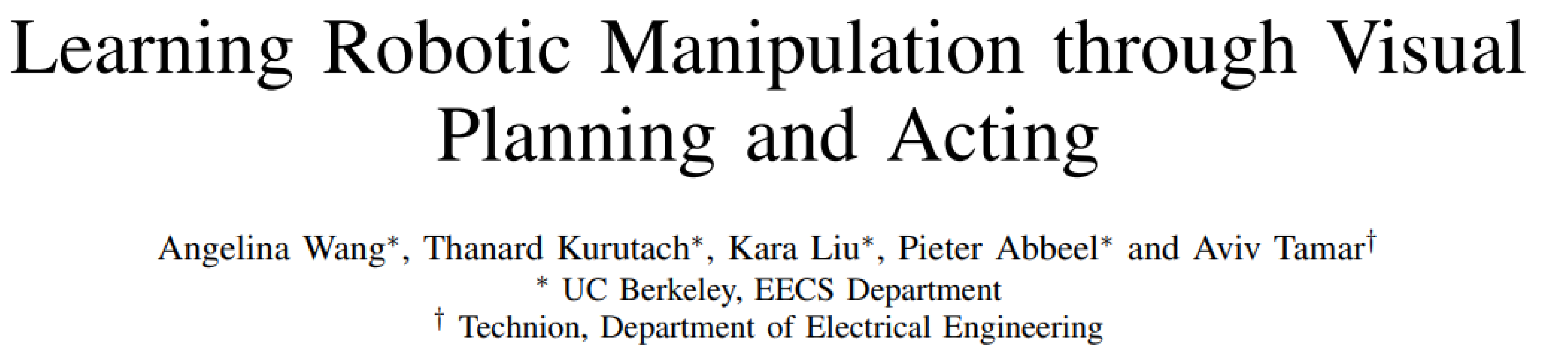
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Model-based Linear Deformable Object Manipulation

by Yunlong Wang – 12.01.2023

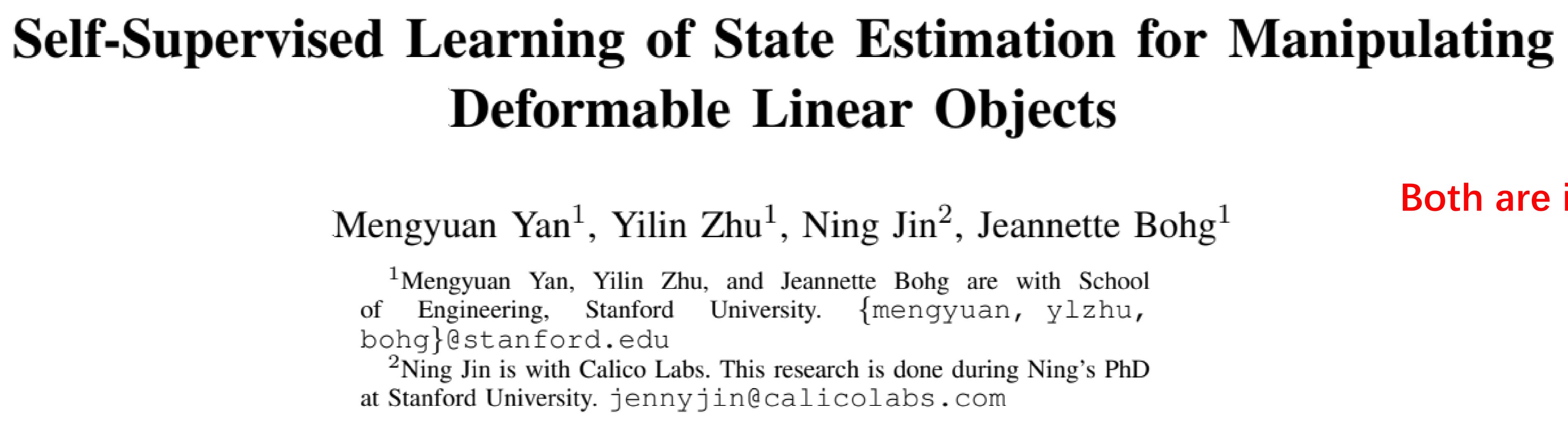
Outline

Robotics: Science and Systems 2019
Freiburg im Breisgau, June 22-26, 2019



Learning Robotic Manipulation through Visual Planning and Acting

Angelina Wang*, Thanard Kurutach*, Kara Liu*, Pieter Abbeel* and Aviv Tamar†
* UC Berkeley, EECS Department
† Technion, Department of Electrical Engineering



Self-Supervised Learning of State Estimation for Manipulating Deformable Linear Objects

Mengyuan Yan[1], Yilin Zhu[1], Ning Jin[2], Jeannette Bohg[1]

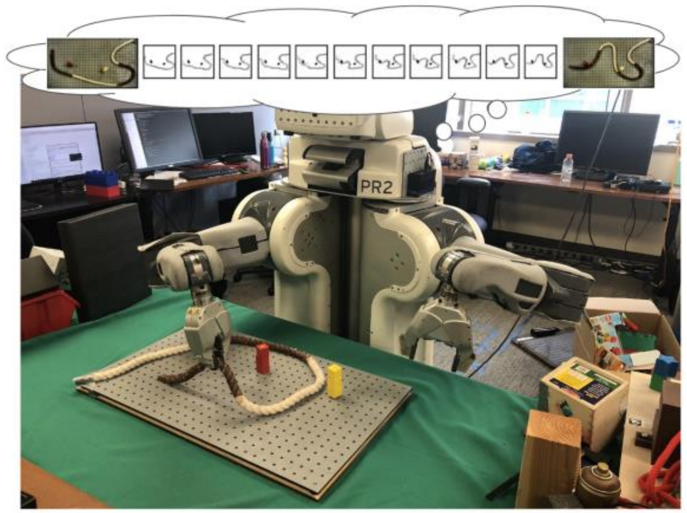
[1]Mengyuan Yan, Yilin Zhu, and Jeannette Bohg are with School of Engineering, Stanford University. {mengyuan, ylzhu, bohg}@stanford.edu
[2]Ning Jin is with Calico Labs. This research is done during Ning's PhD at Stanford University. jennyjin@calicolabs.com

Both are independent study!!!

1 Introduction

What is linear deformable object manipulation?

E.g. rope manipulation.



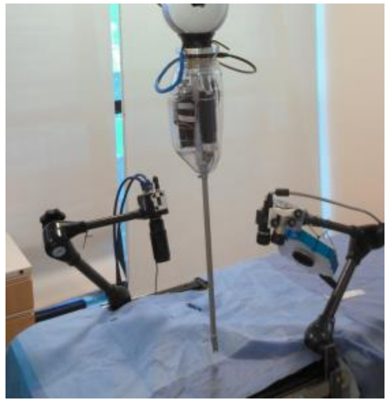
A. Wang et.al 2019

M. Yan et.al 2020

“Move the rope from current state the goal state”

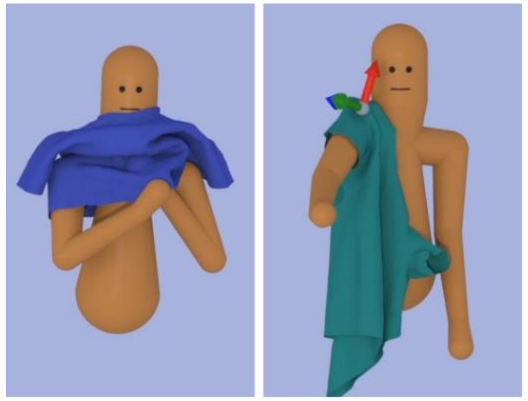
Some applications for deformable object manipulation:

robotic surgery



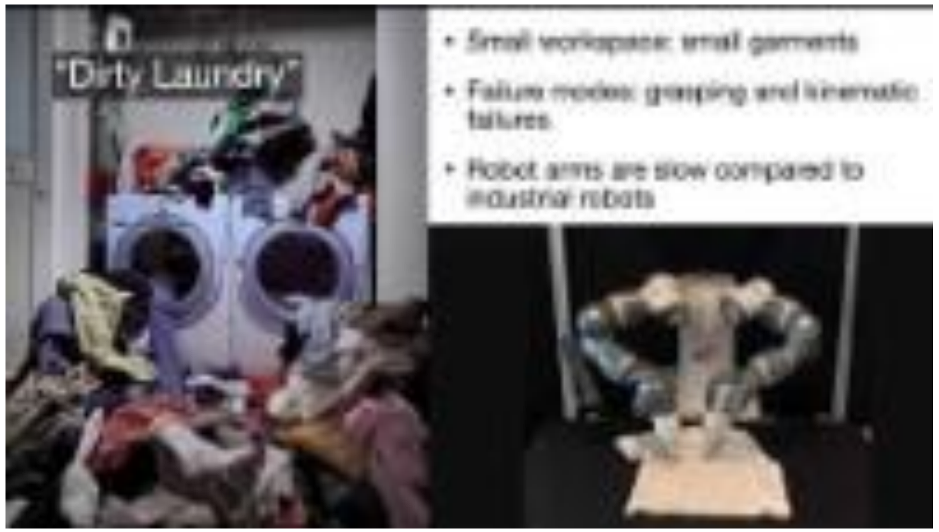
S. Leonard 2014

assistive dressing



A. Clegg 2018

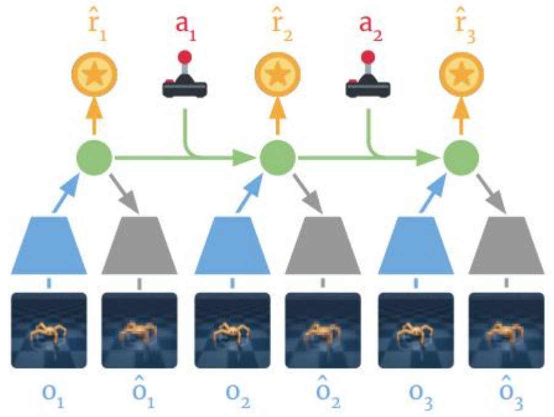
folding clothes



- Small workspace: small garments
- Failure modes: grasping and kinematic failures
- Robot arms are slow compared to industrial robots

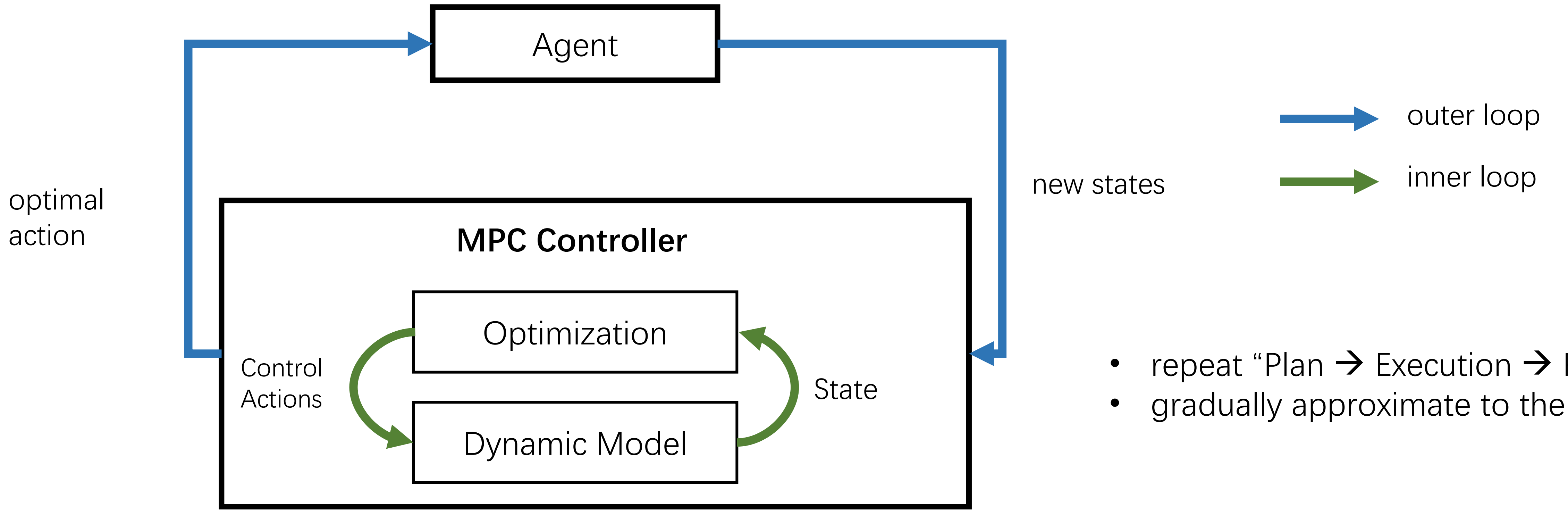
What is **Model-based Approach**?

Learn a model of the environment, i.e., dynamic model, such as Dreamer(D. Hafner 2019)



D. Hafner 2019

What is **Model Predictive Control**?



- repeat “Plan → Execution → Replan”
- gradually approximate to the goal

2 Motivation

Why use the Model-based method?

model-free:

- has black-box policy, it is **hard to interpret.**
- **can not look ahead** over time horizon.

Why use learnable method to learn the model:

Deformable objects have a **high-dimensional state space** ➔ **learnable method**

Why use Model Predictive Control(MPC)?

- The deformable object is **hard to manipulate,** i.e., the actual action result could be differed from plan.

  ➔ use **MPC** to gradually **approximate** to the goal.

3 Methodology

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

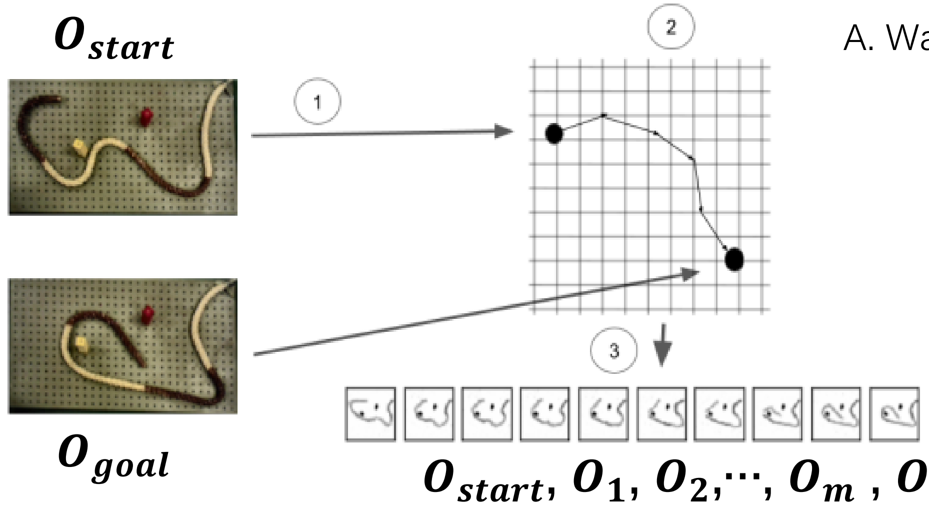
3.1 Learning Robotic Manipulation through Visual Planning and Acting(A. Wang, 2019)

Visual planning and Acting(VPA)

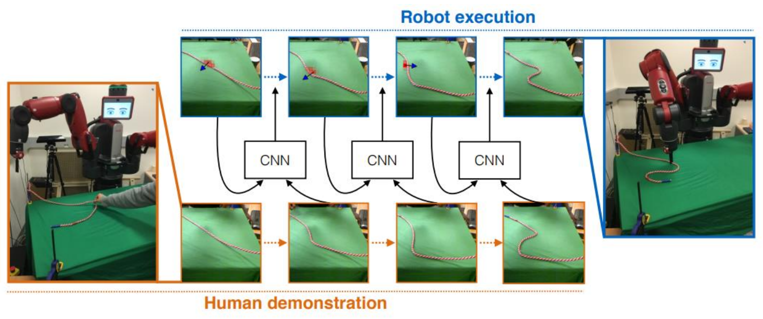
A. Wang, 2019

1) **Plan:** given a pair, o_{start}, o_{goal}, use the CIGAN model M_{CIGAN} to generate a planned sequence of observations $o_{start}, o_1, \ldots, o_m, o_{goal}$.

① Encoding the O_{start} and O_{goal} to latent space.
② Using A^* search to find feasible path in latent space.
③ Decoding the latent path to sequence of images(Visual plan).



2) **Act:** If the length of the plan m is zero, take an action u to reach the goal $u = M_{\text{IM}}(o_{start}, o_{goal})$, then stop. Else:
3) Take an action u to reach the first observation in the plan $u = M_{\text{IM}}(o_{start}, o_1)$ and take a new observation of the current system state o_{new}.

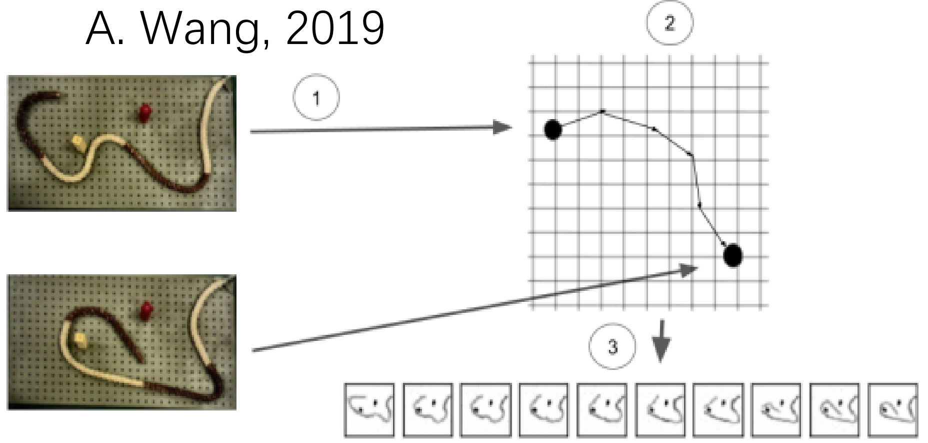


An inverse dynamics model(A. Nair, 2017) by supervised learning

4) **Replan:** update o_{start} to be the current observation o_{new}, and go back to step 1.

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

A. Wang, 2019



Visual planning and Acting

For planning at ②, there are some interesting question need to be considered:

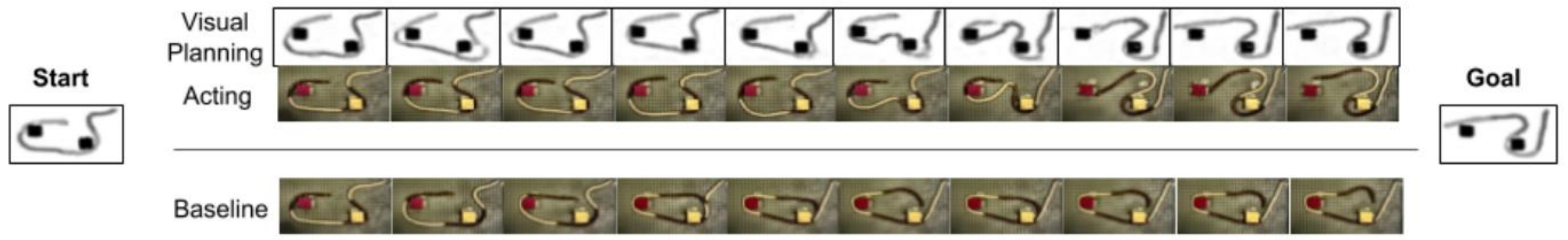
- **why directly searching in the latent space can generate plausible result?**
  - Most of time, the latent space is chaos and implicit, therefore searching is meaningless.
  - But, because of the **special design of InfoGAN**(X. Chen et al. 2016), the **latent space is interpretable.** Hence, searching in the latent space is meaningful.

- **why using A⋆ search?**
  - In the **Causal InfoGAN**(T. Kurutach 2018), **A simple linear interpolation** was used for latent space searching, and **can't not produce realistic result** by rope manipulation, e.g. when in the presence of obstacles, the visual plan tended to go through rather than around, which is unrealistic.
    ➔ the A⋆ search is applied, use Euclidean distance as a heuristic function

Experiment setup and result

Manipulate the rope on PR2 robot.

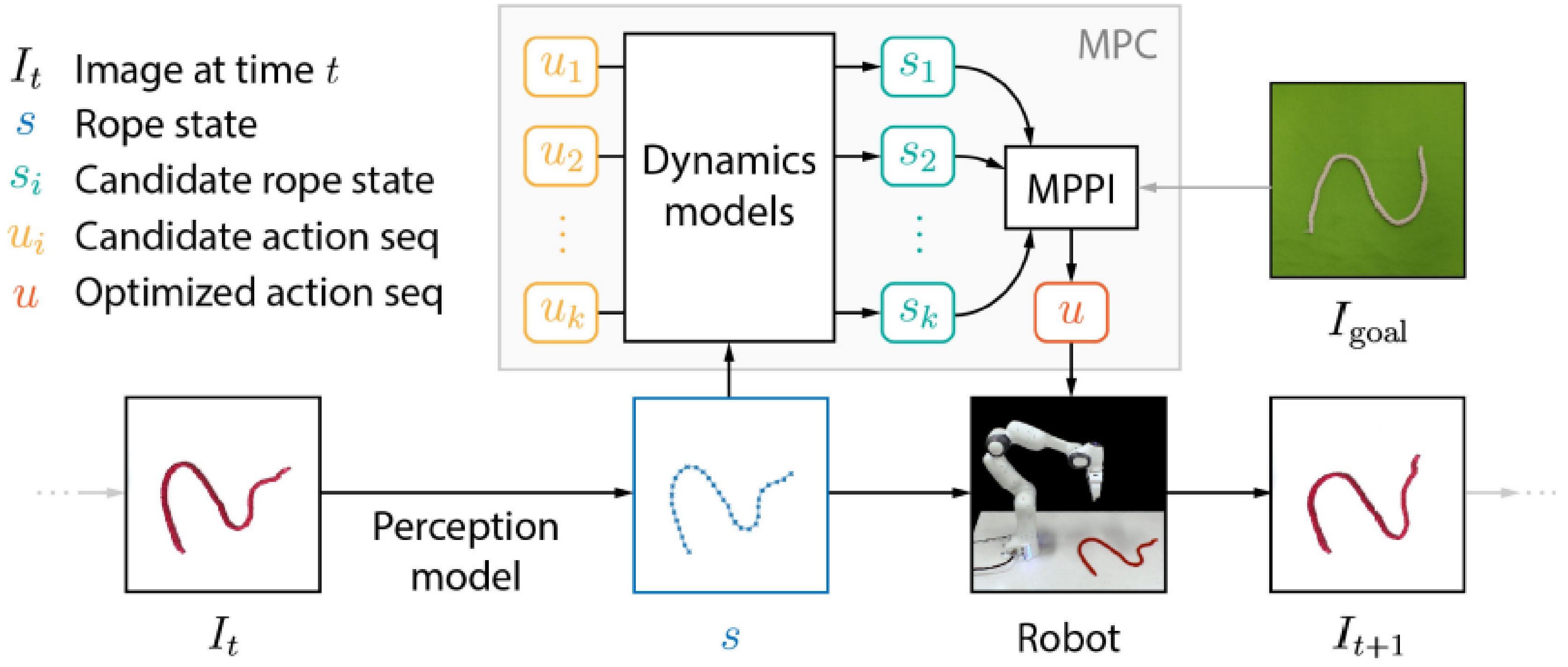
A. Wang, 2019



Baseline denote directly apply inverse model between $O_{start}$and O_{goal}. And it show it can not reach the goal.

3.2 Self-Supervised Learning of State Estimation for Manipulating Deformable Linear Objects(M. Yan, 2020)

Overview of system

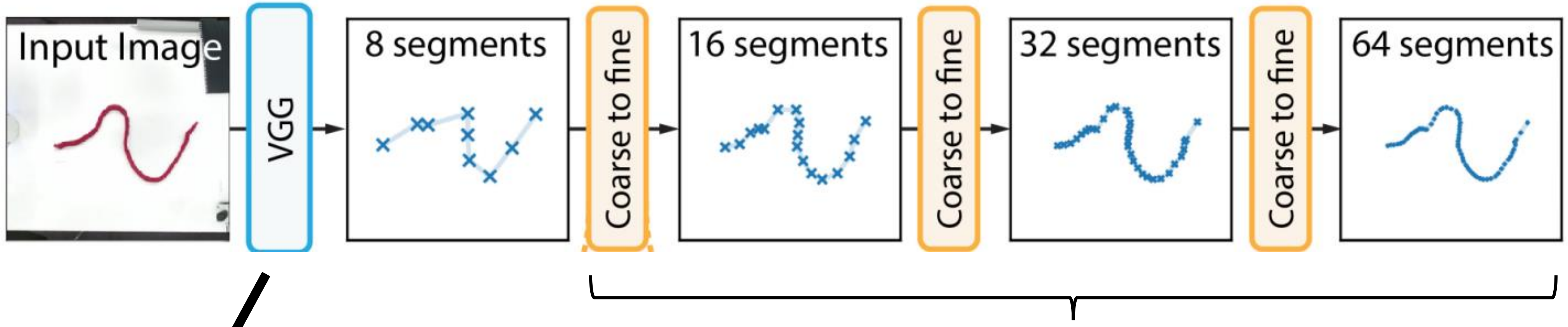


M. Yan, 2020

Perception Model: Coarse-to-fine State estimation

Input: RGB Image
Output: 64 sequence of segments, i.e. 65 points
First train on rendered image, then fine-tuning on real image.

M. Yan et.al 2020

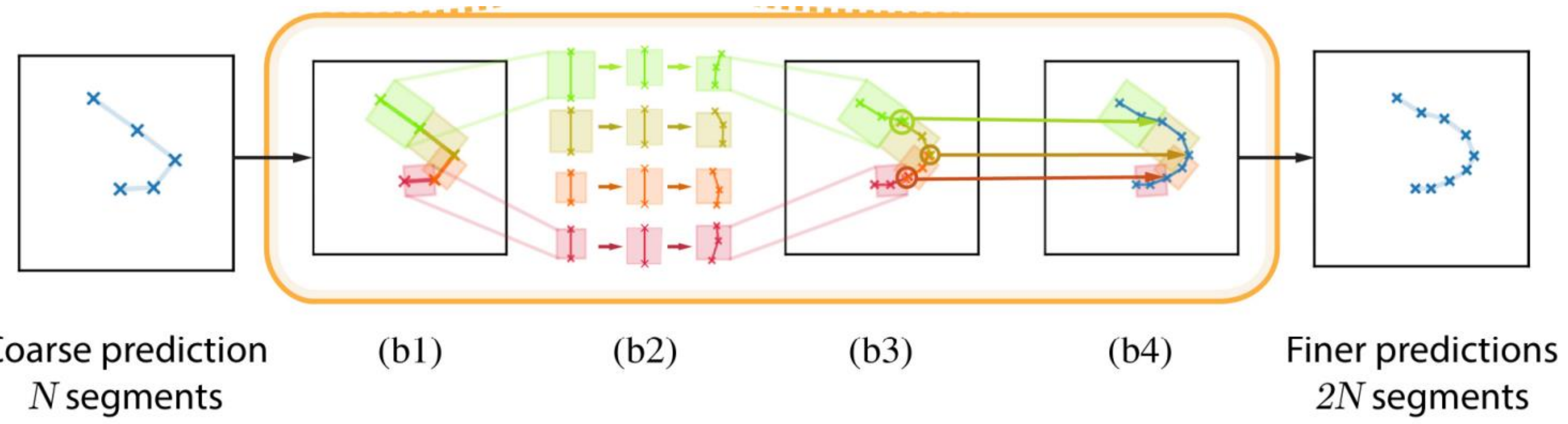


"double the segmentation for each time"

using VGG to estimate
8 straight segments.

Perception Model: **Coarse-to-fine block**

M. Yan et.al 2020



(b1) **Extract square boxes** that defined by the previously estimated segments.

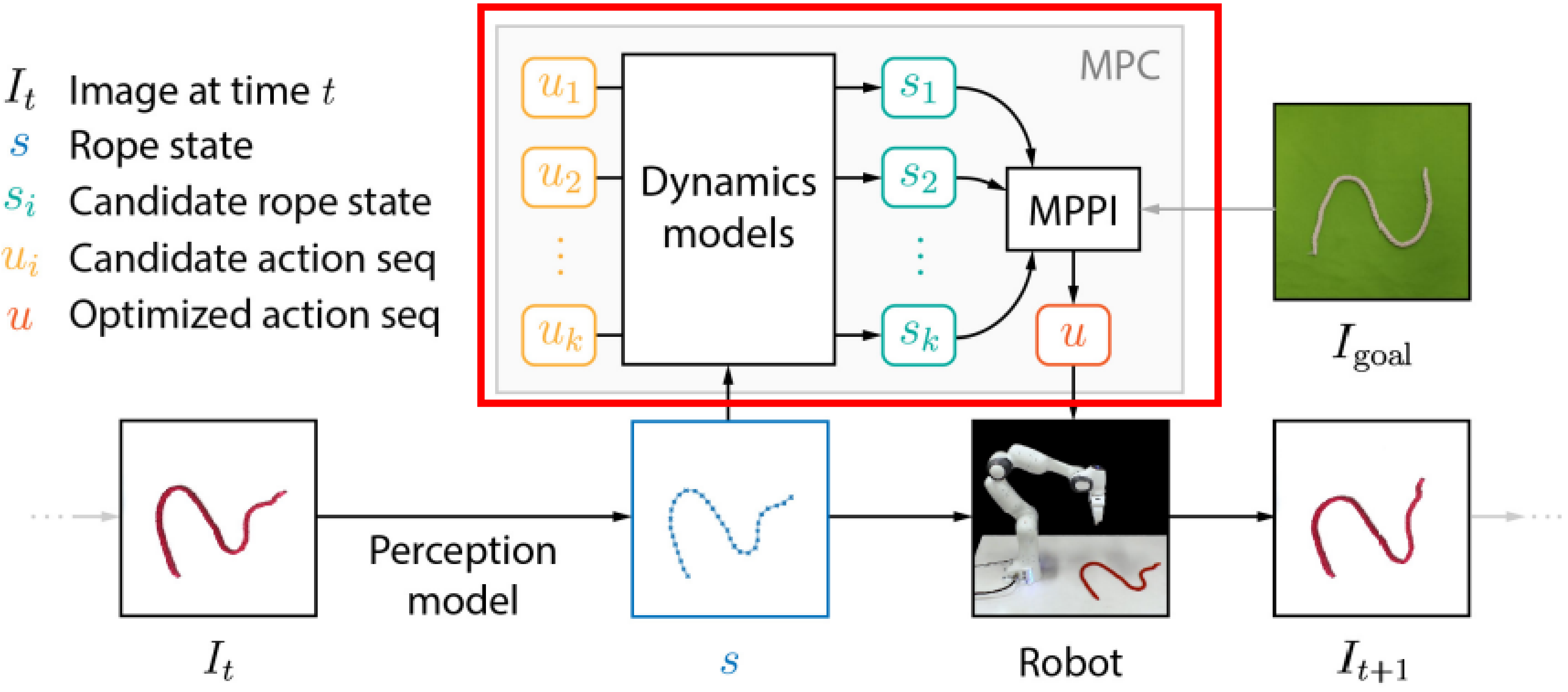
(b2) The boxes are used to extract regions from the VGG feature maps, and fed into a MLP to **estimate the left endpoint, middle point, and right endpoint** in each square region.

(b3) The estimated points are **concatenated**

(b4) endpoints from neighboring regions are **averaged** to obtain a higher resolution estimate, based on twice the amount of segments that more closely model the shape of the rope.

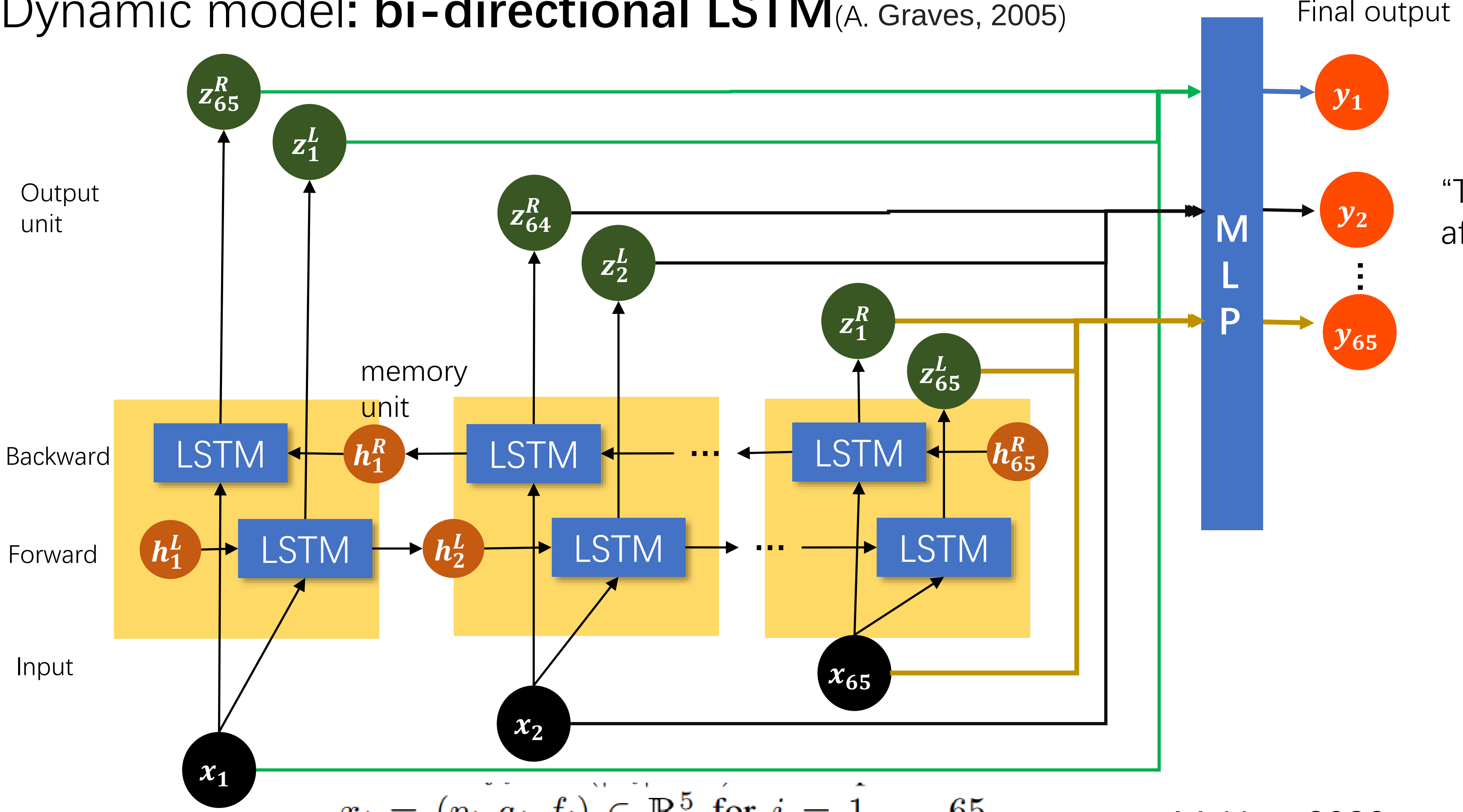
Overview of system

I_t	Image at time t
s	Rope state
s_i	Candidate rope state
u_i	Candidate action seq
u	Optimized action seq



M. Yan et.al 2020

Dynamic model: **bi-directional LSTM**(A. Graves, 2005)



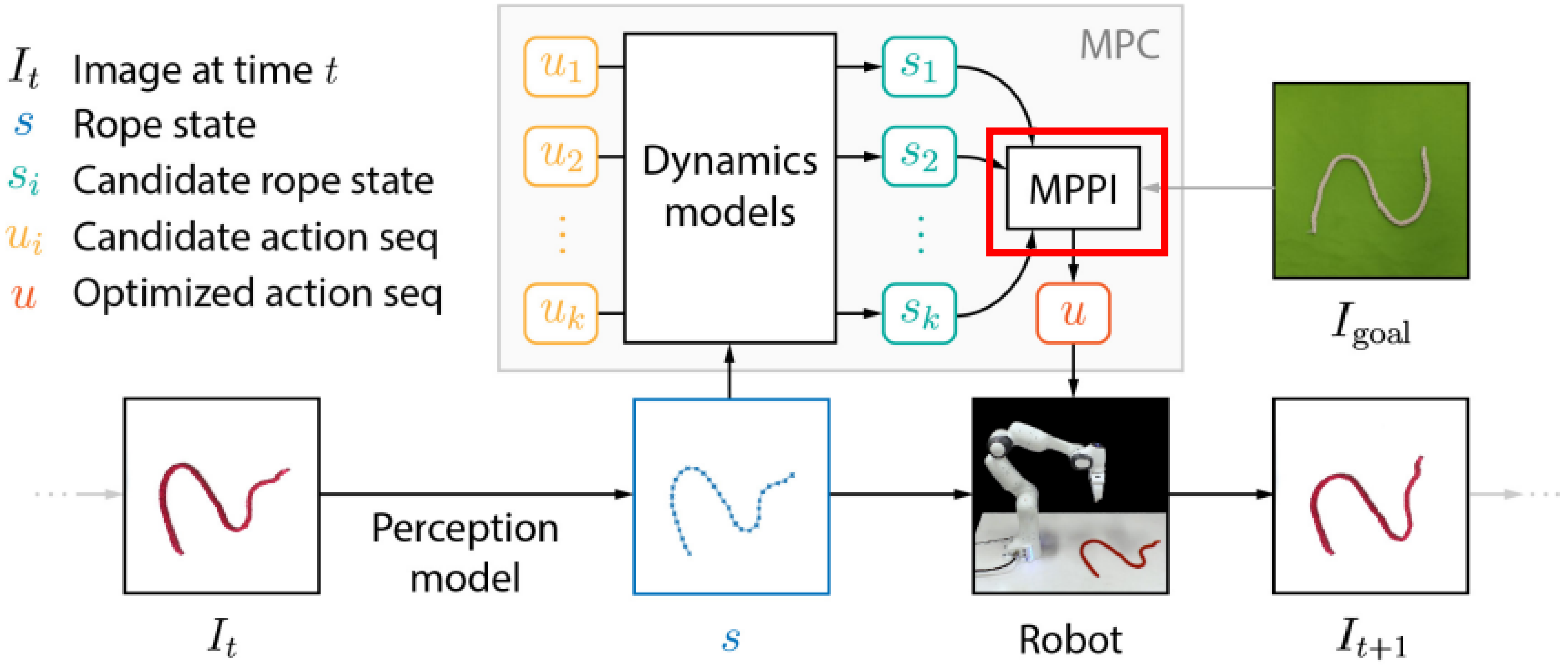
"The position of ith state point after the action is performed"

$$x_i = (p_i, a_i, f_i) \in \mathbb{R}^5 \text{ for } i = 1, \ldots, 65.$$

action $a_i \in \mathbb{R}^2$ position $p_i \in \mathbb{R}^2$ indicator $f_i = \mathbb{1}(|a_i| > 0)$.

M. Yan, 2020

Overview of system



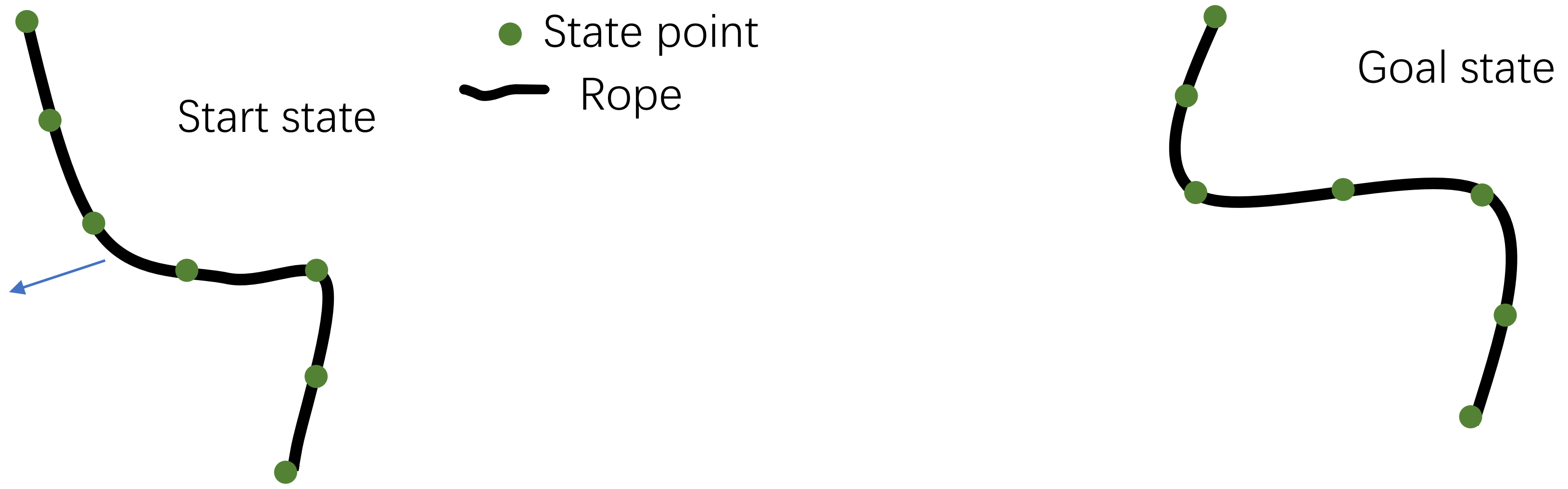
M. Yan et.al 2020

Sampling-based **M**odel **P**redictive **P**ath **I**ntegral control (G. Williams, 2016)

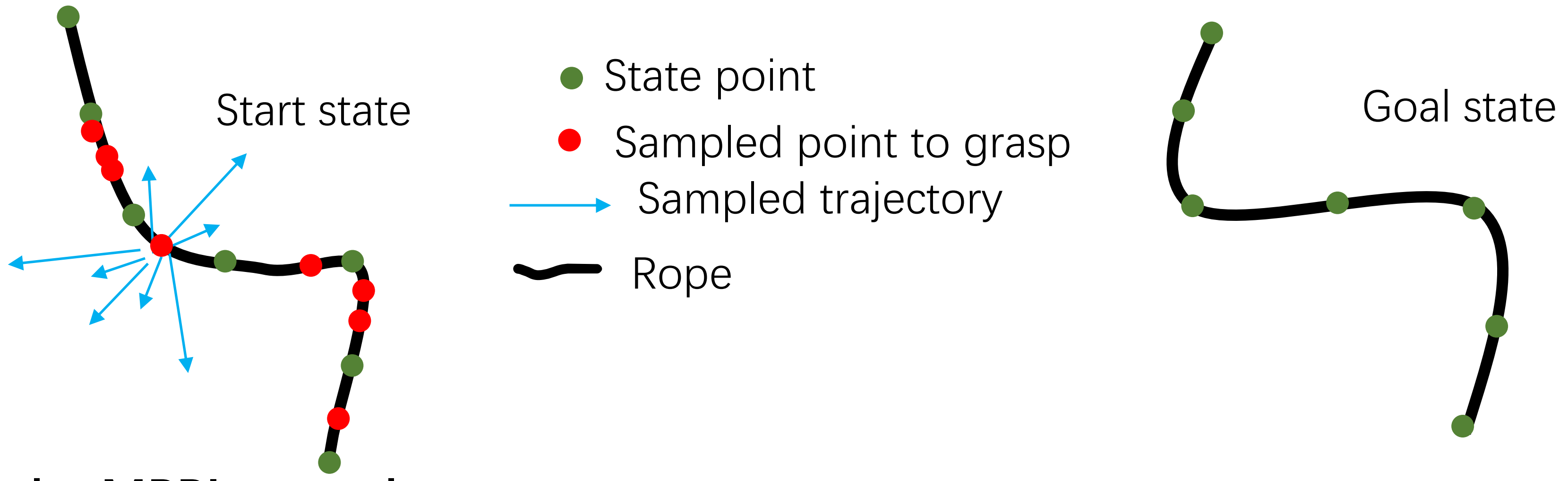
Action planning are formulated as two step:

1. Find the point to grasp the rope.
2. Move the gripper and the rope being grasped in 2D.

For example:



Cont.



The pipeline of the MPPI control:

```
sampling on the rope to get candidate grasp points : $Points_{grasp}$;
for each point in $Points_{grasp}$:
        sampling $n$ candidate action sequences;
        for each action sequence:
                $T_{trajectoris}$ = $Dynamic\_model(actions, states)$
                cost for each state in $T_{trajectoris}$ = distance to goal state
        optimal trajectory per grasping points = **cost-weighted average** of all sampled trajectories
optimal grasp point = the one with lowest cost of its optimal trajectory
```

Experiment setup and result

A. Perception networks comparison

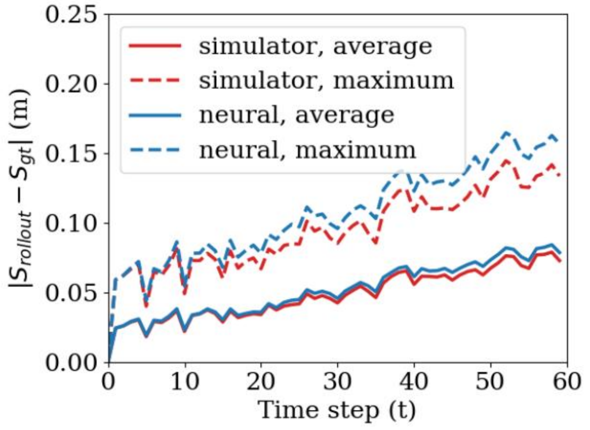
M. Yan et.al 2020

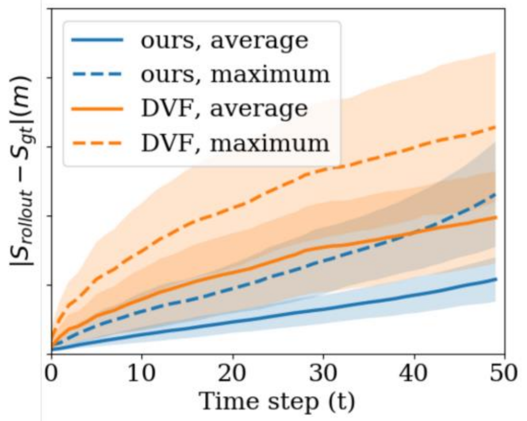
	Train	Test
Baseline: Direct Estimate	0.0104	0.0354
Ours: Coarse-to-Fine	0.0177	**0.0231**

Mean Square Error of the Euclidean distance

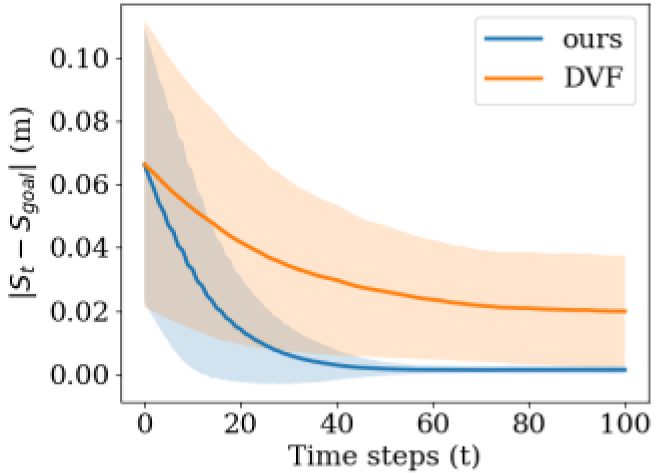
B. Learning dynamics models

M. Yan et.al 2020





C. Overall performance



DVF: visual dynamics model(F. Ebert, 2018)

4 Summary

4.1 Summary for both paper

The first paper(A. Wang 2019):

- **Visual planning**
  - **Encoder**: image ➔interpretable latent space
  - **Searching**: search in latent space ➔ feasible path
  - **Decoder:** path ➔ sequence of images(Visual Plan)
- **Acting**
  - Learnable Inverse Dynamics Models (base on CNN): Visual plan➔ sequence of actions

The Second paper(M. Yang 2020):

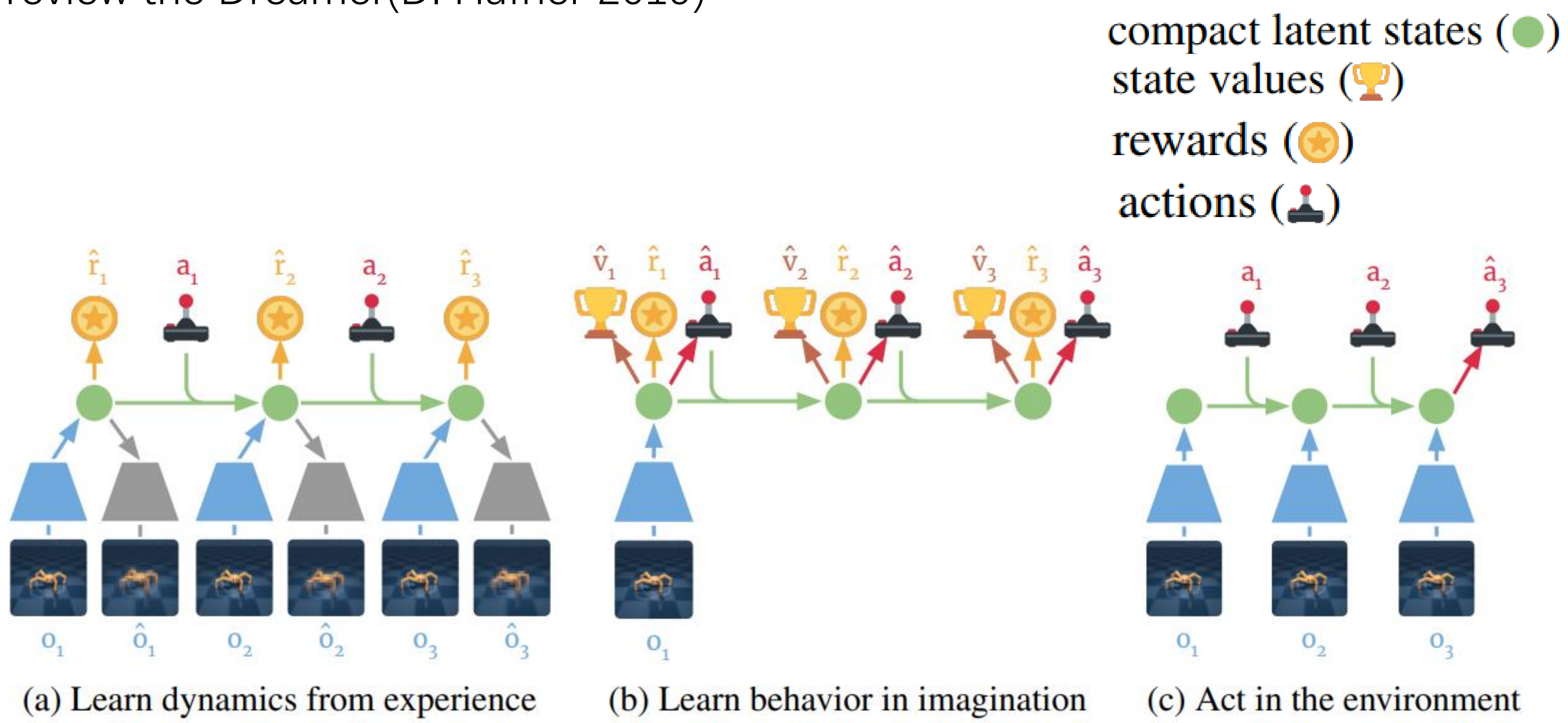
- **Rope state estimator:** self-supervised learning, produce explicit rope state
- **Model Predictive control**
  - **Dynamic model**: $s_{t+1} = Dynamic(s_t, a_t)$
  - **Model Predictive Path Integral control:** combine dynamic model to produce optimal action

The core idea for both paper:

How to extract an interpretable latent space, in which the search or optimization can generate meaningful results.

4.2 Methodology comparison

First, let we review the Dreamer(D. Hafner 2019)

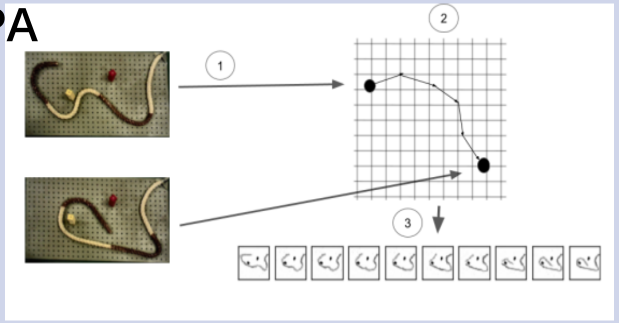
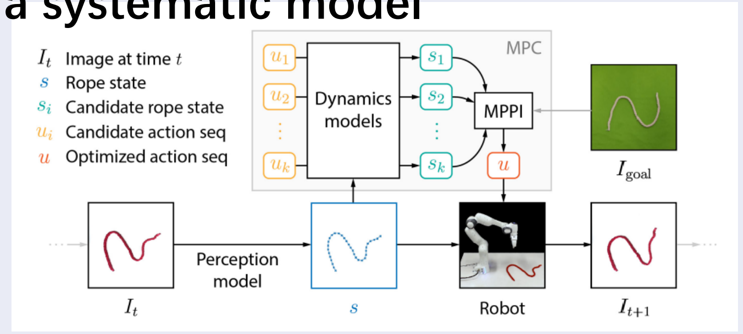
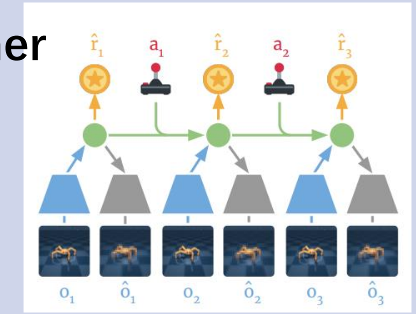


compact latent states (●)
state values (🏆)
rewards (⭐)
actions (🕹)
$\hat{r}_1$ a_1 $\hat{r}_2$ a_2 $\hat{r}_3$
o_1 $\hat{o}_1$ o_2 $\hat{o}_2$ o_3 $\hat{o}_3$
$\hat{v}_1$ $\hat{r}_1$ $\hat{a}_1$ $\hat{v}_2$ $\hat{r}_2$ $\hat{a}_2$ $\hat{v}_3$ $\hat{r}_3$ $\hat{a}_3$
o_1
a_1 a_2 $\hat{a}_3$
o_1 o_2 o_3

(a) Learn dynamics from experience (b) Learn behavior in imagination (c) Act in the environment

D. Hafner 2019

The Dreamer:
- Learn dynamic model in latent space
- Learn a RL policy to choose action

	State space	Planning method
VPA 	**Latent** state space, **interpretable**	**A* Search** in latent space, then use inverse model to generate action for visual plan
a systematic model  I_t Image at time t s Rope state s_i Candidate rope state u_i Candidate action seq u Optimized action seq MPC u_1 u_2 u_k Dynamics models s_1 s_2 s_k MPPI u I_{goal} Perception model I_t s Robot I_{t+1}	**Explicit** state space, **interpretable**	Use **MPC** to choose action from explicit rope state.
Dreamer  $\hat{r}_1$ a_1 $\hat{r}_2$ a_2 $\hat{r}_3$ o_1 $\hat{o}_1$ o_2 $\hat{o}_2$ o_3 $\hat{o}_3$	**Latent** state space, **uninterpretable**	Learn a **policy** to choose action from latent space.

Some Inspirations from my opinion

1. **For plan/control by given different of sate space**, we can:
   - **Uninterpretable** latent space ➔ Learn a **policy**
   - **interpretable** latent space ➔ **searching** algorithm
   - **Explicit** state space ➔ **Modal Predictive Control**, also when state space is small, lots of other method can be applied.

2. **GMM+EM** is a good idea for self-supervised objective when the task is two class segmentation.

5 Reference

[1] Wang A, Kurutach T, Liu K, et al. Learning robotic manipulation through visual planning and acting[J]. arXiv preprint arXiv:1905.04411, 2019.
[2] Yan M, Zhu Y, Jin N, et al. Self-supervised learning of state estimation for manipulating deformable linear objects[J]. IEEE robotics and automation letters, 2020, 5(2): 2372-2379.
[3] Jaderberg M, Simonyan K, Zisserman A. Spatial transformer networks[J]. Advances in neural information processing systems, 2015, 28.
[4] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial networks[J]. Communications of the ACM, 2020, 63(11): 139-144.
[5] Chen X, Duan Y, Houthooft R, et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets[J]. Advances in neural information processing systems, 2016, 29.
[6] Kurutach T, Tamar A, Yang G, et al. Learning plannable representations with causal infogan[J]. Advances in Neural Information Processing Systems, 2018, 31.
[7] Nair A, Chen D, Agrawal P, et al. Combining self-supervised learning and imitation for vision-based rope manipulation[C]//2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017: 2146-2153.
[8] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures[J]. Neural networks, 2005, 18(5-6): 602- 610.
[9] Hafner D, Lillicrap T, Ba J, et al. Dream to control: Learning behaviors by latent imagination[J]. arXiv preprint arXiv:1912.01603, 2019.
[10] Williams G, Drews P, Goldfain B, et al. Aggressive driving with model predictive path integral control[C]//2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016: 1433-1440
[11] Ebert F, Finn C, Dasari S, et al. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control[J]. arXiv preprint arXiv:1812.00568, 2018.

6 Appendix

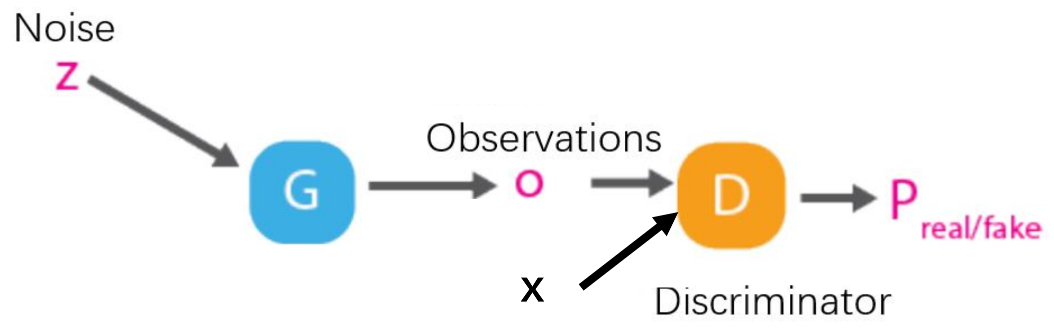
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

6.1 Learning Robotic Manipulation through Visual Planning and Acting

GAN(Goodfellow I, 2020)

"The real image is real" "The fake image is fake"

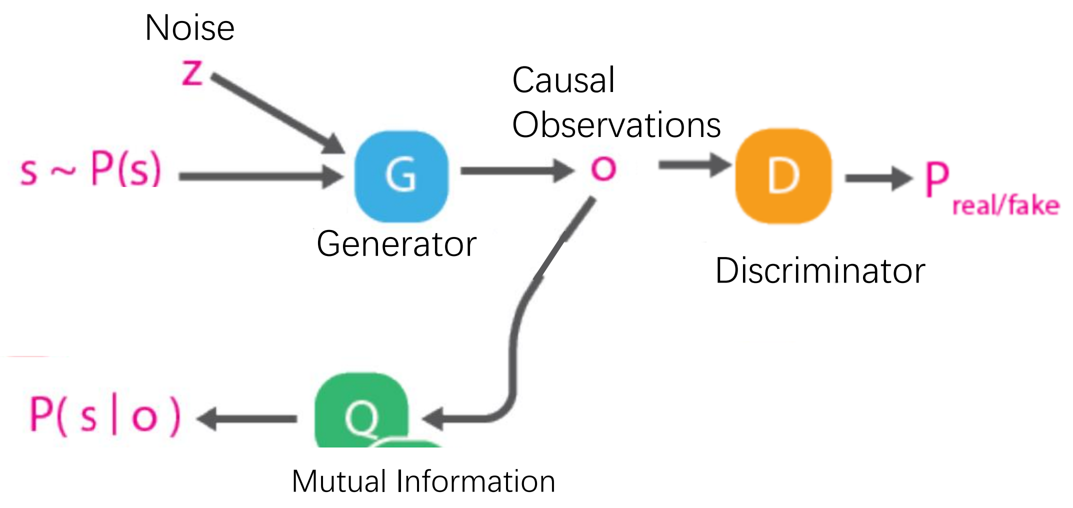
$$\min_G \max_D V(D,G) = \underline{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]} + \underline{\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}.$$



A. Wang et.al 2019

InfoGAN(X. Chen 2016)

$$\min_G \max_D V(G,D) - \boxed{\lambda I(s; G(z,s))}$$

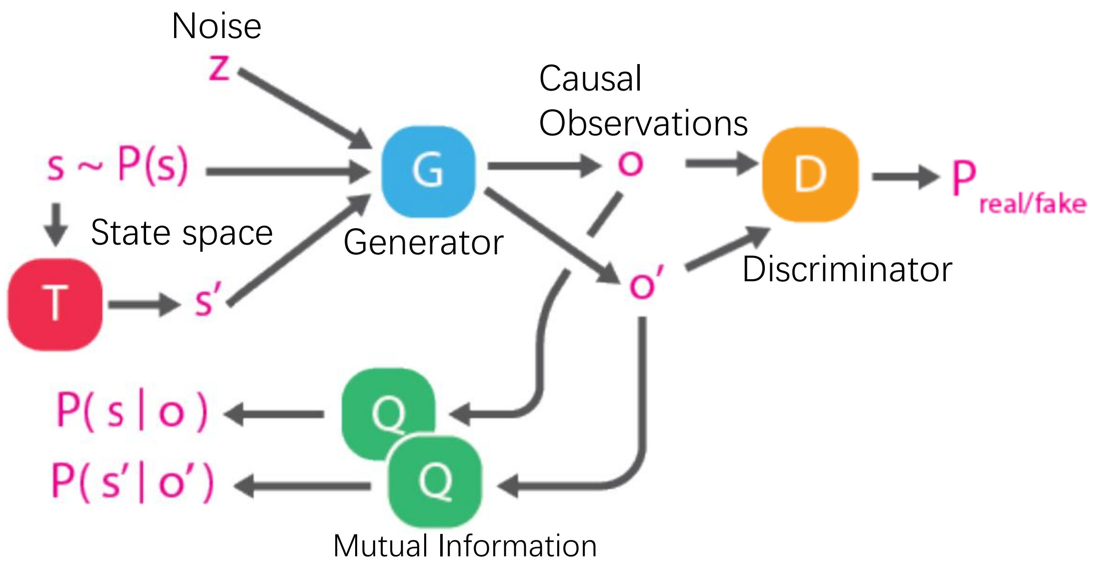


- The input have two parts:
  noise **z** and structured component **s**
- Mutual information is used

Causal InfoGAN(T. Kurutach, 2018)

$$\min_{P(s'|s),G} \max_{D} V(G,D) - \lambda I(s,s';o,o'),$$
$$\text{s.t. } o,o' \sim G(z,s,s'), s \sim P(s), s' \sim P(s'|s)$$

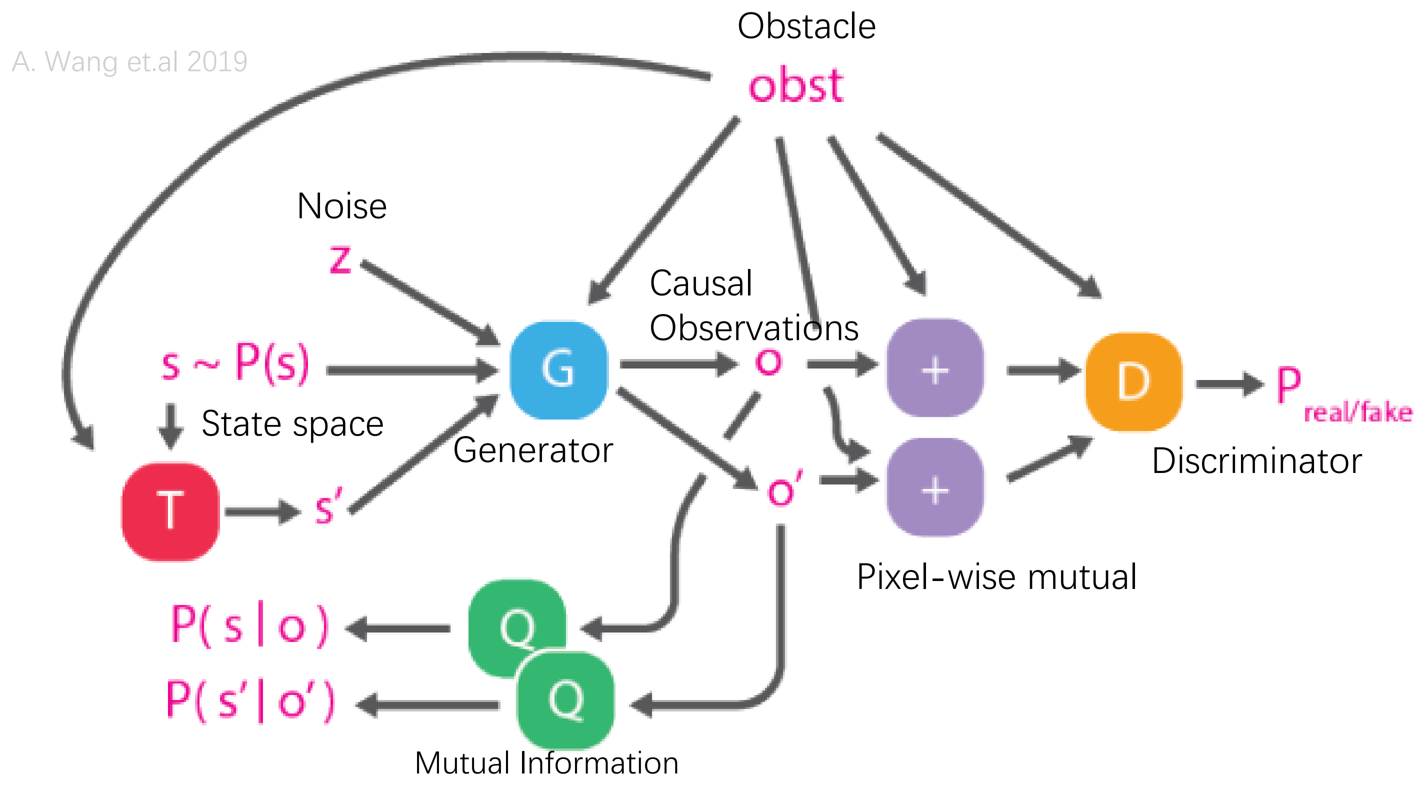
- The CIGAN model learns to generate a pair of sequential observations (o, o') that are similar to sequential observations in the data.
- S': modelled as a local perturbation of the first state s
- **additional learning of dynamics** in latent space



A. Wang et.al 2019

For encoding, Kurutach et al. [21] used an optimization based approach, searching for a latent vector that minimizes the absolute pixel difference with the desired observation. For the planning, the key idea in [21] is that due to the local transition structure of $P(s'|s)$, linear interpolation between s_{start} and s_{goal} results in a feasible plan. In this sense, CIGAN learns a representation that is *compatible* with the planning algorithm. For decoding, the CIGAN generator can be used to sequentially produce pairs of observations from the trajectory.

Context Conditional Causal InfoGAN(A. Wang 2019)



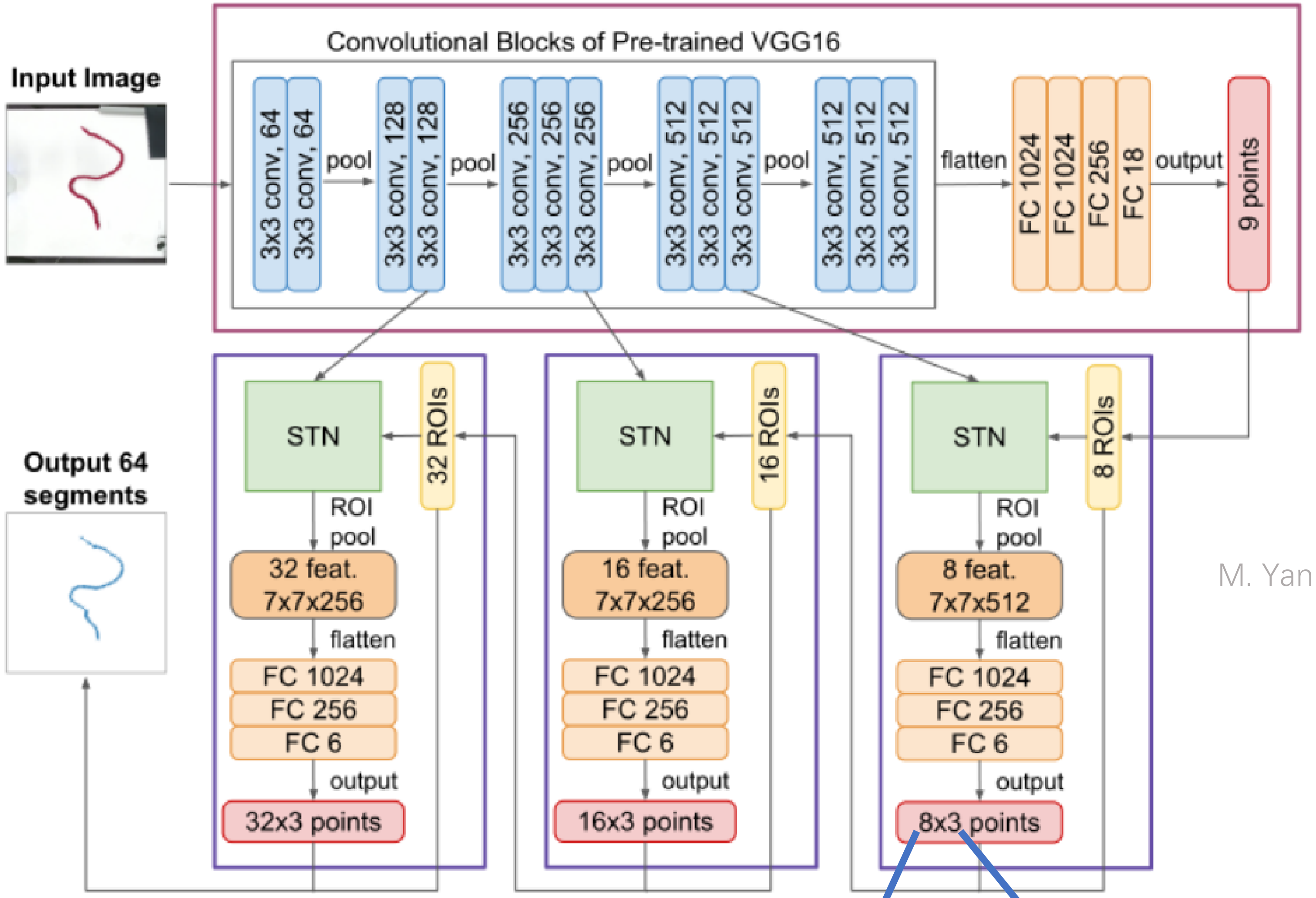
1. Encoding the o_{start} and o_{goal} to latent space s and s'.
2. A* Search in the latent space $P(s'|s)$ to get plausible state path from o_{start} to o_{goal}.
3. Use generator to generate the rope image from state path.

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

6.2 Self-Supervised Learning of State Estimation for Manipulating Deformable Linear Objects

Perception Model:

Detailed architecture



M. Yan et.al 2020

8 segments

Left endpoint,
middle point,
Right endpoint.

Perception Model: **self-supervising learning objective**

M. Yan et.al 2020

Firstly, introduce the gaussian mixture model(GMM) with expectation maximization(EM):

E-step:

component weights

RGB value

means

covariance matrices

Membership weight of pixel with index i that belong to gaussian distribution k

$$w_{ik} = \frac{\alpha_k p_k(x_i|\mu_k, \Sigma_k)}{\sum_{m=1}^{K} \alpha_m p_m(x_i|\mu_m, \Sigma_m)} \tag{1}$$

k: index of gaussian, in total as K
i: index of pixel

M-step:

$$\alpha_k^{new} = \sum_{i=1}^{N} w_{ik}/N,$$
$$\mu_k^{new} = \sum_{i=1}^{N} w_{ik} x_i / \sum_{i=1}^{N} w_{ik}, \tag{2}$$
$$\Sigma_k^{new} = \sum_{i=1}^{N} w_{ik}(x_i - \mu_k^{new})(x_i - \mu_k^{new})^T / \sum_{i=1}^{N} w_{ik}.$$

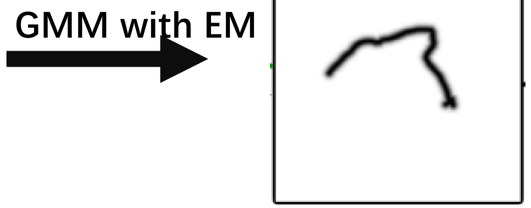
N: total number of pixels

Perception Model: **self-supervising learning objective**

M. Yan et.al 2020

Now, we have the ground-truth label as rope segmentation.

Input image



Rope segmentation

But, the output of our perception model is rope state, not rope segmentation.
Therefore, **How to calculate the loss between binary segmentation and rope state?**

➔ Let membership weight linked with rope state, and here let we use $P(u, v)$ denote the membership weights W_{ik} belong to the rope.

➔ That is, to propose a new $P(u, v)$:

$$P_j(u,v) = \exp\left(-d_j(u,v)^2/\sigma^2\right)$$
$$P(u,v) \;=\; \max_j P_j(u,v)$$

$d_j(u, v)$: denote the distance of pixel (u, v) to its closest points on rope segments j.
σ: denote a learnable parameter that controls the width of the rendered segments.

Perception Model: **self-supervising learning objective**

M. Yan et.al 2020

After the modified membership weighs, the EM for GMM has been changed:

Original EM for GMM

E-step:

$$w_{ik} = \frac{\alpha_k p_k(x_i|\mu_k, \Sigma_k)}{\sum_{m=1}^{K} \alpha_m p_m(x_i|\mu_m, \Sigma_m)}$$

M-step:

$$\alpha_k^{new} = \sum_{i=1}^{N} w_{ik}/N,$$

$$\mu_k^{new} = \sum_{i=1}^{N} w_{ik} x_i / \sum_{i=1}^{N} w_{ik},$$

$$\Sigma_k^{new} = \sum_{i=1}^{N} w_{ik}(x_i - \mu_k^{new})(x_i - \mu_k^{new})^T / \sum_{i=1}^{N} w_{ik}.$$

New EM procedure

$$P(u,v) = \max_j P_j(u,v)$$
$$P_j(u,v) = \exp\left(-d_j(u,v)^2/\sigma^2\right)$$

$$\sum_{(u,v)} -\log\left[P(u,v)P(u,v)^{new} + (1-P(u,v))(1-P(u,v)^{new})\right].$$

How to calculate the loss value from self-supervising learning objective:

Clustering in RGB space can be achieved with the *Expectation-Maximization* (EM) algorithm for *Gaussian Mixture Models* (GMM). Given an initial estimate of GMM parameters Θ, i.e., the component weights α_k, means μ_k, and covariance matrices Σ_k, $1 \leq k \leq K$, the EM algorithm iterates between the E step, which updates the membership weights w_{ik} of data point x_i to cluster k, and the M step, which updates Θ. In the E step, membership weights are updated as:

$$w_{ik} = \frac{\alpha_k p_k(x_i|\mu_k, \Sigma_k)}{\sum_{m=1}^{K} \alpha_m p_m(x_i|\mu_m, \Sigma_m)} \qquad (1)$$

where p_k and p_m are multivariate Gaussian densities. In the M step, GMM parameters are updated as:

$$\begin{aligned} \alpha_k^{new} &= \sum_{i=1}^{N} w_{ik}/N, \\ \mu_k^{new} &= \sum_{i=1}^{N} w_{ik} x_i \Big/ \sum_{i=1}^{N} w_{ik}, \\ \Sigma_k^{new} &= \sum_{i=1}^{N} w_{ik}(x_i - \mu_k^{new})(x_i - \mu_k^{new})^T \Big/ \sum_{i=1}^{N} w_{ik}. \end{aligned} \qquad (2)$$

We propose a differentiable renderer that links $P(u, v)$ to the rope state. The rope state is a sequence of 64 segments. We individually render each segment with end-points p_j, p_{j+1} to get $P_j(u, v)$, and take the pixel-wise maximum, $P(u, v) = \max_j P_j(u, v)$. Rendering of one segment is defined as

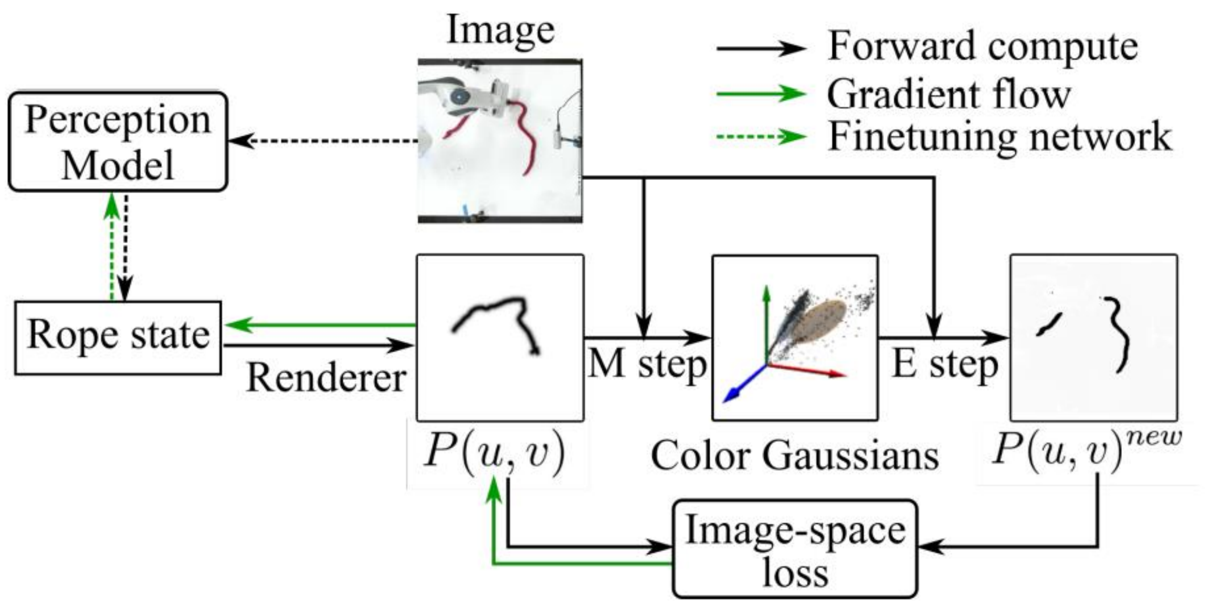
$$P_j(u, v) = \exp\left(-d_j(u, v)^2/\sigma^2\right) \qquad (3)$$

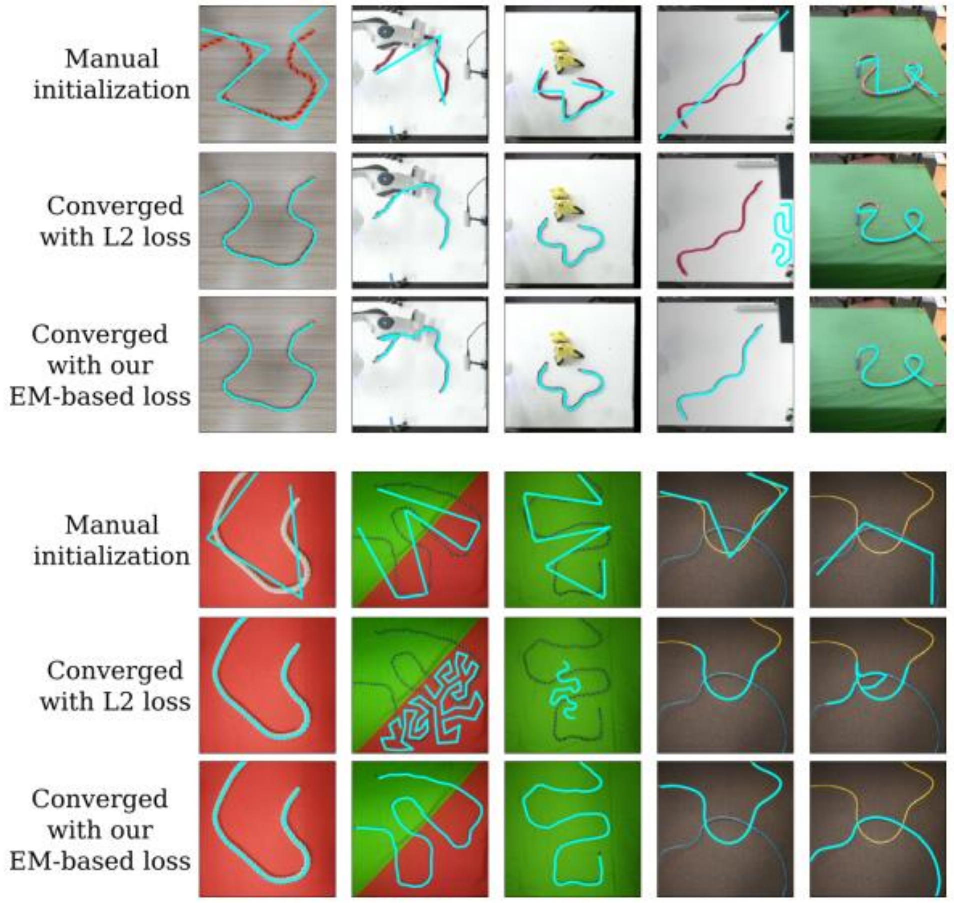
where $d_j(u, v)$ is the distance of pixel (u, v) to its closest point on segment j. σ is a learnable parameter that controls the width of the rendered segments.

Given the initial state estimate from the neural network, we can compute $P(u, v)$ based on Eq. 3. The M step can be applied easily to compute the parameters of the Gaussian clusters in RGB space. We also follow the E step in Eq. 1 to calculate $P(u, v)^{new}$. Then, instead of directly using $P(u, v)^{new}$ for the next M step, we refine the estimated rope state by minimizing the distance between $P(u, v)$ and $P(u, v)^{new}$, defined as

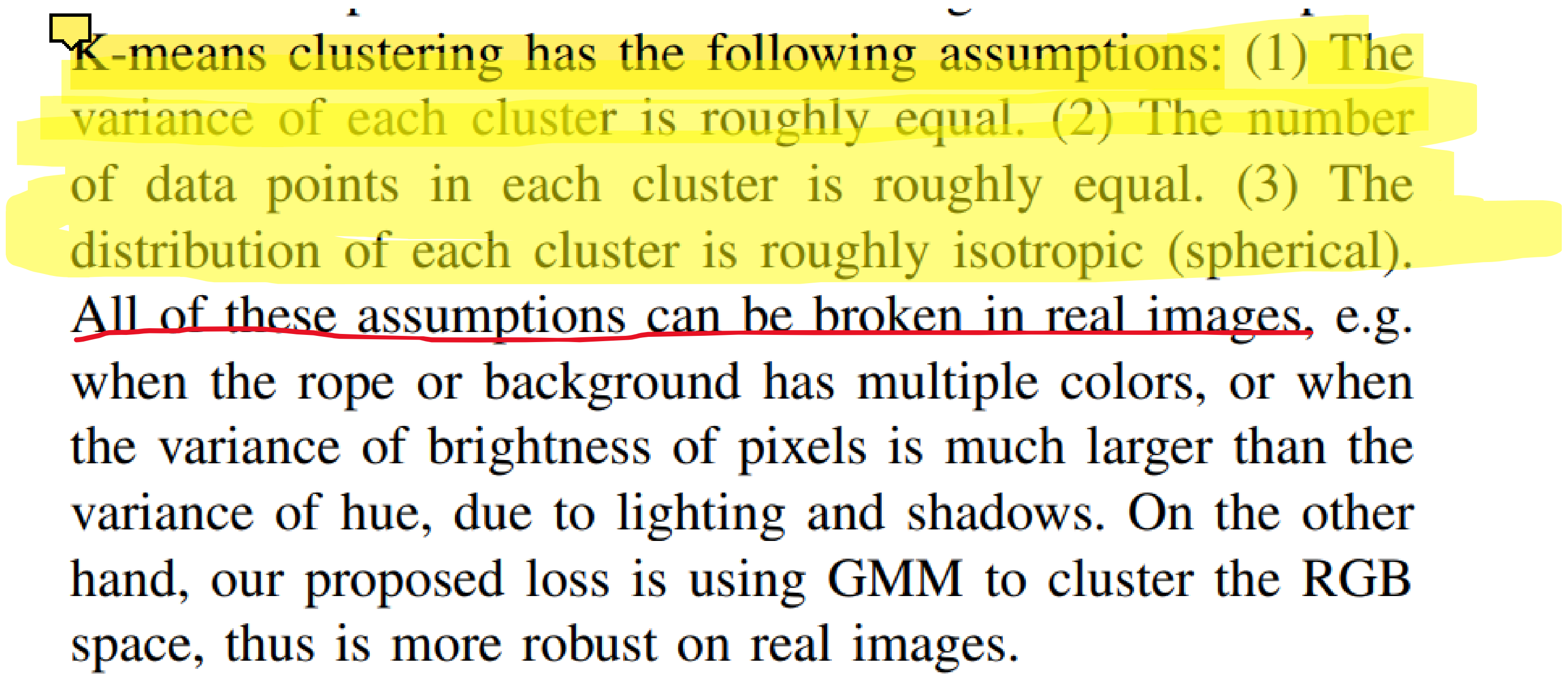
$$\sum_{(u,v)} -\log\left[P(u, v)P(u, v)^{new} + (1 - P(u, v))(1 - P(u, v)^{new})\right]. \qquad (4)$$

Perception Model: **self-supervising learning objective**





Why not use K-means?



K-means clustering has the following assumptions: (1) The variance of each cluster is roughly equal. (2) The number of data points in each cluster is roughly equal. (3) The distribution of each cluster is roughly isotropic (spherical). All of these assumptions can be broken in real images, e.g. when the rope or background has multiple colors, or when the variance of brightness of pixels is much larger than the variance of hue, due to lighting and shadows. On the other hand, our proposed loss is using GMM to cluster the RGB space, thus is more robust on real images.