Generative Adversarial Network based heuristics for sampling based path planning

Project description*

Azamat Kanametov Alina Kolesnikova Timofey Zinenko

May 11, 2021

1 Optimal path planning problem

Non-formal Determine a collision–free path from a start point to a goal point while optimizing a performance criterion such as distance, time or energy

Formal

Given:

- $\mathcal{X} \in \mathbb{R}^n$ – the state space, $n \in \mathbb{N}^n$, $n \geqslant 2$
- $\mathcal{X}_{obs}$ – obstacle space, $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$ – free space
- $x_{init} \in \mathcal{X}_{free}$ – the initial state, $x_{goal} \in \mathcal{X}_{free}$ – the goal state
- $\mathcal{X}_{goal} = \left\{ x \in \mathcal{X}_{obs} \middle| \, \|x - x_{goal}\| < r \right\}$ – the goal region
- Σ – the set of all feasible paths
- $c(\sigma)$ – the cost function, $\sigma \in \Sigma$,

$$Cost(x_i, x_j) = \|x_i - x_j\|, \quad x_i, x_j \in \mathcal{X}_{free}$$

Find: feasible path $\sigma^* : [0,1] \to \mathcal{X}_{free}$

$$\sigma^* = \arg\min_{\sigma \in \Sigma} c(\sigma), \quad s.t. \, \sigma(0) = x_{init}, \sigma(1) \in \mathcal{X}_{goal}$$

*Project code is available here `https://github.com/akanametov/pathgan`

2 Approach

2.1 Background

- Sampling–based algorithms solve path planning problems through constructing space–filling trees to search a path σ.
- The tree is built incrementally with samples drawn randomly from the free space $\mathcal{X}_{free}$
- Drawbacks: the quality of initial solution, the convergence speed

2.2 Idea

- Use generative adversarial network (GAN) to learn promising regions and construct heuristic non–uniform sampling distribution $\mathcal{X}_H \subset \mathcal{X}_{free}$ to reduce sampling space
- Use this heuristic in sampling–base algorithm (e.g., RRT*)

2.3 Algorithm

Algorithm 1: GAN–based heuristic RRT*

Input : x_{init}, x_{goal}, Map – state space in the form of RGB image
Output: $G(V, E)$

$V = x_{init}$, $E = \varnothing$;
$\mathcal{S} \leftarrow$ ROIGenerator(x_{init}, x_{goal}, Map) ;
$\mathcal{X}_H \leftarrow$ Discretization($\mathcal{S}$);
for $i = 1 \cdots MAX\_ITER$ **do**
  if $UseHeuristic = True$ **then**
    if $Rand() > \mu$ **then**
      $x_{rand} \leftarrow$ Non-UniformSample($\mathcal{X}_H$);
    else
      $x_{rand} \leftarrow$ UniformSample();
    end
  else
    $x_{rand} \leftarrow$ UniformSample();
  end
end
$x_{nearest} \leftarrow$ Nearest(G, x_{rand});
$x_{new} \leftarrow$ Steer($x_{nearest}$, x_{rand}, min(MAX_EDGE_LEN, $Distance(x_{nearest}, x_{rand})$));
if $ObstacleFree(x_{nearest}, x_{rand})$ **then**
  Extend(G, x_{new});
  Rewire();
  if $x_{new} \in \mathcal{X}_{goal}$ **then**
    Return $G(V, E)$;
  end
end
Return Failure;

ROIGenerator is trained GAN model which outputs image with promising regions (or regions of interest, ROI). Then discretization is performed: coordinates of points from ROI are extracted. Algorithm makes non–uniform sampling from these points with probability $1 - \mu$ and uniform sampling from all map with probability μ.

3 GAN–based promising region generation

Trained with large amount of empirical promising region data, the GAN model is able to generate ROI. The input of the model is an RGB image representing the state space Map, start state x_{init}, and goal state x_{goal}. The output of the model is also an RGB image where the promising regions are highlighted.

3.1 Dataset generation

Algorithm 2: Dataset generation

output: Maps, points, promising regions
Create some initial maps
Randomly augment environment maps, 64×64
    Color $\mathcal{X}_{obs}$ into black
    Color $\mathcal{X}_{free}$ into white
Randomly choose 100 x_{init} and x_{goal} states from $\mathcal{X}_{free}$
    Color x_{init} into red
    Color x_{goal} into blue
Generate promising regions as ground truth
    Randomly select x_{init}, x_{goal} and run RRT 50 times

3.2 GAN training and evaluation

Algorithm 3: GAN training and evaluation

InputGen : $\mathcal{P}$ – point images (with x_{init} and x_{goal}), $\mathcal{M}$ – map images, $\mathcal{Z}$ – noise data
InputDiscr : $\mathcal{S}_{gt}$, $\mathcal{S}_p$ – ground truths and predicted Regions of Interest
OutputGen : $\mathcal{S}_p$ – generated Region of Interest
OutputDiscr: *True* or *Fake*
Training;
while *Generator does not converge* **do**
    $\mathcal{S}_p \leftarrow$ Generator($\mathcal{P}$, $\mathcal{M}$, $\mathcal{Z}$) ;
    $P_{real\,point} \leftarrow$ PointDiscriminator($\mathcal{S}_{gt}$, $\mathcal{P}$) ;
    $P_{fake\,point} \leftarrow$ PointDiscriminator($\mathcal{S}_p$, $\mathcal{P}$) ;
    $P_{real\,map} \leftarrow$ MapDiscriminator($\mathcal{S}_{gt}$, $\mathcal{M}$) ;
    $P_{fake\,map} \leftarrow$ MapDiscriminator($\mathcal{S}_p$, $\mathcal{M}$) ;
    obtain losses $\mathcal{L}_{D_{point}}(P_{real\,point}, P_{fake\,point})$, $\mathcal{L}_{D_{map}}(P_{real\,map}, P_{fake\,map})$, $\mathcal{L}_G(\mathcal{S}_{gt}, \mathcal{S}_p)$;
    backprop
end
Evaluation;
$\mathcal{S}_p \leftarrow$ Generator($\mathcal{P}$, $\mathcal{M}$) ;
Return $\mathcal{S}_p$

4 Experiment results

4.1 GAN results

Two GANs were trained on ROIs obtained from RRT pathfinding algorithm.

The first GAN is from the paper and has in five times more parameters compared to the second one - Pix2Pix (ours). Due to this parameters of training like *learing rate* for Generator and Discriminator(s) are slightly different for them:

1. Original GAN (from paper): Generator *learing rate = 0.0001*, Map Discriminator *learing rate = 0.00005* and Point Discriminator *learing rate = 0.00005*

2. Pix2Pix GAN (ours): Generator *learing rate = 0.001*, Discriminator *learing rate = 0.0007.*

Each of the GANs was trained on *10 epochs.* Some metrics on test set from generated dataset are shown below:

GAN	mIoU	mDICE	mFID	mIS	number of parameters
Original	**70.2%**	**82.0%**	**79.7**	**1.019**	21,231,827
Pix2Pix	58.1%	72.2%	91.2	1.017	**4,170,477**

To check generalization ability of trained GANS, they were tested on unseen MovingAI maps. Results of these ROIs are presented below:

GAN	mIoU	mDICE	mFID	mIS	number of parameters
Original	**38.4%**	**53.8%**	**88.1**	**1.014**	21,231,827
Pix2Pix	30.8%	46.3%	100.1	1.012	**4,170,477**

4.2 ROI quality evaluation

As proposed in paper we evaluated *connectivity* and *generalization ability.*

Connectivity We set the generated ROI for test images as free space and other areas as obstacle space. Then we execute RRT algorithm: if feasible paths inside free space can be found, it means the ROI is connected

Generalization ability We resized some maps from Moving AI to 64×64, fed it to trained GANs to generate ROI and evaluated connectivity as above

We measured success rate, % (found connected regions by total number of test maps).

GAN	Generated	MovingAI
Original	**65.8%**	54.5%
Pix2Pix	65.4%	**67.4%**

4.3 Optimality evaluation

To illustrate the improvement on RRT*:

1. We randomly choose task for each type of map in the test set

2. We ran RRT* with uniform sampling and RRT* with heuristic for 50 times

For first and best found paths the following metrics were collected:

1. Path cost (as sum of euclidean distances between adjacent nodes), time spent in iterations and seconds
2. Nodes added in graph
3. Number of unique nodes sampled (not necessarily added to the graph if obstacles were found inside the new edge)

Also we computed average statistics:

1. Of the all found paths lengths (#nodes) and their costs
2. Of the nodes added in graph and nodes sampled after each 10 iterations

All collected statistics are available by link `https://akanametov.github.io/pathgan/results/`. From these plots we made the following conclusions:

- RRT* with ROI found first path faster then RRT* with uniform sampling for both trained models, and difference in time becomes more significant for best path (on average it takes half as many iterations for RRT* with ROI)
- RRT* with ROI requires less sampling for finding optimal path
- Paths found by RRT* with ROI have lower costs and variance
- Although connectivity and CV metrics for unseen MovingAI maps are worse than for our initial dataset, RRT* with heuristic performs better in most cases

From the line plots we can also observe that RRT* with heuristic samples less nodes and converges faster to the optimal path. But it can be seen that there is no significant difference between number of nodes in paths found by RRT* and RRT* with heuristic.