Prediction of Delay-Free Scene for Quadruped Robot Teleoperation: Integrating Delayed Data With User Commands

Seunghyeon Ha, Seongyong Kim, and Soo-Chul Lim

***Abstract*—Teleoperation systems are utilized in various controllable systems, including vehicles, manipulators, and quadruped robots. However, during teleoperation, communication delays can cause users to receive delayed feedback, which reduces controllability and increases the risk faced by the remote robot. To address this issue, we propose a delay-free video generation model based on user commands that allows users to receive real-time feedback despite communication delays. Our model predicts delay-free video by integrating delayed data (video, point cloud, and robot status) from the robot with the user's real-time commands. The LiDAR point cloud data, which is part of the delayed data, is used to predict the contents of areas outside the camera frame during robot rotation. We constructed our proposed model by modifying the transformer-based video prediction model VPTR-NAR to effectively integrate these data. For our experiments, we acquired a navigation dataset from a quadruped robot, and this dataset was used to train and test our proposed model. We evaluated the model's performance by comparing it with existing video prediction models and conducting an ablation study to verify the effectiveness of its utilization of command and point cloud data.**

***Index Terms*—Deep learning methods, telerobotics and teleoperation, visual learning.**

I. Introduction

In RECENT years, quadruped robot technology has seen significant advancements, and these developments have facilitated the emergence of various quadruped robots [1]. Having the ability to use legs for locomotion provides significant benefits in navigating rough terrain [2], so various studies have utilized this capability. For instance, Halder et al. [3] proposed a collaborative construction site monitoring system utilizing both quadruped robots and humans, demonstrating enhanced objectivity in evaluations and significant time savings compared to traditional human-only monitoring processes. In another study, Cruz et al. [4] introduced a method for controlling robots through Mixed Reality, which achieved enhanced stability and efficiency of search and rescue (SAR) operations utilizing quadruped robots.

In addition to collaborative work between robots and humans, there have been various research efforts aiming toward the autonomous operation of quadruped robots. For instance, Angelini et al. [5] proposed a framework for the autonomous monitoring of habitats using quadruped robots, while Halder et al. [6] performed 3D modeling of construction sites by autonomously operating a quadruped robot for monitoring purposes. However, these approaches have limitations, as quadruped robots are unable to ensure stability on uneven terrains, often leading to lose balance and fall.

Thus, for quadruped robots that are still lacking in complete autonomy, teleoperation can be used as either a safety measure or an alternative method to control quadruped robots. Indeed, controllable systems like surgical robots and autonomous vehicles have achieved enhanced utility by adopting teleoperation [7], [8], [9]. Similarly, research that has incorporated the use of teleoperation systems in quadruped robots has emerged. Halder et al. [10] constructed a cloud-based teleoperation system using 5G networks that utilized augmented reality to provide stable control and objective monitoring capabilities. Further, Zhou et al. [11] and Cruz et al. [12] developed systems for controlling quadruped robots equipped with manipulators using human gestures, thereby expanding the potential applications of quadruped robots.

Although integrating teleoperation systems can enhance the utility of robots, doing so can also lead to certain challenges. When using a teleoperation system to control a robot remotely, communication delays cause latency in data transmission. In quadruped robot teleoperation, operators receive the remote robot's video as visual feedback to their commands. However, if there is a communication delay that leads to a delay in this visual feedback being provided to the user, the user's controllability decreases and the risks in the remote environment increase [13].

With recent advancements in communication technologies such as 5G networks, there has been a significant reduction in communication delay associated with the transmission of high-bandwidth data (e.g., video, point cloud, etc.). However, there are still certain challenges regarding communication delay caused by physical distance in long-distance communication. In the cloud-based quadruped robot teleoperation system using

Received 27 June 2024; accepted 13 January 2025. Date of publication 29 January 2025; date of current version 11 February 2025. This work was supported by the National Research Foundation of Korea (NRF), Korean Government, under Grant NRF-2020R1A2C1008883. This letter was recommended for publication by Associate Editor S. F. Atashzar and Editor J.-H. Ryu upon evaluation of the reviewers' comments. *(Seunghyeon Ha and Seongyong Kim contributed equally to this work.) (Corresponding author: Soo-Chul Lim.)*

The authors are with the Department of Mechanical, Robotics and Energy Engineering, Dongguk University, Seoul 04620, South Korea (e-mail: shha_419@dgu.ac.kr; sykim_414@dgu.ac.kr; limsc@dongguk.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3536222, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3536222

5G networks proposed in [10], the communication delay was observed to be 200∼300 ms. Consequently, users still face the issue of not receiving delay-free feedback from the remote robot in response to commands due to communication delays.

One method that can be used to address these issues is to predict delay-free video from received delayed video. Various video-to-video prediction models have been researched [14], [15], [16], [17], including models based on an autoencoder, long short term-memory (LSTM) [18], and transformer [19]. These models involve utilizing texture and motion information between input video frames to predict future video frames. However, predicting future videos based solely on its frames has limitations, as it depends heavily on prior frames. As a result, it becomes challenging to predict drastic changes in direction.

To address these limitations, studies have been attempted to incorporate real-time user input into prediction of future videos. For example, Ko et al. [20] generated high update rate videos using an operator's command to control a gripper along with low update rate videos; in another study, Yoon et al. [21] generated a real-time video using the guidance information of the user operating the manipulator along with the delayed video. These studies generated video by utilizing real-time information along delayed video to increase user controllability, in teleoperation systems with communication delay. However, these studies mainly utilized datasets obtained in static backgrounds, the resulting videos with limited environmental variations, such as changes in lighting, motion, or surroundings.

In this letter, we propose a delay-free video generation model that is applicable to quadruped robot teleoperation system, and which utilize information that can predict future video frames in addition to delayed video frames. With the proposed model, delay-free visual feedback to user commands can be predicted in a quadruped robot teleoperation system with communication delays, as shown in Fig. 1. Further, for the experiment of the proposed model, quadruped robot datasets were acquired from places containing various contents (e.g., tree, statue, building, human, etc.).

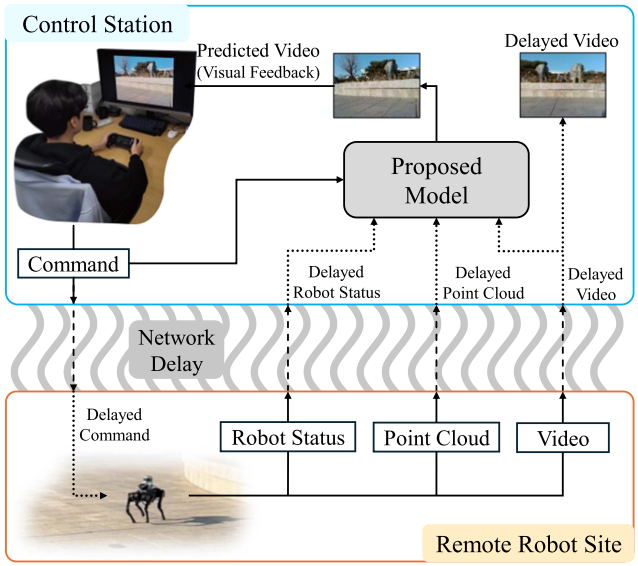

Fig. 1. Framework of generating delay-free video using proposed model. The proposed model predicts delay-free visual feedback on user commands by integrating delayed data (video, point cloud, robot status) transmitted from the remote robot with the user's real-time commands.

II. Related Works

A. Video Prediction Method

The existing competitive video prediction models have mostly been constructed in recurrent neural network (RNN) based model such as LSTM [22]. In particular, many video prediction studies have presented models using Convolutional LSTM (ConvLSTM) [15], which is a modified memory of LSTM that is suitable for the sequential processing of 2D data like images [23], [24]. Further, Wang et al. [16], [25] introduced PredRNN as well as PredRNN-V2 constructed with spatiotemporal LSTM(ST-LSTM), which separates and extracts spatial and temporal information, thereby enhancing the ability to predict future frames using input video frames.

Meanwhile transformer-based models have recently been shown to outperform RNN-based models in sequence data processing tasks such as natural language processing. Transformer-based models have also emerged as promising models for use in video prediction tasks. As a specific example, Yan et al. [26] introduced a model that uses 3D CNN to encode input videos and a transformer for predicting future videos. In another study, Liu et al. [27] introduced ConvTransformer, which modifies the attention mechanism within the traditional transformer to utilize 2D CNNs, thus enabling the processing of 2D sequential data while maintaining its 2D structure. Further, Yuan et al. [28] and Ye et al. [17] proposed the VPTR-NAR model, which utilizes 2D CNNs for attention mechanisms and separates spatial and temporal self-attention, demonstrating competitive results compared to existing state-of-the-art models.

In this study, we modified the video-to-video prediction model VPTR-NAR to construct a model that takes video, point cloud, robot status, and command as input to predict videos.

B. Conditional Video Generation

Generating images or videos can leverage various types of conditional information in addition to input frames. For example, many studies have explored image generation based on text descriptions [29], [30], [31]. Other research has utilized human poses as conditional information to predict future frames [32], [33], [34].

Moreover, similar to our present research, there have been studies applying video prediction in teleoperation systems using conditional information. In teleoperation systems, robots move according to the operator's commands, and the video changes accordingly, thus allowing future videos to be predicted according to the operator's commands [20], [21], [35]. Ko et al. [20] generated a high update rate videos using an operator's commands as conditional information. Similarly, Yoon et al. [21] predicted real-time videos using the manipulator guidance information of the user as conditional information.

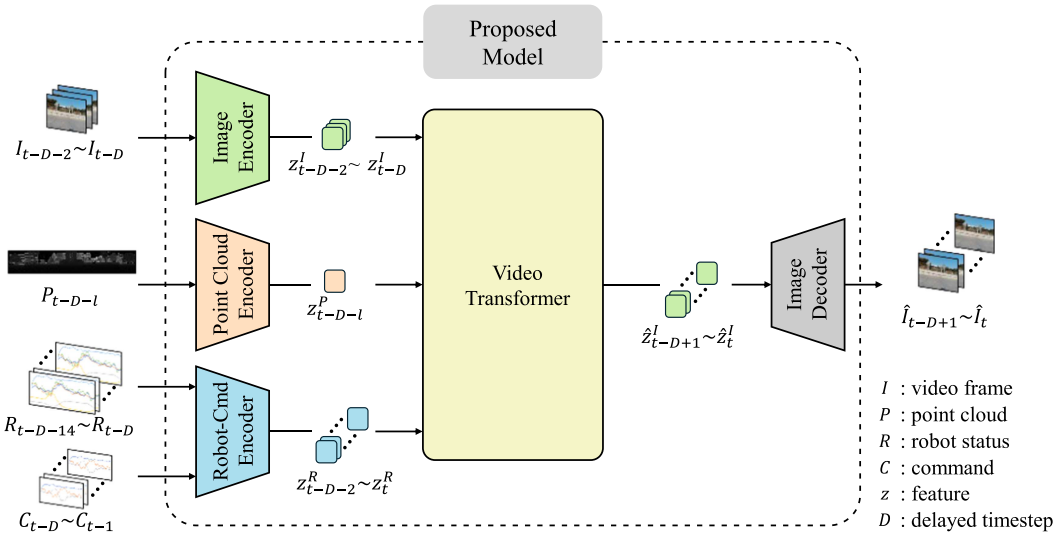

Fig. 2. Network architecture for delay-free video generation. The proposed model consists of *encoders* for each input data ($\boldsymbol{I_{t-D-2}} \sim \boldsymbol{I_{t-D}}$, $\boldsymbol{P_{t-D-l}}$, $\boldsymbol{R_{t-D-14}} \sim \boldsymbol{R_{t-D}}$, $\boldsymbol{C_{t-D}} \sim \boldsymbol{C_{t-1}}$) to extract features from the input data, a *video transformer* to predict the features of the current frame using the features of each data, and an *image decoder* to decode the features of the current frame into RGB image ($\boldsymbol{\hat{I}_{t-D+1}} \sim \boldsymbol{\hat{I}_t}$). ($\boldsymbol{l}$ in, $\boldsymbol{P_{t-D-l}}$ represents the time information of the used point cloud and is used for positional encoding of the video transformer.).

Inspired by such prior research, we constructed a model to accurately predict future videos by leveraging robot status and commands as conditional information.

C. Point Cloud

Point clouds contain 3D geometry and visual information, but they pose difficulties for feature extraction due to their sparse and unordered data representation. Therefore, point cloud data is often transformed into forms such as voxels, points, or 2D projection images to suit the model's objectives [36], [37]. In these methods, voxelization and point-based methods preserve the 3D geometry of point clouds, but they incur computational costs and face challenges in data fusion due to representation differences between images and data.

By contrast, 2D projection methods, although they lose some 3D geometric detail, offer increased computational efficiency and ease of fusion with image data. Leveraging these advantages, Li et al. [38] used 2D depth image extracted by 2D cylindrical projection of point cloud data to achieve successful 3D vehicle detection. Similarly, Chen et al. [39] introduced a 3D detection model that utilizes 2D cylindrical projection images from point clouds along with bird-eye-view (BEV) and RGB images.

In this work, we adopted an approach that involves converting point clouds into 2D depth maps through cylindrical projection for ease of data fusion with RGB images.

III. Method

We propose a video prediction model that provides users with delay-free video in a teleoperation system with network delays between the user and the remote robot, as depicted in Fig. 1. The proposed model utilizes delayed data in form of video, point cloud, and robot status received from the remote robot along with the user's command to predict current video frames. The user receives these predicted video frames as visual feedback on their commands.

In this study, a timestep represents a duration of $1/30$ seconds, aligning with the sampling rate of the video data. The time point t is defined based on the timestamp of the current time. Specifically, I_{t-D}, P_{t-D}, and R_{t-D} indicate the video frame, point cloud, and robot status at time $(t-D)$, respectively, which are received at the control station with a transmission delay, arriving at the current time (t). (Denote D is Delayed timestep) In contrast, C_{t-D+1} to C_t represent the commands sent by the user during the delay period, reflecting the user's inputs at the current moment. Therefore, these command data are accessible during both training and inference. In addition, we refer to the frames generated by our model as predicted current frames ($\hat{I}_{t-D+1} \sim \hat{I}_t$).

In the proposed model, as depicted in Fig. 2, three data encoders are used to extract data features(z^I, z^P, z^R) from input data, including video frames ($I_{t-D-2} \sim I_{t-D}$), point clouds (P_{t-D-l}), robot statuses ($R_{t-D-14} \sim R_{t-D}$), and commands ($C_{t-D} \sim C_{t-1}$). Subsequently, the video transformer uses these features to predict the features of current video frames ($\hat{z}_{t-D+1} \sim \hat{z}_t$), which are then decoded into RGB images to produce predicted video frames ($\hat{I}_{t-D+1} \sim \hat{I}_t$). Based on the observed communication delay of 300 ms in the cloud-based quadruped robot teleoperation system described in [10], we assumed a maximum delay of 500 ms, resulting in $D = 15$. This implies that the model can effectively handle any delay within this range. (e.g., $D = 9$ for a 300 ms delay). The last frame corresponds to a frame 500 ms in the future, based on the delayed video frame (I_{t-D}), which represents the predicted current frame.

The following content describes each of the data encoders that are configured in our proposed model and explains how the transformed features are used to predict current frames using the video transformer.

A. Data Encoders

To extract features from the input data in the proposed model, we constructed three data encoders: Image Encoder, Point Cloud Encoder, and Robot-Command Encoder. Each encoder is pre-trained. The image decoder, which is responsible for converting the features of the current frames predicted by the video transformer into RGB images, is also pre-trained along with the image encoder.

1) Image Encoder & Decoder: We adopted the residual neural network (ResNet [40]) based encoder-decoder from [41] for both the image encoder and decoder. The image encoder and decoder are both pre-trained in an autoencoder structure to convert the input RGB image ($I \in R^{N\times N\times 3}$) into image features ($z^I \in R^{8\times 8\times 512}$) and then reconstruct them back into RGB images ($\hat{I} \in R^{8\times 8\times 3}$). The objective loss for pre-training the image encoder and decoder is shown in (1):

$$\mathcal{L}^I_{rec}\left(I,\hat{I}\right)=\mathcal{L}_{mse}\left(I,\hat{I}\right)+\lambda_{vgg}\mathcal{L}_{vgg}\left(I,\hat{I}\right)+\lambda_{adv}\mathcal{L}_{adv}\left(\hat{I}\right) \tag{1}$$

where L_{mse} represents the mean square error loss, L_{vgg} calculates the perceptual loss [42] in the feature space using the VGG16 network [43], and L_{adv}, which uses the Patch Discriminator introduced in [41], utilizes the least-square generative adversarial network (LSGAN) loss [44] to enhance the reality of the images. The ratios of each loss, which are represented by λ_{vgg} and λ_{adv}, are hyperparameters and set to 0.1 and 0.01, respectively.

The pre-trained image encoder and decoder, within the proposed model, respectively encode the video frame ($I_{t-D-2} \sim I_{t-D}$) into image features ($z^I_{t-D-2} \sim z^I_{t-D}$) and decode the features of the predicted current frames ($\hat{z}^I_{t-D+1} \sim \hat{z}^I_t$) into RGB images ($\hat{I}_{t-D+1} \sim \hat{I}_t$).

2) Point Cloud Encoder: The point cloud is used to predict scenes that are outside the camera's view frame during the robot's rotation. To bridge the modality gap between RGB images and the point cloud, we convert the 3D point cloud shown in Fig. 5(a) into a 2D depth image, as shown in Fig. 5(c) using cylindrical projection [38], similar to [39]. Considering the robot's rotation speed listed in Table I, only points within a 120-degree range are used, thus resulting in a size of 1200×16 for the transformed 2D depth image. The point cloud encoder, which is based on 2D CNN, is pre-trained using an autoencoder structure that is similar to the image encoder.

The point cloud autoencoder, which is used to pre-train the point cloud encoder, encodes the input point cloud (P) into point cloud features ($z^P \in R^{8\times 8\times 512}$) and then reconstructs it back into a point cloud ($\hat{P}$) with the same size as the input. The objective loss for pre-training is shown in (2), where L_{mse} and L_{L1} denote the mean square error loss and L1 loss, respectively, between the input point cloud (P) and the reconstructed point cloud ($\hat{P}$).

$$\mathcal{L}^P_{rec}\left(P,\hat{P}\right)=\mathcal{L}_{mse}\left(P,\hat{P}\right)+\mathcal{L}_{L1}\left(P,\hat{P}\right) \tag{2}$$

In our proposed model, the point cloud encoder converts the input point cloud (P_{t-D-l}) into point cloud features (z^P_{t-D-l}). Due to the difference in sampling rates between the video and point cloud, there is a temporal inconsistency (Video:30Hz, Point Cloud:10Hz). To address this, we use the most recent point cloud data along with its updated timestamp $t-D-l$, where l is defined as time information ($l \in \{0, 1, 2\}$). This time information is used in the positional encoding of the video transformer to align the point cloud data with the video frames.

TABLE I
STATUS INFORMATION OF ROBOT AND OPERATOR'S COMMAND

	Status(dimension)	Max.	Min.	Unit
Robot Status	Base Quaternion (4)	1	-1	-
	Foot Position (12)	0.08	-0.08	m
	Foot Velocity (12)	2.9	-2.9	m/s
	Foot Force (4)	200	0	N
	Base Linear Speed (3)	1.5	-1.5	m/s
	Base Angular Speed (1)	1.5	-1.5	rad/s
	Base Height (1)	0.31	0.26	m
Command	Forward Speed (1)	1	-1	m/s
	Side Speed (1)	1	-1	m/s
	Rotate Speed (1)	2	-2	rad/s

3) Robot-Command Encoder: The Robot-Command Encoder provides the robot status feature that is used to predict current frames. This feature involves both delayed robot status feature ($z^R_{t-D-2} \sim z^R_{t-D}$) and predicted current robot status feature ($z^R_{t-D+1} \sim z^R_t$). To achieve this, the Robot-Command Encoder is composed of a vanilla transformer, with the delayed robot status ($R_{t-D-14} \sim R_{t-D}$) and command ($C_{t-D} \sim C_{t-1}$) being input to the transformer encoder and transformer decoder, respectively.

For synchronization with the video frame, the input robot status requires data from time $t-D-2$ to $t-D$. However, considering the periodic locomotion of the robot, a delayed robot status with a length of 0.5-seconds is used to predict current robot status ($\hat{R}_{t-D+1} \sim \hat{R}_t$). The command data spans from time $t-D$ to $t-1$, which correspond to the time of the video frame to be predicted.

The robot-command encoder is also pre-trained using the objective loss (3), which consists of the mean square error loss L_{mse} and the L1 loss L_{L1} computed between the predicted robot status ($\hat{R}$) and the ground truth robot status (R).

$$\mathcal{L}^R_{rec}\left(R,\hat{R}\right)=\mathcal{L}_{mse}\left(R,\hat{R}\right)+\mathcal{L}_{L1}\left(R,\hat{R}\right) \tag{3}$$

In the overall proposed model, the pre-trained robot-command outputs robot status features ($z^R_{t-D-2} \sim z^R_{t-D}$) and predicted robot status features ($z^R_{t-D+1} \sim z^R_t$). These features are then matched to the size of image and point cloud features before being passed to the video transformer.

B. Video Transformer

The video transformer in the proposed model predicts current video frames using the features extracted from each encoder. To

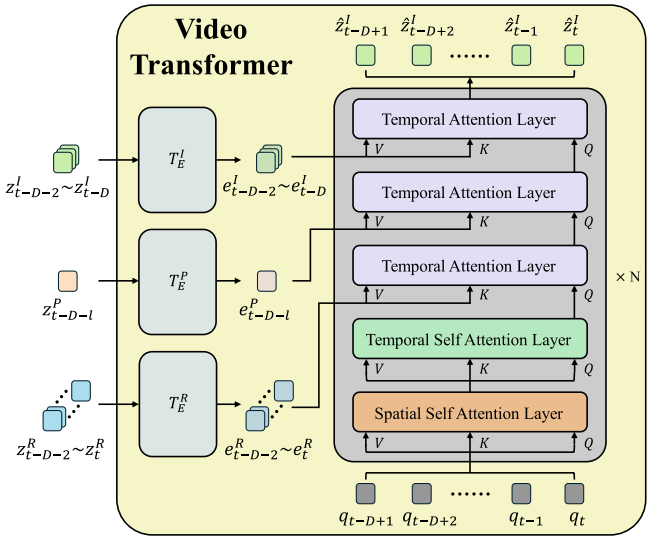

Fig. 3. Architecture of video transformer; $\boldsymbol{T_E}$ represents the Transformer Encoder, and the right gray block represents a Transformer Decoder Block. In the Transformer Decoder, the frame queries ($\boldsymbol{q_{t-D+1} \sim q_t}$) are processed through cross-attention with the encoder output features ($\boldsymbol{e^I_{t-D-2} \sim e^I_{t-D}, e^P_{t-D-l}, e^R_{t-D+1} \sim e^R_t}$), resulting in the output of features for future frames ($\boldsymbol{\hat{z}^R_{t-D+1} \sim \hat{z}^R_t}$).

achieve this, we modified the video prediction model VPTR-NAR, which has demonstrated state-of-the-art performance in video-to-video prediction. Given that we utilize three features (z^I, z^P, z^R) instead of solely relying on past video frame, we constructed three transformer encoders and modified the transformer decoder to handle three encoded features, as shown in Fig. 3.

1) Video Transformer Encoder: The Video Transformer encoder employs the VPTR-NAR encoder without modification. However, to encode the three input features, we constructed three transformer encoders: one for image data, one for point cloud data, and one for robot status features, and these encoders are respectively denoted as T^I_E, T^P_E, and T^R_E. There are slight differences in the positional encoding methods used by these encoders. Unlike the VPTR-NAR encoder, which encodes only past video frames, the robot status feature contains current timestamps. Thus, in T^R_E, positional encoding is applied from time $t-D-2$ to t. Moreover, T^P_E applies positional encoding based on the time information l received. Other mechanisms within the transformer encoders remain consistent with VPTR-NAR, and all encoded features are forwarded to the transformer decoder.

2) Video Transformer Decoder: The Video Transformer decoder differs from the VPTR-NAR decoder in that it receives three encoded features from the encoders. Therefore, we reconfigured the temporal attention layer into three layers to perform cross-attention with each encoder's output feature. The temporal attention layer is organized in the order of robot status, point cloud, and image, as shown in Fig. 3, and it processes frame queries ($q_{t-D+1} \sim q_t$) after passing through spatial and temporal self-attention layers. Frame queries ($q_{t-D+1} \sim q_t$) are initialized to zero, as is done in VPTR-NAR [17]. The initialized frame query passes through the decoder block and is then used as the input frame query for the next decoder block. Finally, the output of the last decoder block is features of current frame, and then these are passed to image decoder for decoding to RGB image. The positional encoding applied in the video transformer decoder follows the same method as in the video transformer encoder.

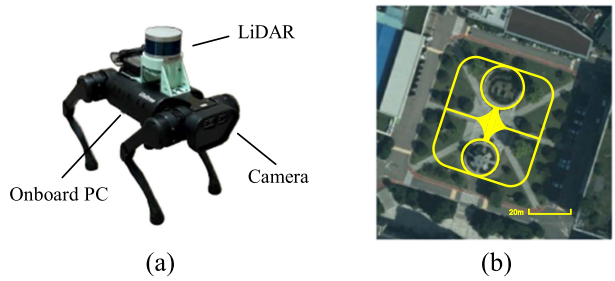

Fig. 4. (a) Is a quadruped robot used to acquire the dataset. The yellow line in (b) is the area navigated by the robot to acquire the dataset.

We designed the video transformer encoder and decoder with 2 and 6 blocks, respectively, while the remaining steps are the same as those applied for VPTR-NAR.

The encoders for image, point cloud, and robot status, as well as the image decoder, are all pre-trained. Consequently, the proposed model is trained using the objective loss of (4), with only the weights of the video transformer being updated during training. Eq. (4) is applied between the predicted video frames ($\hat{I}_{t-D+1} \sim \hat{I}_t$) and ground truth video frames ($I_{t-D+1} \sim I_t$).

$$\mathcal{L}_{Pred} = \sum_{T=t-D+1}^{t} \begin{array}{c} \mathcal{L}_{L1}\left(I_T, \hat{I}_T\right) + \mathcal{L}_{mse}\left(I_T, \hat{I}_T\right) \\ + \lambda_{vgg}\mathcal{L}_{vgg}\left(I_T, \hat{I}_T\right) + \lambda_{adv}\mathcal{L}_{adv}\left(\hat{I}_T\right) \end{array} \tag{4}$$

Here, L_{mse} and L_{L1} represent the mean square error loss and L1 loss, respectively, while L_{vgg} and L_{adv} correspond to the same objective losses in (1). The values of λ_{vgg} and λ_{adv} were set to 0.1 and 0.01, respectively, as hyperparameters.

IV. Experimental Setup and Dataset

To evaluate our method, we acquired navigation data of a quadruped robot. The quadruped robot used in the experiments is the Unitree A1 robot, as depicted in Fig. 4(a), which was equipped with an Intel RealSense D435 camera and a VLP-16 LiDAR for the acquisition of video and point cloud data. The camera captures RGB images with a resolution of 640 × 480 at 30 Hz, while the LiDAR rotates at 600 rpm, collecting approximately 25000 points per frame at 10 Hz. The robot status and commands are recorded at 50 Hz, as presented in Table I. All data is collected in real-time without any delay while the robot is being controlled in the field. Using this robot setup, we randomly traversed the area indicated in yellow in Fig. 4(b) to acquire the data.

For training and testing of the proposed model using the acquired data, we performed preprocessing to construct the dataset. We resized and cropped the collected images to align the field of view (FOV) of the camera with that of the LiDAR. The camera has a vertical FOV of 42 degrees, while the LiDAR's vertical

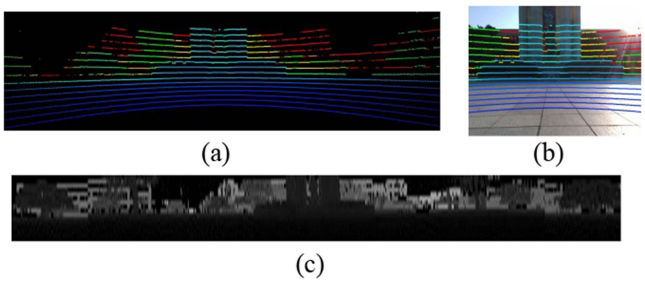

Fig. 5. Illustration of point cloud representation and reference image; (a) is a 3D point representation, (b) is a RGB image of camera at the same view, (c) is a 2D depth image converted through cylindrical projection of (a).

FOV is limited to 30 degrees. As a result, the images contained areas outside the LiDAR's FOV that could not be predicted using the point cloud data. This can be easily understood by referring to Fig. 5(b). To ensure the dataset was appropriate for our experiment, which relies on point cloud information to predict unseen areas in the image, we resized the images to 160×120 and cropped the central 64×64 regions. This adjustment allowed the dataset to better represent the overlapping FOV between the two sensors. In addition, for the point cloud, considering the rotation speed listed in Table I, we use only points within a range of 120 degrees to generate the projected 2D depth images, ultimately resulting in a dataset of size 1200×16.

The constructed dataset comprises a total of 308052 samples; of these, 275661 samples are used for training, while the remaining 32392 samples are used for testing.

In our experiments, our model's average inference time is measured to 28.3ms, with this computational cost measured using an NVIDIA RTX A6000. This time is measured as the duration required to generate 15 features and decode one of them into an RGB image.

V. Result

Our model was compared with existing video prediction methods to validate the proper utilization of the proposed command and point cloud. Comparisons were made with VPTR-NAR [17] and PredRNN-V2 [25], which are reference models in video prediction. These models only utilize past video frames as input, which allows us to observe the differences arising from our model's use of commands to predict current frames. Subsequently, an ablation study was conducted to verify whether our proposed model effectively utilizes command and point cloud information. This study aimed to assess the specific impact of incorporating command and point cloud data on prediction.

A. Comparison With Other Models

To evaluate the effectiveness of our model, we compared it with VP TR-NAR [17] and PredRNN-V2 [25]. Both models were trained to predict 15 current frames using three past frames from our dataset. The evaluation metrics that were considered for comparison included root mean square error (RMSE), peak signal noise ratio (PSNR), structural similarity index measure (SSIM) [45] and learned perceptual image patch similarity (LPIPS) [46], as shown in Table II. RMSE and PSNR measure prediction accuracy and video quality, with lower RMSE and higher PSNR indicating better alignment and less distortion. SSIM and LPIPS assess perceptual similarity, where higher SSIM and lower LPIPS values indicate better preservation of visual structures and closer match to the ground truth.

TABLE II
Comparison of Qualitative Result With Other Models

	RMSE(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Proposed	**0.0132**	**20.5956**	**0.6117**	**0.0684**
VPTR-NAR [17]	0.0188	18.8938	0.5369	0.2336
PredRNN-V2 [25]	0.0207	18.4134	0.4954	0.1138

Bolded entities represent the best performance in each metric.

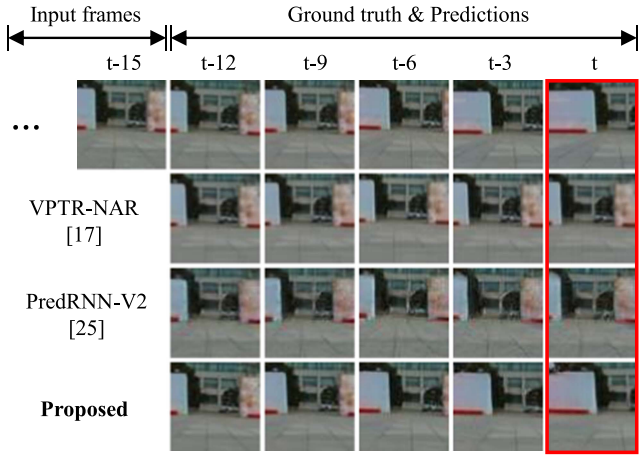

Fig. 6. Qualitative result of each model. This result is in the form of a prediction for a scenario where the robot turns left after moving straight forward.

The proposed model outperformed the baseline models in all metrics. This result is also visually supported by the quantitative results depicted in Fig. 6. For instance, when provided with video frames of a robot moving straight, both VPTR-NAR and PredRNN-V2 could only predict current frames depicting the robot continuing straight. By contrast, the proposed model leveraged the input commands along with the video frames to make more complex predictions about the robot's response to commands, such as turning, as illustrated in Fig. 6. These comparisons demonstrate that our model effectively utilizes commands to predict current frames, thereby distinguishing it from conventional video prediction models.

B. Ablation Study

To verify whether our proposed model was overfitting the dataset used for training and the environment where the dataset was acquired, we conducted an ablation study on command and point cloud data.

1) Response About Command: We tested the model's responsiveness to commands by inputting the same video frame under different command scenarios: forward, left, and right.

As depicted in Fig. 7 *forward*, the proposed model predicts current frames in which the robot goes straight by reflecting the

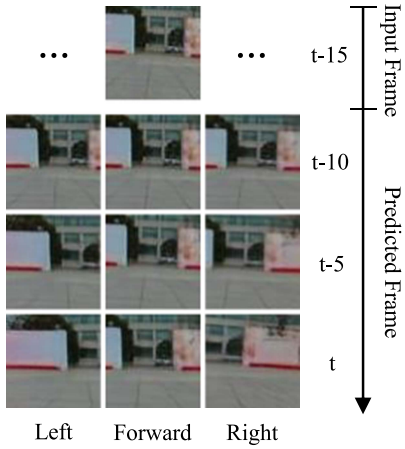

Fig. 7. Qualitative result of command data manipulation. Each prediction is the result of using the same input frame and different commands (turning *left*, moving *forward* and turning *right*).

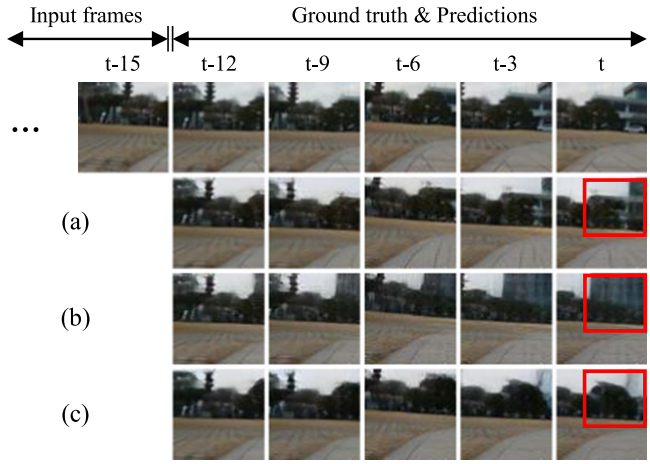

Fig. 8. Qualitative result of point cloud data manipulation. The point cloud used in each prediction is as follows: (a) the *original data*, (b) the *maximum* values and (c) the *minimum* values of the LiDAR measurement range.

input straight command. However, when left and right commands were given, the model successfully predicted current frames in which the robot rotated in the commanded direction, as shown in Fig. 7 *left, right*. This demonstrates our proposed model's capability to predict changes in current frames based on the input command without overfitting to the dataset.

2) Response About Point Cloud: We also verified the model's use of point cloud data to predict areas that were not visible in the input video frame, particularly when the robot was rotating. We compared the predictions that manipulated point cloud data inputted into the proposed model.

The point cloud data used in the experiments consists of the *original* point cloud data, data filled with the *maximum* and *minimum* (zero) values of the LiDAR measurement range. In the process of 2D projection of the point cloud, pixels where no points are measured are set to zero. Thus, pixels filled with zeros represent regions that are too close to the robot or where point measurement is impossible, such as the sky.

The quantitative results are depicted in Fig. 8. First, the proposed model predicts the newly appearing building when the original point cloud was used, as shown in Fig. 8(a). However, in the case that the point cloud data filled with *maximum* values was used, the proposed model incorrectly predicted the building, as highlighted in the red box of Fig. 8(b). It can also be observed from the $t-12$ time step that the model predicts the entire newly appearing right area as a building. By contrast, in the case that the point cloud data filled with *minimum* was used, the model failed to predict the building in the newly appearing area, as shown in Fig. 8(c). Our model instead only predicted sky here, as highlighted in the red box of Fig. 8(c), because a value of zero means that nothing was measured in that area. These results demonstrate that the proposed model can successfully predict scenes outside the camera's view frame.

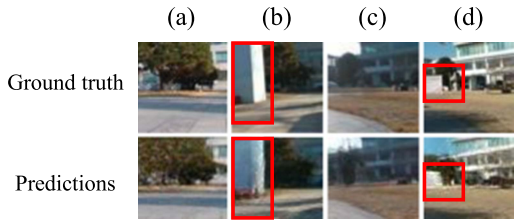

Fig. 9. Qualitative result of varied scenarios. (a) and (b) represent the same location, as do (c) and (d). While each pair depicts the same location, the model successfully generated the newly introduced structures when they were present.

In addition, we conducted experiments to test the model's response to newly introduced structures in the same location. As shown in the figures, the generated images successfully accounted for the presence of new structures. Fig. 9(a) and (b), as well as Fig. 9(c) and (d), depict the same location; however, as highlighted in the red box, new objects were generated. Although we did not experiment in unseen environments, these results suggest that collecting data from a wider variety of locations could enable the model to adapt to more diverse environments in the current.

VI. Conclusion

In this study, we propose a delay-free video generation model that predicts current frames based on user commands. The proposed model, which is modified from VPTR-NAR, can predict current frames by integrating delayed video frames, point cloud, and robot status from remote robot with real-time user commands. In particular, our model predicts areas outside the video frame by converting point cloud data into 2D depth images through cylindrical projection. Through experiments, we demonstrated that the proposed model effectively utilizes both input commands and point cloud data to predict current frames.

In future research, we will enhance our method, which currently assumes a communication delay up to 500ms, to make it applicable to teleoperation systems with longer communication delays, such as the teleoperation of robots in space.

References

[1] R. W. Xu, K. C. Hsieh, U. H. Chan, H. U. Cheang, W. K. Shi, and C. T. Hon, "Analytical review on developing progress of the quadruped robot industry and gaits research," in *Proc. 8th Int. Conf. Automat., Robot. Appl.*, 2022, pp. 1–8.

[2] H. Azpúrua et al., "A survey on the autonomous exploration of confined subterranean spaces: Perspectives from real-word and industrial robotic deployments," *Robot. Auton. Syst.*, vol. 160, 2023, Art. no. 104304.
[3] S. Halder, K. Afsari, E. Chiou, R. Patrick, and K. A. Hamed, "Construction inspection & monitoring with quadruped robots in future human-robot teaming: A preliminary study," *J. Building Eng.*, vol. 65, 2023, Art. no. 105814.
[4] C. Cruz Ulloa, J. del Cerro, and A. Barrientos, "Mixed-reality for quadruped-robotic guidance in SAR tasks," *J. Comput. Des. Eng.*, vol. 10, no. 4, pp. 1479–1489, 2023.
[5] F. Angelini et al., "Robotic monitoring of habitats: The natural intelligence approach," *IEEE Access*, vol. 11, pp. 72575–72591, 2023.
[6] S. Halder, K. Afsari, and A. Akanmu, "A robotic cyber-physical system for automated reality capture and visualization in construction progress monitoring," 2024, *arXiv:2402.07034*.
[7] D. Zhang, W. Si, W. Fan, Y. Guan, and C. Yang, "From teleoperation to autonomous robot-assisted microsurgery: A survey," *Mach. Intell. Res.*, vol. 19, no. 4, pp. 288–306, 2022.
[8] S. Gnatzig, F. Chucholowski, T. Tang, and M. Lienkamp, "A system design for teleoperated road vehicles," in *Proc.10th Int. Conf. Inform. Control, Automat. Robot.*, 2013 , pp. 231–238.
[9] T. Zhang, "Toward automated vehicle teleoperation: Vision, opportunities, and challenges," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11347–11354, Dec. 2020.
[10] S. Halder, K. Afsari, J. Serdakowski, S. DeVito, M. Ensafi, and W. Thabet, "Real-time and remote construction progress monitoring with a quadruped robot using augmented reality," *Buildings*, vol. 12, no. 11, 2022, Art. no. 2027.
[11] C. Zhou, C. Peers, Y. Wan, R. Richardson, and D. Kanoulas, "Teleman: Teleoperation for legged robot loco-manipulation using wearable IMU-based motion capture," 2022, *arXiv:2209.10314*.
[12] C. Cruz Ulloa, D. Domínguez, J. Del Cerro, and A. Barrientos, "A mixed-reality tele-operation method for high-level control of a legged-manipulator robot," *Sensors*, vol. 22, no. 21, 2022, Art. no. 8146.
[13] E. Kim, V. Peysakhovich, and R. N. Roy, "Impact of communication delay and temporal sensitivity on perceived workload and teleoperation performance," in *Proc. ACM Symp. Appl. Percep.*, 2021, Art. no. 4.
[14] B. Wu, S. Nair, R. Martin-Martin, L. Fei-Fei, and C. Finn, "Greedy hierarchical variational autoencoders for large-scale video prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2318–2328.
[15] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 802–810.
[16] Y. Wang, M. Long, J. Wang, Z. Gao, and P. S. Yu, "PredDRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMS," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 879–888.
[17] X. Ye and G.-A. Bilodeau, "VPTR: Efficient transformers for video prediction," in *Proc. 26th Int. Conf. Pattern Recognit.*, 2022, pp. 3492–3499.
[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[19] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Neural Inf. Process. Syst.*, Jun. 2017, vol. 30>, pp. 5998–6008.
[20] D.-K. Ko, D.-H. Lee, and S.-C. Lim, "Continuous image generation from low-update-rate images and physical sensors through a conditional GAN for robot teleoperation," *IEEE Trans. Ind. Inform.*, vol. 17, no. 3, pp. 1978–1986, Mar. 2021.
[21] K.-I. Yoon, D.-K. Ko, and S.-C. Lim, "Real-time video prediction using GANs with guidance information for time-delayed robot teleoperation," *Int. J. Control, Automat. Syst.*, vol. 21, no. 7, pp. 2387–2397, 2023.
[22] S. Oprea et al., "A review on deep learning techniques for video prediction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 6, pp. 2806–2826, Jun. 2022.
[23] S. Lee, H. G. Kim, D. H. Choi, H.-I. Kim, and Y. M. Ro, "Video prediction recalling long-term motion context via memory alignment learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3053–3062.
[24] B. Liu, Y. Chen, S. Liu, and H.-S. Kim, "Deep learning in latent space for video prediction and compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 701–710.
[25] Y. Wang, Z. Gao, M. Long, J. Wang, and S. Y. Philip, "PredRNN++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning," in *Proc.35th Int. Conf. Mach. Learn.*, 2018, pp. 5123–5132.
[26] W. Yan, Y. Zhang, P. Abbeel, and A. Srinivas, "Videogpt: Video generation using VQ-VAE and transformers," 2021, *arXiv:2104.10157*.
[27] Z. Liu et al., "ConvTransformer: A convolutional transformer network for video frame synthesis," 2020, *arXiv:2011.10185*.
[28] Y. Yuan et al., "HRFormer: High-resolution transformer for dense prediction," in *Proc. NeurIPS*, 2021, pp. 1–13.
[29] A. Ramesh et al., "Zero-shot text-to-image generation," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 8821–8831.
[30] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 12868–12878.
[31] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10674–10685.
[32] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1325–1339, Jul. 2014.
[33] V. L. Guen and N. Thome, "Disentangling physical dynamics from unknown factors for unsupervised video prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11471–11481.
[34] M. Seo, H. Lee, D. Kim, and J. Seo, "Implicit stacked autoregressive model for video prediction," 2023, *arXiv:2303.07849*.
[35] K.-W. Lee, D.-K. Ko, Y.-J. Kim, J.-H. Ryu, and S.-C. Lim, "Latency-free driving scene prediction for on-road teledriving with future-image-generation," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 11, pp. 16676–16686, Nov. 2024.
[36] X. Zhao, P. Sun, Z. Xu, H. Min, and H. Yu, "Fusion of 3D LIDAR and camera data for object detection in autonomous vehicle applications," *IEEE Sensors J.*, vol. 20, no. 9, pp. 4901–4913, May 2020.
[37] Y. Cui et al., "Deep learning for image and point cloud fusion in autonomous driving: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 722–739, Feb. 2022.
[38] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," in *Proc. Robot.: Sci. Syst.*, 2016, pp. 1–8.
[39] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6526–6534.
[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[41] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5967–5976.
[42] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. 14th Eur. Conf. Comput. Vis.*, Amsterdam, The Netherlands, 2016, pp. 694–711.
[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[44] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2813–2821.
[45] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[46] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 586–595.