate to the task. *Self-descriptiveness* specifies the communication of the robot's mode of operation and current status so that the human worker knows at any time what is happening during the interaction. The *controllability principle* describes the possibility of the worker to intervene in the process at any time and thus to maintain control over the robot. *Conformity to user expectations* means that the robot's functionality is always in accordance with the expectations of the worker and the operational processes. The principle of *error tolerance* refers to two aspects: on the one hand, to false user input that can be corrected easily and, on the other hand, to the worker's potential to execute manual corrections during the task or process. *Suitability for individualization* describes the possibility of adapting the robot to the needs and abilities of the worker. Finally, the principle of *learnability* includes features that support or simplify learning how to operate the robot.

User acceptance and fundamental design principles are major aspects that can be used to evaluate and to describe the quality of the HRI. There are additional factors that also contribute to the quality of the individual's HRI experience. In addition to the principle of suitability for the task, special attention must be paid to the tasks remaining with the worker when tasks are divided between humans and robots. Thus, unfavorable tasks should not be delegated to the worker, and a too-tight coupling to the robotic system should be avoided. In addition, the process of introducing a robotic system should be carefully prepared. A detailed explanation of the purpose and benefits as well as the operating characteristics are just as much a part of this as is the worker's opportunity to address possible concerns related to the system. A reserved and skeptical worker's attitude is not unusual and should be addressed in an early stage.

HRC Standardization: Ergonomics

Preventing work-related musculoskeletal disorders is possible by implementing ergonomic interventions that can improve workers' physical conditions in manual-handling activities such as heavy load lifting and handling low loads at high frequency. In the Industry 4.0 era, designing appropriate and effective ergonomic tools means taking into consideration all of the opportunities offered by technological innovation, such as online, instrumental-based approaches used for biomechanical risk assessment and the systems adopted for evaluating the physiological and thermal impacts of the HRC technologies used. Furthermore, one of the main challenges in the next few years will be the revision of existing ergonomic international standards and the development of new ones.

Online, instrumental-based approaches make use of wearable miniaturized sensors for accurate and precise kinematics (joint range of motions), kinetics (forces and torques), and surface EMG measurement (muscle behaviors) [6], [8], [42]. These tools allow 1) direct instrumental evaluations of biomechanical risk when traditional methods are not applicable due to their equations and parameters restrictions and 2) the rating of standard methods when they are applicable. These methods also offer the possibility of classifying the biomechanical risk even in the presence of work tasks in which HRC technologies are used. In fact, the traditional methods listed within existing ergonomic international standards for manual handling activities (ISO 11228-1:2003; ISO 11228-2:2007; and ISO 11228-3:2007) do not cover the consideration of biomechanical risk detection when collaborative technologies (e.g., cobots and exoskeletons), in general, are used. This gap, together with the need to strengthen the scientific basis upon which the standards are based [46], represents the reasons that existing standards should be supplemented or revised or, if necessary, that new standards should be developed. The outcomes of HRC-related R&D projects have significant industrial relevance. Generated knowledge should be transferred into related standardization activities. These standardization activities must have two main objectives: 1) to disseminate (at an early stage) knowledge about

relevant existing standards and standardization activities and 2) to assess project results and analyze them for potentials to be transposed into standards or to be used as input into already-existing standardization activities. These will contribute to filling the gaps among existing standards in the field of HRC.

In addition to repercussions based on ergonomic standards, the use of collaborative technologies must also be evaluated for its impact on physiological and thermal workers' response. In fact, although such technologies are seen as a promising option in biomechanical risk reduction [47] in several occupational sectors, their use requires considerations about their suitability, costs, effectiveness, and impact on the occupational safety and health of workers. One of the most important open issues is, without a doubt, the long-term effects of their use on human physiology. Objective measurements should be supported by subjective measurements aimed at testing user acceptance of collaborative technologies through questionnaires or interviews to investigate physical demand, constraints, perceived usefulness, ease of use, intention to use, performance, and comfort or discomfort.

HRC Productivity

From an economic point of view, HRC systems have a great potential to increase productivity in flexible manufacturing systems. In many modern factories, the limitations of full automation have already been reached. Large industrial robot cells, autonomous conveying technologies, advanced sensor systems for visual control and guidance, and so forth are state of the art. Today, the production of passenger cars, for example, is widely automated (approximately 90–95%) until the car reaches the assembly shop. At this final stage of production, manual human work still dominates with a degree of automation of not more than 10–20%. This is because, compared to all prior parts of the manufacturing process (e.g., press shop, body shop, and paint shop), the variety of tasks and the finesse required are much higher and therefore demand higher skills and more flexibility from workers. It is either very complicated to fully automate such tasks (leading to high risks for downtime), or it is not cost-efficient because the available technological solutions are too expensive to reach a reasonable return on investment. Hence, cobot systems with reduced complexity and lower costs may provide a potential solution for increasing productivity in such work systems by supporting some of the ergonomically heavy work of humans in an efficient way. This may reduce the number of workers needed for completing all of the assembly operations (which equals a direct effect on productivity), but it is also beneficial for the remaining workers' health because it prevents musculoskeletal disorders and increases production quality (which equals an indirect effect on productivity) [48]. In any case, however, as highlighted previously, HRC systems must be designed according to basic ergonomic and safety principles and have to consider the needs of human operators to solve issues instead of creating new ones, increase efficiency, and truly help workers and companies.

Conclusions

Cobots have demonstrated a high potential for addressing the flexibility needs of the increasingly competitive manufacturing industry. They can simultaneously increase productivity and reduce work-related musculoskeletal disorders, which represent the single largest category of work-related disease in industrial countries. This can contribute to economic growth and the creation of better, healthier, and more attractive working environments for the future workforce. Because of this unique potential, the cobot market grows, on average, approximately 50% each year [5]. It is important to note that traditional industrial robots are not in danger of extinction; they will continue to play an important role in manufacturing, mainly as fully automated systems. Their primary purpose is to make high volumes of goods quickly and cheaply, which, however, comes at the price of reduced flexibility and costly redeployment.

Acknowledgments

The research presented in this article was carried out as part of the SOPHIA project, which received funding from the European Union's Horizon 2020 research and innovation program under grant 871237.

References

[1] "Employment by A*10 industry breakdowns," Eurostat, Brussels, Belgium. Accessed on: Feb. 25, 2020. [Online]. Available: ttps://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=nama_10_a10_e&lang=en

[2] D. M. West and C. Lansang, "Global manufacturing scorecard: How the US compares to 18 other nations," Brookings, Washington, D.C., July 10, 2018. [Online]. Available: https://www.brookings.edu/research/global-manufacturing-scorecard-how-the-us-compares-to-18-other-nations/

[3] "Unlocking the potential of industrial human–robot collaboration," European Commission, Brussels, Belgium, Feb. 2020. [Online]. Available: https://ec.europa.eu/info/publications/unlocking-potential-industrial-human-robot-collaboration_en

[4] "'A stronger European industry for growth and economic recovery' industrial policy communication update COM(2012) 582 final," European Economic and Social Committee, Brussels, Belgium. Accessed on: Feb. 25, 2020. [Online]. Available: https://www.eesc.europa.eu/en/our-work/opinions-information-reports/opinions/stronger-european-industry-growth-and-economic-recovery-industrial-policy-communication-update-com2012-582-final

[5] "Collaborative robot (cobot) market," MarketsandMarkets, Northbrook, IL. Accessed on: Feb. 25, 2020. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/collaborative-robot-market-194541294.html

[6] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Auton. Robot.*, vol. 42, pp. 957–975, June 2018. doi: 10.1007/s10514-017-9677-2.

[7] S. Bevan, T. Quadrello, R. McGee, M. Mahdon, A. Vavrovsky, and L. Barham, "Fit for work? Musculoskeletal disorders in the European workforce," The Work Foundation, London, UK, 2009.

[8] R. Alberto, F. Draicchio, T. Varrecchia, A. Silvetti, and S. Iavicoli, "Wearable monitoring devices for biomechanical risk assessment at work: Current status and future challenges—a systematic review," *Int. J. Environ. Res. Public. Health*, vol. 15, no. 9, p. E2001, 2018. doi: 10.3390/ijerph15092001.

[9] W. Kim, J. Lee, L. Peternel, N. Tsagarakis, and A. Ajoudani, "Anticipatory robot assistance for the prevention of human static joint overloading in human–robot collaboration," *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 68–75, 2018. doi: 10.1109/LRA.2017.2729666.

[10] M. Lorenzini, W. Kim, E. De Momi, and A. Ajoudani, "A synergistic approach to the real-time estimation of the feet ground reaction forces and centres of pressure in humans with application to human-robot collaboration," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3654–3661 2018. doi: 10.1109/LRA.2018.2855802.

[11] G. Durandau, D. Farina, and M. Sartori, "Robust real-time musculoskeletal modeling driven by electromyograms," *IEEE Trans. Biomed. Eng.*, vol. 65, no. 3, pp. 556–564, 2018. doi: 10.1109/TBME.2017.2704085.

[12] S. Song and H. Geyer, "A neural circuitry that emphasizes spinal feedback generates diverse behaviours of human locomotion," *J. Physiol.*, vol. 593, no. 16, pp. 3493–3511, 2015. doi: 10.1113/JP270228.

[13] S. J. De Serres and T. E. Milner, "Wrist muscle activation patterns and stiffness associated with stable and unstable mechanical loads," *Exp. Brain Res.*, vol. 86, no. 2, pp. 451–458, Sept. 1991. doi: 10.1007/BF00228972.

[14] M. Sartori, D. G. Llyod, and D. Farina, "Neural data-driven musculoskeletal modeling for personalized neurorehabilitation technologies," *IEEE Trans. Biomed. Eng.*, vol. 63, no. 5, pp. 879–893, 2016. doi: 10.1109/TBME.2016.2538296.

[15] M. Sartori, U. Ş. Yavuz, and D. Farina, "In vivo neuromechanics: Decoding causal motor neuron behavior with resulting musculoskeletal function," *Sci. Rep.*, vol. 7, no. 1, p. 13465, 2017. doi: 10.1038/s41598-017-13766-6.

[16] G. Durandau et al., "Voluntary control of wearable robotic exoskeletons by patients with paresis via neuromechanical modeling," *J. Neuroeng. Rehabil.*, vol. 16, p. 91, 2019. doi: 10.1186/s12984-019-0559-z.

[17] P. J. Kieliba et al., "Comparison of three hand pose reconstruction algorithms using inertial and magnetic measurement units," in *Proc. 2018 IEEE-RAS 18th Int. Conf. Humanoid Robots (Humanoids)*, pp. 1–9. doi: 10.1109/HUMANOIDS.2018.8624929.

[18] S. Ciotti, E. Battaglia, N. Carbonaro, A. Bicchi, A. Tognetti, and M. Bianchi, "A synergy-based optimally designed sensing glove for functional grasp recognition," *Sensors*, vol. 16, no. 6, p. E811, 2016. doi: 10.3390/s16060811.

[19] S. Jain, A. Farshchiansadegh, A. Broad, F. Abdollahi, F. Mussa-Ivaldi, and B. Argall, "Assistive robotic manipulation through shared autonomy and a body-machine interface," in *Proc. 2015 IEEE Int. Conf. Rehabilitation Robotics (ICORR)*, pp. 526–531. doi: 10.1109/ICORR.2015.7281253.

[20] M. Aggravi, G. Salvietti, and D. Prattichizzo, "Haptic wrist guidance using vibrations for Human-Robot teams," in *Proc. 2016 25th IEEE Int. Symp. Robot and Human Interactive Communication (RO-MAN)*, pp. 113–118. doi: 10.1109/ROMAN.2016.7745098.

[21] M. Bianchi, "A fabric-based approach for wearable haptics," *Electronics*, vol. 5, no. 4, p. 44, 2016. doi: 10.3390/electronics5030044.

[22] V. H. Andaluz et al., "Transparency of a bilateral tele-operation scheme of a mobile manipulator robot," in *Proc. Int. Conf. Augmented Reality, Virtual Reality and Computer Graphics*, 2016, pp. 228–245. doi: 10.1007/978-3-319-40621-3_18.

[23] L. Johannsmeier and S. Haddadin, "A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes," *IEEE Robot. Autom. Lett.*, vol. 2, no. 1, pp. 41–48, 2017. doi: 10.1109/LRA.2016.2535907.

[24] F. Chen, K. Sekiyama, F. Cannella, and T. Fukuda, "Optimal subtask allocation for human and robot collaboration within hybrid assembly system," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 4, pp. 1065–1075, 2014. doi: 10.1109/TASE.2013.2274099.

[25] P. Tsarouchi, G. Michalos, S. Makris, T. Athanasatos, K. Dimoulas, and G. Chryssolouris, "On a human–robot workplace design and task allocation system," *Int. J. Comput. Integr. Manuf.*, vol. 30, no. 12, pp. 1272–1279, 2017. doi: 10.1080/0951192X.2017.1307524.

[26] A. Roncone, O. Mangin, and B. Scassellati, "Transparent role assignment and task allocation in human robot collaboration," in *Proc. 2017 IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 1014–1021. doi: 10.1109/ICRA.2017.7989122.

[27] M. S. Malvankar-Mehta and S. S. Mehta, "Optimal task allocation in multi-human multi-robot interaction," *Optim. Lett.*, vol. 9, no. 8, pp. 1787–1803, 2015. doi: 10.1007/s11590-015-0890-7.

[28] I. E. Makrini, K. Merckaert, J. De Winter, D. Lefeber, and B. Vanderborght, "Task allocation for improved ergonomics in human–robot collaborative assembly," *Interact. Stud.*, vol. 20, no. 1, pp. 102–133, 2019. doi: 10.1075/is.18018.mak.

[29] E. Lamon, A. De Franco, L. Peternel, and A. Ajoudani, "A capability-aware role allocation approach to industrial assembly tasks," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3378–3385, 2019. doi: 10.1109/LRA.2019.2926963.

[30] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986. doi: 10.1177/027836498600500106.

[31] L. Zaccarian and A. R. Teel, *Modern Anti-windup Synthesis: Control Augmentation for Actuator Saturation*. Princeton, NJ: Princeton Univ. Press, 2011.

[32] K. Merckaert, B. Vanderborght, M. Nicotra, and E. Garone, "Constrained control of robotic manipulators using the explicit reference governor," in *Proc. 2018 /RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 5155–5162. doi: 10.1109/IROS.2018.8593857.

[33] D. Meister, "*Behavioural Analysis and Measurement Methods*," Chichester, UK: Wiley, 1985.

[34] A. Dearden, M. Harrison, and P. Wright, "Allocation of function: Scenarios, context and the economics of effort," *Int. J. Hum.-Comput. Stud.*, vol. 52, no. 2, pp. 289–318, 2000. doi: 10.1006/ijhc.1999.0290.

[35] T. Cox Jr., "Toward the measurement of manufacturing flexibility," *Prod. Invent. Manag. J.*, vol. 30, no. 1, pp. 68–72, 1989.

[36] A. Cherubini, R. Passama, A. Crosnier, A. Lasnier, and P. Fraisse, "Collaborative manufacturing with physical human–robot interaction," *Robot. Comput.-Integr. Manuf.*, vol. 40, pp. 1–13, Aug. 2016. doi: 10.1016/j.rcim.2015.12.007.

[37] J. T. C. Tan, F. Duan, Y. Zhang, K. Watanabe, R. Kato, and T. Arai, "Human-robot collaboration in cellular manufacturing: Design and development," in *Proc. 2009 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 29–34. doi: 10.1109/IROS.2009.5354155.

[38] A. Cherubini et al., "A collaborative robot for the factory of the future: Bazar," *Int. J. Adv. Manuf. Technol.*, vol. 105, no. 9, pp. 3643–3659, 2019. doi: 10.1007/s00170-019-03806-y.

[39] B. Navarro, A. Fonte, P. Fraisse, G. Poisson, and A. Cherubini, "In pursuit of safety: An open-source library for physical human-robot interaction," *IEEE Robot. Autom. Mag*, vol. 25, no. 2, pp. 39–50, 2018. doi: 10.1109/MRA.2018.2810098.

[40] Y. Wu, P. Balatti, M. Lorenzini, F. Zhao, W. Kim, and A. Ajoudani, "A teleoperation interface for loco-manipulation control of mobile collaborative robotic assistant," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3593–3600, 2019. doi: 10.1109/LRA.2019.2928757.

[41] K. Wansoo, M. Lorenzini, B. Pietro, W. Yuqiang, and A. Ajoudani, "Towards ergonomic control of collaborative effort in multi-human mobile-robot teams," in *Proc. 2019 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 3005–3011. doi: 10.1109/IROS40897.2019.8967628.

[42] P. H. Rosen and S. Wischniewski, "Task design in human-robot-interaction scenarios: Challenges from a human factors perspective,"

in *Proc. Int. Conf. Applied Human Factors and Ergonomics*, Cham, 2018, pp. 71–82. doi: 10.1007/978-3-319-60366-7_8.

[43] P. H. Rosen, S. Sommer, and S. Wischniwski, "Evaluation of human–robot interaction quality: A toolkit for workplace design," in *Proc. 20th Congr. Int. Ergonomics Association*, 2018, pp. 1649–1662. doi: 10.1007/978-3-319-96071-5_169.

[44] S. A. Elprama, C. I. Jewell, A. Jacobs, I. E. Makrini, and B. Vanderborght, "Attitudes of factory workers towards industrial and collaborative robots," in *Proc. Companion 2017 ACM/IEEE Int. Conf. Human-Robot Interaction*, pp. 113–114. doi: 10.1145/3029798.3038309.

[45] I. E. Makrini et al., "Working with Walt: How a cobot was developed and inserted on an auto assembly line," *IEEE Robot. Autom. Mag.*, vol. 25, no. 2, pp. 51–58, 2018. doi: 10.1109/MRA.2018.2815947.

[46] T. J. Armstrong et al., "Scientific basis of ISO standards on biomechanical risk factors," *Scand. J. Work Environ. Health*, vol. 44, no. 3, pp. 323–329, Jan. 2018. doi: 10.5271/sjweh.3718.

[47] R. F. Reardon, "The impact of learning culture on worker response to new technology," *J. Workplace Learn.*, vol. 22, no. 4, pp. 201–211, 2010. doi: 10.1108/13665621011040662.

[48] L. Fritzsche, J. Wegge, M. Schmauder, M. Kliegel, and K.-H. Schmidt, "Good ergonomics and team diversity reduce absenteeism and errors in car manufacturing," *Ergonomics*, vol. 57, no. 2, pp. 148–161, 2014. doi: 10.1080/00140139.2013.875597.

Arash Ajoudani, Fondazione Istituto Italiano di Tecnologia, Italy. Email: arash.ajoudani@iit.it.

Philipp Albrecht, Deutsches Institut für Normung, Germany. Email: philipp.albrecht@din.de.

Matteo Bianchi, Centro di Ricerca "Enrico Piaggio" and the Department of Information Engineering, Università di Pisa, Italy. Email: matteo.bianchi@centropiaggio.unipi.it.

Andrea Cherubini, LIRMM, Université de Montpellier, CNRS, France. Email: cherubini@lirmm.fr.

Simona Del Ferraro, Department of Occupational and Environmental Medicine, Epidemiology and Hygiene, INAIL, Italy. Email: s.delferraro@inail.it.

Philippe Fraisse, LIRMM, Université de Montpellier, CNRS, France. Email: fraisse@lirmm.fr.

Lars Fritzsche, imk automotive GmbH, Division Ergonomics, Germany. Email: lars.fritzsche@imk-automotive.de.

Manolo Garabini, Centro di Ricerca "Enrico Piaggio" and the Department of Information Engineering, Università di Pisa, Italy. Email: manolo.grabini@unipi.it.

Alberto Ranavolo, Department of Occupational and Environmental Medicine, Epidemiology and Hygiene, INAIL, Italy. Email: a.ranavolo@inail.it.

Patricia Helen Rosen, Federal Institute for Occupational Safety and Health (BAuA), Germany. Email: rosen.patricia@baua.bund.de.

Massimo Sartori, Department of Biomechanical Engineering, University of Twente, The Netherlands. Email: m.sartori@utwente.nl.

Nikos Tsagarakis, Fondazione Istituto Italiano di Tecnologia, Italy. Email: nikos.tsagarakis@iit.it.

Bram Vanderborght, BruBotics, Vrije Universiteit Brussels and Flanders Make, Belgium. Email: bram.vanderborght@vub.be.

Sascha Wischniewski, Federal Institute for Occupational Safety and Health, Germany. Email: wischniewski.sascha@baua.bund.de.

STUDENT'S CORNER *(continued from page 18)*

But being an engineer and making things work, innovate, and think differently is what makes us push ourselves. Our influence comes from the very fundamental fact that there are not so many women engineers in the industry. As IEEE Women in Engineering and being part of an IEEE project team, we try to give every day everything we've got and apply theory into practice in our technical field of study to achieve innovation. Self-discipline, goals, and consistency are key factors to achieve and complete the milestones that we set every week, and step by step we work among men, for a common goal: to complete the project and advance technology for humanity.

R8T: What's next for the Hermes Team?

After the first prototype, the exoskeleton will compete at Cybathlon 2020. The exoskeleton will be tested at the premises of our sponsor, Helexpo, Thessaloniki, to train our user pilot and see in action how the exoskeleton behaves. After that, many scenarios have been made. In our everyday life, each individual has a plan for personal and professional development. Some will pursue a master's degree in Europe; others have a dream to build a start-up. The main concept stays the same. We build an applicable device that can improve other people's lives. This is our IEEE dream, and we intend to keep it in our hearts. The Hermes exoskeleton will always be the beacon that reminds us who we are and what we do. Apparently, the new breed of engineers will take over the building of Mark II, and new features will advance technological innovation for the betterment of humanity.

São Carlos School of Engineering RAS Student Branch Chapter Hosts Roundtable on Automation and the Labor Market

The IEEE Robotics and Automation Society (RAS) Student Branch Chapter at São Carlos School of Engineering, University of São Paulo (EESC-USP), Brazil, organized a roundtable discussion on 7 November 2019: "Automation and the Labor Market: The Engineer's Profile for the Upcoming Industry." This first event for the recently formed chapter was targeted toward undergraduate and graduate students and included a discussion of the key characteristics of current engineering professionals in Brazil's labor market. Four engineers from different backgrounds and playing very distinct roles professionally, were invited to a panel discussion. They were João Soares, a former student from EESC-USP, who works for the human resources technology company Kenoby; Dr. Jorge Bidinotto, current professor at EESC-USP and former flight test engineer at the largest Brazilian aerospace conglomerate, Embraer S.A.; Jorge Rovilson, an electrical engineer working with automation at GROB Brazil; and RAS member Paulo Polegato, a Ph.D. candidate at EESC-USP with past experience in the field of agricultural robots from his time at Naïo Technologies in France (Figure 1).

Digital Object Identifier 10.1109/MRA.2020.2986569
Date of current version: 10 June 2020

Figure 1. (From left): Victor Noppeney, RAS member and panel host; Paulo Polegato; Dr. Jorge Bidinotto; Jorge Rovilson; João Soares; and Gustavo Lahr, EESC-USP RAS Student Branch Chapter chair.

Universidad Nacional de Colombia Bogotá Campus RAS Student Branch Chapter 2019 Recap

The IEEE Robotics and Automation (RAS) Student Branch Chapter and the Mechatronics Engineering Students Committee of the National University of Colombia, Bogotá campus, combined efforts and resources for activities and are now branded as *CEIMTUN*, to reflect their combined titles.

Much effort was expended on the development of the 12th edition of the largest free national robotics contest, UNRobot. The CEIMTUN team helped establish the RAS Colombia Robotics League initiative and added four new original categories: drone racing, line follower drag racing, Tetris, and the UNRobot Challenge. The new categories caught the attention of national media, including multiple mentions and interviews on radio stations, national TV, and newspapers. For the first time, the Chapter and university achieved official international participation in the Robot Rumble in México (Figure 1). For more information, visit the Chapter website: https://www.ingenieria.bogota.unal.edu.co/ceimtun/

Figure 1. CEIMTUN members at Robot Rumble México 2019. (Source: CEIMTUN.)

—Alejandro Ojeda and Camilo Campo, Universidad Nacional de Colombia

Digital Object Identifier 10.1109/MRA.2020.2986571
Date of current version: 10 June 2020

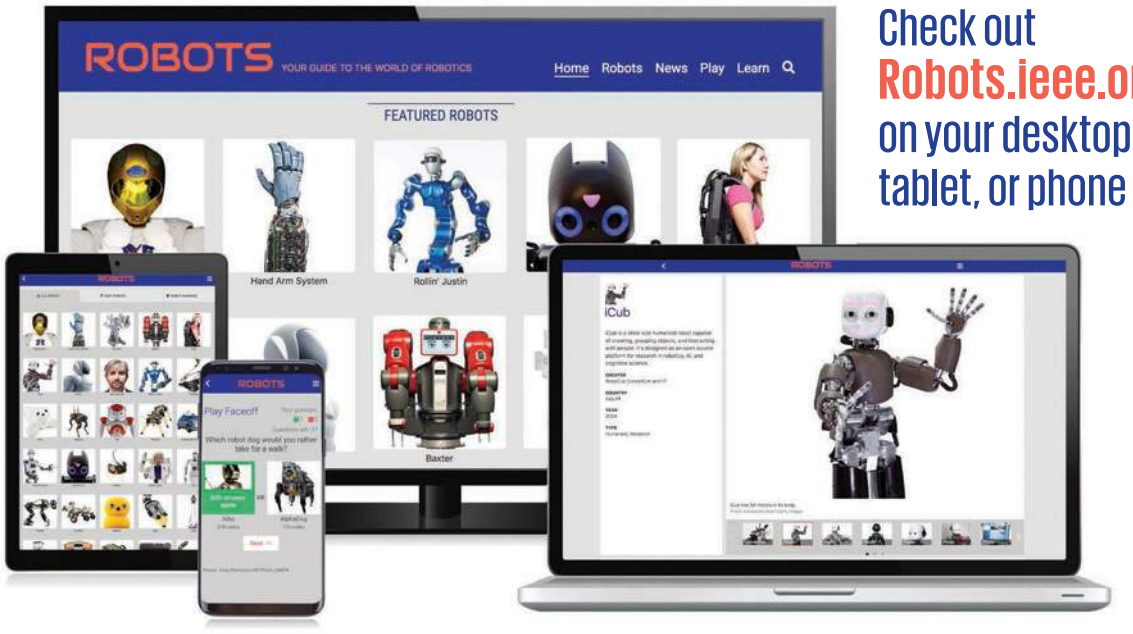
The World's Best ROBOTS GUIDE Is Here!

OP2
Courtesy of ROBOTIS

ROBOTS.IEEE.ORG

***IEEE Spectrum's* new ROBOTS site features more than 200 robots from around the world.**

- Spin, swipe and tap to make robots move.
- Read up-to-date robotics news.
- Rate robots and check their ranking.
- View photography, videos and technical specs.
- Play *Faceoff*, an interactive question game.



Check out Robots.ieee.org on your desktop, tablet, or phone now!

Digital Object Identifier 10.1109/MRA.2020.2986596

2020 IEEE Robotics and Automation Society Award Recipients Announced

The IEEE Robotics and Automation Society (RAS) recognizes and congratulates the following individuals for their outstanding accomplishments and service to the RAS and the robotics and automation community. Please join us in congratulating these outstanding recipients.

RAS Pioneer Award


Dieter Fox

Dieter Fox: "For pioneering contributions to probabilistic state estimation, RGB-D perception, machine learning in robotics, and bridging academic and industrial robotics research."


Lydia Kavraki

Lydia Kavraki: "For pioneering contributions to the invention of randomized motion-planning algorithms and probabilistic road maps."

RAS George Saridis Leadership Award in Robotics and Automation


John Hollerbach

John Hollerbach: "For foundational research on robot calibration and leadership and service in publications, conferences, and society management."


Peter Corke

Peter Corke: "For foundational research on robot vision and for leadership and innovation in robotics education."


Lynne Parker

Lynne Parker: "For foundational research in multirobot systems and leadership in RAS conferences and U.S. national AI policy."

IEEE RAS Distinguished Service Award


Eugenio Guglielmelli

Eugenio Guglielmelli: "For leadership in RAS conferences and publications including the establishment of *IEEE Transactions on Medical Robotics and Bionics*."


Kevin Lynch

Kevin Lynch: "For leadership, service, and innovations in Society publications, conferences, and governance."

Early Academic Career Award in Robotics and Automation


Eric Diller

Eric Diller: "For his contributions to magnetic wireless microscale robots."


Alberto Rodriguez

Alberto Rodriguez: "For his contributions to dexterous robot manipulation."

RAS Early Government or Industry Career Award in Robotics and Automation


Aleksandra Faust

Aleksandra Faust: "For contributions to the integration of machine learning with robot motion planning and control."

IEEE Robotics and Automation Award for Product Innovation

The Fetch Robotics logo.

Fetch Robotics: "For innovations in the VirtualConveyor line of cloud-based autonomous mobile robots for material handling and transport."

Digital Object Identifier 10.1109/MRA.2020.2986572
Date of current version: 10 June 2020

RAS Most Active Technical Committee Award

Technical Committee on Sustainable Production Automation: Chair Andrea Matta and Cochair Ying (Gina) Tang.

RAS Chapter of the Year Award

Private University of Tunis ULT (Université Libre de Tunis) IEEE Student Branch in the Tunisia Section: Advisor Cherif Mhamdi and Chair Khalil Benhaj.

Deadline for IEEE Robotics and Automation Society Local Chapter Initiative Grants

The IEEE Robotics and Automation Society (RAS) Member Activities Board (MAB) awards a limited number of Chapter initiative grants to RAS Chapters for professional development, educational outreach, and other programs. The MAB will review grant proposals at its fall 2020 meeting and award up to US$2,000 on a competitive basis. The deadline for proposals is 15 August 2020. For submission details, please visit: https://www.ieee-ras.org/chapters/support-for-chapters.

Digital Object Identifier 10.1109/MRA.2020.2986573
Date of current version: 10 June 2020

Welcome to 13 New IEEE Robotics and Automation Society Chapters

Congratulations and welcome to the following newly organized IEEE Robotics and Automation Society (RAS) Chapters.

Regions 1–6

- United States
  - University of North Texas RAS Student Branch Chapter in the Fort Worth Section.

Region 8

- Palestine
  - An-Najah National University RAS Student Branch Chapter in the Israel Section
- Tunisia
  - Faculty of Science of Tunis RAS Student Branch Chapter in the Tunisia Section
  - Polytech Sfax Society RAS Student Branch Chapter in the Tunisia Section.

Region 9

- Brazil
  - Universidade Federal do Espírito Santo–Campus São Mateus RAS Student Branch Chapter in the Rio De Janeiro Section
- El Salvador
  - Universidad de Sonsonate RAS Student Branch Chapter in the El Salvador Section
- Mexico
  - Instituto Tecnologico Superior de Patzcuaro RAS Student Branch Chapter in the Centro Occidente Section
- Peru
  - Trujillo Nacional University–La Libertad RAS Student Branch Chapter in the Peru Section.

Region 10

- India
  - Adi Shankara Institute of Engineering and Technology RAS/IEEE Biometrics Council/IEEE Nanotechnology Council/IEEE Sensors Council/IEEE Systems Council Student Branch Chapter in the Kerala Section
  - G H Patel College of Engineering and Technology RAS Student Branch Chapter in the Gujarat Section
  - Pranveer Singh Institute of Technology RAS Student Branch Chapter in the Uttar Pradesh Section
  - St. Joseph's College of Engineering and Technology, Palai, Student Branch Chapter in the Kerala Section
- Pakistan
  - Sir Syed Center for Advanced Studies in Engineering Institute of Technology RAS Student Branch Chapter in the Islamabad Section.

Digital Object Identifier 10.1109/MRA.2020.2986574
Date of current version: 10 June 2020

IEEE ROBOTICS & AUTOMATION MAGAZINE

Special Issue on robotics response for the COVID-19 outbreak

The coronavirus pandemic has quickly become the most dramatic and disruptive event experienced by this generation. The disease has spread very quickly around the world and the growing number of new infections and patients in need of intensive medical care has pushed clinical care beyond their limits, revealing a shortage of trained personnel and lifesaving equipment, such as ventilators. In addition, frontline health professionals operate in highly infectious areas, exposing themselves to the risk of becoming infected. The most common political response to mitigate the spread of the disease has been to promote social distancing, and locking down entire countries. Although being effective, these measures impose heavy social and economic consequences.
All that has an impact on the perspective we can take on robotics and its progress. The robots' possibilities for human replacement and remote operation in risky environments and tasks, as well as in proxying social interaction, have gained interest and value for potential help in the pandemic.

Robotics and automation technologies are already playing a critical role in this crisis, since testing and life supporting equipment are in general automated, but in the past months we have seen the human creativity emerge in the fight against this pandemic, using robots in applications never seen before, such as helping to protect people by disinfecting risky environments, detecting disease, monitoring social distancing, providing remote care, promoting social interaction of confined patients, supporting remote work, delivering medical supplies to hospitals and goods to persons at home or in hard to reach places, etc.
These applications, which typically involve the deployment of robots in normal living environments, their operation by non-skilled personnel and the interaction with the common population, impose significant research challenges that need to be addressed and overcome. Additionally, the use of robots for the regulated fields of health care and for public safety as well as for interaction with common citizens raises ethical, safety and reliability concerns that also need to be carefully considered.

This special issue, edited by the **IEEE RAS Special Interest Group on Humanitarian Technologies (SIGHT)**, aims to present up-to-date results and innovative advanced solutions on how robotics and automation technologies are used to fight the outbreak, giving particular emphasis to works involving the actual deployments of robots with meaningful analysis and lessons learned for the robotics community. The editors will accept both conventional full length contributions and short contributions reporting practical solutions to the problem that have proven effective in the field. The topics of interest for paper submissions include, but are not limited to:

- autonomous or teleoperated robots for hospital disinfection and disinfection of public spaces
- telehealth and physical human-robot interaction systems enabling healthcare workers to remotely diagnose and treat patients
- hospital and laboratory supply chain robots for handling and transportation of samples and contaminated materials
- robots use by public safety and public health departments for quarantine enforcement and public service announcements
- social robots for families interacting with patients or with relatives in nursing homes
- robots enabling or assisting humans to return to work or companies to continue to function
- case studies of experimental use of robots in the COVID-19 pandemic

Important Dates:

- **May 2020** – Call for papers
- **31 July 2020** – Submission deadline
- **15 September 2020** – First decisions
- **30 October 2020** – Resubmission
- **30 November 2020** – Final decision
- **10 December 2020** – Final manuscripts uploaded
- **March 2021** – Publication

Guest Editors:

Lino Marques
University of Coimbra

Cecilia Laschi
Scuola Superiore Sant'Anna (SSSA)

Satoshi Tadokoro
Tohoku University

Kaspar Althoefer
Queen Mary University of London

Robin Murphy
Texas A&M University

Digital Object Identifier 10.1109/MRA.2020.2994824

2020

16–19 June
MED 2020: Mediterranean Conference on Control and Automation. Saint-Raphaël, France. http://med2020.cran.univ-lorraine.fr/

22–26 June
UR 2020: International Conference on Ubiquitous Robots. Virtual Meeting. http://www.ubiquitousrobots.org/2020/index.php

3–6 July
ICARM 2020: International Conference on Advanced Robotics and Mechatronics. Shenzhen, China. http://www.ieee-arm.org/

6–10 July
AIM 2020: International Conference on Advanced Intelligent Mechatronics. Boston, Massachusetts, United States. http://aim2020.org/

13–17 July
2020 IEEE RAS Summer School on Multi-Robot Systems. Prague, Czech Republic. http://mrs.felk.cvut.cz/summer-school/

13–17 July
MARSS 2020: International Conference on Manipulation, Automation, and Robotics at Small Scales. Toronto, Canada. https://marss-conference.org/

17–19 July
ACIRS 2020: Asia-Pacific Conference on Intelligent Robot Systems. Singapore. http://www.acirs.org/

2–6 August
RCAR 2020: Conference on Real-time Computing and Robotics. Asahikawa, Japan. http://www.ieee-rcar.com/#/

10–12 August
IRCE 2020: International Conference on Intelligent Robotics and Control Engineering. Virtual Meeting. http://www.irce.org/index.html

20–24 August
CASE 2020: IEEE International Conference on Automation Science and Engineering. Virtual Meeting. https://www.imse.hku.hk/case2020/

31 August–2 September (postponed from July)
COINS: International Conference on Omni-Layer Intelligent Systems. Barcelona, Spain. https://coinsconf.com/

31 August–4 September
RO-MAN 2020: International Symposium on Robot and Human Interactive Communication. Naples, Italy. http://ro-man2020.unina.it/index.php

1–4 September
ICUAS 2020: International Conference on Unmanned Aircraft Systems. Athens, Greece. http://www.uasconferences.com/2020_icuas/

3–5 September
ICAC 2020 International Conference on Automation and Computing. Portsmouth, England. http://www.cacsuk.co.uk/index.php/conferences/icac-2-1

14–16 September
MFI 2020: International Conference on Multisensor Fusion and Integration. Karlsruhe, Germany. https://mfi2020.org/

18–20 September
CACRE 2020: International Conference on Automation, Control, and Robotics Engineering. Dalian, China. https://www.cacre.org/

8–10 October
ARSO 2020: International Conference on Advanced Robotics and Its Social Impacts. Aichi, Japan. http://ieee-arso2020.org/

13–16 October (postponed from August)
ICMA 2020: International Conference on Mechatronics and Automation. Beijing, China. http://2020.ieee-icma.org/

25–29 October
IROS 2020: International Conference on Intelligent Robots and Systems. Las Vegas, Nevada, United States. http://www.iros2020.org/

9–11 November
IRC 2020: International Conference on Robotic Computing. Taichung, Taiwan. http://www.ieee-irc.org/

18–20 November
ISMR 2020: International Symposium on Medical Robotics. Atlanta, Georgia, United States. http://www.ismr.gatech.edu/

20–22 November
ICRAE 2020: International Conference on Robotics and Automation Engineering. Singapore. http://www.icrae.org/

29 November–2 December
BioRob: IEEE International Conference for Biomedical Robotics and Biomechatronics. In-person or virtual attendance. New York City, United States. https://biorob2020nyc.org/

2–4 December
Humanoids 2020. Munich, Germany. https://humanoids-2020.org/

Digital Object Identifier 10.1109/MRA.2020.2986595
Date of current version: 10 June 2020

AD INDEX

The Advertisers Index contained in this issue is compiled as a service to our readers and advertisers: the publisher is not liable for errors or omissions although every efforts is made to ensure its accuracy. Be sure to let our advertisers know you found them through *IEEE Robotics & Automation Magazine*.

Advertiser	Page	URL	Phone
ATI Industrial Automation	11	www.ati-ia.com	+1 919 772 0115
Barrett Technology, Inc.	Cover 4	www.barrett.com	+1 617 252 9000
Butterfly Haptics	15	butterflyhaptics.com	
Force Dimension	Cover 2	www.forcedimension.com	+49 22 362 6570
KUKA Laboratories	3	www.kuka.com	
Nagamori Foundation	5	www.nagamori-f.org	

445 Hoes Lane, Piscataway, NJ 08854

IEEE ROBOTICS AND AUTOMATION MAGAZINE REPRESENTATIVE

Erik Henson
Naylor Association Solutions
Phone: +1 352 333 3443
Fax: +1 352 331 3525
ehenson@naylor.com

Digital Object Identifier 10.1109/MRA.2020.2995294

IEEE RAS SOCIETY AWARDS

CALL FOR NOMINATIONS

- ***RAS PIONEER AWARD***
- ***IEEE RAS GEORGE SARIDIS LEADERSHIP AWARD IN ROBOTICS AND AUTOMATION***
- ***IEEE RAS DISTINGUISHED SERVICE AWARD***
- ***IEEE RAS EARLY ACADEMIC CAREER AWARD IN ROBOTICS AND AUTOMATION***
- ***IEEE RAS EARLY GOVERNMENT OR INDUSTRY CAREER AWARD IN ROBOTICS AND AUTOMATION***
- ***IEEE INABA TECHNICAL AWARD FOR INNOVATION LEADING TO PRODUCTION***
- ***IEEE ROBOTICS AND AUTOMATION AWARD FOR PRODUCT INNOVATION***
- ***RAS MOST ACTIVE TECHNICAL COMMITTEE AWARD***
- ***RAS CHAPTER OF THE YEAR AWARD***

A description and list of previous recipients of each award is available at www.ieee-ras.org.

Nominators should use the appropriate Nomination Forms, which are available in the awards section of the site.

Nominations can be completed online or sent to ras@ieee.org.

Due 1 August 2020

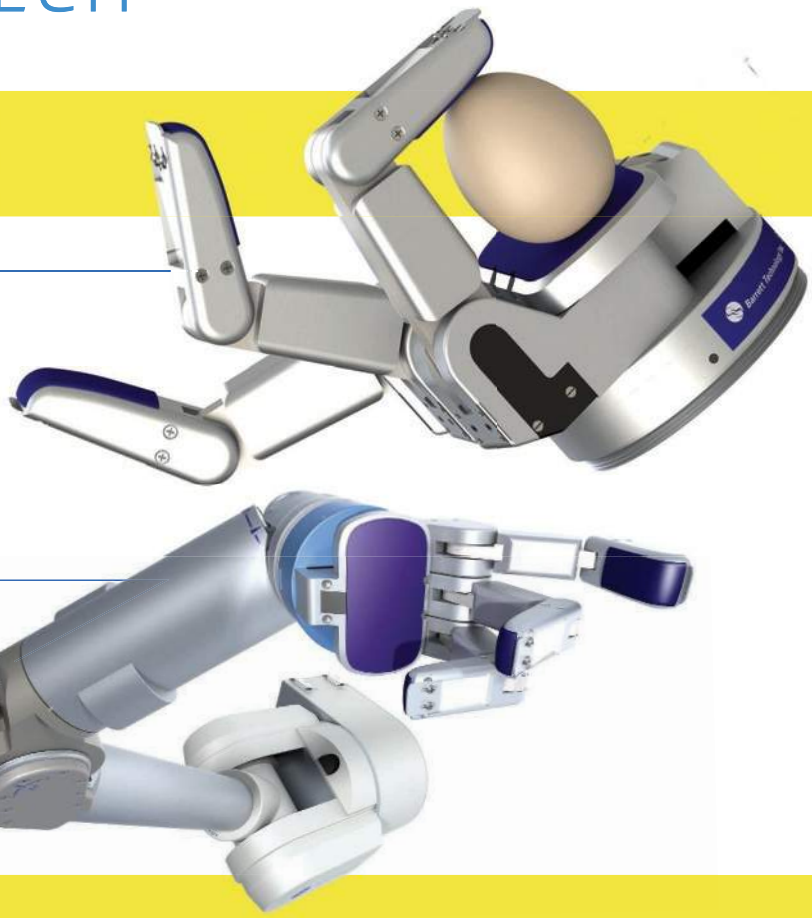
IEEE Robotics & Automation Society

IEEE Advancing Technology for Humanity

Digital Object Identifier 10.1109/MRA.2020.2986597

Barrett TECH

Advanced Robotics



BarrettHand™ BH8-282

- Multiple fingered
- Programmable grasper
- Self contained
- Fingertip sensing
- High precision
- PerceptionPalm (vision, range finding, and laser-grid emitter)

Barrett WAM® Arm

- Highly dexterous
- Backdrivable
- Brushless motors
- Transparent dynamics
- Lower power
- High precision
- No controller cabinet

Human-Robot Interaction Research

Robotics Application

- Teleoperation with force-feedback
- 3D haptics
- Virtual reality (VR)
- Augmented reality (AR)
- Advanced robotics research
- Training simulations
- Rehabilitation research
- Bilateral training
- Teletherapy

Barrett Upper-extremity Robotic Trainer (BURT®)

- Affordable
- Left/Right switchable
- Sitting standing
- Quick setup
- Unity plugin with physics engine
- Bilateral ready
- Slim
- Portable
- Silent
- Safe
- Intelligent
- Networked

Barrett Technology has a large global customer base of leading research organizations utilizing the proven technology of the WAM, BarrettHand, and now Burt for applications in vision, AI, mobile systems, medical, haptics, advanced manipulation, and automation. Visit us at **www.barrett.com** send email to **robot@barrett.com**, or phone us at **US+1.617.252.9000.**