sensors

MDPI

Article

Action Generative Networks Planning for Deformable Object with Raw Observations

Ziqi Sheng †, Kebing Jin †, Zhihao Ma and Hankz-Hankui Zhuo *

School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China; shengzq@mail2.sysu.edu.cn (Z.S.); jinkb@mail2.sysu.edu.cn (K.J.); mazhh7@mail2.sysu.edu.cn (Z.M.)

* Correspondence: zhuohank@mail.sysu.edu.cn

† These authors contributed equally to this work and should be regarded as co-first authors.

Abstract: Synthesizing plans for a deformable object to transit from initial observations to goal observations, both of which are represented by high-dimensional data (namely "raw" data), is challenging due to the difficulty of learning abstract state representations of raw data and transition models of continuous states and continuous actions. Even though there have been some approaches making remarkable progress regarding the planning problem, they often neglect actions between observations and are unable to generate action sequences from initial observations to goal observations. In this paper, we propose a novel algorithm framework, namely `AGN`. We first learn a *state-abstractor model* to abstract states from raw observations, a *state-generator model* to generate raw observations from states, a *heuristic model* to predict actions to be executed in current states, and a *transition model* to transform current states to next states after executing specific actions. After that, we directly generate plans for a deformable object by performing the four models. We evaluate our approach in continuous domains and show that our approach is effective with comparison to state-of-the-art algorithms.

Keywords: AI planning; contrastive learning; action model

Citation: Sheng, Z.; Jin, K.; Ma, Z.; Zhuo, H.-H. Action Generative Networks Planning for Deformable Object with Raw Observations. *Sensors* **2021**, *21*, 4552. https://doi.org/10.3390/s21134552

Academic Editors: Abdeldjalil Ouahabi, Amir Benzaoui and Sébastien Jacques

Received: 14 May 2021
Accepted: 26 June 2021
Published: 2 July 2021

1. Introduction

For future robots to perform general tasks in unstructured environments such as homes or hospitals, they must be able to reason about their domains and plan their actions accordingly. In AI literature, this general problem has been investigated under two main paradigms—automated planning and scheduling [1] (AI planning) and reinforcement learning [2]. At the same time, many objects in human daily life are deformable or nonrigid, such as clothes and ropes. Hence, dealing with deformable objects planning is a significant issue. In this issue, there have been many studies that seek to handle deformable object planning problems [3–5].

Researchers face two main challenges when handling deformable object planning tasks. On the one hand, unlike strict objects planning tasks, it is often difficult to specify logical representation of a state correctly in deformable object related domains. For example, considering designing a logical representation of the state of a deformable object such as a cloth, it is difficult to "logically" specify features of the deformable objects, e.g., bending angles, relative positions of different parts of deformable objects, etc. On the other hand, the action models of deformable things are sophisticated and nonlinear [6], which makes modeling and completing planning task in such deformable object domains challenging.

One category of studies managing the challenges in continuous states and actions domains is model-free learning [7,8]. They either relied on domains whose rewards are instrumented [9–11], or required high-quality demonstrations to guide the learning process [12]. Without high-quality demonstrations, however, model-free learning is notoriously weak, and often needs huge numbers of instances to learn from.

Another category of studies, i.e., model-based learning, has also shown promising in sample-efficient learning [13,14]. Using such model-based learning studies for deformable objects, however, researchers should consider how to represent state and learn action models appropriately. Some approaches take a direct approach to learning complex action models through raw space [4,15]. However, compared with latent space, raw space has too much redundant information, which is not conductive to model learning. The other approaches, such as Agrawal et al. [16] and Nair et al. [17], aim to learn forward dynamic models for manipulating deformable objects. Other model-based studies such as thanard et al. [18] train Causal InfoGANS [19] to both obtain visual representations and action models for planning. However, those techniques are weak due to training instabilities concerned with GANS [20] and cannot generate actions to guide robot to perform tasks.

In this paper, we propose a novel model-based algorithm framework, called `AGN`, which stands for action generative network, to compute action sequences for guiding an agent to perform a task from initial observations to target observations, and predict updating observations after executing actions. `AGN` uses contrastive technology to learn both the underlying heuristic models and transition models for deformable objects at the same time. We assume that using contrastive technology for model-based learning obtains better generalization and latent space structure with its inherent information maximization loss function. We modified the loss function posed in contrastive predictive coding [21] to learning effective transition model and heuristic model jointly. When the latent models for representations, the transition model, and the heuristic model are learned offline, we can use these models to manipulate deformable objects from a certain initial observation to the desired goal observation.

2. Related Work

2.1. *Deformable Object Planning*

There has been a lot of work in the area of robotic manipulation of deformable objects [22]. The deformable object handling problem has been studied via classical methods such as motion planning and manipulation [23]. There has been recent interest in combining deep generative models with structured dynamical systems in the context of variational autoencoders, where the latent space is continuous [24]. Watter et al. [25] used such models to perform the planning via learning latent linear dynamics and exploiting a linear quadratic Gaussian control algorithm. Causal InfoGAN [18] used Gumbel-Softmax to backprop through transitions of discrete binary states, and leveraged the structure of the binary states for planning. Ha et al. [26] presented a representation learning algorithm that learned a low-dimensional latent dynamical system from high-dimensional sequential raw data, e.g., videos.

In the planning literature, most studies relied on manually designed state representations. In a recent work, Konidaris et al. [27] automatically abstracted state representations from raw observations, but relied on a prespecified set of skills for the task. Sriniva et al. [28] introduced universal planning networks that embedded differentiated planning within a goal-directed policy. This planning computation unrolls a forward model in a latent space and infers an optimal action plan through gradient descent trajectory optimization. The plan-by-gradient-descent process and its underlying representations are learned end-to-end to directly optimize a supervised imitation learning objective. Our approach performs a goal-directed deformable planning by using the linear interpolation method and can achieve convergence quicker than other methods.

2.2. *Contrastive Prediction*

It remains a challenge to learn a valid representation in the deformable object domain. Many researchers seek to use contrastive learning methods to handle this problem. For example, Word2Vec [29] optimizes a contrastive loss to demonstrate semantic and syntactic structure in the learned latent space for words. Oord et al. [21] introduce negative examples to learn abstract representations of high-dimensional data, for example, pictures. Tian et al. [30] learn abstract representations by letting different views of images be embedded closely to another, and further from the others through a contrastive loss.

Lately, SimCLR [31] achieves good results in self-supervised learning representations, by introducing a nonlinear transformation between the representation and the contrastive loss.

Different from the above-mentioned work, we aim to consider generating action sequences by introducing action transition relations in `AGN`. Instead of directly planning on the high dimension observation, we choose to plan in low latent space. `AGN` perform a goal-directed planning process by using the heuristic model and transition model iteratively, which is useful and can converge quicker than other methods.

3. Our `AGN` Approach

In this section, we introduce our framework, `AGN`, which stands for **a**ction **g**enerative **n**etwork for deformable object planning (`AGN`). We begin with presenting the problem formulation. After that, we address our `AGN` algorithm framework in detail. Finally, we describe the procedure of solving a goal-directed planning problem with `AGN`.

3.1. Problem Definition

Our training data are a set of trajectories $\mathcal{T} = \langle l_1, l_2, \ldots l_n \rangle$, each trajectory $l_i \in \mathcal{T}$, is defined by $l_i = \langle o_0, a_1, o_1, a_2, \ldots, a_{N-1}, o_N \rangle$, where o_i is a raw observation, e.g., image, and a_i is an action denoted by a N-dimension tensor. Note that each dimension of an action has specific meaning. A raw observation o_i is changed into a new raw observation o_{i+1} after executing action a_i. For example, in the rope domain shown in Figure 1, an action is represented by a vector of five elements: (p_x, p_y, ϕ, c, g), where p_x, p_y are x-coordinate and y-coordinate of the point of the rope where the action is executed. ϕ is the angle of the rope being moved by the action. c is the length of the rope moved by the action, g is a boolean value indicating whether the action should be used for training. As an example, action "$(2.0, 3.0, \pi, 0.05, 1)$" indicates the point $(2.0, 3.0)$ of the rope is moved with π degrees and 0.05 meters length and the action will be used for training due to $g = 1$.

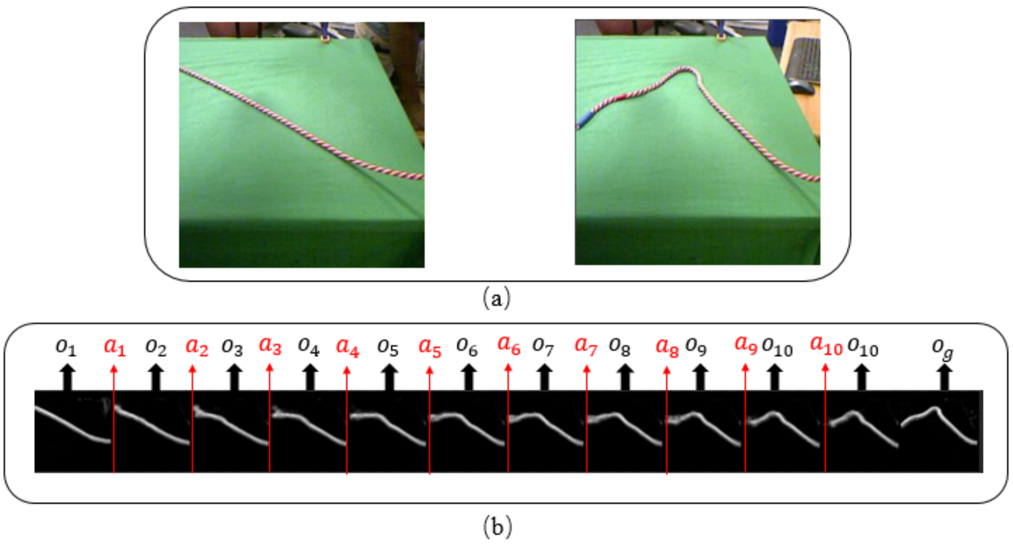

Figure 1. (**a**)Examples of initial observations and goal observations in rope domain. (**b**) Examples of 10 step trajectory, given the initial and goal observations in (**a**)

We define our learning problem as: given a set of training data $\mathcal{T}$, we aim to learn a planning model $\mathbb{M}$, i.e., action generative networks, using $\mathcal{T}$. With the learned $\mathbb{M}$, we formulate our planning problem as a tuple $\mathcal{P} = \langle \mathbb{M}, o_0, o_g, \mathcal{A} \rangle$, where o_0 is an initial observation, o_g is a goal observation, $\mathcal{A}$ is a set of actions. We aim to solve the planning problem $\mathcal{P}$ by generating a trajectory (i.e., a plan) $\sigma = \langle o_0, a_0, \ldots, a_{n-1}.o_g \rangle$ that transforms the initial observation o_0 to goal observation o_g.

An example of initial observation and goal observation is shown in Figure 1a, where the figure on the left side shows an initial observation and a goal observation is shown in the figure on the right side. Figure 1b is a plan of 10 actions transforming o_1 to o_g. Each action in the plan updates the positions of different points of the rope in different directions until reaching the goal.

3.2. Algorithm Framework

In this section, we introduce our proposed framework for learning deformable object manipulation from fully observable raw observations: action generative network (AGN). We begin with the details of our approach. After that, we discuss our planning process with AGN in the next section.

An overview of our AGN approach is shown in Figure 2. The training process of our algorithm contains two steps: we first jointly train an auto-encoder, including a state-abstractor model $E(o;\theta_1) = s$, which extracts the low-dimensional latent abstract state s given a raw observation o, and a state-generator model $D(s,z;\theta_2) = o$, which generates a raw observation given a low-dimensional latent abstract state s and noise z. After that, we train a heuristic model $F(s_t, s_g; \theta_3) = a_t$, which generates action a_t to be executed on state s_t given current state s_t and goal state s_g, and a transition model $T(s_t, a_t; \theta_4) = s_{t+1}$, which generates a new state s_{t+1} given a current state s_t and an action a_t. θ_1, θ_2, θ_3 and θ_4 are parameters of the four models, respectively, which are to be learned with the training data.

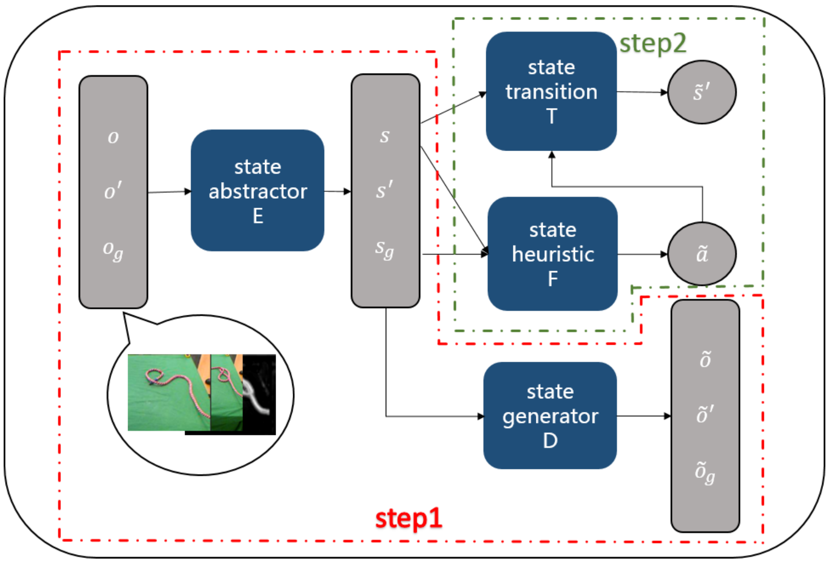

Figure 2. The framework of our action generative network (AGN).

Planning in high-dimensional continuous domains is hard in general. Therefore, we consider planning based on low-dimension latent space. In order to learn the conversion between the high-dimension raw observation and low-dimension latent state, we first learn an auto-encoder, which contains a state-abstractor model $E(o;\theta_1) = s$ and a state-generator model $D(s,z;\theta_2) = \tilde{o}$. The reason of adding noise into state-generator model is to improve the robustness. We jointly learn a state-abstractor model $E(o;\theta_1) = s$ and a state-generator model $D(s,z;\theta_2) = \tilde{o}$ by minimizing the MSE loss comparing the real raw observation o with the reconstructed raw observation $\tilde{o}$, which is defined by Equation (1).

$$L = \|o - D(E(o;\theta_1))\|_2 \tag{1}$$

After learning the projection between high-dimensional raw observations and low-dimensional latent space, then we jointly learn the heuristic model F and the transition model T. The whole process of jointly training heuristic model F and transition model T is shown in Figure 3. Heuristic model F predicts an action $\tilde{a}_t$ given a current state s_t and a goal state s_g by Equation (2). Then transition model T updates current state s_t to a next state s_{t+1} after executing $\tilde{a}_t$ by Equation (3).

$$\tilde{a}_t = F(s_t, s_g; \theta_3) \tag{2}$$

$$\tilde{s}_{t+1} = T(s_t, \tilde{a}_t; \theta_4) \tag{3}$$

where $\tilde{a}_t$ is the predicted action , a_t is the real action. s_t is the current state, $\tilde{s}_{t+1}$ is a predicted next state, s_{t+1} is the real next state, s_g is a goal state. s_t, s_{t+1}, s_g is computed by state-abstractor model given a current observation o_t, the next observation o_{t+1}, a goal observation o_g. Then we train the heuristic model with a loss function defined by Equation (4).

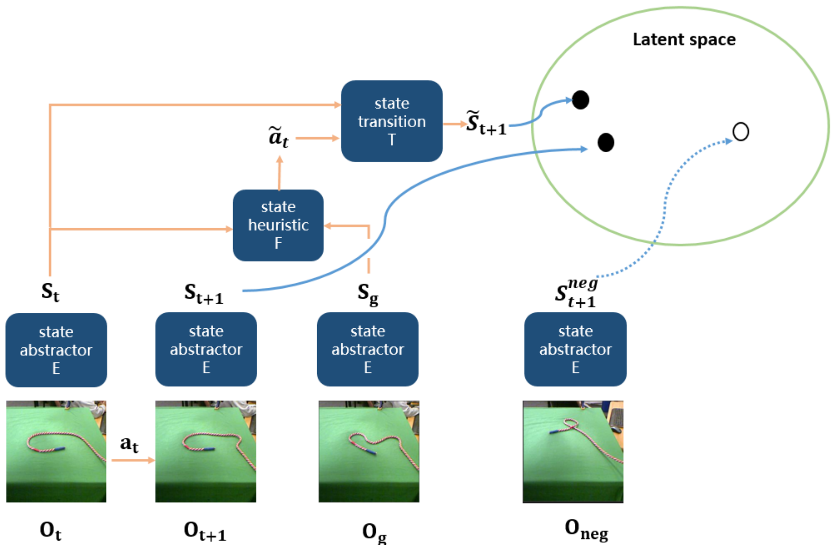

Figure 3. The framework of the joint-training heuristic model and transition model.

$$L_F = \|a_t - \tilde{a}_t\|_2 \tag{4}$$

Next we define an InfoNCE contrastive loss described by Oord et al. [21], which is defined by Equation (6), where $\dot{s}_{t+1} = \langle \dot{s}_{t+1}^0, \ldots, \dot{s}_{t+1}^{k-1} \rangle$ is a set of incorrect latent states different from the real next state s_{t+1}. An incorrect latent state $\dot{s}_{t+1}^i$ is generated by a sample in a set of negative samples $\dot{o}_{t+1} = \langle \dot{o}_{t+1}^0, \ldots, \dot{o}_{t+1}^{k-1} \rangle$. We construct negative samples $\dot{o}_{t+1}$ by random selecting k samples, the latent state of each sample is different from the real next state s_{t+1}. The h function shown in Equation (6) is some similarity function between the computed latent states, which is computed by Equation (5). The motivation behind this objective function is to let the predicted states and their corresponding positive samples be close in latent space.

$$h(z_1, z_2) = exp(-\|z_1 - z_2\|^2) \tag{5}$$

$$L_c = -\mathbb{E}[log \frac{h(\tilde{s}_{t+1}, s_{t+1})}{\sum_{i=0}^{k-1} h(\tilde{s}_{t+1}, \dot{s}_{t+1}^i)}] \tag{6}$$

Then we define an L2 norm of convariance matrix to full the loss L by Equation (7) following tharand et al. [18], aiming at learning a latent planning system such that linear interpolation between states makes for feasible plans. To bring about such latent space, we consider transition probabilities $T_M(s_{t+1}|s, a; \theta_4)$ given as Gaussian perturbations of the state: $s_{t+1} = s + \delta$, where $\delta \sim \mathcal{N}(0, \Sigma_{\theta_4}(s))$, and $\Sigma_{\theta_4}(s)$ is a diagonal convariance matrix. The key idea here is that, if only small local transitions are possible in the system, then a linear interpolation between two states s_0 and s_g has a high probability, and it represents that a feasible trajectory exists in the observation space.

$$L_n = \mathbb{E}_{s \sim P_M} ||\Sigma_{\theta_4}(s)||_2 \tag{7}$$

where the prior probability P_M for each element of s is uniform in [−1,1].

Therefore, the loss function of transition model can be defined by Equation (8). Finally, we jointly learn the heuristic model F and transition model T by minimizing the loss function defined by Equation (9), where λ is a hyper-parameter.

$$L_T = L_c + L_n \tag{8}$$

$$L = \lambda L_F + (1 - \lambda) L_T \tag{9}$$

3.3. *Planning with* `AGN`

After training the state-abstractor model E, state-generator model D, heuristic model F, and transition model T, naturally, we use them for planning to solve deformable object planning problems, aiming at computing an action observation trajectory to reach o_g from o_0. The overall planning process can be divided into three steps.

- Firstly, state-abstractor model E outputs abstract state s_0 and s_g with o_0 and o_g, respectively.
- Secondly, we compute an action sequence reaching s_g from s_0 and derive an action state trajectory $\gamma = \langle s_0, a_0, s_1, a_1, \dots, a_{N-1}, s_N \rangle$ by Algorithm 1. We first perform linear interpolation between s_0 and s_g, and attain an initial sequence η = $[s_0, s_1, \dots, s_n, s_g]$. As for each pair of s_i and s_{i+1}, we compute an action a_i by the heuristic model. If s_i can reach s_{i+1} after executing action a_i, we add state s_i and action a_i into θ. Otherwise, we interpolate a latent state s_{mid} into η between s_i and s_{i+1}. We repeat the above procedures until each pair of states in η can be transformed by an action computed by the heuristic model. Finally, we attain an action state trajectory $\gamma = \langle s_0, a_0, s_1, a_1, \dots, a_{N-1}, s_N \rangle$.

Algorithm 1 planning algorithm.

```
input: $s_0$, $s_g$, $F$, $T$.
output: $\gamma = \langle s_0, a_0, s_1, a_1, \dots, a_{N-1}, s_N \rangle$
1:  do linear interpolation between $s_0$ and $s_g$, get $\eta$ = $[s_0, s_1, \dots, s_n, s_g]$
2:  i=0, $\gamma = []$
3:  while i < n do
4:      $a_i = F(\eta[i], \eta[i+1])$, $\tilde{s}_{i+1} = T(\eta[i], a_i)$
5:      if $||\tilde{s}_{i+1} - \eta[i+1]||_2 < 1e-3$ then
6:          $\gamma = [\gamma|\gamma[i+1]]$, $\gamma = [\theta|a_i]$
7:          $i+=1$
8:      else
9:          $s_{mid} = (\eta[i] + \eta[i+1])/2$
10:         $\eta = \eta[0:i+1] + [s_{mid}] + \eta[i+1:]$
11:         $n+=1$
12:     end if
13: end while
```

- Finally, we compute an action observation trajectory $\sigma = \langle o_0, a_0, o_1, a_1, \dots, a_{N-1}, o_N \rangle$. We first sample k different Gaussian noises randomly. Then we can obtain k different action observation trajectories given an action state trajectory θ and a noise by state-generator model D. At last, we select an optimal action observation trajectory $\sigma = \langle o_0, a_0, o_1, a_1, \dots, a_{N-1}, o_N \rangle$ among the k trajectories.

Since the states and the actions for deformable object are in continuous space, the optimality and determinism of the solutions can hardly be discussed in this paper. In summary, given an initial observation and a goal observation, we can finally obtain a feasible trajectory that is valid and clear compared to other state-of-the-art methods.

4. Experiments

In our experiments, we aimed to (1) visualize the abstract states and planning in `AGN`; (2) show that `AGN` can produce realistic visual plans in a complex dynamical system; (3) show that `AGN` significantly outperforms baseline methods.

We began our investigation with a set of experiments in the rope domain, specifically designed to demonstrate the benefits of `AGN`, where we also compared `AGN` with other methods. We later present experiments on a real dataset of robotic cloth manipulation and verified the influence of two important hyper-parameters. Since both cloth and rope datasets are collected in the real physical environment, the final plan we learned is definitely fitted to a real setting.

4.1. Baselines

In order to evaluate `AGN`, we compared our approach with state-of-the-art algorithms. The first one is the visual forward model [32]; we achieve it by realizing training and planning process purely in pixel space. Secondly, we jointly learn a forward and inverse model following Lee et al. [16]. Finally, we compared `AGN` to causal InfoGAN [18], synthesizing plans to transit from initial observations to goal observations based on the InfoGAN [18] framework.

In consideration of the failure of the visual forward model and the causal InfoGAN to generate action sequences, we have trained an inverse model on the dataset, given a current observation o_t and a next observation o_{t+1}, the action between o, o_{t+1} can be generated.

4.2. Evaluation Criterion

We evaluate our approach based on three aspects:

- Trajectory confidence, to evaluate whether an observation transition is feasible or not.
- Trajectory distance, to evaluate the Euclidean distance between the current observation and the next observation after the current action is performed.
- Final-to-goal distance, to evaluate the Euclidean distance between the final observation and goal observation.

In order to quantitatively analyze the action trajectories we generated, we take the pretraining model proposed by Therand et al. [18], which is called Judge, to evaluate whether an observation transition is feasible or not. Trajectory confidence value is in [0, 1], a higher score represents a higher confidence coefficient. Given an output trajectory $\sigma = \langle o_0, a_0, o_1, a_1, \ldots, a_{N-1}, o_N \rangle$, we can compute the trajectory confidence used by the Judge in Equation (10).

$$tc = \frac{1}{N} \sum_{i=1}^{N-1} Judge(o_i, o_{i+1}) \tag{10}$$

where tc is trajectory confidence, N is the length of trajectory.

Moreover, we train a path distance function EVAL to evaluate the Euclidean distance between the current observation and the next observation after the current action is performed. Trajectory distance is computed by Equation (11).

$$td = \frac{1}{N} \sum_{i=1}^{N-1} EVAL(o_i, a_i, o_{i+1}) \tag{11}$$

where td is trajectory distance and the less td is, the better a trace is, N is the length of trajectory.

We also compare the final-to-goal distance. Final-to-goal distance is the Euclidean distance between the final observation and goal observation, indicating that the smaller the final-to-goal distance is, the better action trajectory is.

Then we introduce the training process of the Judge model and EVAL model in detail:

1. The Judge model takes a pair of observations (o_t, o_{t+1}) as input and outputs a binary result of whether the observation is feasible or not. The training dataset consists of positive observation pairs, which are 1 timestep apart, and negative pairs that are randomly sampled from different rope manipulation trajectories. To avoid the background of rope influencing the training of Judge, we preprocess the rope data using the background subtraction pipeline mentioned above.

   To validate the accuracy of the Judge model, we evaluate it with observation traces to observe the binary outputs. Given an m-length observation trace, Judge takes the first observation and an observation, which is n steps apart, where n is from 1 to $m-1$. The binary output decreases from 1 to 0 smoothly with n increasing, indicating that the Judge model has the ability to recognize a feasible observation pair. We test Judge with 100 traces out of the testing dataset for AGN and the accuracy is 98%.
2. The EVAL model takes a pair of observations $(o_t, \hat{o}_{t+1})$, an action a_t, and an observation o_{t+1} as inputs, where $\hat{o}_{t+1}$ is a predicting next observation and o_{i+1} is a real next observation, they are updated from a current observation o_t after executing action a_t. The EVAL model outputs a distance between o_{t+1} and $\hat{o}_{t+1}$. The training dataset consists of positive next observations, we trained the EVAL model by letting the predict next observation $\hat{o}_{t+1}$ be close to the real next observation o_{t+1}. On a held-out test set, the distance between the predict next observation and real next observation converges to 0.

Note that the Judge and EVAL models are trained independent of `AGN`. Thus, trajectory confidence and trajectory distance are both impartial metrics.

4.3. Rope Manipulation

The rope dataset [17] contains sets of sequential pictures and corresponding actions, collected by a robot operating a rope in a self-supervised manner. The sample size used in the training process is 100,000. Each initial picture is $64 \times 64 \times 3$ RGB. In order to remove interference factors, we converted the images to grayscale images, and used a model *BRM* proposed by Therand et al. [18] to remove the background, aiming at focusing on the object itself, which can avoid the algorithm overfitting to the background.

Regarding the definition of states, we follow the configuration of continuous abstract states specified in [18]. In this section, we intend to verify the effectiveness of the algorithm to handle deformable objects with continuous actions and continuous states.

Table 1 shows the results of `AGN` and baselines in the rope domain. We trained on 800 pairs of test samples to obtain this average results. As shown in Table 1, our algorithm framework outperforms the other baseline in all kinds of evaluation methods, which verifies the reliability of our method. In term of trajectory confidence, it means that we can generate paths that are much more confident and much smoother than other algorithms. As for the trajectory distance, `AGN` is significantly lower than the others. Because the visual forward method and causal InfoGAN neglect actions, they cannot reason about the transition and updating between observations after executing actions. As for final-to-goal distance, `AGN` can generate action trajectories that are closer to the goal observation more effectively, which indicates that `AGN` outperforms the other three algorithms in goal-arrived tasks. Visual forward and joint dynamics are poor at long distance planning; therefore they are often unable to reach the goal. Figure 4 shows six examples generated by `AGN` and each row is a trajectory between initial raw observations and goal raw observations. As shown in Figure 4, given different pairs of initial and goal observations, `AGN` is able to generate a well-shifted and clear observation path.

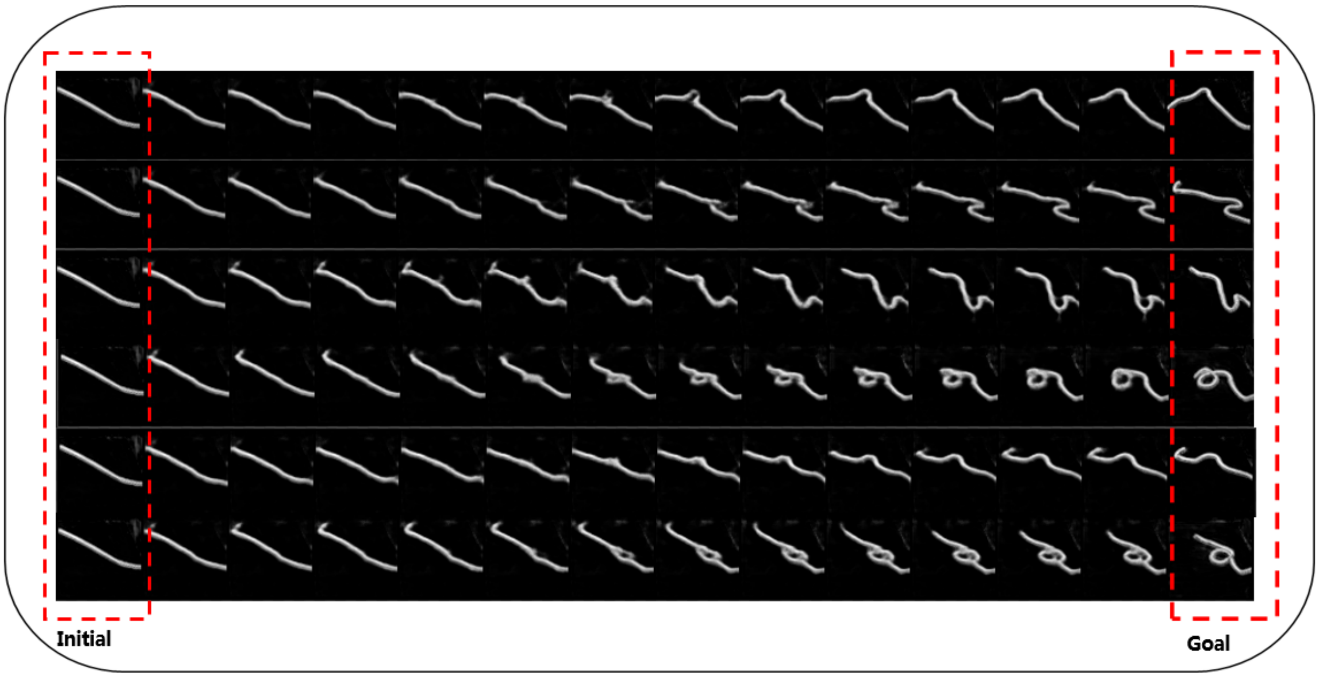

Figure 4. Result for rope manipulation data. The plot shows six planning instances, from left (initial observation) to the right (goal observation).

Table 1. Evaluation of planning result in rope domain.

	Trajectory Confidence	Trajectory Distance	Final–to–Goal Distance
visualforward	0.719	9.7189	5.5484
jointdynamics	0.567	10.680	5.046
ausalinfoGAN	0.884	9.0219	2.29
AGN	**0.935**	**1.432**	**2.126**

4.4. Cloth Manipulation

In this section, we present the results of our experiments in the cloth domain to verify the effectiveness of our algorithm framework. The sample size used in the training process is 400k. Because training on the cloth domain is more difficult than training on the rope domain, we used a larger sample size. As shown in Figure 5, given a pair of an initial observation and a goal observation, we can finally obtain a valid trajectory. Since actions are abstract tensors, they do not have graphical representations. Then we compare origin AGN with AGN training in raw observation space. The last two rows of Figure 5 show that training AGN in raw observation space cannot learn correct action models, leading to bad trajectories.

We jointly trained the heuristic model and transition model; the ratio between heuristic model loss and transition model loss is λ, which is a hyper-parameter, shown in Equation (9). Figure 6 shows the relation between λ and trajectory distance. When $\lambda = 0.2$, the trajectory distance is the smallest, because the heuristic model will inevitably have gradient flow when training transition model. Further, we also compare the performance with different latent state dimension. As shown in Figure 7, where we set $\lambda = 0.2$, when the latent state dimension is 16, we can obtain the smallest trajectory distance. After that, the trajectory distance slowly grows as the latent state dimension increases, because it is hard to express all of the information in an observation with a low dimensional state vector; further, it becomes more difficult to train a neural network with more weights when the dimension size increases.

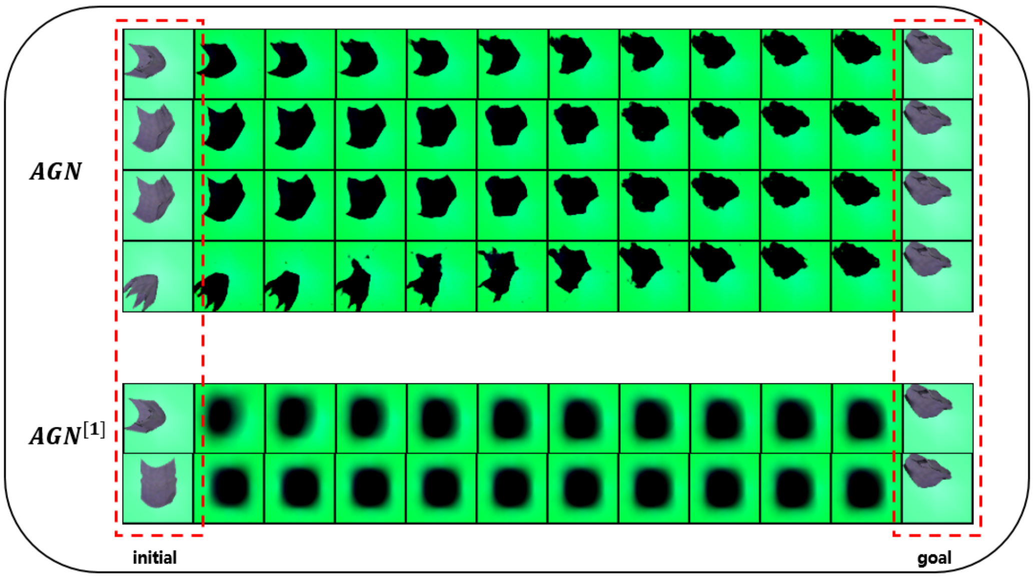

Figure 5. The first four rows are the origin algorithm AGN; the last two rows are $AGN^{[1]}$ training in raw observation space. Given initial observation and goal observation, AGN can attain valid trajectories.

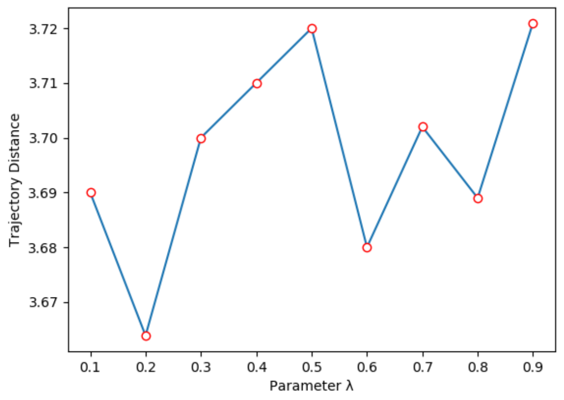

Figure 6. Trajectory distance with different parameter λ.

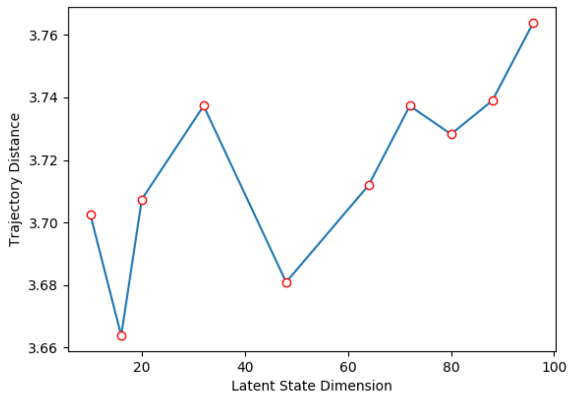

Figure 7. Trajectory distance with different latent state dimension.

5. Conclusions

In this paper we propose a novel planning model learning framework, `AGN`, by considering actions between observations. Based on `AGN`, we learn four models, i.e., the state-abstractor model, state-generator, heuristic model, and transition model, and solve new planning problems with the learned models. Our experimental results show that our `AGN` approach is effective in comparison to baselines. In the future, we would like to extend our work to complex domains and consider objects in our framework that can better leverage the benefit of both deep learning and classical AI planning. It is also interesting to investigate the possibility of applying our `AGN` approach to learning action models [33–36] and recognizing plans [37–39] in the planning community.

Author Contributions: Conceptualization, Z.S., K.J., and H.-H.Z.; methodology, Z.S., K.J., and H.-H.Z.; software, Z.S.; validation, Z.S.; investigation, Z.S.; resources, Z.S.; data curation, Z.S. and Z.M.; writing—original draft preparation, Z.S., K.J., and H.-H.Z.; writing—review and editing, Z.S., K.J., and H.-H.Z.; visualization, Z.S.; supervision, H.-H.Z.; project administration, H.-H.Z.; funding acquisition, H.-H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No. 62076263), Guangdong Natural Science Funds for Distinguished Young Scholar (Grant No. 2017A030306028), Guangdong special branch plans young talent with scientific and technological innovation (Grant No. 2017TQ04X866), Pearl River Science and Technology New Star of Guangzhou and Guangdong Province Key Laboratory of Big Data Analysis and Processing.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ghallab, M.; Nau, D.S.; Traverso, P. *Automated Planning and Acting*; Cambridge University Press: Cambridge, UK, 2016.
2. Sutton, R.S.; Barto, A.G. *Reinforcement Learning—An Introduction*; Adaptive Computation and Machine Learning; MIT Press: Cambridge, MA, USA, 1998.
3. Schulman, J.; Lee, A.X.; Ho, J.; Abbeel, P. Tracking deformable objects with point clouds. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 1130–1137.
4. Wu, Y.; Yan, W.; Kurutach, T.; Pinto, L.; Abbeel, P. Learning to Manipulate Deformable Objects without Demonstrations. *arXiv* **2019**, arXiv:1910.13439
5. Seita, D.; Ganapathi, A.; Hoque, R.; Hwang, M.; Cen, E.; Tanwani, A.K.; Balakrishna, A.; Thananjeyan, B.; Ichnowski, J.; Jamali, N.; et al. Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 9651–9658.
6. Essahbi, N.; Bouzgarrou, B.C.; Gogu, G. Soft Material Modeling for Robotic Manipulation. *Appl. Mech. Mater.* **2012**, *162*, 184–193. [CrossRef]
7. Mirza, M.; Jaegle, A.; Hunt, J.J.; Guez, A.; Tunyasuvunakool, S.; Muldal, A.; Weber, T.; Karkus, P.; Racanière, S.; Buesing, L.; et al. Physically Embedded Planning Problems: New Challenges for Reinforcement Learning. *arXiv* **2020**, arXiv:2009.05524
8. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.I.; Moritz, P. Trust Region Policy Optimization. In Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015; Bach, F.R., Blei, D.M., Eds.; Volume 37, pp. 1889–1897.
9. Hafner, D.; Lillicrap, T.P.; Ba, J.; Norouzi, M. Dream to Control: Learning Behaviors by Latent Imagination. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
10. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.A.; Fidjeland, A.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
11. Levine, S.; Finn, C.; Darrell, T.; Abbeel, P. End-to-End Training of Deep Visuomotor Policies. *J. Mach. Learn. Res.* **2016**, *17*, 1334–1373.
12. Matas, J.; James, S.; Davison, A.J. Sim-to-Real Reinforcement Learning for Deformable Object Manipulation. *arXiv* **2018**, arXiv:1806.07851

13. Nagabandi, A.; Kahn, G.; Fearing, R.S.; Levine, S. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, 21–25 May 2018; pp. 7559–7566.
14. Berenson, D. Manipulation of deformable objects without modeling and simulating deformation. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 4525–4532.
15. Wang, A.; Kurutach, T.; Abbeel, P.; Tamar, A. Learning Robotic Manipulation through Visual Planning and Acting. In Proceedings of the Robotics: Science and Systems XV, Breisgau, Germany, 22–26 June 2019.
16. Agrawal, P.; Nair, A.; Abbeel, P.; Malik, J.; Levine, S. Learning to Poke by Poking: Experiential Learning of Intuitive Physics. In Proceedings of the Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, Barcelona, Spain, 5–10 December 2016.
17. Nair, A.; Chen, D.; Agrawal, P.; Isola, P.; Abbeel, P.; Malik, J.; Levine, S. Combining self-supervised learning and imitation for vision-based rope manipulation. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, 29 May–3 June 2017; pp. 2146–2153.
18. Kurutach, T.; Tamar, A.; Yang, G.; Russell, S.J.; Abbeel, P. Learning Plannable Representations with Causal InfoGAN. *arXiv* **2018**, arXiv:1807.09341
19. Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *arXiv* **2016**, arXiv:1606.03657
20. Srivastava, A.; Valkov, L.; Russell, C.; Gutmann, M.U.; Sutton, C. VAEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R., Eds.; pp. 3308–3318.
21. van den Oord, A.; Li, Y.; Vinyals, O. Representation Learning with Contrastive Predictive Coding. *arXiv* **2018**, arXiv:1807.03748
22. Arriola-Rios, V.E.; Güler, P.; Ficuciello, F.; Kragic, D.; Siciliano, B.; Wyatt, J.L. Modeling of Deformable Objects for Robotic Manipulation: A Tutorial and Review. *Front. Robot. AI* **2020**, *7*, 82. [CrossRef] [PubMed]
23. McConachie, D.; Ruan, M.; Berenson, D. Interleaving Planning and Control for Deformable Object Manipulation. In Proceedings of the Robotics Research, The 18th International Symposium, ISRR 2017, Puerto Varas, Chile, 11–14 December 2017; Amato, N.M., Hager, G., Thomas, S.L., Torres-Torriti, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10, pp. 1019–1036.
24. Chung, J.; Kastner, K.; Dinh, L.; Goel, K.; Courville, A.C.; Bengio, Y. A Recurrent Latent Variable Model for Sequential Data. In Proceedings of the Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, Montreal, QC, Canada, 7–12 December 2015; pp. 2980–2988.
25. Watter, M.; Springenberg, J.T.; Boedecker, J.; Riedmiller, M.A. Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images. *arXiv* **2015**, arXiv:1506.07365
26. Ha, J.S.; Park, Y.J.; Chae, H.J.; Park, S.S.; Choi, H.L. Adaptive Path-Integral Autoencoders: Representation Learning and Planning for Dynamical Systems. In Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, Montréal, QC, Canada, 3–8 December 2018.
27. Konidaris, G.; Kaelbling, L.P.; Lozano-Pérez, T. From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning. *J. Artif. Intell. Res.* **2018**, *61*, 215–289. [CrossRef]
28. Srinivas, A.; Jabri, A.; Abbeel, P.; Levine, S.; Finn, C. Universal Planning Networks: Learning Generalizable Representations for Visuomotor Control. *ICML* **2018**, 4739–4748.
29. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of the Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013, Lake Tahoe, NV, USA; 5–8 December 2013; Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q., Eds.; pp. 3111–3119.
30. Tian, Y.; Krishnan, D.; Isola, P. Contrastive Multiview Coding. In *Computer Vision*; Lecture Notes in Computer Science; Vedaldi, A., Bischof, H., Brox, T., Frahm, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2020; Volume 12356, pp. 776–794.
31. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G.E. A Simple Framework for Contrastive Learning of Visual Representations. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13–18 July 2020; Volume 119, pp. 1597–1607.
32. Kaiser, L.; Babaeizadeh, M.; Milos, P.; Osinski, B.; Campbell, R.H.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; et al. Model Based Reinforcement Learning for Atari. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
33. Zhuo, H.H.; Zha, Y.; Kambhampati, S.; Tian, X. Discovering Underlying Plans Based on Shallow Models. *ACM Trans. Intell. Syst. Technol.* **2020**, *11*, 1–30. [CrossRef]
34. Zhuo, H.H.; Kambhampati, S. Model-lite planning: Case-based vs. model-based approaches. *Artif. Intell.* **2017**, *246*, 1–21. [CrossRef]
35. Zhuo, H.H.; Yang, Q. Action-model acquisition for planning via transfer learning. *Artif. Intell.* **2014**, *212*, 80–103. [CrossRef]
36. Zhuo, H.H.; Muñoz-Avila, H.; Yang, Q. Learning hierarchical task network domains from partially observed plan traces. *Artif. Intell.* **2014**, *212*, 134–157. [CrossRef]
37. Zhuo, H.H. Recognizing Multi-Agent Plans When Action Models and Team Plans Are Both Incomplete. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–24. [CrossRef]

38. Feng, W.; Zhuo, H.H.; Kambhampati, S. Extracting Action Sequences from Texts Based on Deep Reinforcement Learning. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, 13–19 July 2018; pp. 4064–4070. [CrossRef]
39. Zhuo, H.H. Human-Aware Plan Recognition. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 3686–3693.