Appendix

A Model Specification

A.1 Network Architecture

For Context Embedding, we use a ResNet-18 [1] network without the fully-connected layers as a backbone. Spatial Arg-Softmax is performed over the last convolution layer to generate a feature vector with length 512*2. Then it will be concatenated with the one-hot task descriptor. We use a 2-layers relu network to generate the final output with size 128. For LN, we put 4 deconvolution layers after the shared ResNet-18 backbone to recover the origin input size. We add shortcuts to the upsampling layers from previous layers with the same size. During training, LN and Context Embedding(as a part of CMPG) are trained at the same time in a multi-task training manner.

For the encoder of CMPG, an extra 2-layers relu network needs to handle the input trajectory representation, then we use a 3-layers relu network with 2 heads to generate mean and variance respectively(the size of mean and variance should be the same as the number of latent variables). The decoder is basically the same as the decoder, while the extra 2-layers network is for latent variables, and the size of output mean/variance are the same as trajectory representation instead of the number of latent variables. The prior network is also similar to the encoder, but without the extra 2-layers network for handing trajectory representation input.

For the NN baseline, we use a ResNet-18 network with Spatial Arg-SoftMax operations on the final feature map to handle the contextual information, then the features are concatenated with task descriptor to make a direct regression on trajectory representation $\mathbf{w}$.

A.2 Hyperparameters

For all experiments, we use a minibatch with size 200, an Adam [2] optimizer on learning rate of 0.01 without decay. For both simulation and real-world task, we use 128 latent variables. The number of RBF kernel is fixed at 10. The input size of the image is 224×224.

A.3 Implementation Details

We use PyTorch [3] to implement our model. The training and inference are finished on a machine with single NVIDIA TITAN X GPU and a quad-core Intel CPU. Our model is totally trained from scratch without any pre-trained parameters.

B Details of Probabilistic Trajectory Representation and Dynamic Control System

B.1 Kernel Selection

For smooth and continuous trajectory generation, we use Gaussian-kernel with the form:

$$\phi(\mathbf{t}) = \exp\left(-(\mathbf{t}-\mathbf{c})^T(\mathbf{t}-\mathbf{c})\right) \tag{1}$$

where c denotes the RBF kernel center phase location which in our experiments is evenly arranged between $[0, T_{end})$. This kernel satisfies our requirements that $\lim_{\mathbf{t}\to+\infty} f(\mathbf{t}) = 0$, and also satisfies:

$$\int_{-\infty}^{+\infty} \phi(\mathbf{t})\mathrm{d}\mathbf{t} = \pi^{-\frac{\dim(\phi)}{2}}, \lim_{\mathbf{a}\to+\infty}\int_{\mathbf{a}}^{+\infty} \phi(\mathbf{t})\mathrm{d}\mathbf{t} = 0 \tag{2}$$

B.2 Time-domain Response Characteristics of Second-order Systems

For a second-order dynamic system which can be formulated as:

$$\boldsymbol{\omega}_n^2(\mathbf{g}+\boldsymbol{\epsilon}) = \ddot{\mathbf{x}} + 2\boldsymbol{\xi}\boldsymbol{\omega}_n\dot{\mathbf{x}} + \boldsymbol{\omega}_n^2\mathbf{x} \tag{3}$$

where $\boldsymbol{\epsilon}$ denotes an zero-mean perturbation noise whose absolute value is significantly less than goal g, and x denotes the system output. the characteristic polynomial of Equation (3) would be $\boldsymbol{\lambda}^2 + 2\boldsymbol{\xi}\boldsymbol{\omega}_n\boldsymbol{\lambda} + \boldsymbol{\omega}_n^2 = 0$, when $\boldsymbol{\xi} \geq 1$ (over damping system), the characteristic polynomial would have two real roots; $\boldsymbol{\lambda}_{1,2} = -2\boldsymbol{\xi}\boldsymbol{\omega}_n \pm \sqrt{\boldsymbol{\xi}^2-1}$ so the zero-state response would be:

$$\mathbf{x}(\mathbf{t}) = (\mathbf{g}+\boldsymbol{\epsilon})\left(1 - \frac{\exp\left(-(\boldsymbol{\xi}-\sqrt{\boldsymbol{\xi}^2-1})\boldsymbol{\omega}_n\mathbf{t}\right)}{2\sqrt{\boldsymbol{\xi}^2-1}(\boldsymbol{\xi}-\sqrt{\boldsymbol{\xi}^2-1})} + \frac{\exp\left(-(\boldsymbol{\xi}+\sqrt{\boldsymbol{\xi}^2-1})\boldsymbol{\omega}_n\mathbf{t}\right)}{2\sqrt{\boldsymbol{\xi}^2-1}(\boldsymbol{\xi}+\sqrt{\boldsymbol{\xi}^2-1})}\right) \tag{4}$$

and thus we have $\forall\boldsymbol{\epsilon}\exists T_1 > 0$ s.t. $t > T_1 \Rightarrow |\mathbf{x}(\mathbf{t}) - \mathbf{g}| \leq 2|\boldsymbol{\epsilon}|$. Due to $\lim_{\mathbf{t}\to+\infty}\boldsymbol{\phi}(\mathbf{t}) = 0$, Let T_2 denotes the largest T that , thus we have $\forall\boldsymbol{\epsilon}\exists T = T_1 + T_2$ s.t. $t > T \Rightarrow |\mathbf{x}(\mathbf{t}) - \mathbf{g}| \leq 2|\boldsymbol{\epsilon}|$. So we can grantee the convergence by:

$$\lim_{\mathbf{t}\to+\infty}\boldsymbol{\tau}(\mathbf{t}) = \mathbf{g} \tag{5}$$

B.3 Dynamic System Parameters

As discussed in B.2, we build up the dynamic system with $\boldsymbol{\xi} = 1.5$ and $\boldsymbol{\omega}_n = 8$, which means the dynamic system parameter would be:

$$\mathbf{K} = \boldsymbol{\omega}_n^2 = 64 \tag{6}$$

$$\mathbf{B} = 2\boldsymbol{\omega}_n\boldsymbol{\xi} = 24 \tag{7}$$

C Full Experiment Result

Evaluation		apple	softball	banana	soup can	coffee can
NN (baseline)	Success	43.3%	73.3%	56.7%	10.0%	20.0%
	Failure (unfeasible)	13.3%	16.7%	26.7%	3.3%	10.0%
	Failure (wrong)	43.3%	10.0%	16.7%	86.7%	70.0%
CMP - SAS - LN	Success	20.0%	26.7%	23.3%	13.3%	13.3%
	Failure (unfeasible)	13.3%	10.0%	0.0%	3.3%	13.3%
	Failure (wrong)	66.7%	63.3%	76.7%	83.3%	73.3%
CMP - LN	Success	50.0%	80.0%	63.3%	50.0%	26.7%
	Failure (unfeasible)	0.0%	0.0%	0.0%	3.3$	3.3%
	Failure (wrong)	50.0%	20.0%	36.7%	46.7%	70.0%
CMP	Success	73.3%	86.7%	76.7%	80.0%	56.7%
	Failure (unfeasible)	13.3%	10.0%	16.7%	10.0%	6.7%
	Failure (wrong)	13.3%	3.3%	6.7%	10.0%	13.3%

Evaluation		mug	pear	screwdriver	box	bottle
NN (baseline)	Success	36.7%	30.0%	23.3%	56.7%	46.7%
	Failure (unfeasible)	13.3%	6.7%	16.7%	20.0%	13.3%
	Failure (wrong)	50.0%	63.3%	60.0%	23.3%	40.0%
CMP - SAS - LN	Success	10.0%	16.7%	43.3%	40.0%	36.7%
	Failure (unfeasible)	26.7%	26.7%	6.7%	13.3%	10.0%
	Failure (wrong)	63.3%	56.7%	50.0%	46.7%	53.3%
CMP - LN	Success	76.7%	36.7%	46.7%	86.7%	50.0%
	Failure (unfeasible)	0.0%	6.7%	3.3%	0.0%	6.7%
	Failure (wrong)	23.3%	56.7%	50.0%	13.3%	43.3%
CMP	Success	70.0%	66.7%	80.0%	83.3%	83.3%
	Failure (unfeasible)	26.7%	13.3%	10.0%	10.0%	13.3%
	Failure (wrong)	3.3%	20.0%	10.0%	6.7%	3.3%

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[3] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.