Deep-Sarsa Based Multi-UAV Path Planning and Obstacle Avoidance in a Dynamic Environment

Chapter *in* Lecture Notes in Computer Science · June 2018

DOI: 10.1007/978-3-319-93818-9_10

Deep-Sarsa based Multi-UAV Path Planning and Obstacle Avoidance in a Dynamic Environment

Wei Luo[1], Qirong Tang*[2], Changhong Fu[2], and Peter Eberhard[1]

[1] Institute of Engineering and Computational Mechanics, University of Stuttgart, Pfaffenwaldring 9, 70569 Stuttgart, Germany
[2] Laboratory of Robotics and Multibody System, School of Mechanical Engineering, Tongji University, No. 4800, Cao An Rd., Shanghai 201804, P. R. China
corresponding author qirong.tang@outlook.com

Abstract. This study presents a Deep-Sarsa based path planning and obstacle avoidance method for unmanned aerial vehicles (UAVs). Deep-Sarsa is an on-policy reinforcement learning approach, which gains information and rewards from the environment and helps UAV to avoid moving obstacles as well as finds a path to a target based on a deep neural network. It has a significant advantage over dynamic environment compared to other algorithms. In this paper, a Deep-Sarsa model is trained in a grid environment and then deployed in an environment in ROS-Gazebo for UAVs. The experimental results show that the trained Deep-Sarsa model can guide the UAVs to the target without any collisions. This is the first time that Deep-Sarsa has been developed to achieve autonomous path planning and obstacle avoidance of UAVs in a dynamic environment.

Keywords: UAV, Deep-Sarsa, multi-agent, dynamic environment

1 Introduction

Nowadays unmanned aerial vehicles (UAVs) have been applied in many application fields such as cooperative target search [1], mapping [2], goods transportation [3], observation [4] and rescue [5]. To accomplish these missions, UAVs should have the ability to explore and understand the environment, then take safe paths to the target. Besides that, UAV should be able to react to the obstacles in the scenario and avoid them, especially when the environment is dynamic and the obstacles may move in the environment. All these abilities remain as challenges for UAV research.

For path planning, many studies have been carried out for UAVs. Some algorithms, such as A* algorithms [6, 7], artificial potential fields [8], coverage path planning [9], and Q-learning [10, 11] perform well in a static environment. They figure out the path to the target and guide UAVs going through a known environment. However, in most of the cases, the conditions and environment are changing during the mission. For instance, when UAVs shuttle back and forth in the city and transfer goods, they should handle not only static obstacles but

also moving ones. Limited by computational capacity and sensor sampling rate, when UAV uses static path planning methods, the previous experience may not be the best choice, since the situation could be changed, and the best action in the last state may even lead to a dangerous scene. Therefore, a dynamic path planning is more practical for real applications.

In this paper, an on-policy algorithm named Deep-Sarsa is selected for path planning and obstacle avoidance for UAVs in a dynamic environment. It combines traditional Sarsa [12] with a neural network, which takes the place of the Q-table for storing states and predicting the best action [13]. Also, the robot operating system (ROS) [14] and the related simulation platform Gazebo [15] are used in our work, since they provide a simulation platform with a physics engine and the implementation in this simulation platform can also be quickly transferred to real UAVs.

The paper is organized as follows. Section 2 describes the principle of Sarsa and the structure of our trained model. In Section 3, the training process is introduced, and the experiment in the simulation platform is illustrated. Finally, discussions and conclusions are presented in Section 4.

2 Algorithm

Reinforcement learning is one of typical machine learning classes. It is widely used in many robotic applications [16] and obtains rewards from the environment and reinforces the experience. Reinforcement learning algorithms can be divided into two categories: on-policy learning and off-policy learning [17]. Compared with off-policy learning, on-policy learning algorithms gain their experiences during the operation. Therefore, an on-policy algorithm usually has to be more conservative than an off-policy algorithm, since it will not greedily take the maximum reward.

2.1 Sarsa

State-action-reward-state-action (Sarsa) is one of well-known on-policy reinforcement learning algorithms [18]. Exemplary pseudo-code for Sarsa is illustrated in **Algorithm 1**. Similar to Q-Learning, Sarsa requires a table to store Q-values, which indicate the rewards from the environment on the basis of its rules and depend on the individual state s and action a of robots. During the exploration, a robot agent will interact with the environment and get the updated policy on account of its action. The next state s' and action a' will also have a reward based on the previous stored Q-table. To control the learning process, the learning rate α is set up for control of learning speed and the discount factor γ determines the contribution of future rewards.

2.2 Deep Sarsa

The traditional reinforcement learning for path planning usually has a shortcoming since the rules in the certain environment have to be manually constructed.

Algorithm 1 Pseudo-code for Sarsa

initialize $Q(s, a)$ arbitrarily, where s denotes the state of agent and a denotes the action
for each episode **do**
    initialize s
    choose a from s using policy derived from Q
    for each step of episode **do**
        take action a, observe reward r, and next state s'
        choose the next action a' from s' using policy derived from Q
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
        $s \leftarrow s'$; $a \leftarrow a'$
    until s arrives the terminal state

It's a great challenge for the user to find a workable principle to fully describe the connection between the situation of an agent in the environment and the returned reward according to each pair set of state and action. For instance, for the robot path planning, the state usually contains the current position's information of the robot and also other realated information about terrains and targets etc. Hence, there may exist plenty of possible states and the dimension of the Q-table could be extremely large. Besides that, if the environment is changed as the terrains or targets move during the operation, the previously stored Q-values may lead to a wrong action and even can cause a collision. Therefore, the traditional Sarsa algorithm can hardly be applied in a dynamic environment.

In literature, Deep Q-Networks [19] have shown a good performance in playing games, and so researchers realized that deep neural networks could be applied in complex situations and they can discover as well as estimate the non linear connections behind the given data. Hence, one promising solution for dynamic situations could be Deep-Sarsa. Instead of constructing a concrete Q-table and rules for each agent in Sarsa, Deep-Sarsa uses a deep neural network to determine which action the UAV needs to take. Owing to the strong generalization ability to cover different situations, Deep-Sarsa can handle complex state combination, such as information from moving terrains, multi-agent. The structure of Deep Sarsa is illustrated in Fig. 1. The neural network requires only initialization at the beginning, then it learns and understands the environment through the given data from training. Therefore, users can naturally and intuitively define the states of relative position and some motion states as inputs. The output of Deep Sarsa is the 'best' action which is determined by the neural network.

3 Experiment and Simulation

To verify the performance of Deep-Sarsa for UAV's path planning and obstacle avoidance, checking different scenarios in the simulation platform is indispensable. In the experiment, two UAVs in a formation want to pass a small terrain, where two flying obstacles cut their paths and one static obstacle stays in the front of the exit. The simplified environment is illustrated in Fig. 2, where the

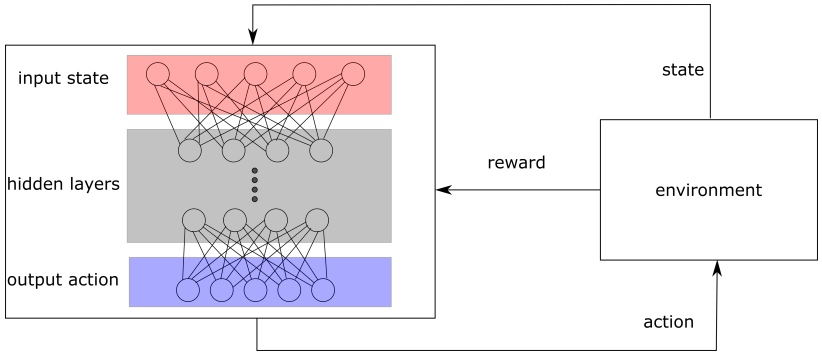


Fig. 1. The structure of Deep-Sarsa

mission UAVs are marked with red circles and the moving obstacles are in the yellow. Besides, two green triangles form a static obstacle and hamper the path to exit which are marked with two blue circles. The UAVs need to figure out the pattern of obstcals' motion and find the exit they need to reach. The rewards of this experiment are defined quite simple, which it is positively defined when the UAVs arrive at the destination and conversely negatively rated in case of any collision with obstacles.

For this scenario, the Deep-Sarsa model should be trained at first and then applied in a simulation environment. In the training phase, the simplified environment is utilized. The agents in training are treated as mass points, used to test the robustness of the algorithm and also generate the trained model. To check the performance of the trained Deep-Sarsa model, a 3D environment is set up in ROS-Gazebo and provides realistic conditions before performaning the experiment on real hardware.

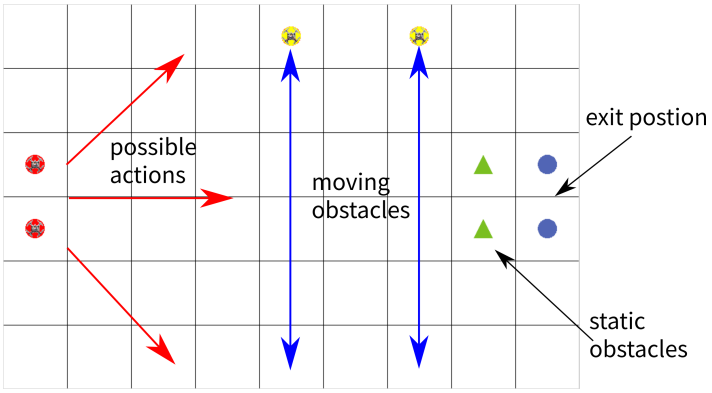


Fig. 2. The simplified training environment

3.1 Training a model

In the beginning of training, UAVs have no knowledge about the environment. To explore the environment, UAVs take action randomly from five different choices, namely up, down, left, right and still, until they have gained enough experience and 'understand' the situation. To balance exploration and safety a decision parameter ϵ is set up, see **Algorithm 2**. In each step, once the UAV takes action and gains the state from the environment, the decision parameter ϵ will be multiplied with a const value λ, which is between zero and one, and reduce the possibility of choosing an action too arbitrary.

Algorithm 2 Action selection strategy

```
for each step of episode in training do
    if random number < ϵ then
        randomly choose an action from up, down, left, right and still
    else
        gain the best action from trained model according to the current state
    ϵ = ϵ * λ, where λ = const
```

Using the **Algorithms 1** and **2**, the flowchart for training a Deep-Sarsa model is illustrated in Fig. 3. In each episode, the current state, the current action and also the next state and action of a UAV will be fed to the Deep-Sarsa model. In this paper, the network used for this scenario is founded through Keras [20] and contains three dense layers with totally 549 trainable parameters. The details about the neural network are illustrated in Table 1. In consideration of real experiments for UAVs the input of the Deep-Sarsa model contains 14 components with the information from relative position between UAV and targets, the relative position between UAV and obstacles, obstacles' moving direction and the rewards. And the output of this Deep-Sarsa model is an array of possibilities for five alternative actions for UAV in each step. The UAV can take its next action consulting the predicted action from the Deep-Sarsa model and also its current state.

Table 1. Neural network layout for Deep-Sarsa

layer index	output shape	number of parameters
1	(21, 16)	352
2	(16, 8)	136
3	(8, 4)	36
4	(4, 5)	25

Based on the training process, 4000 simulation runs are performed to get enough data to train the model. In every episode, if the UAVs successfully arrive

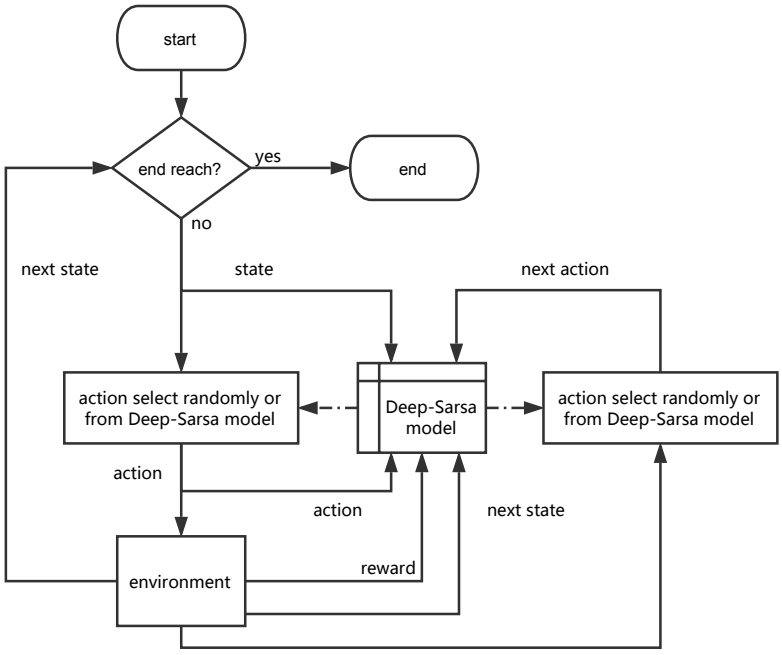


Fig. 3. Flowchart of training a Deep-Sarsa model

in the target zoon, the score will be marked with 1. Otherwise, it will be set to -1 if UAVs crash with obstacles. The sum of all scores is illustrated in Fig. 4. In the first 800 runs, UAVs can hardly get to the target and hit the obstacles. Along with the progress of training, the number of success achieving the goal is increased. On the one hand, the occasionality of action is reduced along with the decreasing values of ϵ. On the other side, the trained model is more robust and can help UAVs steering clear of the terrains and finding the path to the target.

3.2 Test of the trained model

Since the path planning model for UAVs has been trained in a simplified environment, it's necessary to bring the model in a more demanding test environment before implementation on real UAVs. There are two simplifications in the simplified simulation platform. One is using mass points to indicate the UAVs. In the real experiment, the UAVs act under the fundamental physics laws and the fly performance is restricted to motor power and aerodynamics. The other simplication is considering the training state as discrete since the simplified environment is in the grid. The real readings for sensors of UAVs are obviously more consecutive than the states in the simplified environment, and it needs to be proven that the trained model can also be applied in the real environment. Besides, by training, it ignores the delay of communication and information exchanges, which may also cause severe problems in real hardware experiments.

Considering these problems, the test platform in this work is built based on ROS and Gazebo. In Gazebo, a physics engine is included, which can quickly judge the collision between objects during the simulation. When UAVs have an impact on the terrains, they may lead to a crash or change the moving direction. Additionally, a physical model of a UAV is introduced in the simulation. The fly peformance is well simulated, and the controller will not ignore the fly ability of the UAV. The communication between UAVs and trained model is also simulated in the experiment. In the real experiment, the trained model, the UAVs and the server are separated in different network locations according to target requirement. Therefore, the network structure in this work is designed and illustrated in Fig. 5. The bridge between ROS-Gazebo and the trained model is realized through lightweight communications and marshalling (LCM) [21]. It provides a reliable connection and can be deployed not only in the simulation but also for the hardware. During the test, four instances are launched and work parallelly based on the aforementioned structure. Each instance takes charge of one of the operations from the ROS-Gazebo core, broadcasting the state of simulation, Deep-Sarsa model and transmitting the predicted action.

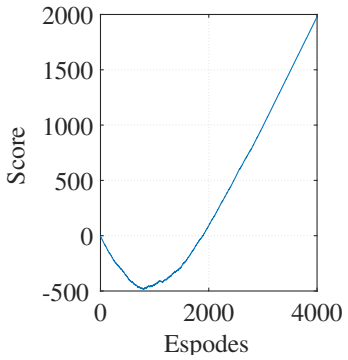


Fig. 4. Training scores of Deep-Sarsa

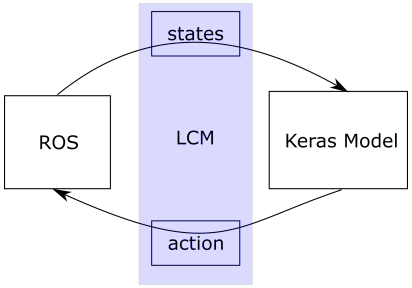
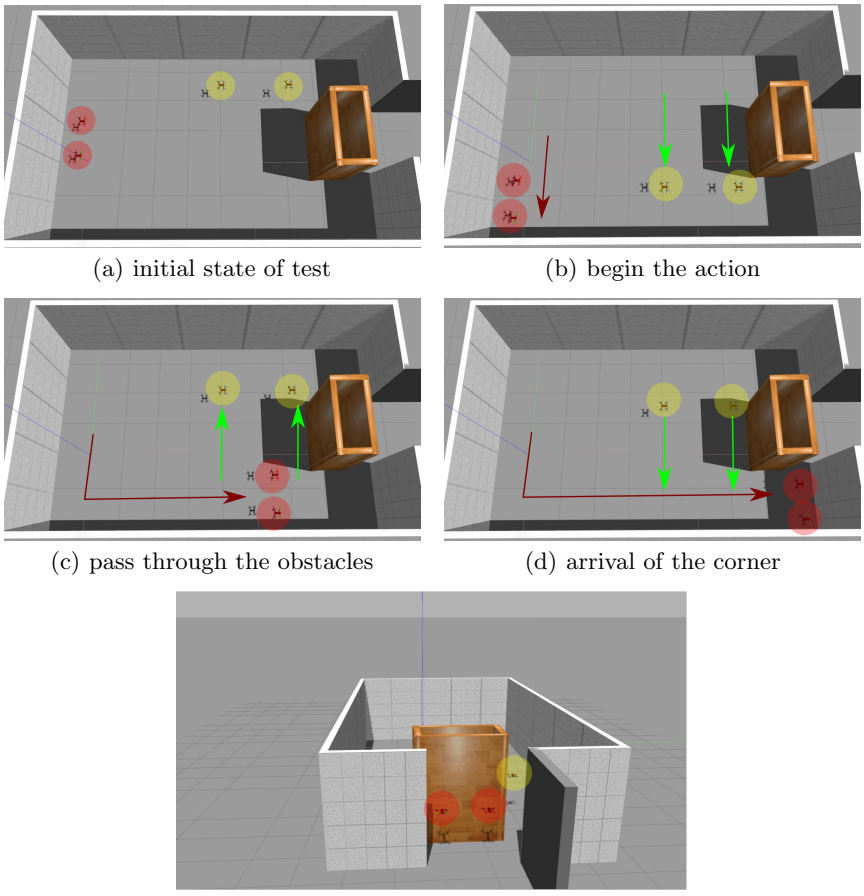


Fig. 5. Communication network structure

In the designed scenario for testing based on ROS-Gazebo, two UAVs are trying to escape from the area, see Fig. 6. The exit is straightforward, but another two UAVs are patrolling in the middle of the terrain and cutting their ways. Based on the trained model in this study, they can take the same strategy, and make a detour to the exit without collisions.

4 Conclusions and Future Works

A method based on Deep-Sarsa for path planning and obstacle avoidance is proposed in this study for UAVs. The proposed approach has been trained in a simplified environment and tested in a ROS-Gazebo simulation platform. The results show the performance of Deep-Sarsa model in the application of path planning



(a) initial state of test (b) begin the action

(c) pass through the obstacles (d) arrival of the corner

(e) arrival the exit

Fig. 6. UAV path planning and dynamic obstacle avoidance in ROS-Gazebo test environment

and obstacle avoidance, especially in a dynamic environment. The trained Deep-Sarsa model can provide a reliable path for UAVs without collisions, although it requires a pre-training process before applying the model. Meanwhile, since not only the model for UAV but also the communication network between different modules have been taken into consideration, the next step is to implement the model and algorithm with real UAV's hardware.

5 Acknowledgements

This work is supported by the project of National Natural Science Foundation of China (No. 61603277), the 13th-Five-Year-Plan on Common Technology, key

project (No. 41412050101), and the Shanghai Aerospace Science and Technology Innovation Fund (SAST 2016017). Meanwhile, this work is also partially supported by the Youth 1000 program project (No. 1000231901), as well as by the Key Basic Research Project of Shanghai Science and Technology Innovation Plan (No. 15JC1403300). All these supports are highly appreciated.

References

1. S. K. Gan and S. Sukkarieh, "Multi-UAV target search using explicit decentralized gradient-based negotiation," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), pp. 751–756, 2011.
2. C. Fu, A. Carrio, and P. Campoy, "Efficient visual odometry and mapping for unmanned aerial vehicle using ARM-based stereo vision pre-processing system," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, (Colorado, USA), pp. 957–962, 2015.
3. I. Maza, K. Kondak, M. Bernard, and A. Ollero, "Multi-UAV cooperation and control for load transportation and deployment," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1, pp. 417–449, 2009.
4. C. Fu, A. Carrio, M. A. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy, "Robust real-time vision-based aircraft tracking from unmanned aerial vehicles," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
5. S. Hayat, E. Yanmaz, T. X. Brown, and C. Bettstetter, "Multi-objective UAV path planning for search and rescue," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Singapore), pp. 5569–5574, 2017.
6. B. M. Sathyaraj, L. C. Jain, A. Finn, and S. Drake, "Multiple UAVs path planning algorithms: A comparative study," *Fuzzy Optimization and Decision Making*, vol. 7, no. 3, pp. 257–267, 2008.
7. S. Hrabar, "3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Nice, France), pp. 807–814, 2008.
8. F. Bounini, D. Gingras, H. Pollart, and D. Gruyer, "Modified artificial potential field method for online path planning applications," in *IEEE Intelligent Vehicles Symposium (IV)*, (Los Angeles, USA), pp. 180–185, 2017.
9. E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.
10. Y. Zhao, Z. Zheng, X. Zhang, and Y. Liu, "Q learning algorithm based UAV path learning and obstacle avoidence approach," in *36th Chinese Control Conference (CCC)*, (Dalian, China), pp. 3397–3402, 2017.
11. N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen, "Autonomous navigation of UAV by using real-time model-based reinforcement learning," in *14th International Conference on Control, Automation, Robotics and Vision*, (Phuket, Thailand), pp. 1–6, 2016.
12. M. Kubat, "Reinforcement learning," *An Introduction to Machine Learning*, pp. 331–339, 2017.
13. D. Zhao, H. Wang, K. Shao, and Y. Zhu, "Deep reinforcement learning with experience replay based on sarsa," *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016.
14. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, pp. 1–6, Kobe, Japan, 2009.

15. N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, (Sendai, Japan), pp. 2149–2154, 2004.
16. J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.
17. S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 287–308, 2000.
18. R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Advances in Neural Information Processing Systems* (D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, eds.), pp. 1038–1044, MIT Press, 1996.
19. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
20. N. Ketkar, "Introduction to keras," *Deep Learning with Python*, p. 97111, 2017.
21. A. S. Huang, E. Olson, and D. C. Moore, "LCM: Lightweight communications and marshalling," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Taipei, Taiwan), pp. 4057–4062, 2010.