2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Proposal and Demonstration of a Robot Behavior Planning System Utilizing Video with Open Source Models in Real-World Environments

Yuki Akutsu[1], Takahiro Yoshida[1], Yuki Kato[1], Yuichiro Sueoka[1], Koichi Osuka[1]

Abstract— **In the field of robotics, researches have sought to control robots capable of dealing with a variety of environments and tasks generically, through the use of foundation models. Among these, the systems for robot behavior planning utilizing video have also been proposed. The system enables the generation of robot behaviors that are not dependent on specific environments or tasks. This is achieved by generating videos based on text input, which utilizes the vast knowledge inherent in the foundation models. Also, by using a visual interface such as video, it is possible to confirm the behavioral indicators on which the robot is operating. Although, there are few examples of research on robot behavior planning utilizing video. Previous studies have emphasized the verification of behavior generation utilizing video, with simplified object manipulation for testing on simulations. This is not enough to demonstrate the usefulness of robot behavior planning utilizing video in real-world environments. In addition, the systems from previous studies are not open, and such systems have not been sufficiently discussed. This paper attempts to construct robot behavior planning utilizing video as an open system, and to verify the validity of the behavior planning using actual machines. In this paper, we first focus on using Robotis's TURTLEBOT3 Waffle Pi and Mobile Manipulator(referred to as "Waffle") to construct robot behavior planning system utilizing video. Second, we create planning videos targeting the pick-and-place motion using the proposed system, and control the arm part of Waffle in the actual machine verification. Finally, by comparing the target coordinates from the planning video with the coordinates observed from the actual machine, we can confirm whether it is possible to control Waffle as planned. Errors are calculated from the coordinate comparison, and the control is performed again. Based on the results, we verify whether the proposed system is useful for controlling robots in real-world environments.**

I. INTRODUCTION

In recent years, robotics research has sought to leverage the knowledge gained from foundation models trained on large amounts of Internet data. This approach aims to improve robot control so that robots can perform generic tasks and adapt to diverse environments [1]–[4].

Among these robot control systems using foundation models, there are systems that use video generation models, such as Unipi [5]. This paper deals with robot behavior planning utilizing video. Utilizing video for robot behavior planning enables the synthesis of new combinatorial behaviors independent of specific environments or tasks. This is achieved by exploiting the rich linguistic representations of the foundation models. In addition, by using a visual interface such as video in planning, it becomes possible to verify the basis for the robot's behaviors.

One such robot behavior planning system utilizing video is Unipi. In the previous research on Unipi, the verification on simulation was advanced. The objective of the previous research was to assess the flexibility of the robot behavior planning. To this end, the simulation was designed to simplify the object grasping process so that the system can automatically grasp the nearest object upon receiving a grasp command. Thus, much work remains to be done in order to understand the usefulness of utilizing video for robot behavior planning in the real-world environments. This study investigates the feasibility of a robot behavior planning system utilizing video in real-world environments.

Before conducting the investigation towards the research objective, it is necessary to build the entire system. This is because that the previous studies have not made the robot behavior planning systems utilizing video open. There are two challenges in constructing the entire system. The first challenge is the difficulty in constructing and training a video generation model capable of performing behavior planning. This challenge is also noted in the field of video generation model, where the complexity of modeling video datasets [6]–[8] and the lack of video-text paired datasets with explanations are cited as causes. Thus, in this study, we use open-source models to overcome the complexity of constructing video generation models and reduce the amount of training data required. The second challenge is the difficulty in independently creating the model to extract control inputs from a video. In this study, we focus on YOLO [9], which is capable of detecting specific objects in videos. By performing additional training on the open-source YOLO model, it becomes possible to reduce the amount of training data required, thus making it a viable option for use as a model. In this paper, we also aim to facilitate deeper discussions about such systems by constructing a robot behavior planning system utilizing video with open-source models.

Fig. 1 illustrates the lightweight system that enables robot behavior planning utilizing video. We explain the flow of the proposed system. First, the task prompt is given to the video generation model. From there, the video generation model generates the video according to the task prompt. Second, the generated video is input into YOLO to obtain the coordinates of the robot's end effector. Third, the coordinates obtained by YOLO are converted into the Cartesian coordinates. By solving the inverse kinematics from the Cartesian coordinates, we obtain the control inputs and use them to control

[1]Graduate School of Engineering, Osaka University, Japan [akutsu.y@dsc.,yoshida.t@dsc.,kato.y@dsc., sueoka@,osuka@]mech.eng.osaka-u.ac.jp

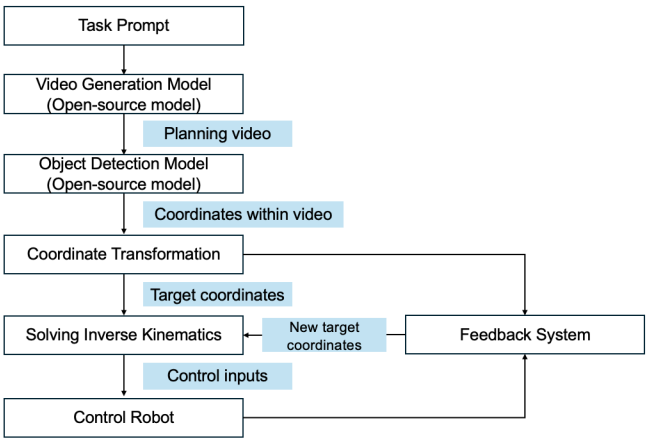

Fig. 1. Structure of a lightweight feedback flow for robot behavior planning utilizing video.

the robot. The results of the control are compared with the robot behavior planning, the control inputs are corrected, and the control is performed again.

We control the ROBOTIS TURTLEBOT3 Waffle Pi and Mobile Manipulator using the proposed system. We conduct a comparison between the target coordinates of the arm obtained from the planning video and the coordinate changes of the arm during the control. Through conducting this verification, we demonstrate the feasibility of a robot behavior planning system using videos in real-world environments.

The structure of this study is as follows. In Sec. II, we discuss the related studies. In Sec. III, we discuss the overview of the video generation model, the object detection model and the coordinate transformation model. In Sec. IV, we explain the environment for controlling Waffle and compare the control results with the target positions from the planning video. Finally, in Sec. V, we summarize the contents of this study and discuss future prospects for this study.

II. RELATED WORKS

A. Video Generation Models

In the field of video generation models, the vast amount of knowledge available on the Internet makes it possible to create of videos that appear to be real. For example, models such as Sora [10], announced by OpenAI, and Imagen Video [11], announced by Google, have made it possible to generate high-quality videos that correspond to the content of text. Although these models are not yet open, we believe that their open-sourcing would improve the range and accuracy of control through videos in the proposed system of this study.

B. Robot Control Systems Using Foundation Models

Research is progressing on robot control systems using foundation models trained on vast amounts of Internet data, with verification being conducted in real-world environments. For example, SayCan [3] plans the robot behavior at the language level from the task prompt and generates actions. Given the task prompt, LLM [12]–[14] breaks down

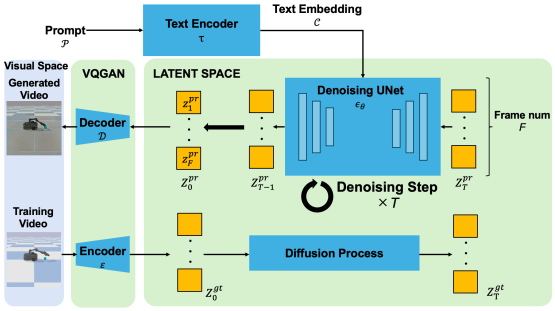

Fig. 2. Structure of ModelScopeT2V composed of three modules: Text Encoder, VQGAN and Denoising UNet.

the task into low-level actions, resulting in the generation of robot behaviors. In SayCan, it is necessary to preset low-level operations to prevent outputs that do not match the actual situation, which limits the range of actions. Robotis Transformers2(RT-2) [4] is another illustration of the same point. It is possible to extract actions from the task prompt by training to directly output the control inputs from the camera's observation images and the task prompt. In RT-2, the generalization ability to recognize objects and environments not present in the training data was demonstrated.

III. PROPOSED SYSTEM

In this section, we first discuss the video generation model for robot behavior planning. Second, we describe the object detection model for detecting the arm of Waffle from videos. Finally, we discuss the model for converting the arm coordinates on the video screen to the Cartesian coordinates.

A. Learning and Results of Video Generation Model

In this paper, we use the ModelScopeT2V [15], an open-source video generation model based on diffusion models. The reason for using this model is that it has been reported to have similar or superior performance both quantitatively and qualitatively. This performance is compared to other diffusion model-based video generation models such as MAKE-A-VIDEO [16] and Imagen Video [11]. The structure of ModelScopeT2V is shown in Fig. 2. We performed learning by combining the ModelScopeT2V with the Low-Rank Adaptation (LoRA) model [17]. LoRA is suggested in the ModelScopeT2V paper as something that could potentially improve the performance of video generation models.

1) Creation of Video Datasets: In order to train a video generation model that enables robot behavior planning, it is necessary to create datasets of the robot. This section explains the contents of the datasets used for training and the settings of the simulation environment used to create the robot video datasets. Firstly, the content of the video is a recording of Waffle picking up an object placed in front of it and moving it forward in the simulation. For the video data, we recorded 24 videos, changing the color and shape of the objects. Secondly, we used Pybullet [18], a 3D physics simulation software, to create the training datasets. The computation cycle was set to 0.01s, and a simulation of

TABLE I

ACTIONS OF WAFFLE AT EACH TIME STEP ON PYBULLET.

Steps	Action
0〜100	set arm position
100〜300	catch object
300〜500	move arm to position(0.2, 0, 0.5)
500〜700	move arm to position(0.3, 0, 0.3)
700〜900	move arm to position(0.4, 0, 0)
900〜1,100	release object
1,100〜1,300	move arm to position(0.3, 0, 0.3)
1,300〜1,500	move arm to position(0.1, 0, 0.2)

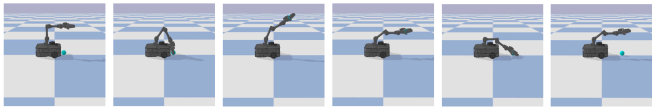

Fig. 3. Video frames of the action of Waffle at each time step on Pybullet for training ModelScopeT2V.

1,500 steps was performed based on the computation cycle. In the simulation environment, Waffle is positioned at the origin, and the Cartesian coordinates of the object are located at (0.1, 0, 0). The camera capturing the simulation situation is also placed at (0, -2.0, 0.5). The camera is set so that Waffle is in the center of the screen. Table I shows the specific steps of how Waffle works in the simulation environment. Fig. 3 also shows the video frames created step by step in the simulation. Here, images are presented that show the state after switching the robot's operation in the simulation.

2) Video Captioning: The video captioning method was carried out with reference to Video BLIP-2 [1], which is capable of captioning video datasets. The model used for captioning in Video BLIP-2 is BLIP-2 [19], which is used for tasks such as captioning and visual question answering based on the input image. However, with BLIP-2, it can only read one image (a single frame of a video), and even if it captures a scene where the robot is moving, it ends up captioning it as if it's stationary. For example, even if you extract a scene where a robot is lifting an object as a frame, it will be described as if the robot is holding the object stationary. In this study, when we tried to explain the robot's movements with BLIP-2, the captions became inadequate, so we used LLM instead of BLIP-2. Fig. 4 shows an overview of the video captioning.

First, we randomly select three frames from the video, arrange the three frames in order, and explain in Japanese what the middle frame is doing. We add Japanese subtitles to be able to check for subtle nuances. At this stage, we use gpt-4-preview [2], which is capable of describing images. Second, the caption becomes too long in the first stage of output, so the output needs to be summarized in one sentence. We use gpt-4 [3]. Third, the input for the ModelScopeT2V is only in English, so the output in the second stage needs to be translated into English. In this stage, we use gpt-3.5-turbo-0613 [4] because only translation tasks are required.

[1]https://github.com/ExponentialML/Video-BLIP2-Preprocessor

[2]https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo

[3]https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo

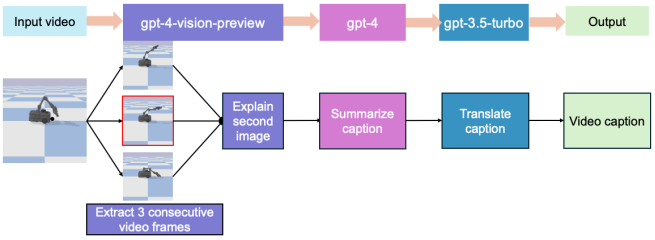

Fig. 4. Flow of creating video captions for the behavior of Waffle.

TABLE II

HYPERPARAMETERS FOR TRAINING MODELSCOPET2V

Parameter	Value
Learning Rate	5e-6
Adam Beta1	0.9
Adam Beta2	0.999
Adam Weight Decay	0
Adam Epsilon	1e-8
Max Grad Norm	1.0
Seed	64
Mixed Precision	fp16
LoRA Rank	1

3) Learning Settings: We fine-tune the video generation model using the AdamW optimizer [20] and train the model on NVIDIA A100-SXM4-40GM. The model is trained using only video-text pair datasets as training data. The hyperparameters set for the training are listed in Table II.

4) Learning Results: Fig. 5 shows the result of creating a 90-frame video with the video generation model at the point where the additional learning steps have been performed 85,000 times in this study. The input prompt at this point is as follows : Waffle, a mobile robot, is performing a task of holding a blue rectangular object horizontally with its gray robotic arm and moving it to a different location.

In Fig. 6, the graph illustrates the loss during the training of the video generation model. The loss distribution makes it difficult to determine whether the overall learning loss is decreasing. However, as can be seen from the yellow approximation line that linearly approximates the loss in Fig. 6, it is evident that the loss is decreasing over time with each step.

B. Object Detection Model

In this study, we focused on YOLO, which is used in the field of object detection as a model for detecting specific objects from videos. Specifically, we trained YOLOv5 [21] to detect the arm part of Waffle. We first explain the method

Fig. 5. Video frames generated by fine-tuning the video generation model.

[4]https://platform.openai.com/docs/models/gpt-3-5-turbo

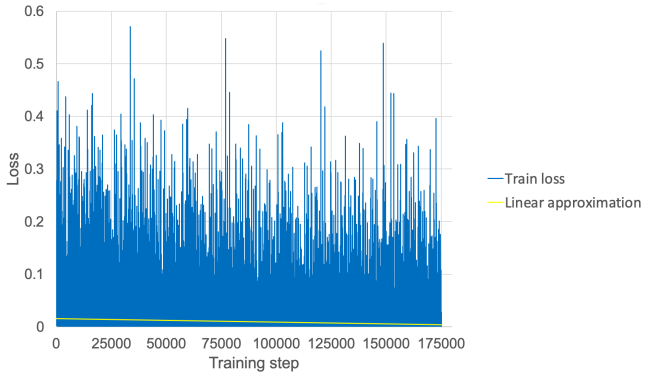

Fig. 6. Train loss during the training of the video generation model.

of training the object detection model which extracts coordinates from the videos of robot behavior planning. Then, we discuss the results of training YOLO.

1) Training The Obeject Detection Model: We tried to control a robot by detecting the position of Waffle's arm in the robot behavior planning video, and solving the inverse kinematics through programming. Thus, the object to be detected is the arm part of Waffle. The video created in Sec. III-A.1 was used as the training datasets for detecting the arm part of Waffle. LabelImg [5] was used to create the ground truth datasets indicating the region of the arm of Waffle from the training datasets.

YOLOv5 was fine-tuned using Stochastic Gradient Descent (SGD) and the model was trained on NVIDIA GeForce RTX4090 [6]. The maximum training step was set to 1,000, and the model was saved. We use EARLYSTOPPING [7], which automatically stops the training if the loss does not change for a certain period of time, so that the training can be automatically stopped before the maximum number of steps is reached. The hyperparameters set in the learning process are listed in Table III.

2) Result of Fine-tuned YOLOv5: Firstly, Fig. 7 shows the change in learning loss. From Fig. 7, the maximum steps set for training YOLOv5 is 1,000 steps, but the training is stopped at about 550 steps. This is due to the influence of the EARLYSTOPPING program. The training results show that the learning loss started at 0.12 and finally converged to about 0.02, marking the end of the training.

Secondly, the result of detecting the arm of Waffle by using fine-tuned YOLOv5 is shown in Fig. 8. The red outline in Fig. 8 indicates the bounding box that encloses the region of the object to be detected. Through fine-tuning in this study, it has been confirmed that objects can be detected even if the position of the arm of Waffle moves within the video.

C. Coordinate Transformation

In Sec. III-B, it became possible to detect the coordinates of the arm of Waffle from the video, however, the coordinates

[5]https://github.com/HumanSignal/labelImg

[6]https://www.nvidia.com/ja-jp/geforce/graphics-cards/40-series/rtx-4090/

[7]$https://pytorch.org/ignite/generated/ignite.handlers.early_stopping.EarlyStopping.html$

TABLE III
HYPERPARAMETERS FOR TRAINING YOLOV5

Parameter	Value
Patient	100
Initial Learning Rate	0.01
Final OneCycle Learning Rate	0.01
SGD Momentum	0.937
Weight Decay	5e-4
Warmup Epoch	3.0
Warmup Bias Learining Rate	0.1
Box Loss Gain	0.05
Class Loss Gain	0.5
Class BECLoss Positive Weight	1.0
Object Loss Gain	1.0
Object BECLoss Positive Weight	1.0
IoU Training Threshold	0.2
Anchor-Multiple Threshold	4.0
Focal Loss Gamma	0
HSV Hue	0.015
HSV Saturation	0.7
HSV Value	0.4
Image Degree	0
Image Translation	0.1
Image Perspective	0
Image Flip Up-Down	0
Image Flip Left-Right	0.5
Mosaic	1.0
Image Mixedup	0
Segment Copy-Paste	0

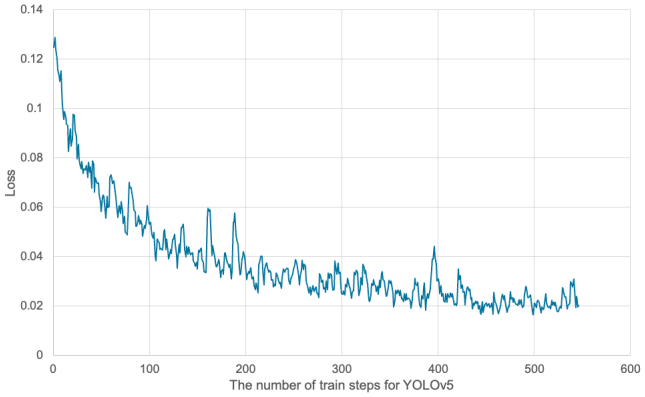

Fig. 7. Change in loss during the training of YOLOv5.

of the object detected by YOLOv5 correspond to the coordinates on the video screen. To solve the inverse kinematics, it is necessary to express the coordinates of the arm of Waffle in the Cartesian coordinates. Thus, this section describes the method for converting the coordinates on the video screen to the Cartesian coordinates. By obtaining the coordinates on the video screen and the Cartesian coordinates for the same video and comparing them, we can determine the relationship between the two sets of coordinates.

First, in Sec. III-C, we create the video in which Waffle changes only in the x-axis direction in the simulation, and it changes only in the y-axis direction to compare the coordinates on the video screen with the Cartesian coordinates. Second, in Sec. III-C.2, we show the relationship between the coordinates of the robot arm on the video and the Cartesian coordinates. Finally, we derive the coordinate transformation equation from each coordinate obtained.

Fig. 8. YOLOv5 detects the robot arm of Waffle in training video.

TABLE IV

ACTIONS OF WAFFLE AT EACH TIME STEP ON PYBULLET FOR DETECTING THE X-COORDINATE OF THE ARM.

Steps	Action
0∼300	set arm position
300∼570	waffle running

1) Creating the Video of the Arm of Waffle Changing only in the X-axis Direction and only in the Y-axis Direction in the Simulation: We describe the simulation environment when the arm of Waffle changes only in the x-axis direction and when it changes only in the y-axis direction. When there is a change only in the x-axis direction, the initial position of the body of Waffle is set to (-0.6, 0, 0). This initial position refers to the limit position where the robot arm appears in the camera that records a video of the simulation. At this point, the position of the robot arm is (-0.45, 0, 0.30). From this initial position, Waffle moves straight along the x-axis while holding the arm in a fixed position, reaching the coordinates (0.43, 0, 0.30). The computation cycle was set to 0.01s, and a simulation of 1,500 steps was performed based on the computation cycle. Additionally, the camera capturing the simulation situation is positioned at (0, -2.0, 0.5) and is oriented so that the origin is centered on the screen. Table IV outlines the specific steps of Waffle's operation in the simulation environment. Fig. 9 shows the video frames that represent the changes in the x-coordinate of the arm of Waffle.

If there is only a change in the y-axis direction, the initial position of the body of Waffle is set to (0, 0, 0). At this point, the position of the robot arm is (0.15, 0, 0.30). From the initial position of the robot arm, the arm of Waffle moves the coordinates to (0.2, 0, 0.01) by following the set arm positions outlined in Table V, and gradually raises the arm to (0.2, 0, 0.5) without changing the x-axis. Also, the camera that captures the simulation situation is placed at (0, -2.0, 0.5), and the camera is oriented so that the origin is centered. Table V shows the specific steps of how Waffle works in the simulation environment. In Fig. 10, we show the video frames of when only the y-coordinate of the arm of Waffle was changed.

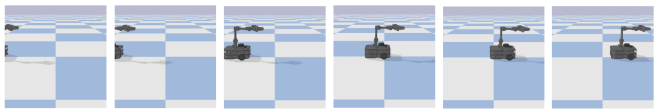

Fig. 9. Video frames of the action of Waffle at each time step on Pybullet for x-coordinate comparison.

TABLE V

ACTIONS OF WAFFLE AT EACH TIME STEP ON PYBULLET FOR DETECTING THE Y-COORDINATE OF THE ARM.

Steps	Action
0∼100	set arm position(0.2, 0, 0.01)
100∼200	move arm to position(0.2, 0, 0.1)
200∼300	move arm to position(0.2, 0, 0.2)
300∼400	move arm to position(0.3, 0, 0.3)
400∼500	move arm to position(0.4, 0, 0.35)
500∼600	move arm to position(0.3, 0, 0.4)
600∼700	move arm to position(0.3, 0, 0.5)

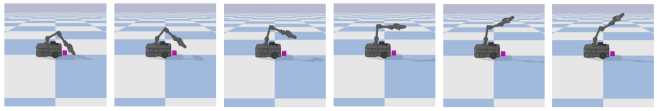

Fig. 10. Video frames of the action of Waffle at each time step on Pybullet for y-coordinate comparison.

2) Methods for Converting the Video Screen Coordinate System to the Cartesian Coordinate System: This section compares the coordinates of the robot arm on the video screen with the Cartesian coordinates of the robot arm in the simulation. The method of obtaining the coordinates is as follows. The coordinates of the robot arm on the video screen were obtained using YOLOv5, which was fine-tuned as described in Sec. III-B. Also, the Cartesian coordinates of the robot arm in the simulation were obtained when creating the dataset in Sec. III-C.1.

First, the figure comparing the coordinates on the video screen when only the x-coordinate of the robot arm is changed, and the Cartesian coordinates in the simulation, is shown as Fig. 11. From Fig. 11, it can be seen that there is a proportional relationship between the coordinates on the video screen when only the x-coordinate of the robot arm is changed and the coordinates in the simulation. If this Fig. 11 is linearly approximated, the equation of the relationship between the two coordinates can be obtained.

$$y = 1.0575x - 0.5362 \tag{1}$$

Second, the figure comparing the coordinates on the video

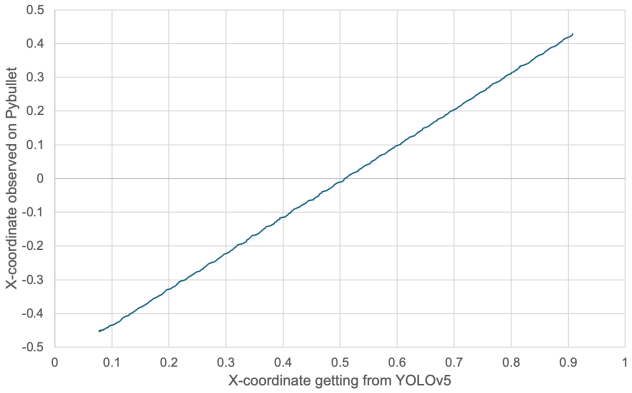

Fig. 11. Comparison of the x-coordinates getting from YOLOv5 and the x-coordinates observed on Pybullet.

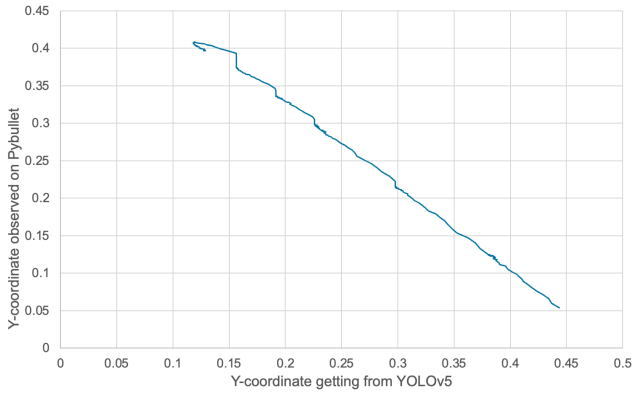

Fig. 12. Comparison of the y-coordinates getting from YOLOv5 and the y-coordinates observed on Pybullet.

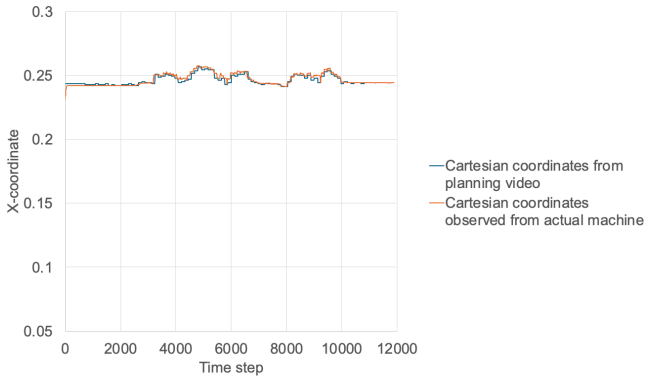

Fig. 14. Comparison between the x-coordinate obtained from the planning video and the actual control x-coordinate.

screen when only the y-coordinate of the robot arm is changed, and the Cartesian coordinates in the simulation, is shown as Fig. 12. From Fig. 12, it can be seen that there is a proportional relationship between the coordinates on the video screen when only the y-coordinate of the robot arm is changed and the coordinates in the simulation. Similar to before, by approximating the graph as a straight line, the equation of the relationship between the two coordinates can be obtained.

$$y = -1.0981x + 0.5441 \tag{2}$$

The x-axis of both the graph comparing x-coordinate and the graph comparing y-coordinate represents the coordinates of the robot arm detected by YOLOv5. Thus, the x-y coordinates in the Cartesian coordinate system can be obtained by substituting the coordinates of the robot arm detected by YOLOv5 into the x values of (1) and (2). It is clear that the derivation of the coordinate transformation equation is obtained.

IV. EXPERIMENTS

The objective of this study is to investigate the feasibility of a robot behavior planning system utilizing video in real-world environments. To demonstrate the research objective, we control the actual machine using an inverse kinematics model based on the target coordinates of the arm. The process of controlling the actual machine is shown in Fig. 13.

A. Experimental Setup

In this section, we describe the specific process by which the experiment is performed. First, the task prompt is entered as text into the video generation model created in Sec. III-A, and the video for the robot behavior planning is generated.

Fig. 13. Control of the actual machine, Waffle.

Since ModelScopeT2V can only accept text input, only the task prompt is entered here. The generated planning video is inputted into YOLOv5, which was trained in Sec. III-B, to obtain the target coordinates of the robot arm on the video screen. Second, the coordinates on the video screen are converted into the Cartesian coordinates using the coordinate transformation equation. Third, the control inputs are obtained by solving the inverse kinematics from the target coordinates in the Cartesian coordinate system. This is how we control Waffle in real-world environments. The method used to obtain the coordinates of the arm of Waffle is to use Acuity Inc.'s motion capture software, Motive [8]. When entering the control inputs, the time sleep is set to one second and the control inputs are sent to the robot, ensuring stable acquisition of the arm's coordinates through motion capture. Finally, by feeding the position changes of the arm obtained by Motive into the feedback system, the target positions are adjusted based on the control error between the target positions of the arm and its position changes, thereby creating new target positions.

B. Comparison of the Results

In this section, we compare the arm's position changes obtained through controlling the actual robot with the target positions of the arm obtained from the planning video. A point of note when comparing each coordinate is that the number of coordinates obtained is different. Specifically, the number of control inputs obtained is 90, since the planning video consists of 90 frames. On the other hand, the number of control data points obtained from the motion capture system for the actual robot is 10,800. There are two reasons for this. The first is that the motion capture system obtains coordinates 120 times per second. The second is that when controlling the robot, one control input is entered per second for the 90 control inputs. When comparing each coordinate in the graph, the 90 control inputs are expanded to 120 per input, resulting in $90 \times 120 = 10,800$, allowing for comparison. Fig. 14 and Fig. 15 show the results of comparing the Cartesian coordinate from the planning video and the Cartesian coordinate observed from the actual device. The

[8]https://www.optitrack.jp/products/software/

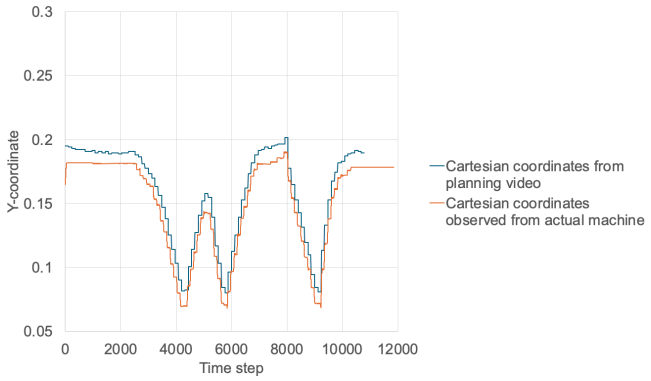

Fig. 15. Comparison between the y-coordinate obtained from the planning video and the actual control y-coordinate.

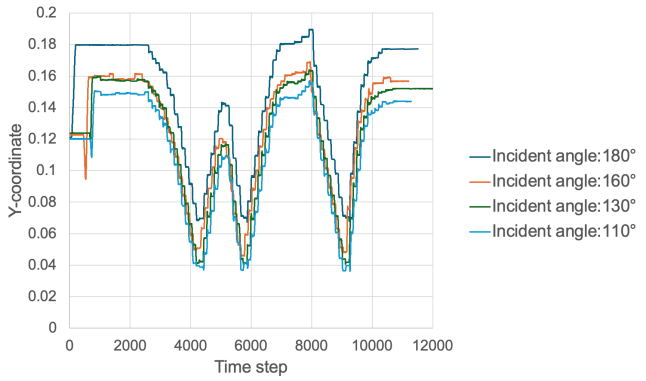

Fig. 16. Comparison of coordinate changes for different incident angle.

time step of the graph is set such that 120 steps correspond to 1 second. Taking the difference between the each coordinate in Fig. 14 and Fig. 15, and averaging the errors over the entire duration of the control gives the mean error.

$$\begin{cases} \epsilon_X = 7.0994\times10^{-4}[\mathrm{m}] \\ \epsilon_Y = 1.1527\times10^{-2}[\mathrm{m}] \end{cases} \tag{3}$$

In this experiment, when controlling the waffle, the manipulator is set to have an incident angle of 180° on the connected link. The reason for this is that, in Sec. III-A.1, the incident angle of the manipulator was set to 180° when the video datasets were created. As space is limited, we have space for no more than an indication of the differences in the incident angle result in different position changes of the arm, even for the same target position. Fig. 16 compares the change in the arm's position for each incident angle when the same target position is entered. From Fig. 16, it can be seen that the variation of the incident angle results in different position changes of the arm for the same target position.

C. Feedback System

The feedback system has three functions. The first function is to compare only the duration of the control period when comparing the target coordinates with the coordinates observed from the actual robot. The reason for this is that when motion capture observes the actual robot, it also captures the initial coordinate data before the robot control begins. We use the cross-correlation function to compare only the control period, excluding the initial coordinate data.

The second function is to calculate the average error by averaging the errors over the entire control period. This method takes the difference between the target coordinates and the observed Cartesian coordinates of the actual robot and divides it by the total control time.

The third function is to correct the target coordinates from the planning video based on the average error. By using the average error to correct the target coordinates, new target coordinates are obtained.

D. Discussion and Evaluation

The results of Sec. IV-C confirmed that the error indicated in (3) had occurred. Based on Sec. IV-B, the target coordinates of the arm were corrected using the feedback system, and the results of the controlled arm are shown in Fig. 17 and Fig. 18.

By taking the difference between the each coordinate in Fig. 17 and Fig. 18, and averaging the errors over the entire control period, we obtain the average error.

$$\begin{cases} \epsilon_X = 3.0951\times10^{-4}[\mathrm{m}] \\ \epsilon_Y = 1.3765\times10^{-4}[\mathrm{m}] \end{cases} \tag{4}$$

From (4), it was observed that by utilizing the feedback system of the proposed system, it is possible to correct errors in the X-coordinate on the order of a few millimeters. Also, focusing on the Y-coordinate, it was found that it is possible to correct errors from several centimeters to several millimeters by using the feedback system. Thus, it is considered that pick and place operations using the proposed system on the actual machine are sufficiently useful. In this paper, we focus on the possibility that errors during a series of actions after picking up an object or while attempting to pick up the object may prevent the success of grasping, so we focus on verification through error correction. In the future, it is planned to measure the success of actually picking up the object.

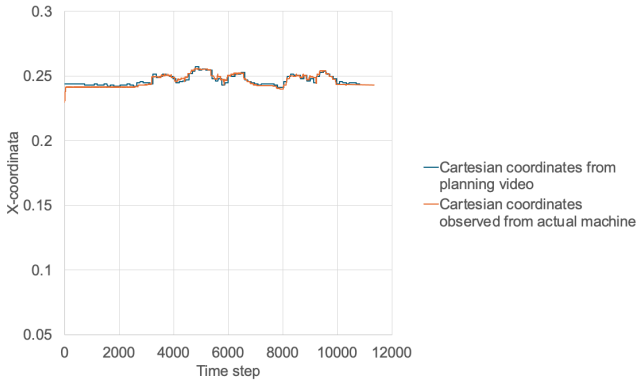

Fig. 17. Comparison between the corrected x-coordinate from the planning video and the actual control x-coordinate.

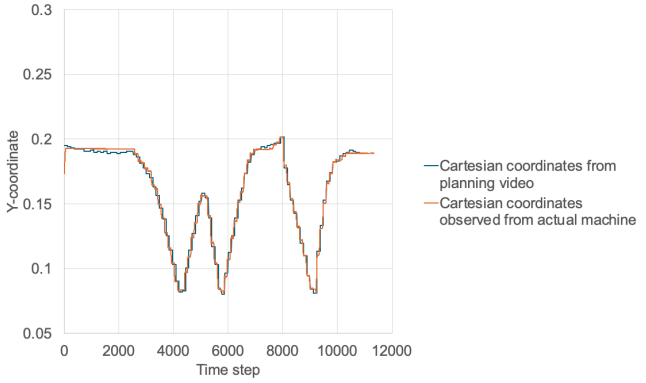

Fig. 18. Comparison between the corrected y-coordinate from the planning video and the actual control y-coordinate.

V. CONCLUSIONS

In this paper, we constructed a robot behavior planning system utilizing video with open systems. Firstly, we fine-tuned ModelScopeT2V and LoRA to create a video generation model that enables robot behavior planning. Secondly, we trained YOLOv5 to detect the arm part of Waffle as a model for obtaining control inputs from a video. Additionally, it became possible to convert the coordinates obtained by YOLOv5 into the Cartesian coordinates by comparing the relationship between the coordinates on the video screen and the Cartesian coordinates. In the experimental setup using the proposed system, errors of 7.0994×10^{-4}[m] in the X-coordinate and 1.1527×10^{-2}[m] in the Y-coordinate were confirmed. By using the feedback system to correct the errors and regenerate the target coordinates, it was confirmed that it was possible to reduce the error between target and actual coordinates to within a few millimeters for both the X and Y coordinates. This makes it possible to perform the same operation as shown in the planning video. Therefore, it is considered that the practical utility of the proposed system is sufficiently evident when used on an actual machine. In the future, we are considering adding a visual feedback system that evaluates the results of robot actions when robot bahavior planning fails, using images to evaluate robot behavior. In addition, we aim to expand the range of robot behavior planning representations by increasing the datasets. This will allow for replanning based on visual feedback.

ACKNOWLEDGMENT

This research was partially supported by JSPS KAKENHI Grant Number 21K14183, 21H05104a and JST [Moonshot R&D][Grant Number JPMJPS2032] and JST ACT-X Grant Number JPMJAX22A9. The authors express their sincere gratitude for the support provided.

REFERENCES

[1] D. Driess, F. Xia, M. S. M. Sajjadi, *et al.*, "PaLM-E: An Embodied Multimodal Language Model," *arXiv e-prints*, p. arXiv:2303.03378, Mar. 2023.

[2] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 894–906. [Online]. Available: https://proceedings.mlr.press/v164/shridhar22a.html

[3] M. Ahn, A. Brohan, N. Brown, *et al.*, "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances," *arXiv e-prints*, p. arXiv:2204.01691, Apr. 2022.

[4] A. Brohan, N. Brown, J. Carbajal, *et al.*, "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," *arXiv e-prints*, p. arXiv:2307.15818, July 2023.

[5] Y. Du, M. Yang, B. Dai, *et al.*, "Learning Universal Policies via Text-Guided Video Generation," *arXiv e-prints*, p. arXiv:2302.00111, Jan. 2023.

[6] I. Skorokhodov, S. Tulyakov, and M. Elhoseiny, "StyleGAN-V: A Continuous Video Generator with the Price, Image Quality and Perks of StyleGAN2," *arXiv e-prints*, p. arXiv:2112.14683, Dec. 2021.

[7] W. Hong, M. Ding, W. Zheng, *et al.*, "CogVideo: Large-scale Pretraining for Text-to-Video Generation via Transformers," *arXiv e-prints*, p. arXiv:2205.15868, May 2022.

[8] S. Yu, J. Tack, S. Mo, *et al.*, "Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks," *arXiv e-prints*, p. arXiv:2202.10571, Feb. 2022.

[9] J. Redmon, S. Divvala, R. Girshick, *et al.*, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv e-prints*, p. arXiv:1506.02640, June 2015.

[10] T. Brooks, B. Peebles, C. Holmes, *et al.*, "Video generation models as world simulators," 2024. [Online]. Available: https://openai.com/research/video-generation-models-as-world-simulators

[11] J. Ho, W. Chan, C. Saharia, *et al.*, "Imagen Video: High Definition Video Generation with Diffusion Models," *arXiv e-prints*, p. arXiv:2210.02303, Oct. 2022.

[12] OpenAI, "Gpt-4 technical report," 2023.

[13] H. Touvron, T. Lavril, G. Izacard, *et al.*, "Llama: Open and efficient foundation language models," 2023.

[14] A. Chowdhery, S. Narang, J. Devlin, *et al.*, "Palm: Scaling language modeling with pathways," 2022.

[15] J. Wang, H. Yuan, D. Chen, *et al.*, "ModelScope Text-to-Video Technical Report," *arXiv e-prints*, p. arXiv:2308.06571, Aug. 2023.

[16] U. Singer, A. Polyak, T. Hayes, *et al.*, "Make-A-Video: Text-to-Video Generation without Text-Video Data," *arXiv e-prints*, p. arXiv:2209.14792, Sept. 2022.

[17] E. J. Hu, Y. Shen, P. Wallis, *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," *arXiv e-prints*, p. arXiv:2106.09685, June 2021.

[18] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[19] J. Li, D. Li, S. Savarese, *et al.*, "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models," *arXiv e-prints*, p. arXiv:2301.12597, Jan. 2023.

[20] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *arXiv e-prints*, p. arXiv:1711.05101, Nov. 2017.

[21] H. Liu, F. Sun, J. Gu, *et al.*, "Sf-yolov5: A lightweight small object detection algorithm based on improved feature fusion mode," *Sensors*, vol. 22, no. 15, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/15/5817