

A METHOD DETAILS

A.1 LATENT DIFFUSION MODELS

Diffusion Models. Diffusion models (Song et al., 2020; Ho et al., 2020) typically contain a forward noising process and a reverse denoising process. During the forward process, we gradually apply noise to real data x_0 : $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{a}_t}x_0, (1 - \bar{a}_t)\mathbf{I})$ over T timesteps, where constants \bar{a}_t are hyperparameters. By applying the reparameterization trick, we can sample $x_t = \sqrt{\bar{a}_t}x_0 + \sqrt{1 - \bar{a}_t}\epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$. During the reverse process, it starts from Gaussian noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and gradually removes noises to recover x_0 : $p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t), \Sigma_\theta(x_t))$. With reparameterizing μ_θ as a noise prediction network ϵ_θ , the model can be trained with $\mathcal{L}_{simple}(\theta) = \|\epsilon_\theta(x_t) - \epsilon_t\|_2^2$. We also learn the covariance Σ_θ following Nichol & Dhariwal (2021); Peebles & Xie (2023) with the full KL loss.

Latent Diffusion and Tokenization. Latent diffusion models (Rombach et al., 2022; Ma et al., 2024) perform diffusion process in a low-dimensional latent space z^{ld} rather than the original pixel space. We leverage the pre-trained VAE in SDXL (Podell et al., 2023) to compress each frame o_t to latent representations: $z_t^{ld} = Enc(o_t)$, and after the denoising process, the latent representation can be decoded back to the pixel space with the VAE decoder: $o_t = Dec(z_t^{ld})$. For each z^{ld} , it is divided into image patches and tokenized by convolutional networks to P tokens with D dimensions (hidden size). Sequencing the image tokens of all T frames, we get the video token in the shape of $T \times P \times D$.

Spatial-Temporal Attention Mechanism. We leverage transformers (Vaswani, 2017) to implement the dynamics module, and use the memory-efficient spatial-temporal attention mechanism (Ma et al., 2024; Bruce et al., 2024; Zhu et al., 2024), where each attention block consists of a spatial attention block and a temporal attention block. The spatial attention operates on the $1 \times P$ tokens within each frame, and the temporal attention operates on the $T \times 1$ tokens across T timesteps at the same location.

A.2 LANGUAGE-VISION REPRESENTATION

The original fine-tuning loss $\mathcal{L}_{LIV} = \mathcal{L}_I(\psi_I) + \mathcal{L}_L(\psi_I, \psi_L)$ is calculated on sampled sub-trajectory batch data $\{o_s^i, \dots, o_k^i, o_{k+1}^i, \dots, o_T^i, g^i\}_{i=1}^B$ from each task \mathcal{T}_i , where $s \in [0, T_i - 1]$, $s \leq k < T_i$. They have the following forms:

$$\begin{aligned} \mathcal{L}_I(\psi_I) &= \frac{1-\gamma}{B} \sum_{i=1}^B [-\mathcal{S}(\psi_I(o_s^i), \psi_I(o_T^i))] + \log \frac{1}{B} \sum_{i=1}^B \exp[\mathcal{S}(\psi_I(o_k^i), \psi_I(o_T^i)) + 1 - \gamma \mathcal{S}(\psi_I(o_{k+1}^i), \psi_I(o_T^i))], \\ \mathcal{L}_L(\psi_I, \psi_L) &= \frac{1-\gamma}{B} \sum_{i=1}^B \left[-\log \frac{e^{(1-\gamma)\mathcal{S}(\psi_I(o_T^i), \psi_L(g^i))}}{\frac{1}{B} \sum_{j=1}^B [e^{(1-\gamma)\mathcal{S}(\psi_I(o_T^i), \psi_L(g^j))}]} \right], \end{aligned} \quad (2)$$

A.3 LOW-LEVEL POLICY

In this work, we use a low-level policy with a similar structure to [diffusion policy](#) (Chi et al., 2023). We show the architecture of this policy in Figure 11. We employ a spatial-temporal attention mechanism. Specifically, the input contains the agent view observation history $o_{t-12:t}^a \in \mathbb{R}^{12 \times 3 \times 128 \times 128}$ and the eye in hand observation history $o_{t-12:t}^e \in \mathbb{R}^{12 \times 3 \times 128 \times 128}$ at timestep t , the proprioception state history $s_{t-12:t} \in \mathbb{R}^{12 \times 123}$, the language tokens extracted from Meta Llama 3.1 8B (Dubey et al., 2024) $g \in \mathbb{R}^{T_g \times 4096}$, the predicted flow for both the agent view $\mathbf{p}_{t:t+16}^a \in \mathbb{R}^{16 \times 529 \times 2}$ and eye in hand view $\mathbf{p}_{t:t+16}^e \in \mathbb{R}^{16 \times 529 \times 2}$, and the predicted future videos for both the agent view $\hat{o}_{t:t+16}^a \in \mathbb{R}^{16 \times 3 \times 128 \times 128}$ and eye in hand view $\hat{o}_{t:t+16}^e \in \mathbb{R}^{16 \times 3 \times 128 \times 128}$. The low-level policies are a denoising model that will predict the gradient field of the action chunking $\nabla E(A_t)$, and after 500 denoising steps, we can get the future action sequences $a \in \mathbb{R}^{16 \times 7}$ where 7 is the action size. We use the action in a receding horizon manner where we only execute 8 steps and we replan the future flow and videos and predict another 16 action steps iteratively.

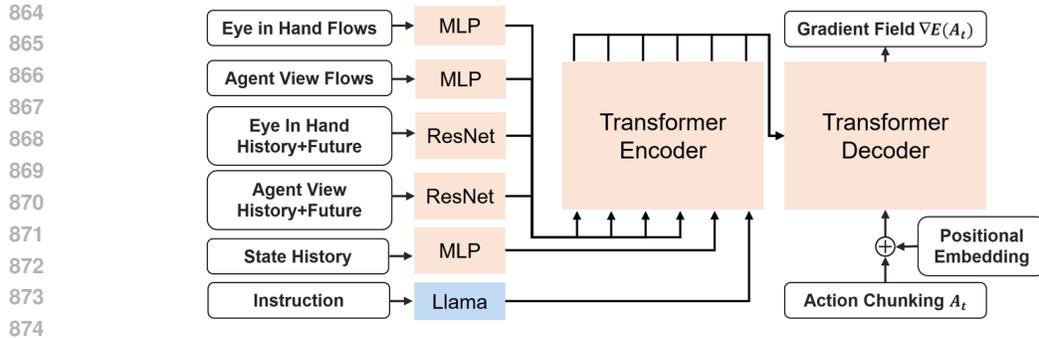


Figure 11: Low-level policy architecture. This is Ours-FV. Ours-F and Ours-V are in the same architecture without corresponding input conditions.

B EXPERIMENT DETAILS

B.1 TRAINING DETAILS

We report the hyperparameters of the models we trained in Table 5 and Table 6. We train all data with observation history equals to 16 and future flow horizon equals to 16.

	CVAE-S	CVAE-B	CVAE-L
Encoder Layer	4	6	8
Decoder Layer	6	8	12
Hidden Size	384	768	1024
Learning Rate	1e-4	5e-5	1e-5
Image Patch Size	8	8	8
Head Number	4	8	12

Table 5: Hyperparameters of CVAE.

	D-S	D-B	D-L
Layers	8	12	16
Hidden Size	384	768	1024
Learning Rate	1e-4	1e-4	1e-4
Head Number	6	12	16

Table 6: Hyperparameters of the dynamics module.

B.2 MORE POLICIES RESULTS

Besides the low-level experiments in Section 5.3, we also perform experiments in the LIBERO-LONG task suite with two other models:

1. **OpenVLA** Kim et al. (2024): this is a large-scale retained that is designed for general-purpose vision-language-action prediction. We use the pretrained checkpoint from the official paper and test with both zero-shot and fine-tuned models. We use 50 demonstrations for each task in LIBERO-LONG to fine-tune the OpenVLA model. For consistency, we use a resolution of 128×128 for fine-tuning. For fair comparisons, we only use the agent view images as input because the original OpenVLA is only trained with third-view images.
2. **Pretrained FLIP on LIBERO-90 and fine-tuned on LIBERO-10 as the high-level planner.** In this setting, we train FLIP on 50×90 action-less demonstrations with a resolution of 64×64 from LIBERO-90, and finetune it with 50×10 from LIBERO-LONG, and use this FLIP model as the planner to train the same low-level policy as in the main paper. Here we only test the flow-conditioned policy version, and call it Ours-90.

918
 919
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 950
 951
 952
 953
 954
 955
 956
 957
 958
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 970
 971

Results. We show the results together with the main paper results in Figure 12. We can see that OpenVLA cannot handle the long-horizon tasks of LIBERO-LONG either with zero-shot or fine-tuned models, showing there is still a long way to go for general-purpose vision-language-action models. Ours-90 performs similarly to Ours-F and Ours-FV, showing that pretraining in other tasks may not bring significant improvement for low-level policy learning. This comforts with the life-long learning results in the original LIBERO paper (Liu et al., 2024a), where they also show that pretraining cannot help (sometimes even hurt) the policy training results.

It is worth noting that, in the original OpenVLA paper, they also fine-tuned the pretrained model on LIBERO-LONG tasks and archived a $53.7 \pm 1.3\%$ success rate. We think the success of their results comes from two aspects, which cannot be true in our setting: 1) we are using a resolution of 128×128 , which may not be large enough to represent the details in the scene. In comparison, OpenVLA uses a resolution of 256×256 . 2) We are using the official demonstrations provided by the LIBERO paper, which may not be as good as the re-collected demonstrations in their demonstrations.

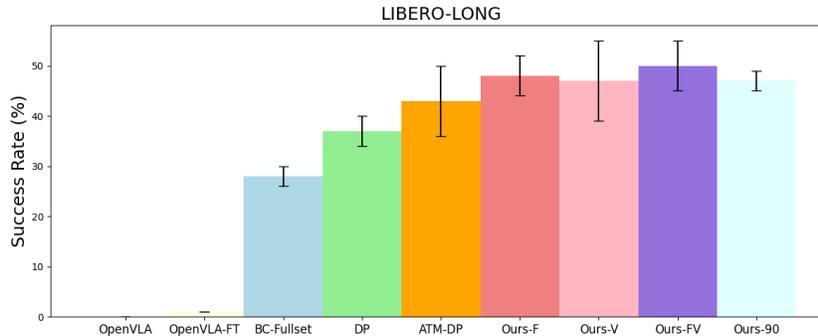


Figure 12: Low-level policy results in LIBERO-LONG.

B.3 ABLATION STUDIES OF ACTION MODULE

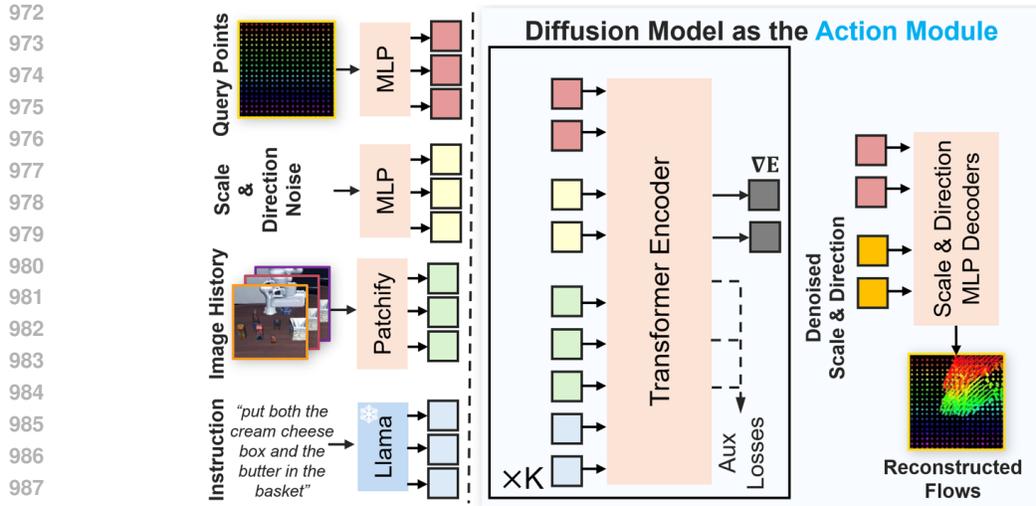
To verify if diffusion models can generate better results for image flow prediction (the action module of FLIP) than a CVAE model, here we design a diffusion model for the flow generation task. We use the same architecture as the CVAE decoder for the diffusion model (as illustrated in Figure 2, where a transformer encoder takes as input the query points tokens, the image history tokens, the language embedding tokens, and noisy inputs, and outputs the scale and direction gradient fields for each query point. The network structure is shown in Figure 13. We perform comparison experiments in both LIBERO-LONG and Bridge-V2 data.

Results. Table 7 shows the results. We can see that CVAE performs better on LIBERO-LONG tasks and the diffusion model performs better on Bridge-V2 tasks, but their performance difference is minor. We can see the main improvement of the action module comes from: 1) predicting the delta action (scale and direction) rather than predicting the absolute coordinates of the image flows; 2) auxiliary losses as stated in our main paper.

Although using the diffusion model may perform better in some tasks, the inference speed is slower than CVAE models. Given that their performance is not too different, we still use CVAE as our default action module.

B.4 REAL WORLD EXPERIMENTS

In order to show the application of FLIP on real-world tasks, here we design two long-horizon real-world manipulation tasks for testing. We use a 6-DOF X-arm as the robot arm, and use two



989 Figure 13: Using diffusion model for the action module. The model also generates the scale and
990 directions for each query point and the full flow trajectory can be reconstructed by them.

991
992
993
994
995
996
997
998
999
1000

	LIBERO-LONG		Bridge-V2	
	ADE ↓	LTDR ↑	ADE ↓	LTDR ↑
ATM (Wen et al., 2023)	19.6	53.8%	18.4	66.1%
Ours-ABS	20.5	57.3%	17.9	59.3%
Ours-NoAUX	14.5	73.2%	12.7	75.6%
Ours(CVAE)	12.7	76.5%	11.9	80.2%
Diffusion	13.4	75.9%	10.2	82.7%

1001 Table 7: Comparison results of CVAE and Diffusion models as the action module for FLIP.

1002
1003
1004
1005
1006
1007
1008
1009
1010

RealSense D435i cameras to get the visual inputs. We put one camera on the other robot arm as the third-view camera and another camera on the wrist of the robot arm as the eye-in-hand camera. The control frequency of both tasks is set to 10Hz. We also train a diffusion policy and an ATM (Wen et al., 2023) policy as the baseline for each task. We perform experiments in a table-top setting, as shown in Figure 14. We test both tasks with 20 random initialized episodes. The two tasks are:

1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

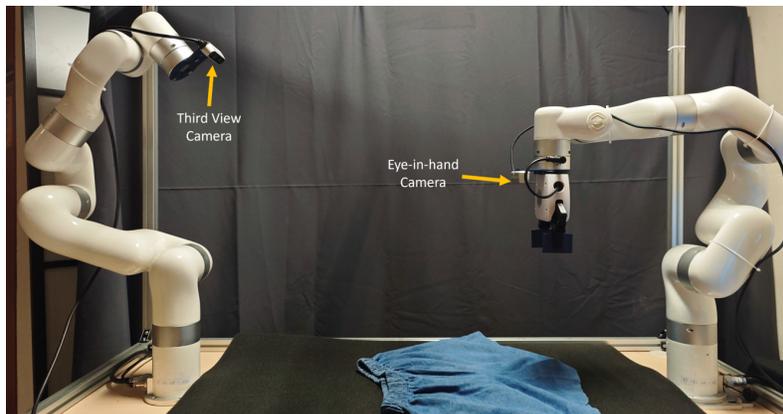


Figure 14: Our real-world experiment setting. We use the right X-arm as the manipulator, and use two RealSense D435i cameras as visual inputs.

1026
 1027
 1028
 1029
 1030
 1031
 1032
 1033
 1034
 1035
 1036
 1037
 1038
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1050
 1051
 1052
 1053
 1054
 1055
 1056
 1057
 1058
 1059
 1060
 1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079

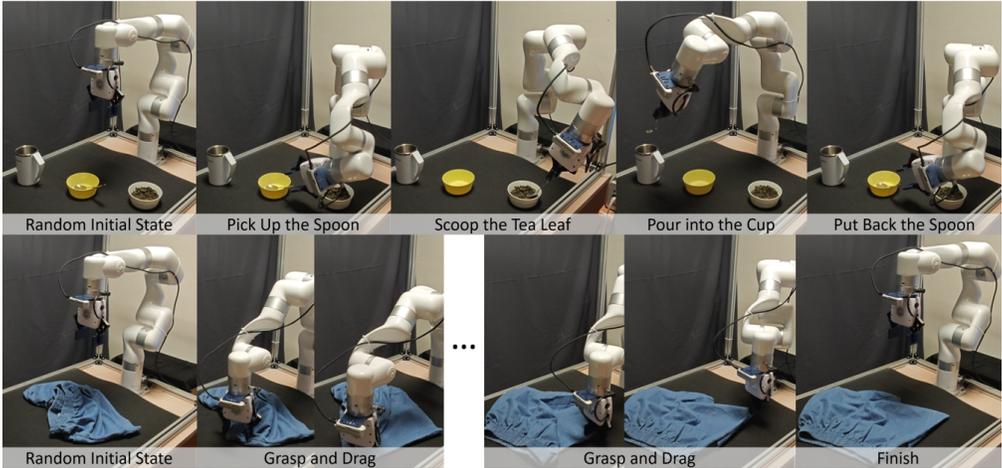


Figure 15: Two tasks in our real-world experiments.

1. **Tea Scooping Task.** In this task, there is a white bowl of dry tea leaf (Tieguanyin), a yellow bowl with an iron spoon, and an empty cup on the table, as shown in Figure 15. The robot is trained to first pick up the spoon from the yellow bowl, then use the spoon to scoop the tea leaf from the white bowl pour the tea leaf into the cup, and finally put back the spoon into the yellow bowl. The positions of both the bowls and the cup are randomized in the space that the robot arm can reach. The task is successful if all three stages are accomplished. The action space of the robot is 7-dim, including the 6-DOF pose and the gripper action. We collect 40 demonstrations with a scripted policy to train FLIP and a flow-guided low-level policy. We choose this task because it is both long-horizon and requires precise grasping and manipulation. It also requires tool use and interaction between rigid body and granular objects.
2. **Cloth Unfolding Task.** In this task, a pair of denim shorts was randomly thrown on the table, and the robot is trained to unfold it to make it flat, as shown in Figure 15. The task is successful if the projection area of the jeans on the table is greater than 85% of its maximum area. The action space of the robot is 6-dim, which consists of a 2-dim end-effector coordinate in the x-y plane, a 1-dim grasping orientation angle, a 2-dim dragging coordinate in the x-y plane, and the gripper aperture. We fix the grasping height to 22 mm for this task. We collect 50 demonstrations for this task with a scripted policy. We choose this task because it is both long-horizon and difficult because of the deformation of the jeans, which requires a long-horizon task planning ability for the policy. The robot has to finish the task in 15 grasps and drags.

Results. Table 8 shows the results of both tasks. We can see that our policy is significantly better than the baselines. These two tasks are very difficult in our setting since we only use no more than 50 demonstrations for each of them, while modern image-based imitation learning algorithms (such as Diffusion Policy (Chi et al., 2023)) usually require more than 500 demonstrations for such kinds of long-horizon tasks. In contrast, our method benefits from the high-level model-based planning advantages and can correct back to the good trajectory when it deviates from it, thus the success rates are improved.

	Diffusion Policy (Chi et al., 2023)	ATM (Wen et al., 2023)	Ours
Tea Scooping	15.0%	25%	55%
Cloth Unfolding	0.0%	0.0%	50.0%

Table 8: Success rates of real robot experiments over 20 evaluations.

B.5 DATA SCALABILITY

To show the performance change of FLIP along with different amounts of training data, here we perform an experiment on LIBERO-LONG to show the planning success rates change with different amount of demonstrations. Results are shown in Table 9. From the results, we can see that with more data for each task, the planning success rates become better.

LIBERO-LONG (Liu et al., 2024a)	
FLIP-10	0%
FLIP-20	5%
FLIP-30	40%
FLIP-40	90%
FLIP-50	100%

Table 9: Planning results change along with data amounts for LIBERO-LONG. The number shown in the left column is the demonstration number for each task during training.

B.6 GENERATIVE PLANNING WITH VISUAL DISTRACTION

In this section, we perform a qualitative experiment to see how will our FLIP be affected in the presence of noise or visual obstructions. To this end, we manually add an image of an apple in the initial image of the LIBERO-LONG tasks, as shown in Figure 16 and Figure 17, and see how FLIP performs generative planning in this setting.

Results. We add the visual distraction image in different areas of the initial image, and we can see that, the video generation model will first fail after several planning steps and generative distorted cups and make the whole image grey. However, before the model fails, the flow generation model still performs very well, which shows that the flow generation model can resist visual distractions, while the video generation model is susceptible to visual distractions.

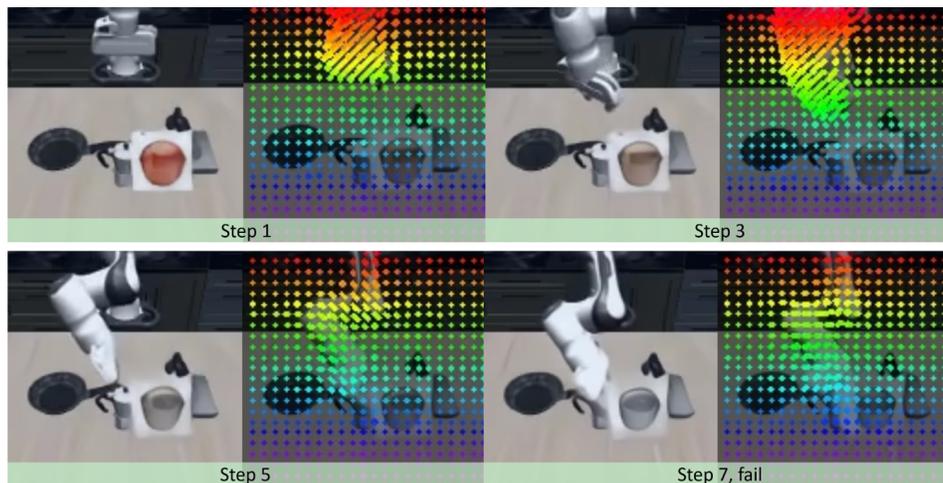


Figure 16: The apple is put on the target cup, and the generative planning fails in 7 steps (the cup is distorted, by which we call the planning fails). Note, the flow model performs well before the image is distorted and turns to grey, which shows that the flow model can capture the geometry of the objects and understand the physical movement requirements of the task.

1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187

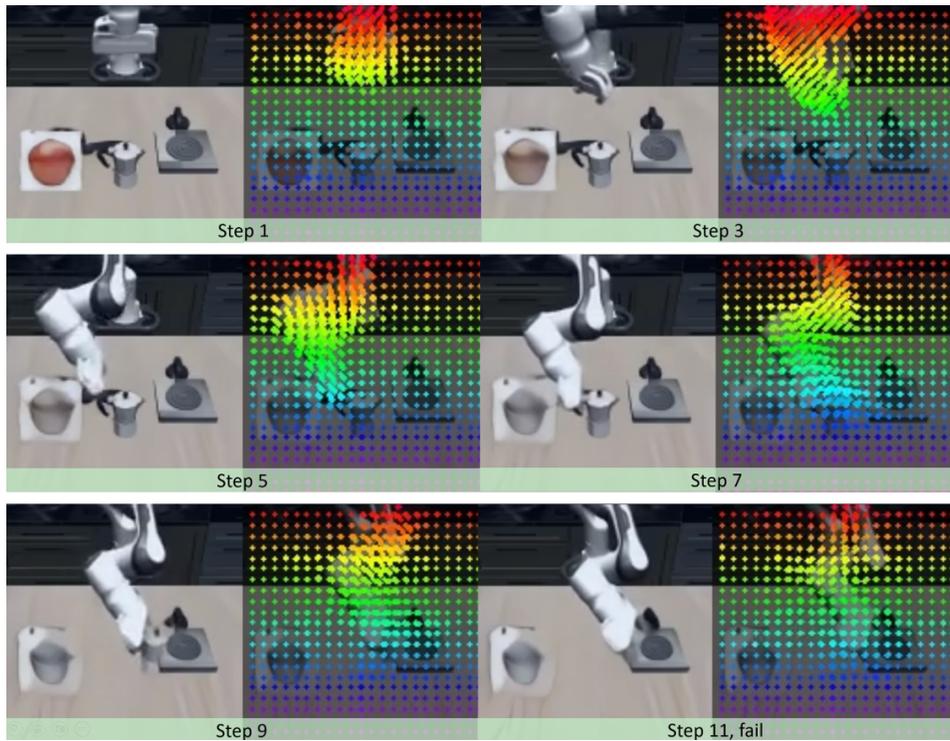


Figure 17: The apple is put away from the target object, and the generative planning can exist more steps.