F3

Faculty of Electrical Engineering
Department of Cybernetics

Doctoral Thesis

Safe Autonomous Reinforcement Learning

Martin Pecka

Supervisor: Tomáš Svoboda
Supervisor–specialist: Karel Zimmermann
Ph.D. Programme: P2612 Electrical Engineering and Information Technology
Branch of study: Artificial Intelligence and Biocybernetics
May 2020, Prague

Acknowledgments | Poděkování

I would like to thank my supervisors doc. Tomáš Svoboda and doc. Karel Zimmermann for the great effort they have put into helping me become a researcher and roboticist. Tomáš's "questions from distance" and Karel's high-level views both greatly improved the research we were conducting together, which resulted in several papers we have published. I have learned a lot working with them. I am also grateful for Tomáš's courage to join various international projects and competitions which have increased the impact of our work and brought me numerous friends and experience. Also, Tomáš's benevolent, yet steady hand as a supervisor allowed me to become not only a researcher, but also a knowledgable robotics programmer and a member of robotics open source community.

A big thanks also belongs to my colleagues and coworkers at Faculty of Electrical Engineering and CIIRC. Dr. Michal Reinštein is a well of knowledge about Gaussian processes and deep learning, dr. Tomáš Petříček's deep insights into many fields from mapping to professional software development have brought me a lot of useful ideas and dr. Vláďa Kubelka's knowledge about inertial navigation systems and sensors was highly inspirational. I am also very grateful to him for having the patience to explain basic electronics knowledge. Among others, I would like to thank prof. Václav Hlaváč, Libor Wagner, dr. Vláďa Smutný, Vojta Šalanský, Ota Jašek and Teymur Azayev for fruitful discussions on various subjects, or just for the good time spent together.

Czech Technical University provided me good conditions for doing all the research and other related activities. I would like to thank all the people whose work together created this inspirational environment. Most of all the heads of departments prof. Václav Hlaváč, prof. Jan Kybic and doc. Tomáš Svoboda. And also the administration staff who tried their best to fulfill our (sometimes hasty) requests, always with professionality—Eva and Klára Matysková, Hana Pokorná, Pavla Marešová, Hana Krautwurmová, Jitka Seguin, Petra Žufníčková, Kristina Lukešová, Petra Ivaničová. Also Dan Večerka and Jan Kreps deserve kudos for their work on IT support and infrastructure.

The whole TRADR team has a special place in my heart. We did quite a great piece of work together. All these long and chill shifts in places never meant to serve as developer bases; the joy and fear watching firemen playing with the precious robots, hoping that our late-night hotfixes would bring some fruits. But also many happy moments spent together just chatting and feeling good. Thank you, Ivana Kruijff-Korbayová, Erik Zimmermann, Rainer Worst, Joachim de Greef, Luigi Freda, Mario Gianni, Abel Gawel, Renaud Dube, Fredrik Båberg, Sergio Caccamo, Daniel Reuter, Chris Rozemuller, Ellie Saad, Tina Mioch, Nanja Smets, Timi Bagosi, Norbert Pahlke, Robert Maul, Ramviyas Parasuraman, Massimiliano Russo and many others.

I would also like to thank the numerous people involved in the CTU-CRAS-Norlab team for SubT Challenge—both for the great work we have done together and for being a good team of people with various interests that is able to stay in one small cottage for almost a month. That is not to be taken for granted, and I appreciate and enjoy the moments spent together chasing the elusive, though reachable goal.

Throughout the years spent with the robots, I used an enormous amount of open-source software developed by thousands of unknown people who have put their efforts together to create an inspir-

ing and reliable background for all work I needed to do. From the few I got to know, I would especially like to thank Louise Poubel, Dirk Thomas, Michael Görner and Robert Haschke for taking the time to review my pull requests and help me improve my code to be a good and reliable contribution to the open source community.

Over the years, my research work has been funded by several projects which I would like to acknowledge: EU FP7 project TRADR[1], H2020 project ENABLE-S3[2] and structural funds project RCI[3]; multiple projects from the Czech Grant Agency[4] and CTU (via SGS grants)[5], and finally study stipends from the Czech state budget.

A special memory belongs to Petr Zuzánek, who's still been in my heart as the smiling guy who did good research, hard work and knew how to enjoy life. His premature passing stroke all of us and left just a void... Have a good rest, maybe stepping a path in sand on a beach.

V neposlední řadě bych chtěl poděkovat Kátě za neobyčejnou podporu mého výzkumu a mých cest s roboty i bez nich, za motivaci pokračovat a být stále lepší, a za občasné vytáhnutí z lákavého IT doupěte do krásné a volné přírody. Velké díky patří i rodičům: za výchovu, která mi ukázala jasný směr životem, a že kráčet cestou dobra se může vyplatit; za podporu věčně cestujícího syna; i za domov, kam se stále rád vracím.

[1] FP7-ICT-609763 TRADR

[2] financed by ECSEL Joint Undertaking under Grant 692455

[3] OP VVV MEYS funded project CZ.02.1.01/-0.0/0.0/16_019/0000765 "Research Center for Informatics"

[4] GA14-13876S and GA17-08842S

[5] SGS13/142/OHK3/2T/13,
SGS15/081/OHK3/1T/13,
SGS16/161/OHK3/2T/13,
SGS18/138/OHK3/2T/13

Abstract

Reinforcement Learning is a technique proven by uncountable use-cases in the robotics community and many other machine-learning fields. It allows training optimal decision policies without knowing precisely which actions are the best at any given moment. A reward function and some number of policy rollouts suffice to estimate the best decision policy.

However, for complex systems or tasks, the number of required rollouts goes to millions or even more. That is not suitable for mobile robots.

This thesis shows several ways of dealing with the problems mentioned above. During our research, we have concluded that the best way to overcome the real-world limitations is to employ simulation and machine learning to some degree. In the first part we explore ways in which guaranteed safe reinforcement learning can be implemented. In the second part we show how active exploration helps to deal with uncertain or even unknown measurements, and how to integrate active exploration into existing machine learning algorithms. The last part shows how imprecise simulation can help training a policy that performs well even in the real world. That is not possible without interaction with the real world, but the presented method greatly decreases the number of required real-world experiments.

Keywords: reinforcement learning, machine learning, autonomous, safe learning, mobile robots, tracked robots, simulation, ROS, Gazebo

Supervisor: Tomáš Svoboda
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo náměstí 13, 121 35, Prague 2
Czech Republic

Abstrakt

Technika posilovaného učení již nesčetněkrát prokázala svou užitečnost v robotice a dalších aplikacích strojového učení. Dovoluje učit strategie řízení robotů bez přesné znalosti, jaká akce je ve kterém stavu ideální. K nalezení optimální strategie stačí dodat funkci užitku a několikrát systém spustit.

Pro komplexní systémy či úlohy ovšem roste počet nezbytných testů na reálném systému do milionů. To je bohužel na mobilních robotech nepřípustné.

V této dizertaci představujeme několik cest, jak se s výše zmíněnými problémy vypořádat. V průběhu výzkumu jsme zkonstatovali, že nejlepší cesta k překonání problémů spojených s učením v reálném světě bude částečné využití simulace. V první části práce rozebíráme způsoby, jakými se dá dosáhnout zaručeně bezpečného posilovaného učení. Druhá část ukazuje, jak se pomocí aktivního průzkumu vyrovnat s nepřesnými, či dokonce chybějícími senzorickými měřeními, a jak aktivní průzkum kombinovat s existujícími metodami strojového učení. Poslední část je věnována myšlence, jak lze využít nepřesný simulátor pro naučení strategií, které budou úspěšné i v reálném světě. To samozřejmě úplně bez interakce s reálným robotem nelze, ale navržená metoda počet takových interakcí výrazně snižuje.

Klíčová slova: posilované učení, strojové učení, autonomní, bezpečné učení, mobilní robot, pásový robot, simulace, ROS, Gazebo

Překlad názvu: Bezpečné autonomní posilované učení

Contents

Figures

Tables

Chapter 1

Introduction and Goals

Autonomous capabilities of mobile robots are more and more crucial for their adoption by end-users. The times of pure teleoperation have long been gone, as well as pre-programmed behaviors ignoring the outer world.

Cybernetics, a field established in late 40s, specifies the direction towards machines that are aware of their environment and act according to feedback gathered by their sensors. For a long time, real applications of cybernetics were limited to simple systems operating in well known or controlled conditions. With the development of better sensors, processors with higher computational capabilities and novel methods, robots started to take their first steps into the unstructured real world.

Currently, Reinforcement Learning seems to be one of the key methods that allow robots to learn good decision policies even for complex tasks. When a precise model of the system (also called *transition probabilities*) is known and every action can be rewarded by a known quantity, the algorithm breaks down to iterations of the Bellman equation until the underlying dynamics system converges.

Problems arise when either the model is known only approximately or not at all. Likewise, episodic rewards (which are awarded only at the end of a whole sequence of actions) impose higher demands on computational power and the amount of real-world interactions. Autonomous cars are a good example of mobile robots in less structured environments. There are still some guarantees given by e.g. road construction standards and traffic laws, but the sensory input can vary greatly, each road is a little different, and so on. Car makers are big companies that have the resources to gather big datasets, construct test courses for training the algorithms, and could potentially afford to run a standard reinforcement learning algorithm on the real robot and gather thousands or even millions of trajectories.

Search and rescue robotics, on the other hand, is a small field with not so many resources, but with even more challenging environment. The robots have to traverse previously unmapped terrain, overcome various obstacles (or decide they can not overcome them), the area is often full of dust, smoke or other substances that degrade sensor performance, geometric priors do not hold... Even in such a hostile environment, search and rescue robots are expected to explore large areas and transmit valuable information to other

rescue team members. Developing algorithms for such robots is very different from car making: gathering datasets is usually expensive and even dangerous, so not so much data can be collected. The datasets usually try to cover large variety of possible environments, but cannot capture them in high detail or many times. Thus, the algorithms have to be written in such a way that they can distill the most out of the limited data collection.

In this thesis we present several contributions that help making reinforcement learning in unstructured environments possible. We begin with definitions and description of the very basic algorithms reinforcement learning builds upon. Second, in Part I, we present a theoretical framework which allows (guaranteed) safe behavior in the exploration phase of reinforcement learning. That, of course, comes with some important limitations on structure of the environment, of the dynamics model and of the tasks. Next, we explore the ways of dealing with incomplete measurements in reinforcement learning. We also provide an efficient algorithm for balancing the cost of finding the missing pieces of data and exploring larger areas (Part II). In Part III, we present a way how simulators can be used to leverage the utility of data collected during real-world execution. That is a core part of success in learning for search and rescue robotics. The millions of samples needed for reinforcement learning are done mostly in simulation and real data are used just to "calibrate" or guide the simulator. This way the real robot is kept safe for the largest part of the learning loop and only a few real-world trajectories are needed. In Part IV, we conclude the thesis and outline possibilities for followup work.

This whole thesis depends on a simulator of the robot behavior. At the beginning of the Ph.D. study, research was conducted showing that there are no publicly available simulators that would be capable of proper and fast simulation of tracked vehicles. That needed to be fixed, so I have implemented a tracked vehicle motion model in Gazebo and published it as open-source. As this topic is interesting, but stands a bit aside from the main topic of this thesis, the research describing the motion model is attached in Appendix A.

The concepts presented in this thesis have not only been tested in various experimental sessions verifying the particular algorithms, but also during two international research projects in the field of search and rescue. *TRADR project* [Kruijff-Korbayová et al., 2015] aimed at finding ways how humans and robots can form a rescue team together and make use of what each of them is best at. The robots had to reach high level of autonomy so that they were useful during the mission and not a burden. Specifically, related to this thesis, the robots were controlling their flippers via the *autonomous traversal* algorithm presented in this work. That helped the rescuers to concentrate on the mission goals and not on how to overcome each obstacle that came into the way of the robots. In *DARPA SubT Challenge* [DARPA, 2019], a team of robots is expected to enter an unknown underground tunnel system and search for traces of survivors. This challenge requires almost full autonomy of the robots, because communication is very limited in tunnel systems. Each robot is therefore responsible for planning its own path and traversing any

obstacles on its way, or detecting which passages it can not traverse due to its physical limits. That is also a place where autonomous capabilities described in this thesis come into play. These projects are described in detail in Appendix B.

Chapter 2

Basics

2.1 Reinforcement learning basics

This section reviews the basic definitions and algorithms used throughout this thesis. The reader should make sure he/she is familiar with these terms. The following subsections are mostly excerpts from our publications mentioned in Appendix E.

2.1.1 Markov Decision Processes

Markov Decision Processes (MDPs) are the standard model for deliberating about reinforcement learning problems. They are based on a lot of simplifications, but are sufficiently robust to describe a large set of real-world problems.

The simplest discrete stochastic MDP [Kaelbling et al., 1996] comprises:

- a finite set of states $\mathbf{S}$
- a finite set of actions $\mathbf{A}$
- a stochastic *transition model* $\mathbf{P}$:
  $\mathbf{P}_t(s, a, s') = Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ for each $s, s' \in \mathbf{S}$, $a \in \mathbf{A}$, where Pr stands for probability
- and the *immediate reward function* $\mathbf{R}\colon \mathbf{S} \times \mathbf{A} \to \mathbb{R}$ (or $\mathbf{R}\colon \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to \mathbb{R}$ if the reward depends on the stochastic action result)

To interpret this definition, we say that the at every time instant t the *agent* is in a state s_t, and by executing action a_t it gets to a new state s_{t+1}. Furthermore, executing a particular action in a particular state may bring *reward* r_t to the agent (defined by $\mathbf{R}$).

The most important and interesting property of MDPs is the *Markov property*. Looking at the definition of the transition model, the next state only depends on the current state and the chosen action. Particularly, the next state is independent of all the previous states and actions but the current one. To give an example, the robot's battery level cannot be treated implicitly

by counting the elapsed time, but rather it has to be modeled as a part of the robot's state.

Once the model is set up, everything is ready for utilizing an MDP. "The agent's job is to find a policy π mapping states to actions, that maximizes some long-run measure of reinforcement" [Kaelbling et al., 1996]. The "long-run" may have different meanings, but there are two favorite optimality models: the first one is the *finite horizon* model, where the term $\mathbf{J}_\pi = \sum_{t=0}^{h} r_t$ is maximized (h is a predefined time horizon and r_t is the reward obtained in time instant t while executing policy π). The dependency of r_t on the policy is no longer obvious from this notation, but this is the convention used in literature when it is clear which policy is used. This model represents the behavior of the robot which only depends on a predefined number of future states and actions.

The other optimality model is called *discounted infinite horizon*, which means we maximize the discounted sum $\mathbf{J}_\pi = \sum_{t=0}^{\infty} \gamma^t r_t$ with $\gamma \in (0, 1)$ being the *discount factor*. The infinite horizon tries to find a policy that is the best one taking into account the whole future. Please note the hidden dependency on the policy π (and the starting state s_0)—it is the policy that decides on which action to take, which in turn specifies what will the reward be.

Other extensions of MDPs to continuous states, time or actions are beyond the scope of this overview. However, some referenced papers make use of these continuous extensions, which proved to be useful for practical applications.

2.1.2 Reinforcement Learning

Reinforcement learning (RL) as an MDP-solving method has been thoroughly examined since 80's. In 1981, [Barto et al., 1981] inspired themselves in the reinforcement learning discoveries in behavioral psychology and devised the *Temporal Difference* machine learning algorithm that had to simulate psychological classical conditioning. In contrast with *supervised learning*, reinforcement learning does not need a teacher's classification for every sample presented. Instead, it just collects rewards (or punishment) on-the-go and optimizes for the expected long-term reward (whereas supervised learning optimizes for the immediate reward). The key advantage is that the design of the rewards is often much simpler and straight-forward than classifying all data samples.

2.1.3 Value Iteration

Value iteration is one of the basic reinforcement learning methods. The long-term reward corresponds to the *value* of each state [Kaelbling et al., 1996]. From such values, we can compose a *policy* which tells the agent to always take the action leading to the state with the highest value. As an addition, state values can be easily interpretable for humans.

To describe the Value iteration algorithm, it is first needed to define the essential notion of the *optimal value of a state*. In this whole subsection we

suppose the discounted infinite horizon model, but analogous results can be shown for finite horizon, too. "The optimal value of a state is the expected infinite discounted sum of reward that the agent will gain if it starts in that state and executes the optimal policy." [Kaelbling et al., 1996] Given a policy π, the induced value function is therefore defined as

$$\mathbf{V}_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} r_k \gamma^k\right], \tag{2.1}$$

where $\mathbb{E}$ denotes the expected value and r_k are the rewards for executing policy π. Taking the best value function over all policies then yields the *optimal value function* $\mathbf{V}^*$: [Kaelbling et al., 1996]

$$\mathbf{V}^*(s) = \max_\pi \mathbf{V}_\pi(s)\,. \tag{2.2}$$

Inversely, if the value function is given, we can derive a policy from it. It is a simple policy that always takes the action leading to the most profitable neighbor state (with the highest value).

One useful formulation of the properties of the optimal value function is the formulation using the recurrent *Bellman equations* which define a dynamic system that is stable for the optimal value function. We can say a state's optimal value is the best immediate reward plus its best neighbor's optimal value: [Kaelbling et al., 1996]

$$\mathbf{V}^*(s) = \max_{a \in \mathbf{A}} \left(\mathbf{R}(s,a) + \gamma \sum_{s' \in \mathbf{S}} \mathbf{P}(s,a,s') \cdot \mathbf{V}^*(s') \right). \tag{2.3}$$

Analogously, we can find the optimal policy using the same Bellman equation:

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathbf{A}} \left(\mathbf{R}(s,a) + \gamma \sum_{s' \in \mathbf{S}} \mathbf{P}(s,a,s') \cdot \mathbf{V}^*(s') \right). \tag{2.4}$$

The Value iteration algorithm is based on trying to compute the solution of Equation 2.4 using iterative Bellman updates (refer to Algorithm 1). In the algorithm, a structure called $\mathbf{Q}$ is used to store the "value" of state-action pairs. In Value iteration it is just a structure to save intermediate results, but it is the core of the Q-learning algorithm (described in section 2.1.4). The stopping criterion of the Value iteration algorithm is not obvious, but Williams and Baird [Williams and Baird, 1993] derived an easily applicable upper bound on the error of the computed value function.

That said, after a sufficient number of those simple iterations, we can compute the almost optimal value function. The number of iterations needed for Value iteration to converge may be impractically high, but it is shown that the optimal policy converges faster [Bertsekas, 1987], thus making Value iteration practical.

Algorithm 1 The Value iteration algorithm [Kaelbling et al., 1996]

Input: an MDP (states $\mathbf{S}$, actions $\mathbf{A}$, rewards $\mathbf{R}$, transition model $\mathbf{P}$)
Output: the optimal value function $\mathbf{V}^*$, resp. the optimal policy π^* derived from the value function

1. $\mathbf{V}(s) :=$ arbitrary function
2. $\pi :=$ the policy derived from $\mathbf{V}$
3. **while** π is not good enough **do**
4. **for all** $s \in \mathbf{S}$ **do**
5. **for all** $a \in \mathbf{A}$ **do**
6. **Update:** $\mathbf{Q}(s,a) := \mathbf{R}(s,a) + \gamma \sum_{s' \in \mathbf{S}} \mathbf{P}(s,a,s')\mathbf{V}(s')$
7. **end for**
8. $\mathbf{V}(s) := \max_a \mathbf{Q}(s,a)$
9. **end for**
10. $\pi :=$ the policy derived from $\mathbf{V}$
11. **end while**
12. $\mathbf{V}^* := \mathbf{V}$, $\pi^* := \pi$

2.1.4 Q-Learning

Just a small change to the Value iteration algorithm results in Q-learning [Watkins and Dayan, 1992]. The basic algorithm is the same as Value iteration, just the update step is done differently (refer to Algorithm 2). The consequence of this change is that no model of the system (transition function $\mathbf{P}$) is needed. It is sufficient to execute all actions in all states equally often, and [Watkins and Dayan, 1992] proved that if Q-learning were run for an infinite time, the computed $\mathbf{Q}$ would converge to the optimal $\mathbf{Q}^*$ (an analogue of $\mathbf{V}^*$). Q-learning is often used in connection with safe exploration.

2.1.5 Policy Iteration

Policy iteration is a completely different approach to computing the optimal policy. Instead of deriving the policy from the Value or Q function, Policy iteration works directly with policies [Howard, 1960]. In the first step, a random policy is chosen. Then a loop consisting of policy evaluation and policy improvement repeats as long as the policy can be improved [Kaelbling et al., 1996] (refer to Algorithm 3 for details). Since in every step the policy gets better, and there is a finite number of different policies, it is apparent that the algorithm converges [Puterman, 1994].

Policy iteration can be initialized by a known, but suboptimal policy. Such policy can be obtained e.g. by a human operator driving the UGV. If the initial policy is good, Policy iteration has to search much smaller subspace and should thus converge more quickly than with a random initial policy [Garcia Polo and Rebollo, 2011].

Algorithm 2 The Q-learning algorithm [Kaelbling et al., 1996]

Input: an MDP (states $\mathbf{S}$, actions $\mathbf{A}$, rewards $\mathbf{R}$, transition model may be unknown), learning rate $\alpha \in \mathcal{R}^+$

Output: the optimal state-value function $\mathbf{Q}^*$, resp. the optimal policy π^* derived from the state-value function

1. $\mathbf{Q}(s,a)$:= arbitrary function
2. π := the policy derived from $\mathbf{Q}$
3. **while** π is not good enough **do**
4. **for all** $s \in \mathbf{S}$ **do**
5. **for all** $a \in \mathbf{A}$ **do**
6. **Update:** $\mathbf{Q}(s,a) \mathrel{+}= \alpha\Big[\mathbf{R}(s,a) + \gamma \max_{a'} \mathbf{Q}(s',a') - \mathbf{Q}(s,a)\Big]$
7. **end for**
8. **end for**
9. π := the policy derived from $\mathbf{Q}$
10. **end while**
11. $\mathbf{Q}^* := \mathbf{Q}$, $\pi^* := \pi$

Algorithm 3 The Policy iteration algorithm [Kaelbling et al., 1996]

1. $\pi' =$ arbitrary policy
2. **repeat**
3. $\pi := \pi'$
4. **Policy evaluation:** (system of linear equations)
   $\mathbf{V}_\pi(s) = \mathbf{R}(s,\pi(s)) + \gamma \sum_{s' \in \mathbf{S}} \mathbf{P}(s,\pi(s),s')\mathbf{V}_\pi(s')$
5. **Policy improvement:**
   $\pi'(s) := \operatorname{argmax}_{a \in \mathbf{A}} \Big[\mathbf{R}(s,a) + \gamma \sum_{s' \in \mathbf{S}} \mathbf{P}(s,a,s')\mathbf{V}_\pi(s')\Big]$
6. **until** $\pi = \pi'$

2.1.6 Policy Gradient Search (PGS)

Policy Gradient methods are used in the continuous setting when the policy $\pi(s)$ has a parametric form and can be written as $\pi(s|\boldsymbol{\omega})$, where $\boldsymbol{\omega}$ is a parameter vector. They stochastically optimize the expected sum of rewards by direct sampling in the policy parameter space. Thus, the learning performance of PGS methods does not depend on the complexity of controlled systems, but only on the number of policy parameters that are optimized [Tedrake et al., 2009]. The general algorithm is described in Algorithm 4. It is based on gradient ascent of the objective function

$$\mathbf{J}(\pi(\cdot|\boldsymbol{\omega})) = \mathop{\mathbb{E}}_{\text{all trajectories } (s_i,\pi(s_i|\boldsymbol{\omega}),r_i)|_{i=0}^{T}} \left[\sum_{i=0}^{T} \gamma^{T-i+1} r_i\right] \tag{2.5}$$

As the policy is fully determined by the parameters $\boldsymbol{\omega}$, the objective function

notation is usually simplified to

$$\mathbf{J}(\boldsymbol{\omega}) = \mathbf{J}_{\pi}(\boldsymbol{\omega}) = \mathbf{J}(\pi(\cdot|\boldsymbol{\omega})) \tag{2.6}$$

Algorithm 4 The Policy Gradient Search algorithm [Kaelbling et al., 1996]

1. $\boldsymbol{\omega'}$ = random parameter vector
2. **repeat**
3. $\quad \tau_{\boldsymbol{\omega'}}$ = sample trajectory $(s_i, a_i, r_i)_{i=0}^{T}$ induced by policy $\pi(\boldsymbol{\omega'})$
4. $\quad \mathbf{J}(\boldsymbol{\omega'}) = \mathbb{E}\left[\sum_{i=0}^{T} \gamma^{T-i+1} r_i\right]$
5. $\quad \boldsymbol{\omega'} = \boldsymbol{\omega'} + \alpha \frac{\partial \mathbf{J}(\boldsymbol{\omega'})}{\partial \boldsymbol{\omega'}}$
6. **until** $\boldsymbol{\omega} \simeq \boldsymbol{\omega'}$

If the policy is differentiable, we can make use of the so called *log-ratio trick* to compute the policy gradient from a rolled-out trajectory $(s_i, a_i, r_i)|_{i=0}^{T}$:

$$\frac{\partial \mathbf{J}(\boldsymbol{\omega'})}{\partial \boldsymbol{\omega'}} \simeq \sum_{i=0}^{T} \nabla_{\boldsymbol{\omega'}} \log \pi(a_i|s_i, \boldsymbol{\omega'}) \left(\sum_{t=i+1}^{T} \gamma^{t-i-1} r_t\right) \tag{2.7}$$

2.2 Absolem UGV

In this section, we describe the custom-made robotic platform used throughout the whole thesis for experimental verification of the results. An overview look at the platform is given in Figure 2.1 and Figure 2.2. It is a tracked ground robot of approximate dimensions $0.8 \times 0.5 \times 0.5$ meters and weight 25 kilograms (and more in case additional sensors are added). Its electromotors and onboard devices are powered by Lithium-Ion batteries which provide 1–3 hours of run time. The two main tracks are attached to the body via hinge joints and are connected with a lockable differential that ensures that either (i) the relative pose of tracks against the body is fixed, or (ii) the angle of left track is inverse to the angle right track relative to the body (the tracks make opposite movements). Each main track has two flippers on its ends, whose angle can be controlled independently. The main tracks are velocity-controlled, and flippers provide either positional or velocity control. There is also a user-configurable limit on maximum current flowing in the flipper motors. If the current should get higher than this limit, the motors switch to a force-compliant mode and let the flipper deviate from the positional/velocity target. This is utilized to set the "stiffness" of the flippers when the robot supports or lifts itself with them.

Main sensory equipment is IMU providing gyro-odometry, an omnidirectional camera Ladybug LB-3 which provides a 360° view of the robot surroundings, and 2D lidar Sick LMS-151 on a rotating pivot. The rotation of the joint is precisely measured and thus a full 3D scan of robot surroundings can be captured using the 2D lidar once every 3 s (which is the duration of

one revolution of the joint). The position of the lidar on the robot body allows it to capture points on various parts of the robot body itself, thus a laser filtering pipeline is implemented which subtracts the expected 3D shape of the robot body from the laser scans so that they only capture the environment and not the robot body itself. For some tasks, a stereo camera can be attached to the robot. We have mainly tried cameras Intel RealSense R200 and D435. The robot has multiple options for wireless communication, but they are not important for this thesis, so we omit the details.

The control software runs on an industrial-grade computer which contains a 4-core/8-thread x86_64 CPU, 16 GB RAM, and SSD drive. Operating system Ubuntu Linux provides a base for the ROS middleware which is a core part of the control software. We use some publicly available ROS packages, but the core functionality is tailor-made for this type of robot. It is running INS odometry fused with track odometry as a base localization algorithm, and combines it with ICP laser mapping to get more precise localization and build long-term maps. Also, angles of flippers relative to their main tracks are available, as well as the angle of main tracks against the robot body. There are 5 basic combinations of flipper positions and motor current limits that were found useful when teleoperating the robot; we refer to them either as *flipper pose*, *flipper posture* or *flipper configurations*.

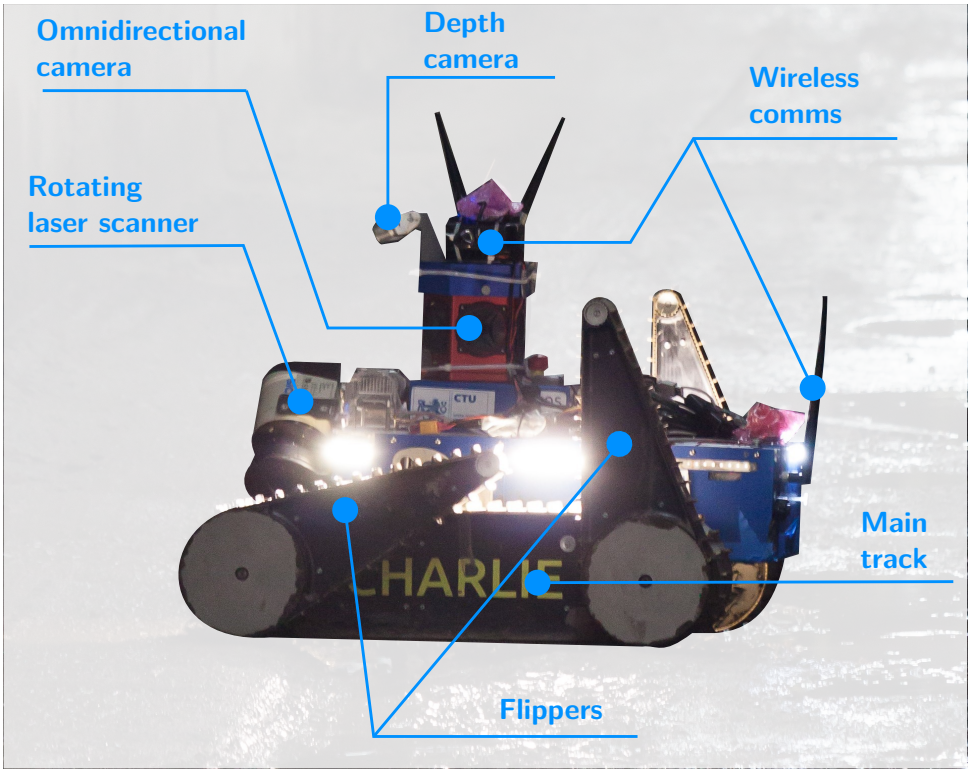


Figure 2.1: Absolem UGV robotic platform—side view

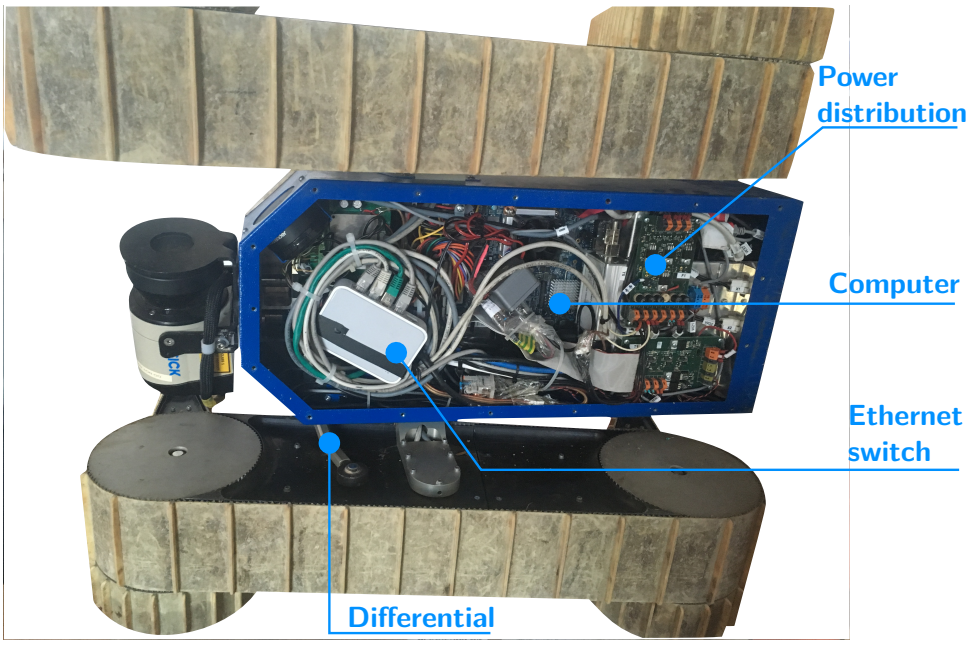


Figure 2.2: Absolem UGV robotic platform—bottom view

Part I

Safe Exploration

In this part, we overview different approaches to safety in (semi)autonomous robotics. Particularly, we focus on how to achieve safe behavior of a robot if it is requested to perform exploration of unknown states. A safe or at least gentle-to-wear, but not completely prohibitive exploration is desired for any interaction with a real system. The presented methods are studied from the viewpoint of reinforcement learning. To collect training data for reinforcement learning, the robot is required to freely explore the state space—which can lead to possibly dangerous situations. The role of safe exploration is to provide a framework allowing exploration while preserving safety. The examined methods range from simple algorithms which utilize precise physical models to sophisticated methods based on previous experience or state prediction. We also address the issues of how to define safety in the real-world applications. Further, our algorithm called CREPS is presented, which can be used to perform safe exploration on a real robotic platform.

The text of this part is largely based on the following papers. It has been edited to fit the thesis structure and goals.

- Pecka, M. and Svoboda, T. (2014). Safe Exploration Techniques for Reinforcement Learning – An Overview. In Hodicky, J., editor, *Modelling and Simulation for Autonomous Systems*, Lecture Notes in Computer Science, pages 357–375, Rome, Italy. Springer. DOI: 10.1007/978-3-319-13823-7

- Pecka, M., Zimmermann, K., and Svoboda, T. (2016). Autonomous Flipper Control with Safety Constraints. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2889–2894, Daejon, South Korea. IEEE. DOI: 10.1109/IROS.2016.7759447

Chapter 3

Safe Exploration Techniques

3.1 Introduction

What do most reinforcement learning methods have in common, is the need for rather large training data sets. For simulated discrete environments it is usually not a problem. But with real robotic hardware, the collection of training samples is not only lengthy, but also dangerous (be it mechanical wear or other effects). Another common feature of RL algorithms is the need to enter *unknown* states, which is inherently unsafe.

Safety is an important issue connected with reinforcement learning, but is often neglected since many algorithms are only tested in simulations. Thus, the first articles focused on maintaining safety during exploration started to appear much later after the "discovery" of RL. Among the first, [Heger, 1994] "borrowed" the concept of a *worst-case criterion* from control theory community. In 1994 he created a variant of Q-learning where maximization of long-term reward is replaced with maximization of minimum of the possible rewards. That basically means his algorithm prefers to never encounter a bad state (or, at least to choose the best of the bad states). This approach has one substantial drawback—the resulting policies are far from being optimal in the long-term–reward sense [García and Fernández, 2012].

In this part we show the various approaches to safe exploration that have emerged so far. In section 3.2 we review methods published up to year 2014, which we covered in [Pecka and Svoboda, 2014]. Further research and followup work is presented in chapter 5.

We classify the methods by various criteria and suggest suitable use cases for them. To better illustrate some practical details, we use the Absolem UGV described in section 2.2 as a reference agent. It may happen that in these practical details we assume some advantages of UGVs over UAVs (Unmanned Aerial Vehicles), like the ability to stand still without much effort, but it is mostly easy to convert these assumptions to UAVs, too.

3.2 Defining Safety

To examine the problems of safe exploration, it is first needed to define what exactly is the *safety* we want to maintain. Unfortunately, there is no unified definition that would satisfy all use cases; thus, several approaches are found in the literature. An intuitive (but vague) definition could be e.g.: "State-space exploration is considered *safe* if it doesn't lead the agent to unrecoverable and unwanted states." It is worth noticing here that *unwanted* doesn't necessarily mean low-reward. In the following subsections we present the main interpretations of this vague definition.

3.2.1 Safety Through Labeling

The largely most used definition of safety is labeling the states/actions with one of several labels indicating the level of safety in that state/action. What varies from author to author is the number and names of these labels.

To start with, [Hans et al., 2008] have the most granular division of state/action space. Their definitions are as follows (slightly reformulated):

- an (s, a, r, s') tuple (transition) is **fatal** if the reward r is less than a certain threshold (s is the original state, a is an action and s' is the state obtained after executing a in state s, yielding the reward r),
- an action a is **fatal** in state s if there is non-zero probability of leading to a fatal transition,
- state s is called **supercritical** if there exists no policy that would guarantee no fatal transition occurs when the agent starts in state s,
- action a is **supercritical** in state s if it can lead to a supercritical state,
- state s is called **critical** if there is a supercritical or fatal action in that state (and the state itself is not supercritical),
- action a is **critical** in state s if it leads to a critical state (and the action itself is neither supercritical nor fatal in s),
- state s is called **safe** if it is neither critical nor supercritical,
- action a is **safe** in state s if it is neither critical, nor supercritical, nor fatal in state s,
- and finally a policy is **safe** if for all critical states it leads to a safe state in a finite number of non-fatal transitions (and if it only executes safe actions in safe states).

Since we will compare other definitions to this one, it is needed to define one more category. A state s is called **fatal** if it is an undesired or unrecoverable state, e.g. if the robot is considered broken in that state. The fatal transition

can then be redefined as a transition ending in a fatal state. Opposite to the precisely defined terms in the definition of Hans et al., the meaning of words "undesired" and "unrecoverable" here is vague and strongly task-dependent.

Continuing on, [Geibel, 2001] defines only two categories—*fatal* and *goal* states. "Fatal states are terminal states. This means, that the existence of the agent ends when it reaches a fatal state" [Geibel, 2001]. This roughly corresponds to our defined set of **fatal** states. *Goal states* are the rest of final states that correspond to successful termination. Since Geibel only considers terminal states for safety, his *goal* states correspond to a subset of **safe** states. The other categories need not be represented, since they are meaningless for final states.

An extension of Geibel's *fatal* and *goal* states is a division presented by [García and Fernández, 2012]. Their *error* and *non-error* states correspond to *fatal* and *goal* states, but they add another division of the space—the *known* and *unknown* states, where *known* states are those already visited (and *known* have empty intersection with *error*). Then they mention a prerequisite on the MDP that if an action leads to a known *error*/*non-error* state, then its slight modification must also lead to an *error*/*non-error* state (a metric over the state space is required).

In the work of [Ertle et al., 2012], again the two basic regions are considered—they are called *desired* and *hazardous* (corresponding to **safe** and **fatal**). However, due to the used learning technique, one more region emerges—the *undesired* region. It contains the whole *hazardous* region and a "small span" comprising *desired* states, and denotes the set of states where no training (safe) samples are available, because it would be dangerous to acquire those samples. In particular, they say that "The hazards must be 'encircled' by the indications of the undesired approaching so that it becomes clear which area [...] is undesired" [Ertle et al., 2012].

A summary of the labeling-based definitions is shown in Figure 3.2. We examined the apparent imbalance between the number of categories Hans et al. define and the other definitions, and that led us to the following observations.

The first observation is that creating labels for actions or transitions is unnecessary. If we need to talk about the "level of safety" of an action, we can use the worst label out of all possible results of that action (which retains compatibility with definitions of Hans et al.). Moreover, as "it is impossible to completely avoid error states" [Geibel and Wysotzki, 2011], we can ignore the effects of the action which have only small probability (lower than a safety threshold)—we will call such effects the *negligible effects*.

A second remark is that the **fatal** and **supercritical** sets can be merged. In [Hans et al., 2008], we haven't found any situation where distinguishing between **supercritical** and **fatal** would bring any benefit. Specifically, they state that: "Our objective is to never observe supercritical states," which effectively involves avoiding fatal transitions, too. And since we avoid both supercritical and fatal, we can as well avoid their union.

Third, safety of a state does not necessarily depend on the reward for getting to that state. E.g. when the UGV performs a victim detection task

as in [Petříček et al., 2019], going away from the target area may be perfectly safe, but the reward for such action should be small or even negative.

Putting these observations together, we propose a novel definition of safety for stochastic MDPs, which is a simplification of the model of Hans et al. and a generalization of the other models:

- A state is **unsafe** if it means the agent is damaged/destroyed/stuck... or it is highly probable that it will get to such state regardless of further actions taken.
- A state is **critical** if there is a not negligible action leading to an unsafe state from it.
- A state is **safe** if no available action leads to an unsafe state (however, there may be an action leading to a critical state).

To illustrate the definition on a real example, please refer to Figure 3.1. In 3.1(a), the UGV is in a **safe** state, because all actions it can take lead again to safe states (supposing that actions for movement do not move the robot for more than a few centimeters). On the other hand, the robot as depicted in 3.1(b) is in a **critical** state, because going forward would make the robot fall over and break. If the robot executed action "go forward" once more, it would come to an **unsafe** state. Right after executing the action it would still not be broken; however, it would start falling and that is **unsafe**, because it is not equipped to withstand such fall and therefore it is almost sure it will break when it meets the ground.

3.2.2 Safety Through Ergodicity

An MDP is called *ergodic* iff for every state there exists a policy that gets the agent to any other state [Moldovan and Abbeel, 2012]. In other words, every mistake can be remedied in such MDP. Moldovan and Abbeel then define *δ-safe policies* as policies guaranteeing that from any state the agent can get to the starting state with probability at least δ (using a *return policy*, which is different from the δ-safe one). Stated this way, the safety constraint may seem intractable, or at least impractical—it is even proved, that expressing the set of δ-safe policies is NP-hard [Moldovan and Abbeel, 2012]. An approximation of the constraint can be expressed in the terms of two other MDP problems which are easily solved; that still leads to δ-safe policies, but the exploration performance may be suboptimal.

In our view, safety through ergodicity imposes too many constraints on the problems the agent can learn. It sometimes happens that a robot has to learn some task after which it is not able to return to the initial state (e.g. drive down a hill it cannot go upwards; a human operator then carries the robot back to the starting position). But the inability to "return home" in no means indicates the robot is in an unsafe state.

3.2.3 Safety Through Costs

Another definition of safety is to define a cost for taking an action/being in a state and minimize the worst-case cost of the generated policies (up to some failure probability). Such approach is presented in [Heger, 1994].

However, unless a threshold is set, this definition leads only to the *safest possible* policies, which are not necessarily *safe*, nor optimal. Expressing the safety using costs is natural for some RL tasks (e.g. when learning the function of a dynamic controller of an engine, the engine's temperature can be treated as a cost). Unfortunately, not all **unsafe** states can be described using such costs in general. In addition, specifying the right costs may be a difficult task.

3.2.4 Safety as Variance of the Expected Return

An alternative to safety as minimization of a cost (either worst-case or expected) is minimizing both the cost and its variance. This approach is called *expected value-variance criterion* [Heger, 1994] and is used mainly in works prior 2000, e.g. [Coraluppi and Marcus, 1999]. A safe policy by this criterion can be viewed as a policy that minimizes the number of **critical** actions (because fatal transitions are expected to yield much larger costs than safe transitions, increasing the variance significantly).

As stated in [García and Fernández, 2012], the worst-case approach is too restrictive and cautious. The other expected value-variance criteria suffer from the same disadvantages as safety through costs—mainly from the general difficulty to tune up the costs.

3.3 Safe Exploration Approaches

Finally, when the theoretical concepts have been shown and the various safety definitions have been presented, we can focus on the main part of this overview. Our categorization of safe exploration techniques is based on the work of [García and Fernández, 2012]. The basic division is as follows: approaches utilizing the expected return or its variance (section 3.3.1), labeling-based approaches (section 3.3.2) and approaches benefiting from prior knowledge (section 3.3.3).

3.3.1 Optimal Control Approaches

Techniques in this category utilize variations of the *expected value-variance* safety criterion. The most basic one is treating the rewards as costs (when a reward is denoted by r_t, the corresponding cost is denoted by c_t). Standard RL methods can then be used to solve the safe exploration task, as described e.g. in [Coraluppi and Marcus, 1999] for discounted infinite horizon.

The RL objective function

$$\mathbf{J}_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c_t\right] \tag{3.1}$$

is called the *risk-neutral objective*. To make this objective *risk-sensitive*, we specify a *risk factor* α and rewrite the objective as: [Heger, 1994]

$$\mathbf{J}_\pi = \frac{1}{\alpha} \log \mathbb{E}\left[\exp\left(\alpha \sum_{t=0}^{\infty} \gamma^t c_t\right)\right] \simeq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c_t\right] + \frac{\alpha}{2} \operatorname{Var}\left[\sum_{t=0}^{\infty} \gamma^t c_t\right] \tag{3.2}$$

which is also called the *expected value-variance criterion*. Unsafe actions should carry high costs, so minimizing both costs and their variance can be seen as seeking a solution with low expected costs and low probability of entering unsafe states.

This approach is a part of theory using *exponential utility functions*, which is popular in optimal control [Mihatsch and Neuneier, 2002]. To complete this section, the worst-case objective function (also called the *minimax objective*) is defined as

$$\mathbf{J}_\pi = \sup\left[\sum_{t=0}^{\infty} \gamma^t c_t\right]. \tag{3.3}$$

As can be seen, the objective functions containing expectations cannot in fact assure that no unsafe state will be encountered. On the other hand, the minimax objective provides absolute certainty of the safety. However, it may happen that some unsafe states can only be reached with a negligible probability. In such cases, the *α-value* criterion defined by [Heger, 1994] can be used—it only takes into account rewards that can be reached with probability greater than α. In the work of [Mihatsch and Neuneier, 2002], a scheme is presented that allows to "interpolate" between risk-neutral and worst-case behavior by changing a single parameter.

[Delage and Mannor, 2007] take into account the uncertainty of parameters of the MDP. It is often the case that the parameters of the MDP are only estimated from a limited number of samples, causing the parameter uncertainty. They propose a possibility that the agent may "invest" some cost to lower the uncertainty in the parameters (by receiving some observations from other sources than exploration). A whole research area then appears—to decide whether it is more valuable to pay the cost for observations, or to perform exploration by itself. We follow this direction in Part II.

An approximation scheme for dealing with transition probability uncertainty is presented in [Nilim and El Ghaoui, 2005]. It considers a robust MDP problem and provides a worst-case, but also robust policy (with respect to the transition probability uncertainty).

A theory generalizing these approaches can be found in [Schneider, 1996]. The theory states that the optimal control decision is based on three terms—the *deterministic*, *cautionary* and *probing* terms.

> The deterministic term assumes the model is perfect and attempts to control for the best performance. Clearly, this may lead

> to disaster if the model is inaccurate. Adding a cautionary term yields a controller that considers the uncertainty in the model and chooses a control for the best expected performance. Finally, if the system learns while it is operating, there may be some benefit to choosing controls that are suboptimal and/or risky in order to obtain better data for the model and ultimately achieve better long-term performance. The addition of the probing term does this and gives a controller that yields the best long-term performance. [Schneider, 1996]

To conclude this section, we think that these methods are not well suited for safe exploration—the expected value-variance and similar criteria provide no warranties on the actual safety. On the other hand, the worst-case approaches seem to be too strict.

3.3.2 Labeling-Based Approaches

The approaches utilizing some kind of state/action labeling (refer to section 3.2.1 for the various labeling types) usually make use of two basic components—a *safety/risk function* and a *backup policy*. The task of the *safety function* is to estimate the safety of a state or action. In the simplest case, the safety function can just provide the labeling of the given action; or it can return a likelihood that the action is safe; and in the best case, it would answer with a likelihood to be safe plus a variance (certainty) of its answer. The *backup policy* is a policy that is able to lead the agent out of the critical states back to the safe area. It is not obvious how to get such a policy, but the authors show some ways how to get one.

In the work of [Hans et al., 2008], the most granular labeling is used, where fatal transitions are said to be the transitions with reward less than a given threshold. The safety function is learned during the exploration by collecting the so-called *min-reward samples*—this is the minimum reward ever obtained for executing a particular action in a particular state. The backup policy is then told to either exist naturally (e.g. a known safe, but suboptimal controller), or it can also be learned. To learn the backup policy, an RL task with altered Bellman equations is used:

$$\mathbf{Q}^*_{min}(s,a) = \max_{s' \in \mathbf{S}} \min \left[\mathbf{R}(s,a,s'), \max_{a' \in \mathbf{A}} \mathbf{Q}^*_{min}(s',a') \right].$$

A policy derived from the computed $\mathbf{Q}^*_{min}$ function is then taken as the backup policy (as it maximizes the minimum reward obtained, and the fatal transitions are defined by low reward). A policy is *safe*, if it executes only safe actions in safe states and produces non-fatal transitions in critical states. To learn such safe policy, a level-based exploration scheme is suggested (although no proofs why it should be better than any other exploration scheme are given). This scheme is based on the idea that it is better to be always near the known safe space when exploring. All unknown actions from one "level" are explored, and their resulting states are queued to the next "level". For

exploration of unknown actions it is proposed that the action should be considered critical until proved otherwise, so the exploration scheme uses the backup policy after every unknown action execution. A disadvantage of this approach is that the agent needs some kind of "path planning" (or resets) to be able to get to the queued states and continue exploration from them.

The PI-SRL algorithm [García and Fernández, 2012] is a way to safeguard the classical policy iteration algorithm. Since the labels *error*/*non-error* are only for final states, the risk function here is extended by a so called *Case-based memory*, which is in short a constant-sized memory for storing the historical (s, a, $\mathbf{V}(s)$) samples and is able to find the nearest neighbors for a given query (using e.g. Euclidean distance). In addition to the *error* and *non-error* states, they add the definition of *known* and *unknown* states, where *known* states are those that have a neighbor in the case-based memory closer than a threshold. A safe policy is then said to be a policy that always leads to *known non-error* final states. To find such policy, the policy iteration is initialized with the safe backup policy and exploration is done via adding a small amount of Gaussian noise to the actions. This approach is suitable for continuous state– and action-spaces.

Another approach is presented in [Geibel, 2001], where the risk and objective functions are treated separately. So the risk function only classifies the states (again only final states) as either *fatal* or *goal*, and the risk of a policy (risk function) is then computed as the expected risk following the policy (where *fatal* states have risk 1 and *goal* states have risk 0). The task is then said to be to maximize the objective function (e.g. discounted infinite horizon) w.r.t. the condition that the risk of the considered policies is less than a safety threshold. The optimization itself is done using modified Q-learning, and the optimized objective function is a linear combination of the original objective function and the risk function. By changing the weights in the linear combination the algorithm can be controlled to behave more safely or in a more risk-neutral way.

A generalization of Geibel's idea to take the risk and reward functions separately can be found in the work of [Kim et al., 2012]. In this work, the constrained RL task is treated as a *Constrained MDP* and the algorithm *CBEETLE* for solving the Constrained MDPs is shown. The advantage of this work is that it allows for several independent risk (cost) functions and doesn't need to convert them to the same scale. However, the CBEETLE algorithm solves the task only approximately, which means some safety criteria may not be satisfied by the optimal policy.

A similar approach of using constrained MDP to solve the problem can be found in the work of [Moldovan and Abbeel, 2012]. They do, however, use the *ergodicity* condition to tell safe and unsafe states apart. Moreover, this approach is only shown to work for toy examples like the grid world with only several thousands of discrete states, which may not be sufficient for real robotics tasks.

The idea of having several risk functions is further developed by [Ertle et al., 2012]. The agent is told to have several behaviors and a separate safety

function is learned for each behavior. This approach allows for modularity and sharing of the learned safety functions among different types of agents. More details on this work will be provided in the next section, because it belongs to learning with teachers.

An approach slightly different from the previously mentioned in this section is using the methods of reachability analysis to solve safe exploration. [Gillula and Tomlin, 2011] define a set of *keep-out states* (corresponding to **unsafe** in our labeling) and then a set called $Pre(\tau)$ is defined as a set of all states from which it is possible to get to a *keep-out* state in less than τ steps. Reachability analysis is used to compute the $Pre(\tau)$ set. *Safe states* are then all states not in $Pre(\tau)$ for a desired τ. This approach, however, doesn't utilize reinforcement learning, it computes the optimal policy using standard supervised learning methods with one additional constraint—that the system must use safe actions near the $Pre(\tau)$ set. On the other hand, the system is free to use whatever action desired when it is not near $Pre(\tau)$.

As was presented in this section, the labeling-based approaches provide a number of different ways to reach safety in exploration. They are, however, limited in several ways—some of them make use of the (usually hard-to-obtain) transition probability, the others may need to visit the unsafe states in order to learn how to avoid them, or need the state-space to be metric.

3.3.3 Approaches Benefiting from Prior Knowledge

The last large group of safe exploration techniques are the ones benefiting from various kinds of prior knowledge (other than the parameters of the MDP). We consider this group the most promising for safe exploration, because "it is impossible to avoid undesirable situations in high-risk environments without a certain amount of prior knowledge about the task" [García and Fernández, 2012].

The first option how to incorporate prior knowledge into exploration is to initialize the search using the prior knowledge. In fact, several works already mentioned in previous sections use prior knowledge—namely the approaches with a backup policy ([Hans et al., 2008], [García and Fernández, 2012]). Also, García and Fernández suggest that the initial estimate of the value function can be done by providing prior knowledge, which results in much faster convergence (since the agent does no more have to explore really random actions, the estimate of the value function already "leads it" the right way).

Another option how to incorporate prior knowledge is by using *Learning from Demonstration* (LfD) methods. Due to the limited space, we will not give the basics of LfD—a good overview is for example in [Argall et al., 2009]. For our overview, it is sufficient to state that LfD methods can derive a policy from a set of demonstrations provided by a teacher. What is important, is that the teacher does not necessarily have to have the same geometrical and physical properties as the trainee (although it helps the process if possible). It is therefore possible to use LfD to teach a 5-joint arm to play tennis, while

using 3-joint human arm as the source of demonstrations (but the learned policy may be suboptimal; RL should then be used to optimize the policy).

In *Apprenticeship Learning* [Abbeel et al., 2007], the reward function is learned using LfD. The human pilot flies a helicopter at his best, and both system dynamics and the reward function are learned from the demonstrations. It is however apparent that the performance of the agent is no longer objectively optimal, but that it depends on the abilities of the human pilot.

Another way of incorporating prior knowledge into the learning process is to manually select which demonstrations will be provided, as in [Ertle et al., 2012]. In the work it is suggested that more teacher demonstrations should come from the areas near the unsafe set, in order to teach the agent precisely where the border between safe and unsafe is located.

Interleaving autonomous exploration with teacher demonstrations is an interesting combination of the mentioned approaches. As in the previous case, some teacher demonstrations are provided in advance, and then the exploration part starts utilizing the teacher-provided information. After some time, or in states very different from all other known states, the agent requests the teacher to provide more examples [Argall et al., 2009, Chernova and Veloso, 2007]. The idea behind this algorithm is that it is impossible to think out in advance what all demonstrations will the agent need in order to learn the optimal policy.

Prior knowledge can generally be provided in terms of (even partial or stochastic) transition model, reward function estimate or safety function estimate. The safety and optimality of the learning process depends on what guarantees can be provided for the estimates. We present a general-purpose safe exploration algorithm called Constrained Relative Entropy Search (CREPS) in chapter 4.

Finishing this section, the algorithms utilizing prior knowledge seem to be the most promising out of all the presented approaches. They provide both a speedup of the learning process (by discarding the low-reward areas) and a reasonable way to specify the safety conditions (via LfD, interleaving or estimates).

(a) A safe state.

(b) A critical state—if the robot went still forward, it would fall down and probably break.

Figure 3.1: An illustration of safe and critical states.

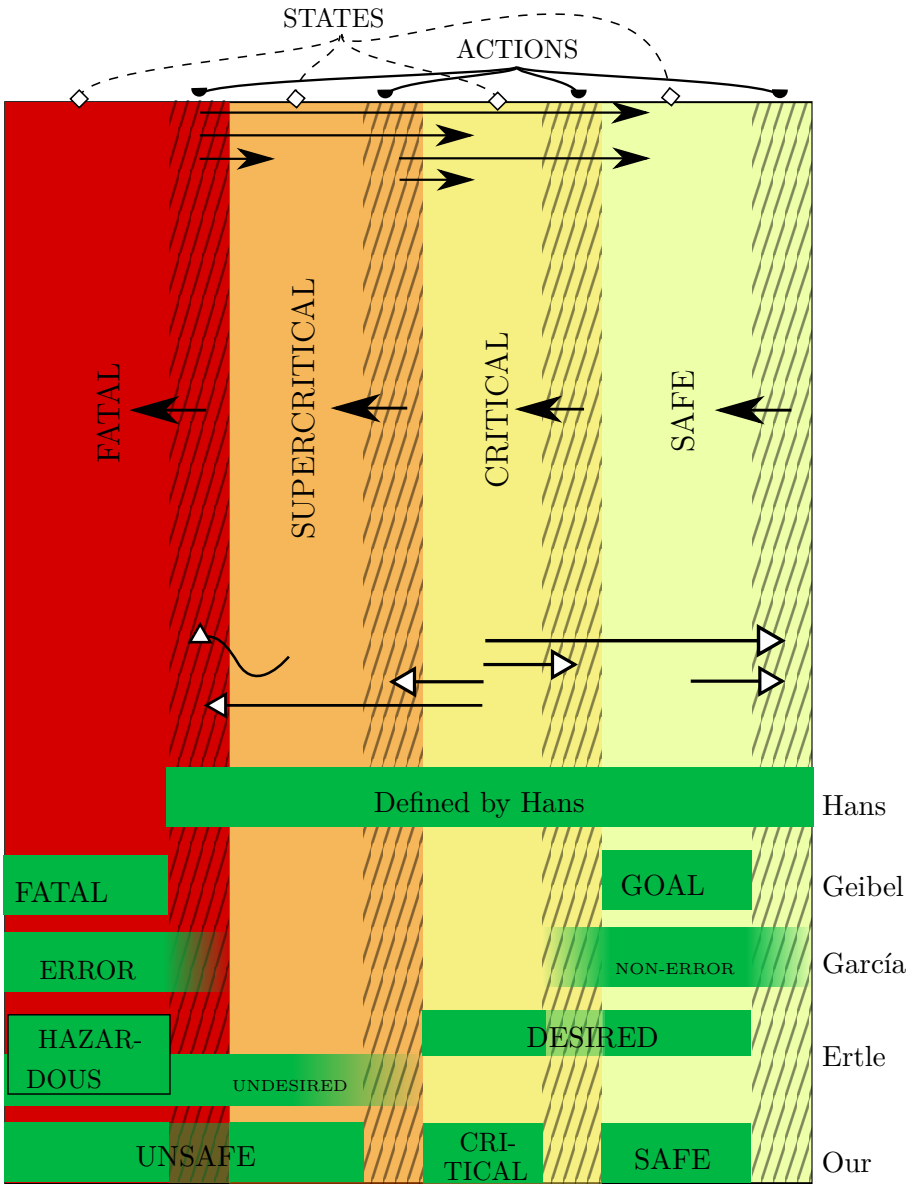


Figure 3.2: *A summary of the definitions of safety.* The basic division is taken from [Hans et al., 2008] and **fatal** states are added. States are drawn with solid background and white-headed arrows (⇾) denote the possible actions in the states. Actions are rendered with striped background and black-headed arrows (➔) end in states where it is possible to end up using the action.

Chapter 4

Constrained REPS

In this section, we describe our algorithm called *CREPS* which aims at solving the safe exploration problem in continuous state- and action-space in case a *cautious simulator* is available.

The algorithm has been published in: Pecka, M., Zimmermann, K., and Svoboda, T. (2016). Autonomous Flipper Control with Safety Constraints. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2889–2894, Daejon, South Korea. IEEE. DOI: 10.1109/IROS.2016.7759447. This section is an adapted version of the publication.

As mentioned in section 2.1.6, the complexity of Policy Gradient Search algorithms doesn't depend on the complexity of the policy or the dynamics model, but only on the number of policy parameters. Such property makes them suitable for learning of controllers for robotic systems for which robust real behavior prediction using the first-principle models is difficult (such as closed form equations describing kinematic and/or dynamic behaviors for rover-terrain interaction, or Navier-Stokes aerodynamic laws). Unfortunately, PGS usually requires many trials which endanger the real system or cause its excessive wear. Therefore, it is usually not used directly on the real system, but on data-driven models. For example, [Kupcsik et al., 2017] demonstrate data-driven PG learning of the ball throwing problem with a robotic arm, and [Tedrake et al., 2009] argue that Policy Gradient learning for aerial maneuvers with an ornithopter may be very efficient. [Transeth et al., 2009] show that for snake-like robots with significant side-slip, no closed form expression of the snake's motion exists, therefore policy learning must resort to approximate simulation.

4.1 Relative Entropy Policy Search Algorithms

Relative Entropy Policy Search (REPS) [Peters et al., 2010] is a model-free gradient policy search algorithm which utilizes Kullback-Leibler divergence (KL divergence) to limit the "distance" of the updated policy from the previous one. This was shown to practically help both stability of the learning and finding policies with higher expected return. Contextual REPS [Kupcsik et al., 2017]

uses a stochastic upper-level policy which generates deterministic lower-level policy samples. The performance of these policies is evaluated by executing them, and is used to estimate the upper-level policy gradient. The Gaussian Process REPS (GPREPS) method by [Kupcsik et al., 2017] adds a Gaussian Process (GP) in the loop, which learns a representation of the system dynamics. The GP is used for better evaluation of the policies without the need for executing more real-world samples. Several PGS methods also take constraints into account: [Uchibe and Doya, 2007] propose constrained policy search for GPOMDPs. However, GPOMDPs belong to early PG algorithms which use the likelihood-ratio trick to compute the gradient of the expected sum of rewards and then update the policy parameters by a user-defined learning rate. [Prashanth, 2014] proposes constrained PG method for Stochastic Shortest Path problem with inequality constraints on Conditional Value-at-Risk (CVaR) as a risk measure. This method does not allow including implicit constraints and cannot be easily extended for general episodic rewards, such as minimum distance of the trajectory from a target position. In this part, we propose a combination of GPREPS with the work of Uchibe and Doya—to extend Contextual REPS with constraints. The proposed method is called *Constrained REPS*, (CREPS). It evaluates the generated policies in a simulator and successively constrains the upper-level policy distribution. This (i) reduces the number of needed samples/iterations and consequently speeds-up the learning process of the model, and (ii) provides a safe policy when used with the real system.

Since it is difficult to provide any guarantees on data-driven models created from real-world samples without any prior knowledge about the underlying physics [Ross and Bagnell, 2012, Akametalu et al., 2014], we replace the GP model from GPREPRS by a *cautious* physics-based simulator that certifies the safety of policies.

4.2 Cautious Simulator

Let us assume a simulator is given that is able to approximately model some real system and assess the safety of executed actions and trajectories (categorize them as **safe** or **unsafe**). It is called *cautious* if the safety assessment can be wrong only in the case when some action is **safe** in the real system and the simulator tells it is **unsafe**. The other kind of error, e.g. misclassifying an **unsafe** action as **safe** is not permitted.

In simple cases, it can be an equation (even implicit), that checks some constraints of arbitrary order. When modeling a complex system, standard software physics and dynamics simulators can be used. The only requirement is that the simulation has to fulfill the cautiousness condition. From the implementation point of view, the *cautiousness* can be a core part of the simulator design or it is achieved by adding noise to the inputs and outputs, and testing more possible values of uncertain parameters (such as track-soil interaction). This way, it should be possible to create cautious simulations of most real-world systems (given the simulator can simulate all the important

interactions and influences).

If there is need for classifying states as **critical**, it is e.g. possible to specify a time limit, and if the simulator doesn't get into an **unsafe** state after trying for the specified time limit, it can categorize the state as **safe**. On the other hand, if some simulation gets into an **unsafe** state, we can mark some number of preceding states as **critical**.

4.3 Contextual REPS

Model-free policy search algorithms usually follow these steps: (i) generate trajectories from the real-world system, (ii) compute a policy maximizing the expected sum of rewards on the so-far-generated trajectories, (iii) use the policy to generate a new real-world trajectory, (iv) repeat from (ii). Contextual REPS [Kupcsik et al., 2017] adds a task-dependent context $\mathbf{s}$ (a changing property of the environment, e.g. the height of an obstacle), from which it extracts a feature vector $\boldsymbol{\phi}(\mathbf{s})$. The policy is hierarchically divided into two parts: *upper-level* policy $q(\boldsymbol{\phi}(\mathbf{s}))$ and *lower-level* policy $\pi(\cdot|\boldsymbol{\omega})$. The *upper-level* policy generates parameter vectors $\boldsymbol{\omega}$ which parametrize the *lower-level* policy. As the *upper-level* policy is usually a distribution over the parameters $\boldsymbol{\omega}$ conditioned by $\boldsymbol{\phi}(\mathbf{s})$, we simplify the notation as $q(\mathbf{s}, \boldsymbol{\omega})$. Samples $\boldsymbol{\omega} \simeq q(\mathbf{s})$ are evaluated on the model (which is called a *rollout*) and the corresponding sums of collected rewards $\mathcal{R}_{\mathbf{s}\boldsymbol{\omega}}^{[i]}$ (called *return*) are recorded. Using this data, Contextual REPS searches for a new *upper-level* policy $p(\mathbf{s}, \boldsymbol{\omega})$, which maximizes the expected sums of rewards while staying close to the so-far-generated trajectories. The distance of trajectories is measured by KL divergence. Bounding the KL divergence between $p(\mathbf{s}, \boldsymbol{\omega})$ and $q(\mathbf{s}, \boldsymbol{\omega})$ as follows:

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \log \frac{p(\mathbf{s}, \boldsymbol{\omega})}{q(\mathbf{s}, \boldsymbol{\omega})} \leq \epsilon,$$

where ϵ specifies the trade-off between exploration and exploitation, was shown to lead to uniform convergence in the whole parameter space [Daniel et al., 2016]. Another constraint imposed on the upper-level distribution in Contextual REPS is to preserve the average distribution of context features:

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \boldsymbol{\phi}(\mathbf{s}) = \hat{\boldsymbol{\phi}},$$

where $\hat{\boldsymbol{\phi}}$ is the average feature value.

4.4 Constrained REPS

We extend Contextual REPS with additional constraints. In particular, for systems which are not inherently safe, or which are prone to wear, we use a cautious physics-based simulator (see section 4.2) to determine a rollout safety $S_{\mathbf{s}\boldsymbol{\omega}}$. The safety equals to 1 if policy $p(\boldsymbol{s}, \boldsymbol{\omega})$ generated a safe trajectory

for context $\mathbf{s}$, and equals to 0 otherwise. We force the upper-level distribution $p(\mathbf{s}, \boldsymbol{\omega})$ to have expected safety bigger than user-defined threshold δ

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega})(1 - S_{\mathbf{s}\boldsymbol{\omega}}) \leq \delta \tag{4.1}$$

Another source of additional constraints is prior knowledge of physical limits such as the maximal joint angles (which are the control actions in our experiment). Violating such constraint is usually not safety-critical, because reaching an impossible pose is often prevented by some low-level motor drivers. However, evaluating many impossible samples naturally slows down the learning process.

Since all these constraints have the same form, we compose a vector $\mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}$ as a collection of evaluated quantities (e.g. safety and mechanical constraints) and vector $\boldsymbol{\delta}$ as a collection of corresponding bounds. Such notation yields the following set of inequalities

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega})(\mathbf{1} - \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}) \leq \boldsymbol{\delta} \tag{4.2}$$

where $\mathbf{1}$ denotes a vector with all-ones of a corresponding dimension.

Constrained REPS searches for an *upper-level* policy distribution $p(\mathbf{s}, \boldsymbol{\omega})$ corresponding to the solution of the following optimization problem:

$$\begin{aligned}
&\max_{p} \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \mathcal{R}_{\mathbf{s}\boldsymbol{\omega}}, \\
&s.t. \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \log \frac{p(\mathbf{s}, \boldsymbol{\omega})}{q(\mathbf{s}, \boldsymbol{\omega})} \leq \epsilon, \\
&\quad \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega})(\mathbf{1} - \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}) \leq \boldsymbol{\delta}, \\
&\quad \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \boldsymbol{\phi}(\mathbf{s}) = \hat{\boldsymbol{\phi}}, \\
&\quad \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) = 1.
\end{aligned} \tag{4.3}$$

We follow the same derivation as proposed in [Kupcsik et al., 2017] and solve the problem by the method of Lagrange multipliers (the detailed derivation is provided in [Kupcsik et al., 2017] and is not given here due to space constraints). By setting the gradient of the corresponding Lagrangian with respect to $p(\mathbf{s}, \boldsymbol{\omega})$ to zero, we obtain the closed form solution

$$p(\mathbf{s}, \boldsymbol{\omega}) \propto q(\mathbf{s}, \boldsymbol{\omega}) \exp\left(\frac{\mathcal{R}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right) \tag{4.4}$$

where $\boldsymbol{\gamma}, \boldsymbol{\theta}$ and η are solutions of the following dual problem

$$\begin{aligned}
&\max_{\eta, \boldsymbol{\gamma}, \boldsymbol{\theta}} \; g(\eta, \boldsymbol{\gamma}, \boldsymbol{\theta}) \\
&s.t. : \boldsymbol{\gamma} > 0, \\
&\qquad \boldsymbol{\eta} > 0,
\end{aligned} \tag{4.5}$$

with

$$g(\eta, \boldsymbol{\gamma}, \boldsymbol{\theta}) = \eta \log \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} q(\mathbf{s}, \boldsymbol{\omega}) \exp\left(\frac{\mathcal{R}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right) + \eta\epsilon + \boldsymbol{\theta}^\top \hat{\boldsymbol{\phi}} + \boldsymbol{\gamma}^\top \boldsymbol{\delta}. \tag{4.6}$$

Using a dataset $\mathcal{D} = [\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}, \mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}, \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}]_{i=1,\dots,N}$ where samples are picked from distribution $q(\mathbf{s}, \boldsymbol{\omega})$, we can rewrite the previous equation as

$$g(\eta, \boldsymbol{\gamma}, \boldsymbol{\theta}; \mathcal{D}) = \eta \log \left[\frac{1}{N} \sum_{i=1}^{N} \exp\left(\frac{\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right)\right] + \eta\epsilon + \boldsymbol{\theta}^\top \hat{\boldsymbol{\phi}} + \boldsymbol{\gamma}^\top \boldsymbol{\delta}, \tag{4.7}$$

Dual problem (4.5) is a convex function with lower bound constraints. We achieved the fastest convergence with the interior point algorithm [Byrd et al., 1999] with supplied gradients:

$$\frac{\partial g}{\partial \eta} = \epsilon + \log \frac{1}{N} \sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}) + \tag{4.8}$$

$$- \frac{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]})(\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}})}{\eta \sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]})},$$

$$\frac{\partial g}{\partial \boldsymbol{\gamma}} = \boldsymbol{\delta} + \frac{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}) \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]})} - \mathbf{1}, \tag{4.9}$$

$$\frac{\partial g}{\partial \boldsymbol{\theta}} = \hat{\boldsymbol{\phi}} - \frac{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}) \boldsymbol{\phi}(\mathbf{s}^{[i]})}{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]})}, \tag{4.10}$$

where $Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}) = \exp\left(\frac{\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma} \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right)$.

Probabilities $p^{[i]}$ of the new upper-level distribution are estimated from the optimal dual variables $\boldsymbol{\theta}, \boldsymbol{\gamma}, \eta$:

$$p^{[i]} \propto \exp\left(\frac{\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right) \tag{4.11}$$

To generate samples from this distribution, we either use weighted maximum likelihood to fit a normal distribution into samples $(\boldsymbol{\omega}^{[i]}, \mathbf{s}^{[i]})$ weighted by probabilities $p^{[i]}$ as suggested in [Kupcsik et al., 2017] or we use importance sampling to generate samples from the non-parametric distribution. Constrained REPS is described in Algorithm 5.

Figure 4.1 shows a toy example. We generate 1000 samples from one-dimensional normal distribution $q(\boldsymbol{\omega})$ with both mean and variance equal

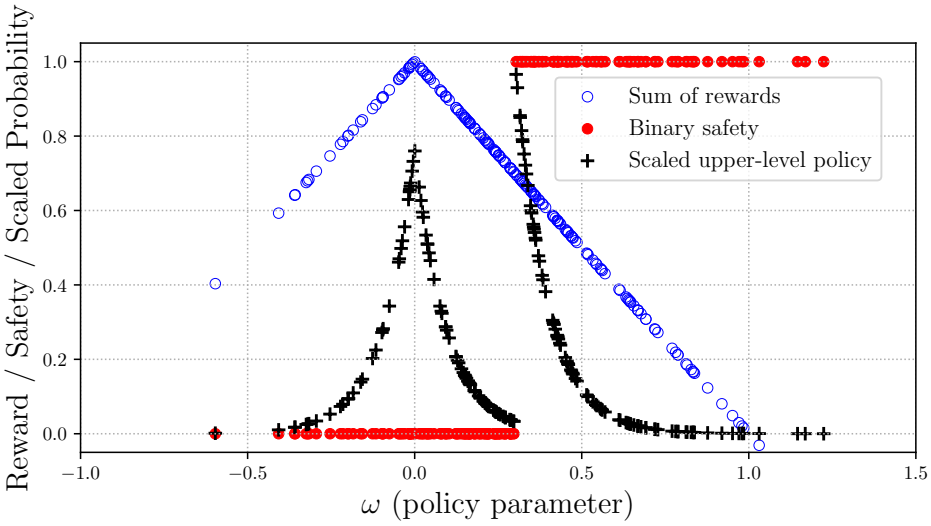


Figure 4.1: Toy example demonstrating solution of problem (4.3). Resulting upper-level probability distribution (4.11) is in black. Please notice the maximum reward is located in an unsafe area. Therefore, CREPS computes a distribution that prefers safe, though suboptimal choices.

to 0.3. We have intentionally chosen the position of rewards maximum into $\boldsymbol{\omega} = 0$, safety equal to one for $\boldsymbol{\omega} > 0.5$ and mean of q into 0.3 to make all constraints active. We set the upper bound on KL-divergence $\epsilon = 0.1$ and the lower bound on safety $\delta = 0.6$. We verify the average safety of samples generated from the distribution given by (4.11) is 0.6064, which is indeed above the required safety bound.

4.5 Experimental Verification

4.5.1 Safe Traversal Task Description

The robot has to learn a *flipper control policy* that would allow it to traverse an obstacle without any prior knowledge about the correct traversal strategy.

This task has been chosen because it very well separates good and bad policies (as well as safe and unsafe). A bad policy is not even able to get the robot on top of the obstacle, and therefore the robot gets stuck in front of it and travels only a short distance (receiving low reward). This task also allows for a wide variety of unsafe policies.

Some results of this experiment are compared to a similar task called *Adaptive Traversability* (AT) presented in [Zimmermann et al., 2014] and improved in [Zimmermann et al., 2015]. The goal of AT is to find a policy to control the flippers so that the robot maximizes a weighted sum of rewards (and minimizes penalties). The training process is a combination of supervised and reinforcement learning and requires a large set of manually annotated data. Since the forward speed is constant in the task, we compare the policies

Algorithm 5 Constrained REPS

1. **Input:** maximal information loss ϵ, vector of constraints $\boldsymbol{\delta}$, context distribution $\mu(\mathbf{s})$, initial upper-level policy $q(\boldsymbol{\omega}|\mathbf{s})$, number of policy updates K, number of samples N.
2. **for** $k = 1, \ldots, K$ **do**
3. **for** $i = 1, \ldots, N$ **do**
4. Observe $\mathbf{s}^{[i]}$ from $\mu(\mathbf{s})$.
5. Generate parameters $\boldsymbol{\omega}^{[i]}$ from $q(\boldsymbol{\omega}|\mathbf{s}^{[i]})$.
6. Using $\boldsymbol{\omega}^{[i]}$ execute lower-level policy on the model and collect $\mathcal{R}_{\mathbf{s}\boldsymbol{\omega}}^{[i]}$, $\mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}^{[i]}$.
7. **end for**
8. Fill in dataset: $\mathcal{D} = [\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}, \mathcal{R}_{\mathbf{s}\boldsymbol{\omega}}^{[i]}, \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}^{[i]}]_{i=1,\ldots,N}$.
9. Optimize dual function: $[\eta, \boldsymbol{\gamma}, \boldsymbol{\theta}] = \text{argmin}_{\eta', \boldsymbol{\gamma}', \boldsymbol{\theta}'}\, g(\eta, \boldsymbol{\gamma}, \boldsymbol{\theta}; \mathcal{D})$.
10. Compute weights $p^{[i]}$ for all samples in $\mathcal{D}$, Eq. (4.11).
11. Update upper-level policy $q(\boldsymbol{\omega}|\mathbf{s})$ with weighted ML (or collect $p_{i=1,\ldots,N}^{[i]}$ for importance sampling).
12. **end for**

using the penalties for high pitch angle and for high acceleration.

We use a similar environment for the *Safe Traversal* (ST) task. The tracked robot starts in front of a standard wooden EUR 1 pallet. The robot is automatically driven forward by a constant speed, and the experiment ends after 30 seconds.

States. States of the ST task are: (i) robot body pitch, and (ii) height of the terrain approximately 20 cm in front of the robot body (read from an octomap built online from laser scans).

Actions. ST policy controls independently the pairs of front and rear flippers using positional control. Therefore, the action space is continuous and 2-dimensional.

Rewards and Safety. In the AT task, safety is not modeled separately, and some safety features are part of the reward. The reward for the AT task is a weighted sum of (i) manually assigned safety penalty, (ii) high pitch/roll angle penalty, (iii) penalty for excessive flipper motion, (iv) robot forward speed reward, and (v) motion roughness penalty measured by accelerometers [Zimmermann et al., 2014].

In the ST task, the reward is simply the distance traveled in 30 seconds over the pallet (the choice of policy influences e.g. track slippage and motor stress, which lower the speed). Safety is modeled explicitly by the cautious simulator, which marks as unsafe all rollouts in which the robot tops over, hits hard on the ground or obstacle (measured as deceleration), or hits objects with delicate parts of its body (e.g. sensors). Please note that this safety definition is related to the robot body, and is independent from the specific task and rewards.

Lower-level Policy. We use a policy that is linear in the states, and that controls front and rear flippers separately. The state vector is 2-dimensional, which yields 3 parameters per action (2 for state variable multipliers and a constant), summing up to 6 policy parameters, $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_6)$, to be learned.

Context. In this experiment, we did not make use of the context—it was always set to zeros. This helped to keep the experiment simple, and was also needed to keep the possibility of comparing ST and AT results. The framework, however, supports the use of context.

4.5.2 Simulator Setup

To run the simulation on a computer, we use the simulator described in Appendix A. The collision model used in the simulator consists of simple-shape collision links (boxes, cylinders) approximating the CAD model of the robot. Therefore, the size of the links should reflect reality very well. Weights, centers of mass, inertias, friction coefficients and other dynamics coefficients are estimated manually. It is important to estimate these parameters precisely enough, so that the simulator can be assumed cautious (which can be easily done using the CAD model). Or, in case of very influential unknown parameters, the simulator has to be run with multiple possible values and the worst-case outcome treated as the simulation result.

The simulator reports unsafe rollout if it encounters any of the unsafe states defined in previous section. However, the rollout is not stopped at that moment (the simulated robot does not break—if it hits hard on the ground, it can still continue). However, if it tops over, it naturally cannot go any further. We utilize this setting to show the safety constraint is not related to low rewards (and also to allow comparing with unconstrained REPS).

4.5.3 Experiment Setup

First, 10 to 30 iterations of the CREPS algorithm are done using only the simulator (until the policy converges). It returns an upper-level policy, from which we can draw lower-level policies that are safe in the simulator and lead to high expected rewards. In each iteration, we test approximately 150 simulated rollouts to estimate the new upper-level policy parameters.

For improvement by real-world samples, the experiment continues with 10 lower-level policies sampled from the optimal upper-level policy and checked in the simulator for safety. If they are safe, they are executed on the real robot. Otherwise, different policies are sampled until the desired number of safe policies is reached.

These real samples are further used to update the upper-level policy in the CREPS algorithm. This way we can correct the policy if the simulator estimated the rewards incorrectly.

4.5.4 Results

We tested the algorithm with several settings to show it consistently converges to high rewards in safe regions. The settings differed in e.g. the initial policy, upper-level policy representation (either a multivariate Gaussian distribution or the Importance Sampling mechanism as described in section 4.4), and the expected safety lower bound (generally 0.8, but in one experiment it was set to 0.0 to simulate unconstrained REPS).

The probability distribution of safe/unsafe policies showed to be very complex during the experiments, and it is far from being Gaussian or uniform. As can be seen in Figure 4.2, most of the time, the mean safety of rollouts is below the desired threshold of 0.8. However, it still tries to reach the threshold. In this case, the Importance Sampling method yields better results, as it better represents the complex distribution.

In the two experiments with unconstrained REPS, comparing Figure 4.2 and Figure 4.3, it is clear the algorithm strived for the highest rewards possible and safety quickly dropped to almost zero (which means the traversal was faster, but the robot hit ground too hard during the rollout). Interestingly, one of the unconstrained experiments reached a level of expected rewards not seen in any of the safety-constrained cases, which suggests that the best policies are unsafe and CREPS correctly avoids these maxima.

Figures 4.2 and 4.3 show that the CREPS algorithm maximizes the rewards in cca the first 10 iterations, and then it holds the good rewards and tries to satisfy the expected safety constraint. As we discussed earlier in this section, since the safe policy distribution is difficult to represent, the expected safety constraint is often broken. However, it does not mean that the robot could be damaged because of this imperfection. It only means that we probably need to sample more lower-level policies until a safe one is found (which is always tested in the simulator before real execution).

We compared the high-pitch and high-acceleration penalties gathered by both an optimal policy for the ST task and also for the AT task. We performed 10 rollouts with each of the methods, and compared the penalties; results are shown in Figure 4.4. The pitch histogram was essentially the same for both tasks, so only the acceleration histograms are shown in Figure 4.4. The figure illustrates that the CREPS policy doesn't generate more dangerous trajectories than the AT policy (which was however trained with a large set of manually annotated data).

Last, we closed the loop improving one of the best policies found in the simulator by real-world reward samples. We executed two CREPS iterations, each with 10 samples. Safety was always checked in the simulator, then the sampled policy was executed, and the real-world reward collected. After two gradient search steps, the expected reward is higher than the best reward achieved in the simulator, as is shown in Table 4.1. It is important to note that the policy search now cannot reuse samples from the simulator, since the reward estimate may be biased by imperfection of the simulator.

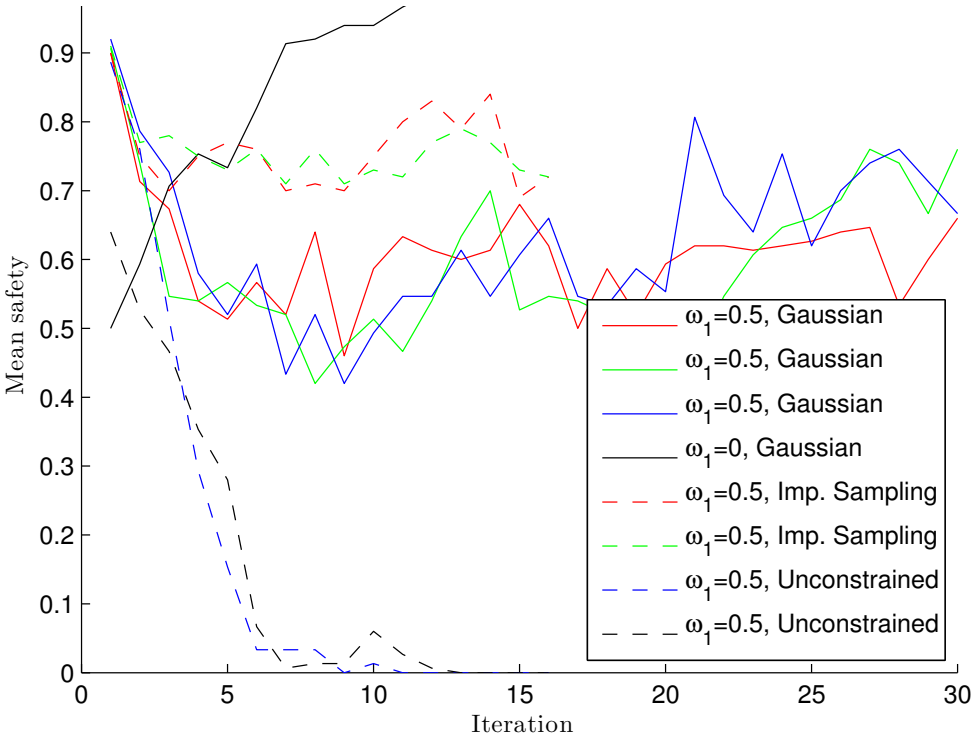


Figure 4.2: Mean safety during rollouts. ω_1 is the initial value of the first element of the policy parameter vector, and it showed to have large influence on safety. Multiple curves with equal description correspond to multiple runs with different values of $\omega_2 \dots \omega_6$ which are omitted from the figure for simplicity. The solid black curve is from a policy initialized close to a local maximum of the safety function. Compare with Figure 4.3 to see that the reward increased very slowly with expected safety higher than the desired threshold of 0.8. The last two policies were not constrained by safety at all (behaving like Contextual REPS), and they quickly found the best rewards lie in the unsafe space. Also note the Importance Sampling experiments tend to achieve higher expected safety.

4.5.5 Conclusion

In a small number of iterations (and with about 2000 simulated trajectories), the robot learned how to safely traverse a previously unknown obstacle, and even the learning process itself was safe—thanks to the cautious simulator. We show that the cautious simulator can be designed in such a way that it does not constrain the possible actions too much and the robot is still able to reach the safe optimal rewards.

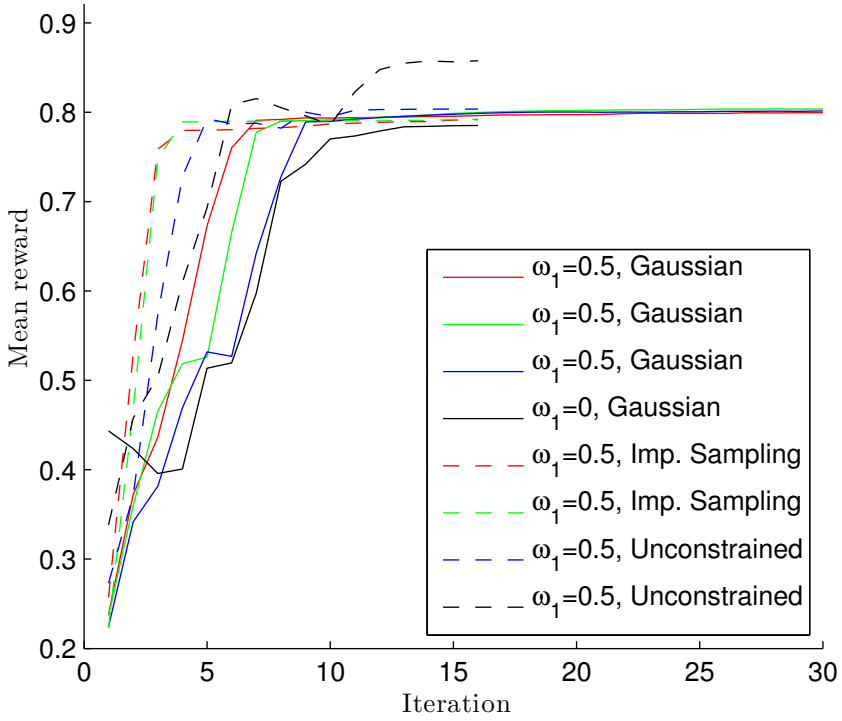


Figure 4.3: Mean reward during rollouts. The last policy not constrained by safety converged to an optimum with the highest expected reward.

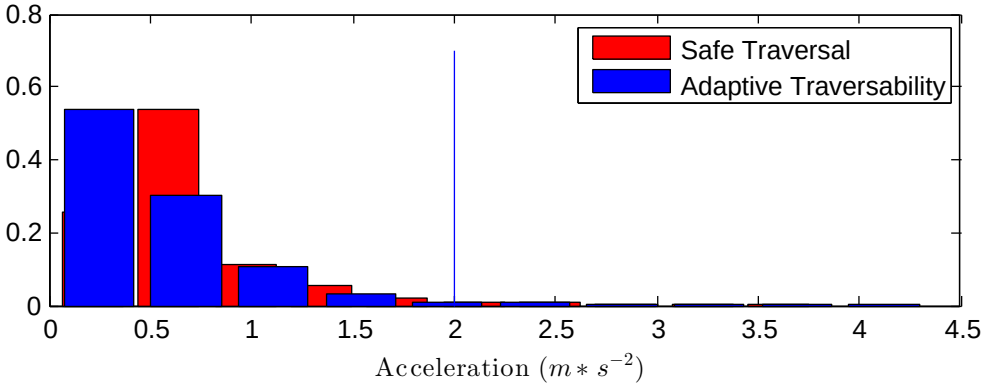


Figure 4.4: Comparison of acceleration during 10 rollouts with AT and with ST (shown as relative histograms). Values over 2 (right to the blue line) are penalized in the AT task.

Iteration	Kind	Policy Converged	Mean Reward
29	Simulated	yes (in sim.)	0.80 ± 0.01
30	Real	no	0.73 ± 0.10
31	Real	yes (in real)	0.85 ± 0.05

Table 4.1: Execution in the real world. The converged simulated policy performance was lower when used in the real world, but after only 2 real-world iterations, the CREPS algorithm converged to the real-world optimum (which is different from the simulated optimum, since the simulator is only approximate). The reader should notice that the simulated optimum had to be close to the real-world optimum, since CREPS doesn't allow large changes of the policy.

Chapter 5

Followup and Related Work

In this section we review relevant literature that was published after our publications [Pecka and Svoboda, 2014] and [Pecka et al., 2016]. An abundance of safe exploration algorithms has been presented. Some followed the categorization from the overview and just improved on existing algorithms and some came with completely new ideas. It is still apparent that the definition of safety is largely domain-dependent, and researchers in one area need completely different safety guarantees and assumptions than researchers in other areas.

Several other overviews of in AI learning safety were published. Almost concurrently with our overview, [García and Fernández, 2015] published another view on safe reinforcement learning, dividing algorithms into two main categories—those modifying the optimization criterion, and those modifying the exploration process. The thesis of [Moldovan, 2016] also reviews many aspects of safe learning on MDPs. Further overviews are given in [Faria, 2018] and [Hernández-Orallo et al., 2019]. [Amodei et al., 2016] provide a structured view on the key aspects that need attention, concentrating on cases where AI could cause harm when badly designed. Being written shortly after the advent of deep networks, they suggest attention should be given to them in the field of safe learning. Four years later, there still are not many works that would connect safety and deep networks. Further, they suggest demonstrations or simulation should be used to speed up learning. As can be seen further in this section, many authors did exactly that. Last, they suggest a benchmarking test suite should be created for safe learning. We can see several such benchmarks appeared since then, such as Safety AI Gym from [Ray et al., 2019] for continuous tasks with large action spaces, Open Physical Environment Benchmark from [Mirzaei et al., 2017] which includes not only the software part, but also 3D printing instructions to create canonical test environments (it however seems to not be actively used or developed anymore), AI safety gridworlds from [Leike et al., 2017], and most recently SafeLife from [Wainwright and Eckersley, 2020] which is an interesting combination of the Game of Life and discrete MDPs.

One of the possible definitions of safety is that it is connected with the reward function, and negative reward areas are considered as unsafe. [Fernandez-Gauna et al., 2015] came with a modular value iteration algorithm that

decomposes the learned task into several modules—one for the optimized behavior, and one for each identified danger (these are called *veto modules*). This modularization speeds up training, and veto modules make sure the final policy does not execute any unsafe actions. [Koert et al., 2016] use an additional reward term to enable obstacle avoidance in trajectories learned by imitation learning. [Hadfield-Menell et al., 2017] note that the reward function does not necessarily be treated as a final design choice. Instead, they propose to treat the given reward function just as an observation of a "true" reward function. [Tangkaratt et al., 2017] present a variant of Contextual REPS suitable for high-dimensional contexts. The reward function is estimated from collected data, and to keep the problem tractable, the context dimensionality is reduced using nuclear norm optimization, which is shown to serve as a convex surrogate on a low-rank matrix constraint. [Paolo et al., 2017] train a control policy for a tracked vehicle, and their definition of safety can be viewed as implicit, or functional—if the vehicle passes an obstacle, it had to do it safely (because unsafe actions would not let it continue). But this is only admissible if learning on a real robot is not intended, or if perfect policy transfer from simulation to reality can be achieved. A surprising approach is presented in [Turner et al., 2019] and [Turner et al., 2020]. They argue that adding completely unrelated terms to the reward function actually robustifies the final policy. Viewing the "original" reward function only as an observation of the *true reward function*, this approach only makes sure that the final policy performs well also for other reward functions than only the specified one—hopefully also for the true function. [Krakovna et al., 2019] define safety as in terms of *unwanted side-effects*. They introduce the *relative reachability* measure which tells how many possible paths will the considered action "close" for future, if it is executed. However, the presented algorithm does not scale well and is only usable in small grid worlds. [Nass et al., 2019] add a risk-related term to the reward function, which affects how safe or risky should the exploration be. Risk is treated as policy variance in a variant of Relative Entropy Policy Search. Different from other reward-hacking methods, [Papini et al., 2019] introduce a framework with guaranteed improvement of the objective function (up to some probability), which effectively makes reward-hacking a guaranteed method for enforcing safety. It comes at the cost of quite limited policy classes which satisfy the requirements of the method. [Vemula et al., 2020] published an algorithm which utilizes an imperfect simulator in a reinforcement learning setting (similar to our idea from chapter 4 or Part III). However, they do not update the simulator output in any way—they just repel the policy search from areas in which the simulator output differs from the observations.

Sometimes the safety function is known in advance and can be expressed as a constraint on policy optimization. Various constrained optimization techniques can then be used, providing various guarantees on the process and its outcome. [Achiam et al., 2017] present an algorithm called *Constrained Policy Optimization*, which is based on theory that gives guarantees on safety of the learning process and the final policy—if all the required quantities

would be known. However, they usually have to be estimated from samples, and that hinders the guarantees. They try to at least recover from bad gradient updates by performing a line search to stay within the current estimate of the safe space, but the guarantees no longer hold absolutely. [Lu et al., 2017] try to identify an unknown linear system while satisfying some known safety constraints. A big help to their algorithm is the requirement to provide a *nominal safe action*, which is an action that would drive the system from any state to a known safe region. [Ge et al., 2019] provide a globally optimal constrained MDP solution in case the safety constraints are linear. The algorithm can work also with other types of constraints, but it needs to linearize them, and thus loses all kinds of guarantees. The algorithm is said to be model-free, but it seems the model is needed to correctly evaluate the safety function. And there are more unclear points in the algorithm description. [Hasanbeig et al., 2019] and [Hasanbeig et al., 2020] generalize the constraint-based reinforcement learning by formulating the problem in Linear Temporal Logic. This way it is possible to specify even complex conditions for both the optimized reward function and the safety constraints. The method is "partly model-free"—it needs to be able to recognize unsafe states in the close vicinity of the agent during learning. The algorithm also uses two kinds of "learners"—an optimistic one, which is unconstrained deep Q-Learning, and a pessimistic one, which builds an approximation of the dynamics model and applies the safety conditions on the actions of the optimistic learner's Q-function. The method seems to be still too sample-inefficient to be used on more complex real systems. An extension to modular deep reinforcement learning is shown in [Zun Yuan et al., 2019], which is however still too sample-inefficient to do experiments with real robots.

A combination of the previous formulations is the work of [Dimitrova et al., 2016], which allows setting both a maximum cost threshold and maximum probability of getting to an error state in a discrete uncertain MDP. With partial knowledge of the transition probability (which is defined by a limited number of uncertain parameters) they augment the original MDP in such a way that the new MDP represents only the safe paths.

Learning safe optimal policies of partially known MDPs requires a different exploration scheme than in normal unconstrained MDPs. Several works tackle this problem by searching policies which do not only use the best possible safe action, but allow for a whole range of safe actions to be executed. This usually helps exploration. [Junges et al., 2016] call such policies *permissive schedulers*. The dangerous environment is modeled as a Markov Chain (to allow for dynamic environments), and the permissive schedulers are used to play a game against the environment. As [Pazis and Lagoudakis, 2011] note, special care is needed in case the number of possible actions is too high when using Q-Learning. Even the sole act of selecting the best action may become computationally expensive.

Other works try to convert the safe learning problem to standard reinforcement learning by constructing some safeguards that watch actions of

the policy and override it in case it could execute an unsafe action. [Alshiekh et al., 2018] prove that if this safeguard (called *shield* in their framework) is *minimally interfering*, the learning process should converge to the safe optimum. However, computation of the shields requires prior knowledge of both reward and safety functions. In [Jansen et al., 2019], the authors extend the framework to the setting of a multiplayer game against stochastic adversaries. [Wabersich and Zeilinger, 2018] construct *Model-Predictive Safety Certification Schemes*, which are similar to shields, but in context of Model-Predictive Control of linear systems. They say these safeguards can be added to any safety-ignorant algorithm to make it explore the whole safely reachable space. [Mannucci et al., 2018] introduce the safe learning algorithm SHERPA, which utilizes a risk-percieving function. Whenever it tells the intended action might be unsafe, a backup policy is used to get to safer states. The backup policy is found automatically when some assumption about the transition function hold. The method of [Fisac et al., 2019] can be used if the transition model is known up to some bounded stochastic disturbances. They use a Gaussian Process to model these disturbances, and with its help they propagate the safe states boundary. And a safe backup controller is constructed which takes over control only on the boundary of the safe region. The algorithm also constantly verifies whether the observed dynamics complies to what the GP model predicts, and in case some larger disturbances are encountered, it is able to quickly react and retract to safer regions, until it gets a better model of the new disturbance. [Rathi et al., 2020] propose a combination of Q-Learning and Model-Predictive Control. The algorithm estimates some "critical" states, in which MPC is run instead of the best action according to Q-function. To speed up learning, it also tries to remember the MPC-generated actions in the Q-function. This method however requires full knowledge the MDP including transition probabilities and safety function.

As reinforcement learning is much easier to solve with a known reward or safety function, many authors decide to learn it in case it is inaccessible. This is usually "paid" by having to accept some regularity or smoothness assumptions about the function. A popular function approximator for this case are Gaussian Processes, which can not only learn the estimated function values, but can also estimate uncertainty of the predictions. This uncertainty is usually used as a measure of safety, or a safeguard telling the probability of ending up in unsafe areas. [Sui et al., 2015] use the Gaussian Process to estimate reward, and use its uncertainty to not only prevent exploration of low-reward zones, but also to guide the exploration to areas with high uncertainty. With a few more assumptions, they prove an ϵ-optimality bound of the found solution with high probability of not entering the unsafe states.

Connecting standard reinforcement learning approaches with optimal control brings interesting and deep insights in safe learning. [Turchetta et al., 2016] started by giving up on reward-optimal exploration, and substituted the reward function with the unknown safety function, again represented as a Gaussian Process and with some strong assumptions on the function

properties. This work concentrated at proving that there is a high probability of exploring all reachable safe states without visiting any unsafe state. [Berkenkamp et al., 2016] look at a very similar task from a different viewpoint, and treat discovering safe areas as finding the maximal region of attraction of the unknown dynamics system. In following work, [Berkenkamp et al., 2017] split the unknown dynamics of the system to a known part and a part which estimates the unmodeled dynamics, and is represented with a Gaussian Process. Reachability theory is then applied to find policies that are safe and optimal. However, the used framework limits both the usable policy classes and safety formulations by many requirements. The control-theory–based safe learning is summed up in the thesis of [Berkenkamp, 2019].

Some methods suggest that a teacher can be used. This teacher is expected to have more complex task knowledge, maybe even know more than what the used state representation can cover. [Martinez et al., 2015] solve a path-planning task with a stochastic, but known transition function and an unkown (but learnable) safety function. Unsafe states are defined as "dead ends" where the planner gets stuck, so they are not actually fatal. The algorithm tries to find planning *excuses* to identify the decisions which were "responsible" for ending in the unsafe state. If the planner finds out that all unexplored actions might lead to unsafe states, it asks the teacher, if the risk they pose is acceptable or not. [Kahn et al., 2017b] present the PLATO algorithm which is an efficient policy search method. It requires that a teacher is available during training (in this case an MPC algorithm with access to perfect sensing). The teacher is used to guide learning of the control policy. If a safe teacher is used, the learning process itself will also be safe. A teacher-friendly method called Parenting is presented in [Frye and Feige, 2019]. There is a semi-autonomous Direct Policy Search running on the agent, and whenever it gets into a not well known state, it requests teacher input. To make it easier for the teacher, it only chooses two actions and asks the teacher which one seems better. The algorithm also has the ability to record "videos", which it can replay to the teacher later and ask him or her which behavior was better.

Deep neural networks also started to be used in the domain of safe exploration. Their hard-to-express function approximation properties usually do not allow to provide any guarantees on the safety of the learning process or the resulting policy, however in some scenarios, being safe *often* can be sufficient. [Kahn et al., 2017a] train a deep obstacle predictor used when controlling ground or aerial vehicles. The predictor is able to provide both mean and variance of the approximated function. The variance is used as a regularizer in exploration—hitting obstacles in uncertain areas is penalized more than in already known areas. This effectively forces the algorithm to slowly expand the known space, where it can act aggressively, while staying cautious in unknown areas. [Held et al., 2017] train manipulation tasks on a PR2 robot in a simulator. They want to transfer the learned policy to the real robot. Some small amount of unsafe actions (e.g. collisions) is allowed during the adaptation, but they argue that with a lower torque limit the

damage the unsafe action would cause will be lower. They thus present an algorithm that starts with the policy learned in simulator and gradually lifts the torque limit as the policy gets better in reality, with the aim to keep expected damage of the policy below some threshold. They exploit properties of the particular learning algorithm (TRPO) to provide guarantees on the expected damage. These guarantees can, however, not be met, as the quantities they work with need to be estimated from samples. [Huang et al., 2019] use reward-hacking to add penalties for hard impacts in a gentle manipulation task. Surprisingly, they discovered that this modification of reward function yields policies which avoid contact at all. So some more reward hacking is needed to actually end up with a policy that achieves gentle manipulation. By executing tens of thousands of trajectories with the real robot, they were also able to learn the deep neural network policy in reality. But it did not leverage any knowledge collected during the simulated experiments. [Fan and Li, 2019] present one of the few deep RL algorithms that try to treat safety as constraints. Their algorithm works in case the safety function can be observed up to Gaussian noise. They use a Gaussian Process to estimate the safety function during learning, and use reward-hacking to repel the policy from unsafe areas and to attract it to areas with high covariance of the GP.

Further research has brought several improvements to REPS-based algorithms. [Abdolmaleki et al., 2015] present a model-based REPS variant which efficiently improves convergence speed of REPS algorithms by optimizing a quadratic surrogate of the objective function. We did not observe problems with premature convergence, so this improvement was not needed. A generalization of KL-divergence called *f-divergence* was presented in [Belousov and Peters, 2017]. This formulation allows to "interpolate" between various behaviors of policy search algorithms by changing a parameter. REPS or TRPO are shown to be just special cases of *f-Divergence Constrained Policy Improvement*. [Frans et al., 2018] developed a meta-learning algorithm which also uses two-level policies. However, they not only suggest the upper-level policy should act on some smaller state space—they also suggest the timescale it operates on should be coarser. Another view on the multi-level policies is that the upper level does not generate context, but latent states of the lower level policies, as is done in e.g. [Haarnoja et al., 2018]. This way, construction of a whole hierarchy of policies is simplified. [Liu et al., 2019] have shown that initializing REPS with a non-unit covariance can help exploration if some symmetries in the state-action space can be identified. [Klink et al., 2019] connect Contextual REPS with curriculum learning. They suggest to start with high value of the context KL-divergence bound to let the agent learn simpler (but maybe unrelated) tasks, and eventually tighten the bound to force it to learn tasks closer to the real context distribution.

[Schulman et al., 2017] point to an interesting property of RL algorithms—Q-learning can be shown to perform the same type of updates as policy search algorithms under some mild constraints. [Aslund et al., 2018] examine the case of *infected MDPs*, which are MDPs in the beginning, but become POMDPs at some time during execution. In other words, environment observations

become imprecise, but the agent does not know when that happened. They propose an algorithm that can act mostly safely in such a setting. [Mason, 2018] suggests "shrinking" the original MDP to a much smaller Abstract MDP, which only captures the "important" states and transitions between them. If this AMDP is constructed accurately, a safe (or abstract) policy can be computed for this smaller MDP much easier. However, it the conversion to AMDP has some imperfections, no safety can be guaranteed. The learned policy can only be considered safe if the learning process fully converged, not in the meantime. [den Hengst et al., 2020] make a weak link between safe learning and personalized learning (which is a softer variant of safe learning).

Chapter 6

Conclusion

In this part, we have first summarized many recent approaches on defining *safety* in the framework of optimal control and reinforcement learning. We have also proposed a definition of safety which divides the state space to *safe*, *critical* and *unsafe* states. We have shown that all other labeling-based safety definitions are covered by our definition.

In section 3.3, many safe exploration methods are categorized into three basic groups—algorithms from optimal control theory, reinforcement learning algorithms based on state labeling, and algorithms utilizing extra prior knowledge. We have shortly summarized the advantages and disadvantages of the particular approaches. An important observation is that some safe exploration algorithms need to visit unsafe states to correctly classify them later, which discards them from usage scenarios where the unsafe states are really fatal.

In chapter 4, we extend an existing Gradient Policy Search algorithm (Contextual REPS) [Kupcsik et al., 2017] by adding implicit constraints that help keeping the gradient in a promising direction. We call the algorithm Constrained REPS (CREPS). One of the additional constraints includes robot (system) safety which is determined by a cautious simulator.

As is shown further in this thesis, transferring the policies learned in simulator to the real robot is much more complex topic. The domain transfer method we used in CREPS depended on the fact that we had 1:1 instances of the environment in the simulator and reality (a standardized pallet), and on the assumption that localization is precise. These are very restrictive requirements which we try to alleviate in the following parts.

The review of literature following our research confirms that there are still many interpretations of what *safety* actually means. More powerful algorithms with less strict requirements were published. Some of them are based on similar ideas as CREPS, adding more guarantees, cases for which specially efficient implementations can be found, or providing more suitable exploration strategies. Other types of algorithms depend on control and reachability theory, and if all relevant functions can be proved to satisfy special smoothness conditions, these approaches are powerful both in performance and safety guarantees. The most promising novel method is—in our point of view—Cautious RL by [Hasanbeig et al., 2020]. It only requires local estimates of

safety of the explored actions and provides a rich language for specifying not only the constraints, but also the task. However, it is still the case that transferring the safe learning algorithms from simulators to reality either loses guarantees, requires special conditions, or requires numerous samples to be gathered by the real robot.

Part II

Dealing with Incomplete Measurements

In this part, we present our algorithm for incorporating data from imperfect sensors into reinforcement learning. First, we shortly overview the problem and sum up the main contributions. Further, we include a verbatim copy of our publication:

- Pecka, M., Zimmermann, K., Reinstein, M., and Svoboda, T. (2017). Controlling Robot Morphology From Incomplete Measurements. *IEEE Transactions on Industrial Electronics*, 64(2):1773–1782. DOI: 10.1109/TIE.2016.2580125

Last, we review followup and relevant literature that has been published since this paper was accepted.

Chapter 7

Problem Statement

Autonomous driving in previously unknown complicated terrain brings many dangers to robotic platforms. If a high level of autonomy is required, the control algorithm has to have a way of telling if it has enough data to decide whether (or how) a part of the terrain can be traversed.

For example, the Absolem tracked robot uses a lidar to capture the shape of the environment. And there are many reasons why some distance measurements may be missing: geometric constraints (occlusions), laser reflectivity problems, dust, water reflections or even sensor failure. As we cannot pose any assumptions on terrain smoothness, holes in the estimated floor can be dangerous. But not all of them—e.g. a 10 cm hole is not a problem because the tracks will easily overcome it. However, if there were a 30 cm long hole, the robot could fall down into it and break irreversibly.

The control algorithm has to be able to differentiate such situations. If the robot had no other sensors available, the only safe option would be to not go over the risky part of the terrain. Fortunately, there are many ways how a missing terrain height measurement can be obtained, other than lidars. On the Absolem platform, we attached a robotic arm which could measure terrain height by touch similarly to how sight-impaired people do. Other alternative sensors were tried, too, including touch sensors on the edges of the flippers [Šalanský et al., 2016], an IR stereo depth camera, or even repositioning the robot itself so that it gets a different viewing angle.

However, most of the alternative height sensors share a common trait—getting the measurement takes a relatively long time, which makes the alternative sensor "expensive" to use. Thus, in our work [Pecka et al., 2017a], we sought for an efficient algorithm that could distill the most information from the current set of measurements, and suggest an optimal sequence of actions in case more information were needed.

Chapter 8

Summary of Contributions

The work captures several aspects of the above-mentioned complex problem: first, we show a definition of an approximate *safety measure* that can efficiently distinguish the situations where more data is needed. Next, we examine several representations that allow for more efficient evaluation of the control policy and compare them with previous approaches. And finally, we show how to find an optimal policy for capturing as small set of additional data as possible. Validity of all the proposed methods is certified by a large-scale real-world experiment and a set of simulation-based experiments.

Chapter 9

Paper: Controlling Robot Morphology from Incomplete Measurements

This paper has been accepted and published in *IEEE Transactions on Industrial Electronics*.

IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS

Controlling Robot Morphology from Incomplete Measurements

Martin Pecka, Karel Zimmermann, *Member, IEEE,* Michal Reinstein, *Member, IEEE,*
Tomas Svoboda, *Member, IEEE,*

***Abstract*—Mobile robots with complex morphology are essential for traversing rough terrains in Urban Search & Rescue missions (USAR). Since teleoperation of the complex morphology causes high cognitive load of the operator, the morphology is controlled autonomously. The autonomous control measures the robot state and surrounding terrain which is usually only partially observable, and thus the data are often incomplete. We marginalize the control over the missing measurements and evaluate an explicit safety condition. If the safety condition is violated, tactile terrain exploration by the body-mounted robotic arm gathers the missing data.**

***Index Terms*—Adaptive control, Intelligent robots, Learning systems**

I. Introduction

SINCE exploration of unknown disaster areas during *Urban Search & Rescue* missions (USAR) is often dangerous, teleoperated robotic platforms are usually used as a suitable replacement for human rescuers. Motivation to our research comes from field experiments with a tracked mobile robot with four articulated subtracks (flippers, see Fig. 1). The robot morphology allows to traverse complex terrain. A high number of articulated parts brings, however, more degrees of freedom to be controlled. Manual control of all available degrees of freedom leads to undesired cognitive load of the operator, whose attention should be rather focused on reaching the higher-level USAR goals. To reduce the cognitive load of the operator, the autonomy of the platform has to be increased; however, it still has to fall within the bounds accepted by the operators—a compromise known as *accepted autonomy* has to be reached [1].

In [2], a Reinforcement-Learning–based *autonomous control* (AC) of robot morphology (configuration of flippers) is proposed. Its goal is to allow smooth and safe traversal of complex and previously unknown terrain while letting the operator specify the desired speed vector. The traversing task is called *Adaptive Traversal* (AT). Natural and disaster environments (such as forests or collapsed buildings) yield many challenges that include incomplete or incorrect data due to reflective surfaces such as water, occluded view, presence of smoke, and deformable terrain such as deep snow or piles of rubble. Since simple interpolation of the missing terrain profile has proved to be insufficient, we presented an improved AC algorithm that better handles incomplete sensory data (using marginalization) [3].

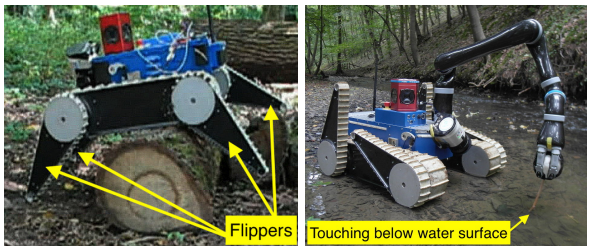


Fig. 1. **Left:** Controlling robot morphology (flippers) allows for traversing obstacles. **Right:** Robotic arm inspects terrain below water surface compensating thus incomplete lidar measurement.

In this work, we extend and improve the AC pipeline introduced in our previously published work [2], [3] (see Fig. 2 for an overview). **The novel contributions include: (i)** introducing a safety measure which allows to invoke tactile exploration of non-visible terrain if needed; **(ii)** several strategies for the tactile exploration with a body-mounted robotic arm; **(iii)** two Q-function representations which allow easier marginalization and achieve comparable (or better) results; **(iv)** and finally, an extensive experimental evaluation of the Autonomous Control. The real-world experiments cover more than 115 minutes of robot time during which the robot traveled 775 meters over rough terrain obstacles.

II. Related work

Many approaches focus on optimal robot motion control in environments with a known map, leading rather to the research field of trajectory planning [4], [5], [6]. Contrary to planning, AC is useful in previously unknown environments and hence can provide crucial support to the actual procedure of map creation. We rather perceive AC as an independent complement to trajectory planning and not as its substitution.

Many authors [7], [4], [8] estimate terrain traversability only from exteroceptive measurements (e.g. laser scans) and plan the (flipper) motion in advance. In our experience, when

Manuscript received November 30, 2015; revised April 7, 2016 and May 10, 2016; accepted May 11, 2016. The research leading to these results has received funding from the European Union under grant agreement FP7-ICT-609763 TRADR; from the Czech Science Foundation under Project GA14-13876S, and by the Grant Agency of the CTU Prague under Project SGS15/081/OHK3/1T/13.

All authors are with the Dept. of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech republic. K. Zimmermann is the corresponding author (phone: +420-22435-5733, email: zimmerk@fel.cvut.cz).

M. Pecka and T. Svoboda are partly with the Czech Institute of Cybernetics Robotics and Informatics, Czech Technical University in Prague, Czech republic.

the robot is teleoperated, it is often impossible to plan the flipper trajectory in advance from the exteroceptive measurements only. The reasons are three-fold: (i) it is not known in advance, which way is the operator going to lead the robot, (ii) the environment is usually only partially observable, (iii) analytic modeling of Robot–Terrain Interaction (RTI) in a real environment is very challenging because the robot can slip or the terrain may deform. Ho et al. [9] directly predict the terrain deformation only from exteroceptive measurements to estimate traversability. They do not provide any alternative solution when exteroceptive measurements are missing. Abbeel et al. [10] use a different approach—they use only proprioceptive measurements for helicopter control, which often works well for aerial vehicles (unless obstacle avoidance is required). We propose that reactive control based on all available measurements is needed for ground vehicles (where obstacle avoidance or robot–ground interaction is essential).

An ample amount of work [11], [12], [13] has been devoted to the recognition of traversal-related manually defined classes (e.g. surface type, expected power consumption or slippage coefficient). However, such classes are often weakly connected to the way the robot can actually interact with the terrain. Few papers describe the estimation of RTI directly. For example, Kim et al. [14] estimate whether the terrain is traversable or not, and Ojeda et al. [15] estimate power consumption on different terrain types. In literature, the RTI properties are usually specified explicitly [15], [16], [14] or implicitly (e.g. state estimation correction coefficient [17], [18]).

Since RTI properties do not directly determine the optimal reactive control, their estimation can be completely avoided. Zhong et al. [19] present a trajectory tracking approach, in which they control a hexapodal robot and utilize force sensors in the legs to detect unexpected obstacles and walk over them. The algorithm tries to minimize the trajectory error caused by obstacles, so that the underlying controller does not need to take them into account. We proposed a different algorithm [2] that explicitly takes the terrain into account (which should yield better results than trying to hide the terrain from the controller). The algorithm is based on Reinforcement Learning, which has been successfully used e.g. in learning propeller control for acrobatic tricks with an RC helicopter [10], [20]. Since it is possible to model the helicopter-air interactions quite plausibly, an RTI model can be used to speed up the learning. In case of ground vehicles, analytical modeling of RTI is very difficult. Therefore, we rather focus on a model-free RL technique called Q-learning (used e.g. to find optimal control in [21]). In Q-learning, state is mapped to optimal actions by taking "argmax" of the so-called Q function (the sum of discounted rewards). In our case, the state space has high dimension (some dimensions with continuous domain), and therefore the Q function cannot be trained for all state–action pairs. Thus, it is modeled either by Regression Forests (RF) or by Gaussian Processes (GP). Regression Forests are known to provide good performance when a huge training set is available [22], with learning complexity linear in the number of training samples. Gaussian Processes present an efficient solution in the context of Reinforcement Learning for control [23].

To deal with incomplete data, the Q function values have to be marginalized over missing features. Such marginalization is often tackled by sampling [24], [25] or EM algorithm [26]. Especially for GPs with Squared Exponential kernel, the Moment Matching marginalization method was proposed by Deisenroth et al. [23]. Marginalization by Gibbs sampling was evaluated for GPs and piecewise constant functions in [3].

We are not aware of any real mobile platform which would use a robot arm as an active sensor for inspecting unknown terrain. Most of the efforts in active inference are directed towards active classification [27], [28], [29] or active 3D reconstruction. Doumanoglou et al. [27] use two robotic arms for folding an unknown piece of cloth whose type is recognized from RGBD data (Kinect). One view is usually insufficient, therefore the cloth needs to be turned around to generate an alternative view. The turning action is implicitly learned with Decision Forests. Bjorkman et al. [28] also recognize objects from RGB-D data. In contrast to [27], Bjorkman et al. use the robotic arm as an active sensor, to touch the self-occluded part of the object in order to reconstruct the invisible 3D shape. While all these classification approaches actively evaluate features in order to discriminate the true (*single*) object class from other possible classes as fast as possible, the Q-learning–based inference presented here evaluates the features in order to find some of the (*multiple*) suitable flipper configurations that allow for a safe and efficient traversal.

III. Overview

Q-learning: The proposed AT solution is adapted from the RL technique called Q-learning (described first to emphasize the differences). The first step in the learning process is driving manually the robot over obstacles to collect a dataset. The state $\mathbf{x}$ (e.g. body pitch angle or terrain shape; see Section IV) is sampled at regular time intervals $t = 0, 1, \ldots, T$. At each time instant t, the operator chooses an action c^t (e.g. the desired flipper positions) that allows to go over the obstacle. After the dataset is collected, each state-action pair $(c^t, \mathbf{x}^t)$ is assigned a reward r^t reflecting suitability of choosing the action in the given state.

Then the iterative Q-learning process starts, which estimates the q^t-values that represent the *sum of discounted rewards* the robot can gather by starting in state $\mathbf{x^t}$, executing action c^t, and always taking the action leading to maximum q from the following state onwards [30]. The q^t and Q values are computed using the recurrent Q-learning formulas [31]:

$$q_i^t := q_{i-1}^t + \alpha\Big[r^t + \gamma \max_{c'} Q_{i-1}(c', \mathbf{x^{t+1}}) - Q_{i-1}(c^t, \mathbf{x^t})\Big] \quad (1)$$

$$Q_i(c, \mathbf{x}) := \mathrm{mean}(q_i^t \mid c^t = c \wedge \mathbf{x^t} = \mathbf{x}) := \mathrm{mean}(q_i(c, \mathbf{x})) \quad (2)$$

where $q_1^t := r^t$, $\alpha \in [0, 1]$ is the learning rate and $\gamma \in [0, 1]$ is the discount factor. From the computation above, it follows that $Q_i(c, \mathbf{x})$ is an unbiased estimator of $E[q_i(c, \mathbf{x})]$.

When the Q-learning is done, we denote $Q = Q_i$ and $q^t = q_i^t$, and the optimal action can be computed as:

$$c^*(\mathbf{x}) \quad = \quad \operatorname*{argmax}_c Q(c, \mathbf{x}) \quad (3)$$

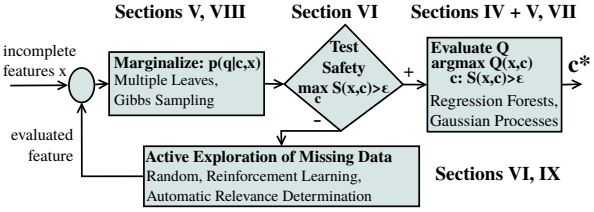


Fig. 2. **Principle overview:** individual blocks in this scheme correspond to Sections IV-VI.

QPDF: In this paper, we generalize the standard Q-learning to an algorithm that learns a distribution called QPDF instead of the Q function. For the QPDF (denoted as $p(q|c,\mathbf{x})$) it holds that

$$Q(c,\mathbf{x}) = E[q(c,\mathbf{x})] = \int q \cdot p(q|c,\mathbf{x})\, dq$$

There are two reasons for modeling the full QPDF: (i) measuring the safety of flipper configurations and (ii) marginalization when only incomplete measurements of $\mathbf{x}$ are available. In Section V, two QPDF models are presented: (i) Regression Forests and (ii) Uncertain Gaussian Processes.

Given the QPDF and full feature vector $\mathbf{x}$, the optimal action $c^*(\mathbf{x})$ is:

$$\begin{aligned} c^*(\mathbf{x}) &= \operatorname*{argmax}_c Q(c,\mathbf{x}) = \operatorname*{argmax}_c E[q(c,\mathbf{x})] = \\ &= \operatorname*{argmax}_c \int q \cdot p(q|c,\mathbf{x})\, dq \end{aligned} \quad (4)$$

Missing Data: While proprioceptive data are usually fully available, the exteroceptive data are often incomplete. This occurs in case of reflective surfaces such as water or in presence of smoke. We denote the missing parts of measurements as $\overline{\mathbf{x}}$, and the available measurements as $\widetilde{\mathbf{x}}$, i.e. $\mathbf{x} = [\overline{\mathbf{x}}, \widetilde{\mathbf{x}}]$. In the case that $\overline{\mathbf{x}}$ is not empty, $p(q|c,\mathbf{x})$ is marginalized over the missing data $\overline{\mathbf{x}}$ to estimate $p(q|c,\widetilde{\mathbf{x}})$. The marginalization processes for different QPDF models are described in Section V. Given the marginalized distribution $p(q|c,\widetilde{\mathbf{x}})$ and measurement $\widetilde{\mathbf{x}}$, the optimal action c^* is estimated by a small modification of Equation 4:

$$c^*(\widetilde{\mathbf{x}}) = \operatorname*{argmax}_c \int q \cdot p(q|c,\widetilde{\mathbf{x}})\, dq. \quad (5)$$

Any state-action pair yielding a negative q-value is interpreted as unsafe considering our definition of the reward function[1]. Therefore, the probability that the q-value is positive (safe) can be computed, and only sufficiently safe state-action pairs are to be considered further. The general trend is that the more features are missing, the higher is the scatter of q-values. Hence, we define the *safety measure*

$$S(c,\widetilde{\mathbf{x}}) = \int_0^\infty p(q \mid c,\widetilde{\mathbf{x}})\, dq, \quad (6)$$

[1]This assumes the user-denoted penalty for dangerous states to be sufficiently high and discount factor sufficiently different from one; see Section IV for definition of the reward function.

that corresponds to the probability of achieving a safe state ($q \geq 0$) with action c. Search for the optimal action c^* (Equation 5) is restricted only to safe actions:

$$S(c,\widetilde{\mathbf{x}}) > \epsilon. \quad (7)$$

Active Exploration: If none of the available actions satisfies the safety condition (Equation 7), the robotic arm is used to measure some of the missing terrain features; see Fig. 2 for the pipeline overview. In Section VI, we propose several strategies that guide the active exploration of missing features in order to find a safe action as fast as possible. If all terrain features have already been measured and there is still no action satisfying the safety condition, manual flipper control is requested from the operator.

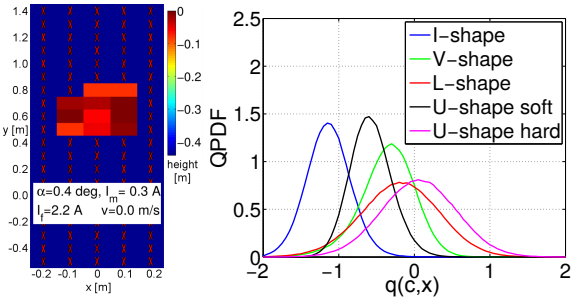


Fig. 3. **Example of insufficient data. An active exploration is necessary.** The left figure shows the input data; the missing heights in the DEM are outlined by red crosses in a blue rectangle, pitch is denoted by α, mean absolute current over both main tracks is denoted I_m, mean absolute current in the engines lifting the front flippers is denoted by I_f. More details on the features are given in Section IV. The right figure contains QPDFs for the five flipper configurations ("*-shape"). The horizontal axis corresponds to the sum of discounted rewards (higher are better), vertical axis contains QPDF. Figure adapted from [3].

Fig. 3 shows an example situation when active exploration is needed. Looking at the right figure, the highest value of the safety measure $S(c,\widetilde{\mathbf{x}})$ is approximately 0.5. If the safety limit ϵ is 0.8, tactile exploration is activated, because no action satisfies the safety limit in the current state.

IV. Adaptive Traversability Task

The AT task is solved for a tracked robot equipped with two main tracks, four independent articulated subtracks (*flippers*) with customizable compliance[2], rotating 2D laser scanner (SICK LMS-151), Kinova Jaco robotic arm, and an IMU (Xsens Mti-G); see Fig. 1. The task is detailed in the following paragraphs, and a short summary is given in Table I.

States: The state of the robot and the local neighboring terrain is modeled as n-dimensional feature vector $\mathbf{x} \in \mathbb{R}^n$ consisting of: **i) exteroceptive features:** Individual scans from one sweep of the rotating laser scanner (3 seconds) are put into an Octomap [32] with cube size of $5\,\mathrm{cm}$. This Octomap is then cropped to close neighborhood of the robot ($50\,\mathrm{cm} \times 200\,\mathrm{cm}$ size). Further, the cubes are aggregated into $10\,\mathrm{cm} \times 10\,\mathrm{cm}$ columns and mean height in each of these

[2]Upper limit of current in the flipper motor used to hold the flipper in position.

TABLE I
DESCRIPTION OF THE STATES, ACTIONS AND REWARDS

State	$\mathbf{x} \in \mathbb{R}^n$	DEM, speed, roll, pitch, flipper angles, compliance, currents in flippers, actual flipper configuration
Actions	$c \in \mathbf{C} = \{1 \dots 5\}$	5 pre-set flipper configurations [2]
Reward	$r(c, \mathbf{x}) : \mathbf{C} \times \mathbb{R}^n \to \mathbb{R}$	$\alpha \times$ user reward $s_{c,\mathbf{x}} + \beta \times$ pitch penalty $+ \gamma \times$ roughness penalty

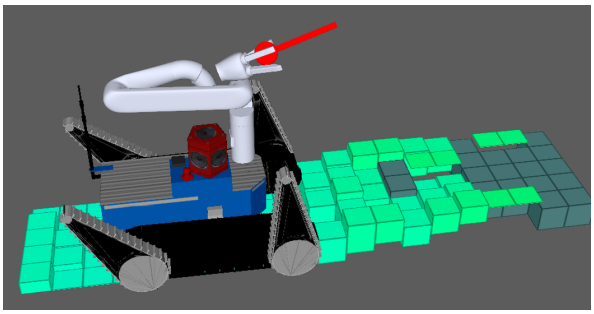


Fig. 4. **Digital Elevation Map (DEM):** Example of the DEM representation with dark green used for missing values and light green representing height estimate included in the feature space.

columns is computed. This yields a local representation of the terrain with x/y sub-sampled to $10\,\mathrm{cm} \times 10\,\mathrm{cm}$ tiles (bins) and vertical resolution of $5\,\mathrm{cm}$. This is what we call a Digital Elevation Map (DEM); see Fig. 4. Heights in the bins are used as exteroceptive features. **ii) proprioceptive features:** Robot speed (actual and desired), roll, pitch, flipper angles, compliance thresholds, actual current in flippers and actual flipper configuration.

Actions: The robot has many degrees of freedom, but only some of them are relevant to the traversal. The speed and heading of the robot are controlled by the operator. AC is used to control the pose of the four flippers and their compliance, yielding together 8 DOF. Further simplification of the action space is allowed by observations made during experiments—only 4 discrete (laterally symmetric) flipper configurations are enough for most of the terrain types, and 2 different levels of compliance are also sufficient. The arm has to be in a stable default "transport" position when the robot moves, so its DOFs are ignored. Finally, 5 *flipper configurations* denoted by $c \in \mathbf{C} = \{1 \dots 5\}$ are defined. These configurations named *I-shape*, *V-shape*, *L-shape*, *U-shape soft* and *U-shape hard* are described in detail in [2].

Rewards: The reward function $r(c, \mathbf{x}) : (\mathbf{C} \times \mathbb{R}^n) \to \mathbb{R}$ assigns a real-valued reward for using c in state $\mathbf{x}$. It is expressed as a weighted sum of (i) user-denoted bipolar penalty $s_{c,\mathbf{x}}$ specifying whether executing c in state $\mathbf{x}$ is *permitted* (safe), (ii) high pitch angle penalty (preventing robot's flip-over), and (iii) the motion roughness penalty measured by accelerometers.

V. QPDF REPRESENTATION AND LEARNING

In our previous work [2] piecewise constant functions were introduced as a method to represent Q functions. For the case of missing features, Gaussian Processes with Rational Quadratic kernel were used to represent Q functions in our following work [3]. In the latter work, Regression Forests are trained on fetaures completed by Gibbs sampling marginalization of the missing features. In this section, we propose two new approaches to QPDF representation that tackle the case of incomplete data.

A. Regression Forests

The first method is based on Regression Forests with incomplete data on their input, representing the QPDF in their leaves (instead of first estimating the missing features and then computing Q from a full feature vector, as the previous method does). Thus we avoid the unnecessary step of reconstructing the missing features, and can directly use the incomplete input to estimate QPDF.

Learning: The QPDF for each configuration is modeled independently by a Regression Forest. The trees are constructed sequentially, always building one until all leaves are *terminal* (see further), and then starting to build another one. To train each particular tree, a training set consisting of m training samples $[\mathbf{x}_1, \dots, \mathbf{x}_m]$ is given, with corresponding q-values $[q_1, \dots, q_m]$. Each training sample $\mathbf{x}_k$ is an n-dimensional vector of features $\mathbf{x}_k = [x_k^1 \dots x_k^n]^\top$. The tree is built by a greedy recurrent algorithm, that selects the splitting feature $j^* \in J = \{1 \dots n\}$ and split threshold s^*. The splitting feature and threshold are selected to minimize the weighted variance of q-values in the left and right sub-tree in each node as follows [3]:

$$(s^*, j^*) = \underset{(s,j)}{\operatorname{argmin}} \; |R_1(s,j)| \cdot \underset{k \in R_1(s,j)}{\operatorname{var}}(q_k) + |R_2(s,j)| \cdot \underset{k \in R_2(s,j)}{\operatorname{var}}(q_k)$$

where $R_1(s,j) = \{k \mid x_k^j \leq s\}$ is the set of indices descending to the left sub-tree, and $R_2(s,j) = \{k \mid x_k^j > s\}$ is the set of indices descending into the right sub-tree. The tree is constructed recursively. If a stopping criterion is satisfied (either minimum number of samples per node, or tree height), a *terminal leaf* is created, which contains discretized QPDF histogram (estimated from q-values of all training samples that descended to that leaf). Specifically, if the value of the splitting feature is unknown in sample $\mathbf{x}_i$ (e.g. occluded), then it descends into both sub-trees.

Marginalization: To obtain the marginalized distribution $p(q|c, \widetilde{\mathbf{x}})$, sample $\widetilde{\mathbf{x}}$ is put to the input of the forest. If a tested feature is missing in $\widetilde{\mathbf{x}}$, the algorithm descends into both sub-trees similarly to the learning procedure. The final QPDF is then a weighted average of histograms in all reached leaves in all trees (properly normalized to be a distribution). Weights are given by prior probabilities of leaves estimated from training data. We call this *Multiple Leaves marginalization*.

B. Gaussian Processes

Gaussian processes [23] are the extension of multivariate Gaussians to infinite-size collections of real valued variables

and can be understood as joint Gaussian distributions over random functions. The essential part of GP learning is given by the choice of a kernel function (parametrized by a set of hyper-parameters $\boldsymbol{\theta}$). We use the common *Squared Exponential* kernel function (SE), for which the *Uncertain* Gaussian Processes are derived in [33]. This allows processing features with unknown or uncertain values. In case Uncertain GPs are not necessary, i.e. Gibbs sampling is used to handle uncertain values (as in [3]), the *Rational Quadratic* (RQ) kernel that performs slightly better than SE can be used. Both SE and RQ kernels enable *Automatic Relevance Determination* [34], which can be interpreted as embedded feature selection performed automatically when optimizing over the kernel hyper-parameters $\boldsymbol{\theta}$. The ARD values are utilized in Section VI-B.

Learning: A standard regression model is used, assuming the data $\mathcal{D} = \{\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]^T, \mathbf{q} = [q_1, \ldots, q_m]^T\}$ were generated according to $q_i = h(\mathbf{x}_i) + \epsilon_i$, where $h : \mathbb{R}^n \to \mathbb{R}$, and $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is independent Gaussian noise. Thus, there is a direct connection between $h(\mathbf{x})$ and the QPDF. For each configuration c, the Uncertain GP learning procedure is used to train a GP model that predicts the given q-values. The learning procedure[3] is described in detail in [33].

Marginalization: GPs consider h as a random function in order to infer posterior distribution $p(h|\mathcal{D})$ over h from the GP prior $p(h)$, the data $\mathcal{D}$, and assumption on smoothness of h [33]. The posterior is estimated to make predictions at inputs (the testing data) $\mathbf{x} \in \mathbb{R}^n$ about the function values $h(\mathbf{x})$, which can be used as the QPDF. Since the posterior is no longer a Gaussian, it is approximated by a Gaussian distribution, using e.g. the Moment Matching method described in [23].

VI. Tactile Terrain Exploration

Given the QPDF, safety condition (Equation 7) is evaluated for all possible configurations. If more than one safe configuration exists, AC chooses the one that yields the highest q-value mean. If none of the configurations is safe, the robot is stopped and Tactile Terrain Exploration (TTE) is triggered (example situation is depicted in Fig. 3). This exploration utilizes the robotic arm to measure the height in DEM bins in which measurements are missing[4]. The arm actively explores the missing heights until the safety condition (Equation 7) is satisfied for at least one configuration, or there are no more missing heights (we refer to both these cases as *final states*). If the state in the latter case is still unsafe, the operator is asked to control the flippers manually.

We propose several TTE strategies. The simplest TTE strategy selects the bin to be explored randomly from the set of all missing bins—we refer to this strategy as **Random**. Further, we propose and evaluate also two better TTE strategies: (i) the Reinforcement-Learning–based strategy trained on synthetically generated training exploration roll-outs (further referred to as **RL** strategy), and (ii) a strategy based on *Automatic Relevance Determination* coefficients for QPDFs modeled by the GP (further referred to as the **ARD** strategy).

[3]Due to the page limitation, the detailed equations are not given here.

[4]The exploration using robotic arm is inherently slow. However, when needed, it is still worth the extra time.

A. RL from Synthetically Generated Training Set

The Reinforcement-Learning–based TTE learns a policy that minimizes the number n of tactile measurements needed to satisfy the safety condition. In our implementation, a **state** is the union of the state used in the AT task (i.e. the proprioceptive and exteroceptive measurements), and the binary mask denoting DEM bins with missing heights. **Actions** are discrete decisions to measure the height in particular bins. **Rewards** equal zero until a final state is reached. In the final state, the roll-out ends and a reward equal to $1/n$ is assigned (i.e. the longer it takes, the lower the reward).

Since it is not easy to collect sufficient amount of real examples with naturally missing features, we generate training samples from the real data with synthetically occluded DEMs. The active exploration policy is thus trained by revealing the already known (but synthetically occluded) heights. The Q-learning algorithm learns the strategy in several episodes. The initial training set is generated by simulating thousands of TTE roll-outs with the *Random* strategy. The Q function is modeled by a Regression Forest similar to the one used in Section V (but this Q function is different from the one used for Autonomous Control!). Once the Q function is learned, the corresponding strategy is used to guide training data collection in the following episode by the DAgger algorithm [35]. In each episode of the DAgger algorithm, the learned policy is used to select bins just with 0.5 probability, otherwise the Random strategy is used (which supports exploration in the policy space). After each episode, the policy is updated using the Q-learning recurrent formula (Equation 1).

B. ARD for Gaussian Processes

In Section V-B, it is mentioned that both SE and RQ kernels allow for *Automatic Relevance Determination* (ARD), which acts as feature selection. The ARD values are computed during kernel hyper-parameters optimization (when training the GP), so no extra computing power is needed. When the learning is done, for each dimension (feature) d of the input data, we have a number $ARD(d)$ that describes how much this dimension influences the output of the GP (lower values mean higher importance). The TTE strategy utilizing ARD values is as follows: **i)** estimate QPDFs using all GP models, **ii)** select the action (GP model) with the highest Q-value (QPDF mean), **iii)** in this GP, compare $ARD(d)$ values for all DEM bin features that are missing in the current state, and choose the bin with the minimum $ARD(d)$ value, **iv)** the chosen bin is then explored using the arm. This corresponds to choosing the missing feature whose value, if known, maximally influences the QPDFs.

EXPERIMENTS

Experimental evaluation is divided into three sections. In Section VII, we test the ability of AC to decrease cognitive load of human operators while maintaining roughly the same or better performance. Experiments in Section VIII demonstrate that if the DEM is partially occluded, the proposed method yields better results than the previous methods. Last,

Section IX compares Random, ARD and RL methods for tactile exploration.

In the experiments, different Q function/QPDF representations are denoted by **PWC** for piecewise constant function proposed in [2], **GP-RQ** stands for Gaussian Processes with Rational Quadratic kernel used in [3], **GP-SE** denotes the Uncertain GPs with Squared Exponential kernel, and finally the Regression Forests defined in Section V are referred to as **Forest**. The PWC and GP-RQ models can be used either with Least Squares (**LSq**) interpolation of missing features, or with **Gibbs** sampling used to marginalize the Q function over the missing data. Regression Forests utilize the **Multiple Leaves marginalization**.

A metric called *success rate* is used throughout the experiments to measure the traversal performance both on training data (in the learning phase) and on test data. It requires that the bipolar manually-assigned part of reward $s_{c,\mathbf{x}}$ defined in Section IV is assigned for all actions in all states in the dataset (not just for a single action, as is required for the learning). The success rate denotes the ratio of states, in which the AC algorithm selects one of the desired (safe) configurations. Formally:

$$\text{success rate}(\mathbf{X}) = \frac{|\{\mathbf{x} \in \mathbf{X} : c = c^*(\mathbf{x}) \wedge s_{c,\mathbf{x}} = 1\}|}{|\mathbf{X}|} \quad (8)$$

where $\mathbf{X}$ is a set of states, and $c^*(\mathbf{x})$ is the optimal configuration from Equation 3 or Equation 5 (depends on the used AC algorithm).

VII. Autonomous Control for Teleoperation

We evaluate performance of the AC algorithm (without tactile exploration) on a large dataset comprising of 8 different obstacles (some of them depicted in Fig. 5) in 3 types of environment (forest, stairs, hallway) with the robot driven by 3 different operators in both MC and AC modes[5]. Each of the traversals is repeated 3-10 times to allow for statistical evaluation. The operators driving the robot are denoted as **E** (Experienced), **IE** (InExperienced) and **IE2** (InExperienced #2). The experiments cover more than 115 minutes of robot time during which the robot traveled over 775 meters.

Experiments in this section only show the results achieved with Regression Forests; other Q function representations were tested in [2], [3], and Uncertain Gaussian Processes were only tested together with the tactile exploration (see Section IX), since without TTE they performed worse than the Regression Forests (and for creating such a large dataset, we had to choose one method).

A. Training Procedure

The algorithm was trained in controlled lab conditions using two artificial obstacles created from EUR pallets[6] and a staircase. The first obstacle is just a single pallet and the second one is a simple simulated staircase composed from three pallets.

[5]See the attached multimedia showing the test drives.

[6]Type EUR 1: $800\times1200\times140\,\text{mm}$, see en.wikipedia.org/wiki/EUR-pallet

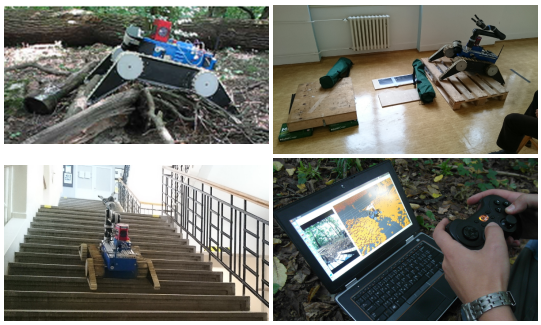


Fig. 5. Top left: *Forest* obstacle. Top right: *Rubble* obstacle. Bottom left: *Stairs* obstacle. Bottom right: Operator controlling the robot using only sensor data.

We trained the QPDFs represented by RF (one QPDF per flipper configuration) using the algorithm described in Section V. Except the standard learning validation metrics, we also evaluated the *success rate* (Equation 8). We trained the RF QPDF model, and we accomplished a success rate of $97\,\%$ (which is shown in Fig. 6).

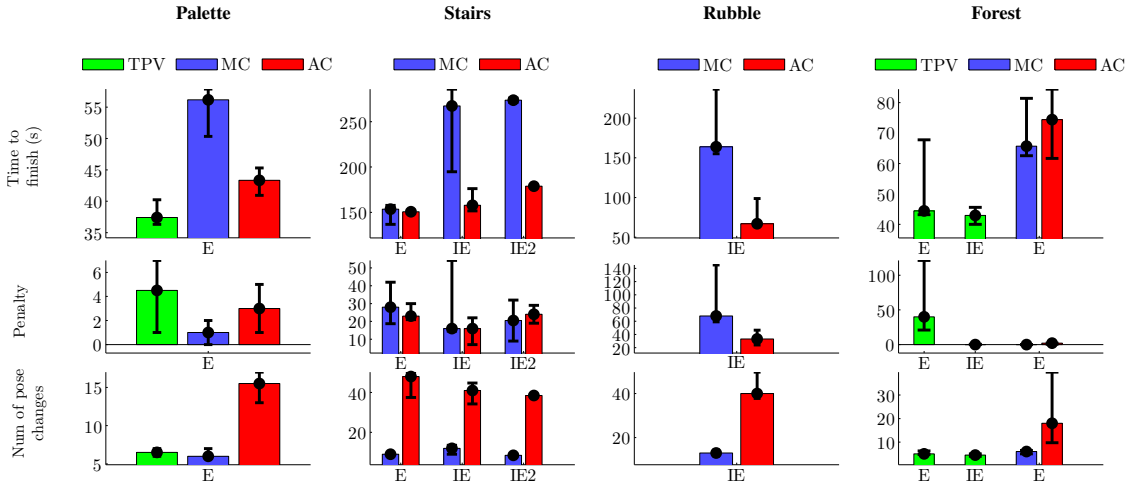
B. Testing Procedure

Each obstacle was traversed multiple times with both manual (**MC**) and autonomous flipper control (**AC**) using RFs following Equation 5, and the sensed states contained naturally missing DEM features. We emphasize that the complexity of testing obstacles was selected in order to challenge robot hardware capabilities. See the examples in Fig. 5 and the elevation maps (DEM) of testing obstacles computed online by the robot in Table IIIb.

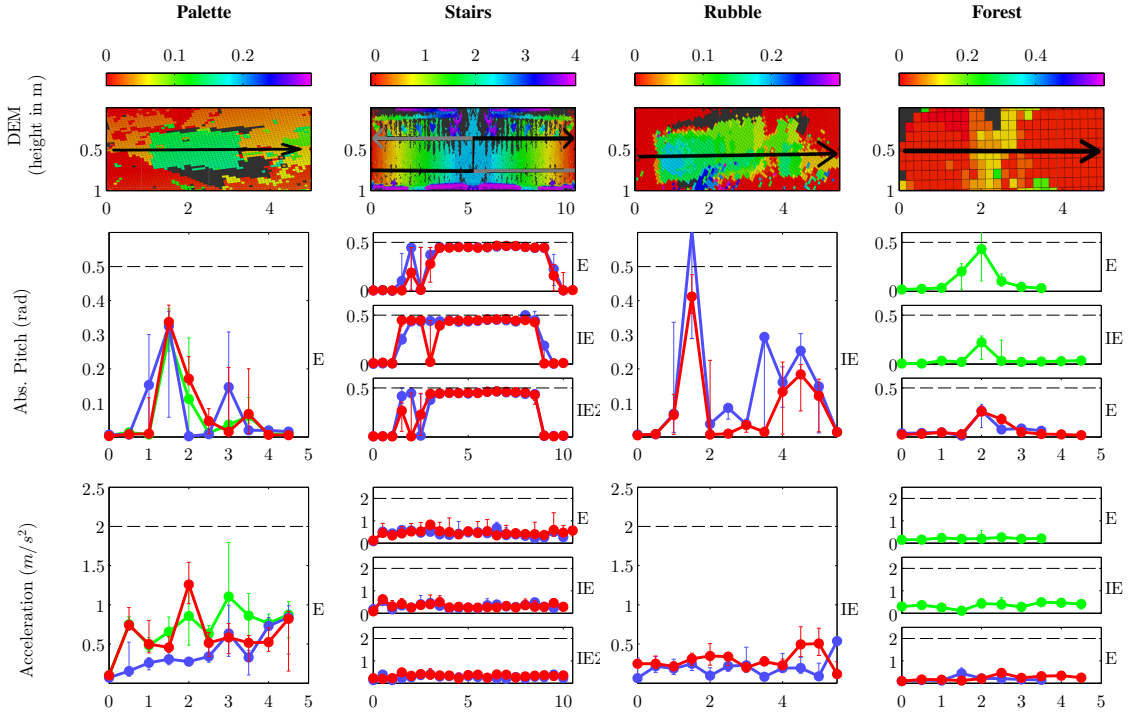
There is an additional mode called **TPV** (*Third Person View*) in which the operator had not only the robot sensory data available, but he directly looked at the robot (thus having much more information than the robot can get). Except for the TPV mode, the operators were only allowed to drive the robot based on data coming from the robot sensors (3D map + robot pose from sensor fusion [36]), which should accomplish a fair comparison of AC and MC. The TPV mode should be treated as a sort of baseline—it is not expected that AC or MC could be better than TPV in all aspects.

To compare AC and MC quality, three different metrics were proposed and evaluated: (i) traversal time (start and end points are defined spatially), (ii) a sum of pitch angle penalty and roughness of motion penalty, and (iii) the number of flipper configuration changes (which increases cognitive load of the operator in MC, and with the current manual controller, it also takes approx. $1\,\text{s}$ to change the flipper configuration and the robot has to be stopped). Table IIIa and Table III show quantitative evaluation of some of the experiments. Table IIIa depicts 4 out of 8 experiments carried out to verify performance of AC using the best method found—Regression Forests with Multiple Leaves marginalization. All errorbars denote quartiles of the measured values and the circles are in the positions of medians.

TABLE II
EXPERIMENTAL EVALUATION OF ADAPTIVE TRAVERSABILITY



(a) **Overall statistics of the experiments.** The computation of penalties is described in (b). High penalties for experienced operator with 3rd person view (TPV) are given by the fact that an experienced operator allows himself to drive harsher to finish the task faster.



(b) **Penalty details.** The horizontal axis always denotes distance traveled during the experiment. Dashed lines in *Pitch* and *Acceleration* show the thresholds (0.5 rad or 2.5 m s^{-2}) for counting a *penalty point* (which are plotted in Table IIIa). *Acceleration* reflects the "roughness of motion" (the higher it is, the worse for the mechanical construction of the robot). It is computed as $\sqrt{a_x^2 + a_z^2}$ and is averaged over 0.2 s intervals (where a_x is the horizontal acceleration perpendicular to robot motion, and a_z is vertical acceleration with gravity subtracted).

Table III summarizes all MC/AC experiments (excluding TPV mode experiments, since they should not be compared with MC/AC).

TABLE III
EVALUATION OF ALL AT EXPERIMENTS

Obstacle	Operator	Time to finish [s]		Penalty		Pose changes	
		MC	AC	MC	AC	MC	AC
Pallet long	E	56.2	**43.3**	**1**	3	**6**	16
Pallet short	E	41.0	**39.3**	**2**	4	**6**	15
Stairs	E	154.0	**150.6**	28	**23**	**9**	48
	IE	267.3	**157.9**	**16**	**16**	**12**	41
	IE2	273.7	**178.8**	**21**	24	**9**	39
Rubble 1	IE	164.0	**66.9**	68	**33**	**13**	40
Rubble 2	IE2	114.0	**63.2**	7	**3**	**10**	26
Forest 1	E	**65.7**	74.4	**0**	2	**6**	18
Forest 2	E	36.8	**35.7**	N/A	N/A	**2**	3
Forest 3	E	132.1	**75.3**	N/A	N/A	**4**	10

Each pair of columns (MC/AC) shows the medians of the 3 metrics evaluated for the experiments. Of each pair, the value in bold is better. Experiments *Forest 2* and *Forest 3* are those conducted in [2]. Both robot construction and AT algorithm changed in the meantime, so the values should not be compared to the new results.

C. Results

It can be seen in Table III that the *Time to finish* with AC tends to be shorter or comparable to MC (and with TPV, it is even shorter, as expected). Subjectively, the operators report a much lower level of cognitive load when driving with AC, which means they can pay more attention to exploration or other tasks.

Penalties with AC are also mostly better or comparable to MC. The *number of flipper configuration changes* for AC is approximately 2- to 4-times higher than for MC. However, with AC, there is no time penalty for changing flipper configurations, and it also adds no more cognitive load to the operator.

From the experiments conducted it follows that AC yields similar or even better performance than MC. Furthermore, AC allows the operator to concentrate rather on higher-level tasks while having the tedious and low-level flipper control done automatically.

VIII. ROBUSTNESS TO MISSING EXTEROCEPTIVE DATA

In this experiment, we quantitatively evaluate robustness to the number of missing features for the various Q/QPDF representations. The robustness is presented as the relation between success rate and the number of synthetically occluded DEM bins.

The Regression Forests first compute the marginalized QPDFs as described in Section V, and then choose a configuration according to Equation 5. The LSq interpolation/Gibbs sampling methods first interpolate or marginalize the missing data, then compute the Q function on the interpolated data and choose the configuration according to Equation 3.

For this experiment, a dataset consisting of hundreds of captured robot states (interoceptive + full exteroceptive features) is used. The bipolar manual annotations $s_{c,\mathbf{x}}$ are assigned to all state-action combinations.

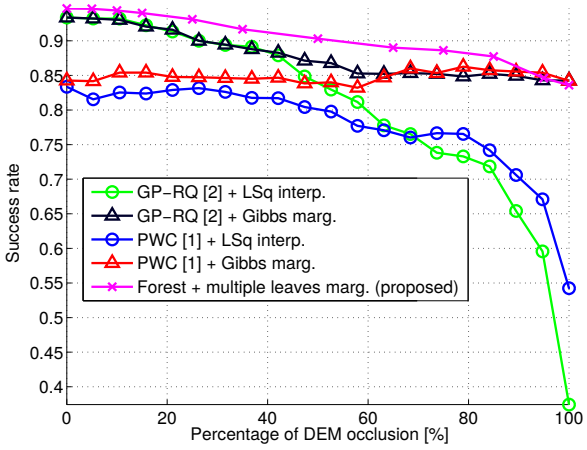


Fig. 6. **Robustness to DEM occlusion:** The chart shows the influence of DEM occlusion (percentage of DEM bins in which measurements are not available) on AC success rate. When $100\,\%$ of DEM is occluded, the marginalized policies still depend on proprioceptive measurements, while LSq interpolation reconstructs only flat terrain.

For $i = 0 \ldots 100$, the set "states$_i$" is generated from the dataset by occluding i DEM bins in each of the captured states $\mathbf{x}$ (the same manual annotation $s_{c,\mathbf{x}}$ is used for all states $\tilde{\mathbf{x}}$ generated from $\mathbf{x}$). To avoid combinatorial explosion, we did not try all combinations of i occluded bins. We chose to successively occlude DEM bins from the front of the robot, until i bins are occluded. Therefore, the dataset the robustness is tested on contains tens of thousands of different states. The success rate in Fig. 6 is computed as success rate(states$_i$) according to Equation 8.

Fig. 6 shows superiority of marginalizing methods over LSq interpolation. Up to a DEM occlusion level of $40\,\%$, all methods behave comparably. The reasons are two-fold: (i) the part of the occluded DEM is far in front of the robot and there is no way to sense it from the proprioceptive measurements, (ii) the obstacle hidden in this part of DEM is usually far enough, therefore the V-shape configuration (the one for flat terrain) is still allowed in most of the testing data. When more than $40\,\%$ are hidden, success rate of the LSq interpolation method drops rapidly down towards $0.4 - 0.5$ (i.e. $40\,\%$-$50\,\%$ of states in which the permitted configuration is selected) for both GP and PWC, while the marginalizing methods preserve high precision. The figure also demonstrates that the proposed Regression Forests provide better success rate than the previous methods [2], [3].

IX. TACTILE TERRAIN EXPLORATION

To compare the strategies for Tactile Terrain Exploration (TTE), we evaluate them on real (test) data with the front $50\,\%$ of the DEM synthetically occluded (since it is not easy to provide a sufficient amount of real examples with naturally missing features). Active exploration is simulated by revealing the already known DEM heights.

The performance of TTE strategies can be expressed as the average number of actively measured bin heights until a

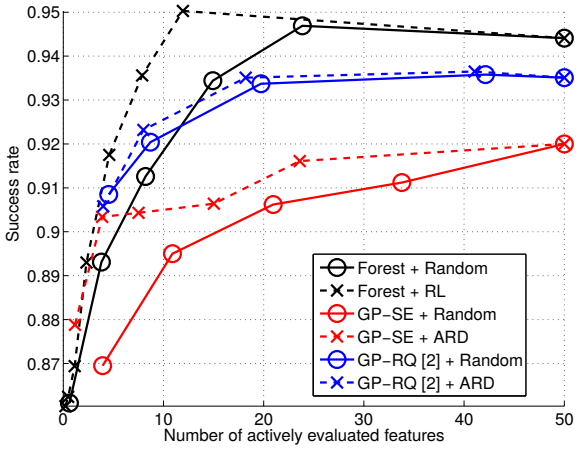


Fig. 7. **Comparison of TTE methods:** Curves on this graph show success rate (with 50 % DEM bins occluded) as a function of the number of measured bin heights. The compared TTE strategies are described in Section VI.

safe configuration is found. However, for this experiment, we let the exploration continue even if a safe configuration has already been found, to see how much further exploration helps. For different QPDF models and TTE strategies, the relation between the number of measured heights and the success rate is depicted in Fig. 7.

An ideal QPDF model and strategy would achieve 100 % success rate with a single evaluated feature, i.e. the upper-left corner in Fig. 7. The closer is the curve to this corner, the better is the method. Results with the lowest success rate were achieved with the GP-SE method (however, the ARD strategy yields a significant improvement). Better results were achieved by the GP-RQ method (for which the ARD strategy yields only small improvement compared to the Random strategy). The reason is that the RQ kernel allows for better generalization than the SE kernel. For less than 15 features actively evaluated (i.e. smaller safety thresholds), the GP-RQ method achieves higher success rate than the Regression Forest method with Random strategy. The best method in this comparison are Regression Forests combined with the RL strategy, which achieve the best success rate.

X. Conclusion

We extended the Autonomous Control algorithm [2], [3] that increases autonomy in mobile robot control and reduces cognitive load of the operator. To deal with only partially observable terrain, missing or incorrect data, we (i) designed and experimentally verified a more occlusion-robust QPDF model, and (ii) we exploit a body-mounted robotic arm as an additional active sensor for Tactile Terrain Exploration. TTE is used in dangerous situations, where all actions have negative expected rewards. The previous methods have to choose one of the actions, even if the best expected reward is negative. By tactile exploration of the unobserved part of the terrain, the reward estimates get better and at least one of them should get positive if the terrain is traversable. Several TTE strategies were proposed and experimentally evaluated. We conclude that the overall highest success rate was achieved by combining Regression Forests with the RL strategy for the arm-based exploration of missing data.

References

[1] G. Kruijff, I. Kruijff-Korbayová, S. Keshavdas, B. Larochelle, M. Janíček, F. Colas *et al.*, "Designing, developing, and deploying systems to support human-robot teams in disaster response," *Advanced Robotics*, vol. 28, DOI 10.1080/01691864.2014.985335, no. 23, pp. 1547–1570, Dec. 2014.

[2] K. Zimmermann, P. Zuzánek, M. Reinstein, and V. Hlaváč, "Adaptive traversability of unknown complex terrain with obstacles for mobile robots," in *Proc. IEEE Int. Conf. Rob. Autom.*, Jun. 2014, pp. 5177–5182.

[3] K. Zimmermann, P. Zuzánek, M. Reinstein, T. Petříček, and V. Hlaváč, "Adaptive traversability of partially occluded obstacles," in *Proc. IEEE Int. Conf. Rob. Autom.*, May. 2015.

[4] F. Colas, S. Mahesh, F. Pomerleau, M. Liu, and R. Siegwart, "3D path planning and execution for search and rescue ground robots," in *Proc. IEEE Int. Conf. Intell. Rob. Syst.*, DOI 10.1109/IROS.2013.6696431, Nov. 2013, pp. 722–727.

[5] M. Brunner, B. Bruggemann, and D. Schulz, "Towards autonomously traversing complex obstacles with mobile robots with adjustable chassis," in *13th Int. Carpathian Cont. Conf.*, DOI 10.1109/CarpathianCC.2012.6228617, May. 2012, pp. 63–68.

[6] C.-C. Tsai, H.-C. Huang, and C.-K. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," *IEEE Trans. Ind. Electron.*, vol. 58, DOI 10.1109/TIE.2011.2109332, pp. 4813–4821, Jan. 2011.

[7] M. Gianni, F. Ferri, M. Menna, and F. Pirri, "Adaptive robust three-dimensional trajectory tracking for actively articulated tracked vehicles (AATVs)," *Jour. Field Rob.*, Apr. 2015.

[8] S. Martin, L. Murphy, and P. Corke, *Experimental Robotics: The 13th International Symposium on Experimental Robotics*. Heidelberg: Springer International Publishing, 2013, ch. Building Large Scale Traversability Maps Using Vehicle Experience, pp. 891–905.

[9] K. Ho, T. Peynot, and S. Sukkarieh, "A near-to-far non-parametric learning approach for estimating traversability in deformable terrain," in *Proc. IEEE Int. Conf. Intell. Rob. Syst.*, DOI 10.1109/IROS.2013.6696756, Nov. 2013, pp. 2827–2833.

[10] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," *Advances Neural Inform. Process. Syst.*, vol. 19, p. 1, 2007.

[11] C. Weiss, H. Frohlich, and A. Zell, "Vibration-based terrain classification using support vector machines," in *Proc. IEEE Int. Conf. Intell. Rob. Syst.*, DOI 10.1109/IROS.2006.282076, Oct. 2006, pp. 4429–4434.

[12] K. Kim, K. Ko, W. Kim, S. Yu, and C. Han, "Performance comparison between neural network and SVM for terrain classification of legged robot," in *Proc. IEEE Soc. Instrument Cont. Eng. Annual Conf.*, Aug. 2010, pp. 1343–1348.

[13] E. M. DuPont, C. A. Moore, and R. G. Roberts, "Terrain classification for mobile robots traveling at various speeds: An eigenspace manifold approach," in *Proc. IEEE Int. Conf. Rob. Autom.*, May. 2008, pp. 3284–3289.

[14] D. Kim, J. Sun, S. Min, O. James, M. Rehg, and A. F. Bobick, "Traversability classification using unsupervised on-line visual learning for outdoor robot navigation," in *Proc. IEEE Int. Conf. Rob. Autom.*, May. 2006, pp. 518–525.

[15] L. Ojeda, J. Borenstein, G. Witus, and R. Karlsen, "Terrain characterization and classification with a mobile robot," *Jour. Field Rob.*, vol. 23, pp. 103–122, Feb. 2006.

[16] K. Ho, T. Peynot, and S. S. Sukkarich, "Traversability estimation for a planetary rover via experimental kernel learning in a Gaussian Process framework." in *Proc. IEEE Int. Conf. Rob. Autom.*, May. 2013, pp. 3475–3482.

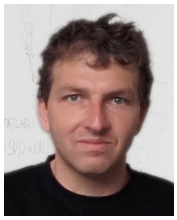
[17] M. Reinstein, V. Kubelka, and K. Zimmermann, "Terrain adaptive odometry for mobile skid-steer robots," in *Proc. IEEE Int. Conf. Rob. Autom.*, May. 2013, pp. 4706–4711.

[18] M. Reinstein and M. Hoffmann, "Dead reckoning in a dynamic quadruped robot based on multimodal proprioceptive sensory information," *IEEE Trans. Rob.*, vol. 29, no. 2, pp. 563–571, Apr. 2013.

[19] G. Zhong, H. Deng, G. Xin, and H. Wang, "Dynamic hybrid control of a hexapod walking robot: Experimental verification," *IEEE Trans. Ind. Electron.*, vol. PP, DOI 10.1109/TIE.2016.2551679, no. 99, pp. 1–1, 2016.
[20] P. Abbeel and A. Y. Ng, "Exploration and apprenticeship learning in reinforcement learning," in *Proc. 22nd Int. Conf. Machine Learning*, DOI 10.1145/1102351.1102352, Aug. 2005, pp. 1–8.
[21] J. Yu, C. Wang, and G. Xie, "Coordination of multiple robotic fish with applications to underwater robot competition," *IEEE Trans. Ind. Electron.*, vol. 63, DOI 10.1109/TIE.2015.2425359, no. 2, pp. 1–8, Feb. 2016.
[22] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake *et al.*, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, Jan. 2013.
[23] M. Deisenroth, D. Fox, and C. Rasmussen, "Gaussian Processes for data-efficient learning in robotics and control," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 37, no. 2, pp. 408–423, Feb. 2015.
[24] D. J. Lizotte, L. Gunter, E. Laber, and S. A. Murphy, "Missing data and uncertainty in batch Reinforcement Learning," in *Proc. Neural Inform. Process. Syst.*, 2008.
[25] M. A. Tanner and W. Wong, "The calculation of posterior distributions by data augmentation," *Jour. Amer. Stat. Assoc.*, vol. 82, pp. 528–540, 1987.
[26] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an EM approach," in *Advances Neural Inform. Process. Syst. 6*. Morgan Kaufmann, 1994, pp. 120–127.
[27] A. Doumanoglou, T.-K. Kim, X. Zhao, and S. Malassiotis, "Active random forests: An application to autonomous unfolding of clothes," in *European Conf. Comput. Vision*. Springer, Sep. 2014, pp. 644–658.
[28] M. Bjorkman, Y. Bekiroglu, V. Hogman, and D. Kragic, "Enhancing visual perception of shape through tactile glances," in *Proc. IEEE Int. Conf. Intell. Rob. Syst.*, Nov. 2013, pp. 3180–3186.
[29] Z. Jia, Y.-J. Chang, and T. Chen, "A general boosting-based framework for active object recognition," in *British Machine Vision Conf.*, Sep. 2010, pp. 1–11.
[30] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, DOI 10.1007/BF00992698, no. 3-4, pp. 279–292, May. 1992.
[31] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Jour. of Artificial Intell. Research*, vol. 4, DOI 10.1613/jair.301, pp. 237–285, May. 1996.
[32] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Rob.*, DOI 10.1007/s10514-012-9321-0, Apr. 2013.
[33] M. P. Deisenroth, *Efficient Reinforcement Learning using Gaussian Processes*. KIT Scientific Publishing, 2010, vol. 9.
[34] C. E. Rasmussen, *Advanced Lectures on Machine Learning*. Berlin, Heidelberg: Springer, 2004, ch. Gaussian Processes in Machine Learning, pp. 63–71.
[35] S. Ross and J. A. Bagnell, "Agnostic system identification for model-based reinforcement learning," in *Proc. 29th Int. Conf. Machine Learning*, Sep. 2012, pp. 1703–1710.
[36] V. Kubelka, L. Oswald, F. Pomerleau, F. Colas, T. Svoboda, and M. Reinstein, "Robust data fusion of multimodal sensory information for mobile robots," *Jour. Field Rob.*, vol. 32, DOI 10.1002/rob.21535, no. 4, pp. 447–473, Jun. 2015.

Martin Pecka received the Mgr. (M.Sc.) degree in theoretical informatics at the Faculty of Mathematics and Physics, Charles University in Prague, Czech republic, in 2012.

He has currently been a Ph.D. student at the Department of Cybernetics, Czech Technical University in Prague (CTU), and a Research Assistant at the Czech Institute of Informatics, Robotics and Cybernetics, CTU in Prague. His main research interests are in the fields of machine learning, specifically reinforcement learning concerning safety of execution, and robotics.



Karel Zimmermann (M'08) received the Ph.D. degree in cybernetics from the Czech Technical University in Prague, Czech Republic, in 2008.

He worked as Postdoctoral Researcher with the Katholieke Universiteit Leuven (2008-2009). Since 2009, he has been a Postdoctoral Researcher at the Czech Technical University in Prague. His current research interests include learnable methods for tracking, detection and robotics.

Dr. Zimmermann serves as a reviewer for major journals such as TPAMI, IJCV and conferences such as CVPR, ICCV, ICRA, IROS. He received the best reviewer award at CVPR 2011 and the main prize for the best PhD thesis in Czech Republic in 2008 (awarded by the Czech Society for Pattern Recognition).

Michal Reinstein (M'11) received the Ing. (M.Sc.) and Ph.D. degrees in engineering of aircraft information and control systems from the Faculty of Electrical Engineering, Czech Technical University in Prague (CTU), Czech republic, in 2007 and 2011, respectively.

He is currently working as Assistant Professor at the Center for Machine Perception, Dept. of Cybernetics, CTU in Prague. His most recent research interests concern application of machine learning and data fusion to satellite imagery aiming to support deployment of large-scale robotic platforms. In the past years since 2011, the main topics of his research were: sensory-motor interaction in legged robots, multimodal data fusion for robots intended for Urban Search & Rescue, and adaptive traversability of unknown terrain using reinforcement learning.

Tomas Svoboda (M'01) received the Ph.D. degree in artificial intelligence and biocybernetics from the Czech Technical University in Prague, Czech republic, in 2000.

Later, he spent three post-doc years with the Computer Vision Group at the ETH Zurich. Currently, he is Associate Professor and Deputy Head of the Department of Cybernetics at the Czech Technical University in Prague, the Director of EECS study programme, and he is also on board of Open Informatics programme. He has published papers on multicamera systems, omnidirectional cameras, image based retrieval, learnable detection methods, and USAR robotics. His current research interests include multimodal perception for autonomous systems, object detection and related applications in automotive industry.

Chapter 10

Followup and Related Work

Both literature and current events in the domain of robotics (e.g. DARPA SubT challenge presented in Appendix B) show that there is high demand for robots that can autonomously and safely traverse various dangerous environments. E.g. [Couceiro et al., 2019] propose to create an automatic system that would help maintaining forests. And forests are one of the most complicated navigation and locomotion environments. Also the previous and current work mentioned in Appendix B requires robots with complex and safe traversal skills. [Bjelonic et al., 2018] solve a very similar navigation and locomotion task for a hexapod robot. They perceive the terrain by a stereo camera, and each leg of the hexapod can also be used as a terrain-sensing device. Their control algorithm does not directly consume the stereo images, but there is a preprocessing step which extracts manually defined features of the terrain. One of the features is the number of "hidden cells" in the field of view, which correspond to *NaNs* or *missing heights* in our algorithm. The control algorithm takes the number of missing heights (but not their positions) into account and changes the gait also based on this number. But the robot is not actively forced to explore the shadowed areas. On the other hand, its terrain proprioception is much more capable than in our case, since it has six fast legs; it can thus learn to actively measure the shadowed heights right in front of the robot just by adapting the gait. [Fankhauser, 2018] concentrated on locomotion of a quadruped robot equipped with 3D laser scanners or stereo cameras. Thanks to the relatively high ground clearance of the robot, cases where there would be substantial parts of the terrain in front of the robot missing do not occur so often. Thus, the presented mapping and foothold search algorithms do not explicitly care about the missing measurements. The planner just does not plan to put a foothold on places where terrain is not measured. [Motoi et al., 2019b] and [Motoi et al., 2019a] targeted at assisted teleoperation of a mobile robot. They found useful for the operator to receive a kind of force feedback into the handheld controller, which would tell him or her if an obstacle is nearby. This idea could also be used in our algorithm—for example the feedback could be telling the operator how uncertain terrain is in the driving direction. [Celemin et al., 2019] train the desired control policy by first letting the robot do standard reinforcement learning, and whenever they see it performs badly, a teacher is present, that can take over control

and show the robot how to do the task it is trying to do. However, this is much easier implemented for compliant manipulators than for mobile robots.

Gaussian Processes are popular in reinforcement learning algorithms, mostly as estimators of the unknown transition probability. With such estimates (which also provide covariance), model-based algorithms can be converted to model-free (paying the cost of approximation and requiring much more samples to become accurate). The covariance estimates also enable the algorithms to recognize unknown situations and react to them. [Cully et al., 2015] use a Gaussian Process to model the performance of locomotion behaviors for legged robots. Whenever the observed performance is too far from the GP posterior (e.g. the robot was damaged), they try different behaviors in order to find a new one that is the best under the new conditions. [McKinnon and Schoellig, 2017] propose an ensemble of Gaussian Processes governed by a Dirichlet Process which decides if the observed transitions are consistent with one of the GPs, or whether a new GP should be added to ensemble, modeling yet unobserved behavior. [Ewerton et al., 2019] also use automatic relevance determination, but they use Pearson correlation of trajectories instead. These relevance factors are used to alter predictions of the GP that captures trajectory distributions.

There are cases where learning is not even required to produce intelligent locomotion skills. Under the assumption of perfect perception (which can be easily achieved in simulation), pure geometrical approaches also show to be effective. [Singh et al., 2014] used simulation to find optimal coefficients of controllers which drive a tracked robot with flippers over obstacles. The control algorithm, however, needs to receive the step height from an external precision measurement sensor. [Kharuzin et al., 2017] developed a obstacle traversal method for a 3-link 6-wheel actively articulated robot. Their perception module is able to read the obstacle height and distance, and the algorithm only works for box-like obstacles. [Yuan et al., 2019] and [Yuan et al., 2020] provide geometrical solutions to real tracked robot locomotion. However, their algorithm expects that perfect sensing of both the terrain and the pose of the robot can be provided, which is often an unrealistic requirement. Even when the geometry of the robot is not known in advance, intelligent locomotion can be achieved via embodiment. [Hoffmann and Pfeifer, 2018] show that intelligence contained in the construction of the body of a robot can play a major role in its performance, and can even be considered essential for cognitive robots.

Deep neural networks started to be more and more important also in the field of Reinforcement Learning. Although their application mostly requires prohibitively many rollouts to be practical on real robots, even the simulation results can often be useful. Many algorithms employ deep networks as a translation layer between high-dimensional input (like images) and a latent state representation. [Finn et al., 2016b] make use of deep autoencoder networks, which are especially useful for dimensionality reduction. Together with further pruning of the latent space, they were even able to train some simple reinforcement learning tasks on a real manipulator with input from

RGB cameras. Recently, when deep neural networks started to perform better than humans in many tasks, requests for better explainability of their behavior appeared. One of the few works that take this requirement into account in the domain of mobile robotics is [Akrour et al., 2019b]. They suggest to use nonparametric policies represented by state-space clusters. If the dimensionality of state space is relatively low and the features are sensible to a human, such policies can indeed be considered explainable. However, extension to more dimensions and parametric policies is unclear.

The most impressive locomotion skills learned by a RL algorithm are so far those trained only in a simulator with millions of rollouts. Having so much data, the true strength of RL can be seen—having a good reward function is enough to achieve close-to-optimal behavior. [Peng et al., 2016] use an ensemble of actor-critic networks to train locomotion of various characters with large state and action spaces. Their model even includes 1.5D perception of the terrain. The result is similar to training motion primitives, as each actor specializes at a different part of the behavior, although the model does not explicitly support periodicity of the behaviors. [Sokolov et al., 2017b] used neuroevolution to find an obstacle traversal policy for a tracked robot using a vertically oriented 2D lidar. As neuroevolution also requires numerous rollouts to train, they succeeded to finish the training in simulator only, and they even had to train a special policy for each kind of obstacle.

One of the key problems when transferring control algorithms from simulators to reality is the imperfection of data. With depth– or range-sensing devices, it is even more problematic, because geometrical constraints make some measurements impossible. This problem is surprisingly overlooked in the literature, so our guess is that most authors employ some kind of averaging, approximation or marginalization on the missing data to get full state information. [Han et al., 2019] handle time-series data, and it can happen that some data do not arrive on time, or at all. Their algorithm utilizes previously captured data or a reference model to learn a prior of the data, and if a point is missing, they fill it from the prior.

As can be seen from the paper introduced in this part of the thesis, performance evaluation of reinforcement learning is not always an easy task. Sometimes the reward function cannot capture all relevant aspects of what we would call “good behavior”. Often, it is difficult to compare algorithms, because the behavior of each has only been shown on a specific platform the authors of the research have access to. Replication of experiments with real hardware has been almost impossible and impractical. One try in this direction was the Open Physical Environment Benchmark created by [Mirzaei et al., 2017]. Unfortunately, this framework did not (yet) get generally accepted. Its only practical use was described in [Mirzaei Buini, 2018], which is however a work from the same author. Another option specifically for benchmarking locomotion could be the standardized NIST step field [Jacoff et al., 2008]. However, there are not many publications that would actually use these step fields for benchmarking—it is usually used in training, or evaluation by visual inspection, but rigorous reports are missing. Probably

the most known use of these step fields is in the RoboCup Rescue League competition [Holz et al., 2019]. However, it still remains unclear what should be the measured quantities (time, accelerations, speed, agility, ...) and how to account for the effect of robot body morphology.

Chapter 11

Conclusion

In this part, we have shown that missing data in 3D perception carry non-trivial information. The most common way of dealing with them—interpolation—is shown to be suboptimal. Data-driven marginalization is more computationally intensive, but provides an increase in safety and performance of the learned policies.

We have presented two variants of Q-learning that utilize the imperfect measurements and efficiently decide whether it is needed to collect more "expensive" samples from the environment. One method is based on representing the Q-function as a regression forest, and the other is based on Ucertain Gaussian Processes. Last, we have presented two efficient methods for collection of the "expensive" samples using tactile exploration—one based on regression forest marginalization and the other on Automatic Relevance Determination for Gaussian Processes.

As the followup literature overview shows, locomotion of tracked robots in difficult terrain has still been an actively researched area. Even research for legged mobile platforms shows that capturing terrain shape data and processing them for locomotion purposes still has open questions. Algorithms that would treat missing measurements as "first-class data" and not just as spaces that need to be estimated, are scarce. Last, evaluation of locomotion policies in unstructured terrain has still been missing a framework widely adopted among robotics researchers, which results in difficult-to-compare publications and algorithms. This stresses the importance of robotics competitions, as detailed in Appendix B.

Part III

Improving the Simulator

This part is dedicated to the problem of policy transfer from an approximate (or imprecise) simulator to reality. We begin with a short summary of the problem we solved and stress out the main contributions of our paper, whose verbatim copy is included:

- Pecka, M., Zimmermann, K., Petrlik, M., and Svoboda, T. (2018). Data-Driven Policy Transfer With Imprecise Perception Simulation. *IEEE Robotics and Automation Letters*, 3(4):3916–3921. DOI: 10.1109/LRA.2018.2857927

We conclude this part with an overview of relevant literature that followed after our paper was accepted.

Chapter 12

Problem Statement

In the CREPS algorithm described in Part I, we used a rather simple method of transferring a simulator-trained policy to the real robot. We trained the policy in simulator until convergence, and then we ran such policy on the real robot, iterating again until the learning process converged on the real robot. That approach, however, requires the simulator to behave very similar to the real world, which is not always possible, desirable or practical.

Contrastingly, in Part II, we proposed a method which learns directly on the real data and does not make use of any kind of simulation. Such method was able to learn a simple discrete-action policy. However, it could not be easily extended to the continuous domain (or it would need enormous amounts of data).

To do the domain transfer more properly, several other approaches are possible. In our work [Pecka et al., 2018], we introduce a data-driven approach that helps to transform the simulated perception to be closer to what the real robot will see. As more and more real data are gathered throughout run of the algorithm, the simulation gets closer and closer to the real world. We argue that in case perception is the most important "gap" between simulation and reality, this method can close the gap and let the learning algorithm perform well even in the real world.

Chapter 13

Summary of Contributions

The contribution of our work lies in proposing a new self-contained learning–planning–transfer loop which simultaneously learns and transfers the policy using a generative model, which refines imprecise perception in simulation.

Also, the proposed method is evaluated on the Absolem UGV described in section 2.2.

Chapter 14

Paper: Data-Driven Policy Transfer With Imprecise Perception Simulation

This paper has been accepted and published in *IEEE Robotics and Automation Letters*.

3916 IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 3, NO. 4, OCTOBER 2018

Data-Driven Policy Transfer With Imprecise Perception Simulation

Martin Pecka, Karel Zimmermann, Matěj Petrlík, and Tomáš Svoboda

***Abstract*—This letter presents a complete pipeline for learning continuous motion control policies for a mobile robot when only a nondifferentiable physics simulator of robot–terrain interactions is available. The multimodal state estimation of the robot is also complex and difficult to simulate, so we simultaneously learn a generative model which refines simulator outputs. We propose a *coarse-to-fine* learning paradigm, where the coarse motion planning is alternated with guided learning and policy transfer to the real robot. The policy is jointly optimized with the generative model. We evaluate the method on a real-world platform.**

***Index Terms*—Learning from demonstration, learning and adaptive systems, reactive and sensor-based planning, domain transfer.**

I. Introduction

HIGH-DIMENSIONAL reactive motion control of complex unmanned ground robots which substantially interact with unstructured terrain is complicated. Main difficulties are threefold: (i) the sample inefficiency and local optimality of state-of-the-art reinforcement learning methods make direct policy optimization on a real platform inconceivable, (ii) the curse of dimensionality of planning methods [1] makes direct search prohibitively time-consuming, and (iii) the simulation inaccuracy of robot–terrain interactions often makes direct usage of simulator-learned policies impossible [2]. We propose a complete policy learning–planning–transfer loop, which addresses all of these issues simultaneously.

The aim of this work is to learn motion control policy for four independently articulated flippers of a tracked skid-steering robot shown in Figure 2. The proposed method exploits an analytically non-differentiable dynamics-engine–based simulator of the real platform [3]. The learned policy maps the local height map and pose of the robot to desired motion of the flippers, which assures smooth traversal over complex unstructured terrain.

The complexity of track–terrain interactions [3] slows the simulation speed down to real-time, therefore collecting a huge number of samples needed for accurate learning is impossible. Consequently, we propose coarse-to-fine policy learning, where the coarse motion planning is alternated with guided learning and policy transfer to the real robot.

The proposed method starts by planning trajectories, which approximately optimize traversal of randomly generated terrains. Then guided learning provides a coarse initial policy. Since it is impossible to simulate the state estimation described in Section IV accurately, the state estimated on the real platform significantly differs from the simulated state. Instead of precise simulation, we suggest learning a conditional generative model of the state estimation procedure, which comprises both the underlying noise of different sensors and the errors caused by fusion of multi-modal measurements. This generative model is optimized together with the policy. In addition to that, the successively learned policy allows to guide the node expansion during planning which helps to obtain more accurate plans faster. This procedure is iterated until convergence.

Contribution of the letter lies in proposing the new self-contained learning-planning-transfer loop which simultaneously learns and transfers the policy using the generative model, which refines imprecise perception in simulation. The method is evaluated on a real platform.

II. Related Work

Direct policy transfer methods: Oßwald *et al.* [4] demonstrated direct transfer of motion navigation policy for Nao humanoid robot. Policy was learned in a precise simulator and then directly used on the real platform and it performed well. Christiano *et al.* [5] suggest learning an inverse dynamics model that can adjust actions from the simulator to execute in the real world as intended. They however require a way to transfer the real-world state into the simulator to execute their algorithm. Nemec *et al.* [6] used value function learned in simulation to bootstrap the real robot learning. We also initialize the policy from the simulator.

Model-based reinforcement learning methods learn simultaneously model and the policy. Since the model learned from the scratch on real trajectories is typically a fast differentiable function [7], [8], direct policy optimization is

Manuscript received February 24, 2018; accepted July 11, 2018. Date of publication July 20, 2018; date of current version August 8, 2018. This letter was recommended for publication by Associate Editor Prof. S. Oh and Editor Prof. D. Lee upon evaluation of the reviewers' comments. This work was supported in part by the European Union under Grant FP7-ICT-609763 TRADR, in part by the Czech Science Foundation under Grant GA17-08842S, in part by OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics", and in part by the Grant Agency of the CTU Prague under Project SGS18/138/OHK3/2T/13. *(Corresponding author: Martin Pecka.)*

The authors are with the Faculty of Electrical Engineering, Department of Cybernetics, Czech Institute of Informatics Robotics and Cybernetics, Czech Technical University in Prague, Prague 12135, Czech Republic (e-mail: peckama2@fel.cvut.cz; zimmerk@cmp.felk.cvut.cz; petrlmat@fel.cvut.cz; svobodat@fel.cvut.cz).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The Supplementary Materials contain two videos. This material is 28 MB in size.

Digital Object Identifier 10.1109/LRA.2018.2857927

often possible. But learning the motion and perception model from real trajectories (i) endangers the robot and (ii) requires prohibitively high number of trajectories. In contrast to these approaches, we already make use of a sophisticated motion model, and mainly focus on the perception transfer.

Data-driven refinement of perception simulator: The problem of transferring perception between different domains is well studied. In computer vision Generative Adversarial Nets [9] (GANs) have been recently used for generating synthetic training images. Shrivastava *et al.* [10] have shown significant performance boost if GANs are used to refine graphics-engine–based images. Similarly, we also refine simulator-generated data.

Guided policy search: In Guided Policy Search [11], guiding samples are utilized in a loop to guide direct policy search into areas of search space which yield the highest reward. However, it does not account for the reality gap between the simulated and real world, and it is impossible to run the algorithm directly on the real platform, since it requires too many samples.

A similar approach to our pipeline was tested by Bousmalis *et al.* [12] for grasping. They use (non-cycle) GAN to transform mostly static simulated images into the real domain, and then a deep network that benefits from the simulated data. In this work we show that using CycleGAN helps the domain transfer even more.

III. Pipeline Overview

Our pipeline follows three main assumptions: (i) the physics-based simulator is slow and analytically non-differentiable, (ii) simulation of the exteroceptive perception such as mapping from multi-modal sensor fusion is not realistic, and (iii) there exists an unknown generative model G which corrects the simulated perception to be close to the real perception. Under these assumptions, we search for control policy π^*, which minimizes the expected sum of traversal costs c of the real robot.

Let us denote p_r^π the probability distribution of trajectories $\tau_r = \{(\mathbf{x}_r^i, \mathbf{a}_r^i)\}_i$ generated by the real robot under policy π, and $p_s^\pi(G)$ the probability distribution of trajectories τ_s generated by the simulator with generative model G under policy π. Each trajectory $(\mathbf{x}^i, \mathbf{a}^i)$ is a sequence of state vectors $\mathbf{x}^i$ and action vectors $\mathbf{a}^i$. We search for policy

$$\pi^* = \arg\min_\pi \mathbb{E}_{\tau_r \sim p_r^\pi}\{c(\tau_r)\}. \tag{1}$$

Using assumption (iii), we rewrite the optimization problem using the simulator distribution $p_s^\pi(G)$ in the objective as follows

$$\arg\min_{\pi,G}\{\mathbb{E}_{\tau_s \sim p_s^\pi(G)}\{c(\tau_s)\} \mid \text{s.t. } p_s^\pi(G) = p_r^\pi\}. \tag{2}$$

Since trajectories collected with the simulator and with the real robot are unpaired, direct supervised training of the generative model is impossible. Consequently, we replace constraint $p_s^\pi(G) = p_r^\pi$ by the saddle point constraint on GAN-like loss $\mathcal{L}_{\text{GAN}}(G, D, \pi)$ induced under policy π

$$\arg\min_{\pi,G} \mathbb{E}_{\tau_s \sim p_s^\pi(G)}\{c(\tau_s)\}$$
$$\text{s.t. } G = \arg\min_{G'} \max_D \mathcal{L}_{\text{GAN}}(G', D, \pi), \tag{3}$$

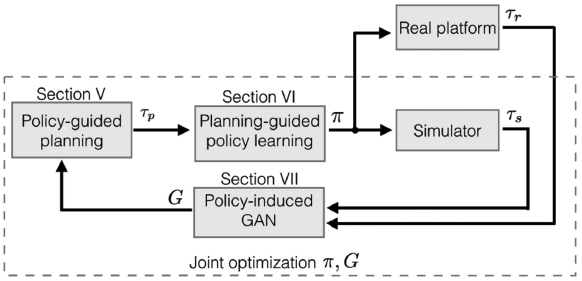


Fig. 1. **Proposed coarse-to-fine policy learning paradigm:** the coarse policy-guided motion planning is alternated with guided learning and policy transfer to the real robot.

where D denotes a discriminator.

If the GAN loss $\mathcal{L}_{\text{GAN}}(G, D, \tau_r, \tau_s)$ is pure GAN loss [9]

$$\mathbb{E}_{\tau_r \sim p_r^\pi} \log D(\tau_r) + \mathbb{E}_{\tau_s \sim p_s^\pi(G)} \log(1 - D(G(\tau_s))),$$

the saddle-point generator provides samples from the true distribution and the equivalence between eq. (2) and eq. (3) holds. In order to achieve fast convergence on the high-dimensional unpaired data, we use CycleGAN loss [13], therefore eq. (3) is an approximation of the original problem.

By assumption (i), any direct optimization of eq (3) is technically intractable. We propose approximated optimization scheme, which minimizes the interaction with the slow simulator and the real robot.

The optimization alternates between (i) planning guiding samples τ_p, which approximately optimize objective

$$\arg\min_{\tau_p'} \mathbb{E}_{\tau_p'}\{c(\tau_s)\}, \tag{4}$$

(ii) collecting real and simulated trajectories τ_r, τ_s, and (iii) searching for the control policy and the generative model which minimize the locally approximated criterion

$$J(\pi, G, \tau_p) = \sum_{(\mathbf{x},\mathbf{a}) \in \tau_p} \|\pi(G(\mathbf{x})) - \mathbf{a}\| \tag{5}$$

subject to locally approximated GAN loss $\mathcal{L}_{\text{GAN}}(G, D, \tau_r, \tau_s)$ around the collected trajectories τ_r, τ_s. The proposed pipeline is summarized in Figure 1 and Algorithm 1.

The generative model G^0 is initialized as identity. The initial policy π^0 is initialized by guided learning (i.e.,we plan initial trajectories τ_p and estimate $\pi^0 = \arg\min_\pi J(\pi, G^0, \tau_p)$). Given the initial policy, real trajectories are collected and alternated optimization (lines 3–8) with K iterations is performed. Finally, a new set of real test trajectories is collected and the whole process is repeated until a satisfactory behavior of the real robot is observed.

IV. Real Platform and Its Simulation Model

The real robot used in our experiments is the Absolem tracked vehicle used in Urban Search and Rescue scenarios [3], [14], which is depicted in Figure 3. It is equipped with a gyro providing its spatial orientation and with a rotating 2D lidar which provides full 3D laser scans at rate 0.3 Hz. The point map built from

Fig. 2. **Robot surmounting unstructured terrain during USAR mission.**

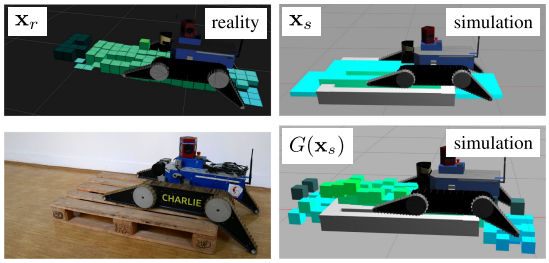


Fig. 3. **Real and simulated DEMs.** A visualization of Digital Elevation Maps (DEMs) is shown above. Dark green cells represent NaNs. **Top left:** DEM captured by the real platform. **Bottom left:** The real pose of the robot on an obstacle. **Top right:** DEM from simulator. The shapes are ideal and all measurements are available. **Bottom right:** DEM from simulator transformed by G to appear realistic.

lidar scans by the state-of-the-art SegMatch algorithm [15] is combined with high-precision track odometry in a multi-modal fusion pipeline [16].

For simulation, we use our custom tracked vehicle dynamics model implemented in the Gazebo simulator [3]. Parts of the simulation are randomized or pseudo-randomized (e.g., search of contact points of colliding bodies, solving of the underlying dynamics equations), so every execution of even a deterministic policy results in slightly different outcomes. This is useful for us, because our pipeline requires a multitude of different trajectories for every control policy. To achieve fast simulation, several simplifications were implemented in the simulated perception pipeline.

The most important of all policy inputs is the *Digital Elevation Map* (DEM) of close robot neighborhood (visualized in Figure 3). It is a horizontal 2D grid of rectangular cells where each cell contains information about the highest 3D point located in it. When there is no point measured inside a cell, a *Not-a-number* (NaN) value is stored. The DEM is treated in the coordinate frame of the robot with pitch and roll angles zeroed out. On the real robot, DEM is constructed from the point map. In simulation, DEM measurement is done in a completely different way to avoid inefficient laser ray-tracing: we directly extract the height of the highest object (excluding robot body) in each DEM cell, which is a fast operation. That means there are no missing measurements in the simulator DEM, and also no noise.

V. Generating Guiding Plans

The simulator is utilized by the path planner to sample trajectories τ_p^k, which are further used in the pipeline as described in Algorithm 1.

Algorithm 1: Overview of the Real Policy Learning.

1: **Initialize:** G^0 as identity and policy π^0.
2: **Collect** real trajectories $\tau_r \sim p_r^{\pi^0}$
3: **for** $k = 0 \dots K$ **do**
4: **Plan** guiding traj. τ_p^k biased by π^k (Section V).
5: **Optimize** policy w.r.t. new generator (Section VI)

$$\pi^{k+1} \leftarrow \arg\min_{\pi} J(\pi, G^k, \tau_p)$$

6: **Collect** simulated trajectories: $\tau_s^{k+1} \sim p_s^{\pi^{k+1}}(G^k)$
7: **Find** trajectory-consistent saddle point (Section VII)

$$G^{k+1} \leftarrow \arg\min_{G} \max_{D} \mathcal{L}_{\mathrm{GAN}}(G, D, \tau_r, \tau_{s^{k+1}})$$

8: **end for**
9: $G^0 \leftarrow G^K$, $\pi^0 \leftarrow \pi^K$ and **repeat** from line 2.

The planner works on a multitude of randomly generated worlds (*training worlds*) with different obstacles, corresponding approximately to the expected real obstacles. Each training world has a predefined length of trajectories the robot has to *safely* traverse to consider the trajectories *valid* (a time limit is also in place).

Different definitions of valid trajectories can be used; they are always closely related to the particular task. We utilize the fact that if the flippers are controlled incorrectly, the robot is not able to overcome obstacles and gets stuck or damaged. Safety of trajectories is given implicitly by several criteria like maximum allowed accelerations, limits on pitch and roll angles, and parts of the robot body which cannot touch any part of the environment.

Input of the planner consists of the training world specification and possibly also a *guiding policy* π. The task is to find a *valid* trajectory τ while keeping planned actions as close to actions of π as possible (if π is given).

The planner uses an RRT-based algorithm of state space search. Planning nodes capture the simulated DEM, robot orientation and flipper configuration. Each expansion of a planning node is evaluated in the simulator and a new planning node is created for the returned state.

Even though a standard RRT planner can find a solution by exploring the state space uniformly in all dimensions, in reality it is often impractically slow. In high-dimensional applications with costly expansion (as in our case), a heuristic must be employed to reduce the required iterations. Kinodynamic RRT* [17] is widely used to compute asymptotically optimal trajectories for robots with linear differential constraints. The method, however, assumes the knowledge of explicit motion model. Another general approach is to first find a discrete geometric path in a simplified search space and then optimize it by generating multiple trajectories with added noise [18] or by biasing the sampling of a guided RRT planner [1], [19], which is the method we use. A whole set of (different) trajectories is expected to satisfy our validity criterion, so methods targeting at getting close to a single optimal trajectory are not suitable.

Policy π is used as a guide by sorting the actions by their similarity to what π would do (we use L_2 norm, but any meaningful norm can be used). If π is not given, actions are selected randomly. Node expansion is realized by executing the action in simulator and checking the feasibility of the obtained node. The tree cannot be optimized by RRT* rewiring [20], due to the uncertainty introduced by executing an action, which prevents connecting any two nodes of the tree. Trajectories generated from the guided RRT are similar to trajectories sampled from the guiding policy, but many sampled trajectories can be invalid, and using the planner filters these automatically out.

An important property of the guiding approach is that with more planning–learning iterations, the plans will be closer to the subspace representable by the chosen policy class, which should in return result in better fit of future policies to future planned paths. The speedup gained by the guiding is utilized to enlarge the searched action space or refine the time resolution.

We propose to start the planning in a reduced action space which is practical to be explored without guiding, and once a guiding policy is available, the dimensionality can be increased. We start with 9 actions and time resolution of 1000 ms, further we add more actions, and last, we refine the time resolution to 200 ms, which is more suitable for real-world execution (but the plans need to be 5-times longer, which would be a significant increase in computation time without guiding).

VI. Guided Learning

With a set of trajectories generated by the path planner, the guided learning phase can start. Generally, it is possible to use any kind of supervised learning in this part. We chose a deep neural network that is crafted to make use both of the 2D structure of DEMs and to handle correctly *Not-a-Number* (NaN) values.

Inputs to the network are DEM, orientation of the robot and current flipper positions. Outputs of the network are the 4 desired flipper positions. Normally, if a NaN value would enter as a part of the DEM, it would silently spread further and could eventually end up in one of the outputs, which is undesirable.

A standard approach is to replace NaNs with a neutral value (like 0) or interpolate them. In Section VIII we show that these approaches yield worse results. Thus, we decided to treat the NaN values as "first-class citizen" because they can also carry useful information (the fact that a measurement is missing can have geometrical reasons).

We propose the following input processing: the DEM is converted into two matrices of the same shape—one with NaNs replaced by zeros, and the other with ones in measured cells and zeros in cells with NaNs (this part of architecture is shared with the GANs described in Section VII). Each of these matrices is fed into its own convolutional layer, and their outputs are multiplied. This effectively means normalizing each patch covered by a convolutional filter by the number of measured values in this patch. From this layer on, no NaN values are in the network, the output of the convolution is flattened, concatenated with the 1D inputs (robot orientation, flipper angles) and finally enters a fully connected layer, whose output are the four desired flipper angles.

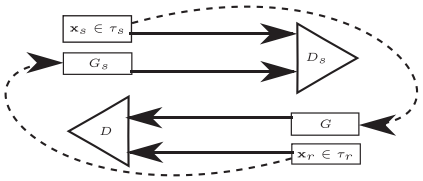


Fig. 4. **CycleGAN architecture.** Two GAN networks interconnected in such a way that input dimension of generator G_s is the same as output dimension of G and vice versa. The discriminators D_s and D serve both for evaluation of single generator loss and the cyclic loss.

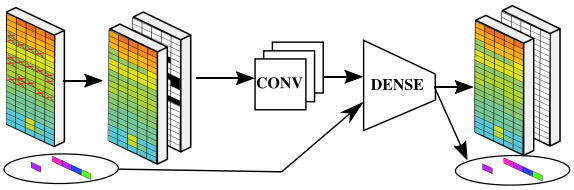


Fig. 5. **Generator architecture.** The raw input is preprocessed to yield a tensor of shape $21 \times 5 \times 2$ which is then used by the rest of the network.

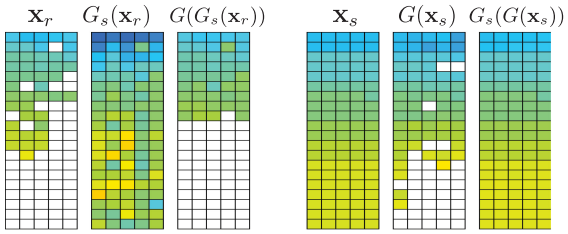


Fig. 6. **DEMs transformed by the generators.** Heights in the DEM: blue $= -1$ m, green $= 0$ m, red $= +1$ m, white $=$ NaN.

The regressor network is optimized using gradient descent to minimize the error between the predicted flipper target positions and those provided in the dataset. The dataset is randomly divided into training and test parts.

VII. Data Transformation via CGANs

The next key step is to find a suitable transformation between the data observed on the real platform and data observed in the simulator.

CycleGANs [13] were shown to be useful in the task of mutual mapping of two domains when only unpaired data are available. Specifically, Shrivastava *et al.* [10] used them to transform a simulated dataset to look real and then applied standard deep learning that expects real data at the inputs.

The mapping from simulated to real data is realized by generator G, while the opposite process is represented by generator G_s. The relation between the generators, their discriminators and input datasets is shown in Figure 4.

The input data with special structure ($20 \times 5\, 2D$ data possibly containing NaNs + 5 scalar constants), are preprocessed similar to Section VI. In generators and discriminators, the input DEM is transformed into a $20 \times 5 \times 2$ tensor where the first channel

contains the DEM with *NaNs* substituted with 0 s and the second channel contains a mask with -1 s at *NaN* cells in the DEM, and 1 s otherwise see Figures 5 and 6 for details.

The scalar inputs (robot orientation and flipper angles) skip these first convolution layers and enter the network later as inputs to a fully connected layer. At the output, the DEM and the scalar values are again separated. This allows the network to work as a standard image-to-image CycleGAN, but also allows it to use the scalar information.

The internal structure of the generators and discriminators contains several convolution layers that use the Leaky ReLU activation function, and a final fully-connected layer.

Our pipeline suggests that the generators should be initialized to identity, which is not generally possible with neural networks containing non-linear activation functions. However, implementing a skip-connection of the input data directly to the fully-connected layer allows this initialization. Identity should be a good initial guess for the generator, because we do not want it to change the data too much.

Both discriminators use the pure GAN loss formulation (see Section III).

Loss function of both generators is defined by their corresponding discriminator (D for generator G; D_s for generator G_s):

$$\mathcal{L}_G(\mathbf{x}) = +\lambda \cdot \sum(\log(D(\mathbf{x})) + \lambda_p \cdot \sum_i ||\mathbf{x}_i - G(\mathbf{x}_i)||$$

We penalize distance of the generated output from the inputs (pixel-wise), as it was shown to stabilize the learning [10]. One additional component of $\mathcal{L}_{G_s}$ can be added that penalizes any NaN values in the output, since we know there are no NaNs in the simulator DEMs.

The cycle loss $\mathcal{L}_c(G, D, \mathbf{x}_r, G_s, D_s, \mathbf{x}_s)$ is defined as

$$\mathcal{L}_D(G(G_s(\mathbf{x}_r))) + \mathcal{L}_{D_s}(G_s(G(\mathbf{x}_s)))$$

Training of the network is done by repeated optimization of all generator and discriminator losses, where $\lambda_c \cdot \mathcal{L}_{cycle}$ is added to the loss of both generators. The training is done on simulated data from τ_s^k and real data from τ_r.

It is usually difficult to tell when to stop GAN training. Although it is not required for the training itself, we constructed a small validation dataset consisting of pairs of data from the simulator and their closest counterparts encountered in the real data. If it is possible to collect more such correspondences, a part of them can be added to the learning process via L_2 loss on these samples. In our tests, adding the correspondences further helped training the GAN, but care must be taken to not overfit the network to the correspondences.

VIII. Experiments

Experimental evaluation of the learned policies is an essential part of the learning loop. After several iterations of the learning, planning and generator optimization, verification in the real world is to be performed.

For the task of terrain traversal with a tracked robot, we designed a real test scenario consisting of flat ground, a pallet and a staircase, which are typical obstacles the robot can encounter. The staircase is subdivided to 6 sections with different characteristics – approach to stairs, on stairs, leaving stairs, and the stairs can go either upwards or downwards. The staircase is traversed with constant forward speed 0.3 m/s three times and the pallet 10 times, resulting in execution of 13 trajectories. Every trajectory is assigned one of three success levels – *good* in case the trajectory was without problems, the robot passed and did not endanger itself; *unclear* if there were minor problems during the execution, but the robot traversed the whole required length (e.g., behavior close to unsafe, the operator had to reduce the otherwise constant travel speed, and so on); finally *fail* level is assigned to trajectories that the robot could not finish or executed an unsafe action. These levels carry numerical value (*good* $=$ 1.0, *unclear* $=$ 0.5, *fail* $=$ 0.0) and policy performance is an average of these values over all executions.

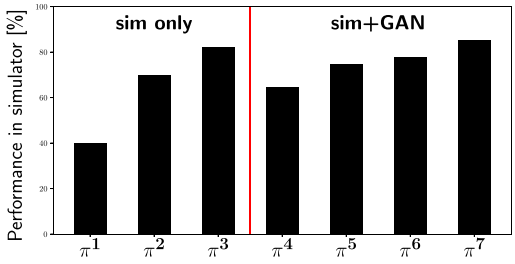


Fig. 7. **Average policy performance in simulated worlds.** Performance of 100% means traversing all test worlds in a safe manner.

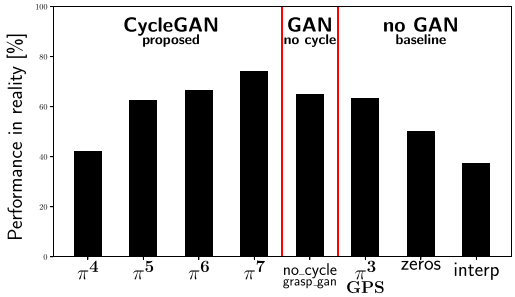


Fig. 8. **Average policy performance in real world.**

Similar obstacles were modeled in the simulator and a set of 8 test worlds was created. The metrics for simulation is proportion of *good* trajectories among all executed. Here *good* means traversing the required length of the trajectory with constant speed 0.3 m/s without executing any unsafe actions (as described in Section IV).

Results of the learning process are summarized in Figures 7 and 8. First, 3 iterations (policies π^1–π^3) were using only the simulator without GAN for adjusting perception. Further simulator-only iterations showed little performance improvement, so we assume the process converged at π^3. Policy π^3 is similar to what Guided Policy Search [11] with Adaptive Guiding Samples would find, so we also call it GPS (we use a different guiding sample generator – RRT instead of DDP, and the RRT planner automatically generates adaptive samples by prioritizing actions similar to the policy decisions).

TABLE I
PATH-PLANNING PERFORMANCE

It.	Guided	GAN	# actions	Δt	Visited nodes	Avg. CPU time
1	×	×	9	1000 ms	116 ± 60	8 min
2	✓	×	9	1000 ms	103 ± 55	5 min
3	✓	×	9	1000 ms	102 ± 54	5 min
4	✓	×	**49**	1000 ms	138 ± 82	15 min
5	✓	**✓**	49	1000 ms	238 ± 13	17 min
6	✓	✓	49	**200 ms**	1239 ± 811	40 min
7	✓	✓	49	200 ms	924 ± 497	35 min
-	×	✓	9	200 ms	-	$\gg$60 min

CPU-core–time and number of visited nodes needed to sample one trajectory by the path planner. Δt is time resolution (i.e.,with $\Delta t = 200$ a trajectory of some defined metric length needs $5\times$ more nodes than with $\Delta t = 1000$). Bold values highlight changes between iterations.

Unfortunately, real-world trajectories cannot be used as guiding samples in GPS, because the simulated and real domains differ too much for the learning to converge.

Testing in real world started in the fourth iteration. Two of the best policies found in simulator were tested in real world and the better one became π^{k+1}.

We cut off the whole pipeline once the policy achieved good performance in the real world (after 7 iterations). That accounts for ca 15 minutes of driving with the real robot to collect the initial τ_r, then 4×13 trajectories for real-world policy verification, which is about 20 minutes. No more real-world execution was needed.*

To see the benefits of our pipeline, we tested running π^3 aka GPS (the best simulator-only policy) directly on the real robot. The performance was, as expected, poor. We also trained two baseline policies ($zeros$ and $interp$) which either zero-out or bi-linearly interpolate the missing values (NaNs) in real data. These policies can have a simpler structure (the second channel for NaNs is removed). They were trained on the same trajectories π^4 was trained on. None of these policies managed to outperform the proposed pipeline. Last, we also tested the importance of the cycle loss in GANs. Policy no_cycle was trained on a dataset transformed by a GAN that was trained without the cycle loss, similar to GraspGAN [12], so we also call it $grasp\_gan$. Validation error of the GAN (as mentioned at the end of Section VII) was about 12% higher than with cycle loss, and performance of the no_cycle policy did also not beat the proposed pipeline.

To train final policy π^7 from scratch, we needed 800 CPU-core–hours (of which 90% is spent on performance verification, which could be lowered) and 50 GPU-hours (highly depends on structures of the policy and GANs).

We also experimentally verified that guiding decreases path-planning time or allows to plan paths in larger action spaces or with longer planning horizon. A summary of computation times is shown in Table I. We also tried unguided planning with 200 ms resolution, but no path was found in one hour.

*See the attached video with policy tests, or http://cmp.felk.cvut.cz/∼peckama2/policy_transfer/ for more information and FullHD video.

IX. Conclusion and Future Work

We have proposed and experimentally evaluated the new self-contained learning–planning–transfer loop, which employs a simulator of robot–terrain interactions. The proposed method simultaneously learned the policy in simulation and transferred it to the real robot. The transfer was achieved by a generative model which corrected imprecisely simulated perception. The experimental evaluation showed that iterations of the learning–planning–transfer loop improve performance of the policy on the real robot. We also showed that it is possible to further refine the action space of guiding policies without compromising computational tractability.

Our ongoing research will focus on possibilities of making the CycleGAN learning policy-aware, so that the generators are trained with policy performance in mind.

References

[1] D. Ferguson and A. Stentz, "Anytime RRTs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 5369–5375.

[2] J. Kober, "Learning motor skills: From algorithms to robot experiments," in, *Information Technology*. New York, NY, USA: Springer, 2014.

[3] M. Pecka, K. Zimmermann, and T. Svoboda, "Fast simulation of vehicles with non-deformable tracks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 6414–6419.

[4] S. Oswald, A. Hornung, and M. Bennewitz, "Learning reliable and efficient navigation with a humanoid," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2010, pp. 2375–2380.

[5] P. Christiano *et al.*, "Transfer from simulation to real world through learning deep inverse dynamics model," arXiv:1610.03518, Oct. 2016.

[6] B. Nemec, M. Zorko, and L. Zlajpah, "Learning of a ball-in-a-cup playing robot," in *Proc. 19th Int. Workshop Robot. Alpe-Adria-Danube Region Robot.*, 2010, pp. 297–301.

[7] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," in *Proc. IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 408–423, Feb. 2015.

[8] R. Tedrake, "LQR-trees: Feedback motion planning on sparse randomized trees," in *Proc. Robot., Sci. Syst.*, 2010.

[9] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[10] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," *Comput. Vision and Pattern Recognition*, 2017.

[11] S. Levine and V. Koltun, "Guided policy search," in *Proc. 30th Int. Conf. Mach. Learn.*, vol. 28, no. 3, 2013, pp. 1–9.

[12] K. Bousmalis *et al.*, "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping," in *Proc. Int. Conf. on Robot. and Autom.*, Sep. 2018.

[13] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. Int. Conf. on Comput. Vision*, 2017.

[14] I. Kruijff *et al.*, "Designing, developing, and deploying systems to support human-robot teams in disaster response," *Adv. Robot.*, vol. 28, no. 23, pp. 1547–1570, 2014.

[15] R. Dube, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "SegMatch: Segment based place recognition in 3D point clouds," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 5266–5272.

[16] J. Simanek, M. Reinstein, and V. Kubelka, "Evaluation of the EKF-based estimation architectures for data fusion in mobile robots," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 2, pp. 985–990, Apr. 2015.

[17] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Optimal motion planning for systems with linear differential constraints," in *Proc. Int. Conf. on Robot. and Autom.*, 2013.

[18] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 4569–4574.

[19] V. Vonasek, J. Faigl, T. Krajnik, and L. Preucil, "RRT-Path: A guided rapidly exploring random tree," in *Robot Motion and Control*, Berlin, Germany: Springer, 2009, pp. 307–316.

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, pp. 846–894, 2011.

Chapter 15

Followup and Related Work

Domain transfer is a difficult task, which shows to be of ultimate importance for real-world robotics. If done correctly, it would allow to utilize the much faster and "cheaper" simulators to train optimal behaviors on real robots. Since GANs first appeared in the work of [Goodfellow et al., 2014] and CycleGANs in [Zhu et al., 2017], they quickly found their way to domain transfer research. [Tzeng et al., 2015b] present a learning scheme which does not exactly employ GANs, but uses the same ideas of training a "generator" (in this case a labeling algorithm), and a discriminator which forces the learning process to discover representations where the old and new domain are indistinguishable. Surprisingly, the proposed algorithm is able to transfer labels from the old domain to a new domain with only a few or even no labels in the new domain (if the labels in both domains have similar and nonuniform distributions). [Tzeng et al., 2015a] similarly show that if we can generate similarly distributed data from two domains, an unsupervised or weakly supervised algorithm can align the distributions and transfer any labeling from one domain to the other. [Christiano et al., 2016] choose an approach that could be said to be opposite to what most other domain transfer works do. They do not train a forward mapping from simulated states to real states—instead, they train an inverse dynamics model for the real robot. They can then use the simulator to extrapolate the state the robot would like to reach, and use the inverse dynamics to compute the real action that would take the robot to that state. This approach however assumes that the real robot state would never diverge too much from the simulation, or that we can run the simulator on the real state observations. This might be possible when no or simple exteroceptive sensors are used, but will probably never work in tasks with complex and imperfect exteroception. Convergence of GANs is generally a fragile problem, where a lot of different factors can play an important role and values of many hyperparameters have to be set correctly. [Wulfmeier et al., 2017] show a thorough study of the influence of various hyperparameters and design decisions on GAN convergence. For RGB images, [Tobin et al., 2017] show that randomizing the textures in a simulator is a possible way to do domain transfer to images captured by a real camera attached to a robot. When the geometrical properties of the two domains remain close enough, and texture is the only difference,

this algorithm learns to ignore the texture and instead learn only on the geometrical features. The real images can then be treated only as a sample of random textures in the simulator. Not only images, but also system dynamics can be randomized, as in [Peng et al., 2018b]. They assume that there is a known approximate model of the system with unknown parameters, and that given optimal parameters, we can simulate the real dynamics up to approximation error. In an Actor-Critic setting, they train a policy against an *omniscient critic* estimating the value function with the knowledge of the dynamics parameters (which are known in simulation). Once trained, only the actor network is used for execution, and the dynamics parameters are thus no longer needed. If parameters of the real system are from a similar distribution to the one the training dynamics samples were drawn from, the learned actor policy should perform well when deployed on the real system, without a single real trajectory recorded during the learning. By training on various dynamics models, the policy should learn to use mainly dynamics-independent features. The dynamics randomization approach is extended in [Muratore et al., 2018] and [Muratore et al., 2019]. They add a measure estimating how good is the current estimate of the objective function, and use this measure as a stopping criterion for learning to prevent overfitting the policy to simulation imperfections or specifics. [Peng et al., 2018a] choose an opposite direction for domain transfer than what is usual in RL. They start with motion capture data of real people and try to learn a simulated policy replicating these motions in various conditions. They verify that experience replay (mentioned further in this chapter) is crucial for the success of learning in high-dimensional spaces. [Fankhauser, 2018] shows in his thesis that it is possible to hand-tune very efficient control policies directly on a real robot. The terrain-adaptive quadruped gait control is definitively a very complex and high-dimensional task. The thesis shows that no learning is required if there is enough time to fine-tune lots of parameters and solve various special cases. However, [Hwangbo et al., 2019] shows that using deep learning, a large part of the hand-tuning can be overcome. They present a terrain-ignorant control policy trained by TRPO that achieves even higher performance on the real robot than any other manually-created policy tried on the robot. Of course, the gap between terrain-ignorant and terrain-adaptive policies is huge, as the state space in the latter is usually several orders of magnitude larger. [Peng et al., 2019] propose an improvement for GAN training that imposes an information-throughput limit between the generator and the discriminator. By limiting the amount of information passed to the discriminator, they make sure it never gets perfect, which has bad consequences for generator gradient estimation. [Kang et al., 2019] propose to learn visual-based control by training a deep Q-function predictor from simulated images, and copying the convolutional layers to a reward estimator network. This network is then trained on real data with weights of the copied perception layers being frozen. The input of the reward estimator is a sequence of past images and a sequence of planned actions up to some horizon. It outputs reward predictions for each of the planned actions. A simple policy is based on the reward estimator—

select such action that maximizes future rewards. A counter-intuitive result is that fine-tuning also the convolutional layers on real data does not help the policy performance. Another approach to domain transfer shown in [Vemula et al., 2020] is to recognize the actions whose outcomes differ too much in the old and new domain, and discourage the policy search algorithm from using such actions. If there exists a policy using only the well-simulated actions, the real-world performance can be expected to be satisfying. [Chen et al., 2020] solve the generally hard problem of evaluating the quality of CycleGAN output. The most used Inception Score and Frechet Inception Distance can only be used on a special type of photos of people and real-world objects. For the task of map quality evaluation, the authors proposed a method based on edge similarity.

For some problems, such as the one presented in this part, it is possible to use some non-learning–based methods to generate a few good trajectories. These methods, like planning, are usually too slow to be computed during policy execution, but may prove extremely useful for the learning phase. Guided Policy Search (GPS) [Levine and Koltun, 2013] is an algorithm that can leverage a few initial trajectories during the policy search via importance sampling. [Montgomery and Levine, 2016] improve this algorithm with a different optimization procedure with automatic estimation of the step size of gradient descent. This way, the algorithm achieves higher quality results while having less hyperparameters to tune compared to GPS. [Montgomery et al., 2017] remove the requirement to preform resets from the basic GPS algorithm. That means the algorithm can be run on-policy and is better suited for running on real robots. The computation cost is higher, though, because an E-M step needs to be performed that assigns historical trajectories to the most likely current local policies. [Mendonca et al., 2019] introduce Guided Meta Policy Search, which is a sample-efficient meta learning algorithm based on GPS. In the meta-learning setting, they first pretrain the meta policy on the demonstrations, which speeds up learning. Then they employ a classical phase of learning multiple guiding policies which further help to optimize the meta objective.

Deep and convolutional neural networks are a very powerful tool for function approximation. [Sünderhauf et al., 2018] overview the problems that arise when deploying deep learning algorithms in robotics. Sample complexity is one of the major problems that needs to be tackled before deep learning successfully settles as a learning technique for robotics. One of the possible ways to reduce the number of required samples is reducing the dimensionality of the states or actions they work with. [Luck et al., 2014] use a modified variant of Principal Component Analysis for finding an efficient embedding of the high-dimensional action space of a humanoid robot. This dimensionality reduction allows running policy search in the much smaller latent space. Thanks to that, the policy search algorithm gets practical even for experiments with real robots. [Munk et al., 2016] alter the classical DDPG Actor-Critic algorithm by adding a third network which learns a mapping from the observations to some low-dimensional latent states. These latent states are

used as input to the actor and critic networks. This approach is shown to improve convergence, and in some cases it also results in better performance than plain DDPG. [Peng et al., 2016] utilize high-level actions similar to Motion Primitives (but non-periodic) to decrease the dimensionality of action space. [Parisi et al., 2017] apply PCA to state observations and use it to extract only task-relevant features from the observations. It is shown to speed up the learning process and sometimes even leads to better policies. [Sermanet et al., 2018] show an algorithm that can extract low-dimensional and view-independent state representation from series of time-synchronized views of a robot. The approach is completely unsupervised and is able to learn spatio-temporal relations just by observing the robot from multiple viewpoints at the same time. Stochastic Latent Actor-Critic algorithm presented in [Lee et al., 2019] also reduces problem dimensionality by converting it to a latent space. This method uses a Bayesian definition of the latent space and the resulting algorithm is fully stochastic and Markovian, with no recurrent cells. It is shown to find better policies than standard SAC. Another algorithm reducing dimensionality via a latent space is [Vezzani et al., 2019]. They learn the latent space representation using a multi-headed network with a shared part which acts as the latent space encoder. Together with motivating the policy to keep high entropy and stay close to high-reward areas, this approach is shown to outperform TRPO in the performance of the resulting policies. [Zhang et al., 2019a] show a very efficient algorithm that can be used with quadratic costs, which employs learning such latent state representation which provides the best performance for evaluating simple local models of dynamics. This makes the dynamics models more precise and better suited for the MPC that is a core part of the algorithm. In the end, the algorithm is able to learn a block stacking problem using a robotic arm only from images within 2 hours of robot run time. [Wu et al., 2019] reduce the dimensionality of a pick-and-place problem by applying structure on the actions. They subdivide them to subactions “pick” and “place” and establish some relations between these parts. Such structure improves learning speed and accuracy. Another interesting way to reduce dimensionality of the problem is to directly turn the original MDP into an abstract MDP with fewer states and actions, as is done in [Zun Yuan et al., 2019]. They expect that the task to learn is given as a temporal logic formula and that only state transitions relevant to this formula are important for the policy. In connection with DDPG, this shows to be a very efficient modular RL method.

Another way to reduce the number of samples that are required for reinforcement learning algorithm convergence is to incorporate prior knowledge, either from the same, or from a similar domain. [Cutler and How, 2015] train a Gaussian Process modeling the unknown dynamics of a robot in simulation (which is usually considered cheap), and use this model to speed up policy search on the real system. Contextual learning is another way to reuse data from previous experience. [Rusu et al., 2016] present the Progressive Neural Networks which are a kind of compositional neural networks. For each task (context sample), there is a “column” of the neural network, which is trained

and optimized for the given task, and once it is finished, the weights in the "column" are frozen and a new "column" is added for a new task. However, all outputs of all layers from the previous "columns" are available to the newly added one, so that it can efficiently reuse what the network has already learned. [Kahn et al., 2017a] use a special type of neural network with dropouts to predict uncertainty in training data. This network, similarly to Gaussian Processes, is able to output a mean and uncertainty of a function estimate. In this particular case, they use this network as a collision predictor for autonomous flight. They use the uncertainty to guide the policy in areas which need more real data, but also modify the behavior to a safer one in these dangerous zones. In the field of meta-learning, MAML algorithm [Finn et al., 2017a] became very popular. Given a differentiable model and objective, it trains a control policy which "ends up" one or few gradient steps from the optimum. It is trained on data from multiple domains and the remaining gradient steps represent adaptation to a specific domain. The adaptation is very sample-efficient, assuming the new environment is sampled from the same distribution used during training. [Marco et al., 2017] show a framework how simulated and real samples can be incorporated into a single learning system, taking into account the fact that obtaining the real samples is more costly than running simulation. The main idea is to alter a Gaussian Process learning the system dynamics so that it learns the simulator dynamics as one part and the differences of the real behavior as a second part. However, the algorithm probably does not scale very well to larger state or action spaces. [Martínez-Tenor et al., 2018] show that previously collected information about the Q function can be used to bias exploration. When selecting an action in a standard Q-learning setting, a bias term is added, which specifies that similar actions should be taken for similar states. They also suggest a short history of actions should be kept to asses if the agent did not enter a low-reward loop. If so, the action-selection algorithm should be allowed to take more diverse actions. [Sun et al., 2018a] show a framework which can interpolate between imitation learning and "pure" reinforcement learning. They expose a parameter that specifies the length of a horizon of a cost-to-go estimator, which is initialized from demonstrations. In this framework, the user can select whether it is more important to keep the sample complexity low, or if better policies are desired. [Lempitsky et al., 2018] show an interesting property of deep neural networks. In image recognition tasks, the largest part of the "success" of the network lies in the structure of its connections, not in the learned weights themselves. [Finn et al., 2017b] present an algorithm for efficient learning on a distribution of MDPs. They expect that they have labels (rewards) available for some of the MDP samples, and use these to initialize a reward predictor for the unlabeled MDPs. An EM loop then interleaves policy optimization and reward predictor improvement. [Stark et al., 2019] show a way to utilize already learned skills (probabilistic motion primitives) for initialization of primitives for new tasks. The closest already known skill is used as prior in search for the optimal solution for the new task. If covariance of the closest skill is also to be transferred, it needs to

be scaled by a hand-tuned factor to mitigate task bias. Prior knowledge can also be provided in terms of known reward function. [Kahn et al., 2020] show that for an end-to-end visual driving task, automatic state annotations allow for generating datasets of size impractical for manual labeling. But if a GPS sensor can tell whether the robot follows the given path and an IMU can tell whether it crashed into an obstacle, the annotations can be extracted automatically and the otherwise difficult RL task is converted to supervised learning. This method, however, needs to visit unsafe states (often) to learn about them, which is usually not an acceptable requirement.

Except for incorporating prior knowledge or reducing the search space, there are other ways to reduce the required number of samples in Deep RL. [Clavera et al., 2018] train not one, but several dynamics models in an ensemble, which can provide more precise modeling of dynamics of the system. If the reward function is a priori known, a meta-learning algorithm can be employed to find a policy for the given RL problem with unknown learned dynamics. This approach is shown to require significantly less training samples than other model-free Deep RL methods (however still impractically much for training on real robots). [Kahn et al., 2018] use a deep recurrent network with LSTM cells to model the unknown dynamics, and perform the reinforcement learning in a generalized computation graph which allows for interpolating between model-free and model-based learning. [Tosatto et al., 2019] found a method similar to Intrinsic Motivation methods that speeds up convergence by guiding the agent to high reward areas. An ensemble of Q-function estimators provides uncertainty of the estimate, and this uncertainty is then added to the objective function to motivate the agent to explore uncertain areas. [Zhang et al., 2019b] show a way how to improve the run-time of a learning algorithm by parallellizing the usual "collect data–update model–update policy" loop used in model-free learning. [Tosatto et al., 2020] present a new off-policy learning formulation via *Nonparametric Bellmann Equation* for which they compute analytical closed-form gradient. Using this gradient, lower variance is achieved than with standard policy gradient methods based on Bellmann Equation. Although the approach does not scale to large search spaces, on the smaller ones it converges with an order of magnitude less samples than state-of-the art off-policy algorithms like DDPG.

Reinforcement learning works with the assumption that the underlying system is an MDP (i.e. outcomes of an action depend solely on the current state, not on the previous states and actions). This requirement is not satisfied when first looking at some tasks, e.g. Atari games do not capture all relevant information in a single screen frame (e.g. velocity of objects is not encoded in the frame). Fortunately, it is possible to make some tasks RL-compatible by expanding the state space, e.g. stacking multiple input frames together; it is also possible to use networks with memory. [Hausknecht and Stone, 2015] take a Deep Q Network and substitute its first layer with LSTM cells. This way, they incorporate memory in the network, and it is able to solve many games which are normally considered Partially Observable MDPs. This comes at the cost that it is not possible to restart the inference

from an intermediate state and immediately expect optimal performance. [van Hoof et al., 2017] examined the time-series dependencies in RL and found a generalized policy update formulation that allows to interpolate between time-independent (MDP) and time-series (non-Markovian) formulations.

Sometimes it is difficult to craft the reward function for reinforcement learning so that the learned policies do exactly what was expected. It can be much easier to just show demonstrations of what we consider a good behavior, and let Inverse Reinforcement Learning deduce the correct reward function. [Finn et al., 2016a] present an Inverse Optimal Control algorithm that takes human demonstrations as input and estimates the distribution from which they are sampled. Based on the estimate of this distribution, they can also estimate the reward function. At the same time, they search for a policy that optimizes the currently estimated objective. [Ewerton et al., 2016a] show a way to incorporate "online" teacher feedback into a learning scheme for Probabilistic Motion Primitives (ProMPs). The algorithm assumes there is a robot (manipulator) with whose actions the human teacher can interfere while they are executed and "override" them with the teacher behavior. This way, the robot can learn the differences between the policy behavior and teacher behavior, and update the MPs to be closer to the teacher feedback. In [Ewerton et al., 2016b], they improve the basic definition of ProMPs to allow multiple phase parameters. An EM-like algorithm can then be used to decompose the motion into shape, phase and amplitude parameters. [Nair et al., 2018] assume that even the teacher trajectories can be suboptimal. They extend the classical DDPG algorithm with another term in the objective, which penalizes difference of the learned behavior from the demonstrations, but only in case the learned behavior is worse than the demonstrations. This way, the exploration can find policies which are substantially better than the teacher's policy. For learning from demonstrations, it is also crucial to a have a good measure of similarity of trajectories. [Urain and Peters, 2019] improve the standard ℓ_2 norm comparison by developing a correlation coefficient that is invariant to linear transformations. It would be although even more helpful to develop a measure invariant also to affine transforms and time-stretching (if that is suitable for the given task).

Many published algorithms stress the point that Reinforcement Learning is more efficient and converges better if the reward function is not used as is, but is instead subtracted from some baseline. A variant of this approach is using the *advantage function* instead of the reward function for policy search. [Schulman et al., 2016] and [Schulman, 2016] propose a Generalized Advantage Estimator network that improves the efficiency of TRPO model-free algorithm and takes its sample complexity closer to model-based learning. However, the experiments with humanoids would still need weeks of data if they were applied to real robots. So the sample complexity becomes only impractical for real-world experiments, not completely out of scope. [Sun et al., 2018b] present the Dual Policy Iteration algorithm, which utilizes two polices—a fast and reactive approximator, and a slower policy utilizing model-based optimal control. The slow policy can get better insights into the

direction where high-reward areas are, but is too slow to be used in practice. So it is only used during learning to guide the faster policy. This approach achieves faster convergence than plain TRPO on the class of problems with smooth dynamics and reward function.

Deep learning algorithms that require vast amount of samples suffer from the so-called *catastrophic forgetting*. If the optimized policy is used for exploration at the same time, it can happen that policy search finds an area with relatively high rewards and starts generating trajectories which are only close to this area. But if there was a similar, maybe only a little bit worse, area, the policy will stop visiting it so often, which makes the weight of errors in this area lower. This way, it can happen that the policy search algorithm completely forgets how to act in this slightly worse area. This forgetting is usually prevented by using replay buffers, which store historical samples and require the policy to take them into account, too. But as literature shows, the naive FIFO implementation of the replay buffers is not the best strategy. [Andrychowicz et al., 2017] propose a variant of replay buffers for contextual/meta RL called Hindsight Experience Replay (HER). The idea is that with sparse rewards, it might be very difficult to find even the first trajectory that gets some positive reward. And until then, standard RL has no prior about which directions might be promising. HER tackles this problem by extending the replay buffer not only with the context-trajectory-reward tuples that were actually executed, but also with tuples that have the same trajectory, but change the context and reward in such a way that it looks like the algorithm actually intended to execute precisely the given trajectory. This way the learning process becomes similar to curriculum learning, where the agent first learns how to perform simpler tasks, and the knowledge gained during learning these smaller steps helps it to finally reach the original goal. According to [Ren et al., 2018], the choice of samples going into a replay buffer is also important. They propose a curriculum-learning–like algorithm which prioritizes samples with low TD error, which should be easier to learn and correct in the next iteration. Another view on HER is provided in [Li et al., 2020]. They generate the false experiences by finding tasks that suit them more than the original task, but in dense rewards setting, where HER does not normally help. The false experiences can either be found using inverse RL or searching for most similar trajectories in a memory buffer. If that is computationally demanding, another alternative is to relabel the false experience with the task for which it has the highest value of the advantage function.

Recent research shows that keeping some entropy in stochastic policies is beneficial for learning. The Soft Actor-Critic algorithm [Haarnoja et al., 2017] shows convergence rate speed-ups of an order of magnitude compared to other model-free Deep RL methods—by requiring that the policy should keep some amount of entropy even at convergence. [Haarnoja et al., 2019] show an experiment where they used Soft Actor-Critic to learn a control policy on a small quadruped within 2 hours of real experiment time. [Akrour et al., 2018] and [Akrour et al., 2019a] provide a policy search algorithm

which intentionally limits not only the KL-divergence from the previous policy, but also entropy loss. They show an approach that can solve the constrained optimization problem without the use of Lagrange multipliers. This approach seems to be more suitable for the cases when the number of samples is very limited. [Parisi et al., 2019b] verified a similar effect for tabular RL. They not only search for the optimal control policy, but also for a different exploration policy based on accumulated visitation counts. Such algorithm was shown to perform better in case the rewards are sparse. [Parisi et al., 2019a] observe that in Actor-Critic methods, high variance of the critic output is destabilizing the interaction between the actor and the critic. They suggest adding a TD-error–related regularizer to the actor network objective function, which effectively decreases the learning rate if the critic is unsure. The method is general and can be used to improve the stability and convergence rate of almost any actor-critic algorithm.

[Tamar et al., 2017] use neural networks different from most of the mentioned algorithms. They encode the Value Iteration policy search algorithm itself into a differentiable neural network, which can then be trained by standard gradient descent. For small discrete problems, it shows better generalization, but it does not scale well to larger or continuous problems. [Shen et al., 2019] present a meta policy search algorithm that does not use gradients for optimization, but employs Evolution Strategies instead. This approach might be suitable for cases where the gradient is difficult to obtain or known to not exist. [Nagabandi et al., 2019] show an unusual meta RL setting—it does not consist of a task distribution and a multilevel policy. Instead, a sequence of last states is taken as the task, and a dynamics model is trained to predict the following sequence of states. This model is then used to perform supervised learning and online adaptation of the meta policy. Such algorithm can learn very robust policies, which are able to adapt in a few seconds to severe changes, e.g. disabling one leg on a quadruped robot. [Rothfuss et al., 2019] found a new formulation of objective for meta-learning called *Low-Variance Curvature* which is shown to have lower variance than standard meta-learning objectives. This naturally speeds up convergence of the learning.

Chapter 16

Conclusion

In this part, one possible way of learning and transferring policies from simulation to reality was presented. Although the experimental results show the method as promising, some pieces of the algorithm still need more attention. First, the GAN training process is fragile, and there are no clear guides on how to enforce (or even measure) convergence for non-photographic data. Human input is thus needed to decide when to stop learning and how to tune the hyperparameters. Also, the design of the neural network has so far been an arbitrary process depending on human intuition. We think that closer connection between the GAN and the policy could be a way to improve the final policy performance on real-world data.

The reviewed literature shows that domain transfer for visuomotor tasks for manipulators has already been solved to a degree which allows practical use. Also transfer between photographic domains (e.g. ImageNet-based) shows very impressive results. But neither the general problem of transfer learning, nor the specific problem of 3D sensing transfer from simulators to reality, have been solved with a generally applicable algorithm. Generative Adversarial Networks (used in domain transfer algorithms) also still suffer from the non-existence of good convergence measures.

One of the most important discoveries in deep learning was experience reuse. Starting with simple replay buffers, more and more insight has been gathered into various aspects like replay buffer management or Hindsight Experience Replay. Information throughput limiting shows to be another helpful technique. Many implementations are available, from autoencoders to direct information-throughput limiting, but all head towards a common goal—force the neural network to find efficient features and data representations.

Last, guiding or prior knowledge utilization seem to be the way to make policy search for real robots tractable. The standard deep policy search algorithms require prohibitively many samples, and thus techniques for utilizing already collected data are needed. There are many approaches—contextual policy search, inverse RL, meta-policy search or just utilization of learned dynamics models.

Part IV

Conclusions

Conclusion

In this thesis, we have shown several approaches that connect Reinforcement Learning and safety in different ways. Classical risk-ignorant Reinforcement Learning has achieved many successes both in simulated and real-world tasks. Recently, deep learning helped RL to succeed in even more areas. However, mobile robotics has not yet benefited from this connection fully. Two main obstacles lie in the way to successful deep RL—the number of samples required for learning and safety of the learning process (because real robots are vulnerable). This thesis presents theory and algorithms showing how these gaps could be closed or at least made smaller.

In the first part, recent approaches to safety in RL are reviewed. Further, the Constrained REPS algorithm is presented. In small continuous state and action spaces, it efficiently performs policy search under given linear constraints. These constraints are general and can encode for example safety of trajectories or other limitations of the real system. The learning process can thus be considered risk-aware, but it is not forced to be risk-averse (as is often the case in other algorithms). As long as the search distribution maintains the given safety constraints, it is free to explore any area. The constraints also guide the exploration process to feasible areas, which decreases the number of required samples. As was shown in the experiments, this method requires high-capacity function approximators to work well. Also, scaling to high-dimensional states such as images is not feasible with this method.

The second part is focused on handling the imperfect sensory data robots have to work with. It stresses the important (but often neglected) idea that missing measurements are as valuable as the actually measured values. We have shown a RL algorithm that utilizes the missing measurements to assess if its planned actions will be safe, or if more measurements need to be taken. In this case, the policy can choose to utilize a robotic arm to measure terrain in places which are obstructed for camera or laser views. As this kind of additional measurements is inherently very slow, it is desired to maximize the utility of each such measurement. We present a way to find a policy that chooses the optimal order of these measurements.

Another approach to utilize the imperfect sensing in a learning loop is presented in the last part. To tackle the problem of sample complexity of deep RL, we propose to do most of the "work" in simulation. Under the

assumption that perception is the major difference between simulation and reality, we develop an algorithm that transfers the simulated perception into a domain closer to the real world. The proposed learning–planning–transfer loop converges to a policy trained mostly in the simulator, which however behaves optimally in the real world. One of the difficult parts of this algorithm is tuning the CycleGAN to converge and recognize when it actually converged.

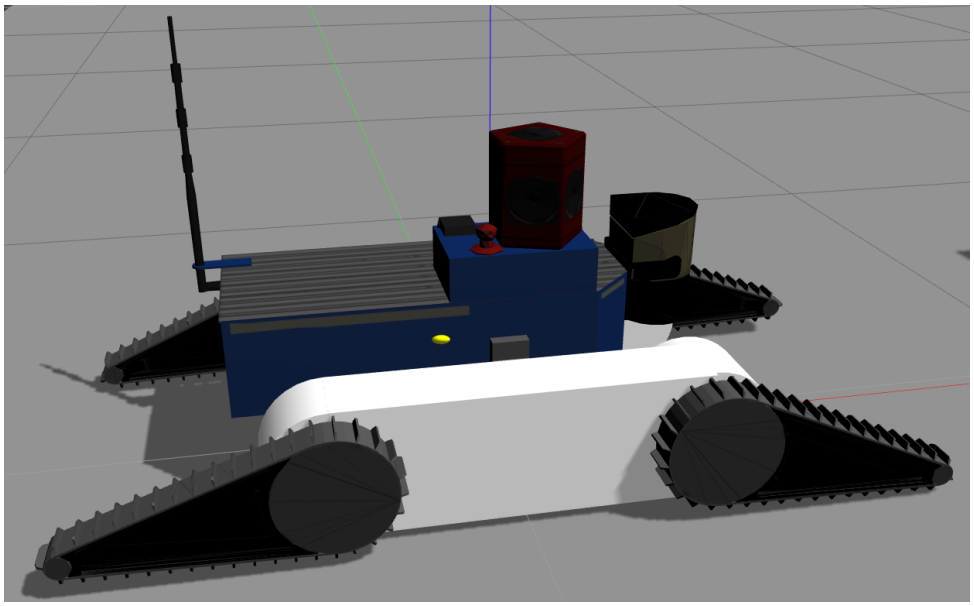
In Appendix A, we also show a prerequisite for most of the work presented in this thesis—a plausible yet fast method for simulation of tracked vehicles. It has been released as open source and to the best of our knowledge, it is the first plausible open-source implementation of non-deformable tracks for mobile robots.

Future Work

Safety and deep neural networks in RL still have many open questions. We list some that we find the most interesting and which we would like to continue investigating.

The safe exploration approach by [Hasanbeig et al., 2020] seems to be the most practical for mobile robots. It does not require deep theoretical analyses of the dynamics model, safety or reward function. Instead, it only requires an *oracle* that can label actions and states in the immediate vicinity of the agent as either safe or unsafe. The method, however, shows no clear way how to transfer it to real robots in large state and action spaces. In simulation, implementation of the *oracle* is straightforward. We would like to investigate if it is possible and practical to construct such oracle also for the real robot with its imperfect sensing and all other complications.

Another promising research path is extending the method from Part III with a tighter feedback between the CycleGAN and the learning part. If the CycleGAN knew not only the gradients with respect to its discriminators, but also the gradients with respect to the policy, the learning process could be more stable and it could provide a convergence measure. Conversely, if the policy will know the limitations of the CycleGAN, it can learn to compensate for them. The review of relevant literature in Part III also shows many ways of improving convergence properties of GANs. And, of course, we would like to integrate the safety constraints from CREPS together with this algorithm, which would yield safe deep reinforcement learning.



Appendices

Appendix A

Tracked Vehicles Simulation

This appendix presents our novel technique that allows for both computationally fast and sufficiently plausible simulation of vehicles with non-deformable tracks. Such simulation model was a requirement for many experiments described throughout this thesis.

The method is based on an effect we have called *Contact Surface Motion*. A comparison with several other methods for simulation of tracked vehicle dynamics is presented with the aim to evaluate methods that are available off-the-shelf or with minimum effort in general-purpose robotics simulators. The proposed method is implemented as a plugin for the open-source physics-based simulator Gazebo using the Open Dynamics Engine.

The text of this appendix is largely based on our work:

- Pecka, M., Zimmermann, K., and Svoboda, T. (2017). Fast simulation of vehicles with non-deformable tracks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6414–6419, Vancouver, Canada. IEEE. DOI: 10.1109/IROS.2017.8206546

It has been edited to fit the thesis structure and goals.

A.1 Introduction

It is common in robotics research that the initial development of algorithms is first conducted in a simulator or game engine to avoid excessive wear of the real vehicle. In this phase, approximate simulation methods usually suffice, differing by the level of approximation and computation time. General-purpose simulators like Gazebo, V-REP, Webots, MORSE and Actin are often used for this task [Drumwright et al., 2010], providing various approximate motion models implemented in their physics engines (ODE, Bullet, Havoc).

Simulation of wheels is straightforward in these simulators, thus all of them provide means to simulate wheeled vehicles, including skid-steer motion of multi-wheel vehicles. However, there is no straightforward approach for tracked vehicle simulation, thus this motion model is not available in most simulators. After an exhaustive search, only two simulators were found that provided a tracked robot in their robot model library—the commercial

simulators Webots and V-REP. However, none of these implementations is both plausible on difficult terrain and computationally light.

The most plausible and general simulation methods for rubber belts are based on finite elements analysis, where the belt is subdivided in many small elements that interact in a defined way. We omit this class of methods here due to their inherent excessive computational complexity which makes them impractical for quick algorithm prototyping. Further argument for omitting these methods is that none of the most used open-source dynamics engines used in robotics supports simulation based on finite elements.

In this appendix, we present a novel technique for non-deformable tracks simulation, which we implemented in the open-source simulator Gazebo [Pecka, 2017]. The method provides a fast, simple and plausible simulation of non-deformable tracks with minimal changes to the simulator code and no changes to its physics engine (ODE). We would like to emphasize that our motivation is to have a fast and plausible method that can be easily integrated into existing robotics simulators and does not require implementation of state-of-the art physics engine components (which are usually absent in the robotics simulators).

We compare this method to other already known motion models. Finally, we propose a set of metrics that allow to compare the methods in terms of plausibility, computational time, and the range of track types that can be simulated by each of the respective techniques.

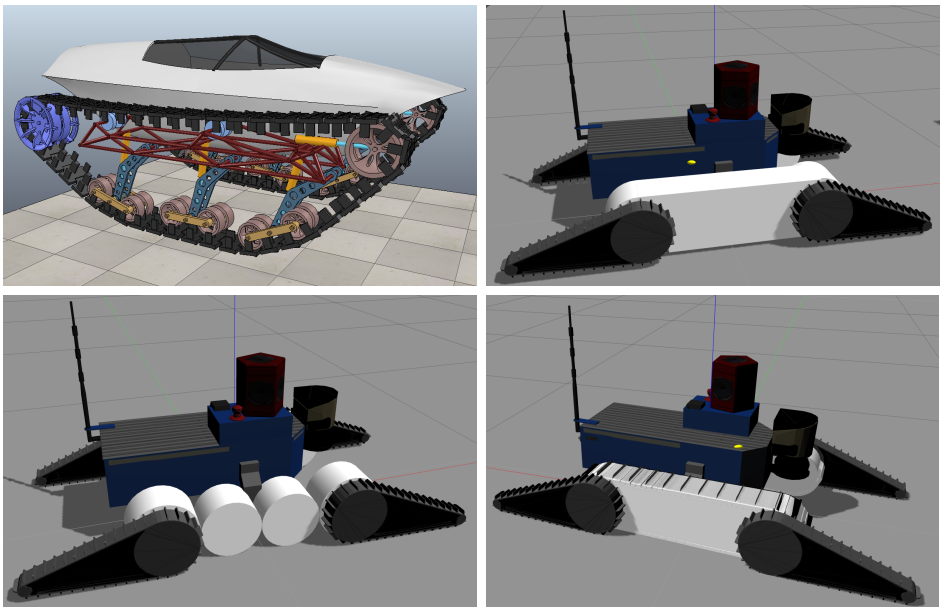


Figure A.1: Track models. ***Top left***: A vehicle with chain-like deformable tracks. This is the model available in model database of the V-REP simulator (courtesy of Qi Wang). ***Top right***: Non-deformable track model used for the proposed method. ***Bottom left***: Track approximated by 4 wheels. ***Bottom right***: Track made of 2 cm plates with grousers.

A.2 Types of Caterpillar Tracks

To clearly specify the type of vehicles this work is focused on, a short taxonomy of track types follows.

Based on the material the track is made of, the two basic types are *metal* tracks and *rubber* tracks. Metal tracks are usually made of many small *track plates* connected together with hinge-like joints. They are used wherever heavy load, high reliability and easy repairability are required. Rubber tracks are made of a continuous steel-reinforced band of rubber. Their common use cases are lighter tracked vehicles (like robots) and conveyor belts.

Another distinctive feature of different track types is the deformability of the outer shape of the track. The deformable track systems need a set of inner (sometimes also outer) wheels keeping the track approximately in the required shape and providing suspension (see Figure A.1). The track can bend in between the wheels, hence the name *deformable tracks*. Metal tracks are usually deformable, and also deformable rubber tracks exist.

Non-deformable tracks have solid guides (infills), which prevent the outer belt shape from bending and deformation (see Figure A.1). This design is often chosen for rubber tracks, and it is the type this comparison is focused on.

A special category—*conveyor belts* and *escalators*—may be added to this taxonomy. In many design principles they are similar to the tracks for vehicles, but the main difference is they are always fixed to the environment and thus have no dynamics as a whole.

Independently from the above categories, tracks can be equipped with *grousers*. These protrusions enlarge the contact surface and help to increase traction in soft materials (depicted in Figure A.1).

A.3 Related Work

Depending on the purpose of the simulation, either very precise and detailed, or approximate models can be used. The former ones have been studied extensively in literature, whereas the approximate models, due to their triviality and inaccuracy, have not been examined profoundly despite their frequent use in nowadays robotics.

A.3.1 Precise Models

Simulation of the deformable tracks can be completely set up using existing robotics simulators—the track consists of a set of solid track plates connected with hinge joints, several wheels and, possibly, suspension of the wheels. All these components are available in simulators like Gazebo, Webots or V-REP. However, this type of simulations is both computationally intensive and very unstable for the high number of constrained dynamic elements [Kenwright and Morgan, 2012]. Only the V-REP simulator provides a reliable simulation of this type, and many parameters have to be very finely tuned for it to

work. [Sokolov et al., 2017b] tried to implement this method in Gazebo, but the reported results are unsatisfactory.

When the general-purpose simulators fail, specialized simulators were developed to simulate the deformable track dynamics. [Wallin et al., 2013] compared several formulations of the mechanical joints when applied to metal tracks. They conclude that each formulation has its advantages and disadvantages and has to be chosen with respect to the specific use-case.

As discussed in the introduction, considerable effort is devoted to simulation of tracks using the Finite Elements Analysis [Arias, 2012], [Ma and Perkins, 2006]. But the precision and computational demands are of higher orders than the methods we focus on.

In agriculture and military research, the track-soil interaction is of high interest (mainly due to sinkage of the track plates). Most of these works seem to only consider planar motion of the vehicle [Ferretti and Girelli, 1999, Janarthanan et al., 2012, Rubinstein and Hitron, 2004] and mainly concentrate on computing correct sinkage-induced behavior. [Yamakawa and Watanabe, 2004] provide a fully three-dimensional simulation taking into account the track-soil interactions and wheel suspension.

A.3.2 Approximate Models

Common feature of the models described in the previous section is that they properly simulate some effects, but are either very computationally intensive, or neglect some other important effects (they e.g. assume motion on flat ground with small obstacles only).

We are not aware of any approximate model for the deformable track type, because its behavior is highly nonlinear and it essentially requires to model the individual parts of the track separately. The rest of this section thus concentrates on approximate models for non-deformable tracks.

In some environments, only flat ground is present (e.g. in household robotics or storehouse helper robots). Then there is effectively only a very small difference between a tracked robot and a 4-wheel robot with skid-steer control. In such scenarios, simulating the track by 4 wheels may be sufficient (with synchronized velocity of the wheels on each side).

In some cases, the tracks can be treated completely passive and the robot motion can be roughly estimated by setting zero friction to the track surface, and pushing the robot with a virtual force instead of driving the tracks. This force can be applied via a P(ID) controller, so that the robot achieves the desired velocity and keeps it. However, the usual effects of friction can not be simulated. Consequently, the robot can not stand on a tilted plane without control force (which the real robot can do).

When negotiation of obstacles needs to be accounted for, the 4-wheel approximation would fail because the robot could not support itself on obstacle edges by the middle parts of the tracks. In this case, the problem is often solved by putting more virtual intersecting wheels inside the track. This approach has been tested in [Sokolov et al., 2016], and is available as

a predefined model in V-REP and Webots simulators. The model still uses the skid-steer wheel control with synchronized wheel velocities on each side. On one hand, it has problems imitating the skid-steer behavior properly. On the other hand, the robot is able to overcome some obstacles and can support itself by any part of the track, and the motion can be initiated in the "physically-correct" way by applying torque to the wheels. But the geometry of such model does not correspond to the real geometry, which is why these models cannot plausibly simulate e.g. climbing up a staircase. We have observed in section A.5 that this model also gets stuck in some cases where the real robot would continue going. These models also do not work very well with the standard *friction pyramid* approximation of friction direction—it is instead needed to use the more precise (and more computationally expensive) *friction cone* model [Drumwright et al., 2010].

The V-REP simulator offers another method of approximate simulation, which is only suitable for conveyor belts and other static elements. It bypasses the physics by directly setting linear velocity of the whole conveyor belt mechanism, letting it interact with other bodies, and resetting all forces that acted on it afterwards. This way, the conveyor belt can exert forces on objects colliding with it, but at the same time, it stays on its place unaffected by any kind of dynamics (because the forces are zeroed-out each simulation step).

A.3.3 Skid-steer Motion

The slippage in the skid-steer behavior is an essential part of motion of tracked vehicles. While it automatically emerges from the precise simulation models as a result of track tension and other forces acting on the individual parts of the track, a kinematic model is also available for approximate or kinematics-only simulations.

[Martínez et al., 2005] define virtual points called *Instantaneous Centers of Rotation* (ICR) which depend on the desired turning radius and on a coefficient called *steering efficiency*. The robot follows a circular path centered at the ICR and if the steering efficiency is equal to 1, the motion is the same as the motion of a geometrically equal differential-drive wheeled vehicle.

[Janarthanan et al., 2011] extend this theory for tracked vehicles with road wheels.

A.4 Model Based on Contact Surface Motion

Our novel method exploits the dynamic simulation formulation as *Linear Complementarity Problem* (LCP), which is used in ODE [Smith et al., 2005] and other robotics simulators. It does not, however, depend on any particular LCP solver implementation.

A.4.1 LCP Formulation

The dynamic simulation problem is an application of Newton's second law:

$$\mathbf{F} = Ma = \frac{d(M\dot{\mathbf{q}})}{dt} \tag{A.1}$$

where t is time, $\mathbf{F}$ is the force acting on the dynamic system, M is the mass and inertia matrix, and $\dot{\mathbf{q}}$ is the linear and angular velocity of the bodies (which is the derivative of the system state $\mathbf{q}$). The force $\mathbf{F}$ is split into *external force* $\mathbf{F_e}$ and *constraint force* $\mathbf{F_c}$ [Kenwright and Morgan, 2012], which is a set of forces generated by joint constraints that keep joint constraints valid in the next time step.

The constraints are written in the form

$$\dot{C}(\mathbf{q}) = J\dot{\mathbf{q}} \geq 0 \tag{A.2}$$

where J is the *constraint Jacobian*. An observation in [Kenwright and Morgan, 2012] states that the direction of the constraint force is given by J, so it is sufficient to search for the constraint force magnitude λ (so that $\mathbf{F_c} = J\,\lambda$).

In simulation, the derivative is discretized into short time steps Δt (usually 1 ms) and the state of the system is integrated step-by-step using Euler's integration [Kenwright and Morgan, 2012]. The state of the system in the next time step $n+1$ can be expressed as

$$\mathbf{q_{n+1}} = \mathbf{q_n} + \mathbf{v_{n+1}}\Delta t$$

where the new velocity vector $\mathbf{v_{n+1}}$ (corresponding to $\dot{\mathbf{q}}$ in the continuous setting) is obtained from Equation A.1:

$$\begin{aligned} \mathbf{v_{n+1}} &= \mathbf{v_n} + M^{-1}(\mathbf{F_e} + \mathbf{F_c})\Delta t \\ &= \mathbf{v_n} + M^{-1}(\mathbf{F_e} + J\lambda)\Delta t \end{aligned}$$

The unknown constraint force magnitude λ is the solution of the following LCP [Kenwright and Morgan, 2012]:

$$\begin{aligned} JM^{-1}J^T\lambda\Delta t + J(\mathbf{v_n} + M^{-1}\mathbf{F_e}\Delta t) &\geq 0 \\ \text{given } \lambda \geq 0,\ J(\mathbf{v_n} + M^{-1}\mathbf{F_e}\Delta t) &\geq 0 \\ (J(\mathbf{v_n} + M^{-1}\mathbf{F_e}\Delta t))^T\lambda &= 0 \end{aligned}$$

A.4.2 Contact Constraint Equations

In each time step, when links L_1 and L_2 collide, a set of contact points $\{C_i\}_{i=0}^{N}$ is generated at places where the links touch or penetrate each other. Every contact point is assigned a *contact joint*, which is a temporary constraint between L_1 and L_2. The set of constraints yielded by the contact joint consists of a position constraint (repelling the two links from each other along the contact normal), and a velocity constraint for friction (stopping

parallel motion of the two links), which often utilizes the Coulomb friction representation [Trinkle and Pang, 1997, Kaufman et al., 2008].

Linear velocity of L_1 is denoted by $\mathbf{v_1}$, angular velocity by $\omega_\mathbf{1}$, and $\mathbf{r_{1i}}$ is the vector from the center of L_1 to C_i; respective definitions hold for L_2. Further, $\mathbf{t_i}$ denotes the main tangential friction direction (which is perpendicular to the contact normal).

The approximate velocity constraint for Coulomb friction at contact point C_i with friction coefficient μ_i is [Trinkle and Pang, 1997]:

$$\frac{\partial C_i}{\partial t} = (\mathbf{v_2} + \omega_2 \times \mathbf{r_{2i}} - (\mathbf{v_1} + \omega_1 \times \mathbf{r_{1i}})) \cdot \mathbf{t_i} = 0 \tag{A.3}$$

$$-\mu_i \leq \lambda_i \leq \mu_i \tag{A.4}$$

which can be interpreted as "stop any motion in direction $\mathbf{t_i}$". The LCP solver tries to find magnitude of the friction force in direction $-\mathbf{t_i}$ (which is bounded by μ_i) that would satisfy this equation.

A.4.3 Contact Surface Motion Model

With the previous definitions, our novel method can be described as a modification of Equation A.3. To account for the desired track velocity v_t, Equation A.3 is adjusted to:

$$\frac{\partial C_i}{\partial t} = v_t$$

which might be interpreted as "find a force that would keep relative motion of L_1 and L_2 at velocity v_t". With this change, the model will move just by applying the modified friction constraints and setting v_t.

Nevertheless, this model is not able to correctly simulate grousers. If the real track has grousers, one way to add a similar effect to the simulation is to increase the friction coefficient.

There are more precise models for contacts with friction [Kaufman et al., 2008], but the practical experiments have shown that even the friction pyramid approximation used in ODE is sufficient for our method to work.

This method can be also easily used for tracks of various shapes. The only requirement is to be able to compute the normals of contact points on the tracks.

A.4.4 Enabling Skid-Steer Motion

The last part to be defined is the friction direction $\mathbf{t_i}$. If only forward motion is required, it can be simply set to be parallel to the tracks. However, this setting causes problems when the robot should turn around using skid-steering motion (since the friction forces are not consistent with the turning maneuver).

Here we connect the dynamic simulation with the kinematic model of tracked vehicle motion by Martínez et al. introduced in section A.3.3. The whole vehicle is said to be following a circular path centered at ICR (or driving straight if ICR is at infinity). Thus, we know the desired trajectory

of all contact points on the track, and we set each direction $\mathbf{t_i}$ to be tangent to this trajectory, see Figure A.2.

Implementation of the proposed method and the wheel-approximated tracks was merged into the Gazebo simulator [Pecka, 2017].

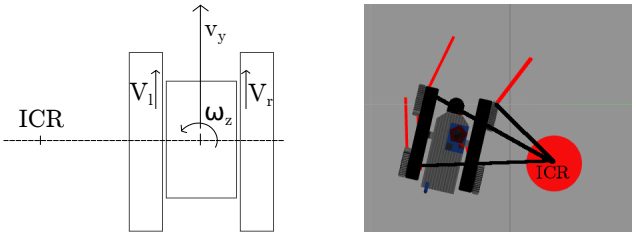


Figure A.2: Instantaneous Center of Rotation. ***Left***: A schematic view of the ICR. If the vehicle doesn't slip to the sides, ICR lies always on the depicted horizontal line passing through the centers of the tracks [Martínez et al., 2005]. The distance of ICR from the center depends on forward velocity v_y and angular velocity ω_z (inverse kinematics), or the speeds of the left and right track V_l and V_r (forward kinematics). ***Right***: Computed directions of the friction forces $\mathbf{t_i}$ (red lines) for the case where ICR lies in the center of the red disk. The friction forces are perpendicular to the (black) lines connecting the contact points with ICR.

A.5 Comparison of Models

In this section, a comparison of methods of modeling non-deformable tracks is presented.

A.5.1 Tested Models

The tested models are described in the following sections (and depicted in Figure A.1). Each model is shortly introduced, and an abbreviation for it is defined, which is used throughout the rest of the text and figures. All the tested models differ only in representation of the main tracks—all other properties, such as mass, inertia, shape etc. were the same for all models.

With each of the models, identification of the most realistic set of parameters was done. The optimized parameters were always *linear* and *angular gain*—ratios that convert control inputs from simulator to velocity commands for the models. Other parameters were added only for the models they make sense with, and consist of *steering efficiency* and friction coefficients in the first and second friction direction.

First, we tried to manually find a suitable set of parameters and estimated the ranges for each of them. Then we did 5 iterations of optimization, in each of which we examined 5 samples from a multivariate Gaussian distribution centered on the so far best set of parameters (with covariance derived from the estimated ranges). Examination of each sample consisted of traversing all defined scenarios with model settings taken from the sample, and summing

up the weighted metric values (defined further in this section). To account for the uncertainty in the simulator, each traversal was tried 3 times and the metric value was averaged over these trials.

Model based on Contact Surface Motion

This is the novel model shortened as **CSM**.

Wheels instead of tracks

Model with 4 wheels instead of each track (**4wheels**) or 8 wheels (**8wheels**). All wheels are velocity-controlled using a skid-steer wheel control mechanism (with wheels on each side synchronized in velocity).

Subdivision to plates

Model with belt subdivided into 10 cm plates (**plates10**) or 2 cm plates (**plates2**) interconnected by hinge joints, plus sprocket and idler wheels. Versions with grousers attached to the track plates are shortened as **plates10g** and **plates2g**. The inner space of the track is filled with a solid box which can collide with the track plates, thus emulating the non-deformability of the track. Only the sprocket wheel is controlled, using torque control. This model requires more tuning in the simulator. To simplify it, the sprocket wheel is represented by a cylinder with infinite friction with the track plates (so that it efficiently transfers force to them without the need to model the teeth and their interaction with the plates). Further, lateral motion of track plates has to be avoided (otherwise, they would slip off the track very easily). This would be best done with a planar joint, which is however not available in Gazebo/ODE. As a workaround, placing two virtual vertical plates to the sides of each track (that collide only with the track plates) yields a similar behavior (although it is not ideal).

No friction

Model with zero friction between the tracks and ground (**no_friction**). The collision shape of the track is the same as in the **CSM** model, but the friction of the track is set to zero, and the whole model is force-controlled by applying a virtual force at its center of mass. The applied force is always perpendicular to the vertical axis of the robot.

The real robot

The **real** robot was also part of the test. It is the Absolem platform described in section 2.2. Position of the robot in 6D space was measured by an IMU combined with track odometry and ICP laser SLAM [Kubelka and Reinstein, 2012].

A.5.2 Test Scenarios

The models were tested in the following scenarios. Each scenario specifies a different metric showing how successful the model was, and was selected specifically to discover weak points of the models. All the scenarios start with the robot in rest, no initial speed, forces or torques. A view on the obstacles in the scenarios is provided in Figure A.3.

CPU time was measured in all scenarios. It represents the (real-world) time difference between the start of first scenario execution, and the end of the last scenario execution (so it is summed up over all scenarios for each model). The simulators were running with high process priority without an upper bound on performance. The time complexity could be probably lowered for most of the models by adjusting the dynamics engine for the particular case; our measurements show CPU time needed by the implementation in the stock simulator without any code modifications.

Where a metric refers to the error from real robot trajectory, it means the scenario was traversed with the real robot, and the trajectory was recorded as a reference.

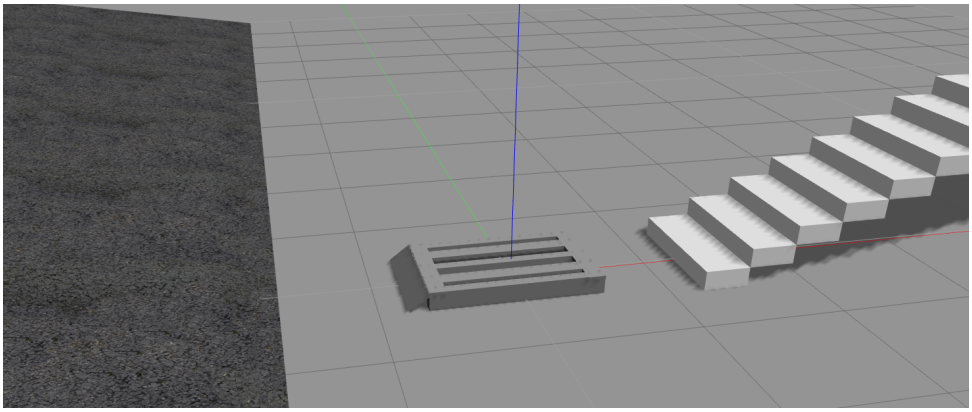


Figure A.3: Obstacles used in test scenarios. Obstacles that appear in the test scenarios (from the left): ramp, pallet, staircase. Also flat ground was used in scenarios. The models of the obstacles are 1:1 models of the obstacles traversed by the real robot.

Straight drive

Drive straight on a building floor using velocity 0.3 $m.s^{-1}$ for 10 seconds. Metric: distance from point $(3.0, 0.0, 0.0)^T$.

Rotating in place

Keep the center at one place while rotating at 0.6 $rad.s^{-1}$ for 10 seconds. Metric: Angular distance from heading 6.0 rad, metric distance from the starting point.

Circular path

Follow a circular path by driving left track at velocity 0.1 $m.s^{-1}$ and right track at velocity 0.3 $m.s^{-1}$ for 10 seconds. Metric: Sum of positional errors (from real robot trajectory) sampled at 10 Hz.

Ramp

Drive straight on a tilted ramp using velocity 0.3 $m.s^{-1}$ for 10 seconds. Metric: Sum of positional errors sampled at 10 Hz, sum of angular errors sampled at 10 Hz.

Staircase

Climb down a staircase using velocity 0.3 $m.s^{-1}$ for 10 seconds. Metric: Sum of positional errors (from real robot trajectory) sampled at 10 Hz, sum of angular errors sampled at 10 Hz.

Stand on staircase

Stand on a staircase with no control commands for 10 seconds. Metric: Distance from the starting point, angular offset from the starting orientation.

Pallet

Climb over a pallet using velocity 0.1 $m.s^{-1}$ for 30 seconds. Metric: Sum of positional errors (from real robot trajectory) sampled at 10 Hz, sum of angular errors sampled at 10 Hz.

Back and forth

Drive using velocity 0.2 $m.s^{-1}$ back and forth 10 times, with 2 seconds between every direction switch. Metric: distance from the starting point.

A.5.3 Test results

Each model was tested 10 times in each scenario, and the values of the metrics were averaged over these tests.

The detailed results are shown in Table A.1. A summary extracted from the test results is given in Table A.2.

From the table, it follows that the track plate models are slower by an order of magnitude or two than the other models. We have also observed, that the 10 cm plates are too rough approximation of the smoothly curved belt, and the resulting model's motion could be described as "bumpy". Last observation for track plate models is that without grousers, the robot is often not able to climb up the pallet. That, however, corresponds to the expected real behavior of a belt without grousers.

The wheeled models are computationally fast and provide good plausibility in most scenarios. They suffer from unrealistic slippage in the *stand on staircase* scenario, because the friction forces have unrealistic directions. The *pallet* scenario showed to be a big problem for these methods—if a sharp edge (e.g. a step or pallet edge) touches the track in a point where neighboring wheels intersect, the model suddenly stops moving as a result of unrealistic forces and their directions. We think it is not a bug in our implementation, since the same behavior was also observed with the wheeled track model available in V-REP simulator (which even uses a different dynamics engine—Bullet).

The *no_friction* model provided good results in all tested scenarios, except *stand on staircase*. That failure is obviously caused by the missing friction between tracks and ground. It was the fastest tested model.

The proposed *Contact Surface Motion* model was the second fastest tested model. It provided good results in all tested scenarios except *circular path*. Here, the parameter optimization was not able to find a set of parameters that would provide good performance for both *rotate in place* and *circular path*; with the best set of parameters, the robot was turning too quickly in the *circular path* scenario. Together with *no_friction*, only these two models traversed the pallet without problems.

A.6 Conclusion

Simulation of tracked vehicles is a complicated task even when it is narrowed down only to simulation of non-deformable tracks. The presented *Contact Surface Motion* model proved to be one of the fastest methods that still provides highly plausible results in most cases. It is the first computationally-light method allowing the use of precise geometry of the tracks while keeping plausible dynamic behavior. It can be utilized not only for simulation of tracked vehicles, but also for conveyor belts, treadmills and any other kind of moving planar surfaces.

The Gazebo plugin that implements tracked vehicles using *Contact Surface Motion* was merged to mainline Gazebo on May 2019. It has been used by several other robotics labs since then, including CSIRO Data61, LIRS at Kazan Federal University, or TU Eindhoven.

	Metric	**csm** (proposed)	**4wheels**	**8wheels**	**no_friction**
Straight	d_t	$\mathbf{0.1 \pm 0.0}$	$\mathbf{0.1 \pm 0.0}$	$\mathbf{0.1 \pm 0.0}$	0.2 ± 0.1
Rotate	d_ω	$\mathbf{0.1 \pm 0.0}$	0.5 ± 0.0	1.6 ± 0.0	$\mathbf{0.1 \pm 0.1}$
Circular	$\sum d_t$	157.8 ± 5.7	$\mathbf{45.5 \pm 1.7}$	116.9 ± 6.4	47.4 ± 2.2
Back&forth	d_{st}	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	0.1 ± 0.0
Ramp	$\sum d_\omega$	$\mathbf{1.8 \pm 0.0}$	2.2 ± 0.1	$\mathbf{1.8 \pm 0.0}$	2.0 ± 0.0
	$\sum d_t$	8.0 ± 2.7	$\mathbf{4.6 \pm 1.3}$	5.3 ± 0.8	16.8 ± 4.3
Staircase	$\sum d_\omega$	14.0 ± 0.8	10.2 ± 0.1	10.7 ± 0.3	17.1 ± 0.4
	$\sum d_t$	$\mathbf{12.1 \pm 1.0}$	16.3 ± 2.3	15.2 ± 0.9	13.0 ± 3.7
Stand on st.	d_{st}	$\mathbf{0.0 \pm 0.0}$	0.1 ± 0.0	0.2 ± 0.0	0.2 ± 0.0
	d_ω	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	0.1 ± 0.0
Pallet	$\sum d_\omega$	$\mathbf{27.6 \pm 0.9}$	27.7 ± 12.5	28.3 ± 17.7	31.8 ± 1.0
	$\sum d_t$	49.9 ± 2.5	116.9 ± 76.2	157.8 ± 46.6	$\mathbf{45.2 \pm 3.4}$
CPU time	time	38.9 ± 1.4	47.5 ± 1.7	82.4 ± 3.3	$\mathbf{33.0 \pm 1.3}$

	Metric	**plates10**	**plates2**	**plates10g**	**plates2g**
Straight	d_t	1.6 ± 0.0	1.3 ± 0.0	1.6 ± 0.1	0.5 ± 0.0
Rotate	d_ω	1.4 ± 0.6	1.0 ± 0.1	2.5 ± 0.2	3.1 ± 0.0
Circular	$\sum d_t$	210.5 ± 30.6	189.5 ± 4.7	564.9 ± 36.3	195.7 ± 6.5
Back&forth	d_{st}	1.3 ± 0.1	0.1 ± 0.0	2.6 ± 0.2	0.3 ± 0.0
Ramp	$\sum d_\omega$	4.6 ± 1.1	2.6 ± 0.4	10.5 ± 4.1	34.7 ± 1.9
	$\sum d_t$	36.3 ± 1.6	61.7 ± 0.2	36.9 ± 3.5	121.9 ± 1.1
Staircase	$\sum d_\omega$	36.0 ± 31.3	$\mathbf{8.4 \pm 1.8}$	25.1 ± 11.1	45.5 ± 1.8
	$\sum d_t$	111.5 ± 5.9	86.8 ± 2.5	14.4 ± 7.0	163.0 ± 1.6
Stand on st.	d_{st}	0.6 ± 0.6	0.2 ± 0.0	0.2 ± 0.2	0.2 ± 0.0
	d_ω	0.4 ± 0.4	0.1 ± 0.0	0.2 ± 0.2	0.1 ± 0.0
Pallet	$\sum d_\omega$	164.4 ± 151.5	47.6 ± 4.7	83.0 ± 13.1	64.3 ± 2.6
	$\sum d_t$	508.3 ± 146.3	181.1 ± 3.8	198.6 ± 72.7	78.4 ± 1.2
CPU time	time	254.9 ± 4.9	2282.6 ± 112.7	203.5 ± 8.3	2241.3 ± 31.8

Table A.1: Numerical comparison of the simulation methods. Numerical results of the conducted experiments. Each model-scenario pair was executed 10 times, and the averages and standard deviations of the defined metrics are shown in the table. Shorthand d_t means the *distance to target point* metric (units are meters), d_{st} is distance from start. Term d_ω denotes the smallest angular offset from target roll-pitch-yaw orientation (units are radians). Terms $\sum d_t$ and $\sum d_\omega$ stand for the *sum of positional errors* or *sum of angular errors* respectively (with units meters and radians). *CPU time* (in last row) is not a scenario, but as it is aggregated over all scenarios for each model, we display it as a row of values. The duration of all scenarios in simulation time is 110 seconds, so a run-time of 30 seconds means the simulation ran at $\frac{11}{3}$ real-time speed on the test notebook. Best results in each scenario are highlighted in bold for better orientation.

	CSM	Wheels	Plates	No friction
Computation speed	✓	✓	×	✓
Plausibility on flat surfaces	✓	✓	✓	✓
Plausibility on rough terrain	✓	×	✓	×
Non-deformable tracks	✓	✓	✓	✓
Deformable tracks	×	×	✓	×
Grousers	×	×	✓	×

Table A.2: Summary results. This table presents an overview based on the results of the conducted experiments. Sign "✓" means that the model is suitable for/supports the given use-case. Sign "×" means that the method is not suitable for/does not support the given use-case.

Appendix B

Involved International Research Projects

Throughout the course of my Ph.D. study, I was part of one European research project and one international robotics challenge team. Both gave me priceless experience with the practical and real-world robotics. Taking the algorithms out into the wild, giving the remote controllers to unknown people and watching them trying to find the physical limits of the systems was a big motivation in making all the algorithms robust and general. Or, in case of the robotics challenge, sending the robots into a deep and dark tunnel and knowing that for the following hour, they will have to "take care of themselves", having no safety people around that would help a stuck robot or stop it a second before a self-destructive action.

Making the algorithms both robust and user-friendly is a long-lasting process, but it bears its fruit. There were many simpler ways or approaches to the problems we tackled, but we knew that solving only the simple cases would not be sufficient. This was the drive that forced us going the difficult, but scientifically more interesting way. Each of the projects had different requirements in general, but the basics related to my work were the same—help the robot in traversing difficult parts of terrain. Thus, the project requirements were well aligned with the goals of this thesis, which helped keeping focused.

B.1 TRADR: Long-Term Human-Robot Teaming for Disaster Response

The main goal of the EU FP7 research project TRADR was to explore ways in which robots and humans can form a cooperating human-robot team in Urban Search&Recsue scenarios [Kruijff-Korbayová et al., 2015]. Following a user-centric design methodology, we iterated countless times the development of the software with practical tests with first responders who gave us feedback. We developed a persistent system for heterogeneous teams of robots and humans that can cover long-lasting, even multiple-day scenarios.

From the robot software developer point of view, one of the most important features of this project was developing tools that allow working in *shared autonomy* mode. In this mode, the robots should autonomously perform

"lower-level tasks" (e.g. flipper control), whereas the "higher-level" control still had to be in the hands of the end-users. Generally, the low-level control may sometimes get into trouble, and in such cases it is very important to find an efficient way to communicate this to the operator. But sometimes the operator himself can try to get the robot into trouble (e.g. because of cognitive overload), and this is where safety comes into play and has to autonomously decide that the robot should rather stop than follow the operators' commands.

When the team of robots is equipped with the described kind of shared autonomy functions, it gets much easier for the operator to drive the robots. And even a higher level of autonomy can be reached—e.g. one operator supervising (semi-)autonomous work of several robots.

In the time frame of TRADR project, a big earthquake struck Italy. The Italian first-responders asked the TRADR team to get to the most destroyed city of Amatrice and use the robots to examine the level of damage to the Sant'Agostino and San Francesco churches [Kruijff-Korbayova et al., 2016]. The robots did successfully investigate both churches and provided Vigili del Fuoco (Italian fire brigade) 3D maps and RGB images of the interiors of the churches. The end-users were extremely satisfied both with the data the robots gathered and with the fact they did not have to risk human lives to gather this essential information.

The project has ended in 2018 getting the best review rating possible. For this, a big "thank you" belongs to all the colleagues who took part in this journey.

B.2 DARPA Subterranean Challenge

In DARPA SubT Challenge, the goal is to build an almost autonomous team of robots that will explore and precisely map a previously unknown underground space. Providing situational awareness to the single mission operator, as well as detecting interesting artifacts throughout the course, is crucial for scoring in the challenge. The underground severely limits the communication between robots and with the base station at the tunnel entrance, which results in the need for a high degree of autonomy. The single mission operator can command the robots, but only as long as communication link is maintained; to get a better score, the robotic team has to be able to autonomously explore the unknown spaces and after some time return to the base reporting what they found.

Our team, CTU-CRAS-Norlab (or, at the beginning, only CTU-CRAS), develops a heterogeneous team consisting of tracked, wheeled and multi-legged ground robots and medium-sized UAVs. As the teams are not permitted to gather any prior knowledge about the competition venues, all the autonomous robots have to be prepared to drive in unknown terrain. The organizers can even make the courses more difficult by adding artificial obstacles. All of this results in the need for high quality safety estimation and terrain traversal capabilities.

In the first competition round held in August 2019 in Pittsburgh, PA, USA, our team scored the 3rd place out of 11 competing teams [Rouček et al., 2020]. The terrain in the coal mine was not difficult, so the autonomous traversal capabilities were not tested very thoroughly and only a simplistic traversal/self-protection algorithm was enough.

The second round of competition took part in February 2020 near Olympia, WA, USA. Among the 10 competing teams, we were again able to rank 3rd, and we again won among all teams which are not funded by DARPA. This round consisted of more complicated industrial terrain, but as we expected that the only real obstacle the robots will need to traverse will be staircases, we deployed a handcrafted control policy which was tuned to work well on staircases.

However, the following competition rounds will certainly include non-trivial obstacles, which will thoroughly test the motion capabilities. We are developing a more capable autonomous traversal algorithm which will not only control the flipper motion, but will also have the ability to reposition the robot so that it encounters obstacles in the most appropriate way.

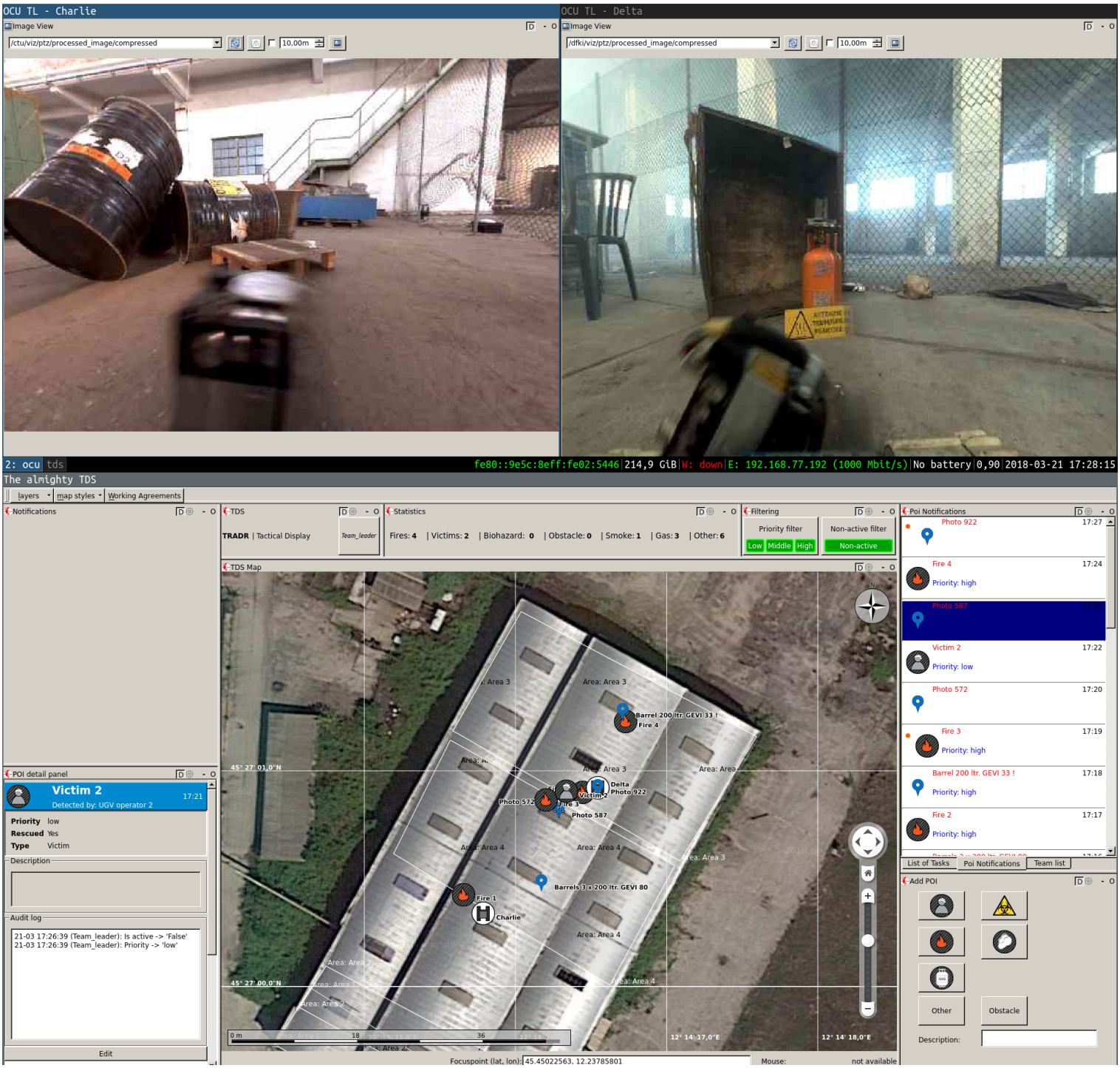


Figure B.1: TRADR: Operator Control Unit view of the Team Leader. **Top:** Live video feed from selected robots. **Bottom:** Overview map of the mission progress with icons representing found objects of interest, positions of other robots, and providing an interface for sharing information with other team members.

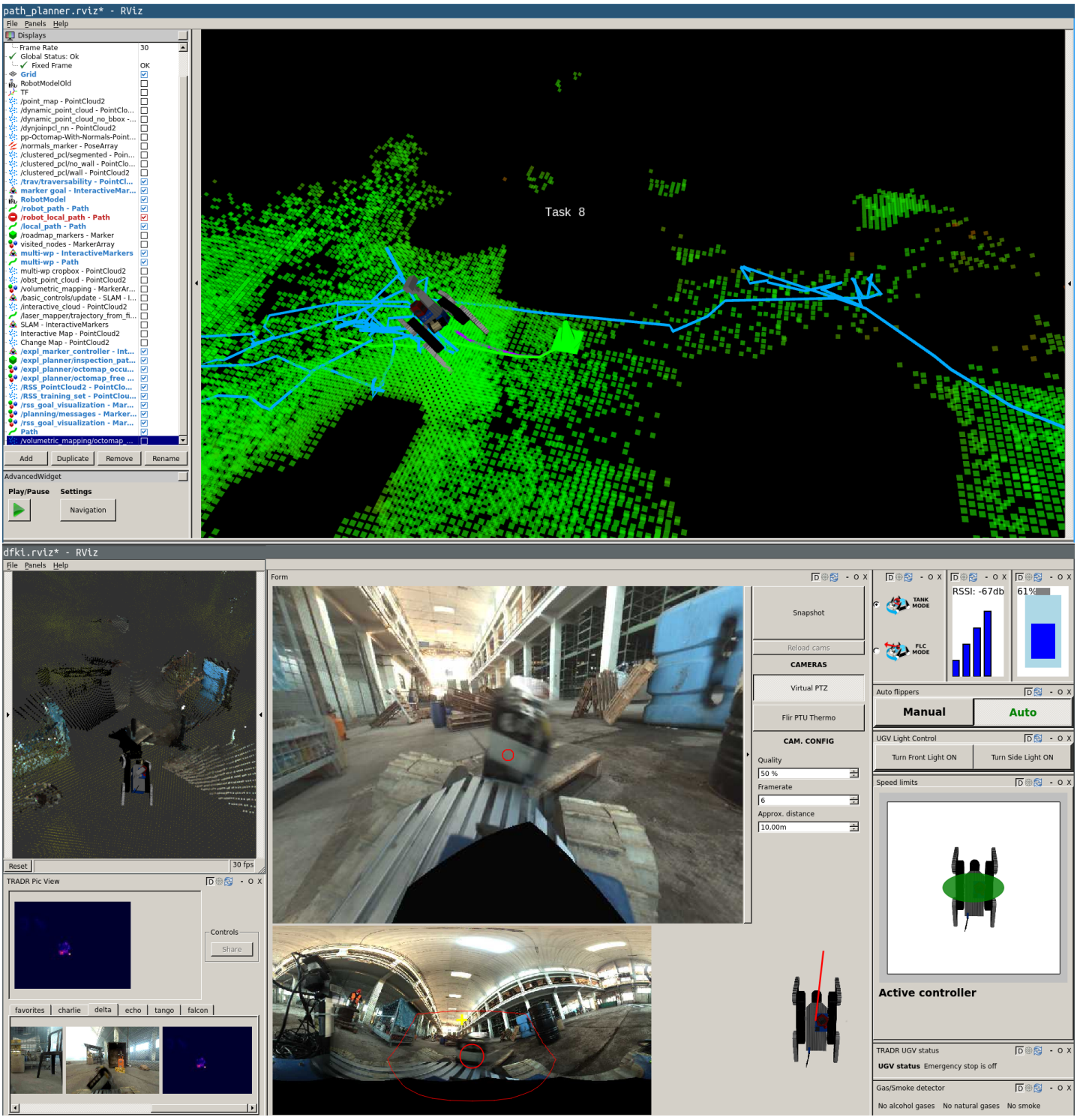


Figure B.2: TRADR: Operator Control Unit view of the robot autonomously traversing a pallet. **Top:** A rendered 3D view of the current robot pose, its plan (green line), and past trajectory (blue line). **Bottom:** A colored 3D map, live video streams from the robot, status information and pictures shared by other team members (lower left).

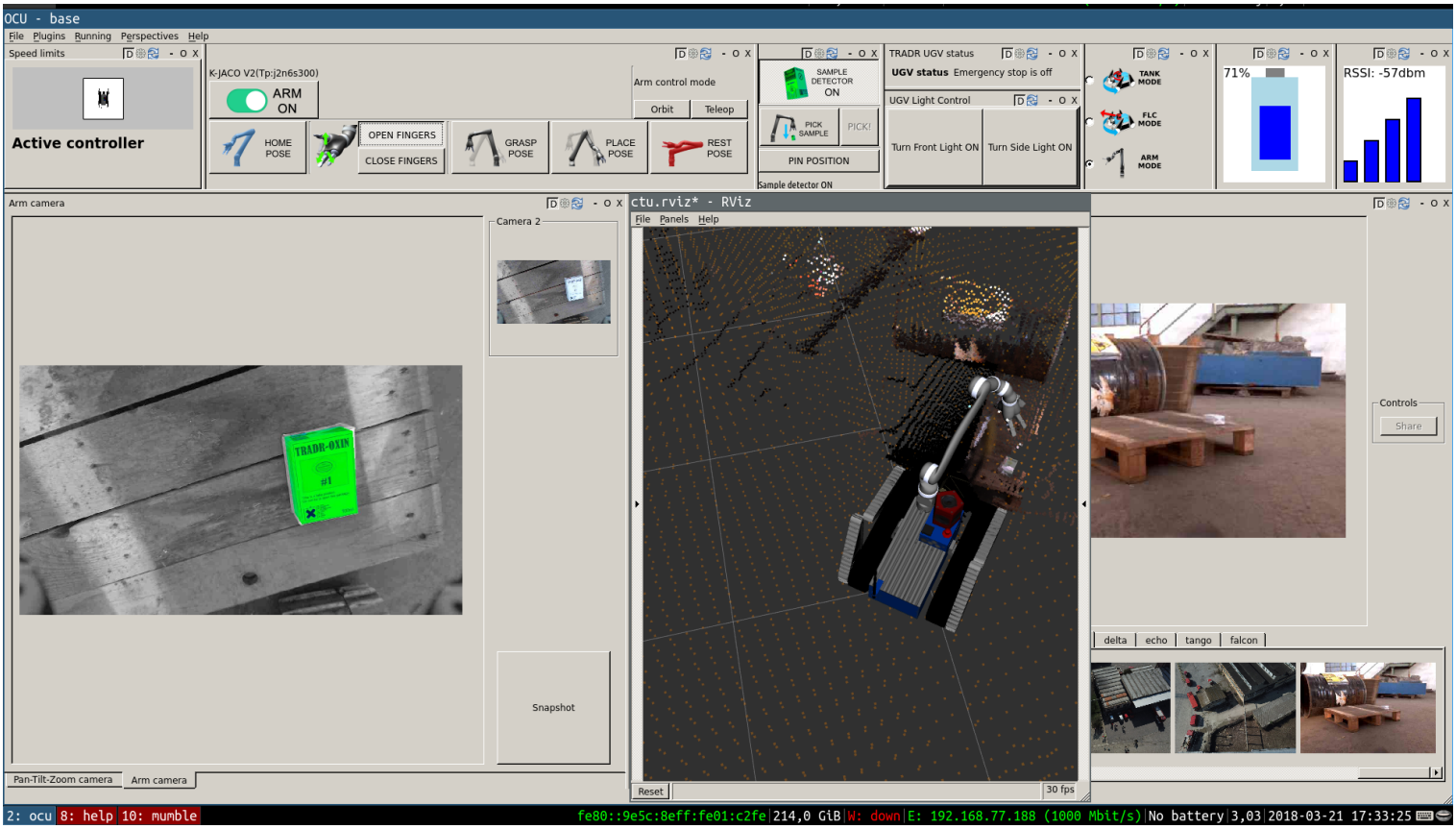


Figure B.3: TRADR: Operator Control Unit view of the robot doing a shared autonomy grasping task. The robot has a 7-DoF arm attached to its body with a camera at the wrist. This camera is used for sample detection (the green-lit rectangle), and the operator can then try to command the robot to grasp the sample automatically, or he can control the arm in a coordinate frame relative to the sample, which showed to be a big help for the operator.

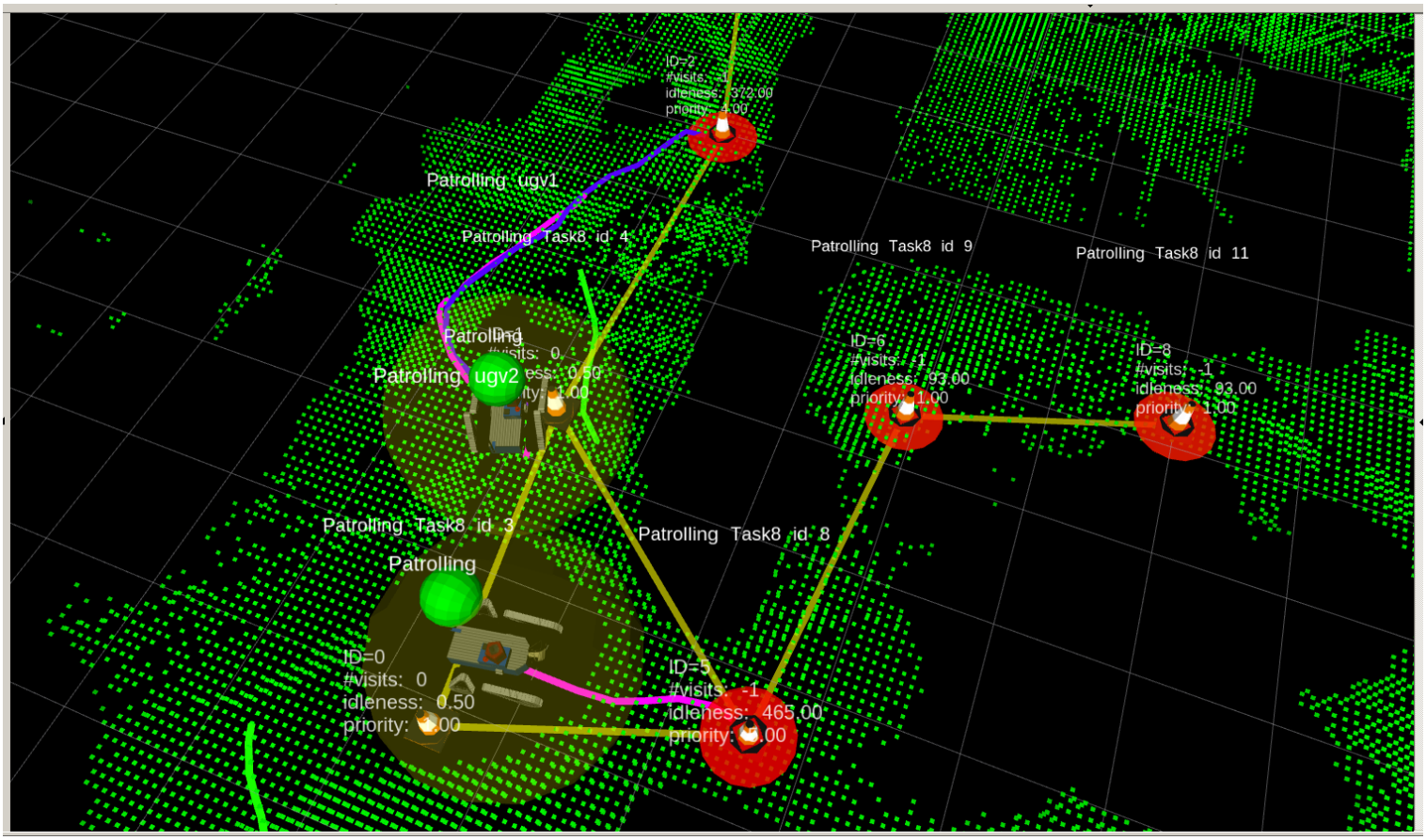


Figure B.4: TRADR: Operator Control Unit view of two robots autonomously patrolling a specified area. The orange-white cones specify places to patrol (*nodes*), and the yellow lines visualize the patrolling graph. Each node displays its idleness (how long it has not been visited), and the control algorithm tries to minimize the idleness of all nodes.

Figure B.5: SubT Challenge: The three UGVs ready to start a training mission. Strong on-body illumination is apparent in this photo, and was required in the dark parts of the tunnels in the competition.

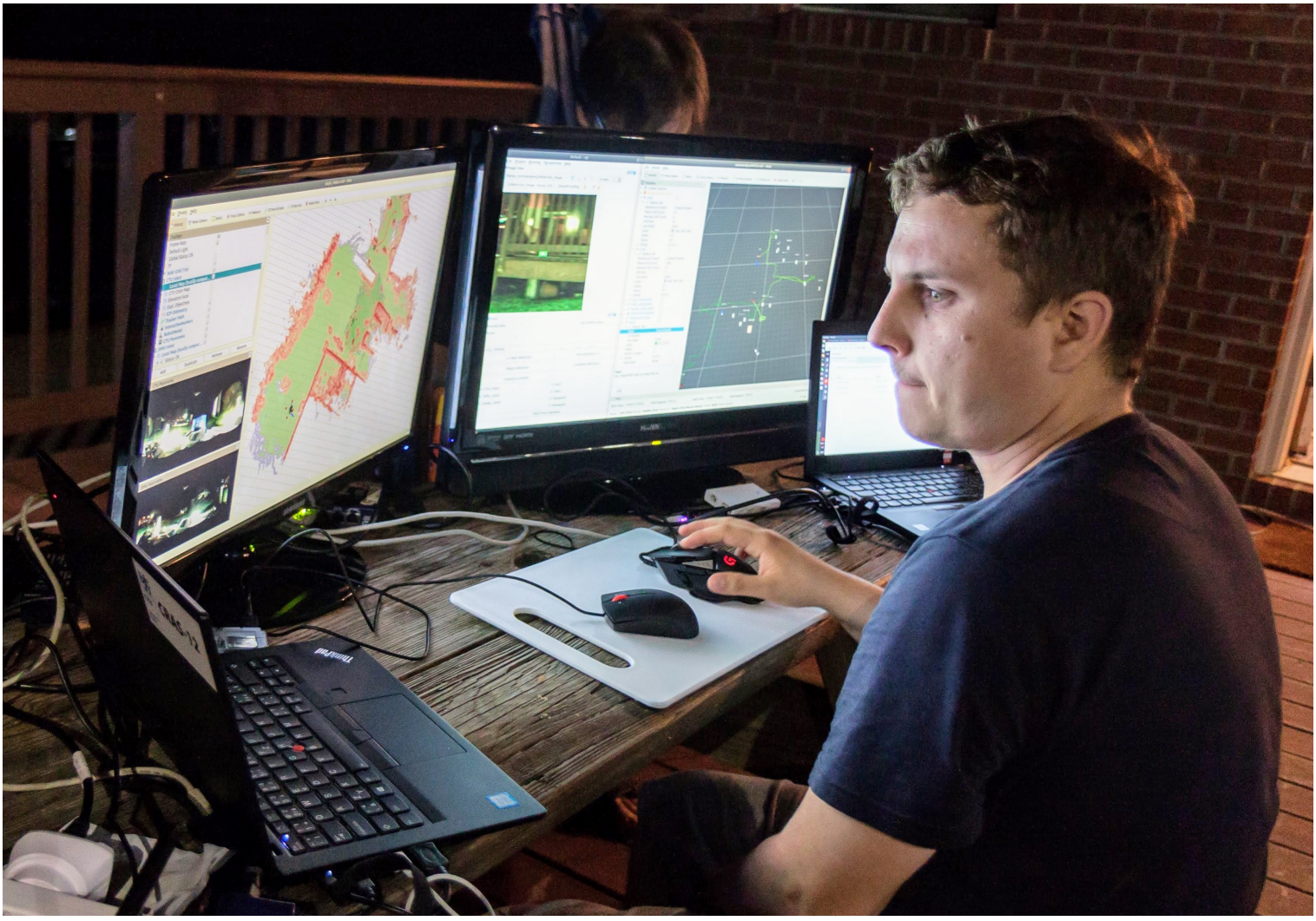


Figure B.6: SubT Challenge: Operator console showing mostly maps of the explored environment, detected artifacts, and if connection is good, also video stream from all exploring robots.

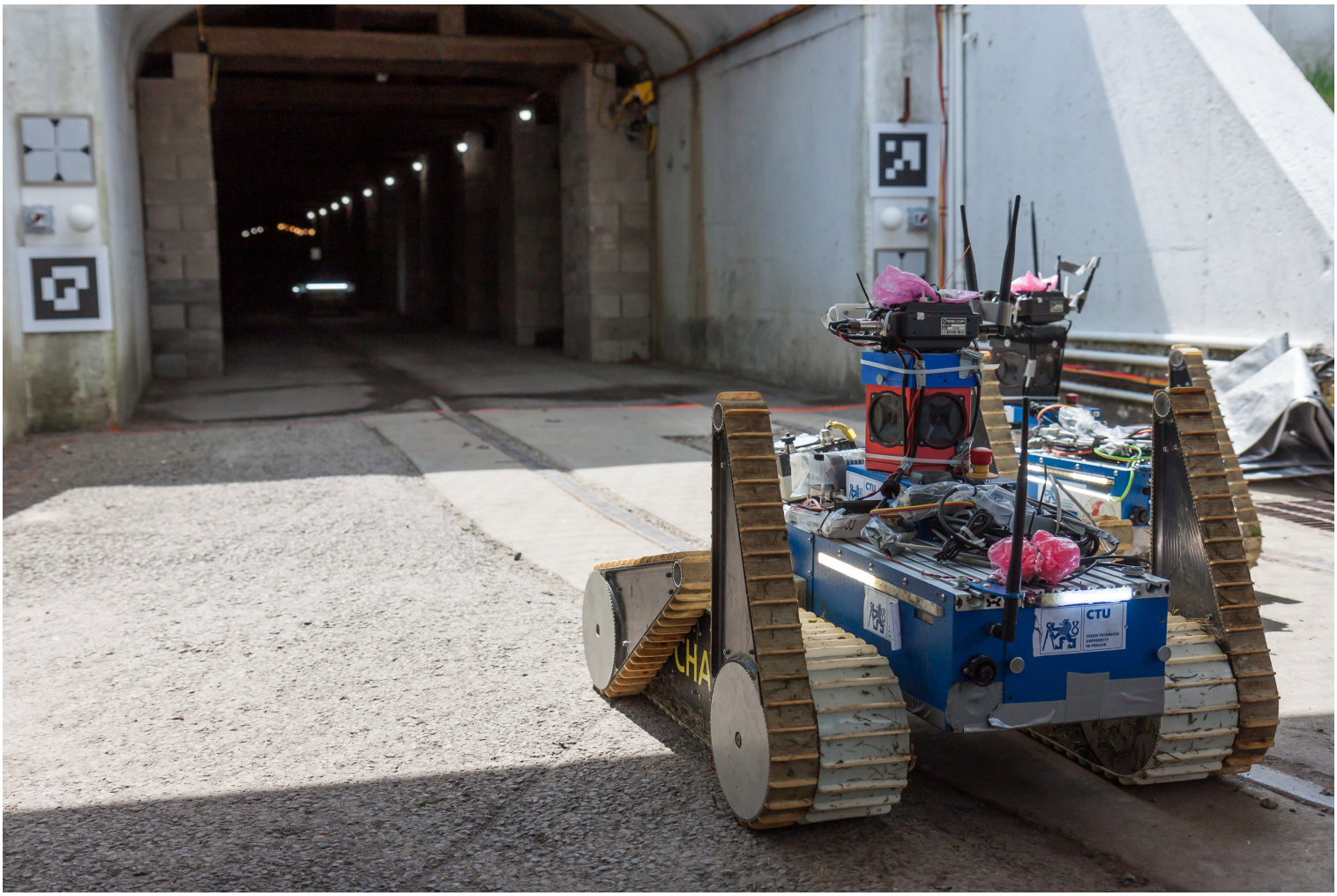


Figure B.7: SubT Challenge: An Absolem robot in front of the competition tunnel entrance. It is observing calibration marks (AprilTags) to align its coordinate frame with the competition coordinate frame.

Figure B.8: SubT Challenge: Three UGVs and one UAV (hard to spot) explore the competition tunnel at the same time.

Figure B.9: SubT Challenge: The CTU-CRAS team with all robots.

Appendix C

Open Source Contributions

A big part of the successes my teams have achieved was thanks to the use of open-source software. Particularly, Robot Operating System (ROS) and Gazebo simulator. But as no software is perfect, we developed many improvements. As a thanks to the community who created these great projects, I contributed and published some of the improvements. And I would like to thank the numerous reviewers who helped to make my contributions even better.

All pull requests can be found via github.com search[1] [2].

Selected contributions to ROS-related projects (descriptions are taken from the repositories):

- **ros-planning/geometric_shapes**: Representation of geometric shapes. 18 pull requests, 12 accepted.
- **ros/ros_comm**: ROS communications-related packages, including core client libraries. 10 pull requests, 4 accepted.
- **ros-planning/moveit**: The MoveIt motion planning framework. 9 pull requests, 4 accepted.
- **ros/xacro**: With xacro, you can construct shorter and more readable XML files by using macros that expand to larger XML expressions. 6 pull requests, 4 accepted.
- **ros-simulation/gazebo_ros_pkgs**: Wrappers, tools and additional API's for using ROS with Gazebo. 5 pull requests, 3 accepted.
- **ros-perception/laser_filters**: Assorted filters designed to operate on 2D planar laser scanners. 4 pull requests, 2 accepted.
- **ros/geometry2**: A set of ROS packages for keeping track of coordinate transforms. 3 pull requests, 1 accepted.
- **ros/robot_state_publisher**: Allows you to publish the state of a robot via the "tf" transform library. 2 pull requests, 1 accepted.

[1] `https://github.com/pulls?page=3&q=is%3Apr+author%3Apeci1`

[2] `https://github.com/osrf/gazebo/search?q=author%3Apeci1&type=Commits`

Selected contributions to Gazebo-related projects:

- **osrf/gazebo**: Open source robotics simulator. 14 pull requests, 9 accepted.
- **osrf/gazebo_tutorials**: Tutorials for gazebo. 7 pull requests, 3 accepted.
- **ignitionrobotics/ign_common**: Ignition Common is a component in the ignition framework, a set of libraries designed to rapidly develop robot applications. 3 pull requests, 2 accepted.
- **ignitionrobotics/ign_math**: Ignition math library is a general purpose math library for robot applications. 2 pull requests, 2 accepted.

Selected maintained ROS packages:

- **peci1/robot_body_filter**: Filters the robot's body out of laser scans or point clouds. Released in ROS Melodic.
- **peci1/rosbash_params**: Tools for writing ros-node-like bash scripts. Released in ROS Kinetic and Melodic.
- **peci1/dynamic_robot_state_publisher**: Improved ROS robot_state_publisher which can update the robot model via dynamic_reconfigure. Released in ROS Kinetic and Melodic.
- **peci1/tf2_server**: An upgraded tf2_ros/buffer_server which allows clients to request only subparts of the TF tree. Released in ROS Melodic.
- **tradr-project/tensorflow_ros_cpp**: Catkin-friendly C++ bindings for tensorflow.
- **tradr-project/tf_remapper_cpp**: More efficient version of tf/tf_remap able to handle TFs at kHz with tens of subscribers. Released in ROS Indigo, Lunar, Kinetic and Melodic.
- **tradr-project/static_transform_mux**: A helper node that makes sure everybody knows about all static transforms, even if they are published by multiple publishers. Released in ROS Indigo, Lunar, Kinetic and Melodic.
- **tradr-project/nimbro_network**: ROS network stack: Topic/service transport over unreliable network connections. Fork of *AIS-Bonn/nimbro_network* with many improvements.
- **peci1/gazebo_custom_sensor_preloader**: Gazebo plugin that allows writing custom Gazebo sensors.
- **peci1/gazebo_noisy_depth_camera**: Noisy depth camera sensor for Gazebo.
- **peci1/gazebo_rotating_lidar**: Gazebo Lidar sensor which accounts for scanning plane motion during a single scan.

Appendix D

Bibliography

[Abbeel et al., 2007] Abbeel, P., Coates, A., Quigley, M., and Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. In *Proceedings of the 2006 Conference on Advances in Neural Information Processing Systems 19*, volume 19, page 1. arXiv: 1705.10528.

[Abdolmaleki et al., 2015] Abdolmaleki, A., Lioutikov, R., Peters, J. R., Lau, N., Reis, L. P., and Neumann, G. (2015). Model-Based Relative Entropy Stochastic Search. In *Advances in Neural Information Processing Systems*, pages 3523–3531.

[Achiam et al., 2017] Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained Policy Optimization. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 22–31, Sydney. PMLR. arXiv: 1705.10528.

[Akametalu et al., 2014] Akametalu, A. K., Fisac, J. F., Gillula, J. H., Kaynama, S., Zeilinger, M. N., and Tomlin, C. J. (2014). Reachability-based safe learning with Gaussian processes. In *53rd IEEE Conference on Decision and Control*, pages 1424–1431. IEEE. DOI: 10.1109/CDC.2014.7039601.

[Akrour et al., 2019a] Akrour, R., Pajarinen, J., Neumann, G., and Peters, J. (2019). Projections for approximate policy iteration algorithms. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 181–190, Long Beach, California, USA. PMLR.

[Akrour et al., 2018] Akrour, R., Peters, J., and Neumann, G. (2018). Constraint-Space Projection Direct Policy Search. In *14th European Workshop on Reinforcement Learning (EWRL)*, pages 1–12.

[Akrour et al., 2019b] Akrour, R., Tateo, D., and Peters, J. (2019). Towards Reinforcement Learning of Human Readable Policies. In *1st Workshop on Deep Continuous-Discrete Machine Learning*, Würzburg, Germany. TU Darmstadt.

[Alshiekh et al., 2018] Alshiekh, M., Bloem, R., Ehlers, R. R., Könighofer, B., Niekum, S., and Topcu, U. (2018). Safe Reinforcement Learning via

Shielding. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 2669–2678. AAAI press. arXiv: 1708.08611.

[Amodei et al., 2016] Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete Problems in AI Safety. pages 1–29. arXiv: 1606.06565.

[Andrychowicz et al., 2017] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., Zaremba, W., Abbeel, P., and Zaremba, W. (2017). Hindsight experience replay. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5048–5058. Curran Associates, Inc. arXiv: 1707.01495.

[Argall et al., 2009] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483. DOI: 10.1016/j.robot.2008.10.024.

[Arias, 2012] Arias, S. G. (2012). Finite Element Analysis of Rubber Treads on Tracks to Simulate Wear Development. In *3DS Simulia Comm. Conf.*

[Aslund et al., 2018] Aslund, C. H. R., El Mhamdi, E. M., Guerraoui, R., and Maurer, A. (2018). Virtuously Safe Reinforcement Learning. arXiv: 1805.11447v1.

[Barto et al., 1981] Barto, A. G., Sutton, R. S., and Brouwer, P. S. (1981). Associative search network: A reinforcement learning associative memory. *Biological cybernetics*.

[Belousov and Peters, 2017] Belousov, B. and Peters, J. (2017). f-Divergence constrained policy improvement. arXiv: 1801.00056.

[Berkenkamp, 2019] Berkenkamp, F. (2019). *Safe Exploration in Reinforcement Learning: Theory and Applications in Robotics*. PhD thesis, ETH Zurich.

[Berkenkamp et al., 2016] Berkenkamp, F., Moriconi, R., Schoellig, A. P., and Krause, A. (2016). Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes. In *2016 IEEE 55th Conference on Decision and Control, CDC 2016*. arXiv: 1603.04915, DOI: 10.1109/CDC.2016.7798979.

[Berkenkamp et al., 2017] Berkenkamp, F., Turchetta, M., Schoellig, A. P., and Krause, A. (2017). Safe Model-based Reinforcement Learning with Stability Guarantees. In *Advances in Neural Information Processing Systems*, pages 909–919. arXiv: 1705.08551.

[Bertsekas, 1987] Bertsekas, D. P. (1987). *Dynamic programming: deterministic and stochastic models*. Prentice-Hall.

[Bjelonic et al., 2018] Bjelonic, M., Kottege, N., Homberger, T., Borges, P., Beckerle, P., and Chli, M. (2018). Weaver: Hexapod robot for autonomous navigation on unstructured terrain. *Journal of Field Robotics*, 35(7):1063–1079. DOI: 10.1002/rob.21795.

[Byrd et al., 1999] Byrd, R. H., Hribar, M. E., and Nocedal, J. (1999). An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900.

[Celemin et al., 2019] Celemin, C., Maeda, G., Ruiz-del Solar, J., Peters, J., and Kober, J. (2019). Reinforcement learning of motor skills using Policy Search and human corrective advice. *The International Journal of Robotics Research*, 38(14):1560–1580. DOI: 10.1177/0278364919871998.

[Chen et al., 2020] Chen, X., Chen, S., Xu, T., Yin, B., Peng, J., Mei, X., and Li, H. (2020). S2OMGAN: Shortcut from Remote Sensing Images to Online Maps. arXiv: 2001.07712v1.

[Chernova and Veloso, 2007] Chernova, S. and Veloso, M. (2007). Confidence-based policy learning from demonstration using Gaussian mixture models. In *AAMAS '07 Proceedings*, pages 1–8. ACM Press. DOI: 10.1145/1329125.1329407.

[Christiano et al., 2016] Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., Abbeel, P., and Zaremba, W. (2016). Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model. arXiv: 1610.03518.

[Clavera et al., 2018] Clavera, I., Rothfuss, J., Schulman, J., Fujita, Y., Asfour, T., and Abbeel, P. (2018). Model-Based Reinforcement Learning via Meta-Policy Optimization. In *2nd Conference on Robot Learning (CoRL 2018)*. arXiv: 1809.05214.

[Coraluppi and Marcus, 1999] Coraluppi, S. P. and Marcus, S. I. (1999). Risk-sensitive and minimax control of discrete-time, finite-state Markov decision processes. *Automatica*.

[Couceiro et al., 2019] Couceiro, M. S., Portugal, D., Ferreira, J. F., and Rocha, R. P. (2019). SEMFIRE: Towards a new generation of forestry maintenance multi-robot systems. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 270–276, New York, NY, USA. IEEE. DOI: 10.1109/SII.2019.8700403.

[Cully et al., 2015] Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. B. (2015). Robots that can adapt like animals. *Nature*, 521(7553):503–507. DOI: 10.1038/nature14422.

[Cutler and How, 2015] Cutler, M. and How, J. P. (2015). Efficient reinforcement learning for robots using informative simulated priors. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2605–2612. DOI: 10.1109/ICRA.2015.7139550.

[Daniel et al., 2016] Daniel, C., Neumann, G., Kroemer, O., and Peters, J. (2016). Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, 17:1–50.

[DARPA, 2019] DARPA (2019). DARPA SubT Challenge. `http://subtchallenge.com/`. Accessed: 27/06/2019.

[Delage and Mannor, 2007] Delage, E. and Mannor, S. (2007). Percentile optimization in uncertain Markov decision processes with application to efficient exploration. In *Proceedings of the 24th international conference on Machine learning - ICML '07*, pages 225–232, New York, USA. ACM Press. DOI: 10.1145/1273496.1273525.

[den Hengst et al., 2020] den Hengst, F., Grua, M., El Hassouni, A., and Hoogendoorn, M. (2020). Reinforcement learning for personalization: a systematic literature review. Technical report, IOS Press.

[Dimitrova et al., 2016] Dimitrova, R., Fu, J., and Topcu, U. (2016). Robust optimal policies for Markov decision processes with safety-threshold constraints. *2016 IEEE 55th Conference on Decision and Control, CDC 2016*, pages 7081–7086. DOI: 10.1109/CDC.2016.7799360.

[Drumwright et al., 2010] Drumwright, E., Hsu, J., Koenig, N., and Shell, D. (2010). Extending Open Dynamics Engine for Robotics Simulation. In *Lecture Notes in Artificial Intelligence*, volume 6472 LNAI, pages 38–50. DOI: 10.1007/978-3-642-17319-6_7.

[Ertle et al., 2012] Ertle, P., Tokic, M., Cubek, R., Voos, H., and Soffker, D. (2012). Towards learning of safety knowledge from human demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5394–5399. IEEE. DOI: 10.1109/IROS.2012.6385714.

[Ewerton et al., 2019] Ewerton, M., Arenz, O., Maeda, G., Koert, D., Kolev, Z., Takahashi, M., and Peters, J. (2019). Learning Trajectory Distributions for Assisted Teleoperation and Path Planning. *Frontiers in Robotics and AI*. DOI: 10.3389/frobt.2019.00089.

[Ewerton et al., 2016a] Ewerton, M., Maeda, G., Kollegger, G., Wiemeyer, J., and Peters, J. (2016). Incremental imitation learning of context-dependent motor skills. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 351–358. IEEE. DOI: 10.1109/HUMANOIDS.2016.7803300.

[Ewerton et al., 2016b] Ewerton, M., Maeda, G., Neumann, G., Kisner, V., Kollegger, G., Wiemeyer, J., and Peters, J. (2016). Movement primitives with multiple phase parameters. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 201–206. IEEE. DOI: 10.1109/ICRA.2016.7487134.

[Fan and Li, 2019] Fan, J. and Li, W. (2019). Safety-guided Deep Reinforcement Learning via Online Gaussian Process Estimation. pages 1–13. arXiv: 1903.02526v2.

[Fankhauser, 2018] Fankhauser, P. (2018). *Perceptive Locomotion for Legged Robots in Rough Terrain*. PhD thesis, ETH Zurich.

[Faria, 2018] Faria, J. M. (2018). Machine Learning Safety: An Overview. In *Proceedings of the 26th Safety-Critical Systems Symposium*, York. Safety-Critical Systems Club.

[Fernandez-Gauna et al., 2015] Fernandez-Gauna, B., Graña, M., Lopez-Guede, J. M., Etxeberria-Agiriano, I., and Ansoategui, I. (2015). Reinforcement Learning endowed with safe veto policies to learn the control of Linked-Multicomponent Robotic Systems. *Information Sciences*, 317(April):25–47. DOI: 10.1016/j.ins.2015.04.005.

[Ferretti and Girelli, 1999] Ferretti, G. and Girelli, R. (1999). Modelling and simulation of an agricultural tracked vehicle. *Journal of Terramechanics*, 36(3):139–158. DOI: 10.1016/S0022-4898(99)00004-X.

[Finn et al., 2017a] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *34th International Conference on Machine Learning, ICML 2017*. arXiv: 1703.03400.

[Finn et al., 2016a] Finn, C., Levine, S., and Abbeel, P. (2016). Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. arXiv: 1603.00448.

[Finn et al., 2016b] Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., Abbeel, P., Xin Yu Tan, Yan Duan, Darrell, T., Levine, S., Abbeel, P., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. (2016). Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE. arXiv: 1509.06113, DOI: 10.1109/ICRA.2016.7487173.

[Finn et al., 2017b] Finn, C., Yu, T., Fu, J., Abbeel, P., and Levine, S. (2017). Generalizing skills with semi-supervised reinforcement learning. *5th International Conference on Learning Representations, ICLR 2017*, pages 1–11. arXiv: 1612.00429.

[Fisac et al., 2019] Fisac, J. F., Akametalu, A. K., Zeilinger, M. N., Kaynama, S., Gillula, J., and Tomlin, C. J. (2019). A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752. DOI: 10.1109/TAC.2018.2876389.

[Frans et al., 2018] Frans, K., Ho, J., Chen, X., Abbeel, P., and Schulman, J. (2018). Meta learning shared hierarchies. In *6th International Conference on Learning Representations, ICLR 2018*, pages 1–11. arXiv: 1710.09767.

[Frye and Feige, 2019] Frye, C. and Feige, I. (2019). PARENTING: Safe Reinforcement Learning from Human Input. arXiv: 1902.06766v1.

[García and Fernández, 2012] García, J. and Fernández, F. (2012). Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564. DOI: 10.1613/jair.3761.

[García and Fernández, 2015] García, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16:1437–1480.

[Garcia Polo and Rebollo, 2011] Garcia Polo, F. J. and Rebollo, F. F. (2011). Safe reinforcement learning in high-risk tasks through policy improvement. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 76–83. IEEE. DOI: 10.1109/ADPRL.2011.5967356.

[Ge et al., 2019] Ge, Y., Zhu, F., Ling, X., and Liu, Q. (2019). Safe Q-Learning Method Based on Constrained Markov Decision Processes. *IEEE Access*, 7:165007–165017. DOI: 10.1109/ACCESS.2019.2952651.

[Geibel, 2001] Geibel, P. (2001). Reinforcement learning with bounded risk. In *IEEE International Conference on Machine Learning*, pages 162–169.

[Geibel and Wysotzki, 2011] Geibel, P. and Wysotzki, F. (2011). Risk-Sensitive Reinforcement Learning Applied to Control under Constraints. *Journal Of Artificial Intelligence Research*, 24:81–108. arXiv: 1109.2147, DOI: 10.1613/jair.1666.

[Gillula and Tomlin, 2011] Gillula, J. H. and Tomlin, C. J. (2011). Guaranteed safe online learning of a bounded system. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2979–2984. IEEE. DOI: 10.1109/IROS.2011.6095101.

[Goodfellow et al., 2014] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Networks. *Advances in Neural Information Processing Systems 27*, pages 2672–2680. arXiv: 1406.2661.

[Haarnoja et al., 2018] Haarnoja, R., Hartikainen, K., Abbeel, P., and Levine, S. (2018). Latent space policies for hierarchical reinforcement learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, pages 1851–1860, Stockholm, Sweden. PMLR. arXiv: 1804.02808.

[Haarnoja et al., 2017] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2017). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *Deep Reinforcement Learning Symposium*. arXiv: 1801.01290.

[Haarnoja et al., 2019] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. (2019). Soft Actor-Critic Algorithms and Applications. arXiv: 1812.05905.

[Hadfield-Menell et al., 2017] Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S., and Dragan, A. D. (2017). Inverse reward design. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6766–6775. Curran Associates Inc. arXiv: 1711.02827.

[Han et al., 2019] Han, H., Liu, Z., Liu, H., and Qiao, J. (2019). Knowledge-Data-Driven Model Predictive Control for a Class of Nonlinear Systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 6(1):1–13. DOI: 10.1109/TSMC.2019.2937002.

[Hans et al., 2008] Hans, A., Schneegaß, D., Schäfer, A. M., and Udluft, S. (2008). Safe exploration for reinforcement learning. In *ESANN 2008 Proceedings, 16th European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning*, pages 143–148.

[Hasanbeig et al., 2020] Hasanbeig, M., Abate, A., and Kroening, D. (2020). Cautious Reinforcement Learning with Logical Constraints. In An, B., Yorke-Smith, N., Seghrouchni, A. E. F., and Sukthank, G., editors, *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems(AAMAS 2020)*, Auckland. International Foundation for Autonomous Agents and Multiagent Systems. arXiv: 2002.12156v2.

[Hasanbeig et al., 2019] Hasanbeig, M., Kroening, D., and Abate, A. (2019). Towards Verifiable and Safe Model-Free Reinforcement Learning. In Gigante, N., Mar, F., and Orlandini, A., editors, *Proceedings of the1st Workshop on Artificial Intelligence and Formal Verification, Logics, Automata and Synthesis (OVERLAY)*, pages 1–9, Rende. CEUR Workshop Proceedings.

[Hausknecht and Stone, 2015] Hausknecht, M. and Stone, P. (2015). Deep Recurrent Q-Learning for Partially Observable MDPs. arXiv: 1507.06527.

[Heger, 1994] Heger, M. (1994). Consideration of risk in reinforcement learning. In *11th International Machine Learning Conference*.

[Held et al., 2017] Held, D., McCarthy, Z., Zhang, M., Shentu, F., and Abbeel, P. (2017). Probabilistically safe policy transfer. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5798–5805. IEEE. arXiv: 1705.05394, DOI: 10.1109/ICRA.2017.7989680.

[Hernández-Orallo et al., 2019] Hernández-Orallo, J., Martínez-Plumed, F., Avin, S., and Ó hÉigeartaigh, S. (2019). Surveying Safety-relevant AI Characteristics. In Espinoza, H., Ó hÉigeartaigh, S., Huang, X., Hernández-Orallo, J., and Castillo-Effen, M., editors, *Proceedings of the AAAI Workshop on Artificial Intelligence Safety 2019*, Honolulu. CEUR Workshop Proceedings.

[Hoffmann and Pfeifer, 2018] Hoffmann, M. and Pfeifer, R. (2018). Robots as powerful allies for the study of embodied cognition from the bottom up. In Newen, A., De Bruin, L., and Gallagher, S., editors, *The Oxford Handbook of 4E Cognition*, pages 840–862. Oxford University Press. DOI: 10.1093/oxfordhb/9780198735410.013.45.

[Holz et al., 2019] Holz, D., Genter, K., Saad, M., and von Stryk, O. (2019). *RoboCup 2018: Robot World Cup XXII*, volume 11374 of *Lecture Notes in Artificial Intelligence*. Springer International Publishing, Cham. DOI: 10.1007/978-3-030-27544-0.

[Howard, 1960] Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. Technology Press of Massachusetts Institute of Technology.

[Huang et al., 2019] Huang, S. H., Zambelli, M., Kay, J., Martins, M. F., Tassa, Y., Pilarski, P. M., and Hadsell, R. (2019). Learning Gentle Object Manipulation with Curiosity-Driven Deep Reinforcement Learning. arXiv: 1903.08542v1.

[Hwangbo et al., 2019] Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., and Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872. DOI: 10.1126/scirobotics.aau5872.

[Jacoff et al., 2008] Jacoff, A., Downs, A., Virts, A., and Messina, E. (2008). Stepfield pallets: Repeatable terrain for evaluating robot mobility. In *Performance Metrics for Intelligent Systems (PerMIS) Workshop*. DOI: 10.1145/1774674.1774680.

[Janarthanan et al., 2011] Janarthanan, B., Padmanabhan, C., and Sujatha, C. (2011). Lateral dynamics of single unit skid-steered tracked vehicle. *International Journal of Automotive Technology*, 12(6):865–875. DOI: 10.1007/s12239-011-0099-4.

[Janarthanan et al., 2012] Janarthanan, B., Padmanabhan, C., and Sujatha, C. (2012). Longitudinal dynamics of a tracked vehicle: Simulation and experiment. *Journal of Terramechanics*, 49(2):63–72. DOI: 10.1016/j.jterra.2011.11.001.

[Jansen et al., 2019] Jansen, N., Könighofer, B., Junges, S., Serban, A. C., and Bloem, R. (2019). Safe Reinforcement Learning via Probabilistic Shields. arXiv: 1807.06096v2.

[Junges et al., 2016] Junges, S., Jansen, N., Dehnert, C., Topcu, U., and Katoen, J.-P. (2016). Safety-Constrained Reinforcement Learning for MDPs. In Chechik, M. and Raskin, J.-F., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, pages 130–146, Berlin, Heidelberg. Springer. DOI: 10.1007/978-3-662-49674-9.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285. DOI: 10.1613/jair.301.

[Kahn et al., 2020] Kahn, G., Abbeel, P., and Levine, S. (2020). BADGR: An Autonomous Self-Supervised Learning-Based Navigation System. arXiv: 2002.05700.

[Kahn et al., 2018] Kahn, G., Villaflor, A., Ding, B., Abbeel, P., and Levine, S. (2018). Self-Supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5129–5136. IEEE. arXiv: 1709.10489, DOI: 10.1109/ICRA.2018.8460655.

[Kahn et al., 2017a] Kahn, G., Villaflor, A., Pong, V., Abbeel, P., and Levine, S. (2017). Uncertainty-Aware Reinforcement Learning for Collision Avoidance. arXiv: 1702.01182.

[Kahn et al., 2017b] Kahn, G., Zhang, T., Levine, S., and Abbeel, P. (2017). PLATO: Policy Learning using Adaptive Trajectory Optimization. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3342–3349. IEEE. arXiv: 1603.00622, DOI: 10.1109/ICRA.2017.7989379.

[Kang et al., 2019] Kang, K., Belkhale, S., Kahn, G., Abbeel, P., and Levine, S. (2019). Generalization through Simulation: Integrating Simulated and Real Data into Deep Reinforcement Learning for Vision-Based Autonomous Flight. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6008–6014. IEEE. arXiv: 1902.03701, DOI: 10.1109/ICRA.2019.8793735.

[Kaufman et al., 2008] Kaufman, D. M., Sueda, S., James, D. L., and Pai, D. K. (2008). Staggered Projections for Frictional Contact in Multibody Systems. *ACM Trans. Graph. (SIGGRAPH)*, 27(5):164:1–164:11. DOI: 10.1145/1409060.1409117.

[Kenwright and Morgan, 2012] Kenwright, B. and Morgan, G. (2012). Practical Introduction to Rigid Body Linear Complementary Problem (LCP) Constraint Solvers. In *Algorithmic and Architectural Gaming Design*, pages 159–201. IGI Global. DOI: 10.4018/978-1-4666-1634-9.ch008.

[Kharuzin et al., 2017] Kharuzin, S., Ivanov, A., and Shmakov, O. (2017). Autonomous Control System for a Vehicle with Actively Articulated Frame. In *Proceedings of 28th DAAAM International Symposium on Intelligent Manufacturing and Automation*, pages 0759–0766. DOI: 10.2507/28th.daaam.proceedings.107.

[Kim et al., 2012] Kim, D., Kim, K.-E., and Poupart, P. (2012). Cost-Sensitive Exploration in Bayesian Reinforcement Learning. In *Proceedings of Neural Information Processing Systems (NIPS)*.

[Klink et al., 2019] Klink, P., Abdulsamad, H., Belousov, B., and Peters, J. (2019). Self-Paced Contextual Reinforcement Learning. arXiv: 1910.02826.

[Koert et al., 2016] Koert, D., Maeda, G., Lioutikov, R., Neumann, G., and Peters, J. (2016). Demonstration based trajectory optimization for generalizable robot motions. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 515–522. IEEE. DOI: 10.1109/HUMANOIDS.2016.7803324.

[Krakovna et al., 2019] Krakovna, V., Orseau, L., Martic, M., and Legg, S. (2019). Penalizing side effects using stepwise relative reachability. In Espinoza, H., Yu, H., Huang, X., Lecue, F., Chen, C., Hernández-Orallo, J., Ó hÉigeartaigh, S., and Mallah, R., editors, *Proceedings of the Workshop on Artificial Intelligence Safety 2019*, Macau. CEUR Workshop Proceedings. arXiv: 1806.01186.

[Kruijff-Korbayová et al., 2015] Kruijff-Korbayová, I., Colas, F., Gianni, M., Pirri, F., de Greeff, J., Hindriks, K., Neerincx, M., Ögren, P., Svoboda, T., and Worst, R. (2015). TRADR Project: Long-Term Human-Robot Teaming for Robot Assisted Disaster Response. *KI - Künstliche Intelligenz*, 29(2):193–201. DOI: 10.1007/s13218-015-0352-5.

[Kruijff-Korbayova et al., 2016] Kruijff-Korbayova, I., Freda, L., Gianni, M., Ntouskos, V., Hlavac, V., Kubelka, V., Zimmermann, E., Surmann, H., Dulic, K., Rottner, W., and Gissi, E. (2016). Deployment of ground and aerial robots in earthquake-struck Amatrice in Italy (brief report). In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 278–279. IEEE. DOI: 10.1109/SSRR.2016.7784314.

[Kubelka and Reinstein, 2012] Kubelka, V. and Reinstein, M. (2012). Complementary filtering approach to orientation estimation using inertial sensors only. In *2012 IEEE International Conference on Robotics and Automation*, pages 599–605. IEEE. DOI: 10.1109/ICRA.2012.6224564.

[Kupcsik et al., 2017] Kupcsik, A., Deisenroth, M. P., Peters, J., Loh, A. P., Vadakkepat, P., and Neumann, G. (2017). Model-based contextual policy search for data-efficient generalization of robot skills. *Artificial Intelligence*, 247:415–439. DOI: 10.1016/j.artint.2014.11.005.

[Lee et al., 2019] Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. (2019). Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model. pages 1–19. arXiv: 1907.00953.

[Leike et al., 2017] Leike, J., Martic, M., Krakovna, V., Ortega, P. A., Everitt, T., Lefrancq, A., Orseau, L., and Legg, S. (2017). AI Safety Gridworlds. arXiv: 1711.09883v2.

[Lempitsky et al., 2018] Lempitsky, V., Vedaldi, A., and Ulyanov, D. (2018). Deep Image Prior. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9446–9454. IEEE. DOI: 10.1109/CVPR.2018.00984.

[Levine and Koltun, 2013] Levine, S. and Koltun, V. (2013). Guided Policy Search. In *30th International Conference on Machine Learning, ICML 2013*, volume 28, pages 1038–1046. PMLR.

[Li et al., 2020] Li, A. C., Pinto, L., and Abbeel, P. (2020). Generalized Hindsight for Reinforcement Learning. arXiv: 2002.11708.

[Liu et al., 2019] Liu, Z., Ploeger, K., Stark, S., Rueckert, E., and Peters, J. (2019). Learning walk and trot from the same objective using different types of exploration. arXiv: 1904.12336.

[Lu et al., 2017] Lu, T., Zinkevich, M., Boutilier, C., Roy, B., and Schuurmans, D. (2017). Safe Exploration for Identifying Linear Systems via Robust Optimization. pages 1–23. arXiv: 1711.11165v1.

[Luck et al., 2014] Luck, K. S., Neumann, G., Berger, E., Peters, J., and Amor, H. B. (2014). Latent space policy search for robotics. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1434–1440. IEEE. DOI: 10.1109/IROS.2014.6942745.

[Ma and Perkins, 2006] Ma, Z. D. and Perkins, N. C. (2006). A super-element of track-wheel-terrain interaction for dynamic simulation of tracked vehicles. *Multibody System Dynamics*, 15(4):347–368. DOI: 10.1007/s11044-005-9001-3.

[Mannucci et al., 2018] Mannucci, T., van Kampen, E.-J., de Visser, C., and Chu, Q. (2018). Safe Exploration Algorithms for Reinforcement Learning Controllers. *IEEE Transactions on Neural Networks and Learning Systems*, 29(4):1069–1081. DOI: 10.1109/TNNLS.2017.2654539.

[Marco et al., 2017] Marco, A., Berkenkamp, F., Hennig, P., Schoellig, A. P., Krause, A., Schaal, S., and Trimpe, S. (2017). Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1557–1563. IEEE. DOI: 10.1109/ICRA.2017.7989186.

[Martinez et al., 2015] Martinez, D., Alenya, G., and Torras, C. (2015). Safe robot execution in model-based reinforcement learning. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6422–6427. IEEE. DOI: 10.1109/IROS.2015.7354295.

[Martínez et al., 2005] Martínez, J. L., Mandow, A., Morales, J., Pedraza, S., and García-Cerezo, A. (2005). Approximating Kinematics for Tracked Mobile Robots. *The International Journal of Robotics Research*, 24(10):867–878. DOI: 10.1177/0278364905058239.

[Martínez-Tenor et al., 2018] Martínez-Tenor, A., Fernández-Madrigal, J. A., Cruz-Martín, A., and González-Jiménez, J. (2018). Towards a common implementation of reinforcement learning for multiple robotic tasks. *Expert Systems with Applications*, 100:246–259. DOI: 10.1016/j.eswa.2017.11.011.

[Mason, 2018] Mason, G. R. (2018). *Safe Reinforcement Learning Using Formally Verified Abstract Policies*. PhD thesis, University of York.

[McKinnon and Schoellig, 2017] McKinnon, C. D. and Schoellig, A. P. (2017). Learning multimodal models for robot dynamics online with a mixture of Gaussian process experts. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 322–328. IEEE. DOI: 10.1109/ICRA.2017.7989041.

[Mendonca et al., 2019] Mendonca, R., Gupta, A., Kralev, R., Abbeel, P., Levine, S., and Finn, C. (2019). Guided Meta-Policy Search. In Wallach, H. M., Larochelle, H., Beygelzimer, A., D'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 9653–9664. arXiv: 1904.00956.

[Mihatsch and Neuneier, 2002] Mihatsch, O. and Neuneier, R. (2002). Risk-sensitive reinforcement learning. *Machine learning*, 49(2-3):267–290. DOI: 10.1023/A:1017940631555.

[Mirzaei et al., 2017] Mirzaei, H., Fathollahi, M., and Givargis, T. (2017). OPEB: Open physical environment benchmark for artificial intelligence. In *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*, pages 1–6, New York, NY, USA. IEEE. arXiv: 1707.00790, DOI: 10.1109/RTSI.2017.8065980.

[Mirzaei Buini, 2018] Mirzaei Buini, H. (2018). *Control System Design Automation Using Reinforcement Learning*. PhD thesis, University of California, Irvine.

[Moldovan, 2016] Moldovan, T. M. (2016). *Safety, Risk Awareness and Exploration in Reinforcement Learning*. PhD thesis, University of California at Berkeley, Berkley.

[Moldovan and Abbeel, 2012] Moldovan, T. M. and Abbeel, P. (2012). Safe Exploration in Markov Decision Processes. In *Proceedings of the 29th International Conference on Machine Learning*. arXiv: 1205.4810.

[Montgomery et al., 2017] Montgomery, W., Ajay, A., Finn, C., Abbeel, P., and Levine, S. (2017). Reset-free guided policy search: Efficient deep reinforcement learning with stochastic initial states. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3373–3380. IEEE. DOI: 10.1109/ICRA.2017.7989383.

[Montgomery and Levine, 2016] Montgomery, W. and Levine, S. (2016). Guided policy search via approximate mirror descent. In *Advances in Neural Information Processing Systems*. arXiv: 1607.04614.

[Moskvin et al., 2020] Moskvin, I., Lavrenov, R., and Magid, E. (2020). Modelling a crawler robot using wheels as pseudo-tracks: model complexity vs performance. Technical report, Kazan Federal University.

[Motoi et al., 2019a] Motoi, N., Kobayashi, M., and Masaki, R. (2019). Remote Control Method for Mobile Robot Based on Force Feedback Generated using Collision Prediction Map. *IEEJ Journal of Industry Applications*, 8(4):727–735. DOI: 10.1541/ieejjia.8.727.

[Motoi et al., 2019b] Motoi, N., Masaki, R., and Kobayashi, M. (2019). Remote Control Method with Force Assist Based on Collision Prediction Calculated from Each Turning Radius in Mobile Robot. In *2019 IEEE International Conference on Mechatronics (ICM)*, pages 477–482, New York, NY, USA. IEEE. DOI: 10.1109/ICMECH.2019.8722932.

[Munk et al., 2016] Munk, J., Kober, J., and Babuska, R. (2016). Learning state representation for deep actor-critic control. In *2016 IEEE 55th Conference on Decision and Control, CDC 2016*. DOI: 10.1109/CDC.2016.7798980.

[Muratore et al., 2019] Muratore, F., Gienger, M., and Peters, J. (2019). Assessing Transferability from Simulation to Reality for Reinforcement Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–12. arXiv: 1907.04685, DOI: 10.1109/tpami.2019.2952353.

[Muratore et al., 2018] Muratore, F., Treede, F., Gienger, M., and Peters, J. (2018). Domain Randomization for Simulation-Based Policy Optimization with Transferability Assessment. In *2nd Conference on Robot Learning (CoRL 2018)*, pages 1–14.

[Nagabandi et al., 2019] Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. (2019). Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–17. arXiv: 1803.11347.

[Nair et al., 2018] Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Overcoming Exploration in Reinforcement Learning with Demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE. arXiv: 1709.10089, DOI: 10.1109/ICRA.2018.8463162.

[Nass et al., 2019] Nass, D., Belousov, B., and Peters, J. (2019). Entropic Risk Measure in Policy Search. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1101–1106. IEEE. arXiv: 1906.09090, DOI: 10.1109/IROS40897.2019.8967699.

[Nilim and El Ghaoui, 2005] Nilim, A. and El Ghaoui, L. (2005). Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Operations Research*, 53(5):780–798. DOI: 10.1287/opre.1050.0216.

[Paolo et al., 2017] Paolo, G., Tai, L., and Liu, M. (2017). Towards continuous control of flippers for a multi-terrain robot using deep reinforcement learning. arXiv: 1709.08430.

[Papini et al., 2019] Papini, M., Pirotta, M., and Restelli, M. (2019). Smoothing Policies and Safe Policy Gradients. arXiv: 1905.03231v1.

[Parisi et al., 2017] Parisi, S., Ramstedt, S., and Peters, J. (2017). Goal-driven dimensionality reduction for reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. DOI: 10.1109/IROS.2017.8206334.

[Parisi et al., 2019a] Parisi, S., Tangkaratt, V., Peters, J., and Khan, M. E. (2019). TD-regularized actor-critic methods. *Machine Learning*, 108(8-9):1467–1501. DOI: 10.1007/s10994-019-05788-0.

[Parisi et al., 2019b] Parisi, S., Tateo, D., Hensel, M., D'Eramo, C., Peters, J., and Pajarinen, J. (2019). Long-Term Visitation Value for Deep Exploration in Sparse Reward Reinforcement Learning. arXiv: 2001.00119.

[Pazis and Lagoudakis, 2011] Pazis, J. and Lagoudakis, M. G. (2011). Reinforcement learning in multidimensional continuous action spaces. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 97–104. IEEE. DOI: 10.1109/ADPRL.2011.5967381.

[Pecka, 2017] Pecka, M. (2017). Gazebo pull request "Added support for tracked vehicles". `https://osrf-migration.github.io/gazebo-gh-pages/#!/osrf/gazebo/pull-requests/2652`.

[Pecka and Svoboda, 2014] Pecka, M. and Svoboda, T. (2014). Safe Exploration Techniques for Reinforcement Learning – An Overview. In Hodicky, J., editor, *Modelling and Simulation for Autonomous Systems*, Lecture Notes in Computer Science, pages 357–375, Rome, Italy. Springer. DOI: 10.1007/978-3-319-13823-7.

[Pecka et al., 2018] Pecka, M., Zimmermann, K., Petrlik, M., and Svoboda, T. (2018). Data-Driven Policy Transfer With Imprecise Perception Simulation. *IEEE Robotics and Automation Letters*, 3(4):3916–3921. DOI: 10.1109/LRA.2018.2857927.

[Pecka et al., 2017a] Pecka, M., Zimmermann, K., Reinstein, M., and Svoboda, T. (2017). Controlling Robot Morphology From Incomplete Measurements. *IEEE Transactions on Industrial Electronics*, 64(2):1773–1782. DOI: 10.1109/TIE.2016.2580125.

[Pecka et al., 2015] Pecka, M., Zimmermann, K., and Svoboda, T. (2015). Safe Exploration for Reinforcement Learning in Real Unstructured Environments. In Wohlhart, P. and Lepetit, V., editors, *CVWW '15: Proceedings of the 20th Computer Vision Winter Workshop*, pages 85–93, Graz, Austria. Graz University of Technology. DOI: 10.3217/978-3-85125-388-7.

[Pecka et al., 2016] Pecka, M., Zimmermann, K., and Svoboda, T. (2016). Autonomous Flipper Control with Safety Constraints. In *2016 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 2889–2894, Daejon, South Korea. IEEE. DOI: 10.1109/IROS.2016.7759447.

[Pecka et al., 2017b] Pecka, M., Zimmermann, K., and Svoboda, T. (2017). Fast simulation of vehicles with non-deformable tracks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6414–6419, Vancouver, Canada. IEEE. DOI: 10.1109/IROS.2017.8206546.

[Peng et al., 2018a] Peng, X. B., Abbeel, P., Levine, S., and Van De Panne, M. (2018). DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4). DOI: 10.1145/3197517.3201311.

[Peng et al., 2018b] Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810. IEEE. arXiv: 1710.06537, DOI: 10.1109/ICRA.2018.8460528.

[Peng et al., 2016] Peng, X. B., Berseth, G., and van de Panne, M. (2016). Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics*, 35(4):1–12. DOI: 10.1145/2897824.2925881.

[Peng et al., 2019] Peng, X. B., Kanazawa, A., Toyer, S., Abbeel, P., and Levine, S. (2019). Variational Discriminator Bottleneck: Improving Imitation Learning, Inverse RL, and GANs by Constraining Information Flow. In *7th International Conference on Learning Representations, ICLR 2019*, pages 1–27. arXiv: 1810.00821.

[Peters et al., 2010] Peters, J., Ulling, K., and Al, Y. (2010). Relative Entropy Policy Search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, pages 1607–1612.

[Petříček et al., 2019] Petříček, T., Šalanský, V., Zimmermann, K., and Svoboda, T. (2019). Simultaneous exploration and segmentation for search and rescue. *Journal of Field Robotics*, 36(4):696–709. DOI: 10.1002/rob.21847.

[Prashanth, 2014] Prashanth, L. (2014). Policy Gradients for CVaR-Constrained MDPs. In *International Conference Algorithmic Learning Theory*, pages 155–169. DOI: 10.1007/978-3-319-11662-4_12.

[Puterman, 1994] Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition.

[Rathi et al., 2020] Rathi, M., Ferraro, P., and Russo, G. (2020). Driving Reinforcement Learning with Models. arXiv: 1911.04400v2.

[Ray et al., 2019] Ray, A., Achiam, J., and Amodei, D. (2019). Benchmarking Safe Exploration in Deep Reinforcement Learning. In *Deep Reinforcement Learning Workshop 2019*.

[Ren et al., 2018] Ren, Z., Dong, D., Li, H., and Chen, C. (2018). Self-Paced Prioritized Curriculum Learning with Coverage Penalty in Deep Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2216–2226. DOI: 10.1109/TNNLS.2018.2790981.

[Ross and Bagnell, 2012] Ross, S. and Bagnell, J. (2012). Agnostic system identification for model-based reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK. arXiv: 1203.1007.

[Rothfuss et al., 2019] Rothfuss, J., Asfour, T., Lee, D., Clavera, I., and Abbeel, P. (2019). PrOMP: Proximal meta-policy search. In *7th International Conference on Learning Representations, ICLR 2019*, pages 1–25. OpenReview.net. arXiv: 1810.06784.

[Rouček et al., 2020] Rouček, T., Pecka, M., Čížek, P., Petříček, T., Bayer, J., Šalanský, V., Heřt, D., Petrlík, M., Báča, T., Spurný, V., Pomerleau, F., Kubelka, V., Faigl, J., Zimmermann, K., Saska, M., Svoboda, T., and Krajník, T. (2020). Darpa subterranean challenge: Multi-robotic exploration of underground environments. In Mazal, J., Fagiolini, A., and Vasik, P., editors, *Modelling and Simulation for Autonomous Systems*, pages 274–290, Cham. Springer International Publishing. DOI: 10.1007/978-3-030-43890-6_22.

[Rubinstein and Hitron, 2004] Rubinstein, D. and Hitron, R. (2004). A detailed multi-body model for dynamic simulation of off-road tracked vehicles. *Journal of Terramechanics*, 41(2-3):163–173. DOI: 10.1016/j.jterra.2004.02.004.

[Rusu et al., 2016] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive Neural Networks. arXiv: 1606.04671.

[Šalanský et al., 2016] Šalanský, V., Kubelka, V., Zimmermann, K., Reinstein, M., and Svoboda, T. (2016). Touching without vision: terrain perception in sensory deprived environments. In Čehovin, L., Mandeljc, R., and Štruc, V., editors, *Computer Vision Winter Workshop*, Rimske Toplice. CVWW.

[Schneider, 1996] Schneider, J. G. (1996). Exploiting model uncertainty estimates for safe dynamic control learning. *Neural Information Processing Systems*, 9:1047–1053.

[Schulman, 2016] Schulman, J. (2016). *Optimizing Expectations: From Deep Reinforcement Learning to Stochastic Computation Graphs*. PhD thesis, University of California, Berkeley.

[Schulman et al., 2017] Schulman, J., Chen, X., and Abbeel, P. (2017). Equivalence Between Policy Gradients and Soft Q-Learning. pages 1–15. arXiv: 1704.06440.

[Schulman et al., 2016] Schulman, J., Moritz, P., and Levine, S. (2016). High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *4th International Conference on Learning Representations, ICLR 2016*.

[Sermanet et al., 2018] Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., and Brain, G. (2018). Time-Contrastive Networks: Self-Supervised Learning from Video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE. arXiv: 1704.06888, DOI: 10.1109/ICRA.2018.8462891.

[Shen et al., 2019] Shen, Y., Yang, K., Yuan, Y., and Liu, S. C. (2019). Meta Reinforcement Learning with Distribution of Exploration Parameters Learned by Evolution Strategies. arXiv: 1812.11314v2.

[Singh et al., 2014] Singh, S., Jadhav, B. D., and Krishna, K. M. (2014). Posture control of a three-segmented tracked robot with torque minimization during step climbing. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4200–4207. IEEE. DOI: 10.1109/ICRA.2014.6907470.

[Smith et al., 2005] Smith, R. et al. (2005). Open Dynamics Engine. `http://www.ode.org/ode.html`.

[Sokolov et al., 2017a] Sokolov, M., Afanasyev, I., Klimchik, A., and Mavridis, N. (2017). HyperNEAT-based flipper control for a crawler robot motion in 3D simulation environment. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2652–2656, Macau SAR. IEEE. DOI: 10.1109/ROBIO.2017.8324819.

[Sokolov et al., 2017b] Sokolov, M., Afanasyev, I., Lavrenov, R., Sagitov, A., Sabirova, L., and Magid, E. (2017). Modelling a crawler-type UGV for urban search and rescue in Gazebo environment. *Proceedings of International Conference on Artificial Life and Robotics*, 22:360–363. DOI: 10.5954/ICAROB.2017.GS2-4.

[Sokolov et al., 2016] Sokolov, M., Lavrenov, R., Gabdullin, A., Afanasyev, I., and Magid, E. (2016). 3D modelling and simulation of a crawler robot in ROS/Gazebo. In *Proceedings of the 4th International Conference on Control, Mechatronics and Automation - ICCMA '16*, pages 61–65, New York, NY, USA. ACM Press. DOI: 10.1145/3029610.3029641.

[Stark et al., 2019] Stark, S., Peters, J., and Rueckert, E. (2019). Experience Reuse with Probabilistic Movement Primitives. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1210–1217. IEEE. arXiv: 1908.03936, DOI: 10.1109/IROS40897.2019.8968545.

[Sui et al., 2015] Sui, Y., Gotovos, A., Burdick, J. W., and Krause, A. (2015). Safe exploration for optimization with Gaussian processes. In *32nd International Conference on Machine Learning, ICML 2015*.

[Sun et al., 2018a] Sun, W., Bagnell, J. A., and Boots, B. (2018). Truncated Horizon Policy Search: Combining Reinforcement Learning & Imitation Learning. In *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net. arXiv: 1805.11240.

[Sun et al., 2018b] Sun, W., Gordon, G. J., Boots, B., and Bagnell, J. A. (2018). Dual Policy Iteration. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 7059–7069. Neural information processing systems foundation. arXiv: 1805.10755.

[Sünderhauf et al., 2018] Sünderhauf, N., Brock, O., Scheirer, W., Hadsell, R., Fox, D., Leitner, J., Upcroft, B., Abbeel, P., Burgard, W., Milford, M., and Corke, P. (2018). The limits and potentials of deep learning for robotics. *International Journal of Robotics Research*, 37(4-5):405–420. DOI: 10.1177/0278364918770733.

[Tamar et al., 2017] Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. (2017). Value iteration networks. In *IJCAI International Joint Conference on Artificial Intelligence*. arXiv: 1602.02867, DOI: 10.24963/ijcai.2017/700.

[Tangkaratt et al., 2017] Tangkaratt, V., Van Hoof, H., Parisi, S., Neumann, G., Peters, J., and Sugiyama, M. (2017). Policy search with high-dimensional context variables. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 2632–2638.

[Tedrake et al., 2009] Tedrake, R., Jackowski, Z., Cory, R., Roberts, J. W., and Hoburg, W. (2009). Learning to fly like a bird. In *14th International Symposium on Robotics Research*, Lucerne, Switzerland.

[Tobin et al., 2017] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE. arXiv: 1703.06907, DOI: 10.1109/IROS.2017.8202133.

[Tosatto et al., 2020] Tosatto, S., Carvalho, J., Abdulsamad, H., and Peters, J. (2020). A Nonparametric Off-Policy Policy Gradient. arXiv: 2001.02435.

[Tosatto et al., 2019] Tosatto, S., D'Eramo, C., Pajarinen, J., Restelli, M., and Peters, J. (2019). Exploration Driven by an Optimistic Bellman Equation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE. DOI: 10.1109/IJCNN.2019.8851736.

[Transeth et al., 2009] Transeth, A. A., Pettersen, K. Y., and Liljebäck, P. (2009). A survey on snake robot modeling and locomotion. *Robotica*, 27(7):999–1015. DOI: 10.1017/S0263574709005414.

[Trinkle and Pang, 1997] Trinkle, J. and Pang, J.-S. (1997). On Dynamic Multi-Rigid-Body Contact Problems with Coulomb Friction. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 77:267–279. DOI: 10.1002/zamm.19970770411.

[Turchetta et al., 2016] Turchetta, M., Berkenkamp, F., and Krause, A. (2016). Safe Exploration in Finite Markov Decision Processes with Gaussian Processes. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4312–4320, Barcelona, Spain. Curran Associates Inc. arXiv: 1606.04753, DOI: 10.5555/3157382.3157579.

[Turner et al., 2019] Turner, A. M., Hadfield-Menell, D., and Tadepalli, P. (2019). Conservative agency. In Espinoza, H., Yu, H., Huang, X., Lecue, F., Chen, C., Hernández-Orallo, J., Ó hÉigeartaigh, S., and Mallah, R., editors, *CEUR Workshop Proceedings*, volume 2419, Macao. CEUR Workshop Proceedings. arXiv: 1902.09725.

[Turner et al., 2020] Turner, A. M., Hadfield-Menell, D., and Tadepalli, P. (2020). Conservative Agency via Attainable Utility Preservation. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 385–391, New York, NY, USA. ACM. DOI: 10.1145/3375627.3375851.

[Tzeng et al., 2015a] Tzeng, E., Devin, C., Hoffman, J., Finn, C., Abbeel, P., Levine, S., Saenko, K., and Darrell, T. (2015). Adapting Deep Visuomotor Representations with Weak Pairwise Constraints. arXiv: 1511.07111.

[Tzeng et al., 2015b] Tzeng, E., Hoffman, J., Darrell, T., and Saenko, K. (2015). Simultaneous Deep Transfer Across Domains and Tasks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4068–4076. IEEE. DOI: 10.1109/ICCV.2015.463.

[Uchibe and Doya, 2007] Uchibe, E. and Doya, K. (2007). Constrained reinforcement learning from intrinsic and extrinsic rewards. In *IEEE 6th International Conference on Development and Learning*, pages 163–168. DOI: 10.1109/DEVLRN.2007.4354030.

[Urain and Peters, 2019] Urain, J. and Peters, J. (2019). Generalized Multiple Correlation Coefficient as a Similarity Measurement between Trajectories. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1363–1369. IEEE. arXiv: 1906.09802, DOI: 10.1109/IROS40897.2019.8967884.

[van Hoof et al., 2017] van Hoof, H., Tanneberg, D., and Peters, J. (2017). Generalized exploration in policy search. *Machine Learning*, 106(9-10):1705–1724. DOI: 10.1007/s10994-017-5657-1.

[Vemula et al., 2020] Vemula, A., Oza, Y., Bagnell, J. A., and Likhachev, M. (2020). Planning and Execution using Inaccurate Models with Provable Guarantees. arXiv: 2003.04394.

[Vezzani et al., 2019] Vezzani, G., Gupta, A., Natale, L., and Abbeel, P. (2019). Learning latent state representation for speeding up exploration. arXiv: 1905.12621.

[Wabersich and Zeilinger, 2018] Wabersich, K. P. and Zeilinger, M. N. (2018). Linear Model Predictive Safety Certification for Learning-Based Control. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 7130–7135. IEEE. arXiv: 1911.01832, DOI: 10.1109/CDC.2018.8619829.

[Wainwright and Eckersley, 2020] Wainwright, C. L. and Eckersley, P. (2020). SafeLife 1.0: Exploring Side Effects in Complex Environments. In Espinoza, H., Hernandez-Orallo, J., Chen, X. C., Ó hÉigeartaigh, S., Huang, X., Castillo-Effen, M., Mallah, R., and McDermid, J., editors, *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI 2020)*, New York. CEUR Workshop Proceedings. arXiv: 1912.01217.

[Wallin et al., 2013] Wallin, M., Aboubakr, A. K., Jayakumar, P., Letherwood, M. D., Gorsich, D. J., Hamed, A., and Shabana, A. A. (2013). A comparative study of joint formulations: Application to multibody system tracked vehicles. *Nonlinear Dynamics*, 74(3):783–800. DOI: 10.1007/s11071-013-1005-6.

[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4):279–292. DOI: 10.1007/BF00992698.

[Williams and Baird, 1993] Williams, R. J. and Baird, L. C. (1993). Tight performance bounds on greedy policies based on imperfect value functions. Technical report, Northeastern University, College of Computer Science.

[Wu et al., 2019] Wu, Y., Yan, W., Kurutach, T., Pinto, L., and Abbeel, P. (2019). Learning to Manipulate Deformable Objects without Demonstrations. arXiv: 1910.13439.

[Wulfmeier et al., 2017] Wulfmeier, M., Bewley, A., and Posner, I. (2017). Addressing appearance change in outdoor robotics with adversarial domain adaptation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1551–1558. IEEE. arXiv: 1703.01461, DOI: 10.1109/IROS.2017.8205961.

[Xu et al., 2017] Xu, W., Verl, A. W., and Ohishi, K. (2017). Motion Control for Novel Emerging Robotic Devices and Systems. *IEEE Transactions on Industrial Electronics*, 64(2):1623–1625. DOI: 10.1109/TIE.2016.2617831.

[Yamakawa and Watanabe, 2004] Yamakawa, J. and Watanabe, K. (2004). A spatial motion analysis model of tracked vehicles with torsion bar type suspension. *Journal of Terramechanics*, 41(2-3):113–126. DOI: 10.1016/j.jterra.2004.02.007.

[Yuan et al., 2019] Yuan, Y., Wang, L., and Schwertfeger, S. (2019). Configuration-Space Flipper Planning for Rescue Robots. In *2019*

IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pages 37–42. University of York, IEEE. arXiv: 1905.02984, DOI: 10.1109/SSRR.2019.8848978.

[Yuan et al., 2020] Yuan, Y., Xu, Q., and Schwertfeger, S. (2020). Configuration-Space Flipper Planning on 3D Terrain. arXiv: 1909.07612v2.

[Zhang et al., 2019a] Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M. J., and Levine, S. (2019). SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97, pages 7444–7453. PMLR. arXiv: 1808.09105.

[Zhang et al., 2019b] Zhang, Y., Clavera, I., Tsai, B., and Abbeel, P. (2019). Asynchronous Methods for Model-Based Reinforcement Learning. arXiv: 1910.12453.

[Zhu et al., 2017] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251. IEEE. arXiv: 1703.10593, DOI: 10.1109/ICCV.2017.244.

[Zimmermann et al., 2014] Zimmermann, K., Zuzanek, P., Reinstein, M., and Hlavac, V. (2014). Adaptive Traversability of unknown complex terrain with obstacles for mobile robots. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5177–5182. IEEE. DOI: 10.1109/ICRA.2014.6907619.

[Zimmermann et al., 2015] Zimmermann, K., Zuzanek, P., Reinstein, M., Petricek, T., and Hlavac, V. (2015). Adaptive traversability of partially occluded obstacles. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3959–3964. IEEE. DOI: 10.1109/ICRA.2015.7139752.

[Zun Yuan et al., 2019] Zun Yuan, L., Hasanbeig, M., Abate, A., and Kroening, D. (2019). Modular Deep Reinforcement Learning with Temporal Logic Specifications. arXiv: 1909.11591v2.

Appendix E

Publications

Articles in Peer-reviewed Journals

- Pecka, M., Zimmermann, K., Reinstein, M., and Svoboda, T. (2017). Controlling Robot Morphology From Incomplete Measurements. *IEEE Transactions on Industrial Electronics*, 64(2):1773–1782. DOI: 10.1109/TIE.2016.2580125
- Pecka, M., Zimmermann, K., Petrlik, M., and Svoboda, T. (2018). Data-Driven Policy Transfer With Imprecise Perception Simulation. *IEEE Robotics and Automation Letters*, 3(4):3916–3921. DOI: 10.1109/LRA.2018.2857927

Conference Proceedings

- Pecka, M. and Svoboda, T. (2014). Safe Exploration Techniques for Reinforcement Learning – An Overview. In Hodicky, J., editor, *Modelling and Simulation for Autonomous Systems*, Lecture Notes in Computer Science, pages 357–375, Rome, Italy. Springer. DOI: 10.1007/978-3-319-13823-7
- Pecka, M., Zimmermann, K., and Svoboda, T. (2015). Safe Exploration for Reinforcement Learning in Real Unstructured Environments. In Wohlhart, P. and Lepetit, V., editors, *CVWW '15: Proceedings of the 20th Computer Vision Winter Workshop*, pages 85–93, Graz, Austria. Graz University of Technology. DOI: 10.3217/978-3-85125-388-7
- Pecka, M., Zimmermann, K., and Svoboda, T. (2016). Autonomous Flipper Control with Safety Constraints. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2889–2894, Daejon, South Korea. IEEE. DOI: 10.1109/IROS.2016.7759447
- Pecka, M., Zimmermann, K., and Svoboda, T. (2017). Fast simulation of vehicles with non-deformable tracks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6414–6419, Vancouver, Canada. IEEE. DOI: 10.1109/IROS.2017.8206546

Appendix F

Responses to Author's Publications

- Pecka, M. and Svoboda, T. (2014). Safe Exploration Techniques for Reinforcement Learning – An Overview. In Hodicky, J., editor, *Modelling and Simulation for Autonomous Systems*, Lecture Notes in Computer Science, pages 357–375, Rome, Italy. Springer. DOI: 10.1007/978-3-319-13823-7

    - Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete Problems in AI Safety. pages 1–29. arXiv: 1606.06565
    - Berkenkamp, F., Turchetta, M., Schoellig, A. P., and Krause, A. (2017). Safe Model-based Reinforcement Learning with Stability Guarantees. In *Advances in Neural Information Processing Systems*, pages 909–919. arXiv: 1705.08551
    - Junges, S., Jansen, N., Dehnert, C., Topcu, U., and Katoen, J.-P. (2016). Safety-Constrained Reinforcement Learning for MDPs. In Chechik, M. and Raskin, J.-F., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, pages 130–146, Berlin, Heidelberg. Springer. DOI: 10.1007/978-3-662-49674-9
    - Turchetta, M., Berkenkamp, F., and Krause, A. (2016). Safe Exploration in Finite Markov Decision Processes with Gaussian Processes. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4312–4320, Barcelona, Spain. Curran Associates Inc. arXiv: 1606.04753, DOI: 10.5555/3157382.3157579
    - Alshiekh, M., Bloem, R., Ehlers, R. R., Könighofer, B., Niekum, S., and Topcu, U. (2018). Safe Reinforcement Learning via Shielding. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 2669–2678. AAAI press. arXiv: 1708.08611
    - Mannucci, T., van Kampen, E.-J., de Visser, C., and Chu, Q. (2018). Safe Exploration Algorithms for Reinforcement Learning Controllers. *IEEE Transactions on Neural Networks and Learning Systems*, 29(4):1069–1081. DOI: 10.1109/TNNLS.2017.2654539

- Martínez-Tenor, A., Fernández-Madrigal, J. A., Cruz-Martín, A., and González-Jiménez, J. (2018). Towards a common implementation of reinforcement learning for multiple robotic tasks. *Expert Systems with Applications*, 100:246–259. DOI: 10.1016/j.eswa.2017.11.011
- Ren, Z., Dong, D., Li, H., and Chen, C. (2018). Self-Paced Prioritized Curriculum Learning with Coverage Penalty in Deep Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2216–2226. DOI: 10.1109/TNNLS.2018.2790981
- Ge, Y., Zhu, F., Ling, X., and Liu, Q. (2019). Safe Q-Learning Method Based on Constrained Markov Decision Processes. *IEEE Access*, 7:165007–165017. DOI: 10.1109/ACCESS.2019.2952651
- Dimitrova, R., Fu, J., and Topcu, U. (2016). Robust optimal policies for Markov decision processes with safety-threshold constraints. *2016 IEEE 55th Conference on Decision and Control, CDC 2016*, pages 7081–7086. DOI: 10.1109/CDC.2016.7799360
- Fernandez-Gauna, B., Graña, M., Lopez-Guede, J. M., Etxeberria-Agiriano, I., and Ansoategui, I. (2015). Reinforcement Learning endowed with safe veto policies to learn the control of Linked-Multicomponent Robotic Systems. *Information Sciences*, 317(April):25–47. DOI: 10.1016/j.ins.2015.04.005
- Turner, A. M., Hadfield-Menell, D., and Tadepalli, P. (2020). Conservative Agency via Attainable Utility Preservation. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 385–391, New York, NY, USA. ACM. DOI: 10.1145/3375627.3375851
- Held, D., McCarthy, Z., Zhang, M., Shentu, F., and Abbeel, P. (2017). Probabilistically safe policy transfer. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5798–5805. IEEE. arXiv: 1705.05394, DOI: 10.1109/ICRA.2017.7989680
- Faria, J. M. (2018). Machine Learning Safety: An Overview. In *Proceedings of the 26th Safety-Critical Systems Symposium*, York. Safety-Critical Systems Club
- Leike, J., Martic, M., Krakovna, V., Ortega, P. A., Everitt, T., Lefrancq, A., Orseau, L., and Legg, S. (2017). AI Safety Gridworlds. arXiv: 1711.09883v2
- Jansen, N., Könighofer, B., Junges, S., Serban, A. C., and Bloem, R. (2019). Safe Reinforcement Learning via Probabilistic Shields. arXiv: 1807.06096v2
- Krakovna, V., Orseau, L., Martic, M., and Legg, S. (2019). Penalizing side effects using stepwise relative reachability. In Espinoza, H., Yu, H., Huang, X., Lecue, F., Chen, C., Hernández-Orallo, J., Ó hÉigeartaigh, S., and Mallah, R., editors, *Proceedings of the Workshop on Artificial Intelligence Safety 2019*, Macau. CEUR Workshop Proceedings. arXiv: 1806.01186

- Huang, S. H., Zambelli, M., Kay, J., Martins, M. F., Tassa, Y., Pilarski, P. M., and Hadsell, R. (2019). Learning Gentle Object Manipulation with Curiosity-Driven Deep Reinforcement Learning. arXiv: 1903.08542v1
- Papini, M., Pirotta, M., and Restelli, M. (2019). Smoothing Policies and Safe Policy Gradients. arXiv: 1905.03231v1
- Lu, T., Zinkevich, M., Boutilier, C., Roy, B., and Schuurmans, D. (2017). Safe Exploration for Identifying Linear Systems via Robust Optimization. pages 1–23. arXiv: 1711.11165v1
- Hernández-Orallo, J., Martínez-Plumed, F., Avin, S., and Ó hÉigeartaigh, S. (2019). Surveying Safety-relevant AI Characteristics. In Espinoza, H., Ó hÉigeartaigh, S., Huang, X., Hernández-Orallo, J., and Castillo-Effen, M., editors, *Proceedings of the AAAI Workshop on Artificial Intelligence Safety 2019*, Honolulu. CEUR Workshop Proceedings
- Fan, J. and Li, W. (2019). Safety-guided Deep Reinforcement Learning via Online Gaussian Process Estimation. pages 1–13. arXiv: 1903.02526v2
- Hasanbeig, M., Abate, A., and Kroening, D. (2020). Cautious Reinforcement Learning with Logical Constraints. In An, B., Yorke-Smith, N., Seghrouchni, A. E. F., and Sukthank, G., editors, *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems(AAMAS 2020)*, Auckland. International Foundation for Autonomous Agents and Multiagent Systems. arXiv: 2002.12156v2
- Rathi, M., Ferraro, P., and Russo, G. (2020). Driving Reinforcement Learning with Models. arXiv: 1911.04400v2
- Mason, G. R. (2018). *Safe Reinforcement Learning Using Formally Verified Abstract Policies*. PhD thesis, University of York
- Shen, Y., Yang, K., Yuan, Y., and Liu, S. C. (2019). Meta Reinforcement Learning with Distribution of Exploration Parameters Learned by Evolution Strategies. arXiv: 1812.11314v2
- den Hengst, F., Grua, M., El Hassouni, A., and Hoogendoorn, M. (2020). Reinforcement learning for personalization: a systematic literature review. Technical report, IOS Press
- Berkenkamp, F. (2019). *Safe Exploration in Reinforcement Learning: Theory and Applications in Robotics*. PhD thesis, ETH Zurich
- Wainwright, C. L. and Eckersley, P. (2020). SafeLife 1.0: Exploring Side Effects in Complex Environments. In Espinoza, H., Hernandez-Orallo, J., Chen, X. C., Ó hÉigeartaigh, S., Huang, X., Castillo-Effen, M., Mallah, R., and McDermid, J., editors, *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI 2020)*, New York. CEUR Workshop Proceedings. arXiv: 1912.01217

- Frye, C. and Feige, I. (2019). PARENTING: Safe Reinforcement Learning from Human Input. arXiv: 1902.06766v1
- Hasanbeig, M., Kroening, D., and Abate, A. (2019). Towards Verifiable and Safe Model-Free Reinforcement Learning. In Gigante, N., Mar, F., and Orlandini, A., editors, *Proceedings of the1st Workshop on Artificial Intelligence and Formal Verification, Logics, Automata and Synthesis (OVERLAY)*, pages 1–9, Rende. CEUR Workshop Proceedings
- Turner, A. M., Hadfield-Menell, D., and Tadepalli, P. (2019). Conservative agency. In Espinoza, H., Yu, H., Huang, X., Lecue, F., Chen, C., Hernández-Orallo, J., Ó hÉigeartaigh, S., and Mallah, R., editors, *CEUR Workshop Proceedings*, volume 2419, Macao. CEUR Workshop Proceedings. arXiv: 1902.09725
- Aslund, C. H. R., El Mhamdi, E. M., Guerraoui, R., and Maurer, A. (2018). Virtuously Safe Reinforcement Learning. arXiv: 1805.11447v1

- Pecka, M., Zimmermann, K., Petrlik, M., and Svoboda, T. (2018). Data-Driven Policy Transfer With Imprecise Perception Simulation. *IEEE Robotics and Automation Letters*, 3(4):3916–3921. DOI: 10.1109/LRA.2018.2857927
  - Chen, X., Chen, S., Xu, T., Yin, B., Peng, J., Mei, X., and Li, H. (2020). S2OMGAN: Shortcut from Remote Sensing Images to Online Maps. arXiv: 2001.07712v1

- Pecka, M., Zimmermann, K., Reinstein, M., and Svoboda, T. (2017). Controlling Robot Morphology From Incomplete Measurements. *IEEE Transactions on Industrial Electronics*, 64(2):1773–1782. DOI: 10.1109/TIE.2016.2580125
  - Bjelonic, M., Kottege, N., Homberger, T., Borges, P., Beckerle, P., and Chli, M. (2018). Weaver: Hexapod robot for autonomous navigation on unstructured terrain. *Journal of Field Robotics*, 35(7):1063–1079. DOI: 10.1002/rob.21795
  - Sokolov, M., Afanasyev, I., Klimchik, A., and Mavridis, N. (2017). HyperNEAT-based flipper control for a crawler robot motion in 3D simulation environment. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2652–2656, Macau SAR. IEEE. DOI: 10.1109/ROBIO.2017.8324819
  - Mirzaei, H., Fathollahi, M., and Givargis, T. (2017). OPEB: Open physical environment benchmark for artificial intelligence. In *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*, pages 1–6, New York, NY, USA. IEEE. arXiv: 1707.00790, DOI: 10.1109/RTSI.2017.8065980
  - Motoi, N., Kobayashi, M., and Masaki, R. (2019). Remote Control Method for Mobile Robot Based on Force Feedback Generated using

Collision Prediction Map. *IEEJ Journal of Industry Applications*, 8(4):727–735. DOI: 10.1541/ieejjia.8.727

- Xu, W., Verl, A. W., and Ohishi, K. (2017). Motion Control for Novel Emerging Robotic Devices and Systems. *IEEE Transactions on Industrial Electronics*, 64(2):1623–1625. DOI: 10.1109/TIE.2016.2617831
- Motoi, N., Masaki, R., and Kobayashi, M. (2019). Remote Control Method with Force Assist Based on Collision Prediction Calculated from Each Turning Radius in Mobile Robot. In *2019 IEEE International Conference on Mechatronics (ICM)*, pages 477–482, New York, NY, USA. IEEE. DOI: 10.1109/ICMECH.2019.8722932
- Couceiro, M. S., Portugal, D., Ferreira, J. F., and Rocha, R. P. (2019). SEMFIRE: Towards a new generation of forestry maintenance multi-robot systems. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 270–276, New York, NY, USA. IEEE. DOI: 10.1109/SII.2019.8700403
- Mirzaei Buini, H. (2018). *Control System Design Automation Using Reinforcement Learning*. PhD thesis, University of California, Irvine
- Yuan, Y., Xu, Q., and Schwertfeger, S. (2020). Configuration-Space Flipper Planning on 3D Terrain. arXiv: 1909.07612v2
- Kharuzin, S., Ivanov, A., and Shmakov, O. (2017). Autonomous Control System for a Vehicle with Actively Articulated Frame. In *Proceedings of 28th DAAAM International Symposium on Intelligent Manufacturing and Automation*, pages 0759–0766. DOI: 10.2507/28th.daaam.proceedings.107
- Han, H., Liu, Z., Liu, H., and Qiao, J. (2019). Knowledge-Data-Driven Model Predictive Control for a Class of Nonlinear Systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 6(1):1–13. DOI: 10.1109/TSMC.2019.2937002

- Pecka, M., Zimmermann, K., and Svoboda, T. (2015). Safe Exploration for Reinforcement Learning in Real Unstructured Environments. In Wohlhart, P. and Lepetit, V., editors, *CVWW '15: Proceedings of the 20th Computer Vision Winter Workshop*, pages 85–93, Graz, Austria. Graz University of Technology. DOI: 10.3217/978-3-85125-388-7

  - Mannucci, T., van Kampen, E.-J., de Visser, C., and Chu, Q. (2018). Safe Exploration Algorithms for Reinforcement Learning Controllers. *IEEE Transactions on Neural Networks and Learning Systems*, 29(4):1069–1081. DOI: 10.1109/TNNLS.2017.2654539
  - Martinez, D., Alenya, G., and Torras, C. (2015). Safe robot execution in model-based reinforcement learning. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6422–6427. IEEE. DOI: 10.1109/IROS.2015.7354295

  - Paolo, G., Tai, L., and Liu, M. (2017). Towards continuous control of flippers for a multi-terrain robot using deep reinforcement learning. arXiv: 1709.08430
  - Mason, G. R. (2018). *Safe Reinforcement Learning Using Formally Verified Abstract Policies*. PhD thesis, University of York

- Pecka, M., Zimmermann, K., and Svoboda, T. (2016). Autonomous Flipper Control with Safety Constraints. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2889–2894, Daejon, South Korea. IEEE. DOI: 10.1109/IROS.2016.7759447

  - Moskvin, I., Lavrenov, R., and Magid, E. (2020). Modelling a crawler robot using wheels as pseudo-tracks: model complexity vs performance. Technical report, Kazan Federal University
  - Mason, G. R. (2018). *Safe Reinforcement Learning Using Formally Verified Abstract Policies*. PhD thesis, University of York
  - Yuan, Y., Wang, L., and Schwertfeger, S. (2019). Configuration-Space Flipper Planning for Rescue Robots. In *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 37–42. University of York, IEEE. arXiv: 1905.02984, DOI: 10.1109/SSRR.2019.8848978
  - Sokolov, M., Afanasyev, I., Lavrenov, R., Sagitov, A., Sabirova, L., and Magid, E. (2017). Modelling a crawler-type UGV for urban search and rescue in Gazebo environment. *Proceedings of International Conference on Artificial Life and Robotics*, 22:360–363. DOI: 10.5954/ICAROB.2017.GS2-4
  - Sokolov, M., Afanasyev, I., Klimchik, A., and Mavridis, N. (2017). HyperNEAT-based flipper control for a crawler robot motion in 3D simulation environment. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2652–2656, Macau SAR. IEEE. DOI: 10.1109/ROBIO.2017.8324819
  - Paolo, G., Tai, L., and Liu, M. (2017). Towards continuous control of flippers for a multi-terrain robot using deep reinforcement learning. arXiv: 1709.08430

- Pecka, M., Zimmermann, K., and Svoboda, T. (2017). Fast simulation of vehicles with non-deformable tracks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6414–6419, Vancouver, Canada. IEEE. DOI: 10.1109/IROS.2017.8206546

  - Sokolov, M., Afanasyev, I., Klimchik, A., and Mavridis, N. (2017). HyperNEAT-based flipper control for a crawler robot motion in 3D simulation environment. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2652–2656, Macau SAR. IEEE. DOI: 10.1109/ROBIO.2017.8324819