Instruction-conditioned Visual Prediction for Robotic Actions

Han Gao Yiyang Dong Xixi Chen Yumeng Lu Juntao Liu
School of Data Science, The Chinese University of Hong Kong, Shenzhen
{121090129, 121090106, 121090057, 121090388, 121090339}@link.cuhk.edu.cn

Abstract

Predicting the visual outcome of a robot's action is essential for planning and understanding behavior in interactive environments. In this project, we explore an image-conditioned frame prediction task, where the model is given an observed image and a textual instruction (e.g., "beat the block with the hammer") and asked to generate the image 50 frames into the future. We build our dataset using the RoboTwin simulation platform and focus on three tasks: block_hammer_beat, block_handover, and blocks_stack_easy. Each episode provides RGB-D observations aligned with action descriptions. We plan to fine-tune an InstructPix2Pix-style model and evaluate the quality of generated frames using SSIM and PSNR metrics. Our approach aims to bridge the gap between robotic simulation data and text-conditioned visual understanding.

1. Introduction

Understanding and predicting the visual consequences of robot actions is a core challenge in robotics and computer vision. In real-world applications, being able to forecast how a scene changes in response to a robotic manipulation is crucial for safe planning, autonomous operation, and interactive behavior modeling. With the emergence of large-scale generative models and simulation platforms, it becomes feasible to train models that predict future visual states conditioned on both current observations and high-level commands.

In this project, we formulate an instruction-conditioned image prediction task in a simulated robotic environment. Specifically, given a current RGB image and a textual instruction (e.g., "beat the block with the hammer"), the model is required to predict what the robot will see 50 frames into the future. Our main objective is to fine-tune a pretrained InstructPix2Pix model to accept a current frame and a natural language command as inputs and generate the corresponding future frame. To support this, we construct a dataset using RoboTwin, a dual-arm robot simulation platform that provides RGB-D observations and expert demonstrations across three manipulation tasks: `block_hammer_beat`, `block_handover`, and `blocks_stack_easy`. We plan to evaluate model performance using perceptual metrics such as SSIM and PSNR.

2. Related Works

Our task lies at the intersection of instruction-conditioned image generation, robotic behavior simulation, sequential visual modeling, and perceptual image quality assessment. In this section, we review key studies that inspired our approach.

Instruction-Conditioned Image Generation. The core of our task is to predict a future image conditioned on both a current observation and a textual instruction. InstructPix2Pix [1] introduces a diffusion-based framework for editing images using natural language prompts, achieving high-quality results on diverse editing tasks. Similarly, Pix2Pix [2] proposed a conditional GAN model for image-to-image translation, such as converting edge maps to photorealistic images. More recently, RDT-1B [3] demonstrated the effectiveness of large diffusion models in bimanual robotic manipulation, indicating the growing potential of foundation models in generating action-conditioned visual outcomes.

Robotic Simulation and Benchmarks. To generate realistic robotic scenes for training and evaluation, we utilize RoboTwin [4], a dual-arm robot simulation platform designed around generative digital twins. It supports diverse manipulation tasks, multi-view RGB-D rendering, and kinematic replay. GR-MG [5] further contributes to this domain by proposing a multi-modal goal-conditioned policy that leverages partially annotated data, which is conceptually related to our instruction-based scenario setup.

Sequential Visual Prediction. While our current implementation is non-recurrent, future-frame prediction is inherently a temporal problem. Long Short-Term Memory

(LSTM) networks [6] have historically shown strong performance in modeling sequential data, including robotic motion and video prediction tasks. Such methods offer insights into how temporal dependencies can be captured more effectively in future iterations of our framework.

Perceptual Image Quality Evaluation. To quantitatively evaluate the fidelity of our predicted images, we adopt Structural Similarity Index (SSIM) and Peak Signal-to-Noise Ratio (PSNR). SSIM [7] models human visual perception by considering structural, luminance, and contrast similarities. Hore and Ziou [8] provide a comparative study, highlighting SSIM's superiority over PSNR in many distortion scenarios.

These studies jointly inform our methodological choices, from data generation and model architecture to evaluation strategies, helping us design a coherent pipeline for instruction-driven robotic image prediction.

3. Methods

In this section, we detail our approach for predicting future frames of robotic manipulation tasks by fine-tuning a pretrained InstructPix2Pix model using data generated from the RoboTwin simulation environment.

3.1. Problem Description

We formulate the problem as predicting the image I_{t+50} given the current frame I_t and a textual instruction T. Here, T denotes one of three high-level commands: "beat the block with the hammer", "handover the blocks", or "stack blocks". The model was required to produce an RGB image of resolution at least 128×128 representing the robot's observed view at time step $t + 50$.

3.2. Data Generation with RoboTwin

To simulate robotic manipulation tasks and obtain a training dataset, we used RoboTwin , an open-source virtual simulation platform built on the `SAPIEN` physics engine. RoboTwin enables high-fidelity robotic simulations with RGB-D rendering, point clouds, and realistic physics-based interactions.

Data Overview We focus on generating image sequences for three robotic manipulation tasks: `block_hammer_beat`, where a robotic arm strikes a block with a hammer; `block_handover`, involving the transfer of a block to a designated target position; and `blocks_easy_stack`, which entails stacking two blocks in a simple configuration. The simulation generates data matrices that can be converted into RGB images, capturing visual observations from four different camera viewpoints, along with the corresponding robot state information over time. Each episode corresponds to one complete action sequence and contains over 50 frames.

Data Generation The simulation ran on a Tesla T4 GPU provided by Google Colab. We first cloned the RoboTwin repository and installed all required dependencies, including `sapien`, `torch`, `open3d`, `trimesh`, among others, following the official setup instructions. Then, we modified the default configuration files (`.yaml`) provided by *RoboTwin* for all three tasks to generate image sequences tailored to our needs. Specifically, we set `episode_num = 100` for each task to generate 100 trajectories per task. To ensure that RGB data convertible to images is retained, we set `rgb = true`, while disabling unnecessary modalities by setting `depth` and `pointcloud` to `false`. The `save_freq` parameter controls the frequency at which frames are saved from the simulation. Considering the varying durations required to complete different tasks, we chose task-specific values: for `block_hammer_beat`, we set `save_freq = 60`; for `block_handover`, `save_freq = 100`; and for `blocks_easy_stack`, `save_freq = 90`. To simplify the training, we only considered the image data captured from the `head_camera`. This choice was based on the limitations of other camera views: the `left_camera` often fails to capture the object entirely, the `right_camera` provides only partial visibility, and the `front_camera` also suffers from limited coverage. While the `head_camera` may occasionally have its view obstructed by the robot arm, it still offers the most consistent and informative perspective among all available options. Lastly, for each pickled data file, we used the RGB data matrices to generate the corresponding photo frame in `.png` format. The full dataset is hosted at: `https://drive.google.com/drive/folders/1m7kTra226lENiAgNGKFBa8GsdI0ex3-l?usp=sharing`

3.3. Fine Tune

3.3.1. Data Simulation

In terms of data augmentation, we refer to the data augmentation in GR-MG projects, and sets up different pretreatment processes for static background images and hand images, and different data augmentation parameters are used for different data augmentation parameters. Such processing can effectively enhance the generalization ability and improve the robustness of the model. In addition, the model is also used: random translation, horizontal and vertical flip, affine transformation, color jitter, gaussian blur, sharp adjustment, automatic contrast adjustment.

3.3.2. Model

Visual Feature Extraction The visual feature extraction is responsible for extracting the visual information from the input image frame, and we choose to use the pre-trained ResNet18 model as the visual feature extractor, because ResNet18 has a powerful image extraction ability. In use, we removed the last complete connecting layer of ResNet18, which was able to output a 512 dimensional eigenvector, which contains visual information such as the shape, color, texture and other visual information of the object in the image.

Temporal Modeling When dealing with the video frame sequence, there is a time of dependency between adjacent frames. LSTM can effectively capture this long-term dependency relationship and thus improve the dynamic changes of the sequence of geographical solutions. In the code we define the `videolstm` class to implement the LSTM. In the forward method of the roboticactionpredictor class, after the LSTM processes the sequence of characteristics, the recombination of the attention mechanism can be more effectively used to generate the image using the sequence information.

Text Encoding In this project, we define a dictionary `TASK_INSTRUCTIONS` to implement text encoding. This dictionary maps different task types to the corresponding text instructions.

Feature Fusion Feature fusion combines the sequence characteristics of `videolstm` and the visual characteristics of the last frame with the relevant information of the task. In order to enhance the focus on important information in the sequence, we introduced a single-self-attention mechanism, implemented by the `timedistributedatmodule`. Refer to the similar robot predictive experiment (RDT-1B: a Diffusion Foundation Model for Bimanual Manipulation) self-attention mechanism.

The `timedistributedatmodule` uses the output of `videolstm` as input. The class includes: calculating the attention score, generating the attention weight for the characteristics of the attention processing.

The concatenated features go through a fully connected layer with a ReLU activation function and a dropout layer. The ReLU activation function adds nonlinear features to make the model more expressive. The dropout layer randomly drops some neurons with a certain probability to stop the model from overfitting.

In the end, this creates a combined eigenvector, which combines sequence and visual information, providing more comprehensive information for subsequent image generation.

Generator/Decoder We use Stable Diffusion Instruct-Pix2Pix as the generator/decoder. Stable Diffusion Instruct-Pix2Pix is a powerful diffusion-based model that can generate high-quality images based on the input text instructions and initial images.

In this model, the input text instructions come from the task type and are obtained through the `TASK_INSTRUCTIONS` dictionary conversion; The initial image is the last frame of the input sequence. The model takes the fused feature vectors and text instructions as input, and then uses the Stable Diffusion InstructPix2Pix pipeline to generate the predicted future frames. By setting this parameter reasonably, we can generate a predicted image that not only meets the task requirements but also has a certain consistency with the input image.

GR-MG Regularizer The GRMG regularizer constrains the training process of the model by calculating the gradient loss between the predicted frame and the target frame at different scales. It will downsample the predicted frame and the target frame, calculate the gradients, and then calculate the mean square error loss between the gradients.

3.3.3. Training

The loss function of this study consists of two parts: SSIM loss and similarity penalty. SSIM loss measures the structural similarity between the predicted frame and the target frame. Considering the image brightness, contrast and structural information comprehensively, it can accurately evaluate the image quality. Similarity penalty is the mean square error between the predicted frame and the current frame, preventing the predicted frame from being overly similar to the current frame and prompting the model to learn future changes. The total loss is the weighted sum of the two, expressed as

$$L_{total} = L_{SSIM} + 0.2 \times L_{MSE}.$$

The weight of 0.2 balances the guidance of the two losses on the model training and helps the model predict future frames. During the training process, we adopt the Exponential Moving Average (EMA) technique to update the model parameters. EMA maintains a set of shadow parameters, which are obtained by exponentially weighted averaging the model parameters. In each training step, the shadow parameters are updated according to the following formula:

$$\theta_{EMA} = \alpha\theta_{EMA} + (1 - \alpha)\theta$$

where θ_{EMA} is the shadow parameter, θ is the current model parameter, and α is the decay coefficient, which is usually set to 0.999.

The first frame at time t and the frame at $t + 50$ were paired to form the input-output training samples, accompanied by a task-specific instruction.

3.4. Evaluation Metrics

To evaluate the quality of the predicted future frames, we adopted two standard metrics from the image generation literature: SSIM (Structural Similarity Index) and PSNR (Peak Signal-to-Noise Ratio). SSIM captures perceptual similarity by comparing structural features such as edges and textures, which is critical for ensuring visual consistency in robotic environments simulated by *RoboTwin*. PSNR measures pixel-level fidelity and is sensitive to reconstruction errors, making it a suitable choice for assessing the accuracy of *InstructPix2Pix*'s frame prediction. Both metrics were computed between the predicted frame and the ground-truth RGB frame at time step $t + 50$, allowing us to quantify how well the model anticipates future observations conditioned on the given action instruction.

4. Experiments

We evaluated our method on a dataset generated using the RoboTwin simulator, covering three robotic manipulation tasks: `block_hammer_beat`, `block_handover`, and `blocks_stack_easy`. For each task, we collected 100 episodes with RGB frames recorded from the head camera. The first frame at time t and the frame at $t + 50$ were paired to form the input-output training samples, accompanied by a task-specific instruction. We split the dataset into 80% training and 20% testing. Fine-tuning was performed on the InstructPix2Pix model using the settings described in the Method section. Finally, we evaluated the similarity between the predicted future frame generated by the fine-tuned InstructPix2Pix model and the ground-truth frame at the time step $t + 50$, using SSIM and PSNR as evaluation metrics.

In detail, the final model was conducted on the Google Colab platform using an NVIDIA A100 Tensor Core GPU. The dataset is divided into the training set and the validation set in an 8:2 ratio.

Static background image enhancement: Use RandomShiftsSingleAug ($pad = 10$) for a wide range of random offsets to simulate the background changes when the robot performs tasks at different positions.

Hand image enhancement: A smaller RandomShiftsSingleAug ($pad = 4$) offset is adopted to avoid the loss of hand information caused by excessive offset. Meanwhile, we set a smaller rotation Angle (5 degrees) and a narrower Gaussian blur sigma range (0.1 to 1.0) to retain the key features of the hand.

We use the Stable Diffusion InstructPix2Pix model from Hugging Face (timbrooks/instruct-pix2pix). The model is loaded via the diffusers library, where `num_inference_steps` is set to 20 and `guidance_scale` is set to 3.0 during inference. The structures and settings of the relevant UNet, VAE, and other components are all derived from the model's configuration files on the Hugging Face Hub.

During the training process, the batch size is set to 32, and the number of training rounds is 5. The optimizer adopts $AdamW$. The learning rate of the LSTM layer is 10^{-4}, and the learning rate of the feature fusion layer is 10^{-5} (0.1 times that of the LSTM layer). The loss function is the weighted sum of the SSIM loss and the similarity penalty, with the weight of the similarity penalty being 0.2. During the training process, EMA technology was also introduced, and the attenuation coefficient α was set to 0.999.

Furthermore, for the stack blocks experiment, due to the large amount of data, the training speed is too slow. To accelerate the training speed, only 60% of the training set samples are randomly sampled during training.

Results Training loss curves are shown in Figure1 - Figure3
Some sample outputs are shown in Figure4 - Figure6
The Table1 shows the SSIM and PSNR for the sample outputs.

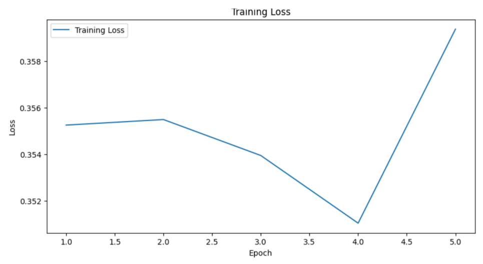

Figure 1. Training loss for block_hammer_beat

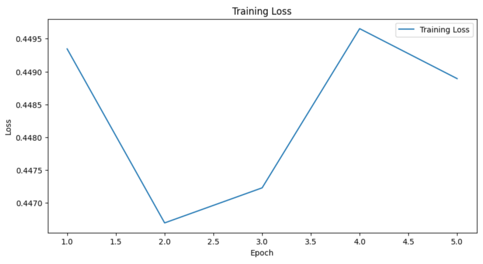

Figure 2. Training loss for block_handover

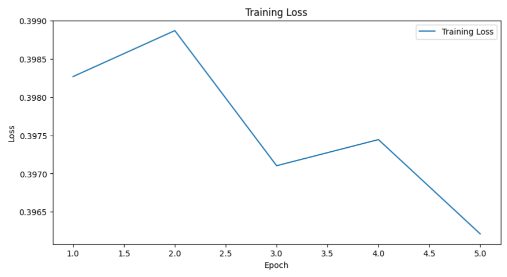

Figure 3. Training loss for blocks_stack_easy

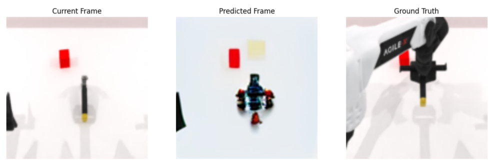

Figure 4. Sample output for block_hammer_beat

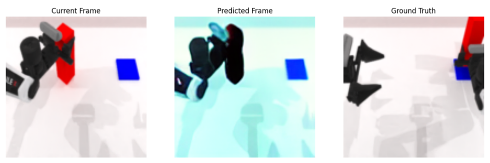

Figure 5. Sample output for block_handover

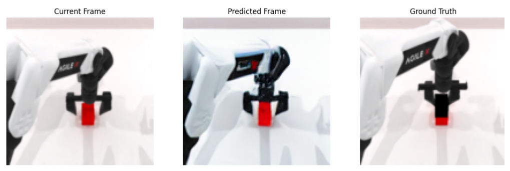

Figure 6. Sample output for blocks_stack_easy

Task	SSIM	PSNR
`block_hammer_beat`	0.8186	16.5382
`block_handover`	0.8010	14.1699
`blocks_stack_easy`	0.7703	16.8474

Table 1. SSIM and PSNR for sample outputs

Whether in terms of the quality of the pictures or the accuracy of the predictions, it has performed well. For simple actions, the model can generate predictions closely related to the actual situation. Based on the evaluation representation of indicators such as SSIM and PSNR, the model also ensures the high fidelity of the generated images.
For more complex actions, such as stacking blocks, more complex training is required, involving complex spatio-temporal interactions and fine-grained operation details. The model needs more training and verification, and improved training can be carried out by expanding the data set, reducing the batch size, etc.

5. Conclusion and Future Work

In this work, we explored future-frame prediction in robotic manipulation tasks by fine-tuning a pretrained Instruct-Pix2Pix model on a custom dataset generated with the RoboTwin simulation environment. Our model effectively leveraged both visual observations and textual action instructions to generate realistic predictions 50 frames into the future. This model combines techniques such as visual feature extraction of ResNet18, temporal modeling of LSTM, and feature fusion of the attention mechanism. The training results show that Instruct pix2pix and the LSTM model can predict the robot action sequence quite well. In the training, we adopted the comprehensive loss function integrating SSIM and MSE losses, and combined GRMG regularization and EMA for parameter update, effectively improving the accuracy and stability of the model. In terms of the quality of the images on the accuracy of prediction, we put forward the model of the image quality and precision of prediction showed excellent performance.

For future work, we plan to extend our dataset with more diverse tasks and longer temporal horizons. Additionally, incorporating multi-view image inputs or fusing robot state information (e.g., joint positions) could further improve prediction fidelity. Exploring transformer-based temporal models or diffusion-based video prediction approaches also represents promising directions for more accurate and coherent long-term predictions. For the robot action prediction model, there is still much work that can be further improved. We can explore more complex and refined models to better capture the movement trajectories of robots, better capture image features, and enhance the effect of feature fusion. Expand the task of robot behavior prediction to other more complex and real scenarios.

6. Distribution of the Workload

Xixi Chen (20%): Contributed to the implementation of the data generation pipeline and the writing of the Methods section.
Yumeng Lu (20%): Contributed to part of the work of data generation and the writing of Introduction, Related Works and the Methods section.
Han Gao(20%): Contributed to the design and training of the robotic action prediction model, and was responsible for writing the relevant Methods and Experiments sections.
Yiyang Dong(20%): Contribute to the fine-tuning of the instructPix2Pix model and the organization of its input and

output.
Juntao Liu(20%): Participated in the training of the robotic action prediction model and proofread the content and format of the final report.

References

[1] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to Follow Image Editing Instructions. *arXiv preprint arXiv:2211.09800*, 2022. 1

[2] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv preprint arXiv:1611.07004*, 2016. 1

[3] Siyuan Liu, Linfeng Wu, Bowen Li, Hongjie Tan, Haochen Chen, Zheng Wang, Kai Xu, Hang Su, and Jun Zhu. RDT-1B: A Diffusion Foundation Model for Bimanual Manipulation. *arXiv preprint arXiv:2410.07864*, 2024. 1

[4] Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, et al. RoboTwin: Dual-Arm Robot Benchmark with Generative Digital Twins. *arXiv preprint arXiv:2504.13059*, 2025. 1

[5] Peng Li, Haoyu Wu, Yuwei Huang, Chunhui Cheang, Lei Wang, and Tao Kong. GR-MG: Leveraging Partially Annotated Data via Multi-Modal Goal-Conditioned Policy. *arXiv preprint arXiv:2408.14368*, 2024. 1

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. 2

[7] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 2

[8] Alain Hore and Djemel Ziou. Image Quality Metrics: PSNR vs. SSIM. *Proceedings of the 20th International Conference on Pattern Recognition (ICPR)*, pages 2366–2369, 2010. 2