



AFRL-AFOSR-VA-TR-2022-0032

Robust Adaptive Autonomy in Contested Environments

Girish Chowdhary
OKLAHOMA STATE UNIVERSITY
401 WHITEHURST HALL
STILLWATER, OK, 74078
USA

09/28/2021
Final Technical Report

DISTRIBUTION A: Distribution approved for public release.
--

Air Force Research Laboratory
Air Force Office of Scientific Research
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 28-09-2021		2. REPORT TYPE Final		3. DATES COVERED (From - To) 01 Jul 2015 - 30 Jun 2020	
4. TITLE AND SUBTITLE Robust Adaptive Autonomy in Contested Environments				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER FA9550-15-1-0146	
				5c. PROGRAM ELEMENT NUMBER 61102F	
6. AUTHOR(S) Girish Chowdhary				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) OKLAHOMA STATE UNIVERSITY 401 WHITEHURST HALL STILLWATER, OK 74078 USA				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AF Office of Scientific Research 875 N. Randolph St. Room 3112 Arlington, VA 22203				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/AFOSR RTA2	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-AFOSR-VA-TR-2022-0032	
12. DISTRIBUTION/AVAILABILITY STATEMENT A Distribution Unlimited: PB Public Release					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The research undertaken in this YIP proposal seeks to develop theoretical underpinnings and practical algorithms for Robust Adaptive Autonomy in Contested Environments (RAACE) for mixed manned-unmanned aerial teams. The specific objectives of this YIP proposal are: • Create a new computationally and spatiotemporally scalable class of BNP and BNP-inspired predictive models and algorithms for nonstationary stochastic processes with both smooth and discrete transitions. • Create Nonstationary model based reinforcement learning (N-MBRL) algorithms for risk-averse adaptive decision making in contested and time-varying environments. • Develop algorithms to enable autonomous agents to infer latent human sub-goals directly from human demonstrations and feedback in contested environments. • Validate the developed algorithms through data-simulations, multi-agent simulations, and flight experimentation involving humans and machines working together.					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 145	19a. NAME OF RESPONSIBLE PERSON HAL GREENWALD
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (Include area code) 703-588-8441

Final Report for AFOSR YIP FA9550-15-1-0146 *Robust
Adaptive Autonomy in Contested Environments* PI:
Girish Chowdhary

October 14, 2020

Contents

1 Introduction	6
1.1 Personnel and Facilities Development	8
1.2 Major Changes: Move to University of Illinois at Urbana Champaign in 2016	8
1.3 On the Organization of This Report	9
1.4 Publications	9
I Progress in the Year 2018-end of project	11
1 Multi-agent Aerial Monitoring of Moving Convoys using Elliptical Orbits	11
1.1 Introduction	11
1.2 Related Work	12
2 Problem Description and Notation	13
3 Preliminaries	14
4 Vector-field guidance law	15
5 Linear speed profile for aerial agents	16
5.1 Parameter computation for the aerial agents	16
5.2 Speed computation for the aerial agents	17
6 Elliptical orbit computation	19
7 Cooperation for equi-parametric formation	20
8 Simulations and Experiments	22
II Progress in the Year 2017-18	27

1 Using Reinforcement Learning as Helper to Humans in Difficult Multi-Agent Environments	27
2 Harnessing Adversarial Attacks to Improve Robustness of Deep Reinforcement Learning	32
2.1 Introduction	32
2.1.1 Related Work	33
2.2 Background	34
2.2.1 Adversarial attack on Deep Learning Network classifiers (Fast Signed Gradient Method)	34
2.2.2 Deep Q learning (DQN) and Deep Double Q Learning (DDQN)	35
2.2.3 Radial Basis Function based Q learning	35
2.2.4 Deep Deterministic Policy Gradient (DDPG)	35
2.3 Method	35
2.3.1 Adversarial Attack	35
2.3.2 Adversarial Training through Robust Control framework	40
2.4 Experimental setup	40
2.4.1 DDQN	40
2.4.2 RBF	41
2.4.3 DDPG	41
2.4.4 Adversarial Training	42
2.5 Robust Training Colormap for Cartpole	42
2.6 Results	42
2.6.1 Adversarial Attack	42
2.6.2 Robust Training	43
2.7 Conclusion	45
3 Cross Domain Transfer Learning using Target Apprentice	45
3.1 Transfer Learning with Target Apprentice (TA-TL)	46
3.1.1 Phase I: Learning in the Source task	47
3.1.2 Phase II: Intertask Mapping	47
3.1.3 Phase III: Transfer Learning through policy adaptation	47
3.1.4 Algorithm: TA-TL	48
3.1.5 Analysis	49
3.2 Experiments & Results	50
3.3 Same-Domain Transfer	50
3.3.1 Grid World to Windy Grid World	50
3.3.2 Inverted Pendulum (IP) to time-varying IP	51
3.4 Cross-Domain Transfer	53
3.5 Cart-Pole to Bicycle	53
3.5.1 Mountain Car (MC) to Inverted Pendulum (IP)	54
3.6 Negative transfer	55
4 Results Connecting Evolving Gaussian Processes to Koopman Operator Theory & Dynamic Mode Decomposition	55
4.1 Koopman Operator Theory	56
4.2 Dynamic Mode Decomposition	56
4.3 Theoretical Connection between Koopman & E-GP	57
4.4 Experimental Connection between DMD & E-GP	57

5 Path Planning for Mobile Inference of Spatiotemporally Evolving Systems	59
Using Evolving Gaussian Processes	59
5.1 Spectral Analysis of Evolving Gaussian Process Model and Resulting Algorithms	59
5.2 k -Invariant Subspaces Clustering	60
5.3 Scoring Paths	60
5.4 Generating High-Scoring Random Paths	61
5.5 Experimental Results	61
5.5.1 Simple Synthetic Examples	61
5.5.2 Fluid Flow Past a Cylinder	62
6 A Dynamic Risk Form of Entropic Value at Risk	63
6.1 Introduction	63
6.2 Preliminaries	63
6.2.1 Static Risk	64
6.2.2 Stochastic Ordering	66
6.2.3 Dynamic Risk	66
6.3 Dynamic Risk Form of EVaR	67
6.4 Discussion	68
III Progress in the Year 2016-17	70
1 Nonparametric Kernel Observer Model and Inference Algorithms for Spatiotemporally Evolving Systems	70
1.1 Kernel Observer to Model Nonlinear Fluid Motion	70
1.1.1 Experimental Results: Computational Flow Dynamics Data	70
1.2 Random Sensor Placement	74
1.2.1 Theoretical Results: Random Sensor Placement	74
1.2.2 Simulation Results: Random sensor placement	77
2 Harnessing Adversarial Attacks on Deep Reinforcement Learning for Improving Robustness	79
2.1 Introduction	79
2.2 Adversarial Attack	80
2.2.1 Naive adversarial attack	80
2.2.2 Gradient based adversarial attack	81
2.2.3 SGD based attack	83
2.3 Robust Control framework	83
2.4 Results	83
2.4.1 Adversarial Attack	85
2.4.2 Robust Training	85
2.5 Ongoing and Future Work	86
3 Human Inspired Reinforcement Learning	86
4 Cross Domain Transfer Learning using Target Apprenticeship	87
4.1 Transfer Learning with Target Apprenticeship (TA-TL)	88
4.1.1 Phase I: Learning in the Source task	89
4.1.2 Phase II: Intertask Mapping	89
4.1.3 Phase III: Transfer Learning through policy adaptation	89

4.1.4	Algorithm: TA-TL	89
4.1.5	Analysis	91
5	Experiments & Results	92
5.1	Same-Domain Transfer	93
5.1.1	Grid World to Windy Grid World	93
5.1.2	Inverted Pendulum (IP) to time-varying IP	94
5.2	Cross-Domain Transfer	95
5.3	Cart-Pole to Bicycle	96
5.3.1	Mountain Car (MC) to Inverted Pendulum (IP)	96
5.4	Negative transfer	97
6	GP regression in presence of non-Gaussian measurement noise	98
6.1	Main Contribution: Meta-GP	98
6.2	Online Posterior Predictive Distribution for arbitrary likelihoods	98
6.3	GP Approximation of the Non-Gaussian Likelihood	99
6.4	Inference with Meta-GP	100
6.5	Implementation	101
6.6	Classification	101
6.7	Regression Results	102
6.7.1	Regression	102
6.7.2	Two-Dimensional Function	103
6.7.3	Boston Housing Data	103
6.7.4	Multimodal Likelihood	104
6.8	Classification Results	104
6.8.1	USPS Data Set	104
6.8.2	MNIST Data Set	104
6.8.3	CIFAR-10 Data Set	105
IV	Progress in the Year 2015-16	106
1	Nonparametric Kernel Observer Model and Inference Algorithms for Spatiotemporally Evolving Systems	106
1.1	Kernel Observer	106
1.2	Preliminaries on Rational Canonical Structures	108
1.3	Design of Measurement Operator K using Rational Canonical Structure theorem	108
1.4	Main Results	109
1.5	Insights into Theoretical Results	112
1.6	Experimental Results	112
1.6.1	Comparison With Nonstationary Kernel Methods	112
1.6.2	Prediction of Global Ocean Surface Temperature	113
2	Uninformed-to-Informed Exploration: When have you Learned Enough to Predict Informative Change?	115
2.1	Probabilistic Inequalities	116
2.2	Lévy Process	117
2.3	Formulation: Exploration as Multi-Play N-Armed Restless Bandits	117
2.3.1	Multi-Play N-Armed Restless Bandit Formulation	118
2.3.2	Modeling the Information Gain	118

2.3.3	Bi-Homogeneous Poisson Exposure Process Model	119
2.4	Probabilistic Bounds for Uninformed-to-Informed Exploration using the bi-Pep Model	120
2.5	Information Gain Dataset Prediction Results	122
2.6	Informed Exploration Results	123
2.7	Uninformed-to-Informed Exploration Results	124
3	Chance-Constrained Optimization for Decision-making in Presence of Risk	126
4	Ongoing work in Human-Machine collaborative Autonomy	128
4.1	Learning human-implicit task subgoals from human demonstrations	129
4.2	Modeling subgoal interdependence and Inference of current task	130
4.3	Dog fight domain	132
4.4	Collaborative Task Execution: Intent Aware Shared Control	133
5	Ongoing work in robust inference and Multi-Agent Simulations	134
5.1	Multi-Agent Simulation Environment MAGE	134

1 Introduction

The research undertaken in this YIP proposal seeks to develop theoretical underpinnings and practical algorithms for Robust Adaptive Autonomy in Contested Environments (RAACE) for mixed manned-unmanned aerial teams.

The specific objectives of this YIP proposal are:

- Create a new computationally and spatiotemporally scalable class of BNP and BNP-inspired predictive models and algorithms for nonstationary stochastic processes with both smooth and discrete transitions.
- Create Nonstationary model based reinforcement learning (N-MBRL) algorithms for risk-averse adaptive decision making in contested and time-varying environments.
- Develop algorithms to enable autonomous agents to infer latent human subgoals directly from human demonstrations and feedback in contested environments.
- Validate the developed algorithms through data-simulations, multi-agent simulations, and flight experimentation involving humans and machines working together.

Towards that objective, the following contributions have resulted since the first year of this YIP project (2016)

1. Creating algorithms to enable a team of UAS to monitor a moving Convoy: As a capstone activity, we developed a novel scheme that will let a single remote user monitor a convoy by delegating the task of surrounding the convoy to an algorithm. Our approach enables the surveillance of a dynamic ground convoy, moving along a nonlinear trajectory, by aerial agents that maintain a uniformly spaced formation on a time-varying elliptical orbit encompassing the convoy. Elliptical orbits are used as they are more economical than circular orbits for circum-navigating the group of targets in the moving convoy. This includes an algorithm for computing feasible elliptical orbits, a vector guidance law for agent motion along the desired orbit, and a cooperative scheme to control the speeds of the aerial agents and quickly achieve the desired formation. The proposed scheme achieves these tasks while accounting for the linear and angular speed constraints of the aerial agent which is modelled using unicycle kinematics. The scheme was validated through simulations and actual experiments, with ground robots as the convoy, and a team of quadrotors as the aerial agents, in a motion capture environment. We relied on this work on an assumption of known bounds on the velocity of the ground agents. We will pursue extensions on this work when the ground vehicle convoy is not collaborating and actively trying to evade.
2. Using RL to help human operators make decisions in partially observed multi-agent environment: We demonstrated that RL can be used to help humans make decisions in partially observed environments. Partially observed environments with multiple agents are difficult for human operators to understand and act upon. Our goal is to develop policies for monitoring agents such that a full picture of the environment can be predicted with partial observations.

This ongoing work is described in Section [1](#)

3. Nonparametric Kernel Observers: We developed a new nonparemetric machine learning model and inference algorithms for inference over domains that change with space and time. Fundamental insights relating to minimal number of sensors required were developed. One

of the surprising findings was that, the rational canonical decompositions, a field that has had much impact on linear systems theory, can be utilized to lend fundamental insights into how to monitor domains that change over space and time with minimal sensors. The results are of significance in designing sensor networks and mobile sensing strategies for monitoring dynamically changing contested environments.

4. Making unobservable spaces observable so that humans can make sense of multi-agent partially observable scenarios: We utilized kernel observers to create algorithms that create control policies for mobile sensors (UAS) that learn to predict a spatiotemporally evolving function with only partial measurements. This work is foundational to creating algorithms for path planning of UAS in contested and partially observed environment so that a full picture of the environment can be formed using sparse and asynchronous measurements. **This work is described in Section 4**
5. Adversarial Reinforcement Learning: With the advent of deep learning in the last two years, our goal here was to assess the robustness of deep learning algorithms in the presence of synthetically injected noise. Our hypothesis is that incorporation of this noise during the training phase will improve the robustness of deep learning algorithms to nonstationarities in the environment and parameters. In 2016-17 we created a framework to analyze the robustness of deep Reinforcement Learning to adversarial attacks. We engineer specific attacks, and two of our interesting contributions include: 1. Modern deep RL algorithms are susceptible to even naive attacks, yet these attacks do not affect traditional Radial Basis Function kernel based RL algorithms as badly. 2. Attacks can be utilized in a robust control framework to improve the robustness of RL found policies.
6. Transferring policies between domains: A challenge with RL is that it is difficult to generalize a policy from one domain to another. This can lead to several challenges when RL is used in human-machine teaming, since humans and anthropogenic environments are nonstationary. Furthermore, this lack of domain transferability also limits the ability of RL algorithms to translate to real-world situations based on “practice” in simulation with humans. We are developing novel transfer learning algorithms that use policy adaptation techniques inspired loosely from adaptive control. Unlike prevalent policy initialization techniques which have no sample efficiency guarantees, we have developed strong sample efficiency guarantees for our policy adaptation techniques.
7. Ongoing work in Human-Machine Autonomy: We have identified and are in process of investigating the following three important aspects of Human-Machine collaborative Autonomy: 1. Learning human-implicit task subgoals from human demonstrations, 2. Modeling subgoal interdependence and Inference of current task, 3. Collaborative task execution.
8. Uninformed to Informed Search: We developed a new Bayesian stochastic process model, termed the Bi-Homogeneous Poisson Exposure Process model to predict the accumulation of information gain in environments that change over space and time. Fundamental probabilistic inequalities relating to the number of samples required for confident phase transitions from uninformed search strategies (such as random or grid based search) to an informed search strategy that maximizes the accumulated information gain in a search sortie were developed. These results hold significance in developing information maximizing search algorithms for Unmanned or Manned Aerial vehicles or teams that have limited range or endurance.
9. Chance Constrained Model Predictive Control: We developed new algorithms for Model Predictive Control in the presence of probabilistic constraints. The algorithms were de-

veloped in the Model Predictive Control formulation to leverage the elegance and analyzability of that formulation. Instead of the traditional MPC formulation, where constraints are enforced as inequalities, the constraints in our formulation are enforced in terms of the probability of violating a constraint. This leads to less conservative results and control algorithms that can better incorporate and deal-with risk without being overtly conservative.

10. Non-Gaussian measurement noise: We have strong results on a computationally efficient and recursive GP inference technique in the presence of non-Gaussian measurement noise. These results are discussed in Section 6. Interestingly, we were able to obtain results that were comparable to those from deep Neural Networks without requiring data augmentation or multiple passes through data. We have further developed the Multi Agent Game Emulator platform, which is flight-simulator designed to enable multiple manned and unmanned aircraft to simultaneously interact in a common cloud environment. Our main contribution in that area has been to tie our work closely with the open source Robot Operating System's GAZEBO simulator, that was used extensively in the DARPA robotics challenge.

1.1 Personnel and Facilities Development

Dr. Chowdhary was promoted to Associate Professor in 2020 at UIUC. He holds now a joint appointment with Agricultural and Biological Engineering and the Computer Science department. He was also awarded Donald Biggar Willet Faculty Fellow title, and won the Dean's Award for excellence in engineering research in 2020 from the Grainger College of Engineering. He was also awarded the Dave Ward Memorial Award by Aerospace Control and Guidance Systems committee (ACGSC) in September 2015.

The project contributing students of the PI are Ali Abdollahi (graduated 2016 with MS), Rakshit Allamaraju (graduated 2016 with MS), Allan Axelrod, Girish Joshi, and Harshal Maske (graduated May 2018 with PhD), Anay Pattanaik, Denis Osipychiev, and Joshua Whitman. Allan, Girish, and Denis are PhD students. Girish Joshi, Harshal Maske, Denis Osipychiev, and Joshua Whitman were currently partially funded by this project. A new addition in late 2019 was Dr. Aseem Borkar, a postdoc who worked on the capstone project of algorithms to help a human operator monitor a convoy using autonomous UAS planning and control algorithms..

The project has led to further development and validation of the Stabilis autopilot software and hardware developed by Chowdhary's group, and the MAGE multi-agent simulation system. These systems were utilized in a recent DARPA Improv project in which Chowdhary participated as a co-PI. The work developed in this project was used to inspire a collaboration between Sandia and PI's group in the area of robust autonomy. A new DARPA project is being awarded to PI and collaborators at UIUC to develop an experimental testbed where some of the algorithms from this work can be evaluated. The TerraSentia ground rover system developed by Chowdhary at UIUC also greatly benefited from this project. The TerraSentia system is now being used as a demonstration and capability development platform in a number of DoD projects.

1.2 Major Changes: Move to University of Illinois at Urbana Champaign in 2016

The PI and many of the contributing students have moved to the University of Illinois at Urbana Champaign on July 1 2016. Access to new resources, facilities, and educational opportunities for students, this move is enabling a stronger research program that will better serve this project's and AFOSR's need. The funding transition is now complete. An additional year of funding (18-19) has been approved.

1.3 On the Organization of This Report

The work performed in this grant built the foundation for the PI's successful research program. Accordingly a number of different directions were explored through the past 4 years and the report tries to combine them all in one exhaustive document. The AI, machine learning, and autonomy communities are undergoing rapid progress, we are accordingly adapting our hypothesis and directing our work to leverage emerging strengths to better server the goals of making AI algorithms that enable better human-machine collaborative autonomy.

1.4 Publications

Peer reviewed conferences and journals

- Whitman J., Maske H., Kingravi H., Chowdhary G., Evolving Gaussian Processes and Kernel Observers for Learning and Control in Spatiotemporally Varying Domains with applications in agriculture, weather monitoring, and fluid dynamics, IEEE Control Systems Magazine, 2020 (accepted).
- Maske H., Kingravi H., Chowdhary G., Sensor Selection via Observability Analysis in Feature Space, American Control Conference (ACC), WI, 2018.
- Joshi G., Chowdhary G., Cross-Domain Transfer in Reinforcement Learning using Target Apprentice, International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 2018.
- Liu M., Chowdhary G., Bruno D. S., How J. P., Shih-Yuan L., Gaussian Processes for Learning and Control: Tutorial with Examples, IEEE Control Systems Magazine, accepted, March 2018
- Seifert David, Chowdhary Girish, Muhlegg Maximillian, Holzapfel Florian, Online Gaussian Process Regression with Non-Gaussian Likelihood, American Control Conference, Seattle WA, May 2017
- Allamaraju Rakshit, Chowdhary Girish, Communication Efficient Decentralized Gaussian Process Fusion for Multi-UAS Path Planning, American Control Conference, Seattle WA, May 2017
- Denis Osipychiev, Duy Tran, Weihua Sheng, and Girish Chowdhary, Human Intention-Based Collision Avoidance for Autonomous Cars, American Control Conference, Seattle WA, May 2017
- Harshal Maske, Hassan Kingravi, and Girish Chowdhary. Sensor Selection via Linear Systems Model in Feature Space. In Conference on Decisions and Control, Melbourne, Australia, 2017. IEEE. Submitted.
- Kingravi Hassan, Maske Harshal, Chowdhary Girish, Kernel Observers: System Theoretic Modeling and Inference of Sptiotemporal Processes, NIPS BNP Workshop 2016.
- Axelrod Allan, Chowdhary Girish, A Hybridized Bayesian Parametric-Nonparametric Approach to the Pure Exploration Problem, NIPS Workshop: BNP the next generation 2016.
- Harshal Maske, Girish Chowdhary, and Prabhakar Pagilla. Intent aware shared control in Off-nominal situations. In Conference on Decisions and Control, Las Vegas, USA, 2016. IEEE. Submitted.

- Axelrod Allan, Chowdhary Girish, Uninformed-to-Informed Exploration in Real-World Environments, AI-STATS 2016, submitted.
- R Allamraju, G Chowdhary and C Crick, Multi Agent Game Emulator (MAGE) for Collaborative Flight Autonomy Research, International Conference of Robotic Automation 2016, submitted

Highly Selective Conferences with Double Blind Peer Review

- Whitman J., Chowdhary G., Learning Dynamics Across Similar Spatiotemporally Evolving Systems, Conference on Robot Learning (CORL), San Jose, CA, Nov 2017, accepted for long-talk, long-talk acceptance rate 10
- Hassan Kingravi, Harshal Maske, and Girish Chowdhary. Kernel observers: Systems theoretic modeling and inference of spatiotemporally varying processes. In Advances in Neural Information Processing Systems (NIPS), Barcelona, Spain, 2016.

Peer-Reviewed Workshop Papers

- Anay Pattanaik, Victor Sui, Denis Osipychyev, and Girish Chowdhary. Adversarial Attacks on (Deep) Reinforcement Learning. In Multidisciplinary Conference on Reinforcement Learning and Decision Making, Ann Arbor, Michigan, USA, 2017.
- Whitman Joshua, Chowdhary Girish, Learning Dynamics Across Similar Spatiotemporally Evolving Systems, Reinforcement Learning and Decision Making (RLDM) 2017, Ann Arbor, MI.
- Joshi Girish, Chowdhary Girish, Cross-Domain transfer learning using target apprentice, RLDM 2017, Ann Arbor, MI.

Working/submitted Papers

- Hassan Kingravi, Harshal Maske, and Girish Chowdhary. Kernel observers: Systems theoretic modeling and inference of spatiotemporally varying processes. Submitted to Transactions on Automatic Control
- Anay Pattanaik, Shuijing Liu, Zhenyi Tang, and Girish Chowdhary. Harnessing Adversarial Attacks on Deep Reinforcement Learning for Improving Robustness. ARXIV, being submitted to conferences
- Denis Osipychyev, Girish Chowdhary, Using Reinforcement Learning to Make Multi-Agent Partially Observable Games more Intuitive to Humans.

Awards

- Chowdhary and team wins best systems paper award at Robotics Science and Systems 2018 conference
- Chowdhary awarded the Ward Memorial Lecture Award, by Aerospace Control and Guidance Systems Committee (ACGSC), for contributions to aerospace GNC
- Axelrod finalist for best student paper award at AIAA SciTech Axelrod received AAAI travel awards

Thesis

- Maske Harshal, Learning, inference, and control for construction robots and spatiotemporal processes, May 2018, University of Illinois at Urbana Champaign.
- Abdollahi Ali; Adaptive Multi-Objective Optimizing Flight Controller, MS Thesis, Oklahoma State University, June 2016.
- Allamaraju Rakshit; Communication efficient decentralized sensor fusion in sensor networks, Oklahoma State University, MS Thesis, Oklahoma State University, Dec 2015.

Workshops organized

- Collaborative sensing, learning, and control for human-machine systems, ACC 2016, Boston MA, co-organized with Nisar Ahmed, Luca Bertucelli, and Soumik Sarkar
- Whats the big deal with Deep Learning?, CDC 2017, Melbourne, Australia, co-organizing with Alexander Gerhard, Hassan Kingravi, Chinmay Soman.

Part I

Progress in the Year 2018-end of project

1 Multi-agent Aerial Monitoring of Moving Convoys using Elliptical Orbits

1.1 Introduction

We propose a novel scheme for the surveillance of a dynamic ground convoy, moving along a nonlinear trajectory, by aerial agents that maintain a uniformly spaced formation on a time-varying elliptical orbit encompassing the convoy. Elliptical orbits are used as they are more economical than circular orbits for circum-navigating the group of targets in the moving convoy. The proposed scheme includes an algorithm for computing feasible elliptical orbits, a vector guidance law for agent motion along the desired orbit, and a cooperative strategy to control the speeds of the aerial agents in order to quickly achieve and maintain the desired formation. The aerial agent is modelled using unicycle kinematics, and the proposed scheme achieves mission objectives while accounting for the linear and angular speed constraints of the aerial agent. The scheme is validated through simulations and actual experiments, with ground robots as the convoy, and a team of quadrotors as the aerial agents, in a motion capture environment.

The problem of tracking ground targets using aerial agents is well studied in literature, both for single and multiple Unmanned Aerial Vehicles (UAVs). Many solutions to this problem focus on circumnavigating the targets using circular orbits for targets typically moving on straight paths. The main contribution of this work is a strategy to accommodate multiple aerial agents flying in a formation on a time varying elliptical orbit, to circumnavigate around a moving ground convoy. A convoy is a group of ground vehicles trailing one behind the other on a common path. Thus to circumnavigate and such a group of targets often elliptical orbits are more economical in terms of path length as compared to circular orbits.

[15] and [16] proposed a algorithm to compute a time varying elliptical orbit encompassing the convoy and a vector-field guidance law to drive a single aerial agent, modelled using unicycle kinematics, to track this orbit. For a multi-UAV mission the objective is to achieve and maintain a multi-agent formation on elliptical orbit that equally divides the elliptical orbit in terms of the

parametric length of the ellipse. The original contributions on this work are:

1. Improvements to the vector field guidance law better tracking of the desired elliptical orbit.
2. An updated algorithm for computing feasible elliptical orbits to encompass the target convoy, while accounting for the motion constraints of the aerial agents.
3. A strategy to control the linear speeds of the aerial agents to achieve uniform spacing on the elliptical orbit.
4. A cooperation scheme for fast convergence to the formation having equal parametric spacing on the elliptical orbit.

Our work is applicable to a large class of UAVs, such as autonomous fixed wing aircraft, helicopters and quadrotors. This is because we approximate the aerial agents as unicycle kinematic agents with constraints on minimum and maximum linear speed and maximum angular speed. We validate our results with MATLAB simulations for unicycle agents, and for a quadrotor model using a ROS-Gazebo based software in the loop (SITL) simulator. We have also implemented the proposed strategy with a team of quadrotors and a convoy of ground robots in a motion capture environment.

The paper is organised as follows: Section 1.2 reviews some related work. Section 2 describes the problem setting, assumptions and notation. Section 3 discusses some preliminary concepts that are used for developing the proposed strategy. Section 4 discusses the vector field guidance law used to drive the aerial agents to the elliptical orbit. In Section 5 the linear speed profile of the aerial agents is designed to maintain the desired formation. Section 6 discusses the scheme for calculating the time varying elliptical orbit around the convoy. In Section 7, the main cooperation algorithm for attaining the desired aerial formation is discussed. Section 8 validates the proposed strategy by simulations and actual experiments.

1.2 Related Work

This section is a short overview of some related literature. An extended survey can be found in [137]. Our proposed scheme focuses on monitoring a ground convoy by aurally circumnavigating it with UAVs. This is done using a vector field based guidance law to track elliptical orbits encompassing the ground convoy. In literature such strategies are generally used for tracking circular orbits for UAV path-planning [111] and atmospheric sensing [87] applications. They have also been developed for tracking circular orbits around stationary and moving target, while respecting curvature demands of the underlying vector field in [123]. For multiple UAVs monitoring a single target [48] or a closely moving group of targets [114] while flying on a circular orbit, the UAVs are treated as unicycle kinematic agents and the phase separation is maintained by controlling the linear speed of the UAVs. This is close in spirit to our linear speed control strategy whereby the aerial agents maintain a formation that divides the parametric length of a time varying elliptical orbit into equal parts.

The vector field guidance law in [48] is extended to track a racetrack like orbit in [50], where the UAV alternates between tracking a circular segment and a straight line. Frew [49] extends the law in [48] to tracking an elliptical orbit whose parameters depend on estimation uncertainties of the target states. In both, the UAVs track a target moving on a straight line path. In contrast, our strategy guides the aerial agents to track a flexible ellipse that adapts its shape and orientation to the convoy trajectory.

Another approach for aerial monitoring of ground targets is cooperative cyclic pursuit, where multiple agents cooperatively converge to a circular orbit with an equi-spaced formation around the ground target, both stationary [52] and moving [99], with aerial agents that follow unicycle kinematics. In [100], such a cooperative scheme is proposed for achieving a circular formation on orbits having varying radii around the target. The scheme proposed in this work generalises

over ellipses and circular orbits of varying sizes and gives a more robust orbit tracking as a result of the vector field guidance law.

Other approaches to formation control include [171] and [118], where the agents switch between revolutions on a circular orbit around slow moving targets and fixed formation paths to move with faster moving targets, e.g., to track a point moving on a piecewise linear path. In [172], linear and circular paths are considered for the target and the emphasis is on analysing the error caused by wind. In [79], the target is assumed to follow a smooth trajectory specified a priori as the level curve of a known function. In [108] a backstepping-based controller is designed to follow a vector field to achieve a multiUAV formation on a circular orbit around a target while respecting the input constraints of the UAVs. Most of these approaches are designed towards tracking circular orbits. For a convoy of ground targets an elliptical orbit can be more economical in terms of path length, see [15], [16] for the single agent case. A consensus based strategy for elliptical formation around a ground target is proposed in [98]. Here the agent models considered are single and double integrator. Our strategy, for a unicycle type aerial agent with input constraints, is applicable for a large class of UAV platforms.

2 Problem Description and Notation

We assume a target ground convoy of N_T vehicles trailing one behind the other on some unknown common path. Each vehicle moves with a maximum speed of $V_{T_{max}}$. We number the vehicles $1, 2, \dots, N_T$ along the direction of travel of the convoy with vehicle N_T being the lead vehicle. Our objective is to design a cooperation strategy for a team of N_A aerial agents for monitoring the moving convoy. The aerial agents are modelled using unicycle kinematics flying at a commanded altitude as follows:

$$\begin{aligned} \dot{x}_{Ai}(t) &= V_{Ai} \cos(\psi_{Ai}(t)), \quad \dot{y}_{Ai}(t) = V_{Ai} \sin(\psi_{Ai}(t)), \\ \dot{\psi}_{Ai}(t) &= \omega_{Ai}(t) \end{aligned} \quad (1)$$

where for agent i , $(x_{Ai}(t), y_{Ai}(t))$ are the position coordinates, $\psi_{Ai}(t)$ is the heading angle, $V_{Ai} \in [V_{A_{min}}, V_{A_{max}}]$ is the commanded linear speed, and $\omega_{Ai}(t)$ is the commanded angular velocity of the monitoring agent satisfying $|\omega_{Ai}(t)| \leq \omega_{max}$. We make the following assumptions:

1. The upper bound on convoy speed $V_{T_{max}}$ and the convoy vehicle positions $(x_{Ti}(t), y_{Ti}(t))$ for all $i \in \{1, \dots, N_T\}$ are known to the aerial agents. This could be through communication for a cooperating convoy or appropriate sensors on board the aerial agents.
2. The maximum speed of the ground convoy and the linear speed bounds of the aerial agent are related as follows:

$$0 \leq V_{T_{max}} \ll V_{A_{min}} < V_{A_{max}} - 2V_{T_{max}}. \quad (2)$$

That is, the aerial agents are assumed to have a sufficient speed advantage over the ground convoy which is reasonable as they have to circumnavigate the moving convoy.

3. The aerial agents have a sufficiently large communication range for effective cooperation.

We denote the rotation matrix from the right handed global reference frame to a tilted frame with tilt angle $\theta(t)$ as

$$R_\theta(t) = \begin{bmatrix} \cos(\theta(t)) & \sin(\theta(t)) \\ -\sin(\theta(t)) & \cos(\theta(t)) \end{bmatrix}. \quad (3)$$

3 Preliminaries

We now discuss some useful properties of elliptical orbits, used later for developing the guidance law and the cooperation strategy between aerial agents for achieving the desired formation. The parametric equation of the ellipse will be:

$$x(s) = a \cos(s), \quad y(s) = b \sin(s) \quad (4)$$

Here s is a parameter taking values in $[0, 2\pi)$, a is the semi-major axis length and b is the semi-minor axis length, with $a \geq b > 0$. The equation of the ellipse then is $x(s)^2/a^2 + y(s)^2/b^2 = 1$.

The first and second time-derivatives of (4) are

$$\dot{x}(s) = -a \sin(s)\dot{s}, \quad \dot{y}(s) = b \cos(s)\dot{s} \quad \text{and} \quad (5)$$

$$\begin{aligned} \ddot{x}(s) &= -a \cos(s)\dot{s}^2 - a \sin(s)\ddot{s}, \\ \ddot{y}(s) &= -b \sin(s)\dot{s}^2 + b \cos(s)\ddot{s} \end{aligned} \quad (6)$$

If a point moves on the ellipse with a parametric speed $\dot{s} > 0$, then from (5), the actual speed of the point is given by

$$V(s) = \sqrt{\dot{x}(s)^2 + \dot{y}(s)^2} = \sqrt{a^2 \sin^2(s) + b^2 \cos^2(s)} \dot{s} \quad (7)$$

From (5) and (6), the curvature of the ellipse at parameter value s is given by

$$\kappa(s) = \frac{|\dot{x}(s)\ddot{y}(s) - \dot{y}(s)\ddot{x}(s)|}{(\dot{x}(s)^2 + \dot{y}(s)^2)^{\frac{3}{2}}} = \frac{ab}{(a^2 \sin^2(s) + b^2 \cos^2(s))^{\frac{3}{2}}}, \quad (8)$$

and the radius of curvature is given by

$$R(s) = \frac{1}{\kappa(s)} = \frac{(a^2 \sin^2(s) + b^2 \cos^2(s))^{\frac{3}{2}}}{ab}. \quad (9)$$

Define $G(s) = \sqrt{a^2 \sin^2(s) + b^2 \cos^2(s)}$. Successively differentiating G , we have

$$\frac{dG(s)}{ds} = \frac{(a^2 - b^2) \sin(2s)}{2\sqrt{a^2 \sin^2(s) + b^2 \cos^2(s)}} \quad (10)$$

$$\frac{d^2G(s)}{ds^2} = \frac{(a^2 - b^2) \cos(2s)}{G(s)} - \frac{(a^2 - b^2)^2 \sin^2(2s)}{4G(s)^3} \quad (11)$$

By the first order necessary conditions for maxima or minima of G , setting $\frac{dG(s)}{ds} = 0$ gives the following cases:

Case 1: If $a = b$, then the ellipse is a circle with $\frac{dG(s)}{ds} = 0$, for all $s \in [0, 2\pi)$, i.e., $G(s) = K$ where K is a constant.

Case 2: If $a > b$, then $\sin(2s) = 0$, so $s = (2n+1)\pi/2$, $n\pi$ for $n \in \{0, 1\}$, as $s \in [0, 2\pi)$. By the second order sufficiency condition for maxima or minima of G , from (11),

$$\begin{aligned} \left. \frac{d^2G(s)}{ds^2} \right|_{s=n\pi} &= \frac{a^2 - b^2}{b} > 0, \text{ and} \\ \left. \frac{d^2G(s)}{ds^2} \right|_{s=\frac{(2n+1)\pi}{2}} &= \frac{b^2 - a^2}{a} < 0. \end{aligned}$$

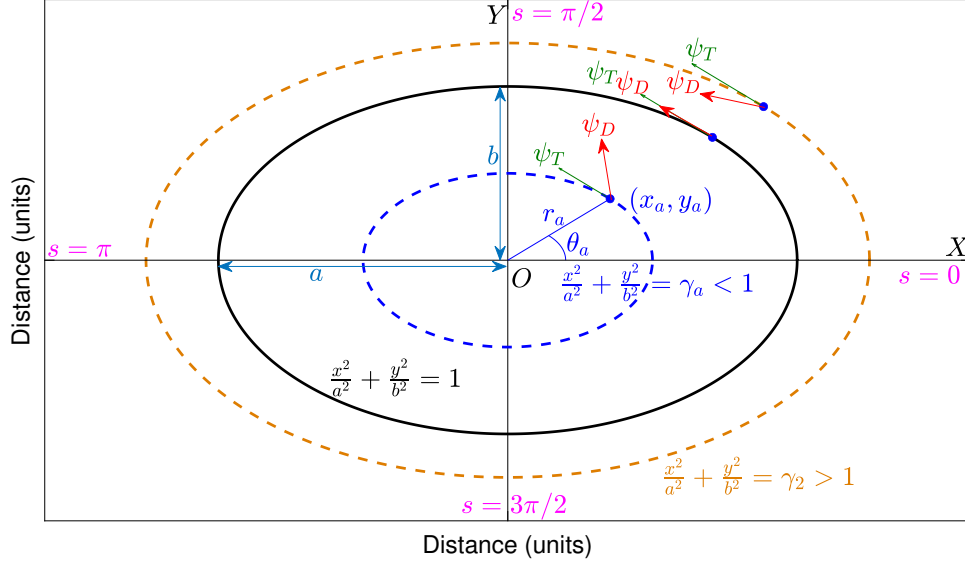


Figure 1: Geometric representation of (x_a, y_a) , (r_a, θ_a) and (s_a, γ_a) . The s values listed in Table 1 are shown in magenta. The desired orbit is plotted in solid black. The commanded aerial agent heading and local tangential heading shown in red and green respectively.

Thus $s = n\pi$ minimises G , and $s = (2n+1)\pi/2$ maximises G .

Consider a point moving with constant parametric speed $\dot{s} = s_v$ on the ellipse. From (7), (8) and (9), $V(s) = G(s)s_v$, $\kappa(s) = ab/G(s)^3$ and $R(s) = G(s)^3/ab$ respectively, which implies that $s = n\pi$ minimises $V(s)$ and $R(s)$ and maximises $\kappa(s)$, and vice versa for $(2n+1)\pi/2$. Table 1 summarises the maximum and minimum values of $V(s)$, $R(s)$ and $\kappa(s)$.

Table 1: Extrema values on an ellipse for a point moving with constant $\dot{s} = s_v$ and $n \in \{0, 1\}$

s	$V(s)$	$R(s)$	$\kappa(s)$
$n\pi$	$V_{min} = bs_v$	$R_{min} = b^2/a$	$\kappa_{max} = a/b^2$
$(2n+1)\pi/2$	$V_{max} = as_v$	$R_{max} = a^2/b$	$\kappa_{min} = b/a^2$

4 Vector-field guidance law

The vector field guidance law computes a desired heading command that guides the aerial agent towards the desired elliptical orbit described by (4). Our guidance law is an update of the guidance law used in [16], with a small modification for improved tracking performance.

If (x_{AE}, y_{AE}) are the agent position coordinates expressed in the coordinate frame centered and aligned with the desired elliptical orbit (see Fig. 1), then the commanded heading of the aerial agent has two components

$$\psi_D = \psi_T + \psi_O, \quad (12)$$

where ψ_T is the tangential direction to the concentric ellipse described by (16) at position (x_{AE}, y_{AE}) as in Fig. 1 and ψ_O is an offset term responsible for generating the heading command to reach the desired orbit. These are computed as

$$\psi_T = \arctan 2 (d_c b^2 x_{AE}, -d_c a^2 y_{AE}), \quad (13)$$

$$\psi_O = d_c \arctan (k_\gamma |\kappa(s_A)| (\gamma_A - 1)), \quad (14)$$

where d_c is chosen according to (19) to select the direction of motion along the elliptical orbit. For computation of ψ_O , $k_\gamma > 0$ is a gain used as tuning parameter, γ_A is computed using (20) and curvature κ is computed using (8) at parameter s_A computed using (18). If the aerial agent position is outside the desired orbit, $\gamma_A > 1$. Thus a positive offset ψ_O is added to ψ_T which causes the desired heading ψ_D to point inward from the local concentric orbit towards the desired elliptical orbit as in Fig. 1. If the aerial agent position is inside the desired orbit ($\gamma_A < 1$), a negative offset ψ_O is added to ψ_T which causes the desired heading ψ_D to point outward (from the local concentric orbit) towards the desired elliptical orbit as shown in Fig. 1. If the aerial agent position is on the desired orbit ($\gamma_A = 1$), no offset is added causing the desired heading $\psi_D = \psi_T$ along the tangent to the desired elliptical orbit as shown in Fig. 1. The aerial agents track ψ_D by commanding the angular velocity of the aerial agent as follows:

$$\omega_A(t) = k_\psi (\psi_D(t) - \psi_{AE}). \quad (15)$$

where k_ψ is a control gain, ψ_{AE} is the heading angle of the aerial agent expressed relative to the orbit centric reference frame. It is computed as: $\psi_{AE} = \psi_A - \theta_E$, where ψ_A is the heading angle of the aerial agent and θ_E is the tilt angle of the ellipse, both with respect to the stationary global reference frame. The angular velocity computed of the aerial agent given by (15) is constrained as follows: $|\omega_{Ai}(t)| \leq \omega_{max}$.

We introduce the curvature term $\kappa(s_A)$ in the ψ_O to increase the sensitivity of the guidance law in the high curvature regions of the elliptical orbit. This gives better tracking and transient behaviour as compared to a constant gain k_γ as in [16]. For proof of convergence of the guidance law to the desired elliptical orbit, see [16], as exactly the same proof applies here.

5 Linear speed profile for aerial agents

In [15, 16] the single aerial agent monitoring the convoy was commanded to fly with a constant linear speed and the angular speed was controlled so as to guide the agent to the desired elliptical orbit. Switching to the multi-agent framework, the objective now is to obtain a parametrically equi-spaced formation of N_A aerial agents on the ellipse. That is, the parametric separation between any pair of consecutive aerial agents on the ellipse is $\delta_s = 2\pi/N_A$. This is done by controlling the linear speeds of the aerial agents. To begin with, consider a stationary elliptical orbit described by (4).

5.1 Parameter computation for the aerial agents

If the aerial agents start with equi-parametric separations on the ellipse, then they can maintain it by moving at a constant parametric speed $\dot{s} = s_v$. But it is not necessary that they start on the desired elliptical orbit. For an agent at position (x_{AE}, y_{AE}) relative to the orbit-centered coordinate frame, we need to calculate the parameter value s_A for the agent. Given a desired orbit centered at origin O with axis constants a and b , every point in space lies on an ellipse of the form $\frac{x^2}{a^2} + \frac{y^2}{b^2} = \gamma$ concentric with the desired orbit, where $\gamma < 1$ for points inside the

desired orbit, $\gamma = 1$ for points on it, and $\gamma > 1$ for points outside, as shown in Fig. 1. Then the parametric equation for these concentric ellipses is given by

$$x(\gamma, s) = a\sqrt{\gamma}\cos(s), \quad y(\gamma, s) = b\sqrt{\gamma}\sin(s) \quad (16)$$

Thus the coordinates of the point (x_{AE}, y_{AE}) can be represented by (16) for some value of s_A and γ_A . If the corresponding polar coordinates of (x_{AE}, y_{AE}) in the orbit centric reference frame are (r_A, θ_A) , then

$$\theta_A = \arctan 2(y_{AE}, x_{AE}), \quad (17)$$

and $(x_A, y_A) = (r_A \cos(\theta_A), r_A \sin(\theta_A))$. Equating these with $(x(\gamma_A, s_A), y(\gamma_A, s_A))$ in (16), we have $a\sqrt{\gamma_A}\cos(s_A) = r_A \cos(\theta_A)$ and $b\sqrt{\gamma_A}\sin(s_A) = r_A \sin(\theta_A)$. Hence $\frac{b}{a}\tan(s_A) = \tan(\theta_A)$. Thus s_A is computed as

$$s_A = d_c \arctan 2(a \sin(\theta_A), b \cos(\theta_A)), \quad (18)$$

where d_c is a direction selection input selected as

$$d_c = \begin{cases} -1, & \text{for clockwise motion on orbit} \\ 1, & \text{for counter-clockwise motion on orbit} \end{cases} \quad (19)$$

Hence given agent's position (x_{AE}, y_{AE}) and desired orbit constants a, b , we can compute using (17) and (18) the parameter s_A on the concentric ellipse given by (16). The value of γ_A is given by

$$\gamma_A = \frac{x_{AE}^2}{a^2} + \frac{y_{AE}^2}{b^2} \quad (20)$$

From (18), it can be shown that $s_A = \theta_A$, when $\theta_A = n\pi$ and $(2n+1)\pi/2$ for $n \in \{0, 1\}$, as illustrated in Fig. 1. Also, (18) is a bijection between $\theta_A \in [0, 2\pi)$ and $s_A \in [0, 2\pi)$

5.2 Speed computation for the aerial agents

The objective of the linear speed control is to achieve and maintain an aerial agent formation that divides the parametric length of the ellipse (2π) into N_A equal parts. To do this the commanded linear speed for each agent comprises of three components:

- 1) nominal speed profile V_{Ei}
- 2) correction component V_{Ci}
- 3) Feed-forward compensation component $\bar{V}_T$.

Agents moving with equal constant parametric speed $\dot{s}_A$ on an ellipse maintain their initial parametric separation on the ellipse. Thus we first compute a nominal linear speed profile for the agent corresponding to the constant parametric speed $\dot{s}_A = s_v$ for motion on the stationary elliptical orbit described by (16). From Table 1 the maximum and minimum speeds on the ellipse for such a motion are $V_{max} = a\sqrt{\gamma_A}s_v$ and $V_{min} = b\sqrt{\gamma_A}s_v$. Suppose we want to constrain V_{min}, V_{max} to a given interval $[V_{Emin}, V_{Emax}]$. Then in order to symmetrically utilize the interval, we select s_v such that the average speed $(V_{min} + V_{max})/2$ resulting from constant parametric rate $\dot{s} = s_v$ equals $(V_{Emin} + V_{Emax})/2$, leading to

$$s_v = \frac{V_{Emin} + V_{Emax}}{(a + b)\sqrt{\gamma_A}}. \quad (21)$$

To ensure that the motion due to this parametric rate does not violate the given constraint, we need $[V_{min}, V_{max}] \subseteq [V_{Emin}, V_{Emax}]$. This leads to:

$$\frac{V_{Emin}}{b\sqrt{\gamma_A}} \leq s_v \text{ and } s_v \leq \frac{V_{Emax}}{a\sqrt{\gamma_A}} \Rightarrow \frac{V_{Emin}}{V_{Emax}} \leq \frac{b}{a}. \quad (22)$$

We select a, b in Section 6 such that (22) is satisfied. The nominal linear speed corresponding to the constant parametric rate s_v for an aerial agent i at position coordinates (x_{AEi}, y_{AEi}) , moving on a stationary ellipse described by (16), is now computed using (7) as

$$V_{Ei} = \sqrt{\gamma_{Ai}(a^2 \sin^2(s_{Ai}) + b^2 \cos^2(s_{Ai}))} s_v, \quad (23)$$

where s_{Ai} is calculated using (17) and (18), and γ_{Ai} and $\dot{s}_{Ai} = s_{vi}$ computed using (20) and (21) respectively. Then from (21) and assuming (22) is satisfied, $V_{Ei} \in [V_{Emin}, V_{Emax}]$.

To drive the parametric spacing between neighbouring agents to the desired separation of $2\pi/N_A$ on the ellipse, we add a control component V_{Ci} to the nominal speed V_{Ei} of the agents. V_{Ci} is computed based on cooperation with a neighbouring agent as explained in the Section 7. The resulting linear speed magnitude command then is

$$\tilde{V}_{Ai} = V_{Ei} + V_{Ci}. \quad (24)$$

For monitoring a moving convoy, the elliptical orbit is not stationary. In Section 6 we propose a method to compute a desired orbit centered at the target centroid $(\bar{x}_T, \bar{y}_T)$ (average position of the ground convoy vehicles). The velocity of the target centroid is computed by filtering the target centroid position using an exponential smoothing filter, and then computing the discrete derivative $(\bar{V}_{Tx}, \bar{V}_{Ty}) = (\Delta \bar{x}_T / \Delta t, \Delta \bar{y}_T / \Delta t)$ with respect to the stationary global reference frame. Since the speed of the ground convoy vehicles is bounded above by V_{Tmax} , the magnitude of velocity of the target centroid is also bounded above by V_{Tmax} . Thus this velocity vector captures the motion of the center of the ellipse, and is added as a feed-forward component to the aerial agent velocity corresponding to linear speed $\tilde{V}_{Ai}$ (given by (1)). Since the aerial agents follow unicycle kinematics their linear velocity components along the axes of the global reference frame cannot be commanded separately. Thus an approximate compensation is used where the magnitude of the resultant vector is commanded as the linear speed of the agent i as follows

$$V_{Ai} = \sqrt{(\tilde{V}_{Ai} \cos(\psi_{Ai}) + \bar{V}_{Tx})^2 + (\tilde{V}_{Ai} \sin(\psi_{Ai}) + \bar{V}_{Ty})^2} \quad (25)$$

Since the feedforward component can at most increase or decrease the agent speed by V_{Tmax} , the range of speeds available to design $\tilde{V}_{Ai}$ is $[V_{Rmin}, V_{Rmax}]$, where $V_{Rmax} = V_{Amax} - V_{Tmax}$ and $V_{Rmin} = V_{Amin} + V_{Tmax}$. This interval is not empty due to assumption (2).

In order to allow for a band of speeds above and below the nominal linear speed profile given by (23), we select a value $\delta \in (0, 1]$, and select V_{Emin} and V_{Emax} (bounds on the nominal speed profile (23)) such that:

$$\begin{aligned} (V_{Emin} + V_{Emax})/2 &= (V_{Rmax} + V_{Rmin})/2 \\ V_{Emin} - V_{Emax} &= \delta(V_{Rmax} - V_{Rmin}) \end{aligned}$$

Solving this system of linear equations yields

$$V_{Emin} = (1 - \delta)V_{Rmax} + (1 + \delta)V_{Rmin}, \quad (26)$$

$$V_{Emax} = (1 + \delta)V_{Rmax} + (1 - \delta)V_{Rmin}. \quad (27)$$

The intervals $[V_{Emax}, V_{Rmax}]$, $[V_{Rmin}, V_{Emin}]$ allow room for the control effort V_{Ci} . Finally, the total linear speed command V_{Ai} given by (25) is saturated to V_{Amin} or V_{Amax} so as to maintain $V_{Ai} \in [V_{Amin}, V_{Amax}]$, if the contribution of V_{Ci} is large.

6 Elliptical orbit computation

Next we discuss an algorithm for computing the elliptical orbit for encompassing the ground convoy. The essential idea is similar to [16], with some key differences. Instead of considering a regression line, we choose a reference line parallel to the line segment joining the first and last vehicle positions in the convoy (i.e., convoy targets with indices $i = 1$ and $i = N_T$), and passing through the target centroid $(\bar{x}_T, \bar{y}_T)$, which is the mean position of the convoy vehicles. The angle made by the reference line with respect to the global reference frame is denoted by θ_E .

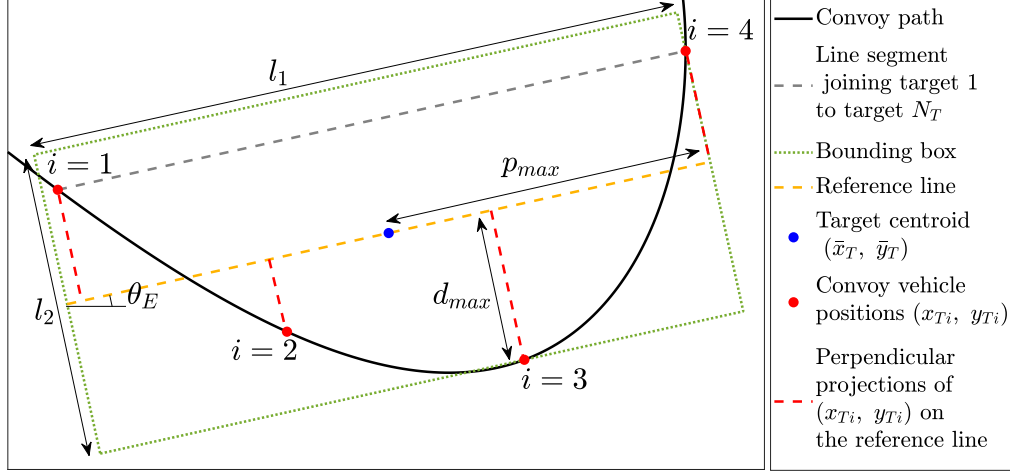


Figure 2: Bounding box construction for a target convoy with $N_T = 4$

The perpendicular projections from the convoy target positions to this reference line are computed. The length of the bounding box is selected as $l_1 = 2p_{max}$, where p_{max} is the distance between the target centroid and the farthest projection point on the reference line. The breadth is selected as $l_2 = 2d_{max}$, where d_{max} is the maximum perpendicular distance of the convoy vehicle positions from the reference line. The bounding box is centered at the target centroid $(\bar{x}_T, \bar{y}_T)$. An example of this construction for a convoy of $N_T = 4$ vehicles is shown in Fig. 2.

Since this strategy is implemented iteratively with k being the iteration count, and a constant loop delay Δt , the Algorithm 1 takes current convoy position values as inputs and computes the values of $l_1(k)$, $l_2(k)$.

To select the ellipse that contains this bounding box, we consider the minimum area ellipse circumscribing the bounding box according to the following Lemma.

Lemma 6.1. ([16]) *The area of an ellipse $\mathcal{E} : x^2/a^2 + y^2/b^2 = 1$ circumscribing a rectangle of dimensions $l_1 \times l_2$ with $l_1 > l_2 > 0$ is minimized by $a = l_1/\sqrt{2}$ and $b = l_2/\sqrt{2}$.*

To ensure that the elliptical orbit is feasible, its minimum radius of curvature given in Table 1 should exceed the worst case minimum turn radius of the aerial agents, i.e.,

$$R_{min} \geq \frac{V_{A_{max}}}{\omega_{max}} \Rightarrow b \geq \sqrt{a \frac{V_{A_{max}}}{\omega_{max}}} \quad (28)$$

Algorithm 1 BOUNDING_BOX

Inputs: (X_T, Y_T) , k , $(\bar{x}_T(k), \bar{y}_T(k))$, $\theta_E(k)$
Outputs: $l_1(k)$, $l_2(k)$
Initialise: $x_{min} = 0$, $x_{max} = 0$, $d_{max} = 0$, $p_{max} = 0$

- 1: **for** $i \in \{1, \dots, |X_T|\}$ **do**
- 2: $\begin{bmatrix} x_r \\ y_r \end{bmatrix} = R_{\theta_E}(k) \begin{bmatrix} X_T[i] - \bar{x}_T(k) \\ Y_T[i] - \bar{y}_T(k) \end{bmatrix}$
- 3: **if** $d_{max} \leq |y_r|$ **then**
- 4: $d_{max} = |y_r|$
- 5: **end if**
- 6: **if** $x_{min} \geq x_r$ **then**
- 7: $x_{min} = x_r$
- 8: **end if**
- 9: **if** $x_{max} \leq x_r$ **then**
- 10: $x_{max} = x_r$
- 11: **end if**
- 12: **end for**
- 13: $p_{max} = \max(x_{max}, |x_{min}|)$
- 14: $l_1(k) = 2p_{max}$
- 15: $l_2(k) = 2d_{max}$

Since $a \geq b$

$$a \geq \sqrt{a \frac{V_{A_{max}}}{\omega_{max}}} \Rightarrow a \geq \frac{V_{A_{max}}}{\omega_{max}} \quad (29)$$

We also have a constraint on a and b given by (22). Imposing constraints from (22), (28) and (29), along with Lemma 6.1 we select the major and minor axes of the elliptical orbit as:

$$a(k) = \max\left(\frac{l_1(k)}{\sqrt{2}}, \frac{l_2(k)}{\sqrt{2}}, \frac{V_{A_{max}}}{\omega_{max}}\right) \quad (30)$$

$$b(k) = \max\left(\frac{l_1(k)}{\sqrt{2}}, \frac{a(k)V_{E_{min}}}{V_{E_{max}}}, \sqrt{\frac{a(k)V_{A_{max}}}{\omega_{max}}}\right) \quad (31)$$

7 Cooperation for equi-parametric formation

Each aerial agent indexed by $i \in \{1, \dots, N_A\}$ knows its own pose, i.e., $(x_{Ai}, y_{Ai}, \psi_{Ai})$, using a localisation source (GPS, motion capture, etc.), and computes its parameters s_{Ai} and γ_{Ai} , using (17), (18) and (20). In order to achieve an equi-parametric formation, each aerial agent broadcasts a communication packet (using the BROADCAST_PACKET function in Algorithm 2) containing the following:

- Its identification index $i \in \{1, \dots, N_A\}$
- Agent parameter s_{Ai}
- Ready flag fl_{Ri}

- Formation height flag fl_{Hi}
- Orbit flag fl_{Oi}

The guidance law of Section 4 guarantees that all aerial agents reach and latch onto the desired orbit. To avoid collisions, they are commanded to fly at different altitudes at initialization of the mission. For each communication packet received, the RECEIVE_PACKET function in algorithm 2 updates the array entries in $s_{A_{arr}}$, $fl_{R_{arr}}$, $fl_{H_{arr}}$ and $fl_{O_{arr}}$ at the index locations corresponding to the agent index i received in the packet. Using FIND_NEIGHBOUR function the agent i identifies the next agent i_N on the desired elliptical orbit along the direction of motion on ellipse, based on the parameter values received in $s_{A_{arr}}$. If two agents indexed i and j with $j > i$ have the same parameter value, the tie is broken by assigning the $i_N = j$ for agent i . To guarantee reaching an equi-parametric formation having parametric separation of $\Delta_s = \frac{2\pi}{N_A}$ between consecutive pairs of agents, we compute the velocity control component V_{Ci} as follows: *For agents $i \neq 1$:* We compute the parametric separation error (between actual separation of agent i from i_N and the desired separation Δ_s) and the resulting correction component V_{Ci} as

$$D_s = s_{Ai_N} - s_{Ai} - \Delta_s \quad (32)$$

$$V_{Ci} = k_s D_s \quad (33)$$

The function NORMALIZE in Algorithm 2 is used to normalize the parameter differences in the interval $[-\pi, \pi]$. If the separation $s_{Ai_N} - s_{Ai} > \Delta_s$, the commanded $V_{Ci} > 0$ and increases the speed of agent i move parametrically closer to i_N and vice-versa. Each agent raises its ready flag fl_{Ri} when D_s is small (starting from the agent i following agent 1).

For agent $i = 1$: Initially $V_{C1} = 0$ and agent 1 moves with linear speed corresponding to $\dot{s} = s_v$. This makes agent 1 behave like an anchor on the ellipse for constant parametric speed motion and the remaining agents attain their formation separations relative to agent 1. When the desired separations are attained within a threshold D_{Th} (say $D_{Th} = 0.1$), and all the other agents have raised the flag fl_{Ri} , then agent 1 also commands control speed V_{C1} according to (33). It sets the fl_{R1} and fl_{H1} flags on achieving the parameter separation within the threshold D_{Th} from its neighbour i_N . The setting of the fl_{H1} , causes all the agents to set fl_{Hi} , which in turn commands all agents to fly at the same altitude.

To prevent excessive step changes in the linear speed commands of the aerial agents due to changing neighbours, the above cooperation scheme is only initiated when the fl_{Oi} and fl_{Oi_N} flags are set indicating that the agent i and its neighbour i_N have reached sufficiently close to the desired orbit. The fl_{Oi} flag is set when the error $|\gamma_{Ai} - 1|$ is within some threshold γ_{Th} (say $\gamma_{Th} = 0.1$).

In cooperative approaches for target detection such as cyclic pursuit, there is a possibility that the final multi-agent formation attained at equilibrium has more than one agent at the same position on the orbit around the target (equi-parametric formations having separations Δ_s which are integer multiples of $2\pi/N_A$). Such formations are generally initial condition dependent (as shown in Fig. 9 and 12 in [17]). The cooperative approach proposed here avoids such scenarios. Also, the neighbours on the ellipse are assigned based on the order in which the agents reach the ellipse, and are not fixed a priori. This allows the agents to settle to the equi-parametric separations faster. Algorithm 2 gives the pseudo-code to implement the proposed cooperation strategy for each aerial agent in a decentralized manner.

Algorithm 2 AERIAL_AGENT_COOPERATION

Inputs: $i, s_{A_i}, \gamma_{A_i}, D_{Th}$
Functions: FIND_NEIGHBOUR, BROADCAST_PACKET, RECEIVE_PACKET, NORMALIZE
Initialization: $\Delta_s = 2\pi/N_A$, $fl_{Ri} = \text{False}$, $fl_{Hi} = \text{algofalse}$, $fl_{Oi} = \text{algofalse}$ for all $i \in \{1, \dots, N_A\}$
1: $[s_{A_{arr}}, fl_{R_{arr}}, fl_{H_{arr}}, fl_{O_{arr}}] = \text{RECEIVE_PACKET}()$
2: $i_N = \text{FIND_NEIGHBOUR}(s_{A_{arr}})$
3: $s_{A_{iN}} = s_{A_{arr}}[i_N]$, $fl_{R_{iN}} = fl_{R_{arr}}[i_N]$, $fl_{H_{iN}} = fl_{H_{arr}}[i_N]$, $fl_{O_{iN}} = fl_{O_{arr}}[i_N]$
4: $D_s = \text{NORMALIZE}(\text{NORMALIZE}(s_{A_{iN}} - s_{A_i}) - \Delta_s)$
5: $V_{Ci} = 0$
6: **if** $|\gamma_{A_i} - 1| < \gamma_{Th}$ **then**
7: $fl_{Oi} = \text{True}$
8: **end if**
9: **if** $fl_{Oi} = \text{True}$ **and** $fl_{O_{iN}} = \text{True}$ **then**
10: **if** $i_N = 1$ **and** $|D_s| < D_{Th}$ **then**
11: $fl_{Ri} = \text{True}$
12: **else if** $fl_{R_{iN}} = \text{True}$ **and** $|D_s| < D_{Th}$ **then**
13: $fl_{Ri} = \text{True}$
14: **end if**
15: **if** $i = 1$ **then**
16: $V_{Ci} = 0$
17: **if** $fl_{R_{iN}} = \text{True}$ **then**
18: $V_{Ci} = k_s D_s$
19: **end if**
20: **else**
21: $V_{Ci} = k_s D_s$
22: **end if**
23: **end if**
24: $\text{BROADCAST_PACKET}(i, s_{A_i}, fl_{Ri}, fl_{Hi})$

8 Simulations and Experiments

Table 2: Simulation and experiment cases and their parameters

	N_T	Convoy path	Convoy speed profile	V_{Tmax}	N_A	V_{Amin}	V_{Amax}	ω_{max}	d_c	k_s	k_ψ	k_γ	δ
MATLAB Simulation 1	6	Curve	Stationary	0	5	0.4	1.2	1.5	1	0.5	1.5	20	0.8
MATLAB Simulation 2	5	Lissajous curve	Time varying speed	0.2	6	0.3	1	1	-1	0.5	1	20	0.8
MATLAB Simulation 3	6	Way-points	Constant speed	0.1	7	0.25	0.8	1	1	0.7	1	20	0.8
SITL Simulation	5	Road network	Constant speed	0.15	4	0.3	0.8	1	1	0.5	2	20	0.8
Hardware Experiment 1	4	Lissajous curve	Time varying speed	0.08	3	0.3	0.7	1	1	0.5	10	4	0.8
Hardware Experiment 2	4	Way-points	Time varying speed	0.15	3	0.3	0.7	1	1	0.5	10	2.5	0.8

We validated the presented multi-agent convoy monitoring strategy in simulation and with real hardware experiments. We used unicycle kinematics and different convoy paths with smooth turns of different curvatures, sharp turns and other such non-linearities in the MATLAB simulation. Here we do not consider the z-axis separation as it is a simulation to validate the cooperative scheme to achieve the desired formation. Since most aerial platforms can be approximated as unicycle agents, our strategy can be used with fixed-winged UAVs, helicopters and quadcopters. To demonstrate this, the proposed strategy has been simulated with a ROS-Gazebo based Software-In-The-Loop (SITL) simulator, where the aerial agents follow the 6-DOF dynamical model of a quadrotor. The presented strategy has also been implemented using the Bitcraze Crazyflie 2.1 quadrotors (equipped with the flowdeck sensor) in a motion capture arena

for two different cases. Table 2 summarises the cases considered for the simulations and the experiments.

For the SITL simulation the ground convoy follows waypoints on a road network like path with sharp turns to simulate motion in an urban environment. The SITL setup is the same as the one used in [17], where in the 3DR Iris drone with the Pixhawk autopilot board is simulated. For the ground convoy we use a generic differential drive robot model. For both the SITL simulation and the experiments, the quadrotors have internal PID control loops which track linear velocity commands along the global coordinate frame axes, i.e., (V_x, V_y, V_z) and the angular velocity command ω_z about the vertical z axes. We approximate the quadrotor as a unicycle agent, by mapping the linear speed V_A and angular velocity ω_A given by (25) and (15) as follows:

$$\begin{aligned} V_x &= V_A \cos(\psi_A), \quad V_y = V_A \sin(\psi_A) \\ V_z &= k_z(z_{cmd} - z_A), \quad \omega_z = \omega_A, \end{aligned} \quad (34)$$

where z_{cmd} is the commanded height of the quadrotor and z_A is height of the of the quadrotor in the global reference frame. The PID loops in the SITL simulation and the Crazyflie 2.1 quadrotor track the commanded linear speed and angular velocity satisfactorily as shown in Fig. 3.

For the experiments we used small differential drive robots (designed in-house, refer Remark 8.1) to move on the convoy paths. The positions of the convoy robots and the quadrotors are available from the motion capture system comprising of eight Vicon MX T40 motion capture cameras. The control implementation is decentralised in the sense that the control code for each convoy robot and each quadrotor is run as a separate ROS Node. The control node of each quadrotor receives its pose feedback and position coordinates of the convoy robots from the Vicon system. It cooperates the control nodes for other quadrotors by broadcasting the BROADCAST_PACKET described in Section 7 on ROS topics. The control node for each quadrotor publishes motion commands to the Crazyflie 2.1 quadrotors via the Crazyradio USB dongle-antenna. The ROS package used to facilitate this is the crazyflie.ROS package [72]. The ROS node controlling each robot gets its pose information from the Vicon system and sends motion commands to the corresponding robot over a Wi-Fi network. The Raspberry Pi 3B computer on each robot executes these commands. The experimental setup is illustrated in Fig. 4.

Due to space limitations, we only discuss the plots of Experiment 2 given in Fig. 5. The plots for the remaining simulation and experiment cases can be found in the videos, with web-links given in Remark 8.1. From Fig. 5, we see that after an initial transient duration the value of γ_A (given by (20)) remains close to 1, indicating that all the aerial agents faithfully traverse the desired elliptical orbit as they monitor the moving convoy. D_s given by (32) is the error between the actual s_A parameter separation, and the desired parametric separation of $\Delta_s = 2\pi/N_A$ between each agent i and its neighbour i_N . We see from Fig. 5 that the error D_s remains in the neighbourhood of 0, indicating that the aerial agents achieve and maintain a formation that divides the parametric length of the ellipse onto equal parts of length $\Delta_s = 2\pi/N_A$. The z_A plot shows the initial altitude separation between the quadrotors for collision free motion on the ellipse. Equal altitudes are commanded when the fl_{Hi} flag is set for each of the quadrotors according to the Algorithm 2. The commanded V_A and ω_A plots show that the linear speed and angular velocity constraints of the aerial agents are always enforced. A similar explanation holds for the plots of the remaining simulations and experiment.

Remark 8.1. *The video of the simulations and experiments can be found at the following web-links:*

MATLAB Simulations: <https://youtu.be/SDaYUSqzGLY>

SITL Simulation: <https://youtu.be/1l0vCzmoZeU>

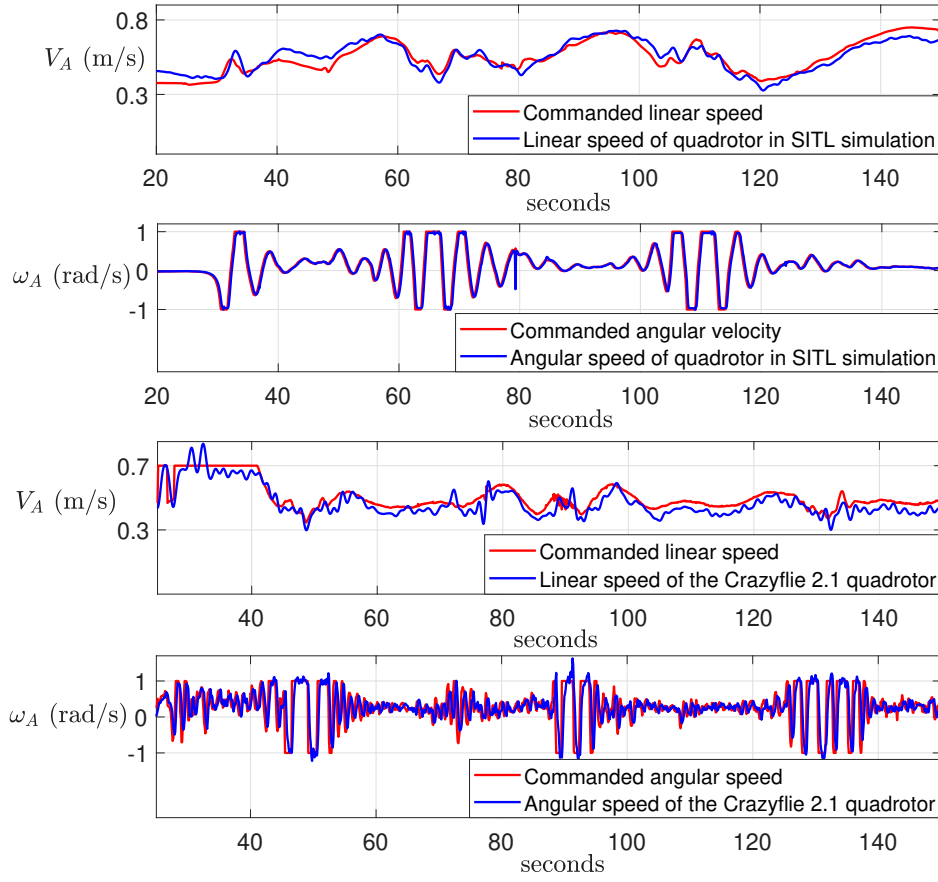


Figure 3: PID tracking performance for quadrotors in the SITL simulation and the experiments

Experiments: <https://youtu.be/6o5ywP-lu3k>

Convoy robot design: <https://youtu.be/BBLrnqkzpAg>

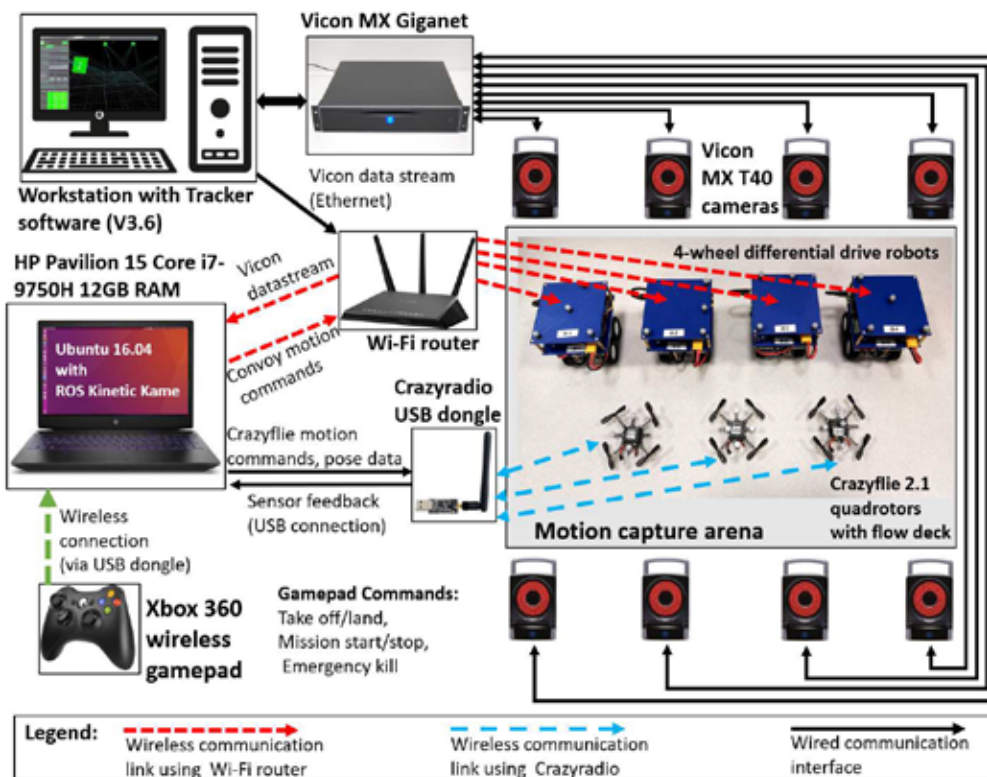


Figure 4: Experiment Setup

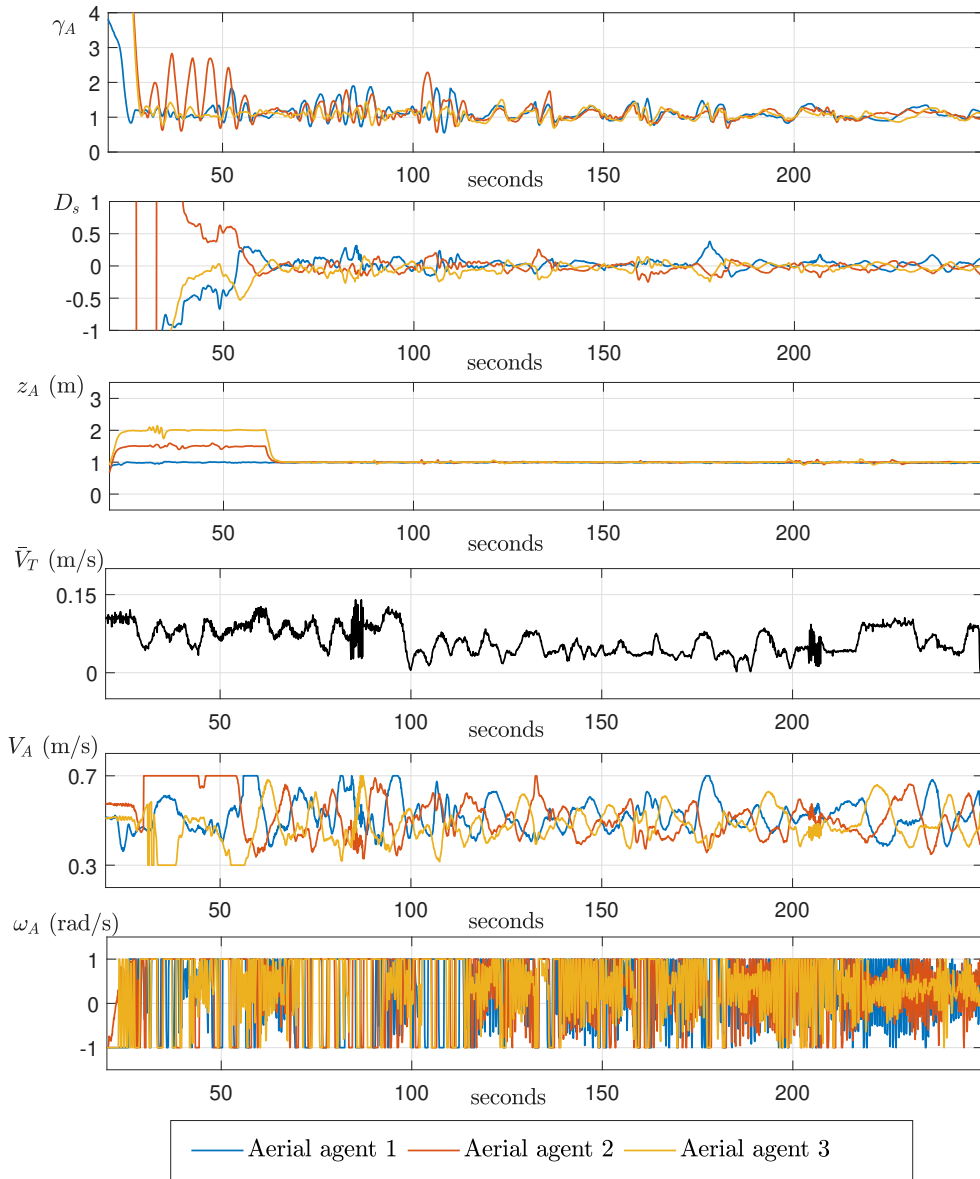


Figure 5: Plots for Experiment 2

Conclusions

We have proposed a multi-agent cooperative scheme for a team of UAVs to fly on slowly varying elliptical orbits to monitor a ground convoy moving on an arbitrary continuous trajectory. The scheme computes a moving ellipse that encompasses the moving convoy. By using vector field guidance strategy, the aerial agents converge to a trajectory that traverses the moving elliptical orbit repeatedly. To achieve and maintain a multi-agent formation on the elliptical orbit that divides its parametric length into equal partitions, a cooperative control scheme to modulate the linear speed of the aerial agents is proposed. Also an algorithm is developed to ensure convergence to the desired formation and to avoid grouping of aerial agents at the same position due to input saturation. The proposed scheme maintains an approximately equi-parametric formation over the time varying elliptical orbit around the moving convoy. The proposed strategy has been validated by simulations in MATLAB and with a realistic simulation in a ROS-gazebo based SITL simulator for quadrotors. It has also been experimentally validated with quadrotors and ground robots in a motion capture arena.

Part II

Progress in the Year 2017-18

1 Using Reinforcement Learning as Helper to Humans in Difficult Multi-Agent Environments

The recent development of Artificial Intelligence (AI) and in particular Reinforcement Learning (RL) demonstrate that AI agents can learn to complete a variety of tasks using a reward feedback. It is shown that RL agents utilizing Q-learning outperform humans in tasks such as Atari Games [105] [104] [61], excelling at games such as Breakout where coordination is imperative. However, these learning agents were trained over the course of several days to achieve a level of performance which humans can achieve in orders of magnitude less time. Later work has improved the learning rate, and achieved comparable performance with less data [65], but the disparity in the learning rate remains. Because large amount of training data and processing time are required to reach this level of performance, application of RL is largely limited to simulated environments which may not reflect the real world [21]. Substantially more complex tasks such as real-time strategy (RTS) computer games, stock markets and real robots are to this day intractable for online learning algorithms and require multi-level solutions. Such systems include model-based training, graph search, reinforcement learning, Monte-Carlo search, learning from demonstration [148].

In addition, the learning agents requires a very careful reward function selection and parameter tuning to be able to converge to a solution. These make the solution individual for each problem and cannot be easily transferred from one to another. Many applications, especially those involving human-interaction, still require a significant performance improvement in the learning algorithms. It is our big interest to solve the complex tasks competitively to a human in, for instance, military operation planning and computer game AI.

In our work, we compared the performance of human learners to that of Q-learning agents for specially designed grid-world decision-making problems with varying transition dynamics [39a]. The experiments demonstrated that the prior context knowledge is important for human learning, and that it is one of the essential instruments humans use to solve complex tasks. We showed that humans relied on intuition to map the transitions dynamics, and that they expected

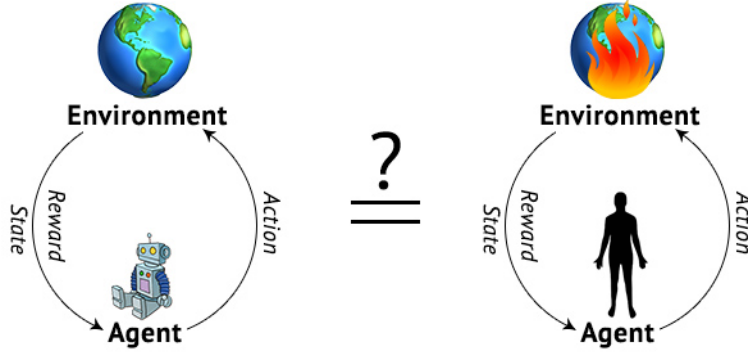


Figure 6: Reinforcement Learning Framework

their intuition to be correct. Once their intuition failed, humans tried to fit the new observed transitions into their prior beliefs, and to update the model of the environment. This caused a significant reduction in learning performance, and delayed the convergence of the learned policy. The experiments also demonstrated that, once they found that transitions were non-intuitive, humans switched to a random policy based exploration with no motivating intuition. In all the above experiments, RL agents showed a unique learning rate. In addition, our results demonstrated that human performance improved in non-intuitive tasks when suggestions were given from a tabular Q-learning algorithm running in the background.

To throw together human and AI, we developed a complex dynamic environment so called Capture the Flag (CtF) that mimics multi-player computer games [11](#). In this simulation, two teams compete against each other trying to capture the enemy flag in conditions of partial information and visibility. The goal of each team is to survive, destroy the enemy or capture the enemy flag. This simulation allows to compare different scripts and RL algorithms by playing them against each other or against a real human. The environment simulates a partially observable problem that provides a map with the hidden and visible areas available for the user or planning algorithm. The fully observable map is available to spectators for demonstration purpose only and represents the whole state space of the environment. The simulation setup is easily scalable in size of the map, number of units and complexity. By default, each team controls 4 ground units and 2 aerial units. These two types of units have different abilities what adds a non-homomorphic problem.

The developed package includes sample solutions that may challenge other project participants. These solutions carry simple script-based behaviors and common RL-solutions adapted for this problem. These algorithms may serve as a building bricks for advanced layered solutions. For instance, the included vanilla policy gradient algorithm that controls each unit independently but sharing the model between units. The performance of this policy is shown in Fig. [11a](#). This algorithm forms a decentralized distributed solution that is not optimal with respect to team-based policies. The layered solution, currently under testing and development, involves egocentric hierarchical learning where each unit plans only based on the information around it and with respect to the commands from the centralized system [11b](#).

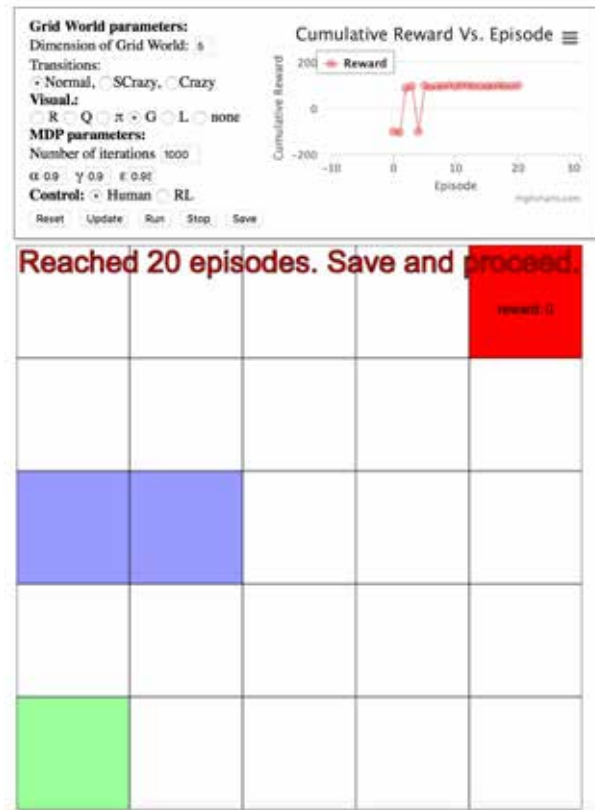
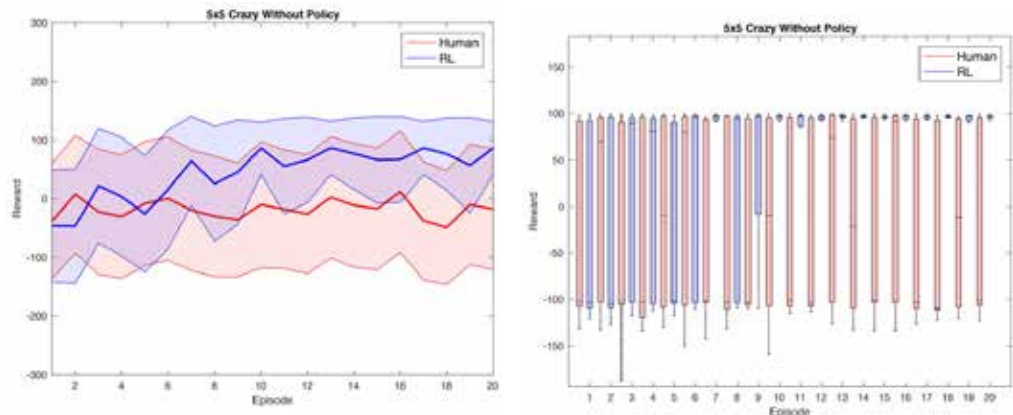
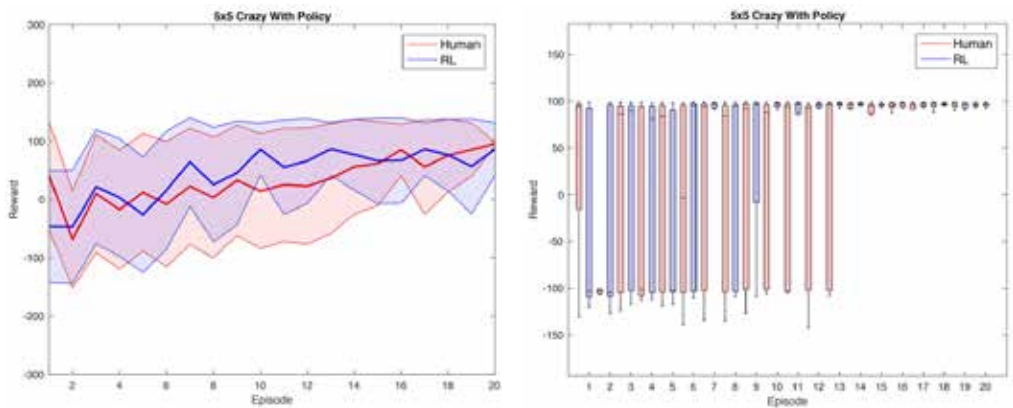


Figure 7: This is a screenshot of the web app implementing the user interface. Users can choose between human and RL control. The results are displayed in the graph at the top of the screen. The visualization shows the reward, the Q function, the learned policy π , or the goal and pit locations. The user can control the transition mode.



(a) Stat plot for human (red) and RL agent (blue), for a 5x5 grid world, with fully random transition dynamics. (b) Box plot for human (red) and RL agent (blue), 5x5 grid world, random transition dynamics.

Figure 8: Task with Crazy Transitions and RL Policy



(a) Stat plot for human (red) and RL agent (blue), 5x5 grid world, random transitions, RL policy suggestions. (b) Box plot for human (red) and RL agent (blue), 5x5 grid world, random transitions, RL policy suggestions.

Figure 9: Task with Crazy Transitions and RL Policy

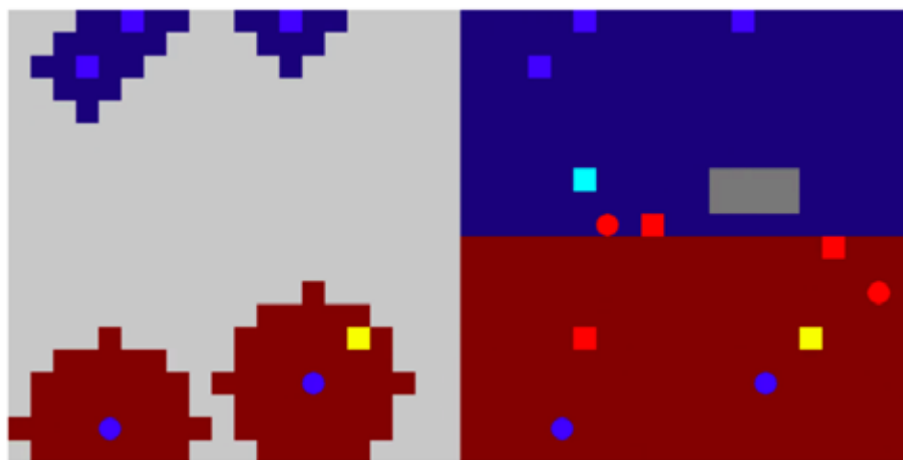


Figure 10: Python-based environment Capture the Flag (CtF) is designed to throw together various AI algorithms or a human.

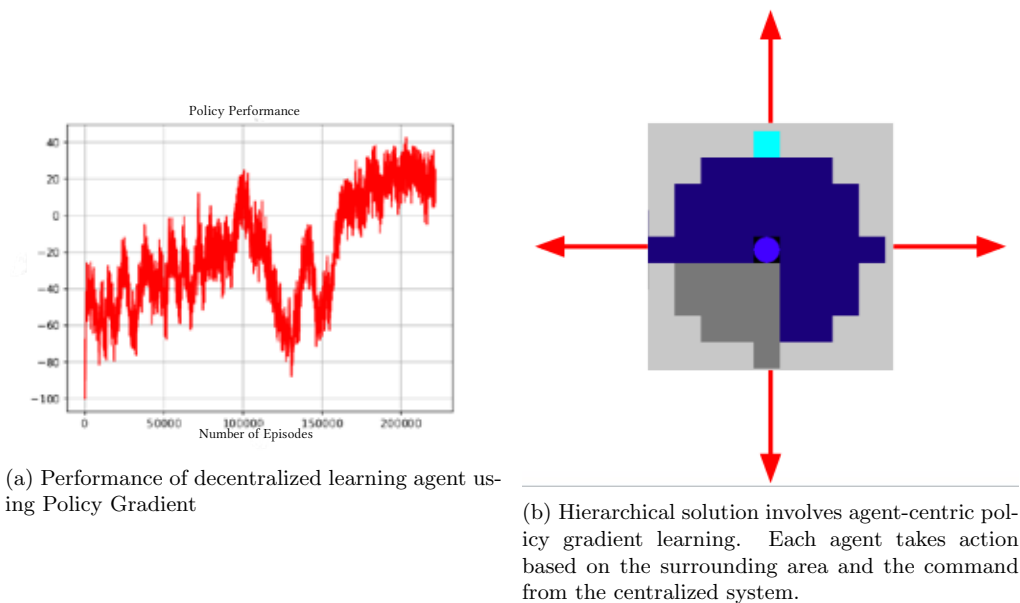


Figure 11: Capture the Flag

2 Harnessing Adversarial Attacks to Improve Robustness of Deep Reinforcement Learning

This Section focuses on engineering adversarial attacks and then improving the robustness of Deep Reinforcement Learning algorithms (DRL) to parameter uncertainties with the help of these attacks. We show that even a naively engineered attack successfully degrades the performance of DRL algorithm. We further improve the attack using gradient information of an engineered loss function which leads to further degradation in performance. Furthermore, these attacks are leveraged during training to improve the robustness of Reinforcement Learning (RL) within robust control framework. We show that this adversarial training of DRL algorithms like Deep Double Q learning and Deep Deterministic Policy Gradients leads to significant increase in robustness to parameter variations for RL benchmarks such as Cart-pole, Mountain Car, Hopper and Half Cheetah environment.

2.1 Introduction

Advances in Deep Neural Networks (DNN) has a tremendous impact in addressing the curse of dimensionality in RL and offers state of the art results in several RL tasks ([91], [145], [95], [105], [147]). However, it has been shown in [59] that DNN can be fooled easily into predicting wrong label by perturbing the input with adversarial attacks. It opens up interesting frontier regarding robustness of machine learning algorithms in general. Robustness assumes greater importance in the context of robotics and safety critical systems where such adversarial noise may lead to undesirable and hazardous situations. Learning robust and high performance policies for continuous state-action reinforcement learning domains is critical to enable successful adoption of Deep Reinforcement Learning (DRL) in robotics, autonomy, and control problems. More specifically, robustness to real world parameter variations, such as changes in the weight, friction, or other environmental parameters of the dynamical system are critical.

We address these challenges in an adversarial training framework. We first engineer “optimal” attack on the DRL agent and then leverage these attacks during training that leads to significant improvement in robustness and improves policy performance in challenging continuous domains. Our approach is loosely inspired from the idea of robust control, in which the best case policy is sought over the set containing the worst possible parameters of the system. We translate this into a problem of best performing policy trained in presence of adversary. The key difference, however, is that while robust control approaches tend to be conservative, our approach leverages the inherent optimization mechanisms in DRL to enable learning of policies that have even higher performance over a range of parameter and dynamical uncertainties like friction, mass etc.

In particular, the contributions of this paper are two-fold: First, we propose an objective function specifically in a reinforcement learning setting whose optimization degrades the performance of RL algorithms. Furthermore, we show that an attack designed to utilize the minima of our objective function ensures that the RL agent subject to the attack is fooled into thinking that it is in a state that leads to the worst possible action for its current state. This in itself is interesting and exposes critical robustness issues with prevalent DRL algorithms. These attacks severely degrade the performance during evaluation of RL policies. Interestingly, we observe that linearly parameterized RL algorithms are more robust to such adversarial attacks as compared to DRL. Our second contribution is improving the robustness of DRL algorithm to parameter uncertainties. We train the DRL with proposed engineered adversarial attack and show that it becomes robust to parameter or model uncertainties, that is, it shows significant improvement in performance across a wide range of parameters. Specifically, we train the DRL adversari-

ally on "default" parameters, then this adversarially trained agent is tested on a wide range of parameters (and performs much better than baseline).

The paper is organized as follows. We provide introduction and related work that has been done in Section 2.1. The background has been discussed in Section 2.2. Adversarial attacks and their use for improving robustness have been described in Section 2.3, the experimental setup has been provided in Section 2.4 and results have been presented in Section 2.5. Finally, concluding remarks and future directions have been discussed in Section 2.7.

2.1.1 Related Work

In [74], interesting results have been presented regarding adversarial attack on reinforcement learning algorithms. The adversarial attacks in [74] are similar in spirit to fast signed gradient Method (FSGM) ([59]). In [74], the probability of an image being classified with a label is replaced by the probability of taking an action as in [59] and results were generated using RL algorithms for Atari game environment (using image observation). As we shall show in Section 2.2.2, the loss function presented in this paper is guaranteed to maximize the probability of taking the worst possible action which is not necessarily the case in [74]. In [74], the adversarial noise is generated without being certain (when the noise is generated) that it indeed causes adversarial attack. In contrast, we exploit the value function to ascertain the efficacy of our attack. In [81], the authors have also used FSGM style of attack to cause performance failure in Atari game using Asynchronous Advantage Actor Critic (A3C) ([103]) algorithm. They reduced the frequency of attack by injecting noise only when trained value function is above a certain threshold. The intuition being that the agent should be disrupted only when it is more likely to win the game. They retrained using adversarial attacks and found that it becomes more resilient to adversarial attacks. [96] proposed attack that is similar in spirit to [24] attack on images. In enchanting based attack, they tried to lure agent in bad states by first using a predictive model of video to try to find a sequence of actions that leads to good state. Then a random sequence of actions were generated which might to bad state, perturbation are generated so that the actions that agent takes is similar to this random sequence of action. However, it is interesting to note that the value function (of trained agents) itself contains information about this possible worst sequence of action. Hence, we have used value function instead to find adversarial action rather than trying to follow another random sequence of action. All these attacks have been performed on trained agents.

An important point regarding prior work on adversarial attacks for DRL ([74], [81], [96]) is that all of them have been performed in environments that use images, that is, high dimensional pixel input. It has been argued ([59]) that images are susceptible to attack given the high dimensional space of pixel inputs. Thus, most attacks on RL algorithms are similar in spirit to attacks used in case of images. Our work for RL algorithm is not restricted to high dimensional image. We also point out concurrent work by [101]. However, they use a heuristic in objective function (minimize $\|u\|_2^2$ where u is the action or control input) for adversarial attack which does not ensure that the attack is optimal as compared to our proposed objective for adversary that is optimal attack.

In [129], an ensemble of models was for robust reinforcement learning. They sample model parameters and perform trajectory rollout with those parameter variations. Further, parameters are selected to be trained with based on worst performing percentile criteria. However, sampling trajectories with uncertain parameters can be risky. Moreover, sampling may be very difficult for a large number of parameters. Another approach to robust reinforcement learning has been provided in [106] and it has been extended with deep network in in [121]. Their objective was also to sample worst performing percentile trajectories. To achieve this, adversary and RL agent

are trained alternatively with an expectation that the agent becomes robust to adversary. This is similar to max-min formulation of robust control. However, finding equilibrium of max-min formulation can be difficult. Our work provides a direct approach for sampling worst performing trajectories wherein the adversary fools the agent into believing that it is in states which lead the agent into taking bad actions leading to sampling of worst performing trajectories. In concurrent work, [101] have proposed adversarial perturbation of states and robust training using this perturbation. However, the objective function that adversary needs to optimize is heuristic and is given by $(\|u\|_2^2)$ where u is the control input. They have reported high variance and the results that also have been reported in [101] is comparison of highest return over several random agents as opposed to averaging out the returns for all the agents.

Another relevant body of work in reinforcement learning is the risk sensitive reinforcement learning where the reward is augmented with risk. Risk can be seen as variance of long-term return. Here, higher variance implies more instability and hence, greater risk. The various risk criterion used in literature are variance penalized ([60]), weighted risk and return criterion ([56]) etc. A survey of risk sensitive RL can be found in [53]. However, these methods do not scale well with respect to space and action complexity as explained in [53].

2.2 Background

We briefly review the adversarial attacks on images ([59]) because they have been extended to image based DRL ([74], [81], [96]). We also review some of the algorithms of DRL that have been used in this paper such as Deep Q Learning (DQN), Double Deep Q Learning (DDQN), and Deep Deterministic Policy Gradient (DDPG).

2.2.1 Adversarial attack on Deep Learning Network classifiers (Fast Signed Gradient Method)

One of the most popular ways to engineer adversarial attacks on deep learning classifiers (that have been extended to DRL) is fast signed gradient method (FSGM) ([59]) where a cost function is crafted whose optimization leads to increase in the probability of the network classifying a given image with a wrong label. It takes into account a linear approximation of deep learning model and engineers an attack. Assuming a linear model, $f(x) = w^T x$ (x represents the input and $f(x)$ represents the output), the change in output due to perturbation of input by an amount η is given by $\tilde{f}(x) = w^T x + w^T \eta$. We can get maximal perturbation in prediction with

$$\eta_{min} = \epsilon \text{sign}(w)$$

Here η_{min} represents the best possible adversarial perturbation which is l_∞ norm constrained (ϵ). However, this attack has also been extended to nonlinear in parameter functions (multi-layer deep neural networks) and has successfully fooled the classifier networks into confident misclassification of images. The adversarial attack is

$$\eta_{min} = \epsilon \text{sign}(\nabla_x J(\theta, x, y))$$

Here, $J(\theta, x, y)$ is the respective loss function that is used during training or testing. Again with the underlying assumption that adversarial noise is l_∞ norm bounded. In case of images, $J(\theta, x, y)$ is the cross entropy loss between the true image label and predicted distribution over labels of image

2.2.2 Deep Q learning (DQN) and Deep Double Q Learning (DDQN)

Deep Q learning (DQN) has achieved superhuman results on Atari games [105]. Q learning is a value function based algorithm (we refer to both state-action value function and state-value function as value function). Q values of a state-action pair represent how good the action being contemplated by the agent is for its current observation. The learning agent updates the Q value using temporal difference error and simultaneously acts to maximize its return in the long run. In DQN algorithm, the agent uses a Deep neural network to approximate this Q function. The DQN algorithm achieved stability in training primarily by using experience replay and the use of target network. For experience replay, it stores past state, action, reward, next state sequence and these are used to update the Q network just as in supervised learning with these sequences being picked randomly from memory. This breaks the strong correlation between samples. For this “supervised” learning type of update, DQN has another neural network, called target network which provides the Q values for the next state(s). The target network is updated by hard transfer of online weights after a fixed number of iterations. This two network structure led to “stability” in training. However, [162] showed that DQN overestimates the Q values and proposed Double Deep Q Learning (DDQN) where the action selection was still performed by online network but its value estimate for update was done using the target network. This mitigated the overestimation of value function.

2.2.3 Radial Basis Function based Q learning

In radial basis function based Q learning ([58]), the DNN is replaced by RBF with gaussian kernel. Here experience replay and target networks are typically not used. The networks learns through stochastic gradient descent using temporal difference error.

2.2.4 Deep Deterministic Policy Gradient (DDPG)

Deep deterministic policy gradient (DDPG) ([95]) uses both an actor and a critic for learning. The critic evaluates a given policy generated by the actor. The weights of critic and actor are updated by gradient descent. The critic network uses the prior concepts of experience replay and target network for its update. The only difference between these update and that of Deep Q learning is that instead of “hard” transfer of weights to target after a fixed number of iterations, there is “soft” transfer of weights where the weights of target network are incremented by a very small amount towards the online network.

2.3 Method

In this section, we explain our methods for adversarial attack and further use these attacks for significantly improving the robustness and performance of reinforcement learning algorithm.

2.3.1 Adversarial Attack

In this subsection, we concentrate on generating adversarial attacks which will cause a trained RL agent to fail. We assume that the mechanism of the attack is by corrupting the observations of the agent of its current state, fooling it into believing it is in a state that causes it to follow a sub-optimal policy for its actual state. We begin by defining what we mean by an adversarial attack in the context of value function based reinforcement learning algorithms:

Definition 2.1. *An adversarial attack is any possible perturbation that leads the agent into increased probability of taking “worst” possible action in that state. Here, the “worst” possible action for a trained RL agent is the action which corresponds to least Q value in that state.*

It is important to note that Def. 2.1 is valid only for value function based algorithms, that is, the algorithms that use value function to predict the optimality of actions in a given state. Most of the popular state-of-art algorithms such as A3C and DDPG are value function based. We shall see the objective function that needs to be optimized for achieving adversarial attacks in the context of reinforcement learning in section 2.2.2. There is a subtle but important difference as compared to image classification wherein attack is considered successful if a given image is classified as any other image. There is no concept of “worst” possible image whereas RL agents can have “worst” possible action. Another assumption is that the agent has been trained sufficiently well so that Q values are close to optimal. This is true in case of tabular Q learning as training time tends to infinity (but this has not been proved in general for nonlinear function approximators like DNN) and serves as motivation for this assumption.

An important point to note is that throughout the paper we have bounded the adversarial attack noise by l_2 norm constraint. It is possible to use other norm constraints as well and [119] showed that distillation is secure under l_∞ . Another point is that the RL environments are scaled so that the state space is normalized between $[0, 1]$. The magnitude of attack is also normalized.

2.3.1.1 Naive adversarial attack

First, we propose a naive method of generating adversarial attack. The idea behind this attack is to generate random noise (several times in current state that the agent is in) and add it to current state with hope that one of these noise samples will cause the agent to take “bad” action, that is, sub-optimal action. The quality of attack can be ascertained by the value function. Algo. 12 outlines the naive adversarial attack for DDQN.

An important point to note is that the attacks are generated during evaluation phase. The adversarial attack is essentially a search across nearby observation which will cause the agent to take wrong action. For generating adversarial attack on the DRL policies, we sample a noise with finite (small) support. Noise is not generated once during every iteration, rather it is sampled for a number of times every iteration with a search for best adversarial noise (β distributed noise with shifted 0 mean was used for generic setting but in our experiments the parameters of β distribution were $(1, 1)$ which corresponds to uniform noise with 0 mean). The particular noise that causes least estimate of the value function is selected as adversarial noise. Then this noise is added to the current observation. Algo. 12 is outlined for naive attack on DDQN.

For naive attack on DDPG, the critic network can be used to ascertain value functions when required and actor network determines the behavior policy to pick action. Thus, the objective function used by adversary in this case is the $Q_{critic}^*(s, a)$, that is, the value function determined by the trained critic network. Algorithm for naive attack on DDPG has been provided in Algo. 4 and is similar to Algo. 12.

2.3.1.2 Gradient based adversarial attack

In this subsection, we show that a proposed cost function different from the one used in traditional FSGM ([74]) is more effective in finding worst possible action in the context of reinforcement learning with discrete actions.

Theorem 2.2. *Let the optimal policy be given by conditional probability mass function (pmf) $\pi^*(a|s)$, the action which has maximum pmf be given as a^* and the worst possible action be given*

Algorithm 3 Naive attack (DDQN)

```
 $a^* = \arg \max_a Q(s, a), Q^* = \max_a Q^{target}(s, a)$   
for  $i = 1 : n$  do  
   $n_i \sim \text{beta}(\alpha, \beta) - 0.5$   
   $s_i = s + \epsilon * n_i$   
   $a_{adv} = \arg \max_a Q(s_i, a)$   
   $Q_{adv}^{target} = Q^{target}(s, a_{adv})$   
  if  $Q_{adv}^{target} < Q^*$  then  
     $Q^* = Q_{adv}^{target}$   
     $s_{adv} = s_i$   
  else  
    do nothing  
  end if  
end for  
return  $s_{adv}$ 
```

Algorithm 4 Naive attack (DDPG)

```
1:  $a^* = U(s), Q^* = Q^{target}(s, a^*)$   
2: for  $i = 1 : n$  do  
3:    $n_i \sim \text{beta}(\alpha, \beta) - 0.5$   
4:    $s_i = s + \epsilon * n_i$   
5:    $a_{adv} = U(s_i)$   
6:    $Q_{adv}^{target} = Q^{target}(s, a_{adv})$   
7:   if  $Q_{adv}^{target} < Q^*$  then  
8:      $Q^* = Q_{adv}^{target}$   
9:      $s_{adv} = s_i$   
10:  else  
11:    do nothing  
12:  end if  
13: end for  
14: return  $s_{adv}$ 
```

by a_w . Then the objective function whose minimization leads to optimal adversarial attack on RL agent is given by

$$J(s, \pi^*) = - \sum_{i=1}^n p_i \log \pi_i^*$$

where $\pi_i^* = \pi^*(a_i|s)$, $p_i = P(a_i)$, the adversarial probability distribution P is given by

$$P(a_i) = \begin{cases} 1, & \text{if } a = a_w \\ 0, & \text{otherwise} \end{cases} \quad (35)$$

In other words, this is the cross entropy loss between the adversarial probability distribution and optimal policy generated by the RL agent

Q values can be converted into pmf by passing them through softmax function.

Proof. We shall show that $\min_s J(s, \pi^*)$ achieves the objective of def. 2.1.

$$\begin{aligned} J(s, \pi^*) &= - \sum_{i=1}^n p_i \log(\pi_i^*) \\ &= -p_w \log(\pi_w^*) = -\log(\pi_w^*) \quad (\text{from 93}) \\ \implies \min_s J(s, \pi^*) &= \min_s -\log(\pi_w^*) \\ \min_s J(s, \pi^*) &= \max_s \log(\pi_w^*) \end{aligned}$$

Since log is monotonically increasing,

$$\min_s J(s, \pi^*) = \max_s \pi_w^*$$

Thus, we have shown that the objective function that should be used for engineering attack on RL algorithm should be given by Theorem 2.2 as it is consistent with Def. 2.1. FSGM algorithm can be used to minimize this objective function. We must point out that this objective function is different from ones in literature 74. The objective functions mentioned in 74 will result in $\min_s \pi^*(a^*|s)$ (a^* is the best possible action for given state s). This leads to decrease in the probability of taking best possible action. This won't necessarily lead to increase in probability of taking worst possible action. The gradient based attack for DDQN has been explained in Algo. 13.

The gradient based attack for DDPG is similar to DDQN with the objective function that adversary need to minimize being given by the optimal value function of critic ($Q^*(s, a)$). Here the gradient is given by

$$\nabla_s Q^*(s, a) = \frac{\partial Q^*}{\partial s} + \frac{\partial Q^*}{\partial U^*} \frac{\partial U^*}{\partial s}$$

where U^* represents the optimal policy given by actor. The algorithm has been provided in Algo. 6 and is similar to Algo. 13.

2.3.1.3 SGD based attack

We also used Stochastic Gradient Descent approach for adversarial attack wherein instead of sampling a few times and selecting best possible attack amongst these samples, we followed the gradient descent for same number of sampling time and selected the state that we end up in as adversarial state.

Algorithm 5 Gradient based attack (DDQN)

```
1:  $a^* = \arg \max_a Q(s, a), Q^* = \max_a Q^{target}(s, a)$ 
2:  $\pi^{target} = softmax(Q^{target})$ 
3:  $grad = \nabla_s J(s, \pi^{target})$ 
4:  $grad\_dir = \frac{\nabla_s J(s, \pi^{target})}{\|\nabla_s J(s, \pi^{target})\|}$ 
5: for  $i = 1 : n$  do
6:    $n_i \sim beta(\alpha, \beta)$ 
7:    $s_i = s - n_i * grad\_dir$ 
8:    $a_{adv} = \arg \max_a Q(s_i, a)$ 
9:    $Q_{adv}^{target} = Q^{target}(s, a_{adv})$ 
10:  if  $Q_{adv}^{target} < Q^*$  then
11:     $Q^* = Q_{adv}^{target}$ 
12:     $s_{adv} = s_i$ 
13:  else
14:    do nothing
15:  end if
16: end for
17: return  $s_{adv}$ 
```

Algorithm 6 Gradient based attack (DDPG)

```
1:  $a^* = U(s), Q^* = Q^{target}(s, a^*)$ 
2:  $grad = \nabla_s Q^{target}(s, a)$ 
3:  $grad\_dir = \frac{\nabla_s Q^{target}(s, a)}{\|\nabla_s Q^{target}(s, a)\|}$ 
4: for  $i = 1 : n$  do
5:    $n_i \sim beta(\alpha, \beta)$ 
6:    $s_i = s - n_i * grad\_dir$ 
7:    $a_{adv} = U(s_i)$ 
8:    $Q_{adv}^{target} = Q^{target}(s, a_{adv})$ 
9:   if  $Q_{adv}^{target} < Q^*$  then
10:     $Q^* = Q_{adv}^{target}$ 
11:     $s_{adv} = s_i$ 
12:   else
13:     do nothing
14:   end if
15: end for
16: return  $s_{adv}$ 
```

2.3.2 Adversarial Training through Robust Control framework

2.3.2.1 Robust Control Framework

In RL, the typical objective that an agent seeks to maximize is its expected long term return (η) (over possible trajectories τ) assuming a fixed transition model $T(s_t, a_t; \phi)$ characterized by parameters ϕ

$$\eta(\pi, T) = \mathbb{E}_{\tau} \left[\sum_{t=0}^T \gamma^t r(s_t, a_t) | s_0, \pi, T \right]$$

However, if there is variation in transition model, then criteria might be to perform well in expectation over all the possible transition models. Thus, leading to optimization of the mean performance of agent. The objective function in this scenario can be modified to

$$\eta(\pi) = \mathbb{E}_T[\eta(\pi, T)]$$

This is popularly known as risk neutral formulation. However, it has an underlying assumption that the distribution over transition model parameters are known apriori. It may not perform well over the transition model distributions because of high variance in transition model distribution. Thus, conditional value of risk (CVaR) can be used as optimization criteria for robust control ([155])

$$\eta_{RC}(\pi) = \mathbb{E}_T[\eta(\pi, T) | \mathbb{P}(\eta(\pi, T) \leq \beta) = \alpha]$$

So, the problem boils down to maximizing the expected return over worst α percentile of returns. Thereafter, for sampling these bad trajectories, [129] changed transition model parameters and sample trajectories by performing rollouts with different parameters. [106] and [121] took an indirect approach where instead of sampling worst trajectory from rollout, they employ an adversary which applies control action and tries to push the RL agent into possible bad states. They train the adversary whose reward was negative of the reward of RL agent resulting in max-min game theoretic formulation. But it is usually difficult to find this equilibrium.

2.3.2.2 Adversarial Training

In contrast to approaches given in [129] and [106] where it might be difficult to find equilibrium, we take a direct approach where the adversary fools the agent into believing that it's in a "fooled" state different from actual state such that the optimal action in "fooled" state leads to worst action in actual current state. In other words, the adversary fools the agent into sampling worst trajectories directly. We first train the algorithm using "vanilla" DRL (DDQN or DDPG), the trained agent is then made robust to model uncertainties through adversarial training. We have used gradient based attack for adversarial training as it performed best amongst all attacks (results presented in Section 2.5). Adversarial training algorithm has been discussed in Algo. 14 (DDQN) and Algo. 8 (DDPG). In our approach, worst α percentile of returns is related to the magnitude of adversarial attack, higher adversary magnitude corresponds to optimization for higher α worst percentile.

2.4 Experimental setup

2.4.1 DDQN

The Deep Double Q learning for cart pole environment used 3 layers of 16 units each with Rectified Linear Unit (ReLU) activation function whereas the mountain car environment used

Algorithm 7 Training with adversarial perturbation (DDQN)

```
1: for  $i = 1 : iterations$  do
2:   Reset the environment and receive observation
3:   while not terminal or not max time steps per episode reached do
4:      $s_{adv} = Grad(Q^{target}, Q, s, \epsilon, n, \alpha, \beta)$ 
5:      $a = \arg \max_a Q(s_{adv}, a)$ 
6:      $s, r = Env(a, s)$ 
7:     Update the weights of network according to DDQN algorithm
8:   end while
9: end for
```

Algorithm 8 Training with adversarial perturbation (DDPG)

```
1: for  $i = 1 : iterations$  do
2:   Reset the environment and receive observation
3:   while not terminal or not max time steps per episode reached do
4:      $s_{adv} = Grad(Q^{target}, U, s, \epsilon, n, \alpha, \beta)$ 
5:      $a = U(s_{adv})$ 
6:      $s, r = Env(a, s)$ 
7:     Update the weights of network according to DDPG algorithm
8:   end while
9: end for
```

2 hidden layers of 100 units each of ReLu activation function. The discount factor for both of them was set at 0.99 and target network update rate were 10^{-2} . The “supervised learning” of networks was done with Adam optimization and learning rate of 10^{-3} . The cartpole environment was trained for 50000 timesteps while Mountain Car was trained for 40000 time steps. The repository that we used was Keras-rl ([122](#)).

2.4.2 RBF

For Cart-pole, each dimension of state input was divided into 3 bins (b). The centroids were uniformly distributed along those bins. The variance of radial activation were $\frac{2}{b^2}$. Discount factor of 0.99 was used. The learning rate was given by 0.001. It was trained for 40000 time steps For Mountain car environment, the learning rate was 0.01 and number of bins were 4. The discount factor was 0.99. Total number of timesteps were 60000

2.4.3 DDPG

For hopper and half cheetah environment, there were 2 hidden layers of 400 and 300 ReLu units for both actor and critic networks. The number of time steps it was trained were 1 million. Discount factor was 0.99. The learning rate of critic network was 10^{-3} while the learning rate of actor was 10^{-4} . Half cheetah also used the same network as hopper. It also had the same learning rate and discount factor. It was trained for 2 million time steps. The repository that we used was rllab ([37](#))

2.4.4 Adversarial Training

For adversarial training, the number of times sampling was done was 200 and the vanilla trained network was re-trained adversarially for same amount of time steps. The adversarial magnitude used was 0.05 for half cheetah and 0.03 for hopper. The sampling frequency 100. We must point out that for the results shown in paper, comparison has been shown between both vanilla and adversarially trained networks that have been trained for exactly same number of timesteps.

2.5 Robust Training Colormap for Cartpole

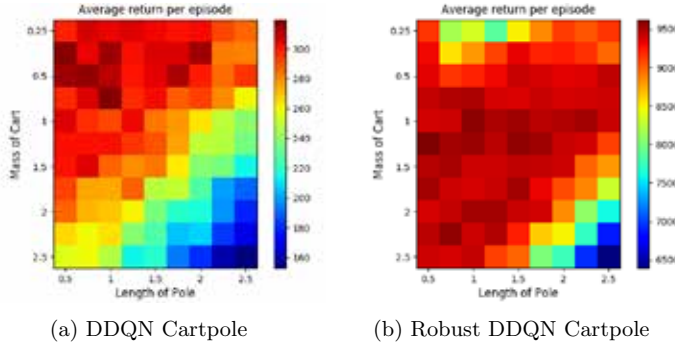


Figure 12: Subfigure (a) shows the average return per episode for cart-pole environment using DDQN algorithm across variation of mass of cart and length of pole. Subfigure(b) shows the same information for adversarially trained DDQN agent. We can observe significant improvement over the return for agent across different parameters. “Zoomed” colormap for DDQN cartpole comparison has been presented.

2.6 Results

In this section, we discuss results related to proposed adversarial attack and adversarially trained robust policy. We report improvement to robustness over two algorithms (DDQN and DDPG). All the experiments have been performed within OpenAi gym environment ([21]) with MuJoCo ([160]) (Fig. [13])

2.6.1 Adversarial Attack

We show that the proposed attack(s) outperform attacks in [74] as shown in Fig. [14]. Here NS refers to naive sampling attack, GB refers to gradient based attack, HFSGM refers to the attack in [74] and SGD refers to the stochastic gradient descent attack. In Fig. [14] the rewards as well as magnitude of adversarial attacks have been normalized to 1 where reward of 1 corresponds to the average reward received without adversarial attack and adversary magnitude of 1 corresponds to maximum possible value of the state. As we can observe from Fig. [14] the adversarial attacks have degraded the performance of deep learning based algorithms. Furthermore, naive sampling based attacks and gradient based attacks perform better FSGM and SGD. Amongst naive sampling and gradient based attack, gradient based attack performs better. Another observation is the relative resilience of RBF based Q learning algorithm as compared to DDQN. This can be explained by the fact that Deep Networks might be providing “jerky” or piecewise linear function approximators

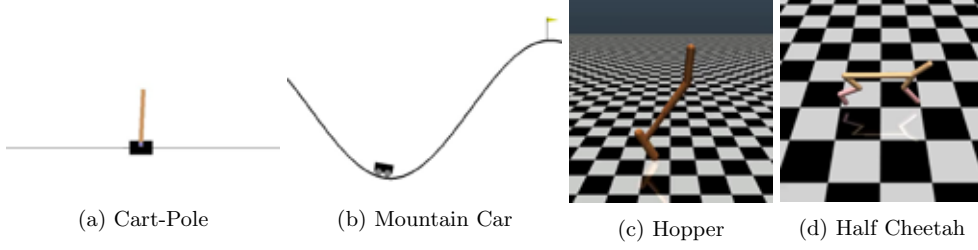


Figure 13: Different environments in OpenAi gym (MuJoCo for simulation of dynamics)

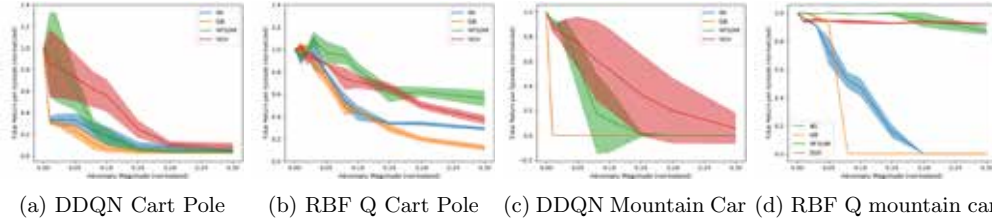


Figure 14: Comparison of different attacks on DDQN and RBF based Q learning. Subfigure (a) and (b) shows adversarial attack on cart-pole environment with DDQN. Subfigures (c) and (d) show adversarial attack on mountain car environment. It can be observed that Gradient Based (GB) attack performs better than Naive Sampling (NS) which in turn outperform Stochastic Gradient Descent (SGD) as well as HFSGM (74). RBF Q learning is relatively more resilient to adversarial attack than DDQN.

as opposed to smoother interpolation of RBF. Thus, a small perturbation in state causes agent to take bad action. The piecewise linear function approximation of DNN can be attributed to the use of “popular” ReLu (or leaky ReLu) activation function as the output is ultimately a composition of piecewise linear functions.

2.6.2 Robust Training

We present results that show significant improvement in robustness because of proposed adversarial training algorithm. For robust adversarial training, we first trained an agent with “vanilla” DRL (DDQN, DDPG) on “default” parameters. The trained agent is then made robust through adversarial training with gradient based adversarial attacks on same parameters. For evaluation, we tested this adversarially trained agent on a wide range of parameters and compared it to “vanilla” DRL. Figures 38 shows improvement in performance owing to robust training. In Fig. 15a, 15b, there is significant improvement in the overall reward for cartpole environment. The agent receives reward of 1 for each time step during which the cartpole is balanced. In 15a, it seems that the agent receives uniform reward because of the scale used. A “zoomed” image of it is provided in Appendix. In mountain car environment (Fig. 15c and 15d), the agent receives a negative reward of -1 for each timestep that it takes to reach the goal. The episode ends either when the car reaches goal or takes up 500 timesteps to reach the target. It is interesting to observe significant gains because of robust training around diagonal where the power coefficient is just enough to push the mountain car. In the lower triangular region, there is no improvement

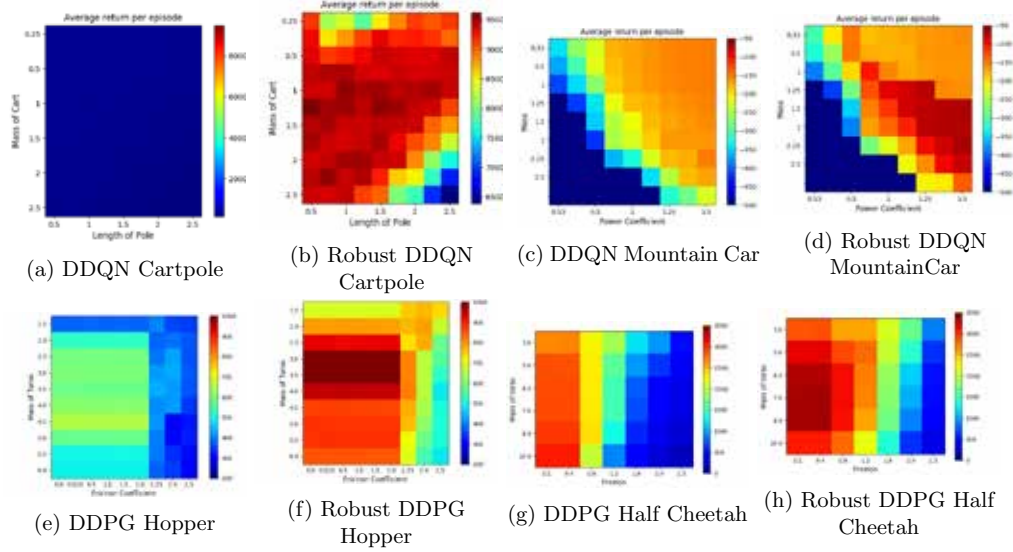


Figure 15: Subfigure (a) shows the average return per episode for DDQN algorithm in Cart-pole environment across variations of mass of cart and length of pole. Subfigure (b) shows the average return per episode of Robust DDQN algorithm. Similar results have been provided in Subfig. (c) and (d) for Mountain Car environment. We can observe high return over wide range of parameters and over "default" parameter owing to robust training. Subfigures (e) and (f) show efficacy of robust training of DDPG algorithm. Subfigures (g) and (h) again demonstrate superior performance of Robust DDPG for Half-Cheetah environment.

because the power coefficient is not enough for the mountain car and is underpowered. In the upper right region, it is overpowered. Hence, it reaches goal easily. For hopper environment (Fig. 15e and 15f), higher return over DDPG can be attributed to adversarial training (robust DDPG). Similar results can also be observed for Half Cheetah environment (Fig. 15g and 15h). These results show average return of 100 episodes for each set of parameter variation over 4 different seeds. The number of training steps is same for both vanilla algorithms and adversarially trained algorithms. Thus, the improvement is not because of more training.

2.7 Conclusion

In this paper, we have proposed adversarial attack for reinforcement learning algorithms. We show that DRL can be fooled easily as compared to reinforcement learning algorithms based on Radial Basis Function (RBF) network. Interestingly, naive adversarial attacks on DRL can degrade it as opposed to robust policy learnt by RBF. We leveraged these attacks to train RL agent that led to robust performance across parameter variations for DDPG and DDQN. Future direction involves providing theoretical relationship between these attacks and robustness of the algorithms (to parameter variation).

3 Cross Domain Transfer Learning using Target Appren-tice

Reinforcement Learning is a machine learning paradigm, where a robotic agent learns the optimal policy for performing a sequential decision making without complete knowledge of the environment. Recent successes in deep reinforcement learning have enabled RL agents to solve complex problems from balancing inverted pendulums [154] to playing Atari games [105]. Despite these recent successes, we do not yet understand how to efficiently transfer the learned policies from one task to another [157]. In particular, while some level of success has been achieved, in transferring RL policies in the same state-space domains, the problem of efficient cross-domain skill transfer is still quite open.

We consider the term “similar” source and target tasks, in the sense that they exploit the same underlying physical principles, but their state spaces can be entirely different. For example in our primary results, we consider the problem of knowledge transfer from balancing Cart-Pole to Bicycle balancing (Figure 16a). While both the system share common dimensionality of state and action spaces but span across a different coordinate frame. The Cart-pole is defined over states of cart and pendulum, $(x, \dot{x}, \theta, \dot{\theta})$ and the action space is lateral force on cart $(-F, 0, F)$. Whereas bicycle dynamics is modelled over handle bar rotation and bicycle roll angle $(\theta, \dot{\theta}, \omega, \dot{\omega})$ with action space is torque applied by rider on the handle bar $(-\tau, 0, \tau)$. While the dynamics and domain of state space of two processes might be completely different, they share a commonality in the underlying physical principles. Both the systems exhibit non-minimum phase dynamics and also the nature of the control policy is same, such that the control action is applied, in the direction of the fall of pendulum or bicycle. This similarity in dynamical behavior of two systems makes learning in a cart-pole domain relevant in bike balancing problem. We also experiment with the problem of transferring the skills learned from the Mountain Car (Figure 16b) [154] to the Inverted pendulum. In the Mountain Car, the agent learns the optimal policy to make an underpowered car climb a mountain. On the other hand, in the pendulum domain, the agent learns to balance a pendulum upright from its initial down position. In both cases, the common physical principle the agent must learn is to exploit the principle of energy exchange. The pendulum must be swung up to the upright position by creating enough angular momentum

through smaller oscillations, and similarly, the car must be made to climb a steeper slope by using energy exchanged by moving up and down the slope of the mountain. In principle, a good RL agent would find it easier to balance a pendulum after it has learned a related task, to make an underpowered car climb a mountain.

Humans are capable of efficiently and quickly generalizing the learned skills between such related tasks. However, RL algorithms capable of performing efficient transfer of policies without learning in the new domain have not yet been reported.

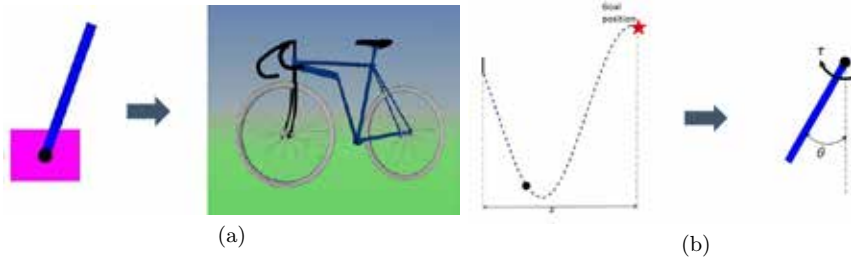


Figure 16: Cross Domain Policy Transfer from: (a) C to Bicycle and (b) Mountain Car to Inverted Pendulum

To address this gap, our main contribution is an algorithm that enables cross-domain transfer in the RL setting. Leveraging notions from apprenticeship learning [1] and adaptive control [28] we propose an algorithm that can directly transfer the learned policy from a source to target task.

3.1 Transfer Learning with Target Apprentice (TA-TL)

This section proposes a novel transfer learning algorithm capable of cross-domain transfer between two related but dissimilar source and target tasks. The presented architecture applies to both continuous and discrete state space systems with discrete action set. Unlike the available state of art TL algorithms, which mainly concentrates on policy initialization in target task RL; we propose to use source policy directly as the optimal policy in the related target model. We achieve this one step transfer through on-line correction of transferred policy with adaptive policy, derived based on model transition error. The presented approach has three distinct phases: Phase I involves finding an optimal policy for source task. For this purpose, we use Fitted Q-Iteration (FQI) RL to solve the source MDP. Since the source task is much simpler and is a smaller problem compared to the target tasks, we assume we can always discover an optimal policy for source task. Phase II; involves discovering a mutual mapping between source and target task state and action space using Unsupervised Manifold Alignment (UMA). Phase III of transfer in the proposed method, is the adaptation of the mapped source optimal policy through policy augmentation in a new target domain.

A possible drawback of the proposed method could be suboptimal transfer if the adaptive policy fails to account for the total model error between projected source and target task. For given a small “ ϵ ” residual model error by adaptive policy results in transferred policy leading to sub-optimal behavior in target space. With the advantage of high sample efficiency of the proposed technique in TL, near-optimal behavior in target task can be acceptable. It is to be noted; we do not engage in exploration in target space for transfer, we only exploit the projected source policy in target space to achieve near-optimal behavior. Nevertheless, with further exploration, we can improve upon the adapted transferred policy and achieve an optimal

solution, but this is left for follow-on work.

Details of three phases of proposed transfer learning technique using target apprentice model are as follows:

3.1.1 Phase I: Learning in the Source task

Fitted Q-Iteration is used to learn optimal policy in the source task. The policy search is not limited to Q-learning and can be extended to any other optimal policy generation methods. A single layer shallow network with radial basis function is used to approximate Q function. For the tasks considered in this paper, shallow networks were found sufficient, but for more complex tasks a deep architecture with multiple layers can be used. This exercise is underway and left for follow-on work.

3.1.2 Phase II: Intertask Mapping

In TL in RL setting, the source and target task have a different representation of state and action spaces. The cross-domain transfer requires an inter-task mapping to facilitate a meaningful transfer. State space $s^{(S)}$ and $s^{(T)}$ belonging to two different manifold, cannot be compared directly, Unsupervised Manifold Alignment (UMA) technique helps to discover alignment between two datasets and provide an one to one and onto inter-task mapping. Using this inter-task mapping, allows us to build one step optimal policy for target task. It is important to note we consider same cardinality and analogous action spaces in analysis and experiments for ease of exposition of the proposed transfer architecture. Problems with distinct, non uniform action spaces will have to use classification methods to find correspondence between action spaces [156, 158]. TL is achieved through augmenting transferred policy by the adaptive policy learned over the target model. The proposed policy transfer and adaption method reuse the source policy in target space resulting in near-optimal behavior. Details of the inter-state mapping are provided in [4, 166] and reference therein.

3.1.3 Phase III: Transfer Learning through policy adaptation

Algorithm 9 Transfer Learning through Policy Adaptation

Require: $\pi^{*(S)}(s)$, χ_S and $\mathcal{P}^{(T)}$ {Inputs: Source Policy, Inter-task mapping and Apprentice Model}
Initialize $s_0^T \sim \rho_0$.
repeat
 $\hat{s}_i^S = \chi_s^+(s_i^T)$ {Project the target task state to source space}
 $a_i^S = \tau^{(S)}\pi^{*(S)}(\hat{s}_i^S)$ {Generate the optimal action using Source policy}
 $s_{i+1}^S = \mathcal{P}^{(S)}(\hat{s}_i^S, a_i^S)$ {Query the source task model at state $\hat{s}_i^S$ and action a_i^S }
 $\hat{s}_{i+1}^T = \chi_s(s_{i+1}^S)$ {Project back the source task propagated state to target space}
 $\pi_{ad}^{(T)} = \mathcal{P}^{(T)}(s_i^T, a_i^T) - \hat{s}_{i+1}^T$ {Compute the adaptive policy}
 $\pi^{*T} = \tau^{(T)}\pi^{*(S)}(\chi_s(s^S)) + \mathcal{K}\pi_{ad}^{(T)}$ {TL policy for target task}
 $a_i^{*T} \sim \pi^{*T}(s_i)$ {Draw action from policy $\pi^{*T}(s_i)$ and propagate the target model}
until $s_i^{(T)} = terminal$

This section presents a transfer algorithm for a pair of tasks in continuous/discrete state and action spaces. Algorithm [15] details TL through policy adaptation using apprentice model.

Empirically we show presented method is sample efficient compared to other methods of transfer; since the sample complexity of learning an optimal policy for initialized target task is reduced to sample complexity of local apprentice model learning, around the optimal trajectory. Algorithm-15 leverages the intertask mapping detailed in subsection 4.1.2 to move back and forth between source and target space for knowledge transfer and adaptive policy learning. The performance of the policy transfer depends on the quality of manifold alignment between source and target tasks. We assume UMA provides a one to one and onto correspondence between source and target state spaces for efficient transfer to happen. Algorithm-15 provides pseudo-code for TL using target apprentice. Steps 1 – 8 provide an architecture for cross-domain policy transfer and step 9 – 12 details policy adaptation through target apprentice learning.

3.1.4 Algorithm: TA-TL

For every initial condition in target task $s_0^T \in \mathcal{S}^{(T)}$; s_0^T are transformed to find the corresponding initial condition on state space in source task.

$$\hat{s}_i^S = \chi_s^+(s_i^T) \quad (36)$$

where χ_s^+ is inverse mapping from target to source space and $\hat{s}_i^S$ represents the image of s_i^T in source space. For the mapped state in source task, a greedy action is selected using learned $Q^{(S)}(\hat{s}_i^S, a_i^S)$ state-action value function.

$$a_i^S = \arg \max_{\pi} (Q^{(S)}(\hat{s}_i^S, a_i^S)) \quad (37)$$

Assumption 1: The cardinality of the discrete action space is same in source and target task,

$$|\mathcal{A}^{(T)}| = |\mathcal{A}^{(S)}| \quad (38)$$

but the limits on action amplitude can be different

$$\|\mathcal{A}^{(S)}\|_2 \leq \tau^{(S)}, \|\mathcal{A}^{(T)}\|_2 \leq \tau^{(T)} \quad (39)$$

Using selected action a_i^S the source model at state $\hat{s}_i^{(S)}$ can be propagated to $\hat{s}_{i+1}^S$.

Assumption 2: Either transition model for the source task is available or that we can sample from a source model simulator. This assumption is not very restrictive since the designer can always select/create related source task for given target task.

$$s_{i+1}^S = \mathcal{P}^{(S)}(\hat{s}_i^S, a_i^S) \quad (40)$$

The propagated state in source task can be mapped back to the target space using inter-task mapping function,

$$\hat{s}_{i+1}^T = \chi_s(s_{i+1}^S) \quad (41)$$

where $\hat{s}_{i+1}^T$ is the image of s_{i+1}^S in target space. From Assumption-1, every selected action in source task has greedy correspondence in target task. Using this equivalence of actions for every selected action in source task an equivalent action in target task is selected as $a_i^T \in \mathcal{A}^{(T)}$. The selected action for target task is augmented with adaptive action $a_{ad}^{(T)} \in \mathcal{A}_{ad}^{(T)}$ derived from adaptive policy,

$$\pi_{ad}^{(T)} = \hat{\mathcal{P}}^{(T)}(s_i^T, a_i^T) - \mathcal{P}_{(S)}^{(T)}(s_i^S, a_i^S) \quad (42)$$

$$a_{ad}^{(T)} \in \pi_{ad}^{(T)}(s^T) \quad (43)$$

where $\hat{\mathcal{P}}^{(T)}(s_i^T, a_i^T) = \mathcal{F}^{(T)}(s_i^T) + \mathcal{B}a_i^T$ is apprentice model, $\mathcal{P}_{(S)}^{(T)}(s_i^S, a_i^S) = \chi_s(\mathcal{P}^{(S)}(s_i^S, a_i^S))$ is projected source model on to target space and $\mathcal{A}_{ad}^{(T)}$ is adaptive action space such that $|\mathcal{A}_{ad}^{(T)}| \geq |\mathcal{A}^{(T)}|$ and $\|\mathcal{A}_{ad}^{(T)}\| \leq \|\pi_{ad}^{(T)}\|_\infty$

Therefore the total transferred policy in target space, in solving a related target task is proposed to be a linear combination of mapped optimal policy and an adaptive policy as follows,

$$\pi^{*(T)}(.) = \pi^{*(S)}(\chi_s(.)) - \pi_{ad}^{(T)}(.) \quad (44)$$

3.1.5 Analysis

Theorem 3.1. *If, for any given small $\epsilon \geq 0$, the difference between true target model and target apprentice model over state-action set $\mathcal{S} \times \mathcal{A}$ is s.t.*

$$\forall (s, a) \in \mathcal{S} \times \mathcal{A}, \quad |\mathcal{P}^{(T)}(s, a) - \hat{\mathcal{P}}^{(T)}(s, a)| \leq \epsilon$$

Assumption 3: $a^S \in \pi^{*(S)}$ is the optimal policy for source task, then the modified policy $\pi^{*(T)}(.) = \pi^{*(S)}(\chi_s(.)) - \pi_{ad}^{(T)}(.)$ is ϵ -optimal in the target task

Proof. Consider a MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{H}, D, \mathcal{R})$, and suppose we have approximation $\hat{\mathcal{P}}$ of transition model, thus the approximate apprentice model can be written as the tuple $\hat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}, \mathcal{H}, D, \mathcal{R})$. We analyze the admissibility of the augmented policy for the target space. For this analysis we will consider deterministic, linearly parametrized discrete transition model for both source and target model as follows,

$$\mathcal{P}^{(S)}(s, a) : s_{i+1}^S = \mathcal{F}^{*(S)}(s_i^S) + \mathcal{B}a_i^S \quad (45)$$

$$\mathcal{P}^{(T)}(s, a) : s_{i+1}^T = \mathcal{F}^{*(T)}(s_i^T) + \mathcal{B}a_i^T \quad (46)$$

where $s^S \in \mathcal{S}^{(S)}$, $s^T \in \mathcal{S}^{(T)}$ and $a^S \in \mathcal{A}^{(S)}$, $a^T \in \mathcal{A}^{(T)}$. Deterministic state transition model is represented as $\mathcal{F}^{*(S)}(s_i^S) = A^{*(S)}\phi(s_i^S)$ and $\mathcal{F}^{*(T)}(s_i^T) = A^{*(T)}\phi(s_i^T)$, where $A^{*(S)}$, $A^{*(T)}$ are state transition matrices for source and target tasks and $\phi(.)$ is nonlinear basis vector and $A\phi(.)$ is state dependent coefficient form of the nonlinear transition model $\mathcal{F}^{(*)}$.

Let $\mathcal{P}_{(S)}^{(T)}(s, a) = \chi_s(\mathcal{P}^{(S)}(s, a))$ be projection of source task onto target space and the target apprentice model can be written as of the form,

$$\hat{\mathcal{P}}^{(T)}(s, a) : s_{i+1}^T = \hat{\mathcal{F}}^{(T)}(s_t) + \mathcal{B}a_i^{(T)} \quad (47)$$

where $\hat{\mathcal{F}}^{(T)}(s_t) = \hat{A}^{(T)}\phi(s_i^{(T)})$ and $\hat{A}^{(T)}$ be approximate transition matrix for target

For ease of exposition and without loss of generality we consider a linear in the state representation with $\phi(s) = s$ for further proof of optimality. Selecting the action from projected optimal policy $a_{(S),i}^{(T)} \in \pi^{*(S)}(\chi_s(s^T))$, augmenting with model adaption term and applying it to target model following holds,

$$s_{i+1}^{(T)} = A^{*(T)}s_i^T + \mathcal{B}a_{(S),i}^{(T)} - \mathcal{B}a_{ad,i}^{(T)} \quad (48)$$

From definition of model adaptive policy (101) and apprentice model (105), above expression can be simplified as

$$\begin{aligned} s_{i+1}^{(T)} &= A^{*(T)}s_i^T + \mathcal{B}a_{(S),i}^{(T)} - \mathcal{B}(\hat{\mathcal{P}}^{(T)}(s_i^T, a_i^T) - \mathcal{P}_{(S)}^{(T)}(s_i^S, a_i^S)) \\ s_{i+1}^{(T)} &= A^{*(T)}s_i^T + \mathcal{B}a_{(S),i}^{(T)} - \mathcal{B}(A^{(T)}s_i^T + \mathcal{B}a_{(S),i}^{(T)} - A_{(S)}^{(T)}s_i^T - \mathcal{B}a_{(S),i}^{(T)}) \end{aligned} \quad (49)$$

Simplifying the above expression,

$$s_{i+1}^T = A_{(S)}^{(T)} s_i^T + \mathcal{B} a_{(S),i}^{(T)} + \mathcal{B} \epsilon \quad (50)$$

where $\epsilon = (A^{*(T)} s_i^T - A^{(T)} \hat{s}_i^T)$

The above expression (108), can be rewritten in terms of source transition model using the inter-task mapping function χ_S as

$$s_{i+1}^T = \chi_S \left(A^{*(S)} s_i^S + \mathcal{B} a_i^S + \mathcal{B} \epsilon \right) \quad (51)$$

Using **Assumption 3**, expression (109) demonstrates using the modified optimal policy (102) in target task is equivalent to projecting the optimal trajectories in source task on to the target task. Assuming UMA maps precisely the source to target task space, we prove the policy (102) leads to ϵ -optimal solution in target model. $\square$

3.2 Experiments & Results

We present results from three experiments to evaluate the proposed transfer learning framework. The first two experiments consider transfer learning in domains with the same state-space but with different transition models and action spaces. The first problem is in discrete state and action space, while the second problem involves continuous state space and non-stationary transition model in the target task. The third experiment focuses on cross-domain transfer where the policy from the mountain car problem is transferred to the inverted pendulum problem. To demonstrate the quality of the transfer, we compare the proposed Target Apprentice TL (TA-TL) against existing state of the art method Transfer Learning via Unsupervised Manifold Alignment (UMA-TL) [4] and stand alone RL (Fitted Q-Learning) on target domain with no initialization from source task learning.

3.3 Same-Domain Transfer

We learn the optimal policy in the source task using FQL. In each problem, distinction in the environment/system parameters makes the source and target tasks different. The target and source domains have the same state-space but different transition models and action spaces. We also do not need target reward model be similar to source task, as the proposed algorithm directly adapts the policy from the source task and does not engage in RL in the target domain.

3.3.1 Grid World to Windy Grid World

The source task in this experiment is Non-Windy (NW) grid world. The state variables describing the system are grid positions. The RL objective is to navigate an agent through obstacles from start to goal position optimally. The admissible actions are up (+1), down (-1), right (+1) and left (-1). The reward function is +10 for reaching goal position, -1 for hitting obstacles and 0 everywhere else. The target domain is same as the source but with added wind which affects the transition model in parts of the state-space. The optimal policy in source task (non-windy grid world) $\pi^{*(S)}$ is learned using Q-Iteration. We do not need any inter-task mapping as the source, and target state spaces are identical. We start with 100 randomly sampled starting position and execute policy $\pi^{*(S)}$ in the target domain and collect samples for apprentice model learning. Empirically, we show the proposed method (TA-TL) provides a sample efficient TL algorithm compared to other transfer techniques. Figure 41a and 41b shows the results of same domain transfer in the grid world, demonstrating TA-TL achieving successful transfer in navigating

through the grid with obstacles and wind bias. Figure 41a and 41b shows the quality of transfer through faster convergence to average maximum reward with lesser training samples compared to UMA-TL and RL methods. The presented algorithm is able to attain maximum average reward in reaching goal position in $\sim 2 \times 10^4$ steps. UMA-TL and RL algorithm achieve similar performance in $\sim 1.2 \times 10^5$ and $\sim 1.7 \times 10^5$ steps respectively, nearly one order higher compared to proposed TA-TL.

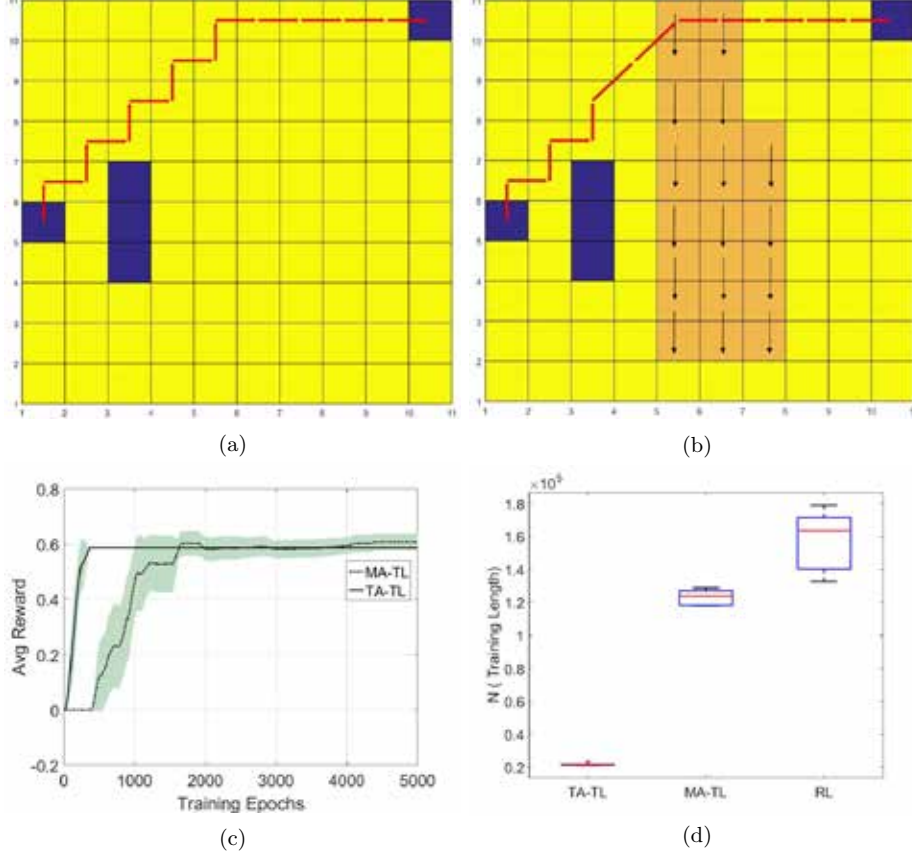


Figure 17: Non windy to Windy grid World Transfer:(a) Average Rewards (b) Training length, Comparing quality of transfer for TA-TL and UMA-TL through convergence rate of Average Reward and Training Length

3.3.2 Inverted Pendulum (IP) to time-varying IP

We demonstrate our approach for a continuous state domain, Inverted Pendulum (IP) swing-up and balance. The source task is the conventional IP domain [154]. The target task is a non-stationary inverted pendulum where the length and mass of the pendulum are continuously time varying with function $L_i = L_0 + 0.5\cos(\frac{\pi i}{50})$ and $M_i = M_0 + 0.5\cos(\frac{\pi i}{50})$, where $L_0 = 1$, $M_0 = 1$ and $i = 1 \dots N$. The state variables describing the system are angle and angular velocity $\{\theta, \dot{\theta}\}$

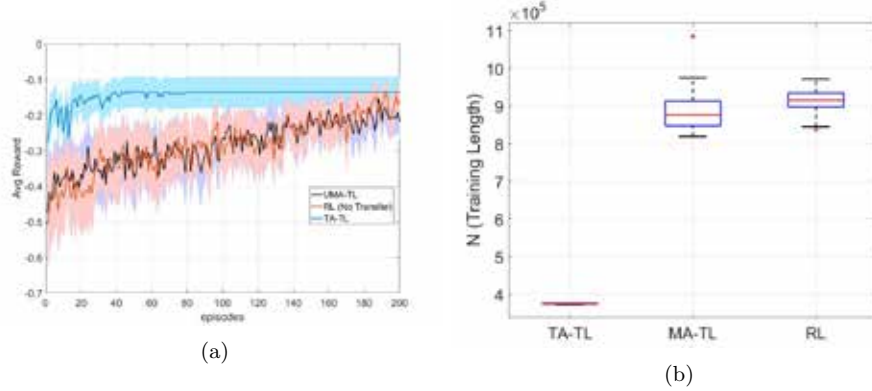


Figure 18: IP to Non-stationary IP Transfer: (a) Average Rewards and (b) Training length

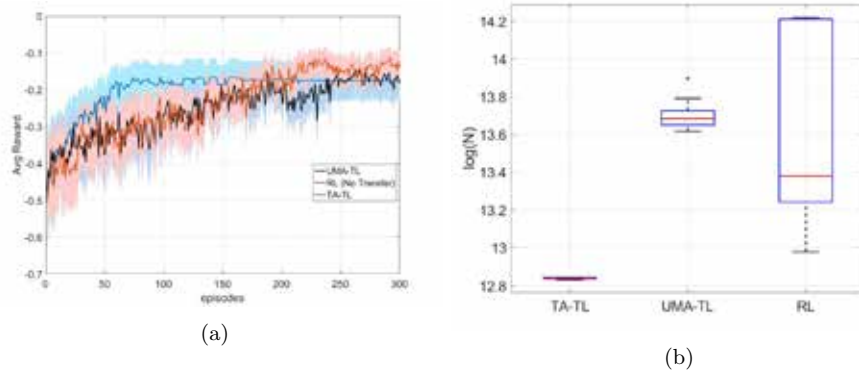


Figure 19: Mountain Car to Inverted Pendulum Transfer: (a) Average Rewards and (b) Training length

where $\theta, \dot{\theta} \in [-\pi, \pi]$. The RL objective is to swing-up and balance the pendulum upright such that $\theta = 0, \dot{\theta} = 0$. The reward function is selected as $r(\theta, \dot{\theta}) = -10|\theta|^2 - 5|\dot{\theta}|^2$, which yields maximum value at upright position and minimum at the down-most position. The action space is: full throttle right ($+\tau$), full throttle left ($-\tau$) and zero throttle. Note that the domain is tricky, since full throttle action is assumed to not generate enough torque to be able to lift the pendulum to the upright position, hence, the agent must learn to swing the pendulum so that it generates oscillations and leverages angular momentum to go to the upright position. The target task differs from the source task in the transition model.

The source task use FQI learning with single layer Radial Basis Functions (RBF) network. The Q function is modeled as linear combination of weights and basis as $Q = w'\xi(s)$. We use RBF bases " $\xi(s)$ " for value function approximation with bandwidth $\sigma = 1.2I$ and 20 centers spanning space $x, \dot{x} \in [-\pi, \pi]$ with network learning rate of $\Gamma = 5 \times 10^{-3}$ for FQI iterations.

Figure 42a and 42b shows the quality of transfer through faster convergence to average maximum reward with lesser training samples for proposed TA-TL method compared to UMA-TL and RL methods.

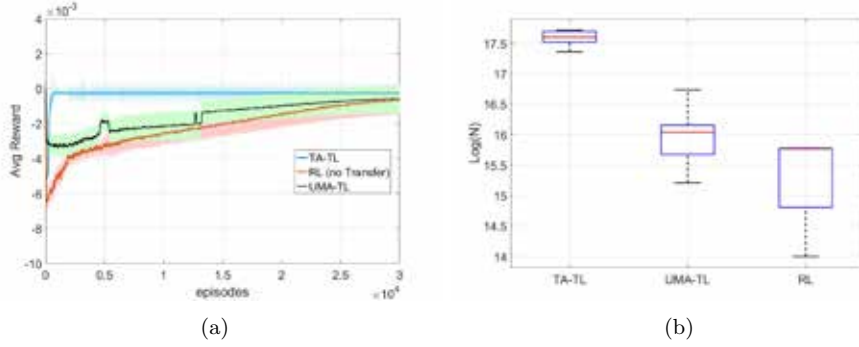


Figure 20: Transfer Cart-Pole to Bike Balancing task: (a) Average Rewards and (b) Total simulation time (seconds) the agent can balance the bike

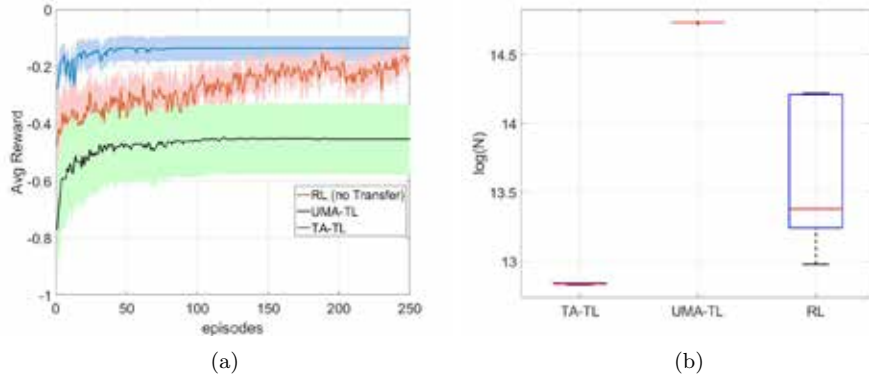


Figure 21: Negative Transfer Inverted Pendulum: (c) Average Reward and (d) Training length

3.4 Cross-Domain Transfer

Next we consider an even more challenging transfer setting: cross-domain transfer. The problem setup is similar to same domain transfer with important distinction being state spaces are different for the source and target tasks.

3.5 Cart-Pole to Bicycle

Our last problem is the task where an agent learns to ride the bicycle. We consider the problem of learning to balance; we do not concentrate on the navigation problem to some goal position. Since RL for navigation is more of trajectory optimization problem, i.e., the agent learn to focus on maneuvering towards the target once it has learned to balance the bicycle upright. And also balancing is a more interesting problem when the bicycle is below critical velocity. Usually critically velocity V_c , above which bicycle is self stabilizing is approximately $V_c = 4m/s$ to $5m/s$. We are trying to learn to balance a unstable bicycle with forward velocity $V < V_c$ i.e. $V = 2.778m/s$. We also simulate the imperfect balance by inducing random noise in the CG displacement of rider up to $2cm$ from zero position.

At every time step agent receives information about the state of the bicycle, the angle and angular velocity of the bike handle bar and the bike from vertical $(\theta, \dot{\theta}, \omega, \dot{\omega})$ respectively. The agent can choose the action of applying torque to handle bar, $T \in [-2Nm, 0, 2Nm]$. The details of bicycle dynamics are beyond the scope of this paper, interested readers are referred to [130] and references there in.

We use the Cart-Pole as source task for learning to balance bicycle. The bicycle is not so different from the cart pole: in both the cases, the objective is to keep the unstable system upright. The objective of balance is achieved in both the systems by moving in the direction of fall. However, the control in the cart pole affects more directly the angle of the pole, i.e., move the cart such that it is always under the pole. In the bicycle, the control is to move the handle bar in the direction of fall. However balancing the bike is not so simple, to turn the bike under itself, one must first steer in the other direction, this is called counter steering. We observe that both cart pole and bicycle has this commonality in dynamical behavior, as both the system have a non-minimum phase that is the presence of unstable zero. They tend to move initially in the direction opposite to application control. This similarity qualifies the cart-pole system as an appropriate choice of source model for bicycle balance task.

Cart pole is characterized by state vector $[x, \dot{x}, \theta, \dot{\theta}]$ i.e., position, the velocity of cart and angle, angular velocity of the pendulum. The action space is the force applied to cart $F \in [-20N, 0, 20N]$. Like previous results, we use UMA to discover the correspondence between state space of bicycle and cart pole model. We use FQI to solve for optimal policy in the source, cart pole model. A linear network with cart pole states $\xi(s) = [x, \dot{x}, \theta, \dot{\theta}]$ is used as basis vector to approximate the action value function, with network learning rate $\Gamma = 1 \times 10^{-4}$.

Figure 44a shows the average reward accumulated by TA-TL, UMA-TL and RL (no transfer) in learning to balance the bicycle. In typical learning process, TA-TL out performs other transfer method and converges to maximum average reward in 1700 episodes. Each episode are trail until the bicycle balances without falling, episode ends if bicycle falls or 1000s is reached. Figure 44b shows total time bicycle was balanced upright by each method. Balancing times for TA-TL methods are highest and achieve ≈ 40 times more balancing time compared to UMA-TL method.

3.5.1 Mountain Car (MC) to Inverted Pendulum (IP)

We have tested the cross-domain transfer between mountain car to an inverted pendulum. The source and target task are characterized by different state and action space. The source task MC is a benchmark RL problem of driving an under-powered car up a hill. The dynamics of MC are described by two continues state variables $(x, \dot{x})$ where $x \in [-1.2, 0.6]$ and $\dot{x} \in [-0.07, 0.07]$. The input action takes three distinct values (+1) full throttle right, (-1) full throttle left and (0) no throttle. The reward function is proportional to negative of the square distance of the car from goal position. The target task is conventional IP as described in the previous experiment.

Cross-domain transfer requires a correspondence between inter-task space manifold for mapping the learned policy and source transition model from source space to target space and back. We utilize UMA to obtain this mapping as described in Section 4.1.2. We do not report the training time to learn the intertask mapping since it is common to both TA-TL and UMA-TL methods. We used a random policy to generate samples for manifold alignment and projected source policy in target space for target apprentice learning. The source task uses FQI learning with single layer RBF network for optimal policy generation. The source Q-function is modeled using function approximation as $Q = w'\xi(s)$ using RBF as bases " $\xi(s)$ " with bandwidth $\sigma = \text{diag}[0.3, 0.1]$ and 20 centers spanning space $x \in [-1.2, 0.6]$ and $\dot{x} \in [-0.07, 0.07]$ with network learning rate of $\Gamma = 0.15 \times 10^{-3}$. For all above results the training length involved with TA-TL method in Figure 43b, 42b and 44b is sample lengths for target apprentice learning. We

compare TA-TL with UMA-TL and generic-RL on target task. We examine the efficiency and effectiveness of transfer methods based on sample efficiency in learning the target task and speed of convergence to maximum average reward. Similar to same domain transfer Figure 43a and 43b shows the quality of transfer for TA-TL through faster convergence to average maximum reward with lesser training samples compared to UMA-TL and RL methods.

3.6 Negative transfer

Finally with our last set of results we demonstrate that the proposed transfer is robust to negative transfers. where either noisy features, imperfect or approximate set of source agents space result in poor or decreased performance in target domain.

Given a target model, the effectiveness of transfer depends on the relevance of the source model to the target task. If the relationship is strong and the transfer method can take advantage of it, the performance in the target task can significantly improve through transfer. However if the source is not sufficiently related or the features of source task does not correspond to the target, the transfer may not improve or even decrease the performance in target task leading to negative transfer.

We show that the UMA-TL suffers from negative transfer, where as proposed TA-TL performs superior compared to UMA-TL or no transfer RL. We demonstrate this through a inverted pendulum upright balance problem. We select inverted pendulum model as source and target systems. The target model is different from source model in sign of the control action. With exactly same dynamics in both source and target model but with the control sign flipped in target, we observed that initializing the action value function from source in target task learning lead to negative transfer. Whereas for the TA-TL method, since we learn the apprentice to target on line, the policy modification term is flipped accordingly and exactly same policy transfer performance is achieved as for same domain transfer between IP to non-stationary IP.

Figure 45a and 45b shows the quality of transfer through faster convergence to average maximum reward with lesser training samples for proposed TA-TL method compared to UMA-TL and RL methods (QL : Q-Learning) and MBRL (Model Based RL). It is to be observed that UMA-TL method converges to much lower average reward and gets stuck in a local optimum point and also the samples observed by UMA-TL in learning the task is also much higher compared to no transfer (RL) and proposed TA-TL methods.

4 Results Connecting Evolving Gaussian Processes to Koopman Operator Theory & Dynamic Mode Decomposition

The Koopman operator is a linear but infinite-dimensional operator that governs the evolution of scalar observables of a state space of an autonomous dynamical system. This operator is a powerful tool for approximating and analyzing nonlinear dynamics. With the introduction of the Dynamic Mode Decomposition (DMD) algorithm, the theory has begun to be applied to the growing need to analyze and engineer high-dimensional, ill-described, and complex systems using “big data” measurements. In previous work, we showed the power and effectiveness of our Evolving Gaussian Processes (E-GP) method, which is a differentially-constrained machine-learning algorithm designed to infer the latent state of a spatiotemporally evolving distributed dynamic system. It has been demonstrated that E-GP can capture the major dynamical features of a nonlinear fluid flow by layering a linear dynamic system over the temporal evolution of the weights of stationary kernels. In the past year, we have established both theoretically and

empirically a relationship between E-GP and the Koopman operator, providing a new way of computing the Koopman modes, eigenvalues, and eigenfunctions.

4.1 Koopman Operator Theory

Koopman operator theory is formulated in the context of studying the "dynamics of observables", that is, the dynamics of the system are analyzed by studying the evolution of functions on the state space, rather than the state space trajectories themselves.

Our formulation will follow that found in the exemplary [23]. The state space shall be denoted as M and the dynamics on M defined by the transition operator

$T : M \rightarrow M$. An *scalar observable* is defined as a function $\phi : M \rightarrow \mathbb{C}$, where ϕ belongs to a function space $\mathcal{F}$ which we will specify later. A real-life interpretation of an observable function would be a sensor measuring some quantity related to the system. Instead of tracking the state trajectory $s, T(s), T^2(s), \dots$, we track the trace $\phi(s), \phi(T(s)), \phi(T^2(s)), \dots$. This corresponds with the familiar state and output equation format used in the control theory community:

$$s_{n+1} = T(s), \quad y_n = \phi(s_n) \quad (52)$$

The Koopman operator, also known as the composition operator, is defined on the space of observables ($U_T : \mathcal{F} \rightarrow \mathcal{F}$) as follows:

$$(U\phi)(s) = (\phi \circ T)(s) = \phi(T(s)) \quad (53)$$

for all $s \in M$. When $\mathcal{F}$ is a vector space, then U is linear. When M is a finite set, then U is a finite-dimensional operator and can be represented by a matrix; however, in the usual case that the state space is not finite (either finite- or infinite-dimensional), then U is infinite-dimensional. Often, we would like to extend the space of scalar observables to a vector space. A natural example of this is when we have N sensors $\phi_1, \dots, \phi_N$ collecting measurements which we might store as a vector to consider writing the measurements of N particular sensors as a vector

4.2 Dynamic Mode Decomposition

Rowley et al. provided one of the first analyses of Dynamic Mode Decomposition (DMD) which established the connection between DMD and the Koopman operator, and helped to explain the output of DMD when applied to nonlinear systems. Since then, a number of modifications have been developed that either strengthen this connection further or enhance the robustness and applicability of the approach. There are two basic approaches to DMD: Arnoldi-like methods and Singular Value Decomposition (SVD) methods.

The data takes the form of a snapshot sequence $V_1^N = \{v_1, v_2, \dots, v_N\}$, $V_1^N = \{v_1, v_2, \dots, v_N$ where $v_i \in \mathbb{R}^M$ $v_i \in \mathbb{R}^M$ is the i -th snapshot of the flow field, and $V_1^N \in \mathbb{R}^{M \times N}$ $V_1^N \in \mathbb{R}^{M \times N}$ is a data matrix whose columns are the individual snapshots. The subscript and superscript denote the index of the snapshot in the first and last columns respectively. These snapshots are assumed to be related via a linear mapping

$$v_{i+1} = Av_i, v_{i+1} = Av_i$$

that remains approximately the same over the duration of the sampling period. Written in matrix form, this implies that

$$V_2^N = AV_1^{N-1} + re_{N-1}^T, V_2^N = AV_1^{N-1} + re_{N-1}^T,$$

where r is the vector of residuals that accounts for behaviors that cannot be described completely by A , $e_{N-1} = \{0, 0, \dots, 1\} \in \mathbb{R}^{N-1}$, $e_{N-1} = \{0, 0, \dots, 1\} \in \mathbb{R}^{N-1}$, $V_1^{N-1} = \{v_1, v_2, \dots, v_{N-1}\}$, $V_1^{N-1} = \{v_1, v_2, \dots, v_{N-1}\}$, and $V_2^N = \{v_2, v_3, \dots, v_N\}$, $V_2^N = \{v_2, v_3, \dots, v_N\}$. Regardless of the approach, the output of DMD is the eigenvalues and eigenvectors of A , which are referred to as the DMD eigenvalues and DMD modes respectively.

4.3 Theoretical Connection between Koopman & E-GP

Having formulated the problem above, we are prepared to examine why and how the eigenvalues and eigenvectors of the matrix $\hat{A}$ can be related to the Koopman operator. In our system, the states are the mean functions f_τ which exist in the state space $\mathcal{H}$, which is an RKHS generated by the positive definite Mercer kernel k . The system operator $\mathcal{A}$ takes the place of the state transformation, and the measurement operator $\mathcal{K}$ is in essence an observable of the state. From this perspective, the finite-dimensional approximation of the Hilbert space is really a dimensionality-reducing convolution operator. Given that in the limit of centers used, $\hat{\mathcal{H}}$ becomes dense in $\mathcal{H}$, we can identify every function with the set of weight vectors in $\mathbb{R}^M$. Since we still want the corresponding operators in approximate space, $\hat{A}$ and K , to be linear, they become matrices. We can now write the Koopman operator U as a matrix as well, defined by

$$UK = K\hat{A}$$

Supposing v is an eigenvector of $\hat{A}$ with eigenvalue λ , $\hat{A}v = \lambda v$, we find that

$$U(Kv) = K\hat{A}v = \lambda Kv$$

which implies that Kv is a Koopman mode with eigenvalue λ .

4.4 Experimental Connection between DMD & E-GP

We demonstrate that our method for obtaining Koopman modes from an E-GP model and the modes derived using Dynamic Mode Decomposition produce the same results by applying both methods to our data sets of fluid flowing past a cylinder at different Reynolds numbers. This is a canonical fluids problem that is well-studied in the fluid dynamics literature, both experimentally and numerically [140, 118, 128]. In order to derive the eigenvalues and Koopman modes of the system, one must simply take the eigenvalues and eigenvectors of the $\hat{A}$ matrix, and transform the latter through the observation matrix K . We have included images of the Koopman modes from both E-GP and DMD for the fluid flowing at Re=1000 below.

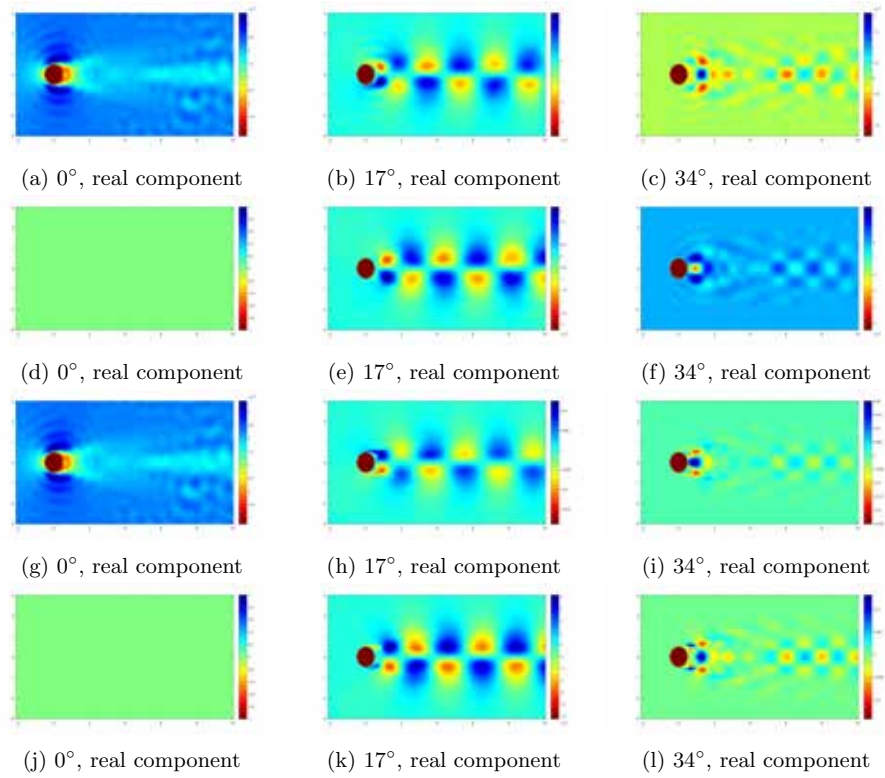


Figure 22: Primary Koopman Modes, $Re=1000$. (a-f) E-GP, (g-l) DMD

5 Path Planning for Mobile Inference of Spatiotemporally Evolving Systems Using Evolving Gaussian Processes

We developed a new method for planning trajectories of moving agents with the goal of deriving the state of a spatiotemporally evolving dynamical systems. This method is based on the Evolving Gaussian Processes (E-GP) model, which uses differential constraints for inference over such large-scale distributed systems. In this paper, spectral analysis of the linear dynamical layer of an E-GP model is used to identify invariant subspaces within a system and choose paths within the domain which are most likely to give the moving agent the information necessary to converge to an accurate state estimate. First, we present a new method of clustering eigenmodes of the system together, and identifying them with separate invariant subspaces. This is followed by a discussion of the importance of several factors in planning optimal trajectories for a moving Bayesian state estimator, particularly the domain-magnitudes and mode-frequencies of different invariant subspaces. Finally, we present a new algorithm for choosing the best path for a given E-GP model. We demonstrate the power of these new methods using both synthetic and real-world data sets.

5.1 Spectral Analysis of Evolving Gaussian Process Model and Resulting Algorithms

In previous work, we showed that the cyclic index of $\hat{A}$ is a nonconservative lower bound on the number of distinct sampling locations required for the observability of the system, and is equal to the number of *invariant subspaces* $\mathcal{H}_i \subset \mathcal{H}$ into which $\mathcal{A}$ can be uniquely decomposed. It is formally defined as the largest geometric multiplicity of an eigenvalue in $\hat{A}$. One way of viewing the invariant subspaces concept is say that information contained in an invariant subspace never leaves that invariant subspace. We hypothesize that the kernel centers associated with the invariant subspaces of a spatiotemporally evolving system are generally associated with spatial regions in the domain (and not just homogeneously spread all over and or mixed with the other invariant subspaces). This hypothesis makes sense both physically and mathematically. In physics, the *principle of locality* states that an object is only directly influenced by its immediate surroundings [10]. If this is the case, and the E-GP model accurately captures the physics of the system, then information (measurable phenomena) may only travel continuously from one point in the domain to another. Mathematically, since a value at any one point in the domain is influenced by the weights of multiple nearby centers, we would expect nearby centers to be connected dynamically, unless separated by “plains” where the values are indistinguishable from noise.

When a square matrix is nicely formed, the Jordan form of a $n \times n$ matrix $\hat{A}$ is block diagonal, and therefore gives a decomposition of the n dimensional Euclidean space into invariant subspaces of $\hat{A}$. The cyclic index, which can be found by counting the geometric multiplicities of eigenvalues in $\hat{A}$, gives the number of invariant subspaces. In reality, data-driven approximations of $\hat{A}$ rarely give such easily interpretable properties. Hence, the need for an algorithm which divides invariant subspaces.

Each block in the Jordan normal form has a set of corresponding eigenvectors and an eigenvalue with geometric multiplicity. When we transform the former back into the domain space, we obtain complex-valued functions which are the Koopman modes of the system. These provide an image of what kind of structures we see in the dynamics. The eigenvalues describe the frequency with which these structures oscillate between their real and imaginary forms, as well as the exponential growth or decay of their magnitudes. In this paper, we demonstrate our method

on systems with eigenvalues primarily on the unit circle (neither growing nor decaying).

5.2 k -Invariant Subspaces Clustering

The intuition behind our k -invariant subspaces clustering algorithm is to replace the Euclidean distance with a different metric of “nearness”, namely one corresponding with the dynamic connections in the space. The $\hat{A}$ matrix provides easy access to these: its rows $\hat{A}_{i*}$ indicate which centers inform the i^{th} value of $w_{t_{index}+1}$, and its columns $\hat{A}_{*j}$ indicate what centers will be informed by the j^{th} value of w_τ . However, in order to cluster without bias, we need to control for the differing frequencies of the eigenmodes. If we do the eigen-decomposition of the matrix as $\hat{A} = UDU^{-1}$, then let $\bar{D}$ be zeroed out except for its eigenvalues on the unit circle, and let $\bar{U}$ be truncated to only the corresponding eigenvectors. Then let $\bar{A} = U\bar{D}U^\dagger$, where $U^\dagger$ is the pseudo-inverse. Much like the pairwise squared deviations of points formulation of the k -means clustering problem, we can now write our problem as

$$\arg \max_S \sum_{i=1}^k \frac{1}{2|S_i|} \sum_{\substack{x_i \neq x_j \\ x_j, x_i \in S_i}} \bar{A}_{ij}^2$$

This problem can be solved with the following algorithm: Note the differences between this and

Algorithm 10 k -Invariant Subspaces Algorithm

- 1: **while** clusters have changed **do**
- 2: **for** each center i **do**
- 3: find cluster k which maximizes the score

$$\frac{1}{|S_k| + 1} (\|\bar{A}_{i, S_k \setminus \{i\}}\|^2 + \|\bar{A}_{S_k \setminus \{i\}, i}\|^2)$$

- 4: reassign center i to cluster with highest score
 - 5: **end for**
 - 6: **end while**
 - 7: **return** clusters
-

the k -means clustering algorithm: first, we are maximizing since the terms represent influence rather than nearness. Secondly, we use $|S_k| + 1$ in the denominator to avoid division by zero. Thirdly, we exclude centers influence on themselves from the score by taking $S_k \setminus \{i\}$.

5.3 Scoring Paths

We have already described some of the important properties of the dynamics of the system. Now we discuss their relevance for path planning. We postulate that the value of taking a measurement at a point in the domain with respect to a particular eigenvector is proportional to the following factors:

1. The spatial extent of the eigenvector. This can be taken to be equivalent to the area under the curve of the magnitude of the complex function corresponding to that eigenvector (normalized by the function peak)

$$SE_i = \frac{1}{\sup_x v_i \hat{\Phi}(x)} \int_X |v_i \hat{\Phi}(x)| dx$$

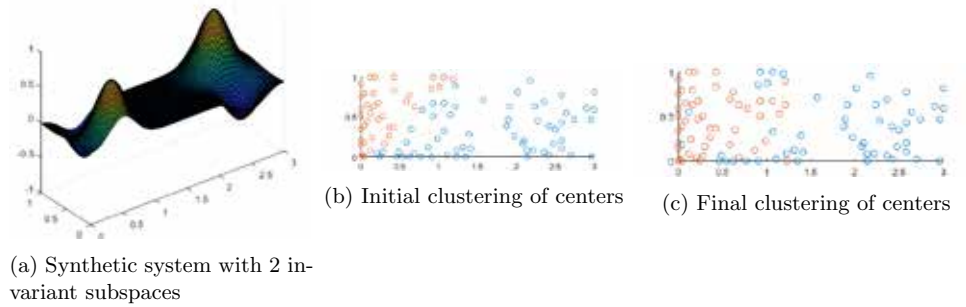


Figure 23: Synthetic System Invariant Subspaces

2. The expected size of the measurement itself, which is roughly equal to $v_i \hat{\Phi}(x)$.
3. The ratio of the frequency with which that Koopman mode is visited to the frequency of its eigenvalue. This factor is included due to the aliasing effect
4. A discount factor equal to a decaying exponential of the number of times that eigenmode has been visited

This scoring method allows one to decide the likeliest candidate amongst a large family of paths

5.4 Generating High-Scoring Random Paths

We developed the following algorithm for generating high-scoring paths:

1. Cluster the centers into invariant subspaces according to the k -invariant subspaces algorithm, with the best reasonable guess(es) for k
2. In each cluster, select a waypoint where the sum of all Koopman modes is maximized
3. Generate a path by selecting waypoints randomly, where the selection is randomly weighted. The weights are equal to the spatial extent of the clusters multiplied by an exponential decay factor for the number of times each has been visited

5.5 Experimental Results

5.5.1 Simple Synthetic Examples

To begin our investigation into this problem, we constructed a number of simple synthetic spatiotemporally evolving systems. One of these is displayed here [23](#). This is a system with two invariant subspaces, in each of which, two hills oscillate up and down in sync at a different frequency than the other invariant subspace. This is ideal for testing our methods. In the example shown, we used 100 centers with an RBF kernel with a bandwidth of 0.2, however in testing we also went as low as 20 centers.

The k -means algorithm divides the system nicely through the middle, as expected.

We found that few purely random paths would converge quickly for the synthetic data set, due to the large plain of zero values in-between the two invariant subspaces. Our method for

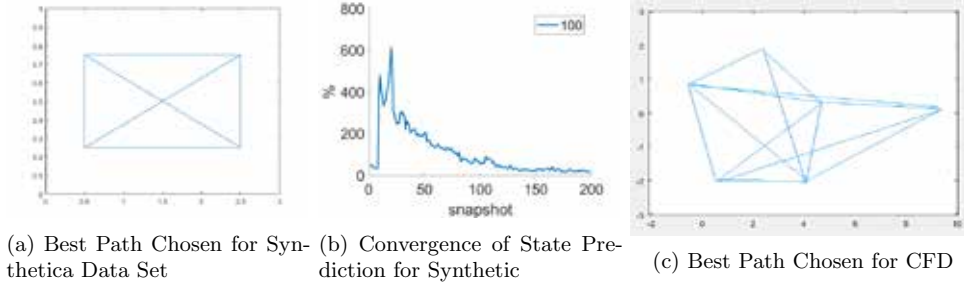


Figure 24: More Results

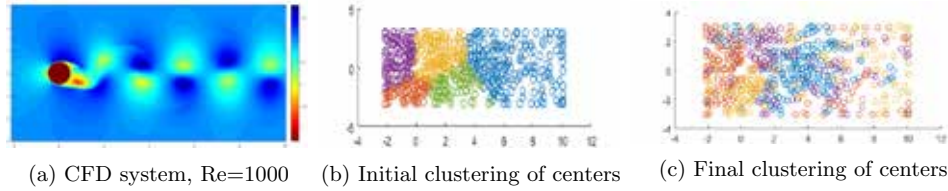


Figure 25: CFD System Invariant Subspaces

generating and selecting paths with waypoints resulted in paths that traveled between the four humps and quickly converged.

5.5.2 Fluid Flow Past a Cylinder

We used CFD methods to generate the states for a canonical fluid mechanics problem: flow past a cylinder at various Reynolds numbers, namely $Re = 100, 300, 600, 800$ and 1000 . This deterministic, high-dimensional spatiotemporal dynamical system is well-studied in the fluid dynamics literature, both experimentally and numerically [140, 118, 128]. In our CFD simulation, we used a 4th-order polynomial expansion with spectral element method on the incompressible Navier-Stokes equation to generate the cylinder flow data. The spatial domain is $[-2, 10] \times [-3, 3]$, excluding the diameter-1 cylinder at the origin. Neumann boundary conditions are applied to the far-field of the cylinder in the y -direction and the outlet of the flow field; and a Dirichlet boundary condition is applied to the inlet. The flow is unstable, periodic, and clearly nonlinear.

For training the E-GP we used 600 centers with an RBF kernel with a bandwidth of 0.4. This output a 600×600 matrix $\hat{A}$ which accurately captures the dynamics of the whole system. Note that *the size of the model has been reduced by almost two orders of magnitude* from the original CFD data.

Since the CFD set has dynamics almost everywhere, most paths are able to converge to the state. However, it is notable that a pre-designed lawnmower path failed to do as well as most random paths. Once again, the random paths generated and chosen by our methods did the best.

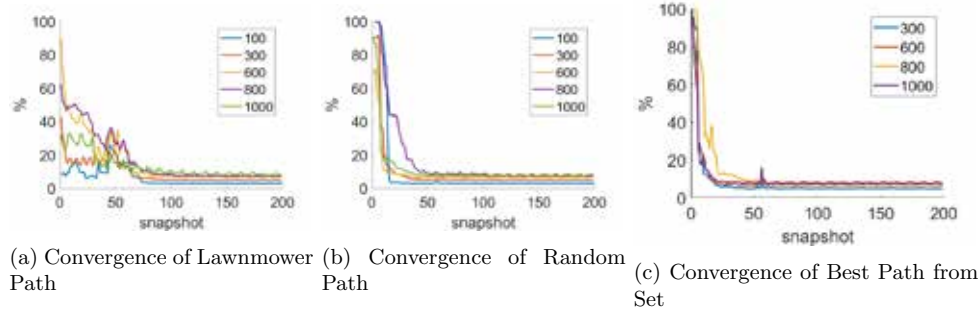


Figure 26: Convergence Results for CFD

6 A Dynamic Risk Form of Entropic Value at Risk

6.1 Introduction

An attractive feature of risk measures is that they can estimate heavy tail values while also distinguishing between actions with the same expected value but different variances [33]. This ranking or ordering of such nuanced stochastic valuations is useful for intelligent systems which may operate in the presence of similarly valued or heavy tailed random variables. Dynamic risk is of further interest as it can account for the ordering relationships of randomly-valued actions over time [135]. In this Section, we examine whether any condition exists where the entropic value at risk (EVaR) is a dynamic risk measure, as well as discuss the potential implications of some interesting conditions where EVaR is not a dynamic risk measure.

This work is organized as follows: In Section 6.2 we briefly overview the relevant information-theoretic quantities and an observation estimation theory. In Section 6.2.1 we introduce theoretical and practical considerations which motivate the focus on EVaR in this Section. In Section 6.2.2 we discuss the superior or equivalent stochastic ordering guarantee of EVaR as compared to the stochastic ordering guarantees of other popular risk measures. In Section 6.2.3 we introduce dynamic risk and its properties. And in Section 6.3 we develop a condition where EVaR is a dynamic risk. In Section 6.4 we briefly show that AVaR can also be a dynamic risk measure and discuss the potential implications of when EVaR is not guaranteed to be a dynamic risk measure.

6.2 Preliminaries

This work focuses on EVaR due to results from disparate disciplines. For readability and clarity, we introduce relative entropy first and work our way through intermediary concepts before defining EVaR. Relative entropy is defined as

$$D_{KL}(Q(X)||P(X)) := \int_{-\infty}^{+\infty} q(X) \ln \left(\frac{q(X)}{p(X)} \right) dX, \quad (54)$$

where $Q(X)$ is a probability measure with a corresponding probability density function (pdf) $q(X)$ and $P(X)$ is another probability measure with a corresponding pdf of $p(X)$. A constraint on the definition is that $Q(X)$ must be absolutely continuous with respect to $P(X)$. That is, if there is any value of X where $p(X) = 0$ then $q(X) = 0$. This absolute continuity condition

is written as $Q(X) \ll P(X)$. The relative entropy is also 0 if and only if $p(X) = q(X)$ for all values of X [85].

The meaning of relative entropy is described as a kind of asymmetric distance measure, meaning that the $D_{KL}(Q(X)||P(X))$ is not necessarily equal to $D_{KL}(P(X)||Q(X))$ in [29]. A clearer description is to state that relative entropy is the subtraction of entropy from cross entropy, where (with some liberties in notation) entropy is defined as

$$H_Q(X) := \int_{-\infty}^{\infty} q(X) \ln \left(\frac{1}{q(X)} \right) dX \quad (55)$$

and cross entropy (with the same liberties in notation) is defined as

$$H_{\times}(q(X), p(X)) := \int_{-\infty}^{\infty} q(X) \ln \left(\frac{1}{q(X)} \right) dX, \quad (56)$$

where, for clarity, the subscript $\times$ is not X and the subscript would be the same if evaluating the cross entropy for some random variable Y .

Entropy is axiomatically derived to uniquely quantify the uncertainty of a random variable in [146]. Cross entropy upper bounds entropy and constitutes the larger uncertainty caused by approximating $q(X)$ with $p(X)$. With that stated, we may write relative entropy as quantifying entropic error

$$D_{KL}(Q(X)||P(X)) = H_{\times}(q(X), p(X)) - H_Q(X). \quad (57)$$

In fact, an axiomatic result from [70]—with axioms similar to [146] but for divergence instead of entropy—shows that the relative entropy between a process at times t and t_0 uniquely quantifies the entropy required in a message (batch of observations) to change $q_{t_0}(X)$ to $q_t(X)$. This quantity is referred to by Hobson as the information gain.

As will be elaborated upon later, dynamic risk relates to a sequence of conditional risk measures, which often includes conditional expectations in the calculations [40]. Banerjee et al. found in [7] that an expectation conditioned on past observations is the optimal predictor with respect to Bregman divergences, where relative entropy (and therefore the information gain) is a Bregman divergence.

6.2.1 Static Risk

This leads us to the Entropic Risk Measure (ERM) which is not to be confused with EVaR, and is defined as

$$ERM[X; \theta] := \frac{1}{\theta} \ln \left(\mathbb{E}_P \left[e^{r\theta} \right] \right), \quad (58)$$

where θ is the risk aversion parameter, $\lim_{\theta \rightarrow 0} ERM[X; \theta] = \mathbb{E}_P[X]$ and $\lim_{\theta \rightarrow 0} ERM[X; \theta] = +\infty$. Note that the ERM is just a scaled cumulant generating function (cgf), where the cgf is simply the logarithm of the moment generating function (mgf)

$$mgf(X) = \mathbb{E} \left[e^{tX} \right], \quad (59)$$

where t is some positive constant. So in order to calculate the ERM, the mgf must exist, so we exclude the consideration of random variables such as a Cauchy random variable.

The ERM may be written as

$$ERM[X; \theta] = \sup_{\substack{Q(X) \\ Q(X) \ll P(X)}} \left\{ E_Q[X] - \frac{1}{\theta} D_{KL}(Q(X)||P(X)) \right\}. \quad (60)$$

Note that the ERM does not take into account (predictively or otherwise) the information gain in its definition in Equation (58). Also, the ERM only upper bound the expected values if the information gain is 0, so it may not be useful for intelligent systems in dynamic environments, depending on the application.

In [2] Ahmadi defined EVaR as

$$EVaR[X; 1 - \alpha] := \inf_{r > 0} \left\{ \frac{1}{r} \ln (\mathbb{E}_{\mathbb{P}} [e^{rX}] / \alpha) \right\}, \quad (61)$$

where $\alpha \in [0, 1]$ is the confidence parameter. Note that in order to calculate EVaR, the mgf of the random variable X must exist. Also, from a coarse visual inspection of Equation (61), one may note that EVaR is an optimization of ERM and the confidence parameter α . More interestingly, the robust (dual) representation of Equation (61) is

$$EVaR[X; 1 - \alpha] = \sup_{\substack{Q(X) \ll P(X) \\ D_{KL}(Q(X)||P(X)) \leq -\ln(\alpha)}} \{\mathbb{E}_Q[X]\}. \quad (62)$$

Thus, unlike ERM, EVaR does upper bound the expected value when the information gain is greater than 0. Also, the constraint on the information gain in Equation (62) may be rewritten as

$$e^{-D_{KL}(Q(X)||P(X))} \geq \alpha. \quad (63)$$

The confidence upper bound for EVaR in Equation (63) combined with the fact that EVaR monotonically decreases with an increase in α means that a lower information gain results in a tighter allowable upper bound of $\mathbb{E}_Q[X]$. In writing the tightest form of EVaR, we have

$$EVaR[X; 1 - \alpha] = \inf_{r > 0} \left\{ \frac{1}{r} (\ln (\mathbb{E}_P [e^{rX}]) + D_{KL}(Q(X)||P(X))) \right\}. \quad (64)$$

Unlike ERM, EVaR is not affected by a change in scaling of the random variable (see positive homogeneity below). In fact, EVaR is a coherent (and therefore also convex) risk [2] while ERM is just a convex risk measure [41]. A coherent risk measure, with the additional properties of translation invariance, sub-additivity, and almost-sure monotonicity. The coherent risk measure properties are listed below using an arbitrary risk measure ρ .

- Positive homogeneity: $\rho(aX) = a\rho(X)$, $a \geq 0$
- Translation invariant: $\rho(X + c) = \rho(X) + c$, $c \in \mathbb{R}$
- Sub-additivity: $\rho(X_1 + X_2) \leq \rho(X_1) + \rho(X_2)$
- Monotonicity: If $X \leq Y$ almost surely then $\rho(X) \leq \rho(Y)$

Other risk measures that are discussed in this Section include the value at risk (VaR) and the average value at risk (AVaR) which are defined as

$$VaR[X; 1 - \alpha] := \inf_{b \in \mathbb{R}} \{b : Pr(X \leq b) \geq 1 - \alpha\} \quad (65)$$

as written in [2] and

$$AVaR[X; 1 - \alpha] := \inf_{d \in \mathbb{R}} \{d + \mathbb{E}[X - d]_+\}, \quad (66)$$

in [139, 27] respectively. Note that VaR is not a coherent risk measure while AVaR is a coherent risk measure.

6.2.2 Stochastic Ordering

Stochastic ordering is purposed with numerically ranking different stochastically valued actions [34], and is a key benefit of using risk measures for intelligent systems. The focus is to determine which measures are most effective at accurately ordering two random variables, let's say X and Y , which are equal in distribution to two functions, $s(Z)$ and $t(Z)$, where Z is a random variable such that $X =_d s(Z)$, $Y =_d t(Z)$ and $s(Z) \leq t(Z) \forall Z$. The most basic stochastic ordering is an order in expectation, where $\mathbb{E}[X] \leq \mathbb{E}[Y]$. A stronger relationship holds with stochastic dominance, where

$$VaR[X; 1 - \alpha] \leq VaR[Y; 1 - \alpha] \quad \forall \alpha \in [0, 1] \quad (67)$$

as stochastic dominance implies an order in expectation, but an order in expectation does not imply stochastic dominance [78]. However, stochastic dominance does not, in general, numerically distinguish random variables with the same expected values but different variances [33]. Risk measures that can distinguish between such random variables are of the stop-loss order [35], where

$$\mathbb{E}[X - h]_+ \leq \mathbb{E}[Y - h]_+ \quad \forall h \in \mathbb{R}. \quad (68)$$

In [35, 8], monotonically increasing convex risk measures such as AVaR, ERM and EVaR are of the stop-loss order. Both the ERM and EVaR have a stronger ordering relationship described in [3] as being strongly monotonic. Strongly monotone risk measures has the following properties

- $X \geq Y$ with probability 1
- $Pr(X > Y) > 0$
- $\text{ess sup } \{X\} > \text{ess sup } \{Y\}$ or $\text{ess sup } \{X\} = \text{ess sup } \{Y\} = +\infty$

Thus, EVaR and ERM have stronger stochastic ordering guarantees than risk measures such as VaR and AVaR, and EVaR is a coherent measure while ERM is only a convex risk measure (as coherency implies convexity [42]).

6.2.3 Dynamic Risk

Dynamic risk measures are conditional risk measures, and in this Section our attention is on coherent dynamic risk measures. Coherent dynamic risk measures are of interest because they upper bound the conditional expectation of a time-varying random variable; i.e, from [43] with some simplification of the notation

$$\rho_t(X) = \text{ess sup}_{Q \approx P \text{ at time } t} \{\mathbb{E}_Q[X|\mathcal{F}_t]\}. \quad (69)$$

That is, a coherent dynamic risk measure bounds the conditional expected value calculated with the probability measure Q on the filtration $\mathcal{F}_t$. The filtration $\mathcal{F}_t$ grows as more observations occur; i.e., $\mathcal{F}_t \subset \mathcal{F}_{t+1} \subset \dots \subset \mathcal{F}_T$ [44].

Some properties of coherent dynamic risk measures are below [40]:

- Conditionally positive homogeneity: $\rho(aX) = a\rho(X)$, $a \geq 0$
- Conditionally translation invariant: $\rho_t(X + X_t) = \rho_t(X) + X_t$, $c \in \mathbb{R}$
- Conditional sub-additivity: $\rho_t(X + Y) \leq \rho_t(X) + \rho_t(Y)$
- Conditional monotonicity: If $X \leq Y$ almost surely then $\rho_t(X) \leq \rho_t(Y)$

A core property of dynamic risk measures is that they are time consistent, which means that if $\rho_{t+1}(X) \leq \rho_{t+1}(Y)$ then $\rho_t(X) \leq \rho_t(Y)$ [45]. Time consistency is equivalent to the following recursion [46]

$$\rho_t(X) = \rho_t(\rho_{t+1}(X)). \quad (70)$$

The idea being that an policy that is optimal at time $t + 1$, where all actions are evaluated using the same dynamic risk measure, is also optimal at time t . It is important to note that time consistency is an asymmetric temporal form of a stochastic order that applies only in the direction of $t + 1$ to t . That is to say that a policy that is optimal one time step from now is also optimal now. However it is subject to the requirement that $X_t \leq Y_t$ with probability 1 and $X_{t+1} \leq Y_{t+1}$ with probability 1 [26].

Some known dynamic risk measures include the ERM and the conditional expectation. Using the Tower Property (see [62]), the conditional expectation is shown to be time consistent below

$$\mathbb{E}[X|\mathcal{F}_t] = \mathbb{E}[\mathbb{E}[X|\mathcal{F}_{t+1}]|\mathcal{F}_t]. \quad (71)$$

Similarly, the Tower property is used for showing that the ERM is time consistent in [46]. For the analysis that follows, it is important to note that ERM is only time consistent if the risk aversion parameter θ is constant in time.

6.3 Dynamic Risk Form of EVaR

It must be acknowledged that EVaR, just like VaR and AVaR [13], is not necessarily a dynamic risk measure. In Theorem [6.1] we prove that EVaR can be a finitely-valued dynamic risk measure for certain values of the confidence parameter α .

Theorem 6.1. *There exists an $\alpha \in [0, 1]$ such that $EVaR[X; 1 - \alpha]$ is time consistent and finitely-valued.*

Proof. If EVaR is time consistent, then the recursion

$$EVaR_t[X; 1 - \alpha] = EVaR_t[EVaR_{t+1}[X; 1 - \alpha]; 1 - \alpha] \quad (72)$$

should be true. We expand the recursion to

$$EVaR_t[X; 1 - \alpha] = EVaR_t \left[\inf_{r_{t+1} > 0} \left\{ \frac{1}{r_{t+1}} \ln \left(\mathbb{E} \left[e^{r_{t+1}X} | \mathcal{F}_{t+1} \right] / \alpha \right) \right\}; 1 - \alpha \right] \quad (73)$$

and then to

$$EVaR_t[X; 1 - \alpha] = \inf_{r_t > 0} \left\{ \frac{1}{r_t} \ln \left(\mathbb{E} \left[e^{r_t \inf_{r_{t+1} > 0} \left\{ \frac{1}{r_{t+1}} \ln \left(\mathbb{E} \left[e^{r_{t+1}X} | \mathcal{F}_{t+1} \right] / \alpha \right) \right\}} | \mathcal{F}_t \right] / \alpha \right) \right\}. \quad (74)$$

Recall that the ERM is only time consistent if the risk aversion parameter θ is a constant. Similarly, we assume that the optimal parameter values at time t and $t + 1$ for EVaR are equal; i.e., $r^* = r_t^* = r_{t+1}^*$. Then the recursion becomes

$$EVaR_t[X; 1 - \alpha] = \frac{1}{r^*} \ln \left(\mathbb{E} \left[e^{r^* \frac{1}{r^*} \ln \left(\mathbb{E} \left[e^{r^*X} | \mathcal{F}_{t+1} \right] / \alpha \right)} | \mathcal{F}_t \right] / \alpha \right) \quad (75)$$

which simplifies to

$$EVaR_t[X; 1 - \alpha] = \frac{1}{r^*} \ln \left(\mathbb{E} \left[\mathbb{E} \left[e^{r^*X} | \mathcal{F}_{t+1} \right] / \alpha | \mathcal{F}_t \right] / \alpha \right). \quad (76)$$

Since α is a constant, it may be brought outside of the conditional expectations. By tower property, we then have

$$EVaR_t[X; 1 - \alpha] = \frac{1}{r^*} \ln \left(\mathbb{E} \left[e^{r^* X} | \mathcal{F}_t \right] / \alpha^2 \right), \quad (77)$$

which is true if $\alpha = 1$ or if $\alpha = 0$ as $\alpha = \alpha^2$ for those values, so we'd have

$$EVaR_t[X; 1 - \alpha] = \frac{1}{r^*} \ln \left(\mathbb{E} \left[e^{r^* X} | \mathcal{F}_t \right] / \alpha \right). \quad (78)$$

We conclude the proof by noting that Equation (78) is finitely valued for $\alpha = 1$ and is not finitely-valued for $\alpha = 0$. $\square$

6.4 Discussion

As mentioned earlier in this Section, the existence of a confidence parameter such that EVaR is a coherent dynamic risk measure is not unique to EVaR. We start this Section by showing that there exists a confidence parameter such that AVaR is a finitely-valued dynamic risk measure as well in Fact 6.2.

Fact 6.2. *There exists an $\alpha \in [0, 1]$ such that $AVaR[X; 1 - \alpha]$ is time consistent and finitely-valued.*

Proof. If AVaR is time consistent, then the recursion

$$AVaR_t[X; 1 - \alpha] = AVaR_t[AVaR_{t+1}[X; 1 - \alpha]; 1 - \alpha] \quad (79)$$

should be true. We expand the recursion to

$$AVaR_t[X; 1 - \alpha] = AVaR_t \left[\inf_{d \in \mathbb{R}} \left\{ d + \frac{1}{\alpha} \mathbb{E}[X - d | \mathcal{F}_{t+1}]_+ \right\}; 1 - \alpha \right] \quad (80)$$

and then to

$$AVaR_t[X; 1 - \alpha] = \inf_{d \in \mathbb{R}} \left\{ d + \frac{1}{\alpha} \mathbb{E} \left[\inf_{d \in \mathbb{R}} \left\{ d + \frac{1}{\alpha} \mathbb{E}[X - d | \mathcal{F}_{t+1}]_+ \right\} - d | \mathcal{F}_t \right]_+ \right\}. \quad (81)$$

If we assume that the optimal parameter values at time t and $t + 1$ for AVaR are equal (i.e., $d^* = d_t^* = d_{t+1}^*$), then the recursion becomes

$$AVaR_t[X; 1 - \alpha] = d^* + \frac{1}{\alpha} \mathbb{E} \left[d^* + \frac{1}{\alpha} \mathbb{E}[X - d^* | \mathcal{F}_{t+1}]_+ - d^* | \mathcal{F}_t \right]_+ \quad (82)$$

which, since α is a constant, reduces to

$$AVaR_t[X; 1 - \alpha] = d^* + \frac{1}{\alpha^2} \mathbb{E} \left[\mathbb{E}[X - d^* | \mathcal{F}_{t+1}]_+ | \mathcal{F}_t \right]_+ \quad (83)$$

and reduces further to

$$AVaR_t[X; 1 - \alpha] = d^* + \frac{1}{\alpha^2} \mathbb{E}[X | \mathcal{F}_t]_+ \quad (84)$$

which is true if $\alpha = 1$ or if $\alpha = 0$. We conclude the proof by noting that Equation (66) is finitely-valued for $\alpha = 1$ and not finitely-valued for $\alpha = 0$. $\square$

Note that the results show that both EVaR and AVaR are dynamic risk measures for similar assumed conditions with the same confidence values. However, it is unclear if the assumed conditions are the only scenarios with EVaR and AVaR may be dynamic risk measures.

It is also worth noting that the form of Equation (64) also relates to the intrinsic reward measure proposed in [73]. Intrinsic reward has been used with success in the reinforcement learning problem domain, despite not having any known stochastic ordering guarantees and intrinsic reward is not coherent for a constant η . It is therefore quite interesting that EVaR—which has stochastic ordering guarantees and is a coherent risk measure—can upper bound intrinsic reward. Specifically, if $\eta = \frac{1}{r^*}$ from Equation (3) in [73], then Equation (64) in this Section upper bounds the intrinsic reward measure as shown in the following Fact.

Fact 6.3. *EVaR $[X; 1 - e^{-D_{KL}(p(\theta|\xi_t, a_t, s_{t+1})|p(\theta|\xi_t))}]$ upper bounds the intrinsic reward from [73] if $\frac{1}{r^*} = \eta$.*

Proof. If $\frac{1}{r^*} = \eta$ then

$$EVaR \left[X; 1 - e^{-D_{KL}(p(\theta|\xi_t, a_t, s_{t+1})|p(\theta|\xi_t))} \right] = \frac{1}{r^*} \ln \left(\mathbb{E} \left[e^{r^* X} \right] / e^{-D_{KL}(p(\theta|\xi_t, a_t, s_{t+1})|p(\theta|\xi_t))} \right), \quad (85)$$

which simplifies to

$$EVaR \left[X; 1 - e^{-D_{KL}(p(\theta|\xi_t, a_t, s_{t+1})|p(\theta|\xi_t))} \right] = \frac{1}{r^*} \left(\ln \left(\mathbb{E} \left[e^{r^* X} \right] \right) + D_{KL}(p(\theta|\xi_t, a_t, s_{t+1})|p(\theta|\xi_t)) \right). \quad (86)$$

In substituting η for $\frac{1}{r^*}$, we then have

$$EVaR \left[X; 1 - e^{-D_{KL}(p(\theta|\xi_t, a_t, s_{t+1})|p(\theta|\xi_t))} \right] = \eta \left(\ln \left(\mathbb{E} \left[e^{\frac{1}{\eta} X} \right] \right) + D_{KL}(p(\theta|\xi_t, a_t, s_{t+1})|p(\theta|\xi_t)) \right). \quad (87)$$

In applying Jensen's quality (i.e., $\eta \ln \left(\mathbb{E} \left[e^{\frac{1}{\eta} X} \right] \right) \geq \mathbb{E}[X]$), we find that

$$EVaR \left[X; 1 - e^{-D_{KL}(p(\theta|\xi_t, a_t, s_{t+1})|p(\theta|\xi_t))} \right] \geq \mathbb{E}[X] + \eta D_{KL}(p(\theta|\xi_t, a_t, s_{t+1})|p(\theta|\xi_t)), \quad (88)$$

which concludes the proof. $\square$

Thus, when EVaR is a finitely-valued dynamic risk measure, there exists a time consistent upper bound to intrinsic reward that may be useful for approaches to navigate the explore-exploit dilemma.

The form Equation (77) with the assumption in Theorem 6.1 also has an interesting consequence when Equation (77) is rewritten in terms of its confidence upper bound

$$EVaR_t[X; 1 - \alpha] = \frac{1}{r^*} \left(\ln \left(\mathbb{E} \left[e^{r^* X} | \mathcal{F}_t \right] \right) + 2D_{KL}(Q(X)||P(X)) \right). \quad (89)$$

Thus, for EVaR to be time consistent with the assumption in Theorem 6.1 there must be no information gained (corresponding to $\alpha = 1$). As noted in [3, 2], if $\alpha = 1$ then EVaR is equal to the expected value. For our purposes, this means that $EVaR_t[X; 1] = \mathbb{E}[X|\mathcal{F}_t]$. Consequently, in terms of the tightest form of EVaR, EVaR is equal to the conditional expectation—which minimizes the information gain as noted by Banerjee et al. [7]—when the information gain is 0 due to the inequality in Equation (63). Similarly, AVaR is only equal to the conditional expectation when $\alpha = 1$, which would occur when an upper bound to the information gain is 0. However, it is important to note that AVaR does not have a direct numerical relationship with

any Bregman divergence, so AVaR may also be limited in any additional results that may be derived as compared to EVaR.

Given the stochastic ordering guarantees, coherency of EVaR, and explicit consideration of the information gain, it is of great interest to discern whether other conditions exist EVaR is a coherent dynamic risk. If not, then intelligent systems in dynamic environments may be restricted to using the ERM or conditional expectations, especially if the information gain is nonzero. As noted in this Section, both the ERM and conditional expectation lack at least one of the properties of EVaR. Thus, future work will focus on discerning whether there are any additional conditions where EVaR is a dynamic risk measure with a nonzero information gain.

Part III

Progress in the Year 2016-17

1 Nonparametric Kernel Observer Model and Inference Algorithms for Spatiotemporally Evolving Systems

1.1 Kernel Observer to Model Nonlinear Fluid Motion

We consider the problem of modeling and generalizing over sets of similar spatiotemporally evolving, nonlinear, dynamical systems. The differentially constrained machine learning model described in section 1 of part I, Evolving Gaussian Processes (E-GP), is the driving force behind our approach to this problem. We have shown that not only can an E-GP model be used to estimate the latent state of physical systems the type described, but that a single such model can generalize over multiple physically-similar systems over a range of parameters using only a few training sets. This is demonstrated on computational flow dynamics (CFD) data sets on fluid flowing past a cylinder over different Reynolds numbers. Surprisingly, though these systems are governed by highly nonlinear partial differential equations (the Navier-Stokes equations), our results show that their major dynamical modes can be captured by a linear dynamical layer over the temporal evolution of the weights of stationary kernels. Furthermore, the models generated by this method provide easy access to physical insights into the system, including the eigenvalues and Koopman modes from Koopman operator theory, unlike comparable methods like Recurrent Neural Networks (RNN). To our knowledge, no neural network has been able to model large-scale CFD systems that evolve in both space and time, and no other model at all has been able to generalize across similar physical situations. The low computational cost of our method suggests that it has the potential to enable machine learning approximations of complex physical phenomena for design and autonomy tasks.

1.1.1 Experimental Results: Computational Flow Dynamics Data

We used CFD methods to generate the states for a canonical fluid mechanics problem: flow past a cylinder at various Reynolds numbers, namely $Re = 100, 300, 600, 800$ and 1000 . This deterministic, high-dimensional spatiotemporal dynamical system is well-studied in the fluid dynamics literature, both experimentally and numerically [140, 18, 128]. In our CFD simulation, we used a 4th-order polynomial expansion with spectral element method on the incompressible Navier-Stokes equation to generate the cylinder flow data. The spatial domain is $[-2, 10] \times [-3, 3]$, excluding the diameter-1 cylinder at the origin. Neumann boundary conditions are applied to the far-field of the cylinder in the y-direction and the outlet of the flow field; and a Dirichlet

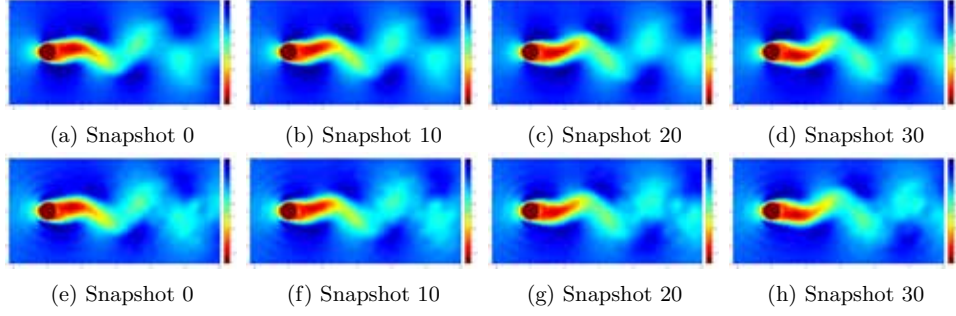


Figure 27: Visualization of Fluid Flow at $Re = 100$, CFD (a-d), E-GP (e-h)

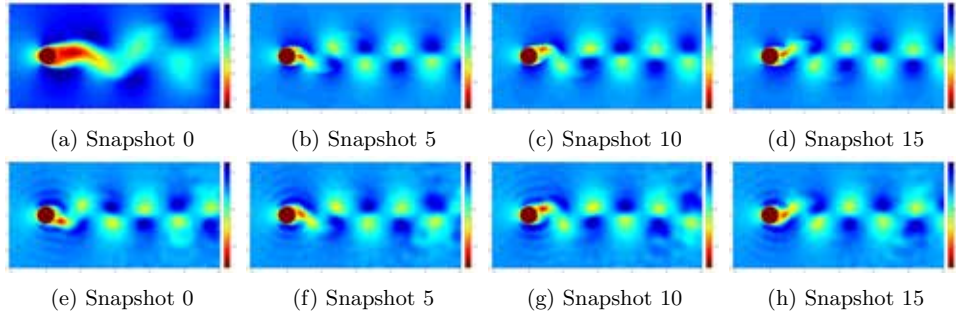


Figure 28: Visualization of Fluid Flow at $Re = 1000$, CFD (a-d), E-GP (e-h)

boundary condition is applied to the inlet. Each data set contains at least 200 snapshots (6 seconds). Each snapshot contains 24,000 velocity data points for $Re=100$ or 95,000 velocity data points for $Re=300,600,800,1000$. Each data set took at least 10 hours in a high performance computer cluster to generate. Figures 27 28(a-d) visualize the horizontal velocity for $Re=100$ and $Re=1000$, with red being the greatest negative velocity and blue the greatest positive velocity. The flow is unstable, periodic, and clearly nonlinear.

1.1.1.1 Modeling Individual Flows

There are three steps to generate an inference model for the kernel observer:

1. After picking the Gaussian RBF kernel $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$ and estimating the bandwidth hyperparameter σ (we utilize the maximum likelihood approach, although other approaches can be used), find an optimal basis vector set $\mathcal{C}$ using the algorithm in [31].
2. Use Gaussian process inference to find weight vectors for each time-step in the training set, generating the sequence $w_\tau, \tau = 1, \dots, T$.
3. Using the weight trajectory, use matrix least-squares with the equation $\hat{A}[w_1, w_2, \dots, w_{n-1}] = [w_2, w_3, \dots, w_n]$ to solve for $\hat{A}$.

In the prediction step, $\hat{A}$ is used to propagate w_τ forward to make predictions. Using $\sigma = 0.4$ and a budget of 600 kernel centers (see Figure 30a 30b), note how they cluster in the most dynamic

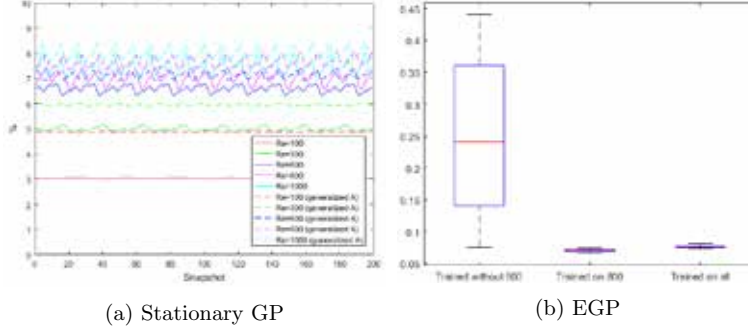


Figure 29: Total Percentage Errors

regions), we find a 600×600 matrix $\hat{A}$ which accurately (Figure 29a) captures the dynamics of the nonlinear system, with total percentage errors of 3% error for $Re=100$ and 7-8% for $Re=1000$. We define the total percentage errors as $E_\tau = \frac{\|y_\tau - \bar{y}_\tau\|_2}{\|\bar{y}_\tau\|_2}$ where $\bar{y}_\tau$ is the output vector for time τ and y_τ is the E-GP estimate at that time. Note that *the size of the model has been reduced by almost two orders of magnitude* from the original CFD data. This process takes about 13 minutes in MATLAB for a 200 snapshot by 95,000 point set on an ordinary Intel i7 4.00 GHz processor.

1.1.1.2 One Transition Matrix for Everything

In order to approach the challenge of generalizing across similar spatiotemporally evolving systems, the first question we must answer is whether we can find an $\hat{A}$ matrix that accurately captures the dynamics of multiple similar flows. The answer to that question is *yes*, and is achieved surprisingly easily. All that is required is to concatenate the weight trajectories from each flow in the least-squares computation of $\hat{A}$. That is, let $W_{Re} = [w_1^{(Re)}, w_2^{(Re)}, \dots, w_{n-1}^{(Re)}]$ and $W'_{Re} = [w_2^{(Re)}, w_3^{(Re)}, \dots, w_n^{(Re)}]$ be the weight trajectory and next weight trajectory for some Re . Then we solve the least-squares problem $\hat{A}[W_{100}, W_{300}, W_{600}, W_{800}, W_{1000}] = [W'_{100}, W'_{300}, W'_{600}, W'_{800}, W'_{1000}]$.

Amazingly, this single model works almost as well on all five data sets as do five individual models trained on each data set separately. This is confirmed by both the total error plots (Figure 29a), which show only slight increases in each of the total percentage error plots, and visual inspection of the dynamic modes displayed. This result is even more surprising in light of the fact that the rate of vortex shedding for each Reynolds number is different. By taking a Fourier transform of the time evolution of a data point located at (0.5,8), we find that for the original data sets the vortex shedding frequency is 0.448 Hz, 1.260 Hz, 1.380 Hz, 1.388, and 1.401 Hz for $Re=100, 300, 600, 800$, and 1000 respectively, and for the E-GP models the frequencies are 0.452 Hz, 1.21 Hz, 1.36 Hz, 1.36 Hz, and 1.36 Hz respectively.

1.1.1.3 Generalizing from Learned Dynamics to Unknown Dynamics

Having seen that it is possible to find a single transition in the weight space that models the dynamics systems over a range of parameters, the next challenge is to be able to model flows with parameters that the model has not been trained on. We derived an $\hat{A}$ matrix from the $Re=100, 300, 600$, and 1000 data sets and tested it against the $Re=800$ data set. The results are below in Figure 29b. For the first 120 snapshots, the total percentage error remains under

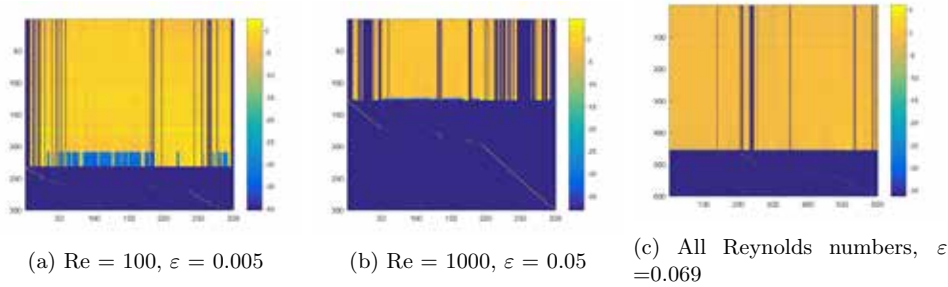


Figure 30: Total Percentage Errors

10%, which is satisfactory. After this, however, the total percentage error curves upwards as the slight errors in the transition matrix compound. Over 800 snapshots, we found an average total percentage error of less than 25%.

1.1.1.4 Linear Dynamical Layer Analysis & Insights

Due to the spatial encoding of the weights which the linear transition model operates on, we are able to analyze the dynamics and find physical insights into the process. We demonstrate two techniques: (1) using eigendecomposition of the transition matrix to discover the eigenfunctions and invariant subspaces of system, and (2) visualizing the most significant spatial interactions in the system.

An invariant subspace of a linear operator is a subspace of the Hilbert space such that any vector in the subspace remains in the subspace under transformation by the operator. By marking which kernel centers are associated with different subspaces, we can spatially separate the space into multiple dynamic modules. The physical insight is some areas of the space are dynamically entangled with each other, and other are independent of each other. For those interested in monitoring spatiotemporally evolving systems, the number and location of the invariant subspaces determines how many and where feedback sensors ought to be for robust prediction of the weights.

Before doing the Jordan decomposition of $\hat{A}$, we zero any elements smaller than some small ε in order to stabilize the algorithm for matrices with many elements close to zero. Afterwards we visualize the eigenvector matrix using a *logarithmic* color chart, as seen in Figures 30a 30b 30c. These plots are for models trained individually on $\text{Re}=100$ and $\text{Re}=1000$ with 300 kernels, and on all five with 600 kernels, for comparison. We see three categories of eigenvector in the rows: (1) Rows at the bottom that have exactly one non-zero elements, (2) In the middle, a couple rows with a dozen significant elements, and (3) at the top a number of rows that affect the majority of the kernel centers in the space.

Each eigenvector of (1) spans its own invariant subspace, and is depicted in magenta circles in Figures 31a 31b 31c. Category (3) is one invariant subspace, depicted with black crosses. Category (2) is subsumed in Category (3). The figures show that the dynamics near/around the cylinder and in its wake are so entangled that a single sensor measurement in that area may be sufficient to estimate over that entire subspace. On the other hand, areas far from the core of dynamic excitement are their own independent, invariant subspaces, and thus must be monitored locally.

Another way to visualize the operation of the linear transition matrix is to plot lines between

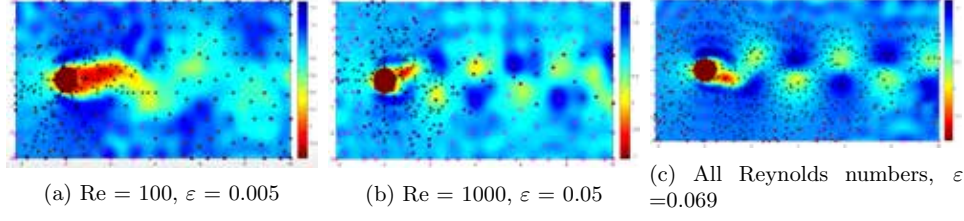


Figure 31: Invariant Subspaces

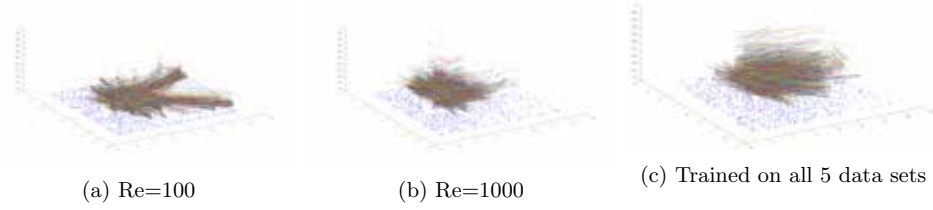


Figure 32: Visualization of Co-Relations in Transition Matrix

kernel centers that are influencing each other strongly. That is, if we draw a line center c_j to c_i for each of the (relatively) largest elements a_{ij} of $\hat{A}$, one can see how the system dynamics are coupled spatially (Figures 32a 32b 32c). We can also plot the magnitude of a_{ij} in a third axis for further insight into the most dominant dynamic connection in the system.

1.2 Random Sensor Placement

We consider the challenging problem of sensor placement through random selection. This process of random selection is a product of the EGP model described in section 1 of part IV. Otherwise to the best of our knowledge there does not exist any sensor placement design based on random selection. We utilize the kernel observer model and the definition of shaded observation matrix to prove Theorems that 1) derive the expected number of randomly placed sensors required to monitor a given spatiotemporal process, and 2) determine the probability with which optimal sensor placement is ensured given that the N number of sensors have been placed.

1.2.1 Theoretical Results: Random Sensor Placement

As discussed earlier, we work with an approximate feature space $\hat{\mathcal{H}}$, with the corresponding transition operator $\hat{A} : \hat{\mathcal{H}} \rightarrow \hat{\mathcal{H}}$ representing finite-dimensional functional evolution. To achieve observability for the pair $(\hat{A}, K)$, row vectors of the corresponding observability matrix $\mathcal{O}$ should form the basis for the $\mathbb{R}^M$ -dimensional $\hat{\mathcal{H}}$ space. According to the rational canonical structure Theorem 169, $\hat{A}$ can successively decompose the dual space $\mathbb{R}^M$ into subspaces $\mathcal{V}_i \subset \mathcal{V}$, $i \in \{1, \dots, \ell\}$, with properties i) $\mathcal{V} = \mathcal{V}_1 \oplus \dots \oplus \mathcal{V}_\ell$, ii) $\hat{A}\mathcal{V}_i \subset \mathcal{V}_i$, and iii) $\hat{A}|_{\mathcal{V}_i}, i \in \{1, \dots, \ell\}$, are cyclic. The integer ℓ is unique and is called the *cyclic index of $\hat{A}$* . Each of these properties contribute towards the theorem on the number of random samples required to achieve observability. First property shows that the space $\mathbb{R}^M$ can be decomposed into ℓ independent subspaces. The second property shows that the vector $w_i \in \mathcal{V}_i$ stays in $\mathcal{V}_i$ even when operated upon by $\hat{A}$. Thus in

order to generate bases for $\mathbb{R}^M$, one needs at least ℓ vectors say $\{w_1, \dots, w_\ell\}$, one each in the subspaces $\mathcal{V}_i$. This holds due to the third property, but requires that the vectors $\{w_1, \dots, w_\ell\}$ are cyclic generators of their corresponding subspaces. Can a randomly selected sensor generate a cyclic generator? A row vector $K_{(i)}$ generated by a randomly selected sensor x_i takes the form

$$K_{(i)} = [k(x_i, c_1), \dots, k(x_i, c_M)]. \quad (90)$$

Here, for radial kernels, the entries corresponding to the sensors closer to x_i tend to be non-zero, whereas the others tend to be zero. The rows $K_{(i)}$ from random sensor placement must be able to generate a basis for a subspace $\mathcal{V}_i$, and thus must be cyclic generators. We will derive the expected number of random sensor placements sufficient for observability for the case where $\hat{A} = \Lambda$ and then generalize the result for any $\hat{A}$. Note Λ is a block diagonal Jordan form. In this case, cyclic generator for each subspace $\mathcal{V}_i$, is a vector v_i with non-zero entries corresponding to the leading entry of the Jordan blocks of $\mathcal{V}_i$. For each subspace $\mathcal{V}_i$, let $\mathcal{C}_{\mathcal{V}_i} \subset \mathcal{C}$ be the centers corresponding to those non-zero entries: then the minimum number of random samples required to generate the bases for $\mathcal{V}_i$ is equal to the number of Jordan blocks comprising $\mathcal{V}_i$. Altogether, the minimum number of random samples required to generate a basis for $\mathbb{R}^M$ is equal to the total number of Jordan blocks in $\hat{A}$. Let ς be the total number of Jordan blocks in $\hat{A}$, then

$$\varsigma = \sum_{\lambda \in \sigma(\hat{A})} \gamma_{\hat{A}}(\lambda) \quad (91)$$

where $\sigma(\hat{A})$ represents the spectrum of $\hat{A}$ whose elements are the eigenvalues of $\hat{A}$, $\gamma_{\hat{A}}(\lambda)$ is the geometric multiplicity corresponding to the eigenvalue λ , which is also equal to the total number of Jordan blocks corresponding to the eigenvalue λ . Define a set of centers $\mathcal{C}_\varsigma$ with elements $\{c_1, c_2, \dots, c_\varsigma\}$, to be the centers corresponding to the leading entries of the Jordan blocks. For any random sample $x \in \Omega$ and $\epsilon > 0$, let $k(x, c_j) > \epsilon$ denote the region $\Omega_j \subset \Omega$ where the kernel evaluation with respect to c_j is greater than ϵ . Then p is defined as

$$p = \min_{c_j \in \mathcal{C}_\varsigma} \frac{\nu(k(x, c_j) > \epsilon)}{\nu(\Omega)}, \quad (92)$$

where ν is a measure in the real analysis sense. Hence, p corresponds to a lower bound on the probability that a random sample lies within the ϵ -shaded region of a particular center c_j . With all of this in place, we can prove the following theorem.

Theorem 1.1. *Given the spatiotemporal function $f(x, \tau)$ with $x \in \Omega \subseteq \mathbb{R}^D$, $\tau \in \mathbb{Z}^+$ and its EGP model [128], the expected number of random but uniformly placed sensor locations required to achieve observability for the pair $(K, \hat{A})$ is ς/p where ς is the summation over geometric multiplicities of each $\lambda \in \sigma(\hat{A})$ given by Equation [91].*

Proof. For each random sample, the probability that it lies within the ϵ -shaded region of a particular center $c_j \in \mathcal{C}_\varsigma$ is at least p . The series of random samples can be considered as Bernoulli trials in which p is the probability of a successful outcome. Note this is assuming worst case scenario that the intersection between any two ϵ -shaded region of centers belonging to the set $\mathcal{C}_\varsigma$ is empty. Observability for the pair $(K, \hat{A})$ is achieved after ς successful outcomes are obtained because each success ensures a row vector with non-zero entry corresponding to the leading entry of the Jordan block.

Let $X_1, X_2, \dots, X_N$ be i.i.d. random variables whose common distribution is the Bernoulli distribution with parameter p . The random variable $X = X_1 + X_2 + \dots + X_N$ denotes the

number of success after n random samples. Since each X_i has the Bernoulli distribution, X will have binomial distribution,

$$\binom{N}{\varsigma} p^\varsigma (1-p)^{N-\varsigma},$$

in which ς is the number of success. The expectation of the binomial distribution is ς/p , hence the proof. $\square$

Theorem 1.2. *Given the spatiotemporal function $f(x, \tau)$ with $x \in \Omega \subseteq \mathbb{R}^D, \tau \in \mathbb{Z}^+$ and its EGP model (128), summation over geometric multiplicities of each $\lambda \in \sigma(\hat{A})$ denoted by ς as in Equation (91), and a constant $\delta \in (0, 1]$, the probability that pair $(K, \hat{A})$ is unobservable after the selection of N random sensors is at most $e^{-\frac{1}{2}(Np-2\varsigma)}$, where p is given by Equation (92).*

Proof. The random variable X from the proof of Theorem 1 has a binomial distribution, which enables the application of a Chernoff-type bound on its tail probabilities. The following result from [107] is directly applied to establish this Theorem. If X is binomially distributed, $\delta \in (0, 1]$, and $\mu = \mathbb{E}[X]$, then $P[X \leq (1-\delta)\mu] < \exp(\mu\delta^2/2)$, in which $\delta = 1 - \varsigma/(Np)$. The expression in the exponent can be simplified to $-\frac{1}{2}Np + \varsigma - \frac{\varsigma^2}{2Np}$. Note that $e^{\frac{\varsigma^2}{2Np}} \leq 1$. This implies that $\exp(\mu\delta^2/2) \leq e^{-\frac{1}{2}(Np-2\varsigma)}$. $\square$

The above results can be extended to the case when $\hat{A} \neq \Lambda$, which involves a change of basis to obtain $\Lambda = P^{-1}\hat{A}P$, where P is the projection map. To compare random placement with deterministic approach, we describe a practical method to determine sensor placement while satisfying the conditions mentioned in Theorem 1.3. We say practical since generating the Jordan normal form of real valued matrix $\hat{A}$ as required by the previous Propositions is computationally infeasible [138]. We resort to a combination of eigendecomposition of $\hat{A}$ and ℓ -shaded property of the operator K . Let the eigendecomposition be given as $\hat{A} = V\Lambda_d V^{-1}$ where V is the eigenvector matrix and Λ_d diagonal eigenvalue matrix. Thus the operation of $\hat{A}$ on weight vector is actually $V\Lambda_d V^{-1}\mathbf{w}_\tau$. We investigate the matrix V^{-1} to determine the sensor data points whose evolution is uncorrelated. Naturally, the set $\mathcal{V}$ of sensor data points can be divided into three disjoint subsets consisting of points: i) correlated with single data point (let this be denoted as $U_1 \subseteq \mathcal{V}$), ii) correlated with all other data points ($U_2 \subseteq \mathcal{V}$), and iii) subset $U_3 = \mathcal{V} \setminus U_1 \cup U_2$. We sequentially place sensors starting with the sensor locations in the set U_1 followed by set U_3 and then U_2 until we obtain ℓ -shaded operator K . This process is summarized in Algorithm 1. As an example consider figure 34a that plots the log of the real part of V^{-1} for a spatial phenomenon over 50 array of sensors (see results for more details). Clearly the sensor data points can be divided into aforementioned subsets, for example $U_1 = \{6, 9, 12, 13, 45, 50\}$.

Algorithm 11 Sensor Placement

Input: Linear operator $\hat{A}$, eigen-decomposition $\hat{A} = V\Lambda_d V^{-1}$.

- Compute the cyclic index ℓ of $\hat{A}$, obtain subsets $U_1, U_2, U_3 \subset \mathcal{V}$ from the inverse of eigenvector matrix V^{-1} .

- Construct set $U = \{U_1, U_3, U_2\}$ with elements in same order.

- Initialize $\mathcal{A} = \emptyset, p = 0$.

while K is not ℓ -shaded **do**

- Select u_p the p^{th} sensor from U and construct $K \in \mathbb{R}^{p \times M}$,

- Check shadedness of K as per Definition 1.1

- $\mathcal{A} = \mathcal{A} \cup u_p, p \leftarrow p + 1$

end while

Output: ℓ -shaded operator K and set $\mathcal{A}$ of selected sensor location.

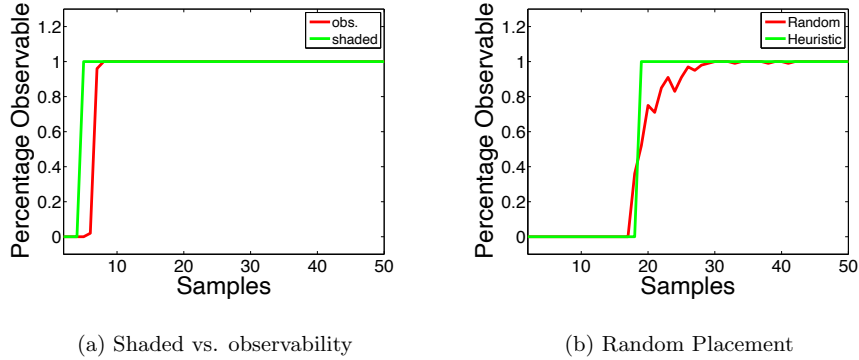


Figure 33: Synthetic experiments to verify (a) Shadedness property, (b) Random sample/sensor placement strategy.

1.2.2 Simulation Results: Random sensor placement

1.2.2.1 Sensor Placement for Synthetic Data Sets

The goal of this experiment is to investigate the dependency of the observability of system given by equation (128) on the shaded observation matrix and to verify random sensor placement results given in Theorems 1.1, 1.2. The domain is fixed on the interval $\Omega = [0, 2\pi]$. A uniform array of $M = 50$ sensor data points $\mathcal{C} = \{c_1, \dots, c_M\}$ over Ω is available for placement. First we construct a dynamics matrix $A = \Lambda \in \mathbb{R}^{M \times M}$, with cyclic index 5. We pick the RBF kernel $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$, $\sigma = 0.02$. Selecting sensors $\mathcal{X} = \{x_1, \dots, x_N\}$, $x_i \in \Omega$ randomly, we compute the ℓ -shaded property and observability for this system. Figure 33a shows how shadedness is a necessary condition for observability, validating Theorem 1.3; the slight gap between shadedness and observability here can be explained due to numerical issues in computing the rank of $\mathcal{O}_T$.

Next, we consider a system with a cyclic index $\ell = 18$ to verify random sensor placement results. We constructed the measurement operator K using the algorithm 11 as well as random sensor selection to generate the sensing locations $\mathcal{A}$. These results are presented in Figure 33b. The plot for random sampling that has been averaged over 100 runs, resembles a c.d.f function of an exponential distribution $F(X = x) = 1 - \exp(-\lambda x)$. This verifies the claim made in Theorem 1.2 as the probability of becoming unobservable is decaying exponentially with the number of sensor placed. Also, fitting an exponential distribution we found that the mean λ^{-1} is very close to the ratio $\frac{s}{p}$, which is the expected number of randomly placed sensors required for observability as per Theorem 1.1. In either case, observability is not achieved if the number of samples $N < \ell$ verifying the result in Proposition 1.

1.2.2.2 Intel Berkeley dataset

We now evaluate our sensor placement approach on a real world dataset. We also utilize the well-known mutual information based approach presented in [82] for sensor placement. For prediction we use two different models, GP with stationary squared exponential kernel, and the EGP in feature space. Note that the latter model is utilized for both sensor placement as well as prediction on test data. This is the key contribution of this work, holistic model for

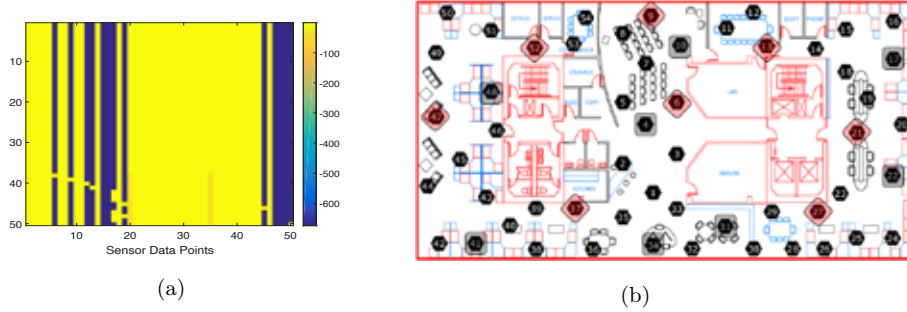


Figure 34: (a) Inverse of Eigenvector Matrix: Color indicates the value of the log of the real part of V^{-1} . Yellow stands for non-zero value while the other end of spectrum dark blue indicates zero. (b) Sensor placement for monitoring temperature at Intel Berkeley Research lab. Possible options for sensor placement are numbered black hexagons. Sensor placement using mutual information [82] and EGP in feature space is shown as squares and diamonds respectively.

spatiotemporal evolution and sensor set selection.

In the Intel Berkeley research lab temperature data, we analyze measurements from 50 wireless sensors in indoor laboratory region spanning 40.5 meters in length and 31 meters in width. Training data consists of three days of temperature data starting March 6th 2004 00:10 hrs at intervals of 10 minutes. Testing data consisted of the corresponding samples on the two following days. We followed the process described in [82] to obtain sensor placement based on mutual information criteria. Resulting placement marked by squares is shown in figure 34b that depicts sensor layout of the Intel Berkeley lab. To obtain sensor placement using EGP we first infer the model in three steps: 1) pick the Gaussian RBF kernel $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$, a search for the ideal σ is performed through minimization of log marginal likelihood [132]; 2) having obtained σ , Gaussian process inference is used to generate weight vectors for each time-step in the training set, resulting in the sequence $w_\tau, \tau \in \{1, \dots, T\}$; 3) matrix least-squares is applied to this sequence to infer $\hat{A}$. The cyclic index of $\hat{A}$ was $\ell = 3$. Next we applied Algorithm 11 to obtain the sensor set $\mathcal{A}$. Eigenvector matrix V^{-1} depicted in figure 34a was an input to this algorithm. Selected sensors are marked as diamonds in figure 34b. As $|\mathcal{A}| = 8$ we restricted the placement of sensors using mutual information to the same number.

Finally we evaluate the RMS errors in prediction over test data. As stated earlier, we used trained stationary GP and EGP models for prediction. In case of stationary GP (SGP), GP is estimated from the sensors selected using mutual information (MI) and EGP model. This estimate is then used to predict over the entire set of sensors. RMS error in prediction at every time-step depicted in figure 35a shows that the sensor placement using EGP improves prediction of temperatures over the entire lab except for some time-steps. In case of prediction using EGP model (figure 35b), considering overall performance sensor placement using EGP outperforms mutual information.

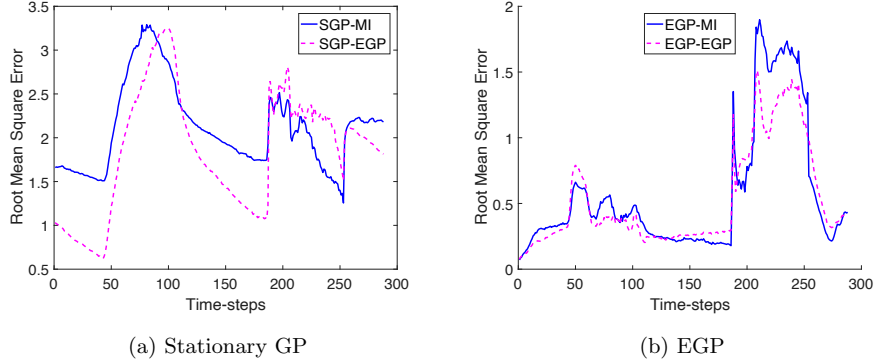


Figure 35: Prediction RMS errors for a) stationary GP and b) EGP using mutual information (MI) and EGP criteria for sensor placement. Total eight sensors were placed.

2 Harnessing Adversarial Attacks on Deep Reinforcement Learning for Improving Robustness

2.1 Introduction

Reinforcement learning in general has achieved several success in computer vision ([84]), dialog generation ([92]) and in robotics domain ([91], [145]). In robotics, it offers an interesting alternative to control systems which is primarily based on domain specific models of the plants. RL algorithms in principle can provide us with controllers that are not application dependent. Considering the breadth of robotic application, and the fact that in many situations exact physical models are either difficult to estimate or subject to first-principles approximation errors, the RL approach has tremendous potential in shaping the future of robotic control and decision making. However, this comes at the cost of curse of dimensionality because most robotic applications are continuous or high dimensional. This means that for real-world robotic applications RL algorithms require feature engineering and generic function approximators. The advances in Deep Neural Networks in recent years have had a tremendous impact on addressing this issue in those situations, especially in supervised learning ([84]) with no requirement of hand crafted features. Thus, using DNN as function approximator in RL algorithm has successful [95], [105] and offers state-of-the-art results in several RL tasks [147], [105].

However, Goodfellow et al. [59] have shown that Deep Networks can be fooled easily into predicting wrong label by perturbing the input with adversarial attacks. It opens up interesting frontier regarding adversarial attacks on machine learning algorithms in general. This assumes greater importance in the context of robotics and safety critical systems where such adversarial noise may lead to undesirable and hazardous situations. An interesting avenue is to see whether we can engineer such attacks on reinforcement learning algorithms. Our work tries to seek answers to this question and propose optimal attacks specific to Reinforcement Learning algorithms. We also attempt to attack Reinforcement learning algorithms that use radial basis function (RBF) as the underlying function approximator. Interestingly, we show that RBFs are relatively robust to adversarial attacks in comparison to DRL. All prior work on adversarial attack on DRL have focused on environments involving images and are similar in spirit to adversarial attack on images. To the best of our knowledge, our work is the first to look into adversarial attacks on

relatively lower dimensional continuous state space.

Another interesting frontier to explore is robustness of Deep RL algorithms to model variations. RL algorithms can be trained in simulator before being deployed in real world. Absence of high fidelity simulator can adversely affect the performance of these algorithms in real world. Thus, this provided another basis for our work, that is, how can we make Deep RL algorithms more robust to parameter mismatch between real dynamics and simulated dynamics.

Our contribution is two fold. We propose an objective function specifically in reinforcement learning setting whose optimization degrades the performance of RL algorithm. Furthermore, we show that the minima of proposed objective ensures that the RL agent takes worst possible action for any given state. This is then used to engineer adversarial attacks on reinforcement learning algorithms by injecting small and bounded noise in the observation space of the algorithm. We show that these attacks severely degrade the performance during evaluation of RL policies. Another interesting aspect of this work is the observation that RBF based RL algorithms were more robust to adversarial attacks as compared to DRL. Our second contribution is improving the robustness of Deep RL algorithm to parameter uncertainties. We train the DRL with proposed engineered adversarial attack and show that it becomes robust to parameter or model uncertainties.

In this section, we would explain the methods adopted for adversarial attack and further look into using these attacks for improving the robustness of reinforcement learning algorithm through robust control framework.

2.2 Adversarial Attack

In this subsection, we concentrate on generating adversarial attacks which will cause a trained reinforcement learning agent to fail. We assume that we can corrupt the observations for the agent which will cause it to follow a sub-optimal policy. We define an adversarial attack in the context of value function based reinforcement learning algorithms.

Definition 2.1. *An adversarial attack is any possible perturbation that leads the agent into increased probability of taking "worst" possible action in that state. Here, the "worst" possible action for a trained RL agent is the action which corresponds to least Q value in that state.*

It is important to note that def. 2.1 is valid only for value function based algorithms, i.e, the algorithms that use value function to predict the optimality of actions in a given state. We shall see in section 2.2.2 the objective function that needs to be optimized for achieving adversarial attacks in the context of reinforcement learning. There is a subtle difference as compared to image classification wherein attack is considered successful if a given image is classified as any other image. There is no concept of "worst" possible image.

An important point to note is that throughout the work we have bounded the adversarial attack noise by l_2 norm constraint. It is possible to use other norm constraints as well. Papernot et al. [119] showed that distillation is secure under l_∞ . Another point is that the RL environments are scaled so that the state space is normalized between $[0, 1]$. Thus, the magnitude of attack is always with respect to this scale.

2.2.1 Naive adversarial attack

First we consider a naive method of generating adversarial attack. The idea behind this attack is to generate random noise (several times in current state that the agent is in) and add it to current state with hope that one of these noise samples will cause the agent to take "bad" action, i.e, sub-optimal action. The quality of attack can be ascertained by the value function. Algo.

[12] outlines the naive adversarial attack for Deep Double Q Learning (DDQN) . An important point to note is that the attacks are generated during evaluation phase. The adversarial attack is essentially a search across nearby observation which will cause the agent to take wrong action. For generating adversarial attack on the Deep Reinforcement Learning policies, we sample a noise with finite (small) support. Noise is not generated once during every iteration, rather it is sampled for a number of times every iteration with a search for best adversarial noise (*beta* distributed noise with shifted 0 mean was used for generic setting but in our experiments the parameters of distribution were (1,1) which corresponds to uniform noise with 0 mean). The particular noise that causes least estimate of the value function is selected as adversarial noise. Then this noise is added to the current observation. The following Algo. [12] is outlined for naive attack on DDQN. We must point out that the extension to other value function based algorithm such as A3C and DDPG is trivial where the critic network can be used to ascertain value functions when required and actor network determines the behavior policy to pick action.

Algorithm 12 Naive attack

```

 $a^* = \arg \max_a Q(s, a), Q^* = \max_a Q^{target}(s, a)$ 
for  $i = 1 : n$  do
   $n_i \sim \text{beta}(\alpha, \beta) - 0.5$ 
   $s_i = s + \epsilon * n_i$ 
   $a_{adv} = \arg \max_a Q(s_i, a)$ 
   $Q_{adv}^{target} = Q^{target}(s, a_{adv})$ 
  if  $Q_{adv}^{target} < Q^*$  then
     $Q^* = Q_{adv}^{target}$ 
     $s_{adv} = s_i$ 
  else
    do nothing
  end if
end for
return  $s_{adv}$ 

```

2.2.2 Gradient based adversarial attack

In this subsection, we show that a proposed cost function different from the one used in traditional FSGM ([74]) is more effective in finding worst possible action in the context of reinforcement learning with discrete actions.

Theorem 2.2. *Let the optimal policy be given by conditional probability mass function (pmf) $\pi^*(a|s)$. Let the action which has maximum pmf be given as a^* and the worst possible action be given by a_w . Then the objective function whose minimization leads to adversarial attack on RL agent is given by*

$$J(s, \pi^*) = - \sum_{i=1}^n p_i \log \pi_i^*$$

where $\pi_i^* = \pi^*(a_i|s)$, $p_i = P(a_i)$, the adversarial probability distribution P is given by

$$P(a_i) = \begin{cases} 1, & \text{if } a_w = 1 \\ 0, & \text{otherwise} \end{cases} \quad (93)$$

In other words, it is the cross entropy loss between the adversarial probability distribution and optimal policy generated by the RL agent

Proof. We shall show that $\min_s J(s, \pi^*)$ achieves the objective of def. 2.1.

$$\begin{aligned}
J(s, \pi^*) &= - \sum_{i=1}^n p_i \log(\pi_i^*) \\
&= -p_w \log(\pi_w^*) = -\log(\pi_w^*) \quad (\text{from 93}) \\
\implies \min_s J(s, \pi^*) &= \min_s -\log(\pi_w^*) \\
\min_s J(s, \pi^*) &= \max_s \log(\pi_w^*)
\end{aligned}$$

Since log is monotonically increasing,

$$\min_s J(s, \pi^*) = \max_s \pi_w^*$$

Thus, we have shown that the objective function that should be used for engineering attack on RL algorithm should be given by Theorem 2.2. FSGM algorithm can be used to minimize this objective function. We must point out that this objective function is different from ones in literature 74. The objective functions mentioned in 74 will result in $\min_s \pi^*(a^*|s)$ (a^* is the best possible action for given state s). This leads to decrease in the probability of taking best possible action. This won't necessarily lead to increase in probability of taking worst possible action.

For performing the attack, we followed approach similar to naive sampling based attack but this time the search is more directed in the gradient direction of objective function described in Theorem 2.2. The algorithm has been described in Algo. 13.

Algorithm 13 Gradient based attack

```

1:  $a^* = \arg \max_a Q(s, a), Q^* = \max_a Q^{target}(s, a)$ 
2:  $grad = \nabla_x J(\theta, x, y)$ 
3:  $grad\_dir = \frac{\nabla_x J(\theta, x, y)}{||\nabla_x J(\theta, x, y)||}$ 
4: for  $i = 1 : n$  do
5:    $n_i \sim \text{beta}(\alpha, \beta) - 0.5$ 
6:    $s_i = s - n_i * grad\_dir$ 
7:    $a_{adv} = \arg \max_a Q(s_i, a)$ 
8:    $Q_{adv}^{target} = Q^{target}(s, a_{adv})$ 
9:   if  $Q_{adv}^{target} < Q^*$  then
10:     $Q^* = Q_{adv}^{target}$ 
11:     $s_{adv} = s_i$ 
12:   else
13:     do nothing
14:   end if
15: end for
16: return  $s_{adv}$ 

```

2.2.3 SGD based attack

We also tried out Stochastic Gradient Descent approach for adversarial attack. Wherein instead of sampling a few times and selecting best possible attack amongst these samples, we followed the gradient descent for same number of sampling time and selected the state that we end up in as adversarial state.

2.3 Robust Control framework

In RL, the typical objective that an agent seeks to maximize is it's expected long term return (η) (over possible trajectories τ) assuming a fixed transition model $T(s_t, a_t; \phi)$ characterized by parameters ϕ

$$\eta(\pi, T) = \mathbb{E}_{\tau}[\sum_{t=0}^T \gamma^t r(s_t, a_t) | s_0, \pi, T]$$

However, if there is variation in transition model, then criteria might be to perform well in expectation over all the possible transition models. Thus, leading to optimization of the mean performance of agent. The objective function in this scenario can be modified to

$$\eta(\pi) = \mathbb{E}_T[\eta(\pi, T)]$$

This is popularly known as risk neutral formulation. However, it has an underlying assumption that the distribution over transition model parameters are known apriori. It may not perform well over the transition model distributions because of high variance in transition model distribution. Thus, Tamar et al. [155] chose conditional value of risk (CVaR) as optimization criteria for robust control

$$\eta_{RC}(\pi) = \mathbb{E}_T[\eta(\pi, T) | \mathbb{P}(\eta(\pi, T) \leq \beta) = \alpha]$$

So, the problem boils down to maximizing the expected return over worst α percentile of returns. Thereafter, for sampling these bad trajectories, [129] changed transition model parameters and sample trajectories by performing rollouts with different parameters. [121] took an indirect approach where instead of sampling worst trajectory from rollout, they employ an adversary which applies control action and tries to push the RL agent into possible bad states. They train the adversary whose reward was negative of the reward of RL agent resulting in max-min game theoretic formulation. However, we take a direct approach where the adversary fools the agent into believing that it's in a "fooled" state different from actual state such that the optimal action in "fooled" state leads to worst action in actual current state. In other words, the adversary fools the agent into sampling of worst trajectories directly. We first train the algorithm using DDQN algorithm. Then it is made robust to model uncertainties through adversarial training. The algorithm has been discussed in Algo. 14. High magnitude of adversarial attack corresponds to highly conservative policy which may not perform well

2.4 Results

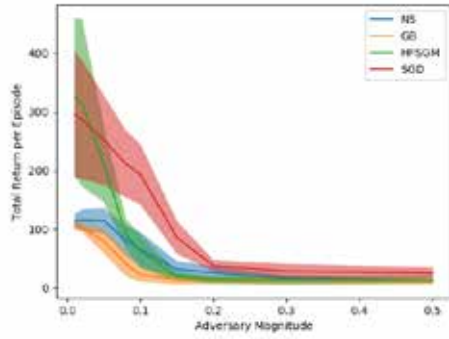
In this subsection, we discuss results related to adversarial attack and robust control policy with training using them.

Algorithm 14 Training with adversarial perturbation

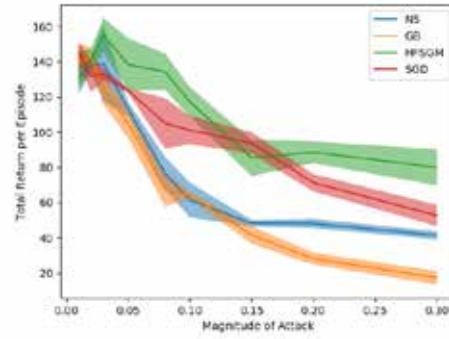
```

1: for  $i = 1 : iterations$  do
2:   Reset the environment and receive observation
3:   while not terminal or not max time steps per episode reached do
4:      $s_{adv} = Grad(Q^{target}, Q, s, \epsilon, n, \alpha, \beta)$ 
5:      $a = \arg \max_a Q(s_{adv}, a)$ 
6:      $s, r = Env(a, s)$ 
7:     Update the weights of network according to DDQN algorithm
8:   end while
9: end for

```

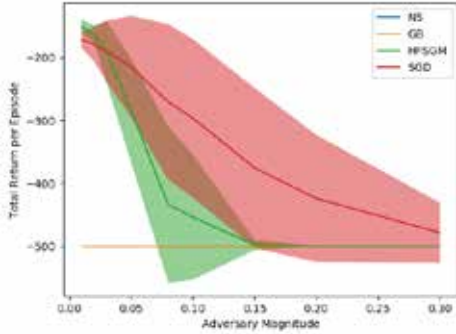


(a) DDQN Cart Pole

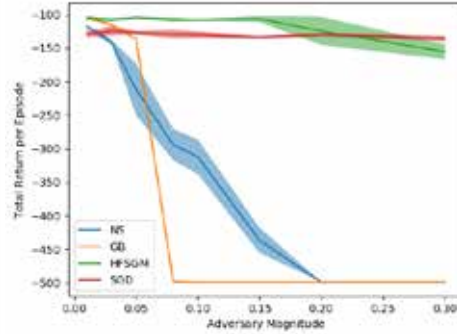


(b) RBF Q Cart Pole

Figure 36: Comparison of different attacks on DDQN and RBF based Q learning for cartpole environment. RBF Q learning is relatively more resilient to adversarial attack.



(a) DDQN Mountain Car



(b) RBF Q mountain car

Figure 37: Comparison of different attacks on DDQN and RBF based Q learning for mountain car environment. RBF Q learning is relatively more resilient to adversarial attack.

2.4.1 Adversarial Attack

Here NS refers to naive sampling attack, GB refers to gradient based attack, HFSFM refers to the attack in [74] and SGD refers to the stochastic gradient descent attack. As we can observe from Fig. [36] and [37] the adversarial attacks have degraded the performance of deep learning based algorithms. Furthermore, naive sampling based attacks and gradient based attacks perform better FSGM and SGD. Another observation is the relative resilience of RBF based Q learning algorithm as compared to DDQN. This can be explained by the fact that Deep Networks might be providing "jerky" or piecewise linear function approximators as opposed to smoother interpolation of RBF. Thus, a small perturbation in state causes agent to take bad action.

2.4.2 Robust Training

Figures [38] shows improvement in performance owing to robust training. There is significant improvement in the overall reward for cartpole environment. The agents receives reward of 1 for each time step during which the cartpole is balanced. In mountain car environment, it's interesting to observe significant gains around diagonal where the power coefficient is just enough to push the mountain car. In, the lower triangular region, there is no improvement because the power coefficient is not enough for the mountain car and is underpowered. In the upper right region, it is overpowered. Hence, it reaches goal easily.

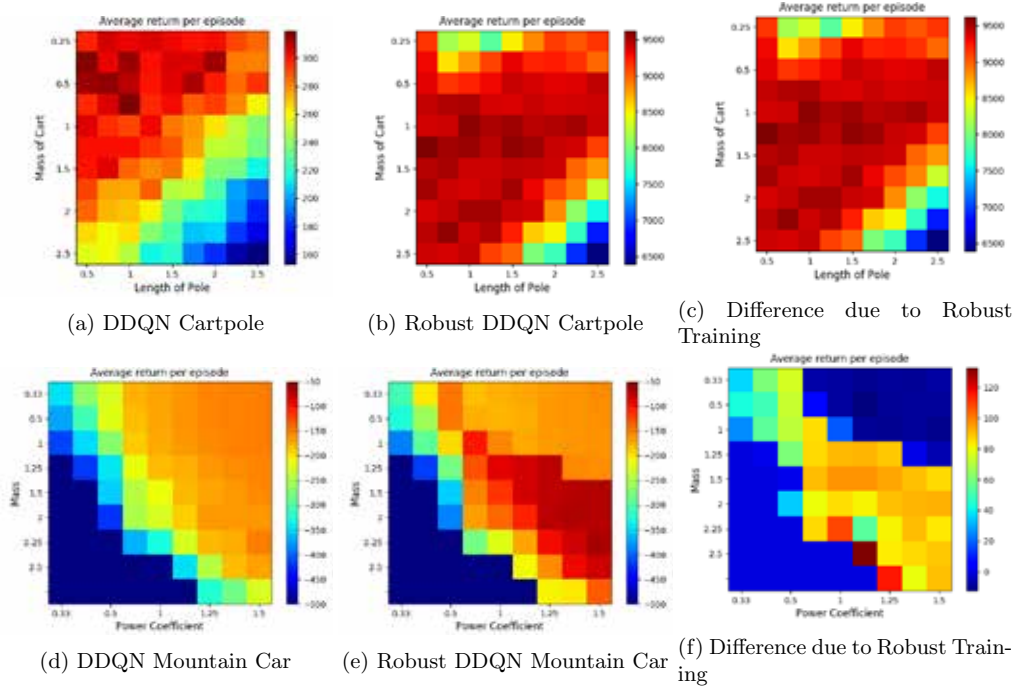


Figure 38: Robust training leads to higher return across parameter variations

2.5 Ongoing and Future Work

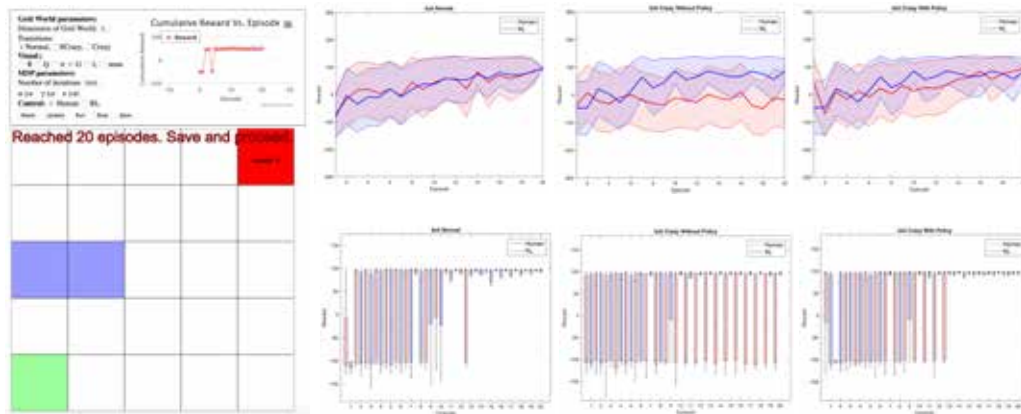
Further work involves gradient based adversarial attack on algorithms that involve continuous action space such as Deep Deterministic Policy Gradient(DDPG) [95] and Asynchronous Actor-Critic (A3C) [103] algorithms. It is interesting to observe that the objective functions in these cases are the critic network itself which lead to bad states. Minimization with respect to state that leads to lower value function of critic network can be considered as adversarial attack. We are also planning to carry out these attacks and robust training in high fidelity physics simulator MuJoCo [160].

3 Human Inspired Reinforcement Learning

We are interested in using a model of human learning to improve the learning performance and generalization of current RL methods [147, 65, 105]. Previous work has discussed the role of foundational context assumptions such as intuitive physics in learning [19, 90, 144]. This intuition can help to accelerate learning of novel tasks by providing a foundation for learning physical interactions in the environment, motivating us to examine tasks in which intuitive physical assumptions about the environment are violated, to determine the effect on learning. In our research, we demonstrate the difference in human and RL-agent learning. We compare the performance of human learners to that of Q-learning agents for specially designed grid-world decision-making problems with varying transition dynamics.

To implement these experiments, we designed a JavaScript app utilizing Canvas for visualization as shown in Fig. 39a. The app generates the grid world interface with the user, and runs a Q-learning agent. This framework allows users to play the game in their web browser, and can later be extended to a crowd-sourcing platform which allows data to be easily collected from many users via the Internet. The game environment, allows the user to specify the size of the grid, the MDP parameters for the RL agent, and the type of visualization for the grid world interface. The user can then play the grid world game or run the RL agent, and see the results displayed in a graph. In the future, we would like to use this framework to explore human machine collaboration for more complex environments, such as those with continuous action spaces, and generalizing these results to systems using Deep Neural Networks.

In this research, we demonstrate that the prior context knowledge is important for human learning, and that it is one of the essential instruments humans use to solve complex tasks. In Fig. 39b-39d we show that humans rely on intuition to map the transitions dynamics, and that they expected their intuition to be correct. Once their intuition fails, humans try to fit the new observed transitions into their prior beliefs, and to update the model of the environment. This causes a significant reduction in learning performance, and delays the convergence of the learned policy. The experiments also demonstrate that, once they find that transitions are non-intuitive, humans switch to a random policy based exploration with no motivating intuition. In all the experiments, RL agents show a unique learning rate. In addition, our results demonstrated that human performance improved in non-intuitive tasks when suggestions were given from a tabular Q-learning algorithm running in the background. These results encouraged us to extend this work to continuous state-action environments, which are much more natural to humans. The future use of a JavaScript based web app, with a server running a database, will allow us to collect crowd-sourced data with a greater population coverage.



(a) Web app implementing the user interface. (b) Fully intuitive transitions. (c) Fully random transitions. (d) Random transitions, RL policy suggestions.

4 Cross Domain Transfer Learning using Target Apprenticeship

Reinforcement Learning is a machine learning paradigm, where a robotic agent learns the optimal policy for performing a sequential decision making without complete knowledge of the environment. Recent successes in deep reinforcement learning have enabled RL agents to solve complex problems from balancing inverted pendulums [154] to playing Atari games [105]. Despite these recent successes, we do not yet understand how to efficiently transfer the learned policies from one task to another [157]. In particular, while some level of success has been achieved, in transferring RL policies in the same state-space domains, the problem of efficient cross-domain skill transfer is still quite open.

We consider the term “similar” source and target tasks, in the sense that they exploit the same underlying physical principles, but their state spaces can be entirely different. For example in our primary results, we consider the problem of knowledge transfer from balancing Cart-Pole to Bicycle balancing. While both the system share common dimensionality of state and action spaces but span across a different coordinate frame. The Cart-pole is defined over states of cart and pendulum, $(x, \dot{x}, \theta, \dot{\theta})$ and the action space is lateral force on cart $(-F, 0, F)$. Whereas bicycle dynamics is modelled over handle bar rotation and bicycle roll angle $(\theta, \dot{\theta}, \omega, \dot{\omega})$ with action space is torque applied by rider on the handle bar $(-\tau, 0, \tau)$. While the dynamics and domain of state space of two processes might be completely different, they share a commonality in the underlying physical principles. Both the systems exhibit non-minimum phase dynamics and also the nature of the control policy is same, such that the control action is applied, in the direction of the fall of pendulum or bicycle. This similarity in dynamical behavior of two systems makes learning in a cart-pole domain relevant in bike balancing problem. We also experiment with the problem of transferring the skills learned from the Mountain Car (Figure 40a) [154] to the Inverted pendulum. In the Mountain Car, the agent learns the optimal policy to make an underpowered car climb a mountain. On the other hand, in the pendulum domain, the agent learns to balance a pendulum upright from its initial down position. In both cases, the common physical principle the agent must learn is to exploit the principle of energy exchange.

The pendulum must be swung up to the upright position by creating enough angular momentum through smaller oscillations, and similarly, the car must be made to climb a steeper slope by using energy exchanged by moving up and down the slope of the mountain. In principle, a good RL agent would find it easier to balance a pendulum after it has learned a related task, to make an underpowered car climb a mountain.

Humans are capable of efficiently and quickly generalizing the learned skills between such related tasks. However, RL algorithms capable of performing efficient transfer of policies without learning in the new domain have not yet been reported.

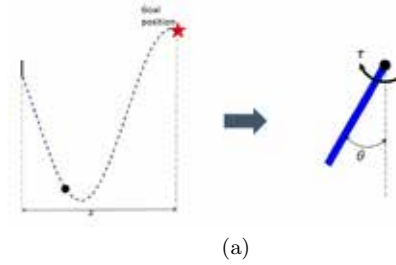


Figure 40: Cross Domain Policy Transfer from: (a) Cart-Pole to Bicycle and (b) Mountain Car to Inverted Pendulum

To address this gap, our main contribution is an algorithm that enables cross-domain transfer in the RL setting. Leveraging notions from apprenticeship learning [1] and adaptive control [28] we propose an algorithm that can directly transfer the learned policy from a source to target task.

4.1 Transfer Learning with Target Apprentice (TA-TL)

This section proposes a novel transfer learning algorithm capable of cross-domain transfer between two related but dissimilar source and target tasks. The presented architecture applies to both continuous and discrete state space systems with discrete action set. Unlike the available state of art TL algorithms, which mainly concentrates on policy initialization in target task RL; we propose to use source policy directly as the optimal policy in the related target model. We achieve this one step transfer through on-line correction of transferred policy with adaptive policy, derived based on model transition error. The presented approach has three distinct phases: Phase I involves finding an optimal policy for source task. For this purpose, we use Fitted Q-Iteration (FQI) RL to solve the source MDP. Since the source task is much simpler and is a smaller problem compared to the target tasks, we assume we can always discover an optimal policy for source task. Phase II; involves discovering a mutual mapping between source and target task state and action space using Unsupervised Manifold Alignment (UMA). Phase III of transfer in the proposed method, is the adaptation of the mapped source optimal policy through policy augmentation in a new target domain.

A possible drawback of the proposed method could be suboptimal transfer if the adaptive policy fails to account for the total model error between projected source and target task. For given a small “ ϵ ” residual model error by adaptive policy results in transferred policy leading to sub-optimal behavior in target space. With the advantage of high sample efficiency of the proposed technique in TL, near-optimal behavior in target task can be acceptable. It is to be noted; we do not engage in exploration in target space for transfer, we only exploit the projected source policy in target space to achieve near-optimal behavior. Nevertheless, with

further exploration, we can improve upon the adapted transferred policy and achieve an optimal solution, but this is left for follow-on work.

Details of three phases of proposed transfer learning technique using target apprentice model are as follows:

4.1.1 Phase I: Learning in the Source task

Fitted Q-Iteration is used to learn optimal policy in the source task. The policy search is not limited to Q-learning and can be extended to any other optimal policy generation methods. A single layer shallow network with radial basis function is used to approximate Q function. For the tasks considered in this work, shallow networks were found sufficient, but for more complex tasks a deep architecture with multiple layers can be used. This exercise is underway and left for follow-on work.

4.1.2 Phase II: Intertask Mapping

In TL in RL setting, the source and target task have a different representation of state and action spaces. The cross-domain transfer requires an inter-task mapping to facilitate a meaningful transfer. State space $s^{(S)}$ and $s^{(T)}$ belonging to two different manifold, cannot be compared directly, Unsupervised Manifold Alignment (UMA) technique helps to discover alignment between two datasets and provide an one to one and onto inter-task mapping. Using this inter-task mapping, allows us to build one step optimal policy for target task. It is important to note we consider same cardinality and analogous action spaces in analysis and experiments for ease of exposition of the proposed transfer architecture. Problems with distinct, non uniform action spaces will have to use classification methods to find correspondence between action spaces [156, 158]. TL is achieved through augmenting transferred policy by the adaptive policy learned over the target model. The proposed policy transfer and adaption method reuse the source policy in target space resulting in near-optimal behavior. Details of the inter-state mapping are provided in [4, 166] and reference therein.

4.1.3 Phase III: Transfer Learning through policy adaptation

This section presents a transfer algorithm for a pair of tasks in continuous/discrete state and action spaces. Algorithm [15] details TL through policy adaptation using apprentice model. Empirically we show presented method is sample efficient compared to other methods of transfer; since the sample complexity of learning an optimal policy for initialized target task is reduced to sample complexity of local apprentice model learning, around the optimal trajectory. Algorithm-[15] leverages the intertask mapping detailed in subsection 4.1.2 to move back and forth between source and target space for knowledge transfer and adaptive policy learning. The performance of the policy transfer depends on the quality of manifold alignment between source and target tasks. We assume UMA provides a one to one and onto correspondence between source and target state spaces for efficient transfer to happen. Algorithm-[15] provides pseudo-code for TL using target apprentice. Steps 1 – 8 provide an architecture for cross-domain policy transfer and step 9 – 12 details policy adaptation through target apprentice learning.

4.1.4 Algorithm: TA-TL

For every initial condition in target task $s_0^T \in \mathcal{S}^{(T)}$; s_0^T are transformed to find the corresponding initial condition on state space in source task.

$$\hat{s}_i^S = \chi_s^+(s_i^T) \quad (94)$$

Algorithm 15 Transfer Learning using target Apprentice model

Input: Source Policy $\pi^{*(S)}(s)$, Inter-task mapping χ_S and Apprentice Model $\hat{\mathcal{P}}^{(T)}$
repeat

 Initialize $(s_0, a_0)^T \in (\mathcal{S} \times \mathcal{A})^T$.

 Project the target task state to source model using inter-task mapping χ_s^+

$$\hat{s}_i^S = \chi_s^+(s_i^T)$$

 Evaluate the action using greedy policy on learned action-value function $Q^{(S)}(s, a)$ in source task,

$$\pi_i^{*(S)} = \arg \max_{\pi} (Q^{(S)}(\hat{s}_i^S, a))$$

$$a_i^S = \pi^{*(S)}(\hat{s}_i^S)$$

 Query the source task model at state $\hat{s}_i^S$ and action a_i^S

$$s_{i+1}^S = \mathcal{P}^{(S)}(\hat{s}_i^S, a_i^S)$$

 Project the source task propagated state to target task model,

$$\hat{s}_{i+1}^T = \chi_s(s_{i+1}^S)$$

 Evaluate the adaptive policy as $\pi_{ad}^{(T)} = \hat{\mathcal{P}}^{(T)}(s_i^T, a_i^T) - \hat{s}_{i+1}^T$

 Project the source policy into target space $\chi_s(\pi^{*(S)}(s, a))$

 TL policy for target task

$$\pi^{*T} = \pi^{*(S)}(\chi_s(s^S)) - \pi_{ad}^{(T)}$$

 Draw action from policy π^{*T} at step i and propagate the target model $a_i^T = a_{(S)i}^{(T)} - a_{ad,i}^{(T)}$

until $s_i^{(T)} \neq \text{terminal}$

where χ_s^+ is inverse mapping from target to source space and $\hat{s}_i^S$ represents the image of s_i^T in source space. For the mapped state in source task, a greedy action is selected using learned $Q^{(S)}(\hat{s}_i^S, a_i^S)$ state-action value function.

$$a_i^S = \arg \max_{\pi} (Q^{(S)}(\hat{s}_i^S, a_i^S)) \quad (95)$$

Assumption 1: The cardinality of the discrete action space is same in source and target task,

$$|\mathcal{A}^{(T)}| = |\mathcal{A}^{(S)}| \quad (96)$$

but the limits on action amplitude can be different

$$\|\mathcal{A}^{(S)}\|_2 \leq \tau^{(S)}, \|\mathcal{A}^{(T)}\|_2 \leq \tau^{(T)} \quad (97)$$

Using selected action a_i^S the source model at state $\hat{s}_i^{(S)}$ can be propagated to $\hat{s}_{i+1}^S$.

Assumption 2: Either transition model for the source task is available or that we can sample from a source model simulator. This assumption is not very restrictive since the designer can always select/create related source task for given target task.

$$s_{i+1}^S = \mathcal{P}^{(S)}(\hat{s}_i^S, a_i^S) \quad (98)$$

The propagated state in source task can be mapped back to the target space using inter-task mapping function,

$$\hat{s}_{i+1}^T = \chi_s(s_{i+1}^S) \quad (99)$$

where $\hat{s}_{i+1}^T$ is the image of s_{i+1}^S in target space. From Assumption-1, every selected action in source task has greedy correspondence in target task. Using this equivalence of actions for every selected action in source task an equivalent action in target task is selected as $a_i^T \in \mathcal{A}^{(T)}$. The selected action for target task is augmented with adaptive action $a_{ad}^{(T)} \in \mathcal{A}_{ad}^{(T)}$ derived from adaptive policy,

$$\pi_{ad}^{(T)} = \hat{\mathcal{P}}^{(T)}(s_i^T, a_i^T) - \mathcal{P}_{(S)}^{(T)}(s_i^S, a_i^S) \quad (100)$$

$$a_{ad}^{(T)} \in \pi_{ad}^{(T)}(s^T) \quad (101)$$

where $\hat{\mathcal{P}}^{(T)}(s_i^T, a_i^T) = \mathcal{F}^{(T)}(s_i^T) + \mathcal{B}a_i^T$ is apprentice model, $\mathcal{P}_{(S)}^{(T)}(s_i^S, a_i^S) = \chi_s(\mathcal{P}^{(S)}(s_i^S, a_i^S))$ is projected source model on to target space and $\mathcal{A}_{ad}^{(T)}$ is adaptive action space such that $|\mathcal{A}_{ad}^{(T)}| \geq |\mathcal{A}^{(T)}|$ and $\|\mathcal{A}_{ad}^{(T)}\| \leq \|\pi_{ad}^{(T)}\|_\infty$

Therefore the total transferred policy in target space, in solving a related target task is proposed to be a linear combination of mapped optimal policy and an adaptive policy as follows,

$$\pi^{*(T)}(.) = \pi^{*(S)}(\chi_s(.)) - \pi_{ad}^{(T)}(.) \quad (102)$$

4.1.5 Analysis

Theorem 4.1. *If, for any given small $\epsilon \geq 0$, the difference between true target model and target apprentice model over state-action set $\mathcal{S} \times \mathcal{A}$ is s.t.*

$$\forall (s, a) \in \mathcal{S} \times \mathcal{A}, \quad |\mathcal{P}^{(T)}(s, a) - \hat{\mathcal{P}}^{(T)}(s, a)| \leq \epsilon$$

Assumption 3: $a^S \in \pi^{*(S)}$ is the optimal policy for source task, then the modified policy $\pi^{*(T)}(.) = \pi^{*(S)}(\chi_s(.)) - \pi_{ad}^{(T)}(.)$ is ϵ -optimal in the target task

Proof. Consider a MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{H}, D, \mathcal{R})$, and suppose we have approximation $\hat{\mathcal{P}}$ of transition model, thus the approximate apprentice model can be written as the tuple $\hat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}, \mathcal{H}, D, \mathcal{R})$. We analyze the admissibility of the augmented policy for the target space. For this analysis we will consider deterministic, linearly parametrized discrete transition model for both source and target model as follows,

$$\mathcal{P}^{(S)}(s, a) : s_{i+1}^S = \mathcal{F}^{*(S)}(s_i^S) + \mathcal{B}a_i^S \quad (103)$$

$$\mathcal{P}^{(T)}(s, a) : s_{i+1}^T = \mathcal{F}^{*(T)}(s_i^T) + \mathcal{B}a_i^T \quad (104)$$

where $s^S \in \mathcal{S}^{(S)}$, $s^T \in \mathcal{S}^{(T)}$ and $a^S \in \mathcal{A}^{(S)}$, $a^T \in \mathcal{A}^{(T)}$. Deterministic state transition model is represented as $\mathcal{F}^{*(S)}(s_i^S) = A^{*(S)}\phi(s_i^S)$ and $\mathcal{F}^{*(T)}(s_i^T) = A^{*(T)}\phi(s_i^T)$, where $A^{*(S)}$, $A^{*(T)}$ are

state transition matrices for source and target tasks and $\phi(\cdot)$ is nonlinear basis vector and $A\phi(\cdot)$ is state dependent coefficient form of the nonlinear transition model $\mathcal{F}(\cdot)$

Let $\mathcal{P}_{(S)}^{(T)}(s, a) = \chi_s(\mathcal{P}^{(S)}(s, a))$ be projection of source task onto target space and the target apprentice model can be written as of the form,

$$\hat{\mathcal{P}}^{(T)}(s, a) : s_{i+1}^T = \hat{\mathcal{F}}^{(T)}(s_t) + \mathcal{B}a_i^{(T)} \quad (105)$$

where $\hat{\mathcal{F}}^{(T)}(s_t) = \hat{A}^{(T)}\phi(s_i^{(T)})$ and $\hat{A}^{(T)}$ be approximate transition matrix for target. For ease of exposition and without loss of generality we consider a linear in the state representation with $\phi(s) = s$ for further proof of optimality. Selecting the action from projected optimal policy $a_{(S),i}^{(T)} \in \pi^{*(S)}(\chi_s(s^T))$, augmenting with model adaption term and applying it to target model following holds,

$$s_{i+1}^{(T)} = A^{*(T)}s_i^T + \mathcal{B}a_{(S),i}^{(T)} - \mathcal{B}a_{ad,i}^T \quad (106)$$

From definition of model adaptive policy (101) and apprentice model (105), above expression can be simplified as

$$\begin{aligned} s_{i+1}^{(T)} &= A^{*(T)}s_i^T + \mathcal{B}a_{(S),i}^{(T)} - \mathcal{B} \left(\hat{\mathcal{P}}^{(T)}(s_i^T, a_i^T) - \mathcal{P}_{(S)}^{(T)}(s_i^S, a_i^S) \right) \\ s_{i+1}^{(T)} &= A^{*(T)}s_i^T + \mathcal{B}a_{(S),i}^{(T)} - \mathcal{B} \left(A^{(T)}s_i^T + \mathcal{B}a_{(S),i}^{(T)} - A_{(S)}^{(T)}\hat{s}_i^T - \mathcal{B}a_{(S),i}^{(T)} \right) \end{aligned} \quad (107)$$

Simplifying the above expression,

$$s_{i+1}^T = A_{(S)}^{(T)}s_i^T + \mathcal{B}a_{(S),i}^{(T)} + \mathcal{B}\epsilon \quad (108)$$

where $\epsilon = (A^{*(T)}s_i^T - A_{(S)}^{(T)}\hat{s}_i^T)$

The above expression (108), can be rewritten in terms of source transition model using the inter-task mapping function χ_s as

$$s_{i+1}^T = \chi_s \left(A^{*(S)}s_i^S + \mathcal{B}a_i^S + \mathcal{B}\epsilon \right) \quad (109)$$

Using **Assumption 3**, expression (109) demonstrates using the modified optimal policy (102) in target task is equivalent to projecting the optimal trajectories in source task on to the target task. Assuming UMA maps precisely the source to target task space, we prove the policy (102) leads to ϵ -optimal solution in target model. $\square$

5 Experiments & Results

We present results from three experiments to evaluate the proposed transfer learning framework. The first two experiments consider transfer learning in domains with the same state-space but with different transition models and action spaces. The first problem is in discrete state and action space, while the second problem involves continuous state space and non-stationary transition model in the target task. The third experiment focuses on cross-domain transfer where the policy from the mountain car problem is transferred to the inverted pendulum problem. To demonstrate the quality of the transfer, we compare the proposed Target Apprentice TL (TA-TL) against existing state of the art method Transfer Learning via Unsupervised Manifold Alignment (UMA-TL) [4] and stand alone RL (Fitted Q-Learning) on target domain with no initialization from source task learning.

5.1 Same-Domain Transfer

We learn the optimal policy in the source task using FQI. In each problem, distinction in the environment/system parameters makes the source and target tasks different. The target and source domains have the same state-space but different transition models and action spaces. We also do not need target reward model be similar to source task, as the proposed algorithm directly adapts the policy from the source task and does not engage in RL in the target domain.

5.1.1 Grid World to Windy Grid World

The source task in this experiment is Non-Windy (NW) grid world. The state variables describing the system are grid positions. The RL objective is to navigate an agent through obstacles from start to goal position optimally. The admissible actions are up (+1), down (-1), right (+1) and left (-1). The reward function is +10 for reaching goal position, -1 for hitting obstacles and 0 everywhere else. The target domain is same as the source but with added wind which affects the transition model in parts of the state-space. The optimal policy in source task (non-windy grid world) $\pi^{*(S)}$ is learned using Q-Iteration. We do not need any inter-task mapping as the source, and target state spaces are identical. We start with 100 randomly sampled starting position and execute policy $\pi^{*(S)}$ in the target domain and collect samples for apprentice model learning. Empirically, we show the proposed method (TA-TL) provides a sample efficient TL algorithm compared to other transfer techniques. Figure 41a and 41b shows the results of same domain transfer in the grid world, demonstrating TA-TL achieving successful transfer in navigating through the grid with obstacles and wind bias. Figure 41a and 41b shows the quality of transfer through faster convergence to average maximum reward with lesser training samples compared to UMA-TL and RL methods. The presented algorithm is able to attain maximum average reward in reaching goal position in $\sim 2 \times 10^4$ steps. UMA-TL and RL algorithm achieve similar performance in $\sim 1.2 \times 10^5$ and $\sim 1.7 \times 10^5$ steps respectively, nearly one order higher compared to proposed TA-TL.

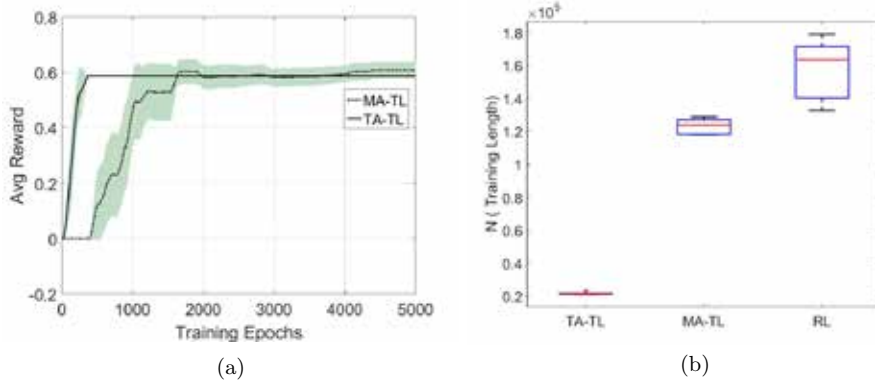


Figure 41: Non windy to Windy grid World Transfer:(a) Average Rewards (b) Training length, Comparing quality of transfer for TA-TL and UMA-TL through convergence rate of Average Reward and Training Length

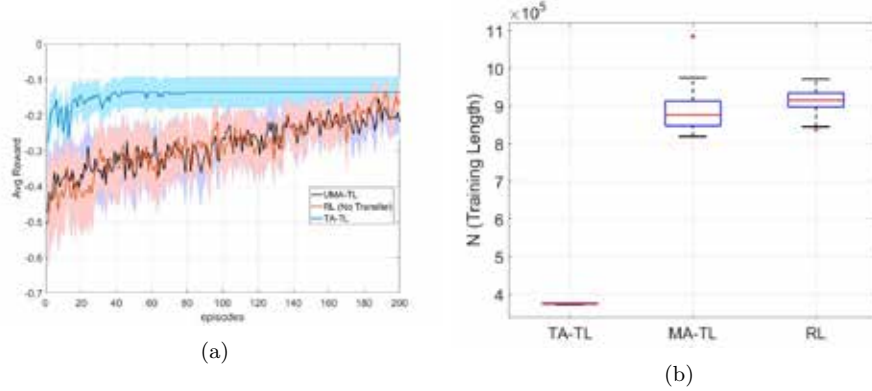


Figure 42: IP to Non-stationary IP Transfer: (a) Average Rewards and (b) Training length

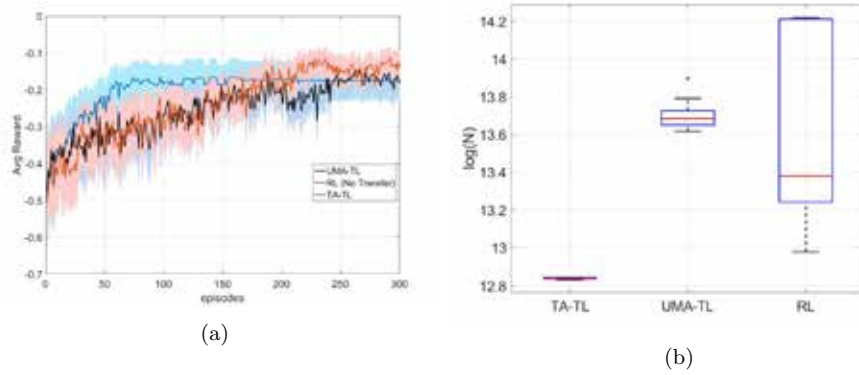


Figure 43: Mountain Car to Inverted Pendulum Transfer: (a) Average Rewards and (b) Training length

5.1.2 Inverted Pendulum (IP) to time-varying IP

We demonstrate our approach for a continuous state domain, Inverted Pendulum (IP) swing-up and balance. The source task is the conventional IP domain [154]. The target task is a non-stationary inverted pendulum where the length and mass of the pendulum are continuously time varying with function $L_i = L_0 + 0.5\cos(\frac{\pi i}{50})$ and $M_i = M_0 + 0.5\cos(\frac{\pi i}{50})$, where $L_0 = 1$, $M_0 = 1$ and $i = 1 \dots N$. The state variables describing the system are angle and angular velocity $\{\theta, \dot{\theta}\}$ where $\theta, \dot{\theta} \in [-\pi, \pi]$. The RL objective is to swing-up and balance the pendulum upright such that $\theta = 0, \dot{\theta} = 0$. The reward function is selected as $r(\theta, \dot{\theta}) = -10|\theta|^2 - 5|\dot{\theta}|^2$, which yields maximum value at upright position and minimum at the down-most position. The action space is: full throttle right ($+\tau$), full throttle left ($-\tau$) and zero throttle. Note that the domain is tricky, since full throttle action is assumed to not generate enough torque to be able to lift the pendulum to the upright position, hence, the agent must learn to swing the pendulum so that it generates oscillations and leverages angular momentum to go to the upright position. The target task differs from the source task in the transition model.

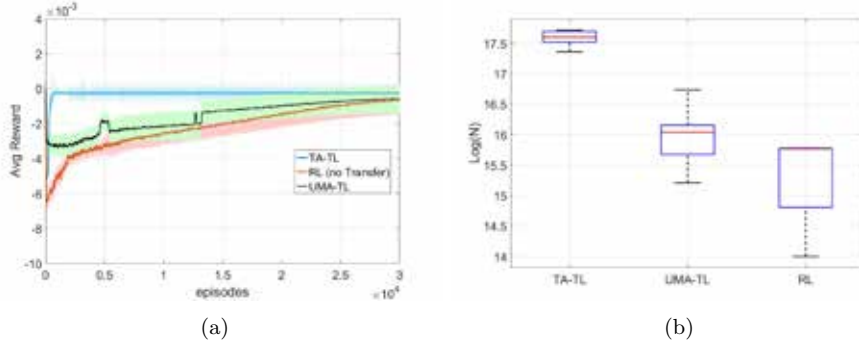


Figure 44: Transfer Cart-Pole to Bike Balancing task: (a) Average Rewards and (b) Total simulation time (seconds) the agent can balance the bike

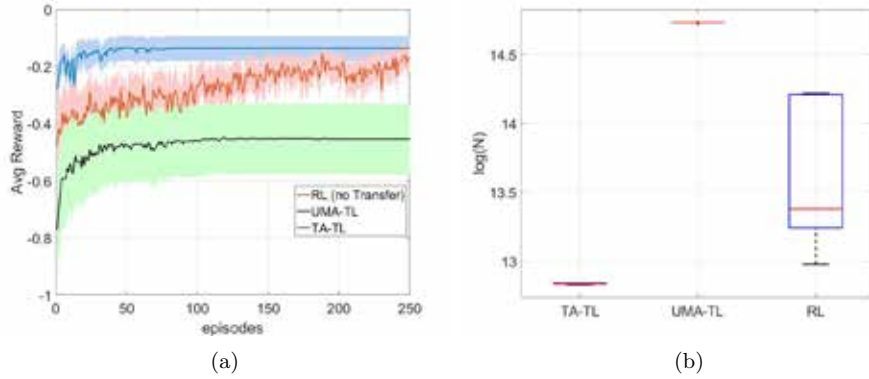


Figure 45: Negative Transfer Inverted Pendulum: (c) Average Reward and (d) Training length

The source task use FQI learning with single layer Radial Basis Functions (RBF) network. The Q function is modeled as linear combination of weights and basis as $Q = w'\xi(s)$. We use RBF bases " $\xi(s)$ " for value function approximation with bandwidth $\sigma = 1.2I$ and 20 centers spanning space $x, \dot{x} \in [-\pi, \pi]$ with network learning rate of $\Gamma = 5 \times 10^{-3}$ for FQI iterations.

Figure 42a and 42b shows the quality of transfer through faster convergence to average maximum reward with lesser training samples for proposed TA-TL method compared to UMA-TL and RL methods.

5.2 Cross-Domain Transfer

Next we consider an even more challenging transfer setting: cross-domain transfer. The problem setup is similar to same domain transfer with important distinction being state spaces are different for the source and target tasks.

5.3 Cart-Pole to Bicycle

Our last problem is the task where an agent learns to ride the bicycle. We consider the problem of learning to balance; we do not concentrate on the navigation problem to some goal position. Since RL for navigation is more of trajectory optimization problem, i.e., the agent learn to focus on maneuvering towards the target once it has learned to balance the bicycle upright. And also balancing is a more interesting problem when the bicycle is below critical velocity. Usually critically velocity V_c , above which bicycle is self stabilizing is approximately $V_c = 4m/s$ to $5m/s$. We are trying to learn to balance a unstable bicycle with forward velocity $V < V_c$ i.e. $V = 2.778m/s$. We also simulate the imperfect balance by inducing random noise in the CG displacement of rider up to $2cm$ from zero position.

At every time step agent receives information about the state of the bicycle, the angle and angular velocity of the bike handle bar and the bike from vertical $(\theta, \dot{\theta}, \omega, \dot{\omega})$ respectively. The agent can choose the action of applying torque to handle bar, $T \in [-2Nm, 0, 2Nm]$. The details of bicycle dynamics are beyond the scope of this paper, interested readers are referred to [130] and references there in.

We use the Cart-Pole as source task for learning to balance bicycle. The bicycle is not so different from the cart pole: in both the cases, the objective is to keep the unstable system upright. The objective of balance is achieved in both the systems by moving in the direction of fall. However, the control in the cart pole affects more directly the angle of the pole, i.e., move the cart such that it is always under the pole. In the bicycle, the control is to move the handle bar in the direction of fall. However balancing the bike is not so simple, to turn the bike under itself, one must first steer in the other direction, this is called counter steering. We observe that both cart pole and bicycle has this commonality in dynamical behavior, as both the system have a non-minimum phase that is the presence of unstable zero. They tend to move initially in the direction opposite to application control. This similarity qualifies the cart-pole system as an appropriate choice of source model for bicycle balance task.

Cart pole is characterized by state vector $[x, \dot{x}, \theta, \dot{\theta}]$ i.e., position, the velocity of cart and angle, angular velocity of the pendulum. The action space is the force applied to cart $F \in [-20N, 0, 20N]$. Like previous results, we use UMA to discover the correspondence between state space of bicycle and cart pole model. We use FQI to solve for optimal policy in the source, cart pole model. A linear network with cart pole states $\xi(s) = [x, \dot{x}, \theta, \dot{\theta}]$ is used as basis vector to approximate the action value function, with network learning rate $\Gamma = 1 \times 10^{-4}$.

Figure 44a shows the average reward accumulated by TA-TL, UMA-TL and RL (no transfer) in learning to balance the bicycle. In typical learning process, TA-TL out performs other transfer method and converges to maximum average reward in 1700 episodes. Each episode are trail until the bicycle balances without falling, episode ends if bicycle falls or 1000s is reached. Figure 44b shows total time bicycle was balanced upright by each method. Balancing times for TA-TL methods are highest and achieve ≈ 40 times more balancing time compared to UMA-TL method.

5.3.1 Mountain Car (MC) to Inverted Pendulum (IP)

We have tested the cross-domain transfer between mountain car to an inverted pendulum. The source and target task are characterized by different state and action space. The source task MC is a benchmark RL problem of driving an under-powered car up a hill. The dynamics of MC are described by two continues state variables $(x, \dot{x})$ where $x \in [-1.2, 0.6]$ and $\dot{x} \in [-0.07, 0.07]$. The input action takes three distinct values (+1) full throttle right, (-1) full throttle left and (0) no throttle. The reward function is proportional to negative of the square distance of the car from goal position. The target task is conventional IP as described in the previous experiment.

Cross-domain transfer requires a correspondence between inter-task space manifold for mapping the learned policy and source transition model from source space to target space and back. We utilize UMA to obtain this mapping as described in Section 4.1.2. We do not report the training time to learn the intertask mapping since it is common to both TA-TL and UMA-TL methods. We used a random policy to generate samples for manifold alignment and projected source policy in target space for target apprentice learning. The source task uses FQI learning with single layer RBF network for optimal policy generation. The source Q-function is modeled using function approximation as $Q = w'\xi(s)$ using RBF as bases " $\xi(s)$ " with bandwidth $\sigma = \text{diag}[0.3, 0.1]$ and 20 centers spanning space $x \in [-1.2, 0.6]$ and $\dot{x} \in [-0.07, 0.07]$ with network learning rate of $\Gamma = 0.15 \times 10^{-3}$. For all above results the training length involved with TA-TL method in Figure 43b, 42b, and 44b is sample lengths for target apprentice learning. We compare TA-TL with UMA-TL and generic-RL on target task. We examine the efficiency and effectiveness of transfer methods based on sample efficiency in learning the target task and speed of convergence to maximum average reward. Similar to same domain transfer Figure 43a and 43b shows the quality of transfer for TA-TL through faster convergence to average maximum reward with lesser training samples compared to UMA-TL and RL methods.

5.4 Negative transfer

Finally with our last set of results we demonstrate that the proposed transfer is robust to negative transfers. where either noisy features, imperfect or approximate set of source agents space result in poor or decreased performance in target domain.

Given a target model, the effectiveness of transfer depends on the relevance of the source model to the target task. If the relationship is strong and the transfer method can take advantage of it, the performance in the target task can significantly improve through transfer. However if the source is not sufficiently related or the features of source task does not correspond to the target, the transfer may not improve or even decrease the performance in target task leading to negative transfer.

We show that the UMA-TL suffers from negative transfer, where as proposed TA-TL performs superior compared to UMA-TL or no transfer RL. We demonstrate this through a inverted pendulum upright balance problem. We select inverted pendulum model as source and target systems. The target model is different from source model in sign of the control action. With exactly same dynamics in both source and target model but with the control sign flipped in target, we observed that initializing the action value function from source in target task learning lead to negative transfer. Whereas for the TA-TL method, since we learn the apprentice to target on line, the policy modification term is flipped accordingly and exactly same policy transfer performance is achieved as for same domain transfer between IP to non-stationary IP.

Figure 45a and 45b shows the quality of transfer through faster convergence to average maximum reward with lesser training samples for proposed TA-TL method compared to UMA-TL and RL methods (QL : Q-Learning) and MBRL (Model Based RL). It is to be observed that UMA-TL method converges to much lower average reward and gets stuck in a local optimum point and also the samples observed by UMA-TL in learning the task is also much higher compared to no transfer (RL) and proposed TA-TL methods.

6 GP regression in presence of non-Gaussian measurement noise

Data with non-Gaussian likelihood in Gaussian Process (GP) regression remains one of the basic issues in supervised learning. We are working on a new algorithm for GP regression over data with non-Gaussian likelihood that does not require costly MCMC sampling, or variational Bayes optimization. In our method, which we term Meta-GP, we model the likelihood by another Gaussian Process point-wise in time. This approach allows for the calculation of the posterior predictive mean and variance in an analytical way pointwise in time, leading to an online inference algorithm. As a result, our method can work with streaming data, is analytically tractable, computationally efficient while being as accurate or better than Expectation Propagation, Laplace Approximation, and MCMC inference methods for non-Gaussian likelihood data.

6.1 Main Contribution: Meta-GP

Consider a regression problem of the form $y = f(\mathcal{X}) + \epsilon$ with the input vector $\mathcal{X} \in D_x \subset \mathbb{R}^n$ and the scalar output $y \in \mathbb{R}$, where the latent function $f(\mathcal{X})$ is to be estimated from samples of y corrupted by noise ϵ . In typical GP regression, the noise ϵ is assumed to be zero mean additive Gaussian white noise: $\epsilon \sim N(0, \sigma_n^2)$. This is equivalent to assuming that the samples y have a Gaussian likelihood. The training set D consists of N observations, $D = \{(\mathcal{X}_i, y_i) | i = 1, \dots, N\}$. The input vector $\mathcal{X}_i$ and the output values y_i are assembled in the design matrix $\mathcal{X} = \{(\mathcal{X}_i)\}_{i=1}^N$ and the column vector $\mathbf{y} = \{(y_i)\}_{i=1}^N$. We use the popular squared exponential kernel $k_0(\mathcal{X}_i, \mathcal{X}_j)$ between the two arbitrary input vectors $\mathcal{X}_i$ and $\mathcal{X}_j$ with the hyperparameter σ_0^2 is termed as the bandwidth of the kernel. Note, that any other kernel can be utilized for the regression of the function $f(\mathcal{X})$ in the presented approach.

6.2 Online Posterior Predictive Distribution for arbitrary likelihoods

Written in a recursive manner, the Bayes' rule for GP inference for any arbitrary likelihood is [116]:

$$p(f_{t+1} | y_{1:t+1}) = \frac{p(y_{t+1} | f_{t+1}) p_t(f_{t+1})}{\int p(y_{t+1} | f_{t+1}) p_t(f_{t+1}) df_{t+1}}. \quad (110)$$

Eq. [110] allows the application of GP regression to online learning problems. Here, the subscript t denotes the number of collected samples at any given time. The prior distribution $p_t(f_{t+1}) = p(f_{t+1} | y_{1:t})$ is conditioned on all of the prior samples $y_{1:t} = \{(y_i)\}_{i=1}^t$, and $f_{t+1} = f(\mathcal{X}_{t+1})$ is the function value at the new data point $\mathcal{X}_{t+1}$. The posterior predictive distribution becomes:

$$p(f_* | y_{t+1}, \mathcal{X}_*) = \int p(f_* | \mathcal{X}_*, f_{t+1}) p(f_{t+1} | y_{1:t+1}) df_{t+1}. \quad (111)$$

Using eq. [110] and [111], we obtain the posterior predictive mean $E[f_*]_{t+1}$ of the underlying function $f_* = f(\mathcal{X}_*)$ at the test input $\mathcal{X}_*$:

$$E[f_*]_{t+1} = \int f_* p(f_* | y_{t+1}, \mathcal{X}_*) df_* = \frac{\int f_* p(f_* | \mathcal{X}_*, f_{t+1}) p(y_{t+1} | f_{t+1}) p_t(f_{t+1}) df_* df_{t+1}}{\int p(y_{t+1} | f_{t+1}) p_t(f_{t+1}) df_{t+1}}. \quad (112)$$

Similarly, predictive posterior variance is:

$$Var[f_*]_{t+1} = \int f_*^2 p(f_* | y_{t+1}, \mathcal{X}_*) df_* = \frac{\int f_*^2 p(f_* | \mathcal{X}_*, f_{t+1}) p(y_{t+1} | f_{t+1}) p_t(f_{t+1}) df_* df_{t+1}}{\int p(y_{t+1} | f_{t+1}) p_t(f_{t+1}) df_{t+1}} \quad (113)$$

Analytic (algebraic) computation of the predictive posterior mean and variance is only possible with Gaussian likelihood [115] [116]. More generally, in the Bayesian framework, an analytic solution exists if there is conjugacy between the prior distribution and the likelihood.

Our goal in this work is to obtain analytic approximation of the posterior mean and variance update equations even when the likelihood is non-Gaussian. To accomplish this, we utilize the properties derived from Stein's lemma [152].

Lemma 6.1. (Stein) Consider the random variable $Q \sim N(\mu, \sigma^2)$ and let $h : \mathbb{R} \rightarrow \mathbb{R}$ be an indefinite integral of the Lebesgue measurable function h' , essentially the derivative of h . Suppose that $E|h'(Q)| < \infty$, then

$$E \left[\frac{\partial h(Q)}{\partial Q} \right] = E \left[\frac{(Q - \mu)}{\sigma^2} h(Q) \right] \quad (114)$$

with the expected value $E[h(Q)] = \int h(Q)\phi(Q)dX$ of the probability $\phi(Q) = N(\mu, \sigma^2)$.

Let $Q = [f_*, f_{t+1}]$ and consider the prior distribution $\phi(Q) = p(f_*|\mathcal{X}_*, f_{t+1})p_t(f_{t+1})$ and $h(Q) = p(y_{t+1}|f_{t+1})$. Applying Stein's lemma for the multidimensional case with variable Q once for the mean $E[f_*]_{t+1}$ in eq. [112] we obtain the update equation for the predictive posterior mean

$$E[f_*]_{t+1} = E[f_*]_t + K_t(\mathcal{X}_*, \mathcal{X}_{t+1})\kappa^{(t+1)} \quad (115)$$

with the update parameter

$$\kappa^{(t+1)} = \frac{\partial}{\partial E[f_{t+1}]_t} \ln \int p_t(f_*, f_{t+1})p(y_{t+1}|f_{t+1})df_{t+1}df_*. \quad (116)$$

For the update equation of the variance $K_t(\mathcal{X}, \mathcal{X}')$, we apply the multidimensional Stein's lemma twice to Eq. [113] which leads to:

$$K_{t+1}(\mathcal{X}, \mathcal{X}') = K_t(\mathcal{X}, \mathcal{X}') + \lambda^{(t+1)} K_t(\mathcal{X}, \mathcal{X}_{t+1}) K_t(\mathcal{X}_{t+1}, \mathcal{X}') \quad (117)$$

with the update parameter

$$\lambda^{(t+1)} = \frac{\partial^2}{\partial E[f_{t+1}]_t^2} \ln \int p_t(f_*, f_{t+1})p(y_{t+1}|f_{t+1})df_{t+1}df_*. \quad (118)$$

The integral in eq. [116] and [118] can be interpreted as a convolution between the distributions $p_t(f_*, f_{t+1})$ and $p(y_{t+1}|f_{t+1})$. If the convolution is analytically derivable for arbitrary likelihoods (it is for Gaussian likelihoods), the first two moments of the posterior predictive distribution would always be analytically computable.

6.3 GP Approximation of the Non-Gaussian Likelihood

The key difference in our approach with existing literature is that instead of explicitly approximating the posterior distribution with Gaussians, we approximate the probability density function of the likelihood of the data as a GP with Gaussian kernels. This is akin to approximating the likelihood as a MoG. This enables us to derive analytic expressions for online inference. This section describes how this approximation is executed. To approximate the likelihood by a GP, choose a training set $Z = \{(y_i, p(y_i|f_i)) | i = 1, \dots, J\}$, where $[y_i \dots y_J]$ constitutes a preselected subset of $D_y \subset \mathbb{R}$ and $p(y_i|f_i)$ are the respective output values from the probability density function (PDF) of the non-Gaussian likelihood. The resulting MoG approximation's predictive mean is a weighted sum of Gaussian radial basis functions (G-RBFs), with their bandwidth being the variance of the Gaussian PDF. To differentiate the bandwidth σ_0^2 in the GP with non-Gaussian

likelihood we denote the bandwidth in the MoG likelihood approximation by σ_L^2 . In practice, a reasonable kernel bandwidth that leads to acceptable approximation error is sufficient since we are approximating a smooth stationary function (PDF of the likelihood), explicit hyperparameter optimization is not needed. Since the GP approximation of the likelihood is computed offline, its effect on the overall computational effort for the algorithm described in the next section is minimal, even with a spatially dense set of kernels. The number of kernels J is chosen and the weights α_j optimized using least-squares GP training [131] until a root mean square error to the original likelihood of less than 1 % is reached. Our approximation in the results use about 200 equally spaced G-RBFs. Using this approximation of the non-Gaussian likelihood, the likelihood can be rewritten as a sum of weighted normal distributions:

$$p(y_i|f_i) = \sum_{j=1}^J \pi_j N(y_i|\mu_j, \sigma_L^2, f_i) = \sum_{j=1}^J \pi_j \frac{1}{\sqrt{2\pi\sigma_L^2}} \exp\left(-\frac{(y_i - \mu_j - f_i)^2}{2\sigma_L^2}\right) \quad (119)$$

with weights $\pi_j = \alpha_j \sqrt{2\pi\sigma_L^2}$. To ensure a true approximation of the likelihood's PDF, it is necessary to normalize the weights α_j . Therefore, we expand the equation of the posterior predictive mean with the normalization constant $\sqrt{2\pi\sigma_L^2}$ and eventually calculate the weights π_j to achieve a sum of normal distributions that approximate the non-Gaussian likelihood.

6.4 Inference with Meta-GP

We substitute the GP model of the non-Gaussian likelihood (Section 6.3) in eq. 119, 116, and 118. Exchanging the sum and the integral we obtain the update parameters

$$\kappa^{(t+1)} = \frac{\partial}{\partial E[f_{t+1}]_t} \ln \sum_{j=1}^J \pi_j \int N(\tilde{\mu}_{j,t+1}|f_{t+1}) p_t(f_{t+1}) df_{t+1} \quad (120)$$

$$\lambda^{(t+1)} = \frac{\partial^2}{\partial^2 E[f_{t+1}]_t} \ln \sum_{j=1}^J \pi_j \int N(\tilde{\mu}_{j,t+1}|f_{t+1}) p_t(f_{t+1}) df_{t+1} \quad (121)$$

Taking the derivatives yields

$$\kappa^{(t+1)} = \xi \frac{1}{g(m)} g'(m) \quad \lambda^{(t+1)} = \frac{-1}{g^2(m)} g'^2(m) + \frac{1}{g(m)} g''(m) \quad (122)$$

with

$$g(m) = \sum_{j=1}^J \pi_j \int N(\tilde{\mu}_{j,t+1}|f_{t+1}) p_t(f_{t+1}) df_{t+1} \quad \text{and} \quad \tilde{\mu}_{j,t+1} = y_{t+1} - \mu_j. \quad (123)$$

Here, $m = E[f_{t+1}]_t$ represents the mean of the distribution after the convolution of the integral term. The update equation can be interpreted as a gradient descent method similar to that suggested in 116. Leveraging this insight, we introduce an additional learning rate ξ to speed-up convergence. *Inserting the sum of weighted Gaussian distributions from the likelihood approximation provides the key advantage that we are now able to solve the convolution in Eq. 122 analytically*, since the convolution of normal distributions is mathematically tractable and analytically computable (see e.g. 22). The approach enables analytical inference for Meta-GP and makes it simultaneously applicable for online and single-pass applications. To emphasize, that these calculations are still computable, the analytic calculations of function $g(m)$ and its first and second derivative of $g(m)$ in Eq. 122 123 were derived.

6.5 Implementation

Algorithm 16 illustrates the implementation of Meta-GP. First, the non-Gaussian likelihood is approximated with a weighted sum of normal distributions as described in Section 6.3. This step is executed offline. The outputs are the weights $\pi = \{(\pi_j)\}_{j=1}^J$ and the centers $\mu_L = \{(\mu_j)\}_{j=1}^J$ of each Gaussian in the MoG. The algorithm updates the means of the weighted Gaussian distribution $\tilde{\mu}_{L,t+1}$ in the likelihood approximation based on the arriving data point y_t . Eventually, Eq. 120-123 in section 6.4 are utilized to update the posterior predictive mean $E[f_*]_{t+1}$ and variance $K_{t+1}(\mathcal{X}, \mathcal{X}')$. For simplicity, we employ the online kernel sparsification algorithm in 32 since it is already in the online form, but we expect that follow-on work random Fourier feature based methods 127, low-rank approximation based methods 168 or GP sparse approximation methods 165, 149, 150 will lead to further improvement in performance. Note that the kernels of the GP with non-Gaussian likelihood do not have to be G-RBFs, and leading hyperparameter optimization techniques can be utilized to optimize them. However, for simplicity, our experiments use G-RBFs with stationary hyperparameters. Finally, the function $f(\mathcal{X})$ is predicted at the test value $\mathcal{X}_*$ using the computed parameters.

Algorithm 16 Meta-GP

- 1: **Input:** Training set $D = \{\mathcal{X}, \mathbf{y}\}$, Test values $\mathcal{X}_*$, Hyperparameters $\theta = [\sigma_0^2, \xi]$, Training set $Z = \{(y_i, p(y_i|f_i)) | i = 1, \dots, J\}$, Variance MoG σ_L^2
 - 2: $[\pi_L, \mu_L]$ = Likelihood Approximation (Z, σ_L^2) (Eq. 119)
with $\pi_L = \{(\pi_j)\}_{j=1}^J$ and $\mu_L = \{(\mu_j)\}_{j=1}^J$
 - 3: **Initialization:** $E[f_*]_1 = \kappa^{(1)}$, $K_1(\mathcal{X}, \mathcal{X}') = k_0(\mathcal{X}, \mathcal{X}') + \lambda^{(t+1)}k_0(\mathcal{X}, \mathcal{X}_1)k_0(\mathcal{X}_1, \mathcal{X}')$ using Eq. 120-123 with $t = 0$
 - 4: **Inference:**
 - 5: **for** $t = 1$ **to** $N - 1$ **do**
 - 6: $\tilde{\mu}_{L,t+1} = y_{t+1}\mathbf{I} - \mu_L$
 - 7: $\kappa^{(t+1)} = \frac{\partial}{\partial E[f_{t+1}]_t} \ln \sum_{j=1}^J \pi_j \int N(\tilde{\mu}_{j,t+1}|f_{t+1}) p_t(f_{t+1}) df_{t+1}$ using Eq. 120+122-123
 - 8: $\lambda^{(t+1)} = \frac{\partial^2}{\partial^2 E[f_{t+1}]_t} \ln \sum_{j=1}^J \pi_j \int N(\tilde{\mu}_{j,t+1}|f_{t+1}) p_t(f_{t+1}) df_{t+1}$ using Eq. 121+122-123
 - 9: $E[f_*]_{t+1} = E[f_*]_t + K_t(\mathcal{X}_*, \mathcal{X}_{t+1})\kappa^{(t+1)}$
 - 10: $K_{t+1}(\mathcal{X}, \mathcal{X}') = K_t(\mathcal{X}, \mathcal{X}') + \lambda^{(t+1)}K_t(\mathcal{X}, \mathcal{X}_{t+1})K_t(\mathcal{X}_{t+1}, \mathcal{X}')$
 - 11: **end for**
 - 12: **Prediction:** $E[f_*]_N$; $Var[f_*]_N = K_N(\mathcal{X}_*, \mathcal{X}_*)$;
-

6.6 Classification

The presented Meta-GP approach for GP regression with non-Gaussian likelihoods is especially useful in classification problems. Classification with GPs has been a difficult problem since probit or logit likelihoods are non-Gaussian 131. For simplifying exposition, we present first binary classification formulation. The training set D consists of N observations, $D = \{(\mathcal{X}_i, y_i) | i = 1, \dots, N\}$ with the input vector $\mathcal{X}_i$ and the corresponding class label $y_i \in \{0, 1\}$. Equivalent to the regression problem, the first step of Meta-GP Classification is the approximation of a latent function $\mathbf{f}(\mathcal{X})$ with the input vector $\mathcal{X}_i$ and the output labels y_i , which are assembled to the design matrix $\mathcal{X} = \{(\mathcal{X}_i)\}_{i=1}^N$ and the column vector $\mathbf{y} = \{(y_i)\}_{i=1}^N$. Considering a probit distribution as non-gaussian likelihood, the likelihood is written as $P(y_i|\mathcal{X}_i, f_i) = \text{sig}(y_i f_i)$ with $f_i = f(\mathcal{X}_i)$ 113. After the posterior predictive mean and variance is obtained as presented in the regression section, the predictive class membership $P(y_*|\mathcal{X}_*, \mathbf{y}, \mathcal{X})$ is gained by averaging out

the test set latent variables

$$P(y_*|\mathcal{X}_*, \mathbf{y}, \mathcal{X}) = \int \text{sig}(y_* f_*) p(f_*|\mathcal{X}_*, \mathbf{y}, \mathcal{X}) df_* = \frac{E_N[f_*]}{\sqrt{(1 + V_N[f_*])}}. \quad (124)$$

The predictive class membership describes, how likely the test input $\mathcal{X}_*$ belongs to the class $y_* = 0$ or $y_* = 1$. Using the posterior predictive mean and variance for $p(f_*|\mathcal{X}_*, \mathbf{y}, \mathcal{X})$, Eq. 124 is tractable [131]. For multiclass classification the binary classification approach is applied for each class, followed by taking the highest probability as prediction label [136].

6.7 Regression Results

In this section we present empirical results for both regression with non-Gaussian noise (likelihoods), and high-dimensional image classification problems with probit or logit non-Gaussian likelihoods. In all results, the G-RBF kernel is used for likelihood approximation, as well as for the latent GP approximation of function $f(\mathcal{X})$, although the latter could be accomplished with other kernels. A one-dimensional bandwidth σ_0^2 was utilized, simplifying hyperparameter optimization to a quick grid search. Although other kernels, non-stationary hyperparameters, and hyperparameter optimization algorithms such as [9, 151] could lead to better results, this is left for future work, since our focus here is on comparisons with other methods without the aid of hyperparameter optimization.

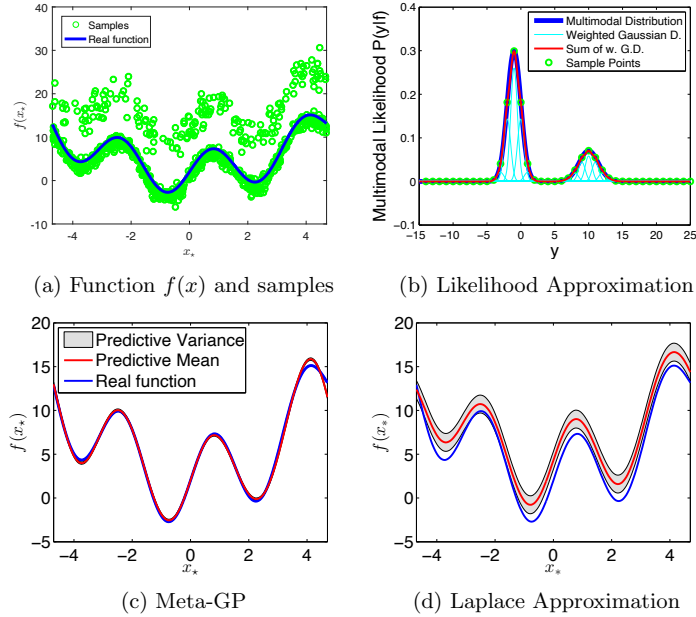


Figure 46: Regression results with multimodal posterior distribution

6.7.1 Regression

We compare the Meta-GP algorithm against EP algorithm [77], LA [163], MCMC [110], and sparse GP with Gaussian likelihood [32]. The approximation accuracy measured through Root

Table 3: RMSE and MAE for the following experiments: 2D-F, Boston housing data.

Experiment	2D-F			Boston		
	RMSE	MAE	Time[sec]	RMSE	MAE	Time[sec]
Meta-GP	0.1377	0.1099	0.6975	0.3853	0.2806	0.9018
Laplace [163]	0.1455	0.1180	63.7827	0.4166	0.3001	1.3333
EP [77]	0.1454	0.1180	11.8148	0.4435	0.3195	1.7330
MCMC [110]	0.1561	0.1246	3517.1771	0.4334	0.3090	217.7982
GP [32]	0.1590	0.1300	1.2185	0.4744	0.3516	0.1642

Mean Square Error (RMSE) and the Mean Absolute Error (MAE) between the true underlying function $f(x)$ and the prediction. These measures are consistent with what is typically used in the stat-of-the-art. The computational time required for the calculation of the posterior distribution and the prediction is used as measures of performance. We used the GPStuff Toolbox (see [164]) for the the EP algorithm, LA, and the MCMC method. Their hyperparameters are found by calculating the maxima of the maximum a posteriori (MAP) distribution. The hyperparameter optimization time is not included in the computation times reported; since our hyperparameter optimization is heavily simplified and quick, this actually works in favor of the methods we compare against.

6.7.2 Two-Dimensional Function

We introduce a two dimensional data set with $N = 2000$ number of observations to highlight the performance improvement in computational time of the Meta-GP in comparison to the other techniques: $Y(x, y) = 0.4 \sin(x)^2 + \cos(y)^2 + 0.25x^2 + \epsilon$ with Student-t noise $\epsilon = St(\nu = 4, \sigma^2 = 1)$. Table 3 summarizes the results of the proposed Meta-GP algorithm in comparison with competing techniques. For this example we achieved an increase in accuracy of 5% with the Meta-GP algorithm when compared against LA and EP. This can be attributed to the fact that the Meta-GP predicts without neglecting higher order terms of the posterior distribution as opposed to the LA and EP algorithms. Yet, the Meta-GP algorithm required significantly less computational time than EP, LA and MCMC. Meta-GP needs 16 times less time compared to the second fastest algorithm of the non-Gaussian inference methods, i.e. EP. Due to its iterative nature, EP is more than 10 times slower on this larger dataset. On the other hand, LA needs to find the mode numerically, and inverts a high ranked variance matrix. On this larger data set, LA requires over 60 sec, while Meta-GP required less than 1 sec. The MCMC technique suffers significantly in this example due its reliance on sampling, and the larger dataset provoking a high dimensional posterior distribution.

6.7.3 Boston Housing Data

The Boston Housing data [63] is a popular regression problem widely used for benchmarking GP inference with non-Gaussian likelihoods (e.g. [86, 163]). The goal is to predict the median house-prices in different parts of the Boston area based on 13 input variables and 506 outputs. The data set is normalized as suggested in [109, 125]. A 10-fold cross-validation is conducted to assess the predicted functions. The student-t probability is used as non-Gaussian likelihood. The Meta-GP algorithm achieves a Root Mean Square Error (RMSE) of 0.3854. That is a better accuracy in the solution than the best performing state-of-the-art method, i.e. LA, which achieved a RMSE of 0.4166. Further, the computation time of the Meta-GP is 0.9018 [sec], what is 50 times shorter than LA which is the fastest competing technique.

6.7.4 Multimodal Likelihood

The posterior distribution when the prior is Gaussian and the non-Gaussian likelihood is “similar” to a Gaussian likelihood (e.g. Student-t likelihood) is in most cases a unimodal distribution [163]. However, if the posterior distribution is multimodal, a single Gaussian is an insufficiently accurate representation of the original distribution. Consequently, if the likelihood is multimodal, posterior approximation methods with Gaussian distributions such as EP, VA, or LA might not lead to good results, whereas Meta-GP should do better. To corroborate this, we consider a dataset with multimodal likelihood such that a multimodal posterior distribution is most likely. The multimodal likelihood chosen is a weighted sum of two Gaussian distributions: $p(y_i|f_i) = \pi_1 N(y_i|\mu_1, \sigma_1^2) + \pi_2 N(y_i|\mu_2, \sigma_2^2)$ with the parameters $\pi_1 = 0.75, \mu_1 = -1, \sigma_1^2 = 1$ and $\pi_2 = 0.25, \mu_2 = 10, \sigma_2^2 = 2$. The training set D consists of $N = 800$ samples. We sampled from each Gaussian distribution separately and determined the weights by the number of samples in proportion to each other. After mixing both samples, we added the resulting noise to the function: $f(x) = 5 \sin(2x) + 0.5x^2 + 2$. The results in Figure 46 show that predictions with Meta-GP are accurate even in the presence of multi-modal likelihoods, whereas predictions of LA are biased from the underlying function $f(x)$ (see Figures 46c and 46d). This is since in this case LA inaccurately approximates the posterior with unimodal Gaussian distribution. This results in the 6 times higher RMSE and MAE with LA when compared with Meta-GP.

6.8 Classification Results

In this section, we evaluate Meta-GP on well-known classification benchmark datasets: USPS and MNIST handwritten digits, and CIFAR 10 containing pictures from the Internet. In all of the presented results, we enforce only a single-pass through the data for Meta-GP, that is, unlike the Deep Neural Network (DNN) results we compare to in the supplementary, no data augmentation or modification of the images or multiple passes through the data are required. For evaluation, the test error and the required time for training the model and predicting over the whole test set are presented. The error rate for binary classification is found by computing the predictive probability from eq. 124, thresholding at 1/2 as in [131] to get hard class label predictions. For multiclass classification we conduct binary classification of each single class separately. The class with the highest predictive probability is selected as the predicted class label according to the one vs. all principle [136].

6.8.1 USPS Data Set

We apply Meta-GP to the USPS Data Set [75] which consists of black and white 16x16 pixels images of handwritten digits. The training data consists of 4649 images and the test set has 4649 images. In the multiclass classification setting, the Meta-GP achieves an error rate of 2.37 %. This is better than the LA classification error rate of 3.1% [131], and 4.9% error rate with Csato’s online Gaussian Process algorithm [32]. Recently, [159] achieved an error rate of 4.73% using the Gaussian process latent variable model with variational inference. In the binary classification setting, Meta-GP achieved highly promising results where nine of the 10 classes reached an error rate less than one percent with a single pass through the data (Table 4). Each binary classification required in average 264 [sec] achieving the presented error rate with 1000 G-RBF kernels with a bandwidth of 4.4.

6.8.2 MNIST Data Set

We applied Meta-GP also to the popular MNIST image data sets [89]. The MNIST data set contains black and white images of handwritten digits, but with a image size of 28x28 pixels. Compared to the USPS data set, MNIST data set contains more images with a training set of 60000 images and a test set of 10000 images. Table 4 presents Meta-GP performance on MNIST.

Table 4: Meta-GP Single-Pass Classification for USPS, MNIST and Cifar-10 Data Set.

CLASS	USPS	MNIST	Cifar-10		
1	0.45 %	0.30 %	CLASS 1	CLASS 2	ACCURACY
2	0.22 %	0.65 %	CAT	CAR	87.40
3	0.52 %	0.73 %	CAT	BIRD	81.05
4	0.62 %	0.78 %	CAT	HORSE	81.05
5	0.65 %	0.59 %	CAT	SHIP	89.60
6	0.58 %	0.44 %	CAR	DEER	91.00
7	0.39 %	0.90 %	CAR	DOG	90.20
8	0.30 %	0.93 %	CAR	TRUCK	75.45
9	0.62 %	1.32 %	CAR	AIRPLANE	87.75
0	0.54 %	0.22 %			
MULTICLASS	2.37 %	2.64 %			

Similar to the USPS data set, the binary classification for each class achieves an error less than 1 percent, except for one class. *That is, the Meta-GP classifies 9,900 out of 10,000 pictures correct with only a single pass and no data augmentation with only 50 G-RBF kernels with a stationary bandwidth 4.4.* The meta-GP achieves an accuracy of over 92.32 % while requiring only 334.5 [sec] for training and prediction of the multiclass classification problem. Note that these results do not use hyperparameter optimization, inclusion of which would further improve results. If higher accuracy is desired, the number of basis vector entries can be increased leading to an error rate of 2.64 % (accuracy of 97.36%) with 1000 G-RBFs. Considering the approximation due to kernel sparseness, this result is comparable to [67]. They achieved an error rate of 1.96 % using MCMC. The variational inference method from [66] reached an error rate of 2.2 %, where a binary problem, not multiclass classification as presented here, was considered between the odd and even digits.

6.8.3 CIFAR-10 Data Set

The CIFAR-10 dataset consists of 60000 32x32 pixel color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images [83]. The images of the CIFAR-10 data set comprise different views and background and the class objectives have different colors within their own class. The input vector $\mathcal{X}$ is high-dimensional with 3072 entries. Table 4 shows the results for binary classification between two chosen classes (Cat and Car). Therefore we removed all other classes from the training and test set. Thus, the training and the prediction was conducted by using the remaining 10000 images of the training set and the 2000 images of the test set. The error rate in Table 4 shows the capability of Meta-GP to handle this high-dimensional data set. For instance, the algorithm is able to distinguish between cars and cats with an accuracy of 87.40 %. It is important to note that the shown results are achieved using only a basis set with 1000 kernels with one-dimensional bandwidth (15) in a G-RBF kernel, and with strictly a single pass through the data. Note further that compared to the dimension of the input vector the training set is quite small. For most purposes, our results are more than acceptable. The performance will likely increase with better kernel allocation [165, 149, 150], larger or augmented training set, or better hyperparameter optimization algorithms (e.g. [9, 151]), however, this is not pursued here since the point of the work is to demonstrate the utility of the vanilla version of Meta-GP.

Part IV

Progress in the Year 2015-16

1 Nonparametric Kernel Observer Model and Inference Algorithms for Spatiotemporally Evolving Systems

Our aim is to speed-up predictive model learning by utilizing past experience to quickly uncover hidden patterns, while ensuring that the model can adapt to unforeseen events. Data on highly complex and stochastic spatial variables such as enemy concentration, or enemy movements is usually available at different intervals of time. Kernel observer framework described below utilizes kernel methods to model data at each instant of time, and the corresponding evolution in time is captured via linear systems model in the functional space thereby utilizing concepts of Observability. The linear systems model is effective in capturing the hidden patterns whereas the observer ensures that the model adapts to any unforeseen events.

Moreover, finding spatial locations of sensor deployment is critical to reliably estimating and monitoring such spatiotemporal systems. Sensor deployment here is synonymous with reconnaissance operations using drones. Proposed kernel observer model takes into account the spatiotemporal dynamics of the underlying phenomena for sensor placement. Deterministic approach for sensor selection is described over sections [1.4](#) [1.6](#). Computationally cheaper approach of random sensor placement in section [1.2](#) discusses theoretical guarantees on the number of such sensors required.

1.1 Kernel Observer

We focus on predictive inference of a time-varying stochastic process, whose mean f evolves as $f_{\tau+1} \sim \mathbb{F}(f_{\tau}, \eta_{\tau})$, where $\mathbb{F}$ is a distribution varying with time τ and exogenous inputs η . Our approach builds on the fact that in several cases, temporal evolution can be hierarchically separated from spatial functional evolution, the prototypical example of which is the *abstract evolution equation*, i.e. ODEs on Banach spaces [\[20\]](#). To make this approach computationally realizable, we restrict the sequence f_{τ} to lie in a reproducing kernel Hilbert space (RKHS) [\[133\]](#). If $k : \Omega \times \Omega \rightarrow \mathbb{R}$ is a positive-definite Mercer kernel on a domain Ω , this implies the existence of a smooth map $\psi : \Omega \rightarrow \mathcal{H}$, where $\mathcal{H}$ is an RKHS with the property $k(x, y) = \langle \psi(x), \psi(y) \rangle_{\mathcal{H}}$. The proposed model assumes spatiotemporal evolution in the input domain corresponds to temporal evolution of the mixing weights of a kernel model alone in the functional domain: this separation allows us to utilize powerful ideas from systems theory for deriving necessary and sufficient conditions for spatiotemporal monitoring. We restrict our attention to the class of functional evolutions $\mathbb{F}$ defined by linear Markovian transitions in an RKHS. In this case, the method is functionally equivalent to a kernel Kalman filter (KKF), with the main contributions being our approximate formulation and the minimization of the number of measurements required to infer the latent state. Both nonstationary kernel methods and KKF's can be derived using Bayesian inference: the main difference is that nonstationary kernel methods are typically truly nonparametric methods in that the time indices τ are part of the observations, and thus the kernel model must retain them for training. On the other hand, KKF's decouple the time and space variables, and thus can learn long-term patterns without including time in the kernel function, at the cost of missing subtler local time-dependent correlations [\[30\]](#).

The proposed model assumes that the spatiotemporal evolution in the input domain corresponds to temporal evolution of the mixing weights of a kernel model alone in the functional domain: this separation allows us to utilize powerful ideas from systems theory for deriving nec-

essary and sufficient conditions for spatiotemporal monitoring. We focus on predictive inference of a time-varying stochastic process, whose mean f evolves as $f_{\tau+1} \sim \mathbb{F}(f_\tau, \eta_\tau)$, where $\mathbb{F}$ is a distribution varying with time τ and exogenous inputs η . Our approach builds on the fact that in several cases, temporal evolution can be hierarchically separated from spatial functional evolution, the prototypical example of which is the *abstract evolution equation*, i.e. ODEs on Banach spaces [20]. To make this approach computationally realizable, we restrict the sequence f_τ to lie in a reproducing kernel Hilbert space (RKHS) [133]. If $k : \Omega \times \Omega \rightarrow \mathbb{R}$ is a positive-definite Mercer kernel on a domain Ω , this implies the existence of a smooth map $\psi : \Omega \rightarrow \mathcal{H}$, where $\mathcal{H}$ is an RKHS with the property $k(x, y) = \langle \psi(x), \psi(y) \rangle_{\mathcal{H}}$.

Let $y \in \mathbb{R}^N$ be the measurements of the function available from N **sensors**, $\mathcal{A} : \mathcal{H} \rightarrow \mathcal{H}$ be a linear transition operator in the RKHS $\mathcal{H}$, and $\mathcal{K} : \mathcal{H} \rightarrow \mathbb{R}^N$ be a linear measurement operator. The model for the functional evolution and measurement is:

$$f_{\tau+1} = \mathcal{A}f_\tau + \eta_\tau, \quad y_\tau = \mathcal{K}f_\tau + \zeta_\tau, \quad (125)$$

where η_τ is a zero-mean stochastic process in $\mathcal{H}$, and ζ_τ is a Wiener process in $\mathbb{R}^N$. To avoid working in dual space and have the parameters grow with the data, we work with an approximate feature map $\hat{\psi}(x) := [\hat{\psi}_1(x) \dots \hat{\psi}_M(x)]$, with **basis functions** $\hat{\psi}_j(x)$ to an approximate feature space $\hat{\mathcal{H}}$, with the property that for every element $f \in \mathcal{H}$, $\exists \hat{f} \in \hat{\mathcal{H}}$ and an $\epsilon > 0$ s.t. $\|f - \hat{f}\|_{\mathcal{H}} < \epsilon$. Some examples of approximate feature maps are:

- **(Dictionary of atoms)** Let Ω be compact. Given points $\mathcal{C} = \{c_1, \dots, c_M\}$, $c_i \in \Omega$, we have a dictionary of atoms $\Psi = \{\psi(c_1), \dots, \psi(c_M)\}$, $\psi(c_i) \in \mathcal{H}$, the span of which is a strict subspace $\hat{\mathcal{H}}$ of the RKHS $\mathcal{H}$ generated by the kernel. Here,

$$\hat{\psi}_i(x) := \langle \psi(x), \psi(c_i) \rangle_{\mathcal{H}} = k(x, c_i). \quad (126)$$

- **(Random Fourier features)** Let $\Omega \subset \mathbb{R}^n$ be compact, and let $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$ be the Gaussian RBF kernel. Then random Fourier features approximate the kernel feature map as $\hat{\psi}_\omega : \Omega \rightarrow \hat{\mathcal{H}}$, where ω is a sample from the Fourier transform of $k(x, y)$, with the property that $k(x, y) = \mathbb{E}_\omega[\langle \hat{\psi}_\omega(x), \hat{\psi}_\omega(y) \rangle_{\hat{\mathcal{H}}}]$ [126]. In this case, if $V \in \mathbb{R}^{M/2 \times n}$ is a random matrix representing the sample ω , then

$$\hat{\psi}_i(x) := \left[\frac{1}{\sqrt{M}} \sin([Vx]_i), \frac{1}{\sqrt{M}} \cos([Vx]_i) \right]. \quad (127)$$

Other examples include FastFood [88] and A la Carte [170]. In the approximate space case, we replace the transition operator $\mathcal{A} : \mathcal{H} \rightarrow \mathcal{H}$ in [125] by $\hat{\mathcal{A}} : \hat{\mathcal{H}} \rightarrow \hat{\mathcal{H}}$. The finite-dimensional evolution equations approximating [125] in approximate dual form are

$$w_{\tau+1} = \hat{A}w_\tau + \eta_\tau, \quad y_\tau = Kw_\tau + \zeta_\tau, \quad (128)$$

where we have matrices $\hat{A} \in \mathbb{R}^{M \times M}$, $K \in \mathbb{R}^{N \times M}$, vectors $w_\tau \in \mathbb{R}^M$, and where we have slightly abused notation to let y_τ, η_τ and ζ_τ denote their $\hat{\mathcal{H}}$ counterparts. Here K is the matrix whose rows are of the form $K_{(i)} = [\hat{\psi}_1(x_i) \hat{\psi}_2(x_i) \dots \hat{\psi}_M(x_i)]$. In systems-theoretic language, the matrix acts as a measurement operator. To the best of our knowledge, this approximate formulation of the kernel Kalman filter is new. We define the *observability matrix* as $\mathcal{O}_\Upsilon = [(K\hat{A}^{\tau_1})^T \dots (K\hat{A}^{\tau_L})^T]^T$ where $\Upsilon = \{\tau_1, \dots, \tau_L\}$ are the set of instances τ_i when we apply the operators K_{τ_i} . A linear system is said to be *observable* if $\text{rank } \mathcal{O}_\Upsilon = M$ for $\Upsilon = \{0, 1, \dots, M-1\}$ [173]. Observability guarantees two critical facts:

- The state w_0 can be recovered exactly from a finite series of measurements $\{y_{\tau_1}, \dots, y_{\tau_L}\}$; in particular, defining $y_{\Upsilon} = [y_{\tau_1}^T, \dots, y_{\tau_L}^T]^T$, we have that $y_{\Upsilon} = \mathcal{O}_{\Upsilon} w_0$.
- A feedback-based *observer* can be designed such that the estimate of w_{τ} , denoted by $\hat{w}_{\tau}$, converges exponentially fast to w_{τ} .

Formally the spatiotemporal modeling and inference problem considered is stated as: given a spatiotemporally evolving system modeled using (128), choose a set of N sensing locations such that even with $N \ll M$, the functional evolution of the spatiotemporal model can be estimated (which corresponds to *monitoring*) and can be predicted robustly. Our approach to solve this problem relies on the design of the measurement operator K so that the pair $(K, \hat{A})$ is observable: any Bayesian state estimator (e.g. a Kalman filter) utilizing this pair is denoted as a **kernel observer**. For the design of K i.e. choosing the number of sensing locations N , we leverage the most important theorem in Linear Algebra, the Rational Canonical decomposition of $\hat{A}$.

1.2 Preliminaries on Rational Canonical Structures

We take a geometric approach towards the choice of sampling locations for inferring w_{τ} in (128). We use the notation $\mathcal{V}$, with $\dim(\mathcal{V}) = M$, to emphasize the fact that these theorems hold for any finite-dimensional vector space. Consider the linear operator $\mathcal{A} : \mathcal{V} \rightarrow \mathcal{V}$, and recall that the definition of observability requires the construction of a linear operator $\mathcal{K} : \mathcal{V} \rightarrow \mathcal{U}$, with $\dim(\mathcal{U}) = N$, such that $\text{rank}[(\mathcal{K})^T \dots (\mathcal{K}\mathcal{A}^{M-1})^T]^T = M$. In most applications, if $N \geq M$, and $\text{rank } \mathcal{K} = N$, it is reasonable to expect that observability may be achieved. However, for our purposes, N must be *significantly* less than M . Therefore, we must design $\mathcal{K}$ with as small a rank as possible. To do so, we require a series of vectors v_i that, under repeated iterations of $\mathcal{A}$, can generate a basis for $\mathcal{V}$. For this task, we will use a fundamental decomposition result from the theory of modules, known as the *rational canonical structure* of $\mathcal{A}$ [169]. The intuition here is that if the sequence $\{v_i\}_i$ can generate this basis, it can be directly used to construct $\mathcal{K}$.

1.3 Design of Measurement Operator K using Rational Canonical Structure theorem

We take a geometric approach towards the choice of sampling locations N . We consider finite-dimensional vector space $\mathcal{V}$, with $\dim(\mathcal{V}) = M$, such that the linear operator $\hat{A} : \mathcal{V} \rightarrow \mathcal{V}$, and recall that the definition of observability requires the construction of a linear operator $K : \mathcal{V} \rightarrow \mathcal{U}$, with $\dim(\mathcal{U}) = N$, such that $\text{rank}[(K)^T \dots (K\hat{A}^{M-1})^T]^T = M$. The goal is to design K with as small number of measurements N as possible, and we have theoretically shown that the lower bound on N can be obtained using Rational Canonical structure theorem. According to this theorem, the structure of an operator $\hat{A}$ decomposes the vector space $\mathcal{V}$ into ℓ -invariant subspaces. Thus a vector v that belongs to a particular subspace, continues to remain in that subspace even after any number of linear operations $\hat{A}v$. In essence, any measurement obtained from a given subspace cannot provide information about the evolution that occurs in other subspaces, thus the lower bound on the number of measurements to construct K is ℓ which is also known as the cyclic index of $\hat{A}$.

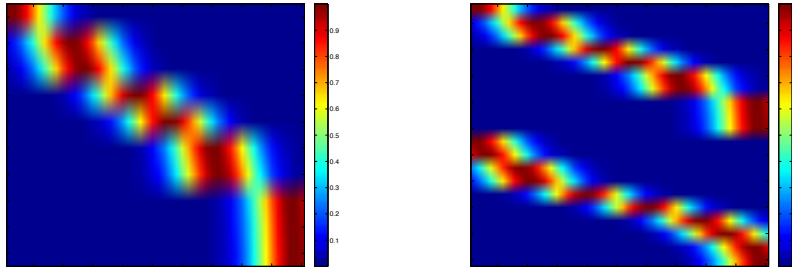
The linear operator $\mathcal{A} : \mathcal{V} \rightarrow \mathcal{V}$ has a characteristic polynomial $\pi(\lambda)$ such that $\pi(\mathcal{A}) = 0$ by the Cayley-Hamilton theorem. The minimal polynomial (MP) of $\mathcal{A}$ is the monic polynomial $\alpha(\cdot)$ of least degree (denoted by $\deg(\cdot)$) given as $\alpha(\lambda) = a_0 + a_1\lambda + \dots + \lambda^{\deg(\alpha)} = 0$, such that $\alpha(\mathcal{A}) = a_0I + a_1\mathcal{A} + \dots + \mathcal{A}^{\deg(\alpha)} = 0$. The MP is unique and divides $\pi(\lambda)$, so that $\deg(\alpha) \leq \deg(\pi)$. The MP of a vector $v \in \mathcal{V}$ relative to $\mathcal{A}$ is the unique monic polynomial ξ_v of least degree such that $\xi_v(\mathcal{A})v = a_0v + a_1\mathcal{A}v + \dots + \mathcal{A}^{\deg(\alpha)}v = 0$. If $\deg(\alpha) = M$, then $\mathcal{A}$ is

cyclic and $\exists v \in \mathcal{V}$, such that the vectors $\{v, \mathcal{A}v, \dots, \mathcal{A}^{M-1}v\}$ form a basis for $\mathcal{V}$; this is the same as saying that the pair $(v^T, \mathcal{A}^T)$ is observable. A subspace $\mathcal{V}_S \subset \mathcal{V}$ s.t. $\mathcal{A}\mathcal{V}_S \subset \mathcal{V}_S$ is *$\mathcal{A}$ -cyclic* if $\mathcal{A}|_{\mathcal{V}_S}$, the restriction of $\mathcal{A}$ to the subspace $\mathcal{V}_S$, is cyclic. If $\alpha(\lambda)$ is the minimal polynomial of $\mathcal{A}$ and $\deg(\alpha) = m < M$, $\exists v \in \mathcal{V}$ such that $\{v, \mathcal{A}v, \dots, \mathcal{A}^{m-1}v\}$ span an m -dimensional $\mathcal{A}$ -cyclic subspace $\mathcal{V}_S$, with v being the *cyclic generator* of $\mathcal{V}_S$. The subspace $\mathcal{V}_S$ decomposes $\mathcal{V}$ relative to $\mathcal{A}$. By the rational canonical structure theorem, $\mathcal{A}$ can be successively decomposed into subspaces $\mathcal{V}_i \subset \mathcal{V}$, $i \in \{1, \dots, \ell\}$, s.t. $\mathcal{V} = \mathcal{V}_1 \oplus \dots \oplus \mathcal{V}_\ell$, $\mathcal{A}\mathcal{V}_i \subset \mathcal{V}_i$, and $\mathcal{A}|_{\mathcal{V}_i}, i \in \{1, \dots, \ell\}$, are cyclic.¹ The integer ℓ is unique and is called the *cyclic index* of $\mathcal{A}$.

One of our main results is to show that the cyclic index is a lower bound on the number of measurements required to reconstruct w_τ (see Prop. 2 and Alg. 17). The matrix transform associated to this theorem is known as the *Frobenius normal form* (denoted by $C \in \mathbb{R}^{M \times M}$): for $\hat{A} \in \mathbb{R}^{M \times M}$, $\exists Q \in \mathbb{R}^{M \times M}$ invertible such that $\hat{A} = QCQ^{-1}$.

1.4 Main Results

In this section, we prove results concerning the observability of spatiotemporally varying functions modeled by the functional evolution and measurement equations (128) formulated in Section 1.1. In particular, observability of the system states implies that we can recover the current state of the spatiotemporally varying function using a small number of sampling locations N , which allows us to 1) track the function, and 2) predict its evolution forward in time. We work with the approximation $\hat{\mathcal{H}} \approx \mathcal{H}$: given M basis functions, this implies that the dual space of $\hat{\mathcal{H}}$ is $\mathbb{R}^M$. Definition 1.1 defines a condition we call *shadedness*, which is a measure of whether information from the sensor locations is propagating to the basis functions $\hat{\psi}_j(x)$ constituting $\hat{\mathcal{H}}$. Proposition 1 states a lower bound on the number of sampling locations required for observability which holds for a given $\hat{A}$. Proposition 2 constructively shows the existence of an abstract measurement map $\hat{K}$ achieving this lower bound. Finally, since the measurement map does not have the structure of a kernel matrix, a slightly weaker sufficient condition for the observability of any $\hat{A}$ is given in Theorem 1.3 using shadedness. Proofs of all claims are in the supplementary material.



(a) 1-shaded (Def. 1.1) (b) 2-shaded (Eq. 129)

Figure 47: Shaded observation matrices (dictionary of atoms).

Definition 1.1. (Shaded Observation Matrix) Let $\mathcal{H}$ be generated from $k : \Omega \times \Omega \rightarrow \mathbb{R}$ positive-definite on a domain Ω , let $\{\hat{\psi}_1(x), \dots, \hat{\psi}_M(x)\}$ be the set of bases generating an approximate feature map $\hat{\psi} : \Omega \rightarrow \hat{\mathcal{H}} \approx \mathcal{H}$, and let $\mathcal{X} = \{x_1, \dots, x_N\}$, $x_i \in \Omega$. Let $K \in \mathbb{R}^{N \times M}$ be the observation matrix, where $K_{ij} := \hat{\psi}_j(x_i)$. For each row $K_{(i)} := [\hat{\psi}_1(x_i) \dots \hat{\psi}_M(x_i)]$, define the

¹In general, the subspaces $\mathcal{V}_i$ are not unique for a fixed $\mathcal{A}$.

set $\mathcal{I}_{(i)} := \{\iota_1^{(i)}, \iota_2^{(i)}, \dots, \iota_{M_i}^{(i)}\}$ to be the indices in the observation matrix row i which are nonzero: in this case, defining the basis subset as $\Psi^{(i)} := \{\psi_{\iota_1^{(i)}}(x), \dots, \psi_{\iota_{M_i}^{(i)}}(x)\}$, we say observation matrix row $K_{(i)}$ is **shaded with respect to** $\Psi^{(i)}$. Then if $\bigcup_{i \in \{1, \dots, N\}} \mathcal{I}^{(i)} = \{1, 2, \dots, M\}$, we denote K as a **shaded observation matrix** (see Figure 47a).

This definition seems quite abstract, so the following remark considers a more concrete example.

Remark 1.2. Let $\hat{\psi}$ be generated by the dictionary given by $\mathcal{C} = \{c_1, \dots, c_M\}$, $c_i \in \Omega$. Note that since $\hat{\psi}_j(x_i) = \langle \psi(x_i), \psi(c_j) \rangle_{\mathcal{H}} = k(x_i, c_j)$, K is the kernel matrix between $\mathcal{X}$ and $\mathcal{C}$. For the kernel matrix to be shaded thus implies that there does not exist an atom $\psi(c_j)$ such that the projections $\langle \psi(x_i), \psi(c_j) \rangle_{\mathcal{H}}$ vanish for all x_i , $1 \leq i \leq N$. Intuitively, the shadedness property requires that the sensor locations x_i are privy to information propagating from every c_j . As an example, note that, in principle, for the Gaussian kernel, a single row generates a shaded kernel matrix².

Algorithm 17 Measurement Map $\tilde{K}$

Input: $\hat{A} \in \mathbb{R}^{M \times M}$
 Compute Frobenius canonical form, such that $C = Q^{-1} \hat{A}^T Q$. Set $C_0 := C$, and $M_0 := M$.
for $i = 1$ **to** ℓ **do**
 Obtain MP $\alpha_i(\lambda)$ of C_{i-1} . This returns associated indices $\mathcal{J}^{(i)} \subset \{1, 2, \dots, M_{i-1}\}$.
 Construct vector $v_i \in \mathbb{R}^M$ such that $\xi_{v_i}(\lambda) = \alpha_i(\lambda)$.
 Use indices $\{1, 2, \dots, M_{i-1}\} \setminus \mathcal{J}^{(i)}$ to select matrix C_i . Set $M_i := |\{1, 2, \dots, M_{i-1}\} \setminus \mathcal{J}^{(i)}|$
end for
 Compute $\tilde{K} = [v_1, v_2, \dots, v_\ell]^T$
Output: $\tilde{K} = \tilde{K} Q^{-1}$

We first prove a lower bound on the number of sampling locations required for observability.

Proposition 1. Let $\mathcal{H}$ be generated from $k : \Omega \times \Omega \rightarrow \mathbb{R}$ positive-definite on a domain Ω , let $\{\hat{\psi}_1(x), \dots, \hat{\psi}_M(x)\}$ be the set of bases generating an approximate feature map $\hat{\psi} : \Omega \rightarrow \hat{\mathcal{H}} \approx \mathcal{H}$, and let $\mathcal{X} = \{x_1, \dots, x_N\}$, $x_i \in \Omega$. Consider the discrete linear system on $\hat{\mathcal{H}}$ given by the evolution and measurement equations (128). Denote the Jordan decomposition of $\hat{A} \in \mathbb{R}^{M \times M}$ by $\hat{A} = P \Lambda P^{-1}$, where $\Lambda = [\Lambda_1 \dots \Lambda_O]$ may have repeated eigenvalues (i.e. $\exists \Lambda_i$ and Λ_j s.t. $\lambda_i = \lambda_j$). Then there exist kernels $k(x, y)$ such that the lower bound ℓ on the number of sampling locations N is given by the cyclic index of $\hat{A}$.

We now show how to construct a matrix $\tilde{K}$ corresponding to the lower bound ℓ .

Proposition 2. Given the conditions stated in Proposition 1, it is possible to construct a measurement map $\tilde{K} \in \mathbb{R}^{\ell \times M}$ for the system given by (128), such that the pair $(\tilde{K}, \hat{A})$ is observable.

The construction provided in the proof of Proposition 2 is formalized as Algorithm 17, which uses the rational canonical structure of $\hat{A}$ to generate a series of vectors $v_i \in \mathbb{R}^M$, whose iterations $\{v_1, \dots, \hat{A}^{m_1-1} v_1, \dots, v_\ell, \dots, \hat{A}^{m_\ell-1} v_\ell\}$ generate a basis for $\mathbb{R}^M$ (see Section 1.3) in $\mathcal{O}(\ell M^3)$ time [153]. Unfortunately, the measurement map $\tilde{K}$, being an abstract construction unrelated to the kernel, does not directly select $\mathcal{X}$. We will show how to use the measurement map to guide a search for $\mathcal{X}$ in Remark 1.4. For now, we state a sufficient condition for observability of a general system.

²However, in this case, the matrix can have many entries that are extremely close to zero, and can be very ill-conditioned.

Theorem 1.3. Suppose that the conditions in Proposition 1 hold, and let ℓ be the cyclic index of $\hat{A}$. Define

$$\mathbf{K} = [K^{(1)^T} \dots K^{(\ell)^T}]^T \quad (129)$$

as the ℓ -shaded matrix which consists of ℓ shaded matrices with the property that any subset of ℓ columns in $\mathbf{K}$ are linearly independent from each other. Then system (128) is observable if Υ has distinct values, and $|\Upsilon| \geq M$.

Algorithm 18 Sampling locations set $\mathcal{X}$

Input: $\hat{A} = C$, lower bound ℓ
 Decompose C to generate invariant subspaces $\hat{\mathcal{H}}_j$, $j \in \{1, 2, \dots, \ell\}$ (see section 1.3)
for $j = 1$ **to** ℓ **do**
 Obtain centers $\mathcal{C}^{(j)}$ w.r.t subspace $\hat{\mathcal{H}}_j$,
 Generate samples $x_i^{(j)}$ to create a kernel matrix $K^{(j)}$ that is shaded only with respect to centers $\mathcal{C}^{(j)}$
end for
Output: Sampling locations set $\mathcal{X} = \{x^{(1)}, x^{(2)} \dots, x^{(\ell)}\}$.

While Theorem 1.3 is a quite general result, the condition that any ℓ columns of $\mathbf{K}$ be linearly independent is very stringent. One scenario where this condition can be met with few measurements is when the feature map $\hat{\psi}(x)$ is generated by a dictionary of atoms with the Gaussian kernel evaluated at sampling locations $\{x_1, \dots, x_N\}$ according to (126), where $x_i \in \Omega \subset \mathbb{R}^d$, and x_i are sampled from a non-degenerate probability distribution on Ω such as the uniform distribution. For a semi-deterministic approach, when the dynamics matrix $\hat{A}$ is block-diagonal, we can utilize a simple heuristic:

Remark 1.4. Let Ω be compact, $\mathcal{C} = \{c_1, \dots, c_M\}$, $c_i \in \Omega$, and let the approximate feature map be defined by (126). Consider the system (128) with $\hat{A} = \Lambda$, and let $\Upsilon = \{0, 1, \dots, M-1\}$. Then the measurement map $\tilde{K}$'s values lie in $\{0, 1\}$; in particular, each row $\tilde{K}^{(i)}$, $i \in \{1, \dots, \ell\}$, corresponds to a basis subset $\Psi^{(i)}$, generated by a subset of centers $\mathcal{C}^{(i)} \subset \mathcal{C}$. Generate samples $x_j^{(i)}$ to create a kernel matrix $K^{(i)}$ that is shaded only with respect to $\Psi^{(i)}$. Once this is done, move on to the next subset of bases corresponding to $\Psi^{(i+1)}$. When all ℓ rows of $\tilde{K}$ are accounted for, construct the matrix $\mathbf{K}$ as in (129). Then the resulting system $(\mathbf{K}, \hat{A})$ is observable. This heuristic is formalized in Algorithm 18.

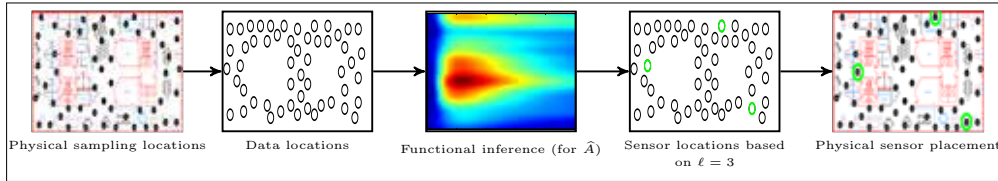


Figure 48: Overall description of how the kernel observer fits in the sensing framework. Physical locations are mapped to data locations, over which historical data is collected as a time series. Functional inference is performed over $\hat{\mathcal{H}}$ to solve for $\hat{A}$. The measurement operator K is then computed (see Figure 49), leading to sensor placement.

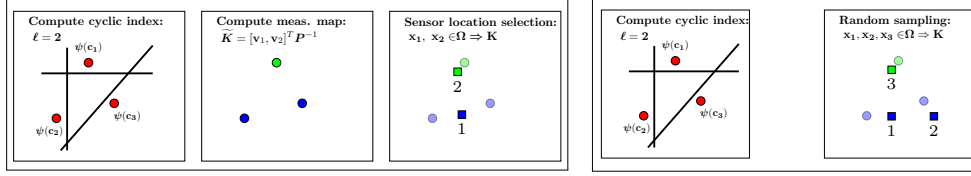


Figure 49: Diagram demonstrating sensor placement using the measurement map or random sampling approaches. The circles represent data locations associated to bases (e.g. $c_j \Leftrightarrow \psi(c_j)$) and the squares represent sensor locations (e.g. $x_i \Leftrightarrow \psi(x_i)$). The cyclic index ($\ell = 2$) indicates how many possible couplings of bases exist, which can be represented as a choice of $\binom{M}{\ell}$ hyperplanes in Ω . If the measurement map is computed (left), the correct couplings are chosen (green vs. blue), and a smaller number of sensors (2) can be placed. Alternatively, random sampling (right) is more computationally efficient, but generally requires more sensors (3).

1.5 Insights into Theoretical Results

The systems-theoretic approach reveals something rather surprising: functions with complex dynamics (with a small cyclic index) can be recovered with less sensor placements than functions with simpler dynamics. Although seemingly counterintuitive, it becomes clear that this is because complex dynamics, which are characterized by a lower geometric multiplicity of the eigenvalues, ensure that the orbit $\Theta := \{\hat{A}w_\tau\}_{\tau \in \mathcal{T}}$ traverses a greater portion of $\mathbb{R}^M \equiv \mathcal{H}$ and thus that fewer sensors can recover more geometric information. On the other hand, in ‘simpler’ functional evolution, Θ evolves along strict subspaces of $\mathbb{R}^M$, and so more independent sensors are required to infer the same amount of information.

In the case described in Remark 1.4 we have a set of centers $\mathcal{C} = \{c_1, \dots, c_M\}$, which generate the bases $\Psi = \{\psi(c_1), \dots, \psi(c_M)\}$. Let the cyclic index be ℓ : this implies that there exist ℓ subsets $\Psi^{(i)}$ of Ψ with at least one element $\psi(c_j)$ each, leading to $\binom{M}{\ell}$ possible choices: Figure 49 represents these choices as hyperplanes separating the subsets. The measurement map described in Alg. 17 induces this *decomposition of bases* $\Psi = \{\Psi^{(1)}, \dots, \Psi^{(\ell)}\}$ in polynomial time. Further, each subset $\Psi^{(i)}$ is directly associated to a subset of centers $\mathcal{C}^{(i)} \subset \mathcal{C}$, which allows us to pick targeted sensor locations $x_i \in \Omega$. In particular, for radially symmetric kernels such as the Gaussian, the centroid of the convex hull of $\mathcal{C}^{(i)}$ is sufficient for generating a sensor placement. The measurement map is a significant theoretical insight into sensor placement for dynamically changing environments, because it directly takes into account the dynamics of the process. Of course, in practice, this may be too expensive for approximate feature spaces with M very large, so one can use random sampling to generate the sensor locations instead, at the cost of N being larger than ℓ . The advantage here though is that since random sampling is computationally inexpensive, different choices of sensor placements can be generated and evaluated relatively quickly.

1.6 Experimental Results

1.6.1 Comparison With Nonstationary Kernel Methods

We compare two variants of the kernel observer on three real-world datasets (**Intel** Berkeley, **Irish** Wind, and **Ozone**) with two non-stationary kernel methods a) Process convolution with local smoothing kernel (PCLSK [54]), b) Latent expansion in Input space (LEIS [120]), and c) a baseline GP, which is a sparse GP model trained using all of the data available per time step. Recall from section 1.1 that the kernel observer is a Bayesian state estimator utilizing the

dynamics-measurement pair $(K, \hat{A})$. The first variant of the kernel observer, called *autonomous* and denoted by AKO, is feedforward only (i.e. K is not utilized), and is a measure of the modeling fidelity of [128]. The second variant, called *feedback*, and denoted by FKO, utilizes K . Model inference for the kernel observer involved three steps: 1) picking $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$, a search for the ideal σ is performed for a sparse Gaussian Process model (with a fixed basis center set $\mathcal{C}$ selected using the method in [31]); 2) having obtained σ , Gaussian process inference is used to generate weight vectors for each time-step in the training set, resulting in the sequence $w_\tau, \tau \in \{1, \dots, T\}$; 3) matrix least-squares is applied to this sequence to infer $\hat{A}$. In the prediction step, in AKO, $\hat{A}$ is used to propagate w_τ forward to make predictions with no feedback, and in FKO, a Kalman filter with $N \geq \ell$ is used to propagate w_τ forward to make predictions.

The Intel Berkeley dataset consists of temperature data from wireless sensors. The training data is taken from 00:20 hrs on March 6th 2004 at intervals of 20 minutes, and testing was performed over unseen data beginning 12:40 hrs of the same day. Out of 50 sensor locations, we uniformly selected 25 locations each for training and testing purposes. Figures 50a and 51a show the results of the timeseries prediction and RMS errors respectively. Here, the cyclic index of $\hat{A}$ was determined to be 2, so N was set to 2 for the kernel observer with feedback. Note that here, even AKO outperforms PCLSK and LEIS overall, and FKO with $N = 2$ significantly outperforms all other methods. The second dataset is the Irish wind dataset, which consists of daily average wind speed (in knots = 0.542 m/s) data collected from year 1961 to 1978 at 12 meteorological stations in the Republic of Ireland. Figures 50b and 51b show the results of the timeseries prediction and RMS errors respectively. Again, the cyclic index of $\hat{A}$ was determined to be 2. In this case, AKO’s performance is comparable to PCLSK and LEIS, while the kernel observer with feedback with $N = 2$ again outperforms all other methods. Finally, the Ozone dataset measures ozone concentration (in parts per billion) measured at 60 stations by the United States Environmental Protection Agency [93] across USA. Due to missing measurements, we only selected data from year 1997 to 2013 for training and evaluation. For each station, we averaged ozone concentration over a period of three months, resulting in four quarters per year. Out of 60 sensor locations, we uniformly selected 30 for training and the remaining locations for testing purposes. Figures 50c and 51c show the results of the timeseries prediction and RMS errors respectively. Here, the cyclic index of $\hat{A}$ was determined to be 1. In this case, AKO’s performance is comparable to PCLSK and LEIS, with FKO with feedback with $N = 1$ performing best. Overall, the kernel observer is an order of magnitude faster than its competitors: see Table 5 for timing results.

Table 5: Total training and prediction times for datasets in §3.2

	Intel Berkeley	Irish Wind	Ozone
<i>Data Size (M-T)</i>	25-72	12-36	30-68
Kernel Observer	2.1 sec	0.1 sec	1.61 sec
PCLSK	121.4 sec	7.0 sec	91.90 sec
LEIS	43.8 sec	2.8 sec	37.41 sec

1.6.2 Prediction of Global Ocean Surface Temperature

The approach was tested on a very large dataset from the National Oceanographic Data Center: with around 3-5 million measurements available per day across the globe. Moreover, the day and night temperature models trained on data from the year 2011, were used to monitor the temperatures for 2012. Note that due to the size of the dataset and the high computational

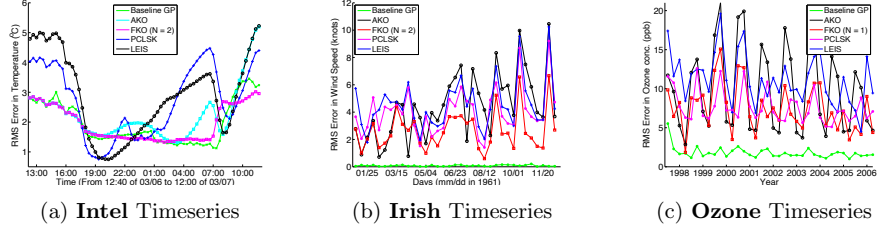


Figure 50: Timeseries comparison of kernel observer to PCLSK and LEIS methods

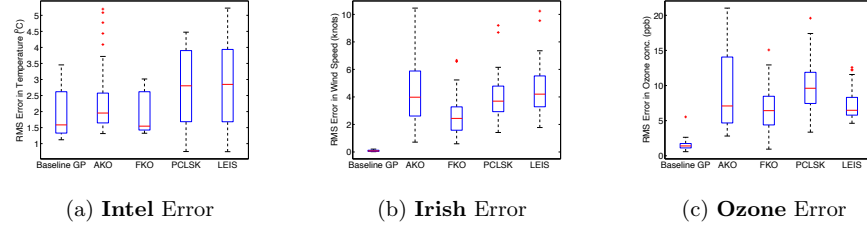


Figure 51: RMS error comparison of kernel observer to PCLSK and LEIS methods

requirements of the nonstationary kernel methods, a comparison with them was not pursued. To build the autonomous kernel observer and general kernel observer models, we followed the same procedure outlined in Section 1.6.1 but with $\mathcal{C} = \{c_1, \dots, c_M\}$, $c_j \in \mathbb{R}^2$, $|\mathcal{C}| = 300$. The Kalman filter for the general kernel observer model used $N \in \{250, 500, 1000\}$ at random locations to track the system state given a random initial condition w_0 . As a fair baseline, the observers are compared to training a sparse GP model (labeled ‘original’) on approximately 400,000 measurements per day. Figures 52b and 52c compare the autonomous and feedback approach with 1,000 samples to the baseline GP; here, it can be seen that the autonomous does well in the beginning, but then incurs an unacceptable amount of error when the time series goes into 2012, i.e. where the model has not seen any training data, whereas KO does well throughout. Figures 52d and 52e show a comparison of the RMS error of estimated values from the real data. This figure shows the trend of the observer getting better and better state estimates as a function of the number of sensing locations N ³. Time required for kernel observer is much lesser than retraining the model every time step, see figure 52f. Success in monitoring demonstrates two things: 1) the modeling process can capture spatiotemporal trends that generalize across years, and 2) the observer framework allows us to infer the state using a number of measurements that are an order of magnitude fewer than available.

³Note that we checked the performance of training a GP with only 1,000 samples as a control, but the average error was about 10 Kelvins, i.e. much worse than KO.

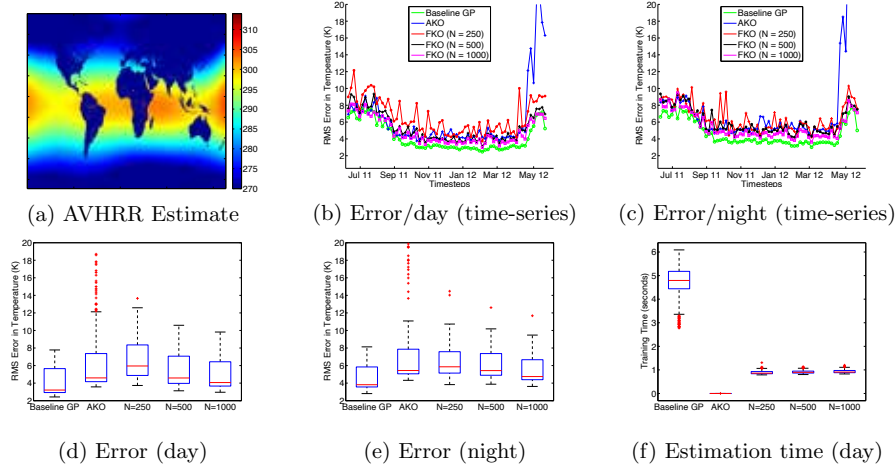


Figure 52: RMS error comparison of kernel observer to PCLSK and LEIS methods

2 Uninformed-to-Informed Exploration: When have you Learned Enough to Predict Informative Change?

Exploration is concerned with selecting actions or samples to gather data for learning, and is one of the most fundamental problems in machine learning. Correct decisions on which actions to take or samples to obtain for improving learning can significantly enhance the performance and sample-efficiency of a machine learning algorithm. More specifically, without optimal exploration, machine learning algorithms that operate in time-varying environments or on datasets generated by time-varying processes, can be unusable. Our objective here is to develop a predictive model and associated inference algorithms that learns to anticipate optimal exploration actions in environments that are uncertain and time-varying. In essence, we are seeking an algorithm that can predict, with confidence, which exploration actions will be optimal given past experience. The methods that we develop are rather general, and applicable to a host of machine learning problems where exploration is critically important, including Reinforcement Learning, Bandit Problems, and supervised or unsupervised learning over very large data sets.

To quantify how much is learned during exploration, many authors have used a form of variance reduction in the context of Bayesian algorithms [161, 76, 25, 55, 142] or entropy reduction (mutual information) [124, 134, 143, 6, 141, 68, 36]. However, as illustrated in Example 2.1, both entropy reduction and variance reduction may be coarse, and even unreliable, measures for learning. The development in this work utilizes instead the information gain IG for quantifying exploration. A particularly interesting property of the information gain is that it *only indicates that nothing has been learned if and only if nothing has truly been learned*; i.e., that $IG(f(Y)||f_0(Y)) = 0$ if and only if $f(Y) = f_0(Y)$ [71] (see Example 2.1).

The lack of a formalized predictive model for the information gain in the literature makes it difficult to understand how to model and predict it. Moreover, a formalized predictive model should allow for online-computable accuracy bounds for the information gain to be developed; and, as we show in Figures 53 and 54, autonomously learning systems do not perform well in real-world datasets without such a bound. The main contributions of our work, summarized

below, are concerned with developing such a formalized predictive model of information gain and deriving associated bounds. We also provide an algorithm that utilizes this model to explore optimally.

In this section, we introduce Example 2.1 which briefly demonstrates that entropy reduction and variance reduction can fail to quantify that substantial learning has occurred, while the information gain succeeds. We also introduce the Bienaymé-Chebyshev inequality in Section 2.1 and the class of Lévy processes in Section 2.2. The interplay between the Bienaymé-Chebyshev inequality with the class of Lévy processes is crucial for the reader to understand how our bound in Lemma 2.9 fits, contextually. This context will help the reader to understand how identically distributed variables are not required for our novel bound in Lemma 2.9.

Example 2.1 (Gaussian Distribution Exploration Example). *An agent measuring temperature in Celsius⁴ has a prior belief of $Y \sim \mathcal{N}(30^\circ\text{C}, 0.2^\circ\text{C}^2)$ and receives a message⁵ which causes its posterior belief to be $Y \sim \mathcal{N}(25^\circ\text{C}, 0.2^\circ\text{C}^2)$. The resultant entropy reduction (ER), variance reduction (VR), and information gain (IG) are then*

$$ER = \frac{1}{2} \ln(2\pi e(0.2^2)) - \frac{1}{2} \ln(2\pi e(0.2^2)) = 0 \text{ nats}, \quad (130)$$

$$VR = 0.2 - 0.2 = 0^\circ\text{C}^2, \quad (131)$$

and

$$\begin{aligned} IG(\mathcal{N}(25, 0.2) || \mathcal{N}(30, 0.2)) &= \frac{1}{2} \left[\ln \left(\frac{0.2}{0.2} \right) + tr \left[\frac{0.2}{0.2} \right] - 1 + (30 - 25)^T (0.2^{-1}) (30 - 25) \right] \\ &= 62.5 \text{ nats}. \end{aligned} \quad (132)$$

Example 2.1 is absolutely important to understand as it shows, in addition to the fact that entropy reduction and variance reduction lack the property in [71], that similar changes in belief may lead to a drastic *undervaluing of learning*. This may be observed by repeating Example 2.1 with prior and posterior beliefs of $Y \sim \mathcal{N}(30^\circ\text{C}, 0.2^\circ\text{C}^2)$ and $Y \sim \mathcal{N}(25^\circ\text{C}, 0.1^\circ\text{C}^2)$, respectively. Moreover, the information gain is parameter invariant [71]. Consequently, in repeating Example 2.1 using a prior belief of $Y \sim \mathcal{N}(300^\circ\text{dC}, 20^\circ\text{dC}^2)$ and $Y \sim \mathcal{N}(250^\circ\text{dC}, 10^\circ\text{dC}^2)$, the entropy reduction value would be changed while the information gain value would be the same as the previous exercise. All of these exercises reinforce that the information gain is more reliable and accurate for learning. Due to these exercises, as well as a theorem of uniqueness which demonstrates that *the information gain uniquely quantifies the entropy contained in a message which causes a prior belief to change into a posterior belief* [70], baseline algorithms are considered relevant to this work if they either 1) use a predictive model on the information gain (informed exploration) or 2) are not data-driven (uninformed exploration).

2.1 Probabilistic Inequalities

The guarantees offered by the Bienaymé-Chebyshev inequality are desirable, namely it provides error bounds on our next sample.

Lemma 2.2 (Bienaymé-Chebyshev Inequality [69]). *Let the random variables $Z(t_1), \dots, Z(t_n) \in [0, +\infty)$ be independent and identically distributed trials. Let $\bar{Z} = \frac{1}{n} \sum_{j=1}^n Z(t_j)$ and $\mu = \mathbb{E}(\bar{Z})$.*

⁴Note that the information gain is parameter invariant, while entropy reduction is not [71]. Thus, a change in variables would change values output by entropy reduction, but would not affect the information gain.

⁵The word choice in Example 2.1 is sourced from the *theorem of uniqueness in information theory*.

Then with $k > 0$,

$$\Pr(|\bar{Z} - \mu| \geq k) \leq \frac{\text{VAR}(Z)}{nk^2}. \quad (133)$$

One limitation, however, lies in the fact that the samples are required to be identically distributed; this is undesirable since we want to study time-varying distributions that are sampled over time. In Lemma 2.9 we extend the Bienaymé-Chebyshev inequality to account for non-identically distributed samples.

2.2 Lévy Process

Here, we review the properties of Lévy processes and how these properties allow the information gain to be modeled as a random variable. A Lévy process $\mathcal{Z}$ for the random variable Z has the following properties for increments $\Delta Z(\Delta t) > 0$ [5]:

1. $\mathcal{Z}$ has independent and stationary increments,
2. $\Delta Z(0) = 0$ with probability one,
3. $\forall k > 0$ and $\forall \Delta t > 0$, $\lim_{\Delta t \rightarrow 0} \Pr(\Delta Z(\Delta t) > k) = 0$.

The most crucial point to note is that $\Delta Z(0) = 0$ with probability one. This shows that a Lévy process can incorporate deterministic observations, yet still model a stochastic process. Also, since Lévy processes are nonstationary processes, as noted by point 1), a Lévy process may facilitate a more nuanced regression than the sample average used in [97]. Moreover, the independent and identically distributed condition in the Bienaymé-Chebyshev inequality in Lemma 2.2 can potentially be relaxed to an independence condition.

2.3 Formulation: Exploration as Multi-Play N-Armed Restless Bandits

In this work we consider the following exploration problem: there exists a set of N sensing locations in a spatial domain from which we may observe a time-varying phenomenon. We denote the set of all sensing locations as $\Omega \doteq \{1, \dots, N\}$. In particular, each sensing location $i \in \Omega$ is associated to a stochastic process $\mathcal{Y}_i$ that we want to estimate. At each time instance t , the sensing location i may provide a *sample* (or a realization) $Y_i(t)$ of the process $\mathcal{Y}_i$. However, due to resource constraints (e.g., cost), we cannot acquire samples from all the N sensing location, and instead we acquire samples only from a subset of them; i.e., we can only observe the samples at $\kappa \leq N$ sensing locations, where κ is a given upper bound. Therefore, the problem we address in this work is learning how to select the subset of Ω , having cardinality κ , where the phenomena Y_i exhibit the greatest change, as quantified by the information gain. The information gain is defined as

$$IG(f(Y)||f_0(Y)) = \int_{f(Y) \ll f_0(Y)} f(y) \ln \frac{f(y)}{f_0(y)} dy, \quad (134)$$

where y is a dummy random variable, Y is a random variable, $f(Y)$ is the posterior belief, $f_0(Y)$ is the prior belief, and $f(Y) \ll f_0(Y)$ denotes that $f_0(Y) = 0 \Rightarrow f(Y) = 0$ [70].

2.3.1 Multi-Play N-Armed Restless Bandit Formulation

In our formulation, we focus on the exploration task, where only κ of $N \geq \kappa$ sensing locations may be sampled at any given time instance t (episode) within the sensing location set Ω . Feedback from the selected subset of sensing locations $\mathcal{S}(t) \subset \Omega$ is used to update the corresponding models for each time-varying process $\mathcal{Y}_i$. We also associate a second stochastic process to each sensing location, which models the evolution of the information gain at each location. We denote this information gain process with the symbol $\mathcal{Z}_i$, for all $i \in \Omega$. Similarly, at each time instance t , sensing location i observes a sample Z_i of the process $\mathcal{Z}_i$. For simplicity, the processes $\mathcal{Y}_i$ and $\mathcal{Z}_i$, associated to each sensing location $i \in \Omega$, are assumed to be statistically independent.

In this formulation, each *arm* of the bandit corresponds to a sensing location $i \in \Omega$. Therefore, the overall goal is to select the best set of κ bandit arms such that some measure on exploration reward is optimized. We call $\mathcal{S}(t)$ the set of sensing locations (arms) selected at time t . Intuitively, since the sensing locations observe time-varying phenomena, the information gain at a given location changes over time. Therefore, our goal is to use a data-driven approach that can dynamically learn which sensing location is best to sample. More formally, the objective of our exploration policy is to maximize the total information gain obtained in each episode of the exploration task. This can be achieved by visiting the sensing locations which are expected to have the most information gain:

$$\begin{aligned} \mathcal{S}(t) \doteq & \underset{\mathcal{S} \subset \Omega}{\operatorname{argmax}} \quad \sum_{i \in \mathcal{S}} \mathbb{E}[Z_i] \\ & \text{subject to} \quad \operatorname{Card}(\mathcal{S}) = \kappa, \end{aligned} \quad (135)$$

where $\operatorname{Card}(\cdot)$ denotes the cardinality of a set.

2.3.2 Modeling the Information Gain

The data received as feedback $y_i \sim Y_i$ from the selected subset of sensing locations, $\mathcal{S}(t)$, informs our model on the data process Y_i . Similarly, the information gain z_i of the feedback y_i informs our model on the information process Z_i . Formally, we use the following hierarchical model

$$\begin{aligned} y_i & \sim Y_i \\ z_i & \sim Z_i | y_i. \end{aligned} \quad (136)$$

To understand why this hierarchical model is meaningful, we must first demonstrate the context by which information gain is a stochastic calculation. Fundamentally, we must first explicitly acknowledge that (134) is a *deterministic* quantity when we are given the posterior $f(Y)$ and prior $f_0(Y)$ beliefs; i.e.,

$$\Pr(Z = IG(f(Y)||f_0(Y))|f(Y), f_0(Y)) = 1. \quad (137)$$

Therefore, taking the sample average as in [97] does not make immediate sense. However the sample average in [97] can be formulated as

$$PIG = \mathbb{E}_Y[IG(f(Y)||f_0(Y))|f_0(Y)], \quad (138)$$

where $f(Z) = f(IG(f(Y)||f_0(Y))|f_0(Y))$ is stochastic because the posterior belief $f(Y)$ is not given. It is crucial to note here that (138) is an expectation on the posterior distribution, and not an expectation on the information gain directly. Understanding this distinction is key to understanding the significance of Fact 2.3

Fact 2.3. *The information gain is a function from Y to Z that is **onto** but not **one-to-one**.*

Hence, an expectation on the information gain, directly, would output the same values as (138), but would *not* require an accurate prediction of the data observed; i.e.,

$$EIG = \mathbb{E}_Z[IG(f(Y)||f_0(Y))|f_0(Y)] = PIG, \quad (139)$$

where the expected information gain EIG is implicitly, rather than explicitly, conditioned on the posterior belief.

2.3.3 Bi-Homogeneous Poisson Exposure Process Model

As mentioned in Section, 2.2 the Lévy process allows us to reconcile that (137) and (139) are both true. The Lévy process also affords us the ability to model the expected information gain of a more general class of stochastic processes.

Fact 2.4. *The information gain Z of any process $\mathcal{Y}$ with stationary increments is generated by a Lévy process; i.e., Z is a Lévy process.*

The rationale for modeling the expected information gain using the bi-homogeneous Poisson exposure process (bi-Pep) model is provided in Facts 2.5 and 2.6

Fact 2.5. *The summation of sequentially-obtained information gain due to each observation at sensor location i is a monotonically increasing function.*

Fact 2.6. *The information gain available at a time-varying process $\mathcal{Y}$ will also vary with time.*

From Fact 2.5 we expect the information gain to increase with respect to the number of messages received at a sensing location i . Similarly, from Fact 2.6 we expect that the available information gain at a sensing location i will vary with time. Therefore, we want to incorporate the number of messages and time of receipt into some model which considers the *information gain per unit-time per sample*.

Definition 2.7. *The Poisson exposure process (Pep) is defined as*

$$f(z|\Lambda(t, n)) = C_{\Lambda(t, n)} \frac{(\Lambda(t, n))^z e^{-\Lambda(t, n)}}{\Gamma(z + 1)}, \quad (140)$$

where $\Lambda(t, n)$ is the mean, $C_{\Lambda(t, n)}$ is the normalizing constant, and $z \in [0, \mathbb{R}^+]$ is the random variable. A Pep is called *homogeneous* if the derivative of the Pep mean is some positive constant; i.e., $\lambda > 0$. Here, we call a Pep *bi-homogeneous* if $\frac{d}{dt} \frac{d}{dn} \Lambda(t, n) = \lambda$, where $\lambda > 0$ is some constant rate parameter.

We may learn the homogeneous rate parameter λ of our bi-Pep model using a gamma distribution whose parameters update as

$$G\left(\lambda \mid \alpha = \frac{\alpha_1 + \alpha_2}{2}, \beta = \beta_1 \beta_2\right), \quad (141)$$

where $\lambda = \frac{\alpha}{\beta}$ and the parameter updates are formed from two separate conjugate, each being mathematically similar to the Bayesian update of two separate homogeneous Poisson processes, as shown in Fact 2.8

Fact 2.8. The parameters α_1 , α_2 , β_1 , and β_2 for the proposed gamma distribution in (141) are analytically updated as

$$\begin{aligned} G(\lambda\Delta t|\alpha_1 + z, \beta_1 + \Delta t) &\propto \text{Pep}(z|\lambda\Delta t)G(\lambda\Delta t|\alpha_1, \beta_1) \\ G(\lambda\Delta n|\alpha_2 + z, \beta_2 + \Delta n) &\propto \text{Pep}(z|\lambda\Delta n)G(\lambda\Delta n|\alpha_2, \beta_2), \end{aligned} \quad (142)$$

where each gamma distribution is a conjugate prior to a corresponding surrogate homogeneous Pep.

2.4 Probabilistic Bounds for Uninformed-to-Informed Exploration using the bi-Pep Model

The Bienaymé-Chebyshev inequality essentially uses the assumption of independent and identically distributed variables to bound the average increase in a summation of random variables. The α parameter of the bi-Pep model encodes the summation of the random variable Z . As the bi-Pep models the expected increase in Z per unit time per unit sample as $\frac{d}{dt} \frac{d}{dn} \Lambda(t, n) = \lambda$, we develop the following Lemma.

Lemma 2.9. Let $\Delta Z(t_1), \dots, \Delta Z(t_n) > 0$ be independent Poisson exposure process increments, $\bar{Z}(t_n) = \frac{1}{nt_n} \sum_{j=1}^n \Delta Z(t_j)$, and $\frac{d}{dt} \frac{d}{dn} \Lambda(\Delta t, \Delta n) = \lambda = \mathbb{E}[\bar{Z}(t_n)]$. Then, with $k > 0$,

$$\Pr(|\bar{Z} - \lambda| \geq k) \leq \frac{\alpha}{k^2 \beta^3} = \frac{\alpha}{k^2 (n(t_n - t_1))^3}. \quad (143)$$

While the design of k is almost arbitrary in Lemma 2.9, vastly different performances may result if k is set too high, which would resemble Figures 53 and 54, or too low, which would result in unnecessarily long durations of uninformed exploration. To facilitate high-performance implementation without requiring expert knowledge, we must design k using values *known to our exploring agent*. As such, we use the bi-Pep model parameters present in (143); i.e., λ , $\alpha = \sum_1^n \Delta Z_j$, and $\beta = n(t_n - t_1)$. High-performance informed exploration will be probabilistically assured when the exponents attached to each of these parameters result in the following properties.

Property 2.10. The probability that the model error threshold k of an exploring agent will be exceeded at a location will decrease with the number of samples n , duration of sampling $(t_n - t_1)$, and the total information collected $\sum_1^n \Delta Z_j$.

Property 2.11. The model error threshold k of an exploring agent at a location should decrease with the number of samples n and duration of sampling $(t_n - t_1)$.

Property 2.12. The total information $\sum_1^n \Delta Z_j$ required at a location prior to initiating informed exploration is directly proportional to the information exposure rate λ .

Property 2.13. The total number of samples n and the duration of sampling $(t_n - t_1)$ required at a location prior to initiating informed exploration is inversely proportional to the information exposure rate λ .

Fact 2.14 extends Lemma 2.9 to incorporate Property 2.10

Fact 2.14. In Lemma 2.9, let $k = \frac{\alpha^\gamma}{\beta^\rho}$, then Property 2.10 is satisfied when $\gamma < \rho$.

Similarly, Fact 2.15 extends Lemma 2.9 to incorporate Property 2.11

Fact 2.15. In Lemma 2.9, let $k = \frac{\alpha^\gamma}{\beta^\rho}$, then the right-hand side of (143) may be rewritten as

$$\frac{1}{\alpha^{(2\gamma-1)}\beta^{(3-2\rho)}} \quad (144)$$

and Property 2.11 is satisfied when $\gamma > \frac{1}{2}$ and $\rho < \frac{3}{2}$.

From Fact 2.15 we can rewrite the right-hand side of (143) to demonstrate that Property 2.12 is satisfied as

$$\frac{1}{\alpha^{(2\gamma-1)}\beta^{(3-2\rho)}} \cdot \frac{\alpha^{(3-2\rho)}}{\alpha^{(3-2\rho)}} = \frac{\lambda^{(3-2\rho)}}{\alpha^{(2\gamma+2-2\rho)}}. \quad (145)$$

From Fact 2.15 we can also rewrite the right-hand side of (143) to demonstrate that Property 2.13 is satisfied as

$$\frac{1}{\alpha^{(2\gamma-1)}\beta^{(3-2\rho)}} \cdot \frac{\beta^{(2\gamma-1)}}{\beta^{(2\gamma-1)}} = \frac{1}{\lambda^{(2\gamma-1)}\beta^{(2+2\gamma-2\rho)}} \quad (146)$$

In the following Theorem, we show that Properties 2.10, 2.12 hold for a finite open interval of values for γ and ρ .

Theorem 2.16. Let $\Delta Z(t_1), \dots, \Delta Z(t_n) > 0$ be independent Poisson exposure process increments, $\bar{Z}(t_n) = \frac{1}{nt_n} \sum_{j=1}^n \Delta Z(t_j)$, and $\frac{d}{dt} \frac{d}{dn} \Lambda(\Delta t, \Delta n) = \lambda = \mathbb{E}[\bar{Z}(t_n)]$. Then, with $\frac{1}{2} < \gamma < \rho < \frac{3}{2}$,

$$\Pr \left(|\bar{Z} - \lambda| \geq \frac{\alpha^\gamma}{\beta^\rho} \right) \leq \frac{1}{\alpha^{(2\gamma-1)}\beta^{(3-2\rho)}} = \frac{1}{\alpha^{(2\gamma-1)} (n(t_n - t_1))^{(3-2\rho)}}, \quad (147)$$

and Properties 2.10, 2.12 hold.

Desirable properties for an error bound for the bi-Pep model include: 1) the probability of error being above an error threshold decays to an equal effect as the information information acquired α , the total duration of observation $(t_n - t_1)$, and the total number of observations n each increase, 2) for any constant exposure rate λ , the error threshold decays to an equal effect as the total duration of observation $(t_n - t_1)$ and the total number of observations n , and 3) the total duration of observation $(t_n - t_1)$ and the total number of observations n should be inversely proportional to the exposure rate λ ; i.e., faster/slower changing environments require a shorter/longer duration of observation $(t_n - t_1)$ and less/more samples n to predict in the environment with meaningful accuracy. We consequently derive the following error bound to ensure the integrity of the bi-Pep model.

Theorem 2.17. Let $\Delta Z(t_1), \dots, \Delta Z(t_n) > 0$ be independent Poisson exposure process increments, $\bar{Z}(t_n) = \frac{1}{nt_n} \sum_{j=1}^n \Delta Z(t_j)$, and $\frac{d}{dt} \frac{d}{dn} \Lambda(\Delta t, \Delta n) = \lambda = \mathbb{E}[\bar{Z}(t_n)]$. Then,

$$\Pr \left(|\bar{Z} - \lambda| \geq \frac{\lambda^{\frac{3}{4}}}{\sqrt{n(t_n - t_1)}} \right) \leq \frac{1}{\sqrt{\alpha n(t_n - t_1)}} = \frac{1}{n(t_n - t_1)\sqrt{\lambda}}. \quad (148)$$

The bound in Theorem 2.17 extends the Bienaymé-Chebyshev inequality such that *samples need not be identically distributed*. While Theorem 2.17 provides an error bound for learning a single bi-homogeneous Pep, we need to obtain error bounds for the entire problem domain before we may trust our learned model for informed exploration. The guarantees on the bi-homogeneous Pep regression by Theorem 2.17 do provide a principled condition for transitioning from uninformed exploration to informed exploration in Corollary 2.18. Consequently, we develop the following Corollary to provide a condition for transitioning between uninformed and informed exploration.

Corollary 2.18 (Uninformed-to-Informed Bound for bi-Pep). *If the environment statistics are time-varying, then the accuracy threshold, k such that $0 < k \leq 1$, for uninformed-to-informed exploration using a bi-homogeneous Poisson exposure process, with a rate parameter in terms of the information gain per unit-time and per sample, is satisfied when*

$$t_{(b,n)} > t_{(b,1)} + \frac{1}{n_b \alpha_b k^2}, \quad (149)$$

where the inequality

$$\frac{1}{\sqrt{\alpha_b \beta_b}} < k \quad (150)$$

holds true, where

$$b = \operatorname{argmax}_i \frac{1}{\sqrt{\alpha_i \beta_i}}$$

2.5 Information Gain Dataset Prediction Results

The bounds derived in Section 2.4 qualify the accuracy condition in the uninformed-to-informed exploration algorithms shown in Algorithm 19 and Algorithm 20 respectively. Two sets of empirical results are shown for exploration simulations on real-world datasets, where the exploration algorithms have no prior knowledge about the dataset statistics. The real-world datasets used for experimentation include the Intel Berkeley (indoor) temperature dataset [14], the European Research Area (ERA) daily interim (outdoor) temperature dataset [11], the Washington rainfall dataset [167], and the Ireland windspeed dataset [64]. Additional details about the aforementioned real-world datasets are available in Appendix D.

For both sets of empirical results, over a hundred total testing conditions are examined to rigorously characterize the performance of each exploration algorithm with respect to the subset of sensing locations which may be sampled during an episode; i.e., κ . The first set of empirical results on the real-world datasets involves the informed exploration algorithms such as PIG [97] and where the accuracy condition for RAPTOR (Algorithm 20) is assumed to be satisfied upon initialization. The second set of empirical results on the real-world datasets involves the uninformed-to-informed exploration algorithms, where the accuracy condition must be satisfied according to Corollary 2.18 for RAPTOR and an analogous accuracy condition is imposed on PIG to yield PIG*.

The first set of empirical results on the real-world datasets shows that the performance for informed exploration variants of PIG [97] and Algorithm 20 is poor when compared to sequential and random sampling methods. The second set of empirical results shows that the performance of the uninformed-to-informed exploration variant of PIG, shown in Algorithm 19, performs at least as well as sequential and random sampling methods. The second set of empirical results also shows that the performance of RAPTOR, shown in Algorithm 20 significantly outperforms Algorithm 19, sequential search, and random search.

Both sets of empirical results use a normal distribution at each sensing location i to model the process $\mathcal{Y}_i$. For each sample received, we calculate the frequentist variance σ_i^2 of the normal distribution. Using the calculated frequentist variance, we can use the following Bayesian update for the normal distribution

$$\mu_{(f,i)} = \frac{\frac{n_i}{\sigma_i^2} \mu_{(f_0,i)} + \frac{y_i}{\sigma_{(f_0,i)}^2}}{\frac{n_i}{\sigma_i^2} + \sigma_{(f_0,i)}^{-2}} \text{ and } \sigma_f^2 = \left(\frac{\sigma^{-2}}{n_i} + \sigma_{f_0}^{-2} \right)^{-1}, \quad (151)$$

where $\mu_{(f,i)}$ is the posterior mean, $\mu_{(f_0,i)}$ is the prior mean, $\sigma_{(f,i)}^2$ is the posterior variance, $\sigma_{(f_0,i)}^2$ is the prior variance, and y_i is the sample mean [57]. The information gain of the posterior normal distribution is then computed as

$$IG(f(Y)||f_0(Y)) = 0.5 \left[\ln \left(\frac{\sigma_{f_0}^2}{\sigma_f^2} \right) + \text{tr} \left[(\sigma_{f_0}^2)^{-1} \sigma_f^2 \right] - 1 + (\mu_{f_0} - \mu_f)^T (\sigma_{f_0}^2)^{-1} (\mu_{f_0} - \mu_f) \right], \quad (152)$$

where 1 is the dimensionality of the data which we use for experiments. The regressions used for (152) uniquely distinguish the empirical performance of PIG and RAPTOR⁻ (the informed exploration variant of RAPTOR). Similarly, the regressions and derived bounds used for (152) distinguish the empirical performance of PIG* and RAPTOR.

2.6 Informed Exploration Results

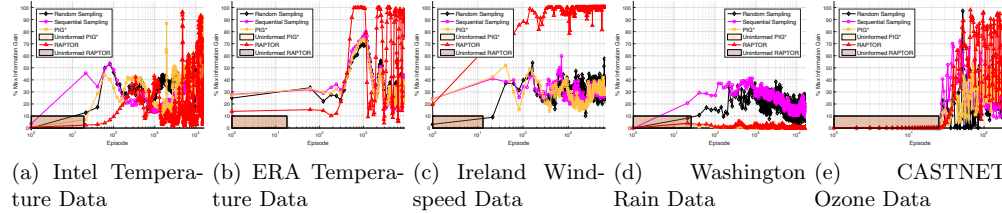


Figure 53: Here, RAPTOR⁻ and PIG are the informed exploration variants of RAPTOR and PIG*, respectively. A complete experiment for a specific testing condition in each real-world dataset is shown. For Figures 53a, 53b, 53c, and 53d we used $\kappa = 6$, $\kappa = 6$, $\kappa = 2$, $\kappa = 2$, and $\kappa = 5$ as our testing conditions, respectively. Note that RAPTOR⁻ and PIG do not use the derived bounds. Rather, they always act in the informed exploration mode, as a result, their performance is compromised. The results can be compared with those in Figure 55 where RAPTOR and PIG*, which do use the bounds to transition from uninformed to informed exploration, perform significantly better.

The PIG informed exploration algorithm is equivalent to Algorithm 19 when one assumes that the accuracy condition is satisfied upon initialization. We similarly implement an informed exploration variant of RAPTOR, where the accuracy condition in Algorithm 20 is assumed to be satisfied upon initialization. The results in Figures 53 and 54 show that the experiment-level and summary performance of the informed exploration algorithms pales in comparison to sequential and random sampling methods.

These poor results are predicted by the information gap hypothesis, and so we take these experimental results as validation of the information gap hypothesis with respect to the information gain measure. In fact, the information gap hypothesis appears to be a type of explore-exploit dilemma for the task of exploration itself, and an insufficient amount of information collection prior to the engagement of informed exploration leads to long durations of sub-optimal performance. Hence, these results ultimately highlight the necessity of establishing principled accuracy conditions for informed exploration, and is the basis for the general uninformed-to-informed exploration proposed.

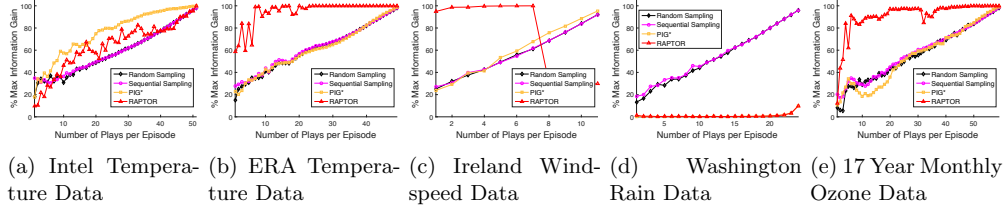


Figure 54: Here, RAPTOR^- and PIG^- are the informed exploration variants of RAPTOR and PIG^* , respectively. The performance of each exploration algorithm during the last episode in an array of over a hundred testing conditions on four real-world datasets is shown. Testing conditions are affected by the designated subset size, η_t ; i.e., the number of bandit arm visits or plays per episode. For each dataset, when the number of plays reached a certain threshold, there is a marked improvement in the performance of RAPTOR^- , and sometimes PIG^- , ahead of the end-condition of testing, where the number of plays allowed per episode is equal to the number of sensing locations or bandit arms in the domain; i.e., $\kappa = K$. This trend may suggest that the information gain is difficult to predict, and so the performance of an exploration algorithm may be determined by coverage. Conversely, if the derived probability bounds for uninformed-to-informed exploration result in significantly improved performance, then we will have empirically supported the information gap hypothesis.

2.7 Uninformed-to-Informed Exploration Results

Due to the poor results of PIG and the informed exploration variant of RAPTOR , we now implement PIG^* and RAPTOR as described in Algorithms 19 and 20 respectively. The results in Figures 55 and 56 show that the experiment-level and summary performance of the uninformed-to-informed exploration algorithms are significantly better than the experiment-level and summary performance of the informed exploration algorithms shown in Figures 53 and 54. The performance of PIG^* is at least comparable to sequential and random sampling, despite having an assumption mismatch between the time-varying environment statistics and the Ped regression used to drive exploration. The RAPTOR algorithm outperforms PIG^* , sequential sampling, and random sampling. The strength of the RAPTOR algorithm performance when compared to PIG^* is solely due to the improved accuracy of the bi-homogeneous Pep regression compared to the Ped regression.

Algorithm 19 PIG*

Input: set $S \in \mathbb{R}^K$, subset size κ
 Initialize $(\alpha_i, \beta_i, \lambda_i, n_i, \eta_0, t_{(i,0)}) \leftarrow 0 \ \forall i$
for
 each epoch at time t **do**
 $\mathbb{E}[Z_i] \leftarrow \frac{1}{n_i} \sum_{j=1}^{n_i} Z_i(t_{(i,j)}) \ \forall i$
 If: PIG* Variant of Corollary 2.18
 $\eta_t \leftarrow$ Informed policy; e.g., (135)
 Else: $\eta_t \leftarrow$ Uninformed Policy (Sequential, Random, etc.)
 If: $i \in \eta_t$ and $n_i=1$
 $\mu_i \leftarrow x_i$
 Else: Update belief on $\mathcal{Y}_i$; e.g., (151) $\forall i \in \eta_t$
 If: $\min_i n_i \geq 2$
 Update belief λ_i ; e.g., (sample average) $\forall i \in \eta_t$
end for

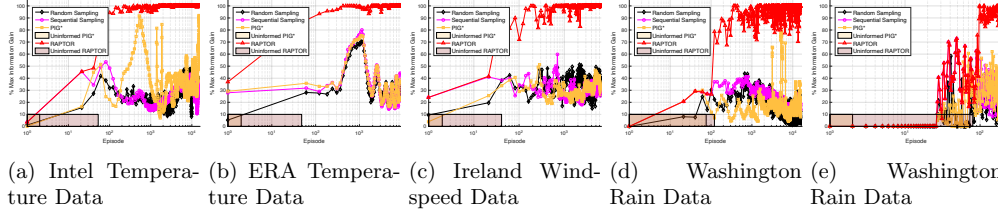


Figure 55: Uninformed-to-informed exploration using RAPTOR is compared against baselines. Here, we show a complete experiment for a specific testing condition involving the uninformed-to-informed exploration algorithms, RAPTOR and PIG*. For Figures 55a, 55b, 55c, and 55d we used $\kappa = 6$, $\kappa = 6$, $\kappa = 2$, $\kappa = 2$, and $\kappa = 5$ as our testing conditions, respectively.

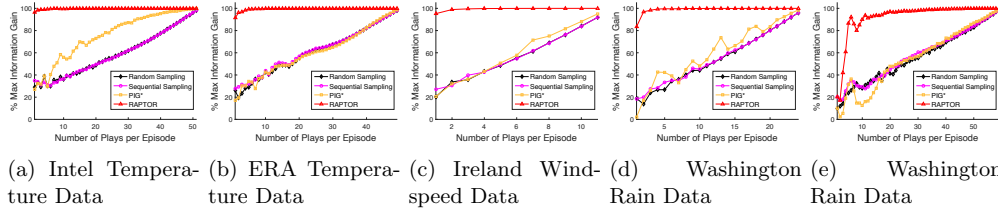


Figure 56: Here, we show an array of testing conditions involving the uninformed-to-informed exploration algorithms, RAPTOR and PIG*. The performance of each exploration algorithm during the last episode in an array of over a hundred testing conditions on four real-world datasets is shown. Testing conditions are affected by the designated subset size, η_t ; i.e., the number of bandit arm visits or plays per episode. Testing conditions are affected by the set subset size, η_t ; i.e., the number of bandit arm visits or plays per episode. Unlike in Figure 54, the performance of both PIG* and RAPTOR is at least as effective as sequential or random sampling approaches across the spectrum of testing conditions.

Algorithm 20 RAPTOR and PIG*

Input: set $S \in \mathbb{R}^K$, subset size κ
Initialize $(\alpha_i, \beta_i, \lambda_i, n_i, \eta_0, t_{(i,0)}, \Delta t_i) \leftarrow 0 \ \forall i$
for
 each epoch at time t **do**
 $\mathbb{E}[Z_i | \Delta t_i, \Delta n_i] \leftarrow \lambda_i \Delta t_i \Delta n_i + Z_i(t_{(i,n_i)}) \ \forall i$
 If: Corollary 2.18
 $\eta_t \leftarrow$ Informed policy; e.g., (135)
 Else: $\eta_t \leftarrow$ Uninformed Policy (Sequential, Random, etc.)
 If: $i \in \eta_t$ and $n_i=1$
 $\mu_i \leftarrow x_i$
 Else: Update belief on $\mathcal{Y}_i$; e.g., (151) $\forall i \in \eta_t$
 If: $\min_i n_i \geq 2$
 Update belief λ_i (141) $\forall i \in \eta_t$
 end for

3 Chance-Constrained Optimization for Decision-making in Presence of Risk

This section briefly introduces Chance-Constrained Model Predictive Control (CCMPC) as a means to shape the reference command in order to abide input and state constraints as well as leverage the full capability of the plant. The priority of all stochastic optimal control architectures is to cancel the uncertainty in the system and track the reference command.

Let $x \in D_x \subset \mathbb{R}^l$ be defined as the state vector of the system where D_x is a subset of $\mathbb{R}^l$ and $u \in \mathbb{R}^m$ be an admissible control input to the system. Let $A \in \mathbb{R}^{l \times l}$ and $B \in \mathbb{R}^{l \times m}$ be state and input matrices, respectively. The discretized version of the plant dynamics subject to a stochastic uncertainty is used as a model for the CCMPC:

$$x(k+1) = Ax(k) + Bu(k) + Bw(k). \quad (153)$$

Here, (A, B) denote the discretized version of the original matrices, $x(k)$ denotes the discrete-time state at the time instant k and $w(k) \in \mathbb{R}^m$ represents an integrated Wiener process. Let $\Delta u(k) = u(k) - u(k-1)$ be a future control input, let $N_c > 0$ and $N_p \geq N_c$ denote the control and prediction horizon, as it is defined in Model Predictive Control formulation. Let $\Delta U = [\Delta u(k) \cdots \Delta u(k+N_c-1)]^T$ with $\Delta U \in \mathbb{R}^{N_c \cdot m}$ be the incremental control sequence that captures the optimal trajectory and let $W = [w(k) \cdots w(k+N_p-1)]^T$ with $W \in \mathbb{R}^{N_p \cdot l}$. Denote $X \in \mathbb{R}^{N_p \cdot l}$ to be the vector of predicted states, then

$$X = G_{xx}x_0 + G_{xu}\Delta U + G_{xw}W, \quad (154)$$

where the matrices G_{xx}, G_{xu}, G_{xw} are generated by repetitive application of (153). The available control authority at the time instant k is assumed to be in the admissible set, $u \in D_u$.

Let F_X denote a convex polytope of state constraints. Consider a cost function $\mathcal{F}(\bar{X}, \Delta U)$, which is convex in $(\bar{X}, \Delta U)$. The CCMPC problem then is

$$\begin{aligned}
& \min_{\Delta U} \mathcal{F}(\bar{X}, \Delta U) \\
& u \in D_u \\
& P(X \notin F_X) \leq \delta \\
& X = G_{xx}x_0 + G_{xu}\Delta U + G_{xw}W.
\end{aligned} \tag{155}$$

Despite the linear dynamics in (153), the CCMPC problem is non-convex. In principle, the CCMPC problem can be solved with nonlinear solvers, although it needs to be relaxed to a convex approximation of the problem for practical implementation so that the nonlinear programming methods will be able to find the global optimum (see [94]). In order to relax the optimal control problem and obtain a solution, either nonlinear solvers or approximating methods can be employed (see e.g. [12, 112]).

To solve the CCMPC problem here, we modify the output constraints by placing a normal distribution, $f(z | \mu, \sigma)$, on each one. The mean of the distribution, μ , is the previous limit for each constraint (X_{min}, X_{max}) and the standard deviation is an arbitrary small value, $\sigma \ll 1$. By using the inverse cdf (or quantile function) of δ in (155), the value by which the constraints are tightened, is found:

$$c = F^{-1}(\delta | \mu, \sigma). \tag{156}$$

Now, based on (156) and the constraints formulation in Model Predictive Control setting, the general MPC problem is solved with the updated constraints as below:

$$\begin{bmatrix} -C_2 \\ C_2 \\ -G_{xu} \\ G_{xu} \end{bmatrix} \Delta U \leq \begin{bmatrix} -U_{min} + C_1 u_{rm}(k-1) \\ U_{max} - C_1 u_{rm}(k-1) \\ -X_{min} + (c - X_{min}) + G_{xx}x(k) \\ X_{max} - (c - X_{max}) - G_{xx}x(k) \end{bmatrix}. \tag{157}$$

Figure 57 shows the performance of the system with CCMPC while original constraints are placed at ± 2 and the threshold for the chance-constraint is at 95%. After tightening the constraints, they are updated to ± 1.97 .

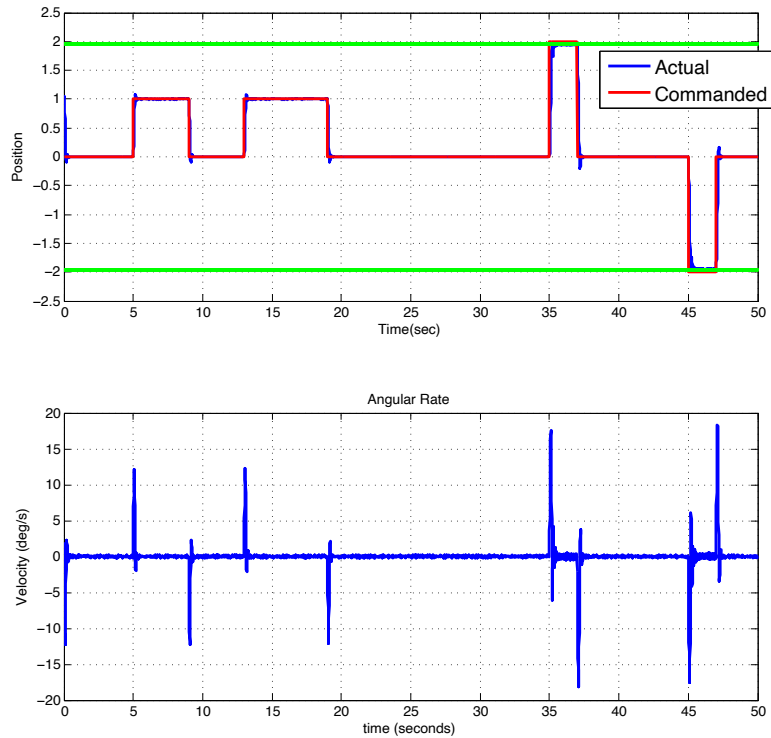


Figure 57: CCMPC Performance

4 Ongoing work in Human-Machine collaborative Autonomy

We have identified and are in process of investigating the following three important aspects of Human-Machine collaborative Autonomy-

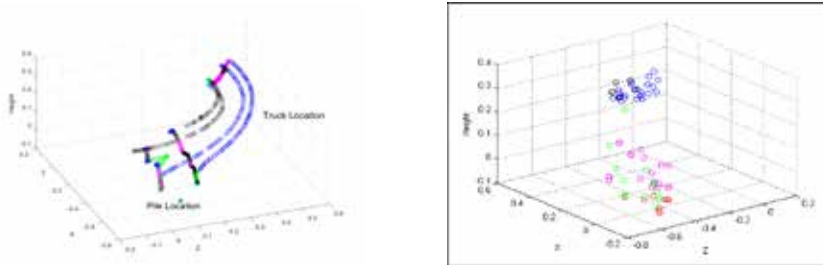
- Learning human-implicit task subgoals from human demonstrations: We have developed a two-step process that first learns subgoals from multiple demonstrations, as well as generalizes a transition model between those subgoals.
- Modeling subgoal interdependence and Inference of current task: Discovery of task subgoals is the first step in understanding the human agents' mental model of performing a particular task. This would certainly allow the UA to act in cohesion. However, the real world missions are comprised of multitudes of task, and there could exist numerous ways in which a particular task is performed. This necessitates the development of non-parametric models that can indefinitely learn task models. We are currently evaluating **Deep Markov models** that will satisfy this requirement. This model will also facilitate rapid inference of ongoing task, which is an utmost requirement for Human-Machine collaborative autonomy.

- Collaborative task execution: Applications where humans and machines have to control a dynamical system together, the key question of interest is what should be the level of autonomy that is appropriate to ensure high task performance without compromising safety. Thus the design of mechanism for sharing control authority between the human and autonomous agent is the key component of collaborative task execution

4.1 Learning human-implicit task subgoals from human demonstrations

We have developed a novel approach to capture the latent subgoals that are used by an expert human operator to perform a task. This approach is computationally efficient in analyzing the human demonstration data and discovers subgoals across multiple demonstrations. Thus there exists two steps in this approach. The first step is to perform autonomous segmentation of the demonstration data that is available to the autonomous agent from the task performance by an expert operator. We then utilize Dirichlet process to discover and cluster all the segments across multiple demonstrations. In the second step, the output of state-segment pairs obtained in the first step is used as an input to the proposed technique of Dirichlet Process-means-Inverse Reinforcement Learning (DP-m-IRL) that autonomously discovers latent subgoals.

For the first step, we have developed a simplified approach for segmentation based on the newly proposed definition of Action primitive. An action primitive characterizes the state for each of the joints of a robot or the states of a machine. Specifically, for a robot with revolute joints, each joint can take either of the three states: clockwise rotation, noisy perturbations (or stationary) or anti-clockwise rotation. Specification of these states for each joint comprises an action primitive. When a robot is performing a task, at any given point of time, its motion can be described by a single unique action primitive. Thus an entire continuous demonstration can be broken into a series of action primitives. Presently, we have applied this approach to the truck loading task performed using 1-14th scale excavator, with the segmentation of continuous demonstration shown in the figure 58a. The figure depicts the motion trajectory of end-effector joint between the pile location and the truck location for two cycles of truck loading. Different colors along the end-effector position depicts decomposition of continuous demonstration into series of action primitive segments.



(a) Action primitive based auto-segmentation of continuous demonstration (b) Possible subgoals: End-points of Action Primitive segments

In the second step we discover the subgoals, based on the rationale that a subgoal is certainly a point in space that demarcates two action primitives. Hence the end-points of all the action primitives are in-fact the potential subgoals, see figure 58b for an example. Final subgoals are obtained utilizing a proposed technique of Dirichlet Process-means-Inverse Reinforcement Learning (DP-m-IRL), which uses a Bayesian non-parametric prior over state action pairs to

assign subgoals. A state action pair is an action primitive segment end-point and the succeeding action primitive. Along with the subgoals this method also returns associated action primitives. These outputs are then used to learn Dynamic Bayesian Network in figure 59 that captures transition between the subgoals (X_i 's) as well as the intermediate transitions between the action primitives (Z_k 's) required to reach the next subgoal. Each action primitive (Z_k) has an associated distribution of Actuator velocities (y_k), from which the velocities are sampled. This probabilistic model will be used by the UA to predict the next action primitive based on the current subgoal and the previous action primitive.

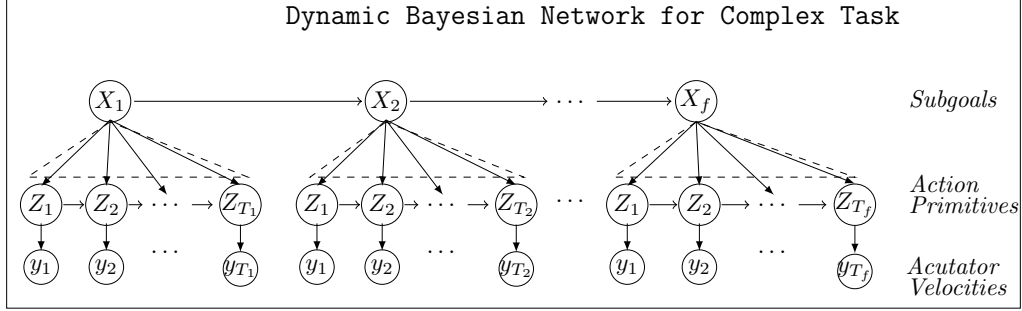


Figure 59: Task Model

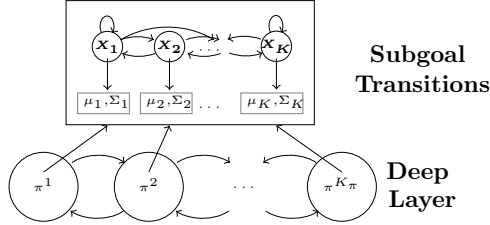


Figure 60: 1-layer Deep Hidden Markov Model: Base layer is a Hidden Markov model with K subgoals (hidden nodes) with Gaussians that define them in space, and a Deep layer that models Markovian dynamics among the set of K_{π} transition probability matrices (TPMs).

4.2 Modeling subgoal interdependence and Inference of current task

In the previous section, we used a Markov chain (inherent in the Dynamic Bayesian network of figure 59) to model subgoal transitions. However, for a real world task, the transitions among subgoals is a time varying complex stochastic process, and would need multiple Markov chains to capture the non-stationarity in the transition function. To model such complex interdependence and transitions among the subgoals we utilize a hierarchical prior of a newly proposed non-parametric approach of Deep Markov model. An example model is shown in figure 60, this is a 1-layer Deep Hidden Markov model (1-l Deep-HMM), where the base layer models subgoal transitions while the non-stationarity in subgoal transitions is captured by the deep layer.

Mathematically, 1-l Deep-HMM (figure 60) is given as

$$\pi^t \sim \Pi_{\pi^{t-1}} \quad (158)$$

$$z_{t+1} \sim \pi^t_{z_t} \quad (159)$$

$$x_{t+1} \sim F(\theta_{z_{t+1}}) \quad (160)$$

At any time t , the next latent mode z_{t+1} is drawn from the transition distribution π^t conditioned upon the current latent mode z_t , where the transition distribution π^t is itself drawn from the hierarchical transition distribution matrix Π given π^{t-1} . We define $\Pi_{i,j} = p(\pi^t = j | \pi^{t-1} = i)$, where $\pi^t \in \Pi^s$ at any t , and $\{\pi^1, \dots, \pi^{K_\pi}\}$ are the elements of set Π^s . Finally the observation x_{t+1} is generated by the distribution $F(\theta_{z_{t+1}})$ where $\theta_{z_{t+1}}$ is a parameter vector associated with state z_{t+1} .

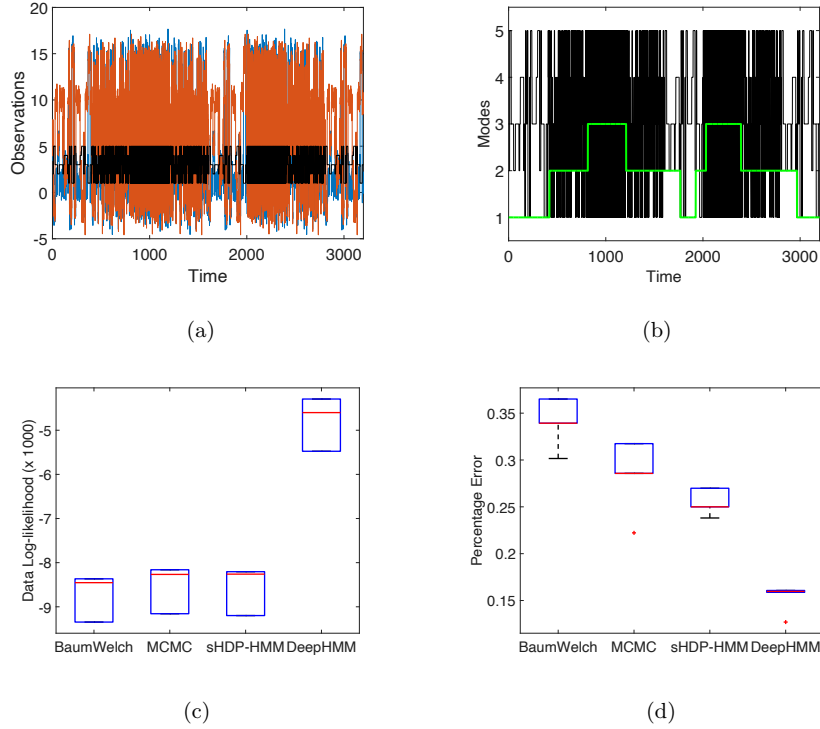


Figure 61: (a) Latent mode sequence (black) and true state sequence for a five state HMM (2d Gaussian emissions), (b) Enlarged view of latent mode sequence overlaid with switches between transition matrices, (c) Predictive likelihood over 100 timeseries data generated by random switching between the TPMs present in (b), (d) Viterbi assignment errors.

We have successfully evaluated and compared the 1-l Deep-HMM on synthetic data generated from a 5 state HMM with 2-d Gaussian emissions. Figure 61a is an example showing the latent mode sequence (in black) laid over the observed data sequence. The mode sequence switches between the 5 states and the observed data is (x, y) position in space. The purpose here is to quantify the benefits of using a Deep Markov model to learn all the existing transition matrices

that are generating the data as compared to the existing methods that learn a single transition matrix. For the time series data of figure 61a, our algorithm identified three TPMs as well as the switching between them, this is overlaid on the latent mode sequence in figure 61b. To compare we also learn the model using existing methods; standard Baum Welch algorithm, MCMC [51] and Bayesian non-parametric sticky-HDPHMM [47]. We tested each method by performing inference over a set of 100 time-series data generated by the same 5 state HMM but with random switch between the three underlying TPMs. This in some way replicates the non-stationary switching between subgoal transitions. We evaluated the performance of each method based on the predictive likelihood of data and percentage error in Viterbi assignments. Results plotted in figures 61c 61d, show that Deep HMM has far higher predictive likelihood as compared to other methods and is very accurate in Viterbi assignments. These experiments suggest that the Deep Markov model has potential in performing accurate inference in real time, and hence further evaluations are under progress.

4.3 Dog fight domain

A potential application of Deep Markov Model is to predict the intention of manned combat aircraft in a dogfight problem. Tactical movements performed in order to gain positional advantage over the opponent, in pursuit scenario during air combat maneuvering is termed as dogfighting. The goals of offensive maneuvering are to remain behind an adversary and to get in a position to shoot. In defensive maneuvering, the aim is to move the bandit out of position and evade the opponent aircraft or to perform disengagement maneuvers, to help facilitate an escape. In neutral engagement both opponents strive for an offensive position. As maneuvers are executed to accomplish any of these objectives, the aircraft invariably expends energy, and pulling lateral accelerations to turn and pitch causes aircraft to slow down, lose altitude or both. Successful pursuit and engagement maneuvers are combination of fundamentals of aerodynamic of flight, geometry of pursuit and physics of managing the aircraft's energy-to-weight ratio, called its specific energy.

Pursuit problem in dog fight scenario, is a problem of target tracking control and objective is to make the system asymptotically follow a target trajectory. Adaptive reactive control for target tracking employs controller comprising of feedback, a feed-forward and adaptive controller. The feedback is designed to exponentially reduce the tracking errors to zero, few orders faster than the desired target response. Feed-forward term is the estimate of the acceleration of the target available through efficiently exploiting the sensor resources available as well as gaining information not directly reported by the individual sensor themselves, but by appropriate sensor data fusion and prediction. And the final component of the controller the adaptive element, adapts to the uncertainty arising due to imperfectly known actions of the pursuing agent or uncertain environment or both.

We will utilize Deep Markov models to serve as an intelligent force to track hostile manned aircrafts in simulations. Our preliminary examinations in section 4.2 have shown that Deep Markov models have high predictive likelihood and are very accurate and quick in performing inference. Moreover, being a non-parametric approach it would learn escape maneuvers performed by hostile manned aircrafts on the go. Essentially these models would identify escape maneuvers being performed and would be capable of predicting the next maneuver such that the uncertainty in the other agents' actions is significantly reduced. Deep Markov models have the potential to better the recent approach by [39] that uses Genetic Fuzzy Trees (GFT) for Unmanned Combat Aerial Vehicle Control in simulated missions. This is due to several reasons, first the design of any Fuzzy based control requires that all possible tracking as well as escape maneuvers be known in advance to develop the Fuzzy inference system. Second, a GFT cannot detect an unknown

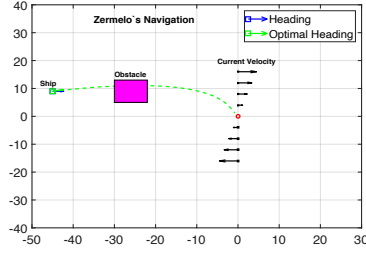


Figure 62: Zermelo's Navigation problem with Off-nominal situation. Human-machine collaborative navigation to origin (red circle).

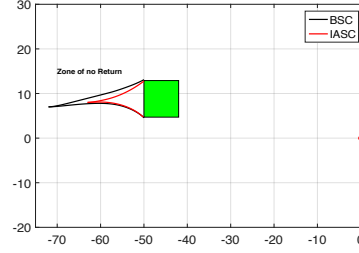


Figure 63: Zone of no Return: Boundary beyond which collision is eminent.

maneuver, and will not be able to autonomously learn the same. Hence the system will neither perform well nor learn in case of a previously unknown maneuver. However, potentially a Deep Markov model can rapidly detect and autonomously learn a new maneuver.

4.4 Collaborative Task Execution: Intent Aware Shared Control

As stated earlier, human-machine shared control is a key component of collaborative task execution. Existing work such as Blended Shared Control (BSC) [38] has established that the shared control can improve cycle times in nominal conditions, that is, when the operating environment satisfies the assumptions made in the design of the optimal augmenting controller. However, these methods can be too slow to relinquish control in off-nominal cases, especially in the case where an operator needs to deviate from the nominally optimal trajectory due to unforeseen obstacles or other uncertainties. Hence the objective was to design and investigate shared control technique that relinquish control in off-nominal situations. We have proposed in [102] a novel technique namely "Intent Aware Shared Control", to addresses this gap by utilizing Joystick measurements to infer operator intent. The presented technique provides autonomous control augmentation to the operator when they are attempting to drive the system along the suggested optimal trajectory, but offers little hindrance to the operator when they are attempting to deal with off-nominal conditions. The proposed technique was evaluated using the Zermelo's Navigation problem. In Zermelo's problem (figure 62), the task for the human-machine team is to navigate a ship to a desired location in the presence of varying currents. In figure 62 the operator along with the autonomous agent controls the ship heading (shown in blue). In doing so, the autonomous agent is aware of the optimal path shown in green not known to the operator. Off-nominal condition is emulated using suddenly appearing obstacle in the path of the ship. The random obstacle is perceived by the operator but remains un-known to the autonomous agent. Overall performance of the shared control technique was judged based on the time required to navigate to the desired location while avoiding collision with the obstacle.

Theoretical results [102] show that the performance of the presented intent aware shared control technique is at least as good as existing techniques in terms of improving the operator's performance, and that it results in improved obstacle reaction time. This was also reinstated by our simulation results depicted using zone of no-return shown in figure 63. Thus the human operator gets more time to avoid an obstacle with Intent aware shared control mechanism, due to its smaller zone of no-return. Over 50 Human-machine interaction experiments were conducted for Zermelo's navigation in the presence of a random pop-up obstacle. Results show a significant

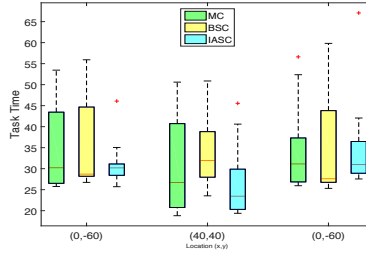


Figure 64: Boxplot comparison of completion times from three different starting locations, for Manual Control (MC), Blended shared control (BSC) and Intent Aware Blended shared control.

reduction in obstacle collision with our method when compared to existing work (see Table 6). Moreover the time-to-destination or completion time in figure 64 indicate superior performance of the proposed shared control approach.

Table 6: Obstacle Collision: Design of IASC controller is responsive to off-nomial situations

Location (x, y)	Obstacle (x, y)	Manual Control	BSC	IASC
(0, -60)	(-45, 8)	1	10	2
(40, 40)	(25, -8)	0	4	3
(0, -60)	(-75, 3)	0	7	1

5 Ongoing work in robust inference and Multi-Agent Simulations

5.1 Multi-Agent Simulation Environment MAGE

The Multi Agent Game Emulator developed by Chowdhary’s group can enable multiple human and computer controller aircraft to simultaneously interact with each other in a cloud based environment. It is built upon the open-source platform flight-gear. We have developed custom interfaces to control the multi-agent interface in flight gear, to transmit data, and to communicate with common ground control stations through the MAVLINK protocol. This has led to a unique highly capable multi-agent flight simulation environment. However, we since realized through feedback from the research community, that the community is better served by integrating the work we have done with existing open source multi-agent environments. Hence, our main contribution this year has been to tie the simulation environment tightly with the Robot Operating System’s GAZEBO multi-agent simulation engine. For this purpose, we teamed up with Dr. Christopher Crick, an expert on ROS and GAZEBO. This work is leading to a commonly usable multi agent flight simulator that conforms to best industry practices, and will be

available to the community for research. The software as it stands is being hosted on the GIT repository: <https://github.com/okstate-robotics/mage>. [80](#)

References

- [1] Pieter Abbeel and Andrew Y Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 1–8. ACM, 2005.
- [2] Amir Ahmadi-Javid. Entropic value-at-risk: A new coherent risk measure. *Journal of Optimization Theory and Applications*, 155(3):1105–1123, 2012.
- [3] Amir Ahmadi-Javid and Malihe Fallah-Tafti. Portfolio optimization with entropic value-at-risk. *arXiv preprint arXiv:1708.05713*, 2017.
- [4] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew E Taylor. Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In *Proc. of AAAI*, 2015.
- [5] David Applebaum. Lévy processes: From probability to finance and quantum groups. *Notices of the AMS*, 51(11):1336–1347, 2004.
- [6] Sahil Garg Ayanian and Nora. Persistent monitoring of stochastic spatio-temporal phenomena with a small team of robots. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.
- [7] Arindam Banerjee, Xin Guo, and Hui Wang. On the optimality of conditional expectation as a bregman predictor. *IEEE Transactions on Information Theory*, 51(7):2664–2669, 2005.
- [8] Nicole Bäuerle and Alfred Müller. Stochastic orders and risk measures: consistency and bounds. *Insurance: Mathematics and Economics*, 38(1):132–148, 2006.
- [9] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyperparameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc., 2011.
- [10] Joseph Berkovitz. Action at a distance in quantum mechanics. 2007.
- [11] Paul Berrisford, DPKF Dee, K Fielding, M Fuentes, P Kallberg, S Kobayashi, and S Upala. The era-interim archive, 2009.
- [12] Lars Blackmore and Masahiro Ono. Convex chance constrained predictive control without sampling. In *AIAA Guidance, Navigation, and Control Conference*, Guidance, Navigation, and Control and Co-located Conferences. American Institute of Aeronautics and Astronautics, 2009.
- [13] Kang Boda and Jerzy A Filar. Time consistent dynamic risk measures. *Mathematical Methods of Operations Research*, 63(1):169–186, 2006.
- [14] Peter Bodik, Wei Hong, Carlos Guestrin, Sam Madden, Mark Paskin, and Romain Thibaux. Intel lab data. Technical report, Intel Berkely Research Lab, Feb 2004.

- [15] Aseem V Borkar, Vivek S Borkar, and Arpita Sinha. Vector field guidance for convoy monitoring using elliptical orbits. In *56th IEEE Conference on Decision and Control (CDC)*, pages 918–924. IEEE, 2017.
- [16] Aseem V Borkar, Vivek S Borkar, and Arpita Sinha. Aerial monitoring of slow moving convoys using elliptical orbits. *European Journal of Control*, 46:90–102, 2019.
- [17] Aseem V Borkar, Swaroop Hangal, Hemendra Arya, Arpita Sinha, and Leena Vachhani. Reconfigurable formations of quadrotors on lissajous curves for surveillance applications. *European Journal of Control*, 2020.
- [18] Chassaing P.H.H.M. Braza, M. and H.H. Minh. Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. *Journal of fluid mechanics*, 165(130), 1986.
- [19] Joshua B. Tenenbaum Samuel J. Gershman Brenden M. Lake, Tomer D. Ullman. Building machines that learn and think like people. *CoRR*, abs/1604.00289, 2016.
- [20] Haim Brezis. *Functional analysis, Sobolev spaces and partial differential equations*. Springer Science & Business Media, 2010.
- [21] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [22] P.A. Bromiley. Products and Convolutions of Gaussian Distributions. Internal Report 2003-003, TINA Vision, 2003.
- [23] Marko Budisic, Ryan Mohr, and Igor Mezic. Applied koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510, 2012.
- [24] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*, 2016.
- [25] Alexandra Carpentier, Alessandro Lazaric, Mohammad Ghavamzadeh, Rémi Munos, and Peter Auer. Upper-confidence-bound algorithms for active learning in multi-armed bandits. In *Algorithmic Learning Theory*, pages 189–203. Springer, 2011.
- [26] Yin-Lam Chow and Marco Pavone. Stochastic optimal control with dynamic, time-consistent risk constraints. In *American Control Conference (ACC), 2013*, pages 390–395. IEEE, 2013.
- [27] Yinlam Chow and Mohammad Ghavamzadeh. Algorithms for cvar optimization in mdps. In *Advances in neural information processing systems*, pages 3509–3517, 2014.
- [28] Girish Chowdhary, Tongbin Wu, Mark Cutler, and Jonathan P How. Rapid transfer of controllers between uavs using learning-based adaptive control. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5409–5416. IEEE, 2013.
- [29] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Hoboken, NJ: Wiley-Interscience, 2006.
- [30] Noel Cressie and Christopher K Wikle. Space-time kalman filter. *Encyclopedia of environmental metrics*, 2002.

- [31] Lehel Csatő and Manfred Opper. Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668, 2002.
- [32] Lehel Csatő and Manfred Opper. Sparse Online Gaussian Processes. *Neural Comput.*, 14(3):641–668, Mar 2002.
- [33] Michel Denuit, Jan Dhaene, Marc Goovaerts, and Rob Kaas. *Actuarial theory for dependent risks: measures, orders and models*, pages 150–152. John Wiley & Sons, 2006.
- [34] Michel Denuit, Jan Dhaene, Marc Goovaerts, and Rob Kaas. *Actuarial theory for dependent risks: measures, orders and models*, pages 105–106. John Wiley & Sons, 2006.
- [35] Michel Denuit, Jan Dhaene, Marc Goovaerts, and Rob Kaas. *Actuarial theory for dependent risks: measures, orders and models*, page 149. John Wiley & Sons, 2006.
- [36] Huanyu Ding and David A Castañón. Optimal solutions for adaptive search problems with entropy objectives. *arXiv preprint arXiv:1508.04127*, 2015.
- [37] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.
- [38] Aaron Enes and Wayne Book. Blended shared control of zermelo’s navigation problem. In *American Control Conference (ACC), 2010*, pages 4307–4312. IEEE, 2010.
- [39] N Ernest, D Carroll, C Schumacher, M Clark, K Cohen, et al. Genetic fuzzy based artificial intelligence for unmanned combat aerial vehicle control in simulated air combat missions. *J Def Manag*, 6(144):2167–0374, 2016.
- [40] Hans Föllmer and Alexander Schied. *Stochastic finance: an introduction in discrete time*, page 456. Walter de Gruyter, 2011.
- [41] Hans Föllmer and Alexander Schied. *Stochastic finance: an introduction in discrete time*, pages 199–201. Walter de Gruyter, 2011.
- [42] Hans Föllmer and Alexander Schied. *Stochastic finance: an introduction in discrete time*, page 178. Walter de Gruyter, 2011.
- [43] Hans Föllmer and Alexander Schied. *Stochastic finance: an introduction in discrete time*, pages 458–463. Walter de Gruyter, 2011.
- [44] Hans Föllmer and Alexander Schied. *Stochastic finance: an introduction in discrete time*, page 262. Walter de Gruyter, 2011.
- [45] Hans Föllmer and Alexander Schied. *Stochastic finance: an introduction in discrete time*, page 465. Walter de Gruyter, 2011.
- [46] Hans Föllmer and Alexander Schied. *Stochastic finance: an introduction in discrete time*, page 466. Walter de Gruyter, 2011.
- [47] Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. The sticky hdp-hmm: Bayesian nonparametric hidden markov models with persistent states. *Arxiv preprint*, 2007.
- [48] Eric Frew and Dale Lawrence. Cooperative stand-off tracking of moving targets by a team of autonomous aircraft. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 6363. AIAA, 2005.

- [49] Eric W Frew. Cooperative standoff tracking of uncertain moving targets using active robot networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3277–3282. IEEE, 2007.
- [50] Eric W Frew, Dale A Lawrence, and Steve Morris. Coordinated standoff tracking of moving targets using Lyapunov guidance vector fields. *AIAA Journal of Guidance, Control, and Dynamics*, 31(2):290–306, 2008.
- [51] Sylvia Fruhwirth-Schnatter. *Finite mixture and Markov switching models*. Springer Science & Business Media, 2006.
- [52] Kevin S Galloway and Biswadip Dey. Station keeping through beacon-referenced cyclic pursuit. In *IEEE American Control Conference (ACC)*, pages 4765–4770. IEEE, 2015.
- [53] Javier Garcia and Fernando Fernandez. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.
- [54] Sahil Garg, Amarjeet Singh, and Fabio Ramos. Learning non-stationary space-time models for environmental monitoring. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.
- [55] Clement Gehring and Doina Precup. Smart exploration in reinforcement learning using absolute temporal difference errors. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1037–1044. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
- [56] Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *J. Artif. Int. Res.*, 24(1):81–108, July 2005.
- [57] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian data analysis*. CRC press, 3rd edition, 2013.
- [58] Alborz Geramifard, Thomas J. Walsh, Stefanie Tellex, Girish Chowdhary, Nicholas Roy, and Jonathan P. How. A tutorial on linear function approximators for dynamic programming and reinforcement learning. *Found. Trends Mach. Learn.*, 6(4):375–451, December 2013.
- [59] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [60] Abhijit Gosavi. Variance-penalized markov decision processes: Dynamic programming and reinforcement learning techniques. *International Journal of General Systems*, 43(6):649–669, 2014.
- [61] Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Advances in neural information processing systems*, pages 3338–3346, 2014.
- [62] Bruce Hajek. *Random processes for engineers*, page 82. Cambridge University Press, 2015.
- [63] David Harrison and Daniel L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, 1978.
- [64] John Haslett and Adrian E. Raftery. Ireland wind data set. Technical report, Trinity College and University of Washington, 1961-1978.

- [65] Frank S He, Yang Liu, Alexander G Schwing, and Jian Peng. Learning to play in a day: Faster deep reinforcement learning by optimality tightening. *arXiv preprint arXiv:1611.01606*, 2016.
- [66] James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *AISTATS*, volume 38 of *JMLR Workshop and Conference Proceedings*, 2015.
- [67] James Hensman, Alexander G Matthews, Maurizio Filippone, and Zoubin Ghahramani. Mcmc for variationally sparse gaussian processes. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1648–1656. Curran Associates, Inc., 2015.
- [68] José Miguel Hernández-Lobato, Michael A Gelbart, Ryan P Adams, Matthew W Hoffman, and Zoubin Ghahramani. A general framework for constrained bayesian optimization using information-based search. *arXiv preprint arXiv:1511.09422*, 2015.
- [69] Christopher Charles Heyde and Eugene Seneta. Studies in the history of probability and statistics. XXXI. the simple branching process, a turning point test and a fundamental inequality: A historical note on I.J. bienaymé. *Biometrika*, 59(3):680–683, 1972.
- [70] Arthur Hobson. A new theorem of information theory. *Journal of Statistical Physics*, 1(3):383–391, 1969.
- [71] Arthur Hobson and Bin-Kang Cheng. A comparison of the shannon and kullback information measures. *Journal of Statistical Physics*, 7(4):301–310, 1973.
- [72] Wolfgang Hönig and Nora Ayanian. *Flying Multiple UAVs Using ROS*, pages 83–118. Springer International Publishing, 2017.
- [73] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.
- [74] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.
- [75] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994.
- [76] Wassim Jouini, Damien Ernst, Christophe Moy, and Jacques Palicot. Upper confidence bound based decision making strategies and dynamic spectrum access. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5. IEEE, 2010.
- [77] Pasi Jylänki, Jarno Vanhatalo, and Aki Vehtari. Robust Gaussian Process Regression with a Student-t likelihood. *J. Mach. Learn. Res.*, 12:3227–3257, Nov 2011.
- [78] Rob Kaas, Marc Goovaerts, Jan Dhaene, and Michel Denuit. *Modern actuarial risk theory: using R*, volume 128, page 228. Springer Science & Business Media, 2008.
- [79] Yuri A Kapitanyuk, Hector Garcia de Marina, Anton V Proskurnikov, and Ming Cao. Guiding vector field algorithm for a moving path following problem. *20th IFAC World Congress*, 50(1):6983–6988, 2017.

- [80] Hassan Kingravi, Harshal Maske, and Girish Chowdhary. Kernel observers: Systems theoretic modeling and inference of spatiotemporally varying processes. In *Advances in Neural Information Processing Systems (NIPS)*, Barcelona, Spain, 2016.
- [81] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.
- [82] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, 9:235–284, 2008.
- [83] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.
- [84] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [85] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, pages 79–86, 1951.
- [86] M. Kuss. *Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning*. PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany, Mar 2006.
- [87] DA Lawrence. Lyapunov vector fields for UAV flock coordination. In *AIAA 2nd Unmanned Unlimited Conference and Workshop, & Exhibit*, page 6575. AIAA, 2003.
- [88] Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the International Conference on Machine Learning*, 2013.
- [89] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.
- [90] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. *arXiv preprint arXiv:1603.01312*, 2016.
- [91] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.
- [92] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.
- [93] Lixin Li, Xingyou Zhang, and Reinhard Piltner. A spatiotemporal database for ozone in the conterminous us. In *Thirteenth International Symposium on Temporal Representation and Reasoning (TIME’06)*, pages 168–176. IEEE, 2006.
- [94] Pu Li, Moritz Wendt, and Günter Wozny. Robust model predictive control under chance constraints. *Computers & Chemical Engineering*, 24(2):829–834, 2000.
- [95] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

- [96] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.
- [97] Daniel Y Little and Friedrich T Sommer. Learning and exploration in action-perception loops. *Frontiers in neural circuits*, 7, 2013.
- [98] Lili Ma. Cooperative target tracking in elliptical formation. In *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 58–63. IEEE, 2019.
- [99] Lili Ma and Naira Hovakimyan. Cooperative target tracking in balanced circular formation: Multiple UAVs tracking a ground vehicle. In *IEEE American Control Conference (ACC)*, pages 5386–5391. IEEE, 2013.
- [100] Lili Ma and Naira Hovakimyan. Cooperative target tracking with time-varying formation radius. In *IEEE European Control Conference (ECC)*, pages 1699–1704. IEEE, 2015.
- [101] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Fei-Fei Li, and Silvio Savarese. Adversarially robust policy learning: Active construction of physically-plausible perturbations. *IEEE International Conference on Intelligent Robots and Systems (to appear)*, 2017.
- [102] Harshal Maske, Girish Chowdhary, and Prabhakar Pagilla. Intent aware shared control in off-nominal situations. In *Conference on Decisions and Control*, Las Vegas, USA, 2016. IEEE. accepted.
- [103] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.
- [104] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- [105] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.
- [106] Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.
- [107] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.
- [108] Tagir Z Muslimov and Rustem A Munasypov. Adaptive decentralized flocking control of multi-uav circular formations based on vector fields and backstepping. *ISA transactions*, pages S0019–0578.
- [109] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.
- [110] Radford M. Neal. Regression and Classification Using Gaussian Process Priors. In A. P. David J. M. Bernardo, J. O. Berger and A. P. M. Smith, editors, *Bayesian Statistics 6*, pages 475–501. Oxford University Press, 1998.

- [111] Derek R Nelson, D Blake Barber, Timothy W McLain, and Randal W Beard. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529, 2007.
- [112] Arkadi Nemirovski and Alexander Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2007.
- [113] H. Nickisch and CE. Rasmussen. Approximations for Binary Gaussian Process classification. *Journal of Machine Learning Research*, 9:2035–2078, Oct 2008.
- [114] Hyondong Oh, Seungkeun Kim, Hyo-sang Shin, and Antonios Tsourdos. Coordinated standoff tracking of moving target groups using multiple UAVs. *IEEE Transactions on Aerospace and Electronic Systems*, 51(2):1501–1514, 2015.
- [115] Manfred Opper. Online versus Offline Learning from Random Examples: General Results. *Phys. Rev. Lett.*, 77:4671–4674, 1996.
- [116] Manfred Opper. A Bayesian Approach to Online Learning. In *Online Learning in Neural Networks*, pages 363–378. Cambridge University Press, 1998.
- [117] Vivek Padmaji, Arpita Sinha, and Hemendra Arya. Time to capture target using cyclic pursuit strategy. In *Infotech@ Aerospace 2012*, page 2526. 2012.
- [118] Derek Paley, Naomi Ehrich Leonard, and Rodolphe Sepulchre. Collective motion: Bistability and trajectory tracking. In *43rd IEEE Conference on Decision and Control (CDC)*, volume 2, pages 1932–1937. IEEE, 2004.
- [119] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.
- [120] Tobias Pfingsten, Malte Kuss, and Carl Edward Rasmussen. Nonstationary gaussian process regression using a latent extension of the input space. URL <http://www.kyb.mpg.de/~tpfingst>, 2006.
- [121] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. *arXiv preprint arXiv:1703.02702*, 2017.
- [122] Matthias Plappert. keras-rl. <https://github.com/matthiasplappert/keras-rl>, 2016.
- [123] Abin Alex Pothen and Ashwini Ratnoo. Curvature-constrained lyapunov vector field for standoff target tracking. *Journal of Guidance, Control, and Dynamics*, 40(10):2729–2736, 2017.
- [124] Jose C Principe and Dongxin Xu. Information-theoretic learning using renyi’s quadratic entropy. In *Proceedings of the First International Workshop on Independent Component Analysis and Signal Separation, Aussois*, pages 407–412, 1999.
- [125] J. R. Quinlan. Combining Instance-Based and Model-Based Learning. pages 236–243. Morgan Kaufmann, 1993.
- [126] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, pages 1177–1184, 2007.

- [127] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008.
- [128] Kandasamy A. Rajani, B.N. and S. Majumdar. Numerical simulation of laminar flow past a circular cylinder. *Applied Mathematical Modelling*, 33(3):1228–1247, 2009.
- [129] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.
- [130] Jette Randlov and Preben Alstrom. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 463–471, 1998.
- [131] Carl Rasmussen and Chris Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.
- [132] Carl Edward Rasmussen. Gaussian processes for machine learning. Technical report, Max Planck Institute for Biological Cybernetics, 2006.
- [133] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.
- [134] Paul Reverdy, Robert C. Wilson, Philip Holmes, and Naomi Ehrich Leonard. Towards optimization of a human-inspired heuristic for solving explore-exploit problems. In *CDC*, pages 2820–2825, 2012.
- [135] Frank Riedel. Dynamic coherent risk measures. *Stochastic processes and their applications*, 112(2):185–200, 2004.
- [136] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, December 2004.
- [137] Cyril Robin and Simon Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40(4):729–760, 2016.
- [138] Jean-Louis Roch and Gilles Villard. Fast parallel computation of the jordan normal form of matrices. *Parallel processing letters*, 6(02):203–212, 1996.
- [139] R Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.
- [140] A. Roshko. On the development of turbulent wakes from vortex streets. *California Institute of Technology*, Report 1191, 1954.
- [141] Dan Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. In *Advances in Neural Information Processing Systems*, pages 1583–1591, 2014.
- [142] Daniel Russo and Benjamin Van Roy. An information-theoretic analysis of thompson sampling. *arXiv preprint arXiv:1403.5341*, 2014.
- [143] Daniil Ryabko. Time-series information and learning. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 1392–1395. IEEE, 2013.

- [144] Adam N Sanborn, Vikash K Mansinghka, and Thomas L Griffiths. Reconciling intuitive physics and newtonian mechanics for colliding objects. *Psychological review*, 120:411, 2013.
- [145] John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897, 2015.
- [146] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 2001.
- [147] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.
- [148] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyia, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.
- [149] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In Y. Weiss, P. B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press, 2006.
- [150] Edward Snelson and Zoubin Ghahramani. Local and global sparse gaussian process approximations. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2, pages 524–531, San Juan, Puerto Rico, 21–24 Mar 2007. PMLR.
- [151] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012.
- [152] Charles M. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9:1135–1151, 1981.
- [153] Arne Storjohann. An $O(n^3)$ algorithm for the frobenius normal form. In *Proceedings of the 1998 international symposium on Symbolic and algebraic computation*, pages 101–105. ACM, 1998.
- [154] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- [155] Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the cvar via sampling. In *AAAI*, pages 2993–2999, 2015.
- [156] Matthew E Taylor and Peter Stone. Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th international conference on Machine learning*, pages 879–886. ACM, 2007.
- [157] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.
- [158] Matthew E Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(Sep):2125–2167, 2007.

- [159] Michalis K. Titsias and Neil D. Lawrence. Bayesian gaussian process latent variable model. In Yee Whye Teh and D. Mike Titterton, editors, *AISTATS*, volume 9 of *JMLR Proceedings*, pages 844–851. JMLR.org, 2010.
- [160] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012.
- [161] Michel Tokic. Adaptive ε -greedy exploration in reinforcement learning based on value differences. In *KI 2010: Advances in Artificial Intelligence*, pages 203–210. Springer, 2010.
- [162] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, pages 2094–2100, 2016.
- [163] Jarno Vanhatalo, Pasi Jylänki, and Aki Vehtari. Gaussian process regression with Student-t likelihood. In *Advances in Neural Information Processing Systems 22*, pages 1910–1918. Curran Associates, Inc., 2009.
- [164] Jarno Vanhatalo, Riihimäki, Jouni Hartikainen, Jylänki, Ville Tolvanen, and Aki Vehtari. Gpstuff: Bayesian Modeling with Gaussian Processes. *Journal of Machine Learning Research*, 14:1175–1179, Apr 2013.
- [165] Jarno Vanhatalo and Aki Vehtari. Modelling local and global phenomena with sparse gaussian processes. In David A. McAllester and Petri Myllymäki, editors, *UAI*, pages 571–578. AUAI Press, 2008.
- [166] Chang Wang and Sridhar Mahadevan. Manifold alignment without correspondence. In *IJCAI*, volume 2, page 3, 2009.
- [167] Martin Widmann and Christopher S Bretherton. Validation of mesoscale precipitation in the ncep reanalysis using a new gridcell dataset for the northwestern united states. *Journal of Climate*, 13(11):1936–1950, 2000.
- [168] Christopher K. I. Williams and Matthias Seeger. Using the Nyström Method to Speed Up Kernel Machines. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.
- [169] W Murray Wonham. *Linear multivariable control*. Springer, 1974.
- [170] Zichao Yang, Andrew Wilson, Alex Smola, and Le Song. {A la Carte-Learning Fast Kernels}. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 1098–1106, 2015.
- [171] Mingfeng Zhang and Hugh HT Liu. Cooperative tracking of a moving target using multiple fixed-wing UAVs. *Journal of Intelligent & Robotic Systems*, 81(3-4):505–529, 2016.
- [172] Bingyu Zhou, Harish Satyavada, and Simone Baldi. Adaptive path following for unmanned aerial vehicles in time-varying unknown wind environments. In *IEEE American Control Conference (ACC)*, pages 1127–1132. IEEE, 2017.
- [173] Kemin Zhou, John C. Doyle, and Keith Glover. *Robust and Optimal Control*. Prentice Hall, Upper Saddle River, NJ, 1996.