MaP, CaP, RaLly! Hybrid Architecture for Planning and Control of UGV in Stochastic Environments

Nathaniel Snyder
Dept. of MAE
UCLA
natsnyder1@gmail.com
UID: 205029016

Brian Wang
NESL, Dept. of CS
UCLA
wangbri1@g.ucla.edu
UID: 605229631

Swapnil Sayan Saha
NESL, Dept. of ECE
UCLA
swapnilsayan@g.ucla.edu
UID: 605353215

1 Introduction

Level 5 (SAE) autonomous vehicles have been touted as the upcoming digital disruption and a physical technological driver of the fourth industrial revolution [1][2]. While societal acceptance and confidence in self-driving technologies are commonplace [3], consumers manifest safety and reliability concerns [1][3]. Implanting real-time situational awareness in stochastic environments embracing probable edge cases entails computational and implementation complexities, exacerbated by the absence of truly dynamic training frameworks and datasets [4] and thus hurdling the advent of level 5 UGV. In particular, real-time trajectory management in UGV in the presence of a large number of stochastic and dynamic obstacles is NP-Hard and in PSPACE [5]. Traditional collision avoidance strategies include reactive navigation, which suffers from sub-optimal and understated trajectory projections without probabilistic guarantees of fulfilling the end goal, and deliberate navigation (pre-planned), which is more likely to converge to global optima but is unable to handle dynamic deviations in a priori environmental belief states [6].

Keeping the challenges in mind, various RL techniques exploiting Q-learning [7][8] and actor-critic methods [5][6][9][10][11] have emerged and have been shown to exhibit superior and promising performance characteristics over formal methods [5][8][9][10] with smaller dependence on belief states in stochastic environments. However, these end-to-end agents are often unable to guarantee convergence to optimal trajectories in the long run and are difficult to train, suffering from reward sparsity, tedious training phase, unpredictable / unsafe agent behavior, hyperparameter sensitivity, low generalizability and divergence in large and complex maps [8][10][12]. In response, a small number of articles attempt to illustrate the benefits of coupling the ability of RL to handle uncertainties with the convergence guarantees of formal methods (e.g. model predictive control (MPC) or probabilistic roadmaps (PRM)) in the context of intelligent transportation and robot locomotion, notably [9][10][12][13][14][15]. [9][12] and [14] assume obstacles are either static or have a predictable motion profile, [13] is task-specific (can only be applied in cruise-control scenarios) and [15] guarantees safety during exploration but no exploitation convergence guarantees. [10] fits the problem statement profile, coupling actor-critic methods from [11] with PRM as obstacle avoiding node and planning node respectively. However, the autonomous training framework requires considerable epochs to pass in order to obtain optimal training data. As a result, imitation learning [8] can ease the training process, with online oracles supplying useful examples covering edge cases.

Motivated from [8] and [10], we are developing **M**a**P**, **C**a**P**, **R**a**L**ly in the ROS-Gazebo testbed, an end-to-end hybrid UGV controller using a **M**odel **P**redictive **C**ontrol (MPC) oracle, which exploits a LiDAR Simulataneous Localization and Mapping (SLAM) to plan, optimize and fulfill user-defined navigation goals, while using a **R**einforcement **L**earning agent to handle dynamic obstacles in the environment. To achieve near-optimal path planning for an Ackermann-drive UGV, a custom MPC trajectory optimizer node has been developed that tunes the output of classic path-planning algorithms (e.g. Dijkstra) to a near-optimal yet feasible path given the environmental and mechanical constraints.

The path and desired controls are then fed into a custom error-tracking finite horizon LQR, which takes the vehicle to the goal point along the planned path in real-time. For the obstacle node, we adopt imitation learning strategy from [8] to teach a RL agent how to follow projections from a stochastic online oracle [8][16] with agent in the loop. The agent is trained in the presence of obstacles, with the objective of minimizing the L2 norm between MPC projection (near-optimal) and RL projection (safe), allowing oracles to intervene in case of sub-optimal choices. Candidates for the RL node include deep Q networks / Q learning (state-of-the-art for collision avoidance [7]) and actor-critic method [5][11].

2 Preliminary Results

The agent action space includes steering ($-\frac{\pi}{6}^c$ to $\frac{\pi}{6}^c$) and throttle (-0.25 m/s to 1.25 m/s), which has been discretized into buckets as hyperparameters. The observation space for the RL node includes the current location of the car $\{(x_t, y_t, \theta_t), x, y \in \mathbb{R}, \theta \in [-\pi, \pi]\}$, LiDAR scan $L_v \in \mathbb{R}^{1080}$, goal point (x_f, y_f, θ_f) and upcoming look-ahead points from the oracle projections $\{\mathbf{x}, \mathbf{y}, \theta, \mathbf{x}, \mathbf{y} \in \mathbb{R}^{15}\}$. The online training oracles include the LQR tracker for trajectory projection (near-optimal) and an user with a joystick for obstacle avoidance (safe). To generate dynamic obstacles, the human marks average obstacle positions across P2P warehouse trajectories in RViz, while an obstacle manager stochastically moves those blocks using Gaussian process in Gazebo. To initiate training for the RL agent, we have recorded $\sim$ 70 minutes ($\sim$ 2 GB) of static obstacle driving trajectories (which will be extended to dynamic obstacles) from human oracle and $\sim$ 30 minutes ($\sim$ 1 GB) of obstacle-less near-optimal trajectories from LQR controller along those same paths. The goal of the agent is to successfully predict sufficiently safe control actions to navigate through the warehouse with obstacles with attempts to follow the near-optimal path mostly. A video demo of the training data collection is available at: `http://shorturl.at/izCIO`

Initialize replay memory D to capacity N;
Initialize action-value function Q with random weights θ ;
Initialize target action-value function $\hat{\text{Q}}$ with weights $\hat{\theta} = \theta$;
for *episode = 1, M* **do**
  for *steps = 1,T* **do**
    if *D.length* $>$ *threshold* **then**
      sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D ;
      if *path terminates at* $j + 1$ **then**
        $q_j = r_j$;
      else
        $q_j = r_j + \gamma \max_a \hat{Q}(\phi_{j+1}, a; \hat{\theta})$;
      end
      $y_j = \hat{Q}(\phi_j; \hat{\theta})$;
      $y_j[a_j] = q_j$;
      Perform gradient descent on $(y_j - Q(\phi_j; \hat{\theta}))^2$ w.r.t. θ ;
      Every C steps set $\hat{\theta} = \theta$;
    else
      store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D ;
    end
  end
end

Layer (type)	Param #	Activation	Out. Shape
Conv. 1	512	ReLu	(1109, 128)
Max. Pool. 1			(554, 128)
B. Norm 1	512		(554,128)
Conv. 2	24640	ReLu	(552,64)
Max. Pool. 2			(276, 64)
B. Norm 2	256		(276,64)
Conv. 3	6176	ReLu	(274,32)
Max. Pool. 3			(137, 32)
B. Norm 3	128		(137,32)
LSTM	295936		(256)
Dense	12593	Linear	(49)

Figure 1: (Left): Implemented DQN algorithm (Right): Sample target network architecture.

For the target network, we implemented deep convolutional long short-term memory (LSTM). motivated from [17]. Convolutional layers are able to extract differential patterns by enforcing a degree of local connectivity among adjacent samples and enabling scale invariance (account for noisy feature discrepancy among similar state-action pairs), making the architecture suitable as a high-dimensional feature extractor. Recurrent layers are able to learn temporal dynamics of feature activations, inferring temporal dependencies [17]. The reward increases linearly as the agent gets close to the goal point with 60% dropout probability. We keep the network architecture and reward function as hyperparameters. Other hyperparameters we intend to explore include target network update frequency, gradient and reward clipping, experience replay density and no. of episodes / iterations. Using the sample framework, in preliminary evolution, we observed converge of Q-mean absolute error within $\sim$ 100 episodes (100 iterations per episode) to around 1.5. DQN continuously updates the Q value for a particular action taken in a path, with the goal of minimizing the measured (optimal) vs predicted Q value. Since exploration is not necessary and the examples resemble near-optimal trajectories, we see fast convergence.

References

[1]. Garidis, Konstantin, et al. *Toward a User Acceptance Model of Autonomous Driving.* Proceedings of the 53rd Hawaii International Conference on System Sciences. 2020.

[2]. Li, Guoping, Yun Hou, and Aizhi Wu. *Fourth Industrial Revolution: Technological Drivers, Impacts and Coping Methods.* Chinese Geographical Science 27.4 (2017): 626-637.

[3]. Wintersberger, Sophie, Muhammad Azmat, and Sebastian Kummer. *Are We Ready to Ride Autonomous Vehicles? A Pilot Study on Austrian Consumers' Perspective*. Logistics 3.4 (2019): 20.

[4]. Hussain, Rasheed, and Sherali Zeadally. *Autonomous cars: Research results, issues, and future challenges.* IEEE Communications Surveys & Tutorials 21.2 (2018): 1275-1313.

[5]. A. Garg, H. L. Chiang, S. Sugaya, A. Faust and L. Tapia, *Comparison of Deep Reinforcement Learning Policies to Formal Methods for Moving Obstacle Avoidance,* 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 2019, pp. 3534-3541.

[6]. E. Meyer, H. Robinson, A. Rasheed and O. San, *Taming an Autonomous Surface Vehicle for Path Following and Collision Avoidance Using Deep Reinforcement Learning,* in IEEE Access, vol. 8, pp. 41466-41481, 2020.

[7]. M. S. Shim and P. Li, *Biologically inspired reinforcement learning for mobile robot collision avoidance,* 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, 2017, pp. 3098-3105.

[8]. Y. Pan et al. *Agile Autonomous Driving using End-to-End Deep Imitation Learning.* Robotics: Science and Systems (RSS). Pittsburgh, PA, USA, 2018.

[9]. A. Faust et al., *PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-Based Planning,* 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, 2018, pp. 5113-5120.

[10]. A. Francis et al., *Long-Range Indoor Navigation With PRM-RL,* in IEEE Transactions on Robotics (2020).

[11]. H. L. Chiang, A. Faust, M. Fiser and A. Francis, *Learning Navigation Behaviors End-to-End With AutoRL*, in IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 2007-2014, April 2019

[12]. C. Greatwood and A. G. Richards. *Reinforcement learning and model predictive control for robust embedded quadrotor guidance and control.* Autonomous Robots 43.7 (2019): 1681-1693.

[13]. N. K. Ure, M. U. Yavas, A. Alizadeh and C. Kurtulus, *Enhancing Situational Awareness and Performance of Adaptive Cruise Control through Model Predictive Control and Deep Reinforcement Learning,* 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 2019, pp. 626-631.

[14]. T. Tram, I. Batkovic, M. Ali and J. Sjöberg, *Learning When to Drive in Intersections by Combining Reinforcement Learning and Model Predictive Control,* 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 2019, pp. 3263-3268.

[15]. T. Koller, F. Berkenkamp, M. Turchetta and A. Krause, *Learning-Based Model Predictive Control for Safe Exploration*, 2018 IEEE Conference on Decision and Control (CDC), Miami Beach, FL, 2018, pp. 6059-6066.

[16]. C-A Cheng et al. *Accelerating Imitation Learning with Predictive Models.* Proceedings of Machine Learning Research 89: 3187-3196 (2019).

[17]. S. S. Saha, S. S. Sandha and M. Srivastava, *Deep Convolutional Bidirectional LSTM for Complex Activity Recognition with Missing Data*, (to appear) in Activity and Behavior Computing - Smart Innovations, Systems and Technologies, Springer (2020).