ATLANTIS PRESS
Advances in Intelligent Systems Research (AISR), volume 150
2017 International Seminar on Artificial Intelligence, Networking and Information Technology (ANIT 2017)

Trajectory Prediction using Conditional Generative Adversarial Network

Thibault Barbié [a] and Takeshi Nishida [b]

Kyushu Institute of Technology, 1-1 Sensui-cho, Tobata-ku, Kitakyushu-shi, Fukuoka-ken, Japan

[a]q595203a@mail.kyutech.jp, [b]nishida@cntl.kyutech.ac.jp

Keywords: Trajectory prediction, generative model, conditional generative adversarial networks.

Abstract. Optimization based planners (OBP) use a linear initialization as a prior of their optimizations which fails to use already acquired knowledge. Most of the time the linear initialization will collide with obstacles which will be the most difficult part of the OBP to optimize. We propose a method to perform trajectory prediction that leverages motion dataset by using a conditional generative adversarial network. Unlike previous methods, our proposed method does not require the dataset during execution time but instead generate new trajectories. We demonstrate the validity of our method on simulation. Our method decreases by 20% the number of colliding trajectories predicted compared to the linear initialization while being very fast.

1 Introduction

Motion planning is an essential part in a robot system by allowing to move safely between obstacles. Optimization based planners (OBP) are motion planners that focus on improving iteratively an initial trajectory by optimizing a cost function. Most of OBP are not using an educated guess for the initial trajectory, but rather just a simple straight line in configuration space (linear initialization). OBP produce high quality trajectories, but their relatively slow speed to converge to a solution prevent them to be used more widely. Hence, improving their prior guess is an important problem.

Trajectory prediction is a field where robots give a prediction of a possible trajectory to solve a given motion planning problem. Usually, a dataset of motion planning problems and their respective solutions is used to perform the prediction. Multiple work has been done on this problem [1-3]. However, all of these methods need the access to the dataset during the execution time because they require the retrieving of trajectories from the dataset. This implies a limitation on the size of the dataset to be used as the larger the dataset the longer it will take to find the best trajectory to use. In this paper, we propose a method that does not require to retrieve trajectories, but instead generate new trajectories.

Recently, a new generative model named “Generative Adversarial Network” (GAN) has received a lot of interests for its high performances to generate high quality data [4,5]. A GAN consist of two neural networks that try to achieve opposite objectives. One, the generator, tries to generate data that are hard to recognize from the genuine data and one, the discriminator, tries to distinguish fake data from real data. Both networks are trained in parallel in a two-player min-max game and if the GAN converges, the generator is eventually able to generate data that are not discernible from the ones inside the dataset. In our case a data in the motion dataset is the concatenation of a planning problem and a trajectory. After training, a GAN would be able to generate a new planning problem and its solution. However, in the trajectory prediction problem the motion planning problem is already defined so it is needed to condition the generation on it.

Conditional GAN (CGAN) is a simple variation on the GAN algorithm [6]. While being trained, the CGAN learns to generate data with additional information. After training, the CGAN is able to produce data conditioned on the added information. Our proposed method uses a CGAN to learn on the motion dataset how to produce trajectories conditioned on the planning problems. After training, the CGAN is able to generate a trajectory solution for a given planning problem.

We conducted experiments to prove that our algorithm produces more collision-free trajectories compared to the linear initialization.

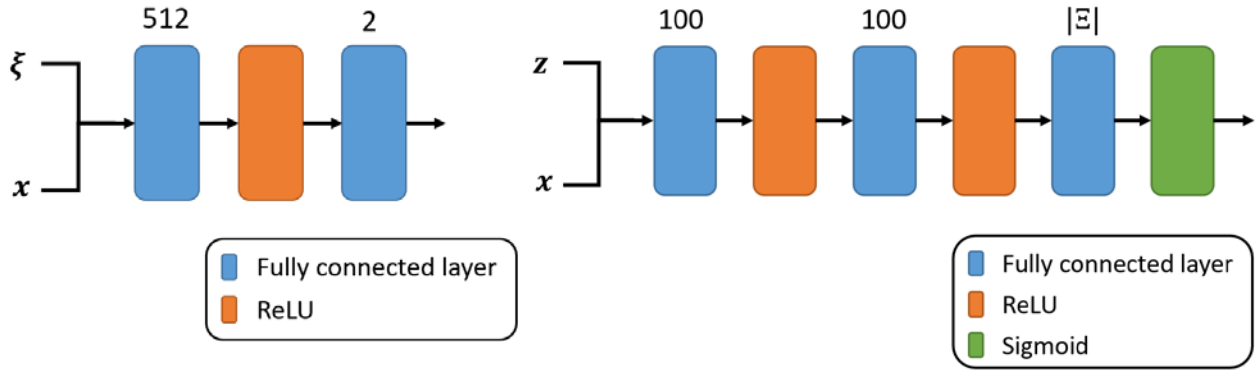

Fig.1. Network architectures of the CGAN. The numbers indicate the number of neurons for each fully connected layer. (Left) The discriminator. (Right) The generator.

2 Problem specification

Let us define X the space of planning problems. A planning problem $\boldsymbol{x}$ in X is a finite vector containing all the necessary information to describe a motion planning problem, e.g. the start and goal points coordinates and information about the positions and orientations of the obstacles. We also define Ξ the space of trajectories. A trajectory $\boldsymbol{\xi}$ in Ξ will be represented by a finite vector such as e.g. a list of waypoints. Consequently, X and Ξ are finite vector spaces. It is important to note that the specific choice of the representation is not important as the algorithm will work the same.

The trajectory prediction problem is to approximate a function $\mathrm{f}: \mathrm{X} \to \Xi$ such that if a planning problem $\boldsymbol{x}$ is given, then $\mathrm{f}(\boldsymbol{x})$ is a trajectory solution of the problem. In practice, we do not aim for such a perfect function, but rather want $\mathrm{f}(\boldsymbol{x})$ to be a "good" prior for optimization based planners to optimize. Here "good" will refer to a collision-free trajectory. Indeed, optimization based planners are mostly optimizing two main criteria: an obstacle avoidance cost function and a smooth cost function. The obstacle avoidance cost function models if a trajectory is colliding with the obstacles or are too close to them. The smooth cost function is measuring how smooth the trajectory is and sometime also includes secondary constraints such as energy cost or length cost. What usually takes more time to optimize is the obstacle avoidance cost function. Therefore, we aim to find an approximation of f that will produce collision-free trajectories.

3 Conditional Generative Adversarial Network

To approximate the aforementioned function f we used a generative model named "conditional generative adversarial network" (CGAN). A generative adversarial network (GAN) could be seen as two players playing a game: a generative model G that plays as a counterfeiter and a discriminative model D which plays as a policeman. The counterfeiter wants to create data that is not discernible from real data while the policeman wants to be able to completely distinguish between the real and fake data. This gives an adversarial setting where both players compete against the other one. In practice, the generative model G takes noise as input with a prior noise distribution $\mathrm{p_z}$. The generator and discriminator are both trained with the following two-player min-max game:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log D(1 - D(G(z)))] \qquad (1)$$

The data we were generating were the concatenation of a motion planning problem and a trajectory $\boldsymbol{d} = (\boldsymbol{x}, \boldsymbol{\xi})$. After being trained, the GAN was able to generate data that could have been inside the dataset. However, when we were generating these new data there was no way to specify a specific motion planning problem; the GAN would create both a new planning problem and its solution which is not what was desired.

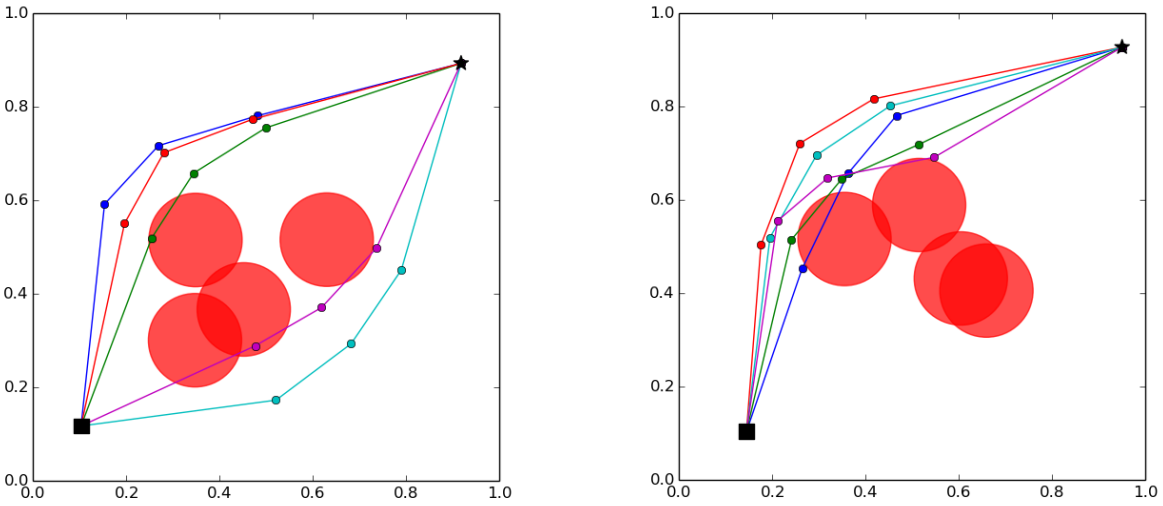

Fig.2. Examples of generated trajectories conditioned on the planning problems.

Consequently, we used a CGAN to be able to generate data conditioned on the motion planning problems. We performed the conditioning by simply feeding the motion planning problems $\boldsymbol{x}$ into both the generator and the discriminator. The architecture of the CGAN we used could be seen Figure 1. GAN and CGAN training could be difficult, but in our case we did not optimize the learning, most of the hyperparameter choices worked directly. We eventually choose to have 200 epochs, a batch size of 150, and the Stochastic Gradient Descent updater but we could have used other configurations. We do not claim to achieve the best performances possibles with these settings. The framework we used to train the networks was Chainer [7].

4 Experiments

To demonstrate the performances of our algorithm we created a motion dataset with the framework "Open Motion Planning Library" (OMPL [8]) by using the RRTConnect motion planner [9]. We created 60,000 problems were the robot should avoid 4 obstacles represented as circles with a fixed radius in a two-dimensional configuration space. During the training, we generated multiple trajectories with the CGAN initialization and the linear initialization to measure if they collided with the obstacles. We used two different metric to measure it. The first one was a binary measure.

$$\mathrm{F}_1(\xi) = \begin{cases} 1 \text{ if colliding} \\ 0 \text{ otherwise} \end{cases}$$

It gave us a collision-free trajectory ratio to quickly be able to compare the methods. However, sometimes two trajectories would collide with the obstacles, but one trajectory will be better than the other one because just a few parts of the trajectory will actually be colliding with the obstacles. This is why we also used a second metric F_2 which computed the percentage of collision in the whole trajectory by sampling the trajectory into 100 waypoints $[\mathrm{q}_1^T\ \mathrm{q}_2^T \cdots \mathrm{q}_{100}^T]$ (with uniform repartition) and computing how many of them were colliding into the obstacles.

$$\mathrm{F}_2(\xi) = \sum_{i=1}^{100} \delta(q_i) \qquad \delta(q) = \begin{cases} 1 \text{ if colliding} \\ 0 \text{ otherwise} \end{cases}$$

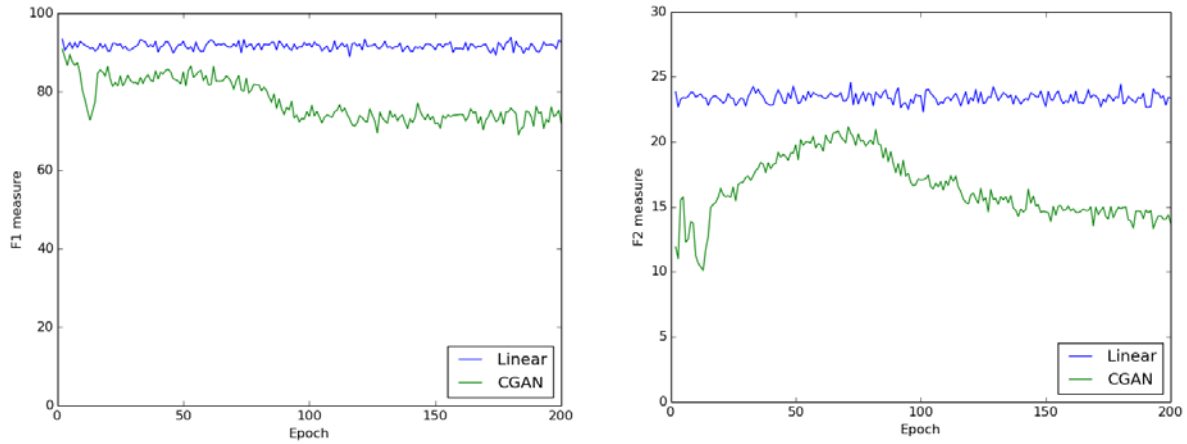

Fig.3. Performances of the CGAN initialization and the linear initialization for the two metrics F_1 and F_2. The CGAN initialization outperforms the linear initialization even after one epoch of training. Compared to the linear initialization, the CGAN initialization decreases the number of colliding trajectories by 20%.

For every epoch of training we randomly generated 1000 motion planning problems and measured the F_1 and F_2 values of both the linear initialization and the CGAN initialization (see Figure 3). The proposed method clearly outperformed the linear initialization even after a few training epochs. Besides, even if a precise measuring had not been done the proposed method was very fast, at least 3000Hz.

5 Conclusion

We developed a new trajectory prediction method based on conditional generative adversarial networks. Predicted trajectories are generated by feeding the generator network with random noise and information about the motion planning problem. Our method produces far more collision-free trajectories than the simple linear initialization. Furthermore, it is simple to implement ahead of optimization based planners which could help them to converge faster to a trajectory solution. Because our method uses a simple feedforward neural network it permits to generate new trajectories without requiring the dataset during execution time. Besides, the small architecture of the networks makes our method very fast and easy to be implemented in small robots. In the future, we aim to test it on real robot systems to prove that it can be used in the real world.

References

[1] Dmitry Berenson, Pieter Abeel, Ken Goldberg. A robot path planning framework that learns from experience. Robotics and Automation (ICRA) 2012 IEEE International Conference on. pp 3671-3678. IEEE. 2012

[2] Nikolay Jetchev, Marc Toussaint. Fast motion planning from experience: trajectory prediction for speeding up movement generation. Autonomous Robots 34. 1-2 (2013): 111-127

[3] Kris Hauser. Large Motion Libraries: Toward a “Google” for Robot Motions. Robotics Challenges and Vision (RCV2013)

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative adversarial nets. Advances in neural information processing systems. pp. 2672-2680. 2014

[5] Alec Radford, Luke Metz, Soumith Chintala, 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

[6] Mirza Mehdi, Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)

[7] Seya Tokui, Kenta Oono, Shohei Hido, Justin Clayton. Chainer: a Next-Generation Open Source Framework for Deep Learning. Proceedings of Workshop on Machine Learning Systems(LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS), (2015)

[8] Ioan A. Sucan, Mark Moll, Lydia E. Kavraki. "The open motion planning library." IEEE Robotics & Automation Magazine 19, no. 4 (2012): 72-82.

[9] James J. Kuffner, Steven M. LaValle. "RRT-connect: An efficient approach to single-query path planning." In Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, vol. 2, pp. 995-1001. IEEE, 2000.