Received 4 November 2022, accepted 2 December 2022, date of publication 6 December 2022, date of current version 12 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3227093

RESEARCH ARTICLE

CATtalk: An IoT-Based Interactive Art Development Platform

YI-BING LIN[1,2,3,4,5,6], (Fellow, IEEE), HELIN LUO[5,7], (Member, IEEE), AND CHEN-CHI LIAO[1]
[1]Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan
[2]College of Humanities and Sciences, China Medical University, Taichung 406, Taiwan
[3]Miin Wu School of Computing, National Cheng Kung University, Tainan 701, Taiwan
[4]Department of Computer Science and Information Engineering, Asia University, Taichung 413, Taiwan
[5]Research Center for Information Technology Innovation, Academia Sinica, Taipei 115, Taiwan
[6]College of Artificial Intelligence, National Yang Ming Chiao Tung University, Tainan 711, Taiwan
[7]Graduate Institute of Animation and Film Art, Tainan National University of the Arts, Tainan 720, Taiwan

Corresponding author: Yi-Bing Lin (liny@nctu.edu.tw)

This work was supported in part by the Ministry of Science and Technology (MOST) under Grant 110-2622-8-A49-005 and Grant 111-2420-H-369-001.

ABSTRACT Interactive art has been significant advanced by the Internet of Things (IoT) and cyber physical interaction technologies, which enables the participants to engage with the art devices. Several tools and platforms have been proposed to create the art devices. However, the interactive artworks are typically developed with these art devices in ad hoc approaches, and the artists need to spend significant programming efforts to integrate the art devices. This paper proposes CATtalk, a platform to create and maintain interactive artworks. The novel idea is to treat all art devices in an interactive artwork as IoT devices that can be transparently reused by reconfiguration in CATtalk. Therefore, the artworks developed independently by individual artists can be quickly integrated to create new interactive applications. Through CATtalk's no-code and low-code mechanisms, the artists can manipulate CATtalk with little or no programing efforts. CATtalk offers a built-in mechanism so that any person in the audience can play with an interactive artwork with his/her smartphone. We also conduct analytic analysis, simulation and measurements to ensure that the interactive art performance in cross-country remote stages are not affected by the communications delays. In our measurements, the average local and remote communication delays are about 0.01 and 0.05 seconds, respectively. If the art performance is designed such that the average delay between two actions of a local (remote) performer is longer than 0.1 seconds, then the probability of out-of-sequence actions is less than 0.01%. That is, the local dancer should perform slower than the remote dancer. Such delay analysis for remote interactive art performance has not been conducted in the literature.

INDEX TERMS Arts and humanities, interaction paradigms, interactive art performance, Internet of Things (IoT), low code, no code.

I. INTRODUCTION

Due to the advancement of the Internet of Things (IoT) and cyber physical interaction technologies, the user experience in interactive art has become more and more immersive and entertaining. These new technologies enable the players (either art performers or the audience) to engage with the physical and the cyber devices (e.g., art devices or games) with both their bodies and minds.

The associate editor coordinating the review of this manuscript and approving it for publication was Agostino Forestiero.

For example, a traditional museum reproduces artworks like paintings, photographs, or films as postcards, posters, books, and DVDs. With the interactive art applications, a modern museum allows the visitors to interact with the artworks and then sells the results of the interaction to the visitors [1]. As another example, many art performance shows enable the dancers to interact with the robots, the animated objects and others [2].

In [3], the authors explored interference and consultation in virtual public space for intermedia art, where human in the physical world interacts with those in the virtual world called "metaverse". The authors showed how intermedia migrates from the physical world to the virtual world as social background changes.

To support human and art device interaction, Information and Communication Technology (ICT) based platforms are essential to accommodate features of art devices including Augmented Reality (AR) and Virtual Reality (VR). The rapid development of consumer-ready hardware for AR/VR has resulted in many platforms for affluent interactive experience. These platforms such as Jorjin smart glasses [4] and HTC Vive offer new possibilities for device-specific interaction, but often with an essential lack of standardization for devices and applications. A systematic review [5] provides a foundation for understanding the directions for interaction studies in immersive environments.

In this paper, we propose CATtalk, a no-code and low-code platform to create and maintain interactive artworks. CATtalk is derived from an IoT application development platform called IoTtalk [6]. The novel idea of CATtalk is to treat all art devices in an interactive artwork as IoT devices. The IoTtalk engine provides telecom-grade communication quality. Therefore, CATtalk is an effective platform for interact art performance involving the performance stages in different countries at the same time. We highlight the importance of the proposed research study as follows:

- The art devices are independent of each other, which simplifies the debug process of the applications. They can be reused by reconfiguration through the IoTtalk engine to be elaborate in Section V. Therefore, the artworks developed independently by individual artists can be integrated to create new interactive applications.
- Through CATtalk's no-code and low-code mechanisms, the artists can manipulate CATtalk with little or no programing efforts. To our knowledge, no platforms like CATtalk have been reported in the literature.
- CATtalk has a built-in mechanism so that when a smartphone scans a CATtalk generated QR code, the smartphone becomes an art device. Therefore, any person in the audience can participate in an interactive artwork with his/her smartphone.

The paper is organized as follows. Section II reviews the related works; Section III proposes the CATtalk architecture; Sections IV-VI describe the no-code and low-code approach for creating the art devices and applications in CATtalk. Sections VII and VIII demonstrate the scenarios for artworks to interact with the audience and the dancers, respectively. In Section IX, we conduct analytic analysis, simulation and measurements to ensure that the interactive art performance in cross-country remote stages are not affected by the communications delays.

II. RELATED WORKS

In [7], the authors investigated the impact of engaging brain-computer interface (BCI)-driven mechanics on player's in-game immersion, and provided qualitative insights on psycho-cognitive effects of using BCIs. An HTC Vive Pro was connected to a VR-dedicated PC. The authors also used an Emotiv Insight 5-channel headset to collect the EEG data. The study found that controlling the game with mental states enhances the in-game immersion and attracts the player's engagement. We will show that the experiments in this study can be easily implemented in CATtalk. In [7], the EEG data are used for analysis only. In CATtalk, the EEG data can serve as real-time feedback to control the VR game. This research direction has not been explored in [7].

The study in [8] used a floor-projector to generate animal's footprints. The Kinect sensors capture the behaviors of the children play with the footprints. The footprints then respond to the children's movements. The experiments showed that this system encourages the imagination of children. The proposed system can be easily implemented in CATtalk as described in Section V.

The study in [9] described an interactive audiovisual synthesis application by analyzing visual changes in the image sequences and map the results to musical scales. The proposed Interactive Visual Audio (IVA) system reads the camera-captured images, extracts the visual features and then maps them to musical sounds. This work can be integrated in a CATtalk project called DanceTalk to enhance the features of the interactive art performance. The details are provided in Section VIII.

In the reverse direction, the study in [10] generated a sequence of three-dimensional human dance poses based on a given music. The authors developed a music feature encoder to generate a set of audio feature vectors, and then used a music genre classifier to estimate the genre of the music. The results are then used to drive 3D-dancing move. This music-to-dance mechanism can be easily accommodated in CATtalk, which will be briefly described in Section VI. A similar music-to-dance mechanism was proposed by the study in [11], which uses live music to drive a legged robot to dance in a diverse fashion. An onboard microphone extracts the beat from the music in real-time, and a dance choreography is created at every new beat by a user-generated dance motion library.

In [1], the authors proposed YUKINKO that allows a visitor of the Museum of Contemporary Art Tokyo to use Apple's iPhone to interact with a picture through projector-camera (ProCam). YUKINKO can be transparently accommodated in CATtalk with extension of new features. The details are provided in Section VII.

In [12] the authors proposed a lightweight social VR platform that allows multiple users in shared virtual environments (e.g., a live TV show) to socially interact and collaborate though the broadband media contents. Since traditional IoT targets on narrowband applications, CATtalk can serve as intelligent control for the VR platform proposed in [12]. Details of the integration can be found in [13].

The study in [2] proposed the exploitation of feminine gender performance in technology. The authors enabled a wearable robotic device to creates an onstage cyborg character.

The robot includes flex sensors triggering LEDs and foldable wings powered by breath or button, attached to the performer by a two-piece vinyl ensemble. The study did not described the mechanism for the performer to control the robot device. This issue can be nicely addressed by CATtalk by providing the robot Application Programming Interface (API) through the procedures described in Section IV.

VR-All-Art [14], Artsy, Artspace, and Paddle8 [15] are platforms that give the artists freedom to create exhibitions in VR, and sell artworks without the limitation of time, space and geography. These platforms do not provide tools for the artists to create their artworks.

Many tools allow the artists to create digital art devices [16], including SculptGL, Random Art Prompt Generator for Kids, Mondrimat, ManadalaGaba, Bomomo, GIPHY, JacksonPollack.org, KRITA, Make Beliefs Comix, Pixilart, Sketchpad, Sumo Paint, Toy Theater and Vectr. These tools do not aim to integrate various art devices like CATtalk. On the other hand, the art devices created by these tools can be easily accommodated in CATtalk as described in Sections IV and VI.

In the above related studies, it is not clear if the software applications developed for their art works are sustainable (for debugging and maintenance). CATtalk allows their art devices to be connected for new applications with low costs. We also investigate the communication delays for the art devices by conducting analytic analysis, simulation and measurements. In this way, we can adjust the interactions of the art performance in cross-country remote stages to ensure that they are not affected by the communications delays. Such remote delay studies for interactive art performance has not been found in the literature.

III. THE CATTALK ARCHITECTURE

This section proposes the CATtalk architecture. We use PuppetTalk (a remote puppet show) [17] as an example to demonstrate how CATtalk works. In this project, a puppeteer uses a smart motion-capture glove (Figure 1 (1)) [18] to control a robot puppet (Figure 1 (3)) and an animated puppet (Figure 1 (4)) through the CATtalk server (Figure 1 (2)). The glove captures the movements of over 20 joints of the fingers and the wrist, digitizes the movements, and translates the digitized data to control the movement of the robotic puppet. The robot puppet is controlled by 9 motors.

The CATtalk architecture consists of the device domain (Figure 2 (1)) and the network domain (Figure 2 (2)). The device domain defines the art devices such as the smart glove and the robot puppet. An art device is installed two software modules. The Sensor & Actuator Application (SA; Figure 2 (3)) implements the driver for the hardware (e.g. the motion sensors of the smart glove and the motors of the robot). The Device Application (DA; Figure 2 (4)) is responsible for the communication with the CATtalk server located in the network domain. The heart of the server is the IoTtalk engine (Figure 2 (5)), an IoT management mechanism responsible for creation and interaction of the art devices [6].

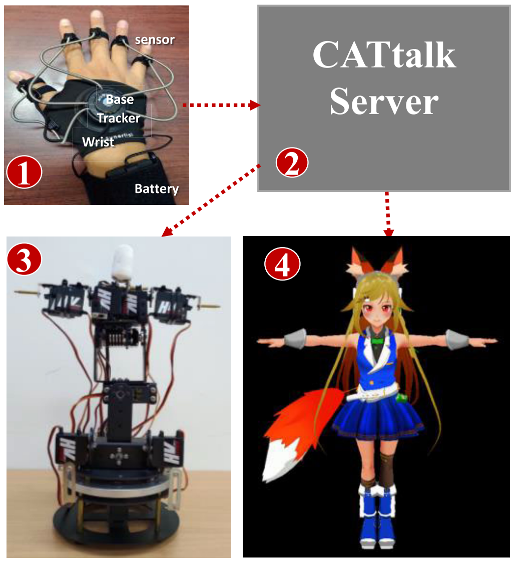


FIGURE 1. PuppetTalk: remote puppet control.

The IoTtalk engine interacts with the DA of an art device through Bluetooth, WiFi, 4G, 5G or Ethernet. The higher-level protocols are Restful or MQTT.

An artist manipulates CATtalk through the Graphical User Interface (GUI; Figure 2 (6)) when an interactive artwork is created in CATtalk as a project called "X-Talk" (Figure 2 (7)). Based on the concept of "device model", X-Talk implements the interaction of the art devices. A device model is an abstract set of the art devices that have similar input and output behaviors (e.g., robot puppet and animated puppet in Figure 1). In the network domain, X-Talk automatically generates the network program connecting the device models to produce the interactive effects designed by the artist. The X-Talk network program is executed by the IoTtalk engine, which connects the art devices to the device models by establishing the communication links (1)-(2), (3)-(2) and (4)-(2) in Figure 1. Details of CATtalk project creation are given in Section V. In a X-Talk, the actions in the interactions (such as the puppeteer's gestures) are stored in the DataTalk subsystem (Figure 2 (8)) and can be edited for future use.

CATtalk provides the ModelGen mechanism (Figure 2 (9)) to create a device model for the corresponding physical art devices. Section IV describes how ModelGen enables the artist to generate the device model with a no-code approach. The created device models are saved in the IoTtalk Database (DB; Figure 2 (10)).

To connect an art device with the corresponding device model in the CATtalk server, the artist needs to install the DA/SA in the art device. The DA/SA can be automatically generated by the DSgen subsystem (Figure 2 (11)). The generated SA/DA are then executed by the art device hardware. More details will be provided in Section VI.

The Authentication, Authorization and Accounting subsystem (AAA; Figure 2 (12)) supports the account management for the artists who use CATtalk. With the AAA, an artist

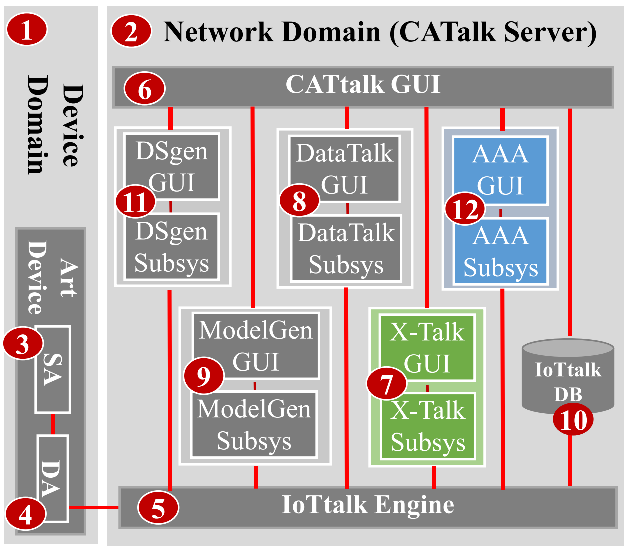


FIGURE 2. The CATtalk architecture.

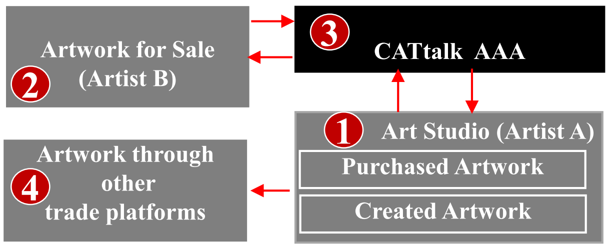


FIGURE 3. A business model based on CATtalk AAA.

can authorize another artist to use his/her art devices or art projects with an appropriate charging procedure. (In the current implementation, the usage is free.) The business model is illustrated in Figure 3. In this model, Artist A (Figure 3 (1)) builds remote puppet control with a smart glove and a robot puppet, and the animated puppet is separately built by Artist B (Figure 3 (2)). Artist A may purchase the animated puppet from Artist B through the CATtalk AAA (Figure 3 (3)). Since both the robot and the animated puppets are developed in CATtalk under the same device model, the animated puppet can be transparently included in Artist A's CATtalk project. Artist A may also sell his/her art applications to other platforms (Figure 3 (4)) such as those described in [15]. The business model of CATtalk is under construction now, which is similar to the one used in Unity [19].

IV. MODELGEN SUBSYSTEM

In CATtalk, an art device is considered as an IoT device, which uses an interface called Input Device Features (IDFs) to send various types of signals to the CATtalk server. Similarly, the device uses the other interface called Output Device Features (ODFs) to receives signals from the CATtalk server. Therefore, an art device can be defined as two sets mathematically. The input set is called the "input device" consisting of the IDFs and the output set is called the "output device"

TABLE 1. MAAPEs A partial list of the IDFs for smartphone and smart glasses.

Smartphone API	Glasses API	IDF	Data type
Accelerometer	Acceleration	Acceleration-I	[float, float, float]
Gyroscope	Gyroscope	Gyroscope-I	[float, float, float]
Magnetic_field	Magnetometer	Magnetometer-I	[float, float, float]
Orientation	Orientation	Orientation-I	[float, float, float]
Brightness	Brightness	Luminance-I	float
Linear_acceleration	Linear Accelerometer	LinearAcceleration-I	[float, float, float]
Gravity	Gravity	Gravity-I	[float, float, float]
Location	Location	Geolocation-I	[float, float]
Keyboard		Keyboard-I	int

TABLE 2. A partial list of the ODFs for smartphone and smart glasses.

Smartphone API	Glasses API	ODF	Data type
Vibrator		Vibration-O	int
Speaker	Speaker	Volume-O	float
	Display Mode	DisplayMode-O	int
Brightness	Brightness	Luminance-O	float
Streaming URL	Streaming media URL	RtspUrl-O	String

consisting of the ODFs. In CATtalk, a smartphone is considered as an art device. The input device of the smartphone includes sensing capabilities such as the microphone, the accelerometer and the gyroscope, and control capabilities such as the keypads. Some sensor capabilities are mapped to the IDFs listed in Table 1, where an IDF is represented by a string followed by "-I". Similarly, the output device of the smartphone includes actuator capabilities such as the display and the speaker. Some actuator capabilities are mapped to the ODFs listed in Table 2, where an ODF is represented by a string followed by "-O". Some device features have both IDF and ODF. For example, brightness of the display is represented by Luminance-I and Luminance-O.

The concept "device model" is defined in CATtalk to group the art devices. The IDFs and the ODFs of an art device are subsets of those of the corresponding device model. For example, the Android smartphones and the Jorjin smart glasses [4] are grouped in the "smartphone" device model. In Tables 1 and 2, the similar capabilities of a smartphone and a smart glasses are defined under the same IDFs and ODFs, respectively. Specifically, the Accelerometer API of the smartphone and the Acceleration API of the smart glasses are mapped to the IDF Acceleration-I with the data type [float, float, float]. The data type of a DF is represented as the DF-parameters. For example, Acceleration-I has three DF-parameters of type "float". In Table 2, the Flash API of the smartphone is mapped to the ODF Flash-O, which is not found in the smart glasses. Similarly, the Display Mode API of the smart glasses is mapped to the ODF DisplayMode-O, which is not found in the smartphone. The Display Mode of the Jorjin glasses allows the artist to switch the glasses to 2D or 3D display mode.

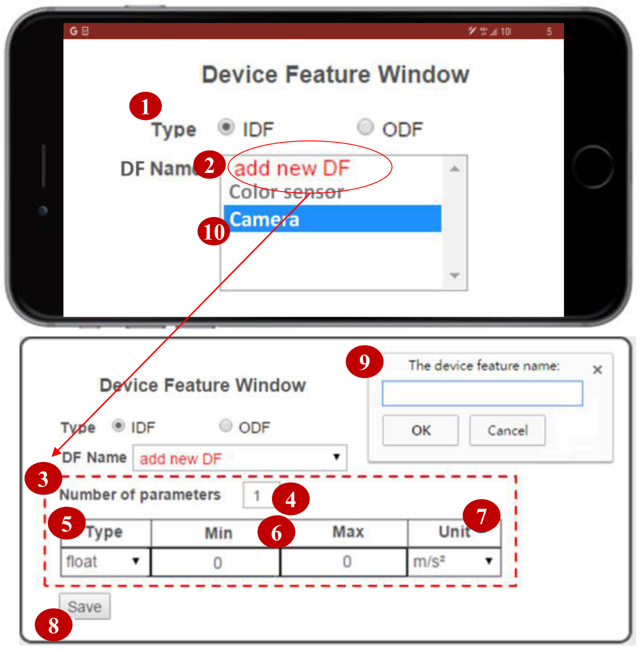


FIGURE 4. Device feature creation.

In CATtalk, an artist uses the ModelGen GUI (Figure 2 (9)) to create the device models without any programming effort. We first describe how the device features are produced, and then elaborate on how a device model is created by grouping the device features. The operation flow for creating device features and models is (6)→(9)→(5)→(10) in Figure 2. The remainder of this section describe the no-code approach for ModelGen. The readers may skip this section if they are not interested in the operation details.

A. DEVICE FEATURE MANAGEMENT

The ModelGen GUI includes a web-based window called "Device Feature Management" to create and manage the IDFs/ODFs. The artist first selects the DF type (Figure 4 (1)), which instructs CATtalk to retrieve all device names of the selected DF type from the IoTtalk DB. Then the Device Feature Management window shows them in the pulldown list "DF Name" (Figure 4 (2)). The default DF type is IDF.

The first item in the list is "add new DF". To create a new DF, the artist clicks this item, and the GUI pops up the DF Parameter Table (Figure 4 (3)). The rows of the table are created based on the number of the DF-parameters (Figure 4 (4)). The default DF-parameter number is one. A DF-parameter is specified by its Type (i.e., the data types such as float, string and so on; see Figure 4 (5)), the Min (minimal) and Max (maximal) values (Figure 4 (6)) and the Unit (e.g., cm, m/s2 and so on; see Figure 4 (7)). For an IDF, the default Min/Max values are automatically assigned by CATtalk. For an ODF, if the Min/Max fields are not filled by the artist, the ODF-parameter takes arbitrary values without any range limits. When the artist clicks the "Save" button (Figure 4 (8)), the window pops up a dialog box for the artist to input the name of the new device feature (Figure 4 (9)). Then CATtalk stores the device feature information into the IoTtalk DB.

The artist can select and edit an existing DF (e.g., Camera; see Figure 4 (10)) from the "DF Name" list. If the artist modifies, for example, the DF-parameter number (Figure 4 (4)), the DF Parameter table will be redrawn based on the new number.

B. DEVICE MODEL MANAGEMENT

An artist creates a device model using the ModelGen web-based GUI called "Device Model Window" (Figure 5 (1)). The Device Model Window shows the "DM Name" pull-down menu (Figure 5 (2)) that allows the artist to select a device model. To create a new device model, the artist selects the "add new DM" in the pull-down menu. The ModelGen GUI pops up the DF module (Figure 5 (3)) and the "Add/Delete DF" module (Figure 5 (4)), which allows the artist to add a DF to this device model. To do so, the artist first selects the DF type from one of the two radio buttons in the "Add/Delete DF" module (e.g., IDF is selected in Figure 5 (5)). Then the module shows all DFs (Figure 5 (6)) of the selected DF type. When the artist selects a DF in the "Add/Delete DF" module, the DF is automatically displayed in the DF module (Figure 5 (3)). After the artist has selected all desired DFs for the device model and clicks the "Save" button (Figure 5 (7)), the GUI pops up a dialog box for the artist to input the name of the new device model (Figure 5 8)) and CATtalk creates the new device model to be saved in the IoTtalk DB.

To configure an existing device model, the artist selects that device model (e.g., Smartphone; Figure 6 (1)) in the "DM Name" pull-down menu. The ModelGen subsystem retrieves the DFs of the device model from the IoTtalk DB, and instructs the GUI to pop up both the DF module (Figure 6 (2)) and the "Add/Delete DF" module (Figure 6 (3)).

To obtain more information about a specific DF, the artist moves the mouse pointer over the DF name in either the DF module or the "Add/Delete DF" module (e.g., Acceleration; see Figure 6 (4)). The ModelGen subsystem obtains the details of the selected DF and the GUI shows them in the Device Feature Window (Figure 6 (5)) next to the Device Model Window. In this way, the artist can conveniently investigate the details of a DF selected in the Device Model Window.

If the artist attempts to delete a device feature (e.g., Microphone) from the device model, he/she simply clicks the DF item in the DF module (Figure 6 (6)), and this DF is shown in the "Add/Delete DF" module automatically. Then the artist unselects the DF in the "Add/Delete DF" module (Figure 6 (7)) to remove it from the device model.

After the artist has completed configuring the device model and clicks the "Save" button (Figure 6 (8)), a dialog box pops up to reconfirm the modifications (Figure 6 (9)). The device model name is shown in this dialog box, which allows the artist to rename the modified device model. When the artist clicks the "OK" button (Figure 6 (10)), the ModelGen

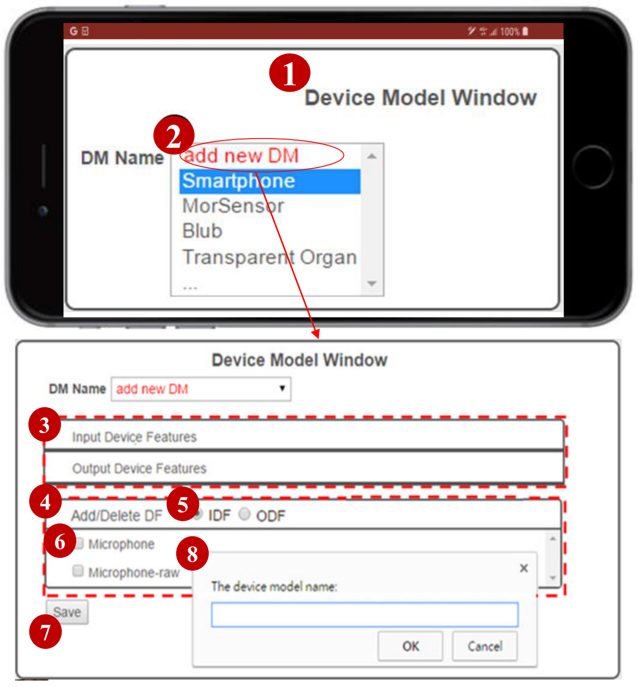
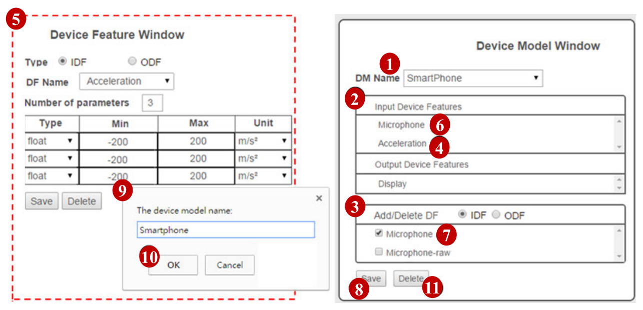


FIGURE 5. Device model creation.



Type	Min	Max	Unit
float	-200	200	m/s²
float	-200	200	m/s²
float	-200	200	m/s²

FIGURE 6. Device model modification.

subsystem stores the information of the specific device model in the IoTtalk DB. The artist may remove the device model from the IoTtalk DB by clicking the "Delete" button (Figure 6 (11)).

In [2], the feminine gender performance technology can be transparently implemented as the CATtalk project with the following device models created by ModelGen: The wearable robotic is an output device with the ODFs for the LED and the foldable wings. The flex sensors and the breath detector or button are the IDF of an input device. The SA/DA for these devices can be created following the procedure described in Section VI.

V. X-TALK CREATION AND MANAGEMENT

Based on the device models created in Section IV, the CATtalk GUI allows an artist to develop interactive artwork through the web-based project window (Figure 7 (1)). The operation flow is (6)→(7)→(5)→(10) in Figure 2. We use PuppetTalk as an example. The GUI for the PuppetTalk project consists of a banner bar and a graph editor called the project window. The banner bar includes the Model pulldown menu. When the artist clicks the menu button (Figure 7 (2)), the names of all device models stored in the IoTtalk DB are listed in the menu. When the Puppet model is selected (Figure 7 (3)), the icon of the Puppet model is shown in the right side of the project window (Figure 7 (4)). There are 6 small icons inside the Puppet icon, which represent the ODFs for controlling the puppet head, the hands, the legs and the wrist. For example, Head-O (Figure 7 (5)) controls the puppet head (Figure 7 (5a)). The connection between the puppet model in the network domain and a puppet device in the device domain is established by clicking the gear icon in the left-upper corner of the puppet icon. Then the DF labelled "x" is connected to the "xa" part of the art device. For example, "5" is connected to "5a" in Figure 7. The connection procedure is similar to that for Bluetooth connection, where the DA of the puppet device establishes an ODF channel for the head motor to receive the Head-O signals from the CATtalk server, and forwards the received signals to the SA. The SA implements the logic to interpret the signals received from the DA, which rotates the head motors. Details of DA and SA can be found in Section VI. In our implementation, an ODF of the robot puppet controls a motor set that consists of one or more motors. The head motor makes 90-degree rotation vertically, and each motor set of the hands and the legs includes three motors. For example, for a hand, the first motor makes horizontal rotation in 180 degrees for the shoulder, the second motor makes horizontal rotation in 90 degrees for the elbow, and the third motor makes horizontal rotation in 90 degrees for the palm. More details can be found in [17].

When the artist selects the glove item from the Model menu twice, two glove icons are shown in the left side of the project window. The small IDF icons inside a glove icon represent the fingers. For example, Index-I (Figure 7 (11)) receives the signals from the index finger (Figure 7 (11a)). Note that for the glove icons in Figure 7, only three fingers are used. In CATtalk, the artists can select a subset of the IDFs/ODFs in a device model when the model is selected. The details are given in Section VI.

A finger controls a motor set of the robot puppet by connecting the IDF for the finger to the ODF for the motor set in the CATtalk GUI. This connection is created by dragging a link between the IDF and the ODF. For example, the Join 1 link in Figure 7 connects Index-I of G1 to Head-O of the robot puppet so that the index finger can controls the robot head. There is a circle in the middle of the join link, which allows the artist to write a short Python program to manipulate the messages delivered in the link. We will elaborate on this "join circle" mechanism in Section VII. Through the join-link connection of the IDFs and the ODFs, CATtalk provides a simple no-code approach for the artist to design the interaction between the art devices.

The artist can easily modify the PuppetTalk project in Figure 7 to become a digital twin project with a robot puppet P1 and an animated puppet P2. The animated puppet mirrors

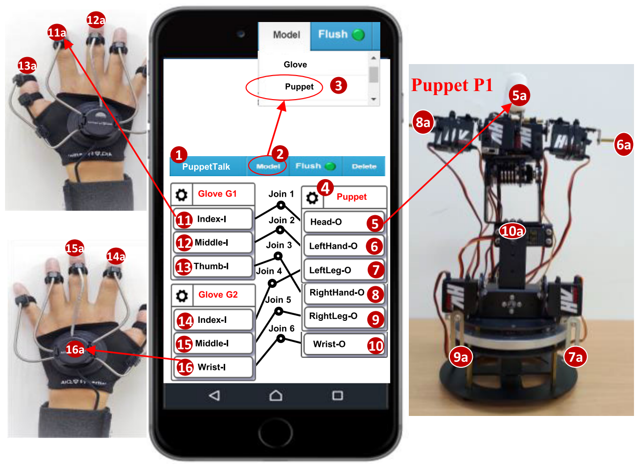


FIGURE 7. The PuppetTalk project.

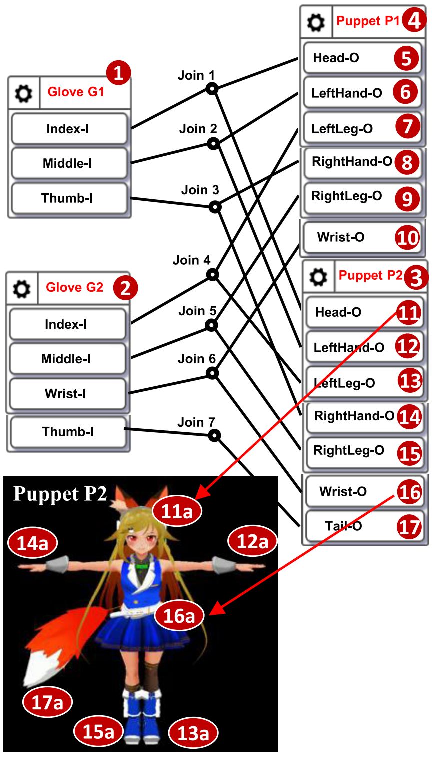


FIGURE 8. The digital twin version of the PuppetTalk project.

the behavior of the robot puppet. This project is illustrated in Figure 8, which extends the PuppetTalk project by adding the animated puppet "P2" (Figure 8 (3)). The connections among gloves G1, G2 and the robot puppet P1 are the same as those in Figure 7.

To create the mirror effect, we simply connect G1/G2 to P2 by forking Joins 1-6 to connect the IDFs to the ODFs of P2. When a signal message arrives at a forked circle, the same message is multicast to all ODF destinations. We add one more IDF Thumb-I for G2 to control the tail of the animated puppet through Tail-O of P2.

The BCI-driven mechanics in [7] can be straightforwardly implemented in CATtalk. We replace the gloves in Figure 8 by the HTC Vive Pro and Emotiv Insight 5-channel headset. The HTC headsets control the puppets just like playing the AR games. The Emotiv headset is connected to a data collection output device provide by DataTalk (the details of this device can be found in [20]). Immersive Experience Questionnaire (IEQ)-based in-game immersion measurement algorithm developed in [7] can be effortlessly plugged in DataTalk if the API for the algorithm is provided as described in Section VI. The results of the game interaction are sent to DataTalk for analysis, and then saved in IoTtalk DB. The operation flow is (7)→(5)→(8)→(5)→(10) in Figure 2. We can further use the player's bio signals (EEG) to control the game by simply making the Join link from the Emotiv headset to the puppets (VR game). Such feedback explores another research direction for the study in [7]. Similarly, the animal footprint system in [8] can be straightforwardly implemented by replacing the smart gloves by Kinect sensors and replacing the animated puppet by the floor projector.

VI. DA AND SA GENERATION

CATtalk provides a simple mechanism to connect the real art device to the device model in the CATtalk server. The operation flow is (6)→(7)→(5)→(10)→(6)→(11) and then offline from (11) to (4) and (3) in Figure 2. In the art device, the SA (Figure 2 (3)) implements the logic of the device (for example, the puppet motor rotation algorithm) and connects it to the DA. It is not trivial for an artist to implement the SA/DA for an art device if he/she is not familiar with CATtalk. In this section, we show how the DSGen subsystem automatically generates the DA/SA for an art device. We use the SwingLight device as an example. This device is a light ball mounted on the top of a thin pole that swings in the wind (Figure 9 (1)), and the color of the light ball can be controlled by smartphones. The artist first creates two ODFs for the SwingLight following the procedure in Section IV.A: The ODF Light-O is a toggle switch that turns on the light for value 1 and turns off the light for value 0. The ODF Color-O switches the color of the light according to a color map (twelve colors are represented by the values range from 0 to 11). Then the artist creates the SwingLight device model following the procedure in Section IV.B. Finally, the device model is saved in the IoTtalk DB.

DSgen uses the device model to generate the SA/DA. We show how DSgen provides a no-code approach to generate the SA/DA for a SwingLight device called "Swing1". The artist first initiates the SwingLight Creation project. Since the artist already created the SwingLight device model, it can be found in the Model pulldown menu. By clicking the SwingLight item (Figure 9 (2)), the SwingLight work window pops up (Figure 9 (3)). From this window, the artist selects the numbers of devices whose lights (Figure 9 (4)) and the color (Figure 9 (5)) can be controlled by the audience. The default numbers are one.

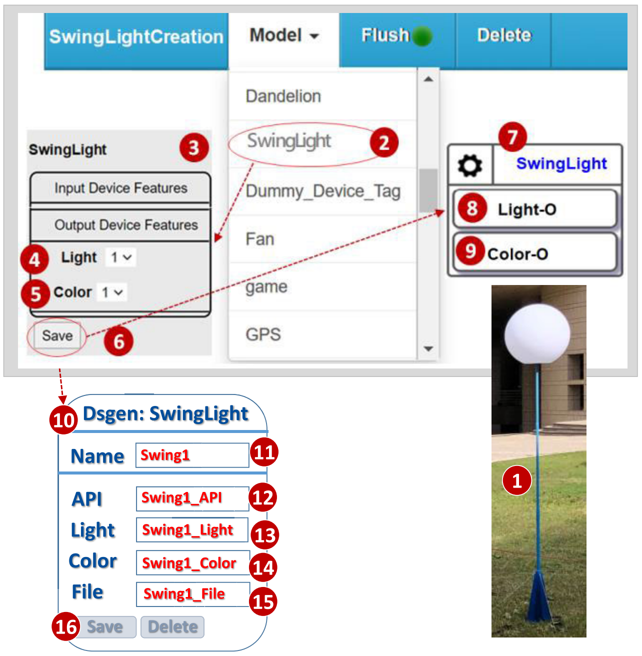


FIGURE 9. The SwingLight creation project.

```
SA01. import time, random, requests
SA02. import DA
SA03. import Swing1_API
SA04. ServerURL = 'https://DomainName'
SA05. Reg_addr = "SL-No-001"
SA06. DA.profile['dm_name']='SwingLight'
SA07. DA.profile['df_list']=['Light-O', 'Color-O',]
SA08. DA.profile['d_name']= 'Swing1'
SA09. DA.register(ServerURL, Reg_addr)
SA10. while True:
SA11.     try:
SA12.         Light_data = DA.pull('Light-O')
SA13.         Swing1_Light (Light_data)
SA14.         Color_data = DA.pull('Color-O')
SA15.         Swing1_Color(Color_data)
SA16.     except Exception as e:
SA17.         print(e)
SA18.         if str(e).find('mac_addr not found:') != -1:
SA19.             print('Reg_addr is not found. Try to re-register.')
SA20.             DA.register(ServerURL, Reg_addr)
SA21.         else:
SA22.             print('Connection fails.')
SA23.             time.sleep(1)
SA24.     time.sleep(0.2)
```

FIGURE 10. The SA pseudo code for Swing1.

When the artist presses the "Save" button (Figure 9 (6)), the CATtalk GUI shows the SwingLight icon in the right side of the project window (Figure 9 (7)), and asks if the artist wants to create the DA/SA for this device model. If so, the DSgen subsystem automatically generates the DA and pops up the DSgen GUI (Figure 9 (10)) to guide the artist to create the SA. For the details of the DA generation, the reader is referred to [21]. The SA pseudo code for SwingLight is listed in Figure 10, where the black lines without yellow marks are a general template already developed in CATtalk. The codes with yellow marks are automatically generated through steps (3)-(6) in Figure 9. The red parts are the codes to be typed by the artist through the DSgen web-based GUI window (Figure 9 (10)). This GUI window tailored for the SwingLight device "Swing1" is automatically generated by the DSgen subsystem. In Figure 10, Line SA01 imports the common Python libraries used by the SA. Line SA02 imports the DA library. Line SA03 imports the Swing1_API library provided by the artist at Figure 9 (12). Line SA04 specifies the URL of the CATtalk server.

In an IoT system, every IoT is identified by a unique ID address. Line SA05 provides the address for the Swing1 device to register to the CATtalk server. This address is either the MAC address of the art device hardware or is automatically generated by DSgen.

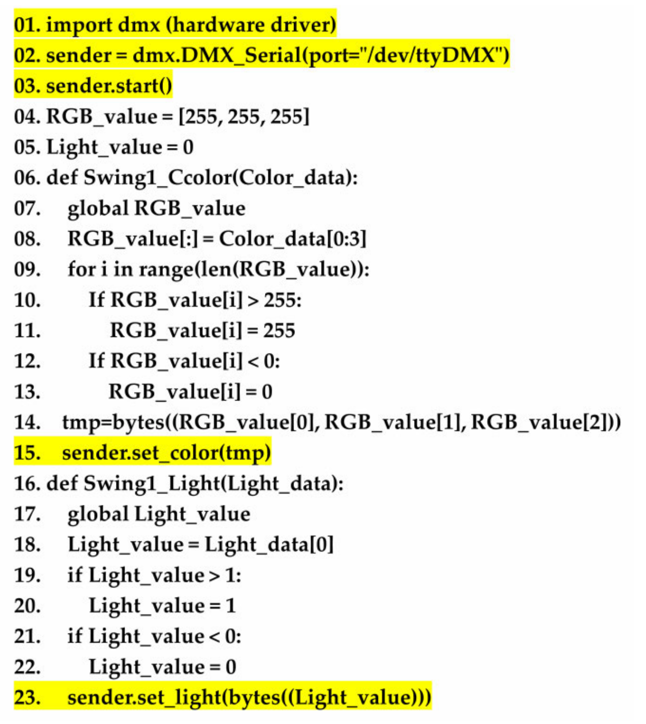
Line SA06 specifies the name of the device model SwingLight, which is determined at Figure 9 (2). Line SA07 lists the DFs of Swing1, which are selected at Figure 9 (4) and (5). Line SA08 specifies the device name "Swing1" filled by the artist at Figure 9 (11). Line SA09 registers Swing1 to the CATtalk server. Lines SA10-SA24 are a while loop to deliver the DF signals sent from the CATtalk server to the art device. At Lines SA12 and SA13, the DA pushes the result of the function Swing-Light to Light-I. This function is provided by the artist at Figure 9 (13). Similarly, the SA handles Color-I at Lines SA14 and SA15. Lines SA16-SA25 handle the connection exceptions for the device.

The complete SA is saved in the file "Swing1_File" specified by the artist (Figure 9 (15)). After the SA code is saved (Figure 9 (16)), the artist installs and executes this file in the Swing1 device to connect to the CATtalk server.

Note that Swing1_Light and Swing1_Color are the names of the functions defined in Swing1_API. In the DSgen approach, the artist is required to write the String1_API library (Figure 9 (12)) that defines the Light and the Color functions. The pseudo code of Swing1_API is listed in Figure 11, where the hardware specific codes are marked yellow.

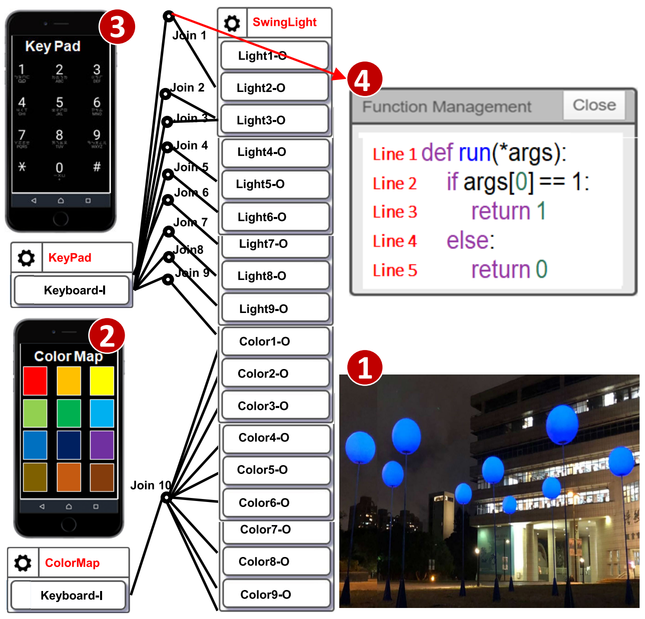
In this API, Lines 1-3 set up the Swing1 hardware DMX. Lines 4 and 5 initialize the color and the light variables. Lines 6-15 define the Swing1_Color function, and Lines 16-23 define the Swing1_Light function. Lines 15 and 23 use the functions provided by the driver of the DMX hardware. DSgen saves Swing1_API in the IoTtalk DB, and other authorized artists can reuse these functions to create the SA for similar art devices.

The program of the music-to-dance (M2D) mechanism in [10] can be used to create the M2D_API library. Then following the DSgen procedure above, the artist can create an M2D input device that drives the puppets in Figure 1 to dance. CATtalk also automatically generates the SA/DA of the art devices controlled by Raspberry Pi and Arduino-based microcontrollers. The details can be found in [21] and [22].



```
01. import dmx (hardware driver)
02. sender = dmx.DMX_Serial(port="/dev/ttyDMX")
03. sender.start()
04. RGB_value = [255, 255, 255]
05. Light_value = 0
06. def Swing1_Ccolor(Color_data):
07.     global RGB_value
08.     RGB_value[:] = Color_data[0:3]
09.     for i in range(len(RGB_value)):
10.         If RGB_value[i] > 255:
11.             RGB_value[i] = 255
12.         If RGB_value[i] < 0:
13.             RGB_value[i] = 0
14.     tmp=bytes((RGB_value[0], RGB_value[1], RGB_value[2]))
15.     sender.set_color(tmp)
16. def Swing1_Light(Light_data):
17.     global Light_value
18.     Light_value = Light_data[0]
19.     if Light_value > 1:
20.         Light_value = 1
21.     if Light_value < 0:
22.         Light_value = 0
23.     sender.set_light(bytes((Light_value))
```

FIGURE 11. Swing1_API.



```
Line 1 def run(*args):
Line 2     if args[0] == 1:
Line 3         return 1
Line 4     else:
Line 5         return 0
```

FIGURE 12. Interactions with the dancers: (a) DanceCamera; (b) Remote puppet show.

VII. INTERACTIONS WITH THE AUDIENCE

CATtalk allows the audience to interact with the art devices through the browsers of their smartphones. We use SwingLight in Section VI as an example to create SwingTalk, a SwingLight array project that allows the audience to play with the SwingLight devices. In this project, 9 SwingLight devices (Figure 12 (1)) are manipulated by two web-based controllers KeyPad and ColorMap in the smartphones. By scanning the QR codes for these two controllers, an arbitrary smartphone can access KeyPad and ColorMap through its browser. Therefore, anyone in the audience can play the swing lights using his/her own smartphone.

In this project, the SwingLight model accommodates 9 devices by specifying the number 9 in Figure 9 (4) and (5). The color buttons of ColorMap (Figure 12 (2)) determine the colors of all SwingLight devices. This mechanism can be simply created by Join 10 (a fork link) that enables Keyboard-I of ColorMap to send the same message to all color ODFs. The *i*-th key of KeyPad (Figure 12 (3)) is a toggle button to turn on/off the light of the *i*-th SwingLight device. We use 9 individual Join links to connect Keyboard-I of KeyPad to the SwingLight devices, where Join *i* link connects to the *i*-th SwingLight device. Since Keyboard-I sends out the number pressed in the KeyPad device, Join *i* link should check if the message is targeted for the *i*-th SwingLight device. The mechanism is implemented as follows.

The artist clicks the circle in the middle of the Join *i* link to pop up the "Function Manager" window (Figure 12 (4)). In this window, the artist implements a simple Python function where args [0] in Line 2 is the keypad value received from the smartphone. Line 2 checks if the received value is for the *i*-th device. If so, CATtalk sends the toggle signal "1" to the Light ODF of the device in Line 3 (the ID in the Join 1 function is "1"). Otherwise, do nothing (i.e., return 0 in Line 5). The function is executed in each of Joins 1-9, where the ID is "*i*" in Line 3 for Join *i*. In this example, CATtalk provides a low-code approach that allows the artist to create intelligence for their interactive art project by writing a small segment of Python code.

In SwingTalk, the KeyPad and the ColorMap are web-based cyber art devices that can be straightforwardly accommodated in CATtalk through the procedure described in Section VI.

Like YUKINKO in Museum of Contemporary Art Tokyo [1], a CATtalk project called MirrorTalk allows a visitor to interact with a digital mirror (Figure 13 (1)) by scanning the QR code (Figure 13 (2)). Then the browser of the smartphone shows the S-Mirror device (Figure 13 (3)). Similar to YUKINKO [1], the visitor of MirrorTalk chooses an abstract pattern through the setup buttons (Figure 13 (4)), places it at anywhere in the mirror area of the smartphone. We can extend the YUKINKO features through a ripple through effect that automatically generates the subsequent patterns in different positions of the digital mirror. The same effect is also shown in the visitor's smartphone.

It may require significantly programming effort to provide this effect in YUKINKO. On the other hand, the effect can be achieved in MirrorTalk through the following no-code procedure. In the MirrorTalk project, the device model (Figure 13 (5)) for S-Mirror consists of an input device placed in the left side of the project window (Figure 13 (6)) and an output device in the right side of the window (Figure 13 (7)). The device model for D-Mirror is an output device (Figure 13 (8)). The S-Mirror icon is mapped to the Smartphone and the D-mirror icon is mapped to the digital mirror. There are two DFs: Position and Pattern. The Patten DF specifies the shape, the color and the abstract pattern. The first pattern is determined by the visitor through the setup area of the browser (Figure 13 (4)). The Position DF specifies the position to place the pattern in the mirror, which is determined when the visitor touch the position on the screen of the smartphone.

In this project, Position-I of S-Mirror (Figure 13 (9)) is connected to Position-O of D-mirror (Figure 13 (11)) and Position-O of S-Mirror (Figure 13 (13)) in Join 1. Similarly, Join 2 connects the Pattern DFs. The path (9)→(13) in Join 1 and the path (10)→(14) in Join 2 create the ripple through effect. By clicking the circle of Join 1, we use the Function Management Window to write a simple program to determine the next position of the pattern. For example, the functions may move the position to the upper-left direction of the screen, or move to a randomly selected position. Similarly, we use Join 2 function to change the shape, the color and the size of the next abstract pattern.

Initially, the visitor puts, for example, the first pattern (a yellow square) at position (15), and then the Join functions put the second pattern (a red circle) at positions (16) in the digital mirror and the same position (17) in the smartphone, respectively. The SA of the smartphone mirror forwards the signals it received from the ODFs to the IDFs after a short default elapsed time. The next pattern is then automatically generated by the Join functions again. If we do not specify any Join functions, then the visitor only sees one yellow square in both mirrors. The SA of S-Mirror should also determine the number of the iterations for the ripple through effect so that the application will not be trapped in an infinite loop. The interaction between the visitor and MirrorTalk are saved in the IoTtalk DB, and can be sent to the visitor later as a souvenir.

In CATtalk, smartphones are intensively used for interaction between the audience and the artwork. We also use this approach for virtual Physics experiments in education [23].

VIII. INTERACTIONS WITH THE DANCERS

In March 2022, we conducted a remote interactive art performance between the National Museum of Fine Arts in Taiwan and the Osaka University of Arts in Japan. This art performance "Your Movements and Smiles will be Everything for Me" connected the art devices and the dancers between Taiwan and Japan.

In this remote performance, the art device "TripodCamera" ((1) in Figure 14 (a)) is a camera mounted on a movable tripod. In the dance performance, this art device reacts to the gesture changes of the local or the remote dancers ((2) in Figure 14 (a)). Figure 15 shows an extension of the DanceTalk project as follows. We use MediaPipe [24] to interpret the gesture of the dancer captured by the camera. Therefore the

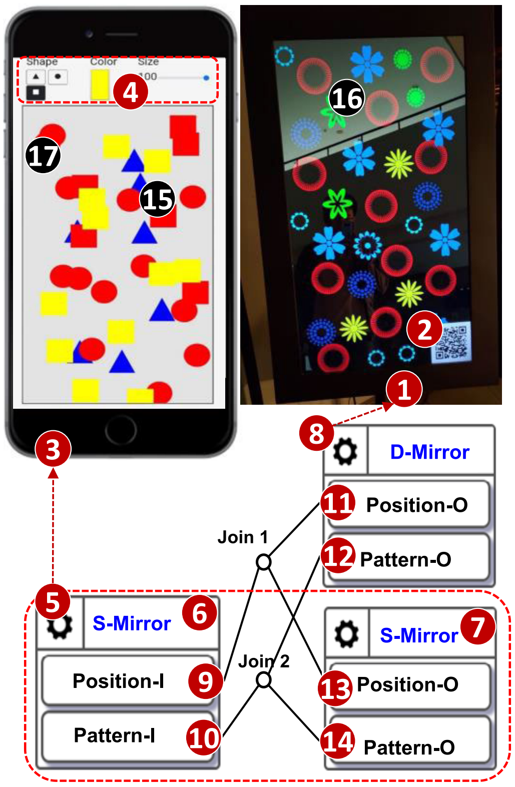


FIGURE 13. MirrorTalk.

SA of the TripodCamera includes the MediaPipe algorithm (Figure 15 (1)). The SA also implements a position algorithm to drive the motors of the tripod.

We develop the "DanceCamera" device model for the TripodCamera device, which has one IDF Instruction-I (the results of MediaPipe) and one ODF Position-O (movement positions for the tripod motors). In the DanceTalk project (Figure 15 (2)), there are two TripodCamera devices D1 and D2, one in Taiwan and the other in Japan. Each of them is represented by an input and an output devices. In this project, the movement of the tripod in Japan is controlled by the dancer in Taiwan (Figure 15 (3)-(6)), and the movement of the tripod in Taiwan is controlled by the dancer in Japan (Figure 15 (5)-(4)). DanceTalk can simply transform the dance of the performers into music. Specifically, we may replace the MediaPipe component by the IVA mechanism proposed in [9], and then fork the Join links to connect to a music playing output device as illustrated in Figure 16 (1). A video about a CATtalk music playing device is given in [25].

In a remote interactive art performance, the delays between the art devices and the CATtalk server affect the actions of the performers. For example, we stream the video produced by the local cameras to the remote locations. Although the two performers in Figure 14 (b) have the same dance style, due to the delay of video streaming, form the Taiwan side, the audience feels that the local dancer ((6) in Figure 14 (b)) leads the remote dancer ((5) in Figure 14 (b)). More often, the actions of a dancer and the art devices controlled by the

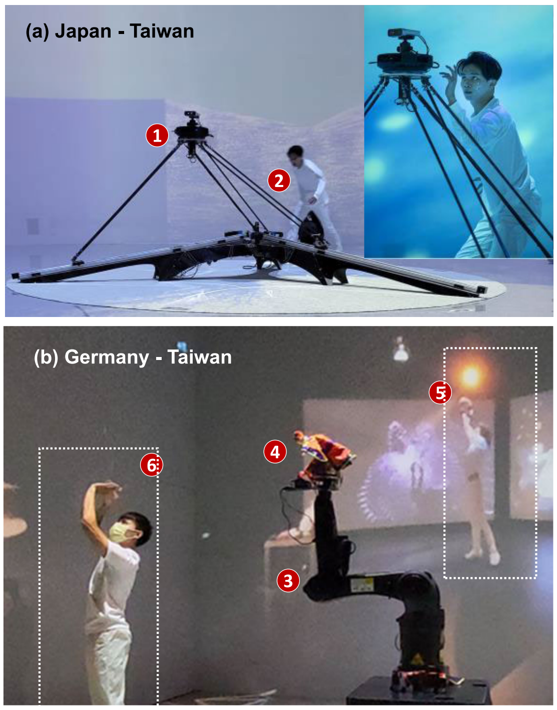


FIGURE 14. Interactions with the dancers: (a) DanceCamera; (b) Remote puppet show.

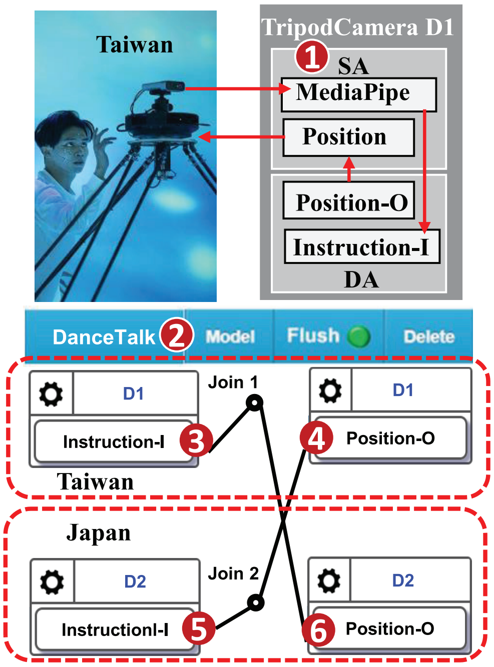


FIGURE 15. DanceTalk (with Tripod Camera).

dancer must occur in sequence. We will address this issue in the next section.

In September 2022, we were invited by TANZAHOi - International Dance and Dance Film Festival Hamburg, Hamburg, Germany, for the performance of "Hand extension,

014 台北國際發明暨技術交易
Taipei International Invention Show & Technomart
1

FIGURE 16. The music playing output device (by courtesy of S. S. Hwang).

inheritance of the palm – Mechanical Puppet Project". In this performance, the performers remotely control robotic arms ((3) in Figure 14 (b)) and physical puppets ((4) in Figure 14 (b)). The performers in both Germany ((5) in Figure 14 (b)) and Taiwan ((6) in Figure 14 (b)) use smart gloves to record hand movements. The relationship between form and meaning represents the history and culture of the puppet show in Taiwan, which introduces an old tradition to new technological possibilities.

Figure 17 illustrates an extension of the remote puppet performance. In this extended DanceTalk project, the TripodCamera device in Figure 14 (a) is replace by a DroneCamera device (a camera carried by an indoor drone; see Figure 17 (D1)). This device has the same IDF and ODF as the TripodCamera device and is therefore included as a member of the DanceCamera device model. Besides the drone, this project accommodates two smart gloves G1 and G2, two robot puppets P1 and P2, and one robot arm A1. In this figure, we use GloveIDFs-I to represents Index-I, Middle-I and Thumb-I in Figure 7. Similarly, we use PuppetODFs-O to represents Head-O, LeftHand-O and RightHand-O in Figure 7. Through the join connections in Figure 17, G1 controls D1 locally; G2 in Germany controls P1 in Taiwan and P2 in Germany; and D1 in Germany controls A1 in Taiwan.

IX. DELAY REQUIREMENT FOR THE REMOTE ART DEVICES

In the current CATtalk version, the artworks intelligently follow the art performers or the audience for the interactions. The actions taken by the art performers are designed by the scripts of the stage shows created by the playwriters. Therefore, to provide real-time interaction, we only need to consider the remote action rate and the out-of-order message delivery probability. This section investigates the delay issue for remote art device interaction. In [17] we have investigated the similar issue, where only local delays are considered. This section considers the delays for both local and remote locations.

Suppose that two smart gloves G1 (located in Taiwan) and G2 (located in Germany) in the PuppetTalk project are connected to the CATtalk server. In this remote collaborative

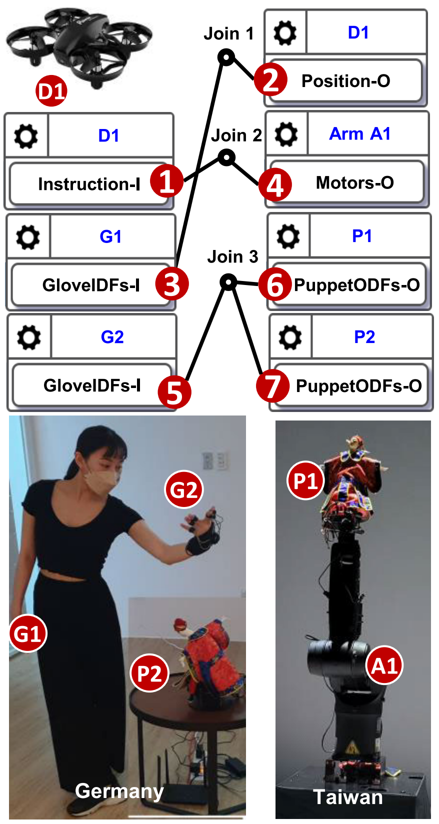


FIGURE 17. DanceTalk (with robot arm).

art performance, we have to consider the message delays between the art devices and the CATtalk server to ensure that the actions in the art performance are executed in sequence. Consider Figure 18. Suppose that G1 takes an action and then G2 takes the subsequent action. The action of G1 results in a signal message Msg1 sent from the DA of G1 at τ_0, which is received by the CATtalk server at τ_3. The delay of Msg1 is $t_3 = \tau_3 - \tau_0$. After an elapsed period $t_1 = \tau_1 - \tau_0 \geq 0$, the action of G2 results in another signal message Msg2 sent at τ_1, which is received by the CATtalk server at τ_2, where the delay of Msg2 is $t_2 = \tau_2 - \tau_1$. The CATtalk server anticipates to receive Msg1 earlier than Msg2 to guarantee in-sequence delivery. Let p_o be the probability of the out-of-order message delivery, then

$$p_o = \Pr[\tau_2 < \tau_3] = \Pr[t_1 + t_2 < t_3] \tag{1}$$

It is clear that the design of the X-Talk project (i.e., PuppetTalk) should ensure that $p_o \to 0$. We derive p_o as follows.

Let t_1 be a random variable with the density function $f_1(t_1)$ and the Laplace transform

$$f_1^*(s) = \int_{t_1=0}^{\infty} f_1(t_1)\,dt_1 \tag{2}$$

Similarly, let t_2 be a random variable with the density function $f_2(t_2)$ and the Laplace transform $f_2^*(s)$, and t_3 be a random variable with the density function $f_3(t_3)$ and the

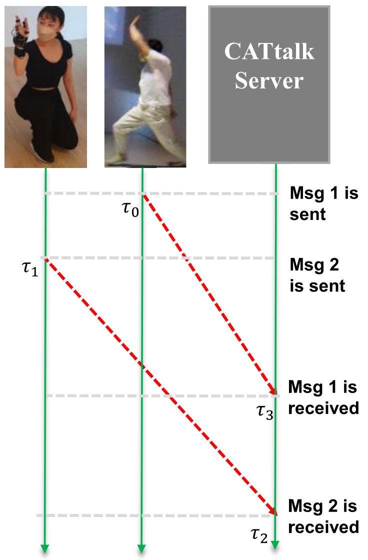


FIGURE 18. Timing diagram for out-of-order message delivery.

Laplace transform $f_3^*(s)$. It is obvious that if $t_1 = 0$ and $f_2(\cdot) = f_3(\cdot)$, then $p_o = 0.5$. Consider $t_1 > 0$. From Eq. (1),

$$p_o = \int_{t_1=0}^{\infty} f_1(t_1) \int_{t_2=0}^{\infty} f_2(t_2) \int_{t_3=t_1+t_2}^{\infty} f_3(t_3)\,dt_3 dt_2 dt_1 \tag{3}$$

If $f_3(t_3)$ is approximated by an Erlang density function, then $f_3(t_3) = f_E(t_3, n_3, \lambda_3)$ with the shape parameter n_3, the scale parameter λ_3 and the cumulative distribution $F_3(t_3) = F_E(t_3, n_3, \lambda_3)$. The general forms of the density and the cumulative distributions for an Erlang variable t with the shape parameter n (a positive integer) and the scale parameter $\lambda > 0$ are

$$f_E(t, n, \lambda) = \frac{\lambda^n t^{n-1} e^{-\lambda t}}{(n-1)!} \quad \text{and}$$
$$F_E(t, n, \lambda) = 1 - \sum_{j=0}^{n-1} \left(\frac{\lambda^j t^j e^{-\lambda t}}{j!}\right) \tag{4}$$

The mean is $\mathrm{E}[t] = \frac{n}{\lambda}$ and the variation is $\mathrm{V}[t] = \left(\frac{1}{n}\right) \mathrm{E}[t]^2$. The Laplace transform of $F_E(t, n, \lambda)$ is

$$f_E^*(s, n, \lambda) = \frac{\lambda^n}{(s+\lambda)^n} \tag{5}$$

We use the Erlang random variable to model the message delays because the Erlang distribution or a mixture of Erlang distributions are typically used to model IoT message delays and computing times [22], [26]. Substitute Eq. (4) into Eq. (3) to yield

$$p_o = \int_{t_1=0}^{\infty} f_1(t_1) \int_{t_2=0}^{\infty} f_2(t_2) \times \int_{t_3=t_1+t_2}^{\infty} \left[\frac{\lambda_3^{n_3} t^{n_3-1} e^{-\lambda_3 t_3}}{(n_3-1)!}\right] dt_3 dt_2 dt_1$$

$$
\begin{aligned}
&= \int_{t_1=0}^{\infty} f_1(t_1) \\
&\quad \times \int_{t_2=0}^{\infty} f_2(t_2) \left\{ \sum_{j=0}^{n_3-1} \left[\frac{\lambda_3^j (t_1+t_2)^j e^{-\lambda_3(t_1+t_2)}}{j!} \right] \right\} dt_2 dt_1 \\
&= \int_{t_1=0}^{\infty} f_1(t_1) \int_{t_2=0}^{\infty} f_2(t_2) \\
&\quad \times \sum_{j=0}^{n_3-1} \left\{ \lambda_3^j \left[\sum_{i=0}^{j} \binom{j}{i} \left(\frac{t_1^i t_2^{j-i} e^{-\lambda_3(t_1+t_2)}}{j!} \right) \right] \right\} dt_2 dt_1 \\
&= \sum_{j=0}^{n_3-1} \sum_{i=0}^{j} \binom{j}{i} \left(\frac{\lambda_3^j}{j!} \right) \left[\int_{t_1=0}^{\infty} t_1^i f_1(t_1) e^{-\lambda_3 t_1} dt_1 \right] \\
&\quad \times \left[\int_{t_2=0}^{\infty} t_2^{j-i} f_2(t_2) e^{-\lambda_3 t_2} dt_2 \right] \qquad (6)
\end{aligned}
$$

From Eq. (2) and the frequency-domain general derivative of Laplace transform, we have

$$\int_{t=0}^{\infty} t f(t)\, dt = -\frac{df^*(s)}{ds} \qquad (7)$$

Substitute Eq. (7) into Eq. (6), we have

$$
\begin{aligned}
p_o &= \sum_{j=0}^{n_3-1} \sum_{i=0}^{j} \binom{j}{i} \left(\frac{\lambda_3^j}{j!} \right) \left\{ (-1)^i \left[\frac{f_1^{*(i)}(s)}{ds^i} \right]_{s=\lambda_3} \right\} \\
&\quad \times \left\{ (-1)^{(j-i)} \left[\frac{f_2^{*(j-i)}(s)}{ds^{j-i}} \right]_{s=\lambda_3} \right\} \\
&= \sum_{j=0}^{n_3-1} \left[\frac{(-\lambda_3)^j}{j!} \right] \sum_{i=0}^{j} \binom{j}{i} \left[\frac{f_1^{*(i)}(s)}{ds^i} \right]_{s=\lambda_3} \\
&\quad \times \left[\frac{f_2^{*(j-i)}(s)}{ds^{j-i}} \right]_{s=\lambda_3} \qquad (8)
\end{aligned}
$$

Suppose that both t_1 and t_2 are Gamma random variables. The Gamma density function $f_G(t, \alpha, \beta)$ with the shape parameter $\alpha > 0$ and the scale parameter $\beta > 0$ is a generalization of the Erlang density function that can represent large variance of the elapsed times. The Laplace transform for $f_G(t, \alpha, \beta)$ is expressed as

$$f_G^*(s, \alpha, \beta) = \frac{\beta^\alpha}{(s+\beta)^\alpha} \qquad (9)$$

Let $f_G^*(s, \alpha_1, \beta_1)$ and $f_G^*(s, \alpha_2, \beta_2)$ be the Laplace transforms for the density functions of t_1 and t_2, respectively. From Eqs. (8) and (9) we have

$$
\begin{aligned}
p_o &= \sum_{j=0}^{n_3-1} \left(\frac{\lambda_3^j}{j!} \right) \sum_{i=0}^{j} \binom{j}{i} \left[\frac{\Gamma(\alpha_1+i)\beta_1^{\alpha_1}}{\Gamma(\alpha_1)(\lambda_3+\beta_1)^{\alpha_1+i}} \right] \\
&\quad \times \left[\frac{\Gamma(\alpha_2+j-i)\beta_2^{\alpha_2}}{\Gamma(\alpha_2)(\lambda_3+\beta_2)^{\alpha_2+j-i}} \right] \qquad (10)
\end{aligned}
$$

Since we assume that Msg1 and Msg2 occur randomly, it is fair to assume that they form a Poisson process and the inter-arrival time t_1 is Exponentially distributed, that is $\alpha_1 = 1$, and $E[t_1] = \frac{1}{\beta_1}$ is determined by the message frequencies of G1 and G2. Therefore,

$$f_1(t_1) = \beta_1 e^{-\beta_1 t_1} \qquad (11)$$

In this case, Eq. (10) is re-written as

$$
\begin{aligned}
p_o &= \sum_{j=0}^{n_3-1} \left(\frac{\lambda_3^j}{j!} \right) \sum_{i=0}^{j} \binom{j}{i} \left[\frac{(i!)\beta_1}{(\lambda_3+\beta_1)^{1+i}} \right] \\
&\quad \times \left[\frac{\Gamma(\alpha_2+j-i)\beta_2^{\alpha_2}}{\Gamma(\alpha_2)(\lambda_3+\beta_2)^{\alpha_2+j-i}} \right] \qquad (12)
\end{aligned}
$$

If α_2 is an integer, then Eq. (12) is simplified as

$$
\begin{aligned}
p_o &= \sum_{j=0}^{n_3-1} \sum_{i=0}^{j} \left[\frac{\lambda_3^j}{(j-i)!} \right] \left[\frac{\beta_1}{(\lambda_3+\beta_1)^{i+1}} \right] \\
&\quad \times \left\{ \frac{[(\alpha_2+j-i-1)!]\beta_2^{\alpha_2}}{[(\alpha_2-1)!](\lambda_3+\beta_2)^{\alpha_2+j-i}} \right\} \\
&= \sum_{j=0}^{n_3-1} \sum_{i=0}^{j} \binom{\alpha_2+j-i-1}{\alpha_2-1} \\
&\quad \times \left[\frac{\beta_1 \beta_2^{\alpha_2} \lambda_3^j}{(\lambda_3+\beta_1)^{i+1}(\lambda_3+\beta_2)^{\alpha_2+j-i}} \right] \qquad (13)
\end{aligned}
$$

Since $E[t_2] = \frac{\alpha_2}{\beta_2}$, we can expressed $\beta_2 = \frac{\alpha_2}{E[t_2]}$. Similarly, $\lambda_3 = \frac{n_3}{E[t_3]}$. Then Eq. (13) is rewritten as

$$
\begin{aligned}
p_o &= \sum_{j=0}^{n_3-1} \sum_{i=0}^{j} \binom{\alpha_2+j-i-1}{\alpha_2-1} \\
&\quad \times \left[\frac{\beta_1 \left(\frac{n_3}{E[t_3]} \right)^j \left(\frac{\alpha_2}{E[t_2]} \right)^{\alpha_2}}{\left(\frac{n_3}{E[t_3]} + \beta_1 \right)^{i+1} \left(\frac{n_3}{E[t_3]} + \frac{\alpha_2}{E[t_2]} \right)^{\alpha_2+j-i}} \right] \qquad (14)
\end{aligned}
$$

We have conducted simulation experiments (similar to those in [22], and [26]) to validate the analytic model. The results are consistent where the errors are within 3%. If $E[t_3] = E[t_2]$, Eq. (14) is expressed as

$$
\begin{aligned}
p_o &= \sum_{j=0}^{n_3-1} \sum_{i=0}^{j} \binom{\alpha_2+j-i-1}{\alpha_2-1} \\
&\quad \times \left[\frac{\beta_1 E[t_3]\, n_3^j \alpha_2^{\alpha_2}}{(n_3+\beta_1 E[t_3])^{i+1}(n_3+\alpha_2)^{\alpha_2+j-i}} \right] \qquad (15)
\end{aligned}
$$

Let $\theta_1 = \beta_1 E[t_3] = E[t_3]/E[t_1]$ then Eq. (15) is expressed as

$$
\begin{aligned}
p_o &= \sum_{j=0}^{n_3-1} \sum_{i=0}^{j} \binom{\alpha_2+j-i-1}{\alpha_2-1} \\
&\quad \times \left[\frac{\theta_1 n_3^j \alpha_2^{\alpha_2}}{(n_3+\theta_1)^{i+1}(n_3+\alpha_2)^{\alpha_2+j-i}} \right] \qquad (16)
\end{aligned}
$$

If $n_3 \to 1$, then Eq. (16) can be approximated as

$$\lim_{n_3 \to 1} p_o = \frac{\theta_1 \alpha_2^{\alpha_2}}{(1+\theta_1)(1+\alpha_2)^{\alpha_2}} \qquad (17)$$

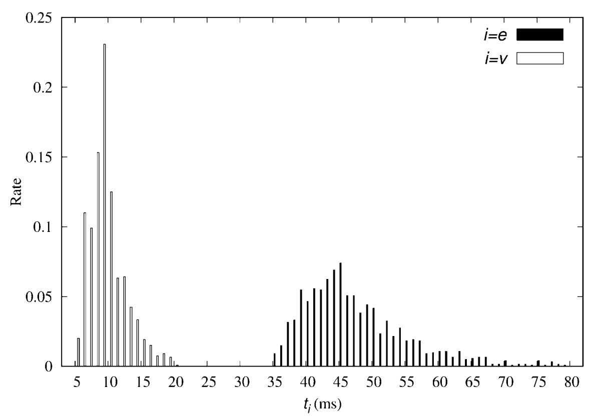


FIGURE 19. The histograms for the remote delays.

When both $n_3 \to 1$ and $\alpha_2 \to 1$, Eq. (17) is approximated as

$$\lim_{n_3\to 1,\alpha_2\to 1} p_o = \frac{\theta_1}{2(1+\theta_1)} \tag{18}$$

For an arbitrary n_3, when $\alpha_2 \to 1$, Eq. (16) is expressed as

$$\begin{aligned}\lim_{\alpha_2\to 1} p_o &= \sum_{j=0}^{n_3-1}\sum_{i=0}^{j}\left[\frac{\theta_1 n_3^j}{(n_3+\theta_1)^{i+1}(n_3+1)^{j-i+1}}\right]\\ &= 1-\left(\frac{n_3^{n_3}}{\theta_1-1}\right)\left[\frac{\theta_1(n_3+\theta_1)^{n_3}-(n_3+1)^{n_3}}{(n_3+1)^{n_3}(n_3+\theta_1)^{n_3}}\right]\end{aligned} \tag{19}$$

If $n_3 \to 1$, Eq. (19) is approximated as

$$\lim_{\alpha_2\to 1,n_3\to 1} p_o = \frac{\theta_1}{2(1+\theta_1)} \tag{20}$$

Eq. (20) is the same as Eq. (18). In Eq. (14), if both $n_3 \to 1$ and $\alpha_2 \to 1$, then

$$p_o = \frac{(\mathrm{E}[t_3])^2}{(\mathrm{E}[t_1]+\mathrm{E}[t_3])(\mathrm{E}[t_2]+\mathrm{E}[t_3])} \tag{21}$$

Let $\theta_2 = \mathrm{E}[t_2]/\mathrm{E}[t_3]$ then Eq. (21) is rewritten as

$$p_o = \frac{\theta_1}{(\theta_1+1)(\theta_2+1)} \tag{22}$$

If $\theta_2 = 1$, then Eq. (22) is the same as Eq. (18).

In CATtalk, the IoTtalk engine and the DA of an art device are installed a timer module to synchronize with the Network Time Protocol (NTP). Therefore, we can measure the one-way delay between the art device and the CATtalk server. Consider the measurements for the local delay t_2 and the remote delay t_3.

Figure 19 illustrates the histograms for t_2 and t_3. The average local delay is $\mathrm{E}[t_2] = 9.9575$ ms ≈ 0.01 $\overset{g}{\text{seconds}}$ and the variance is $\mathrm{V}[t_2] = 0.028\ \mathrm{E}[t_2]^2$. The average remote delay is $\mathrm{E}[t_3] = 47.75833$ ms ≈ 0.05 seconds and the variance is $\mathrm{V}[t_3] = 0.076\ \mathrm{E}[t_3]^2$.

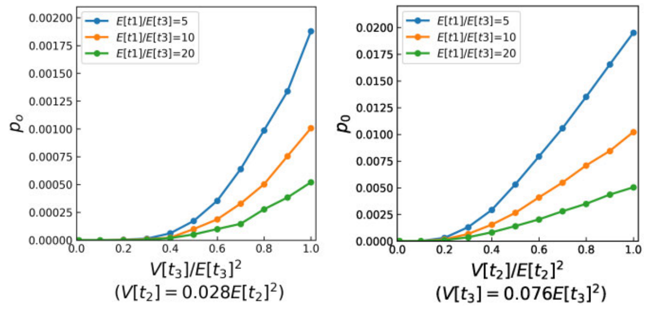


FIGURE 20. p_o against t_1, t_2, and t_3 ($\mathrm{E}[t_2] = 9.9575$ ms, $\mathrm{E}[t_3] = 47.75833$ ms).

Figure 20 plots p_o against t_1, t_2, and t_3 with the observed $E[t_2]$ and $E[t_3]$, where $E[t_1] = (5, 10, 20) \times \mathrm{E}[t_3]$; i.e., 0.239seconds, 0.478seconds, 0.956seconds, respectively. Figure 20 (a) shows the p_o performance seen from the local dancer with the observed $\mathrm{V}[t_2] = 0.028\ \mathrm{E}[t_2]^2$. Figure 20 (b) shows the p_o performance seen from the remote dancer with the observed $\mathrm{V}[t_3] = 0.076\ \mathrm{E}[t_3]^2$. For the variances observed in our measurements, $p_o < 0.01\%$ when $\mathrm{E}[t_1]$ is less than 1 second. Note that for video streaming, the cameras in both locations are remote to the display screens. Therefore, the remote delays should be considered for both locations.

The figure shows that p_o is directly proportional to $\mathrm{E}[t_1]$, and is increased as $\mathrm{V}[t_2]$ and $\mathrm{V}[t_3]$ increase. It is clearly that if the communication links between the art devices and the CATtalk server are stable (i.e., the variances are small than the Exponential scenarios), then the delay $\mathrm{E}[t_1]$ can be small.

The Exponential scenario ($\mathrm{V}[t_2] = \mathrm{E}[t_2]^2$ and $\mathrm{V}[t_3] = \mathrm{E}[t_3]^2$) is used for worst case analysis, which tell you what happens if the communications links are not stable. That is, we use Eq. (22) to derive a upper bound p_0^* for p_o as

$$\begin{aligned} p_o \le p_0^* &\Longrightarrow \theta_1 \le \frac{(\theta_2+1)p_0^*}{1-(\theta_2+1)p_0^*} \approx (\theta_2+1)p_0^*\\ &\Longrightarrow \mathrm{E}[t_1] \ge \frac{\mathrm{E}[t_3]}{(\theta_2+1)p_0^*}\end{aligned} \tag{23}$$

Therefore, if we expected that $p_o \le 2\%$ in the worst case, then the lower bound for $\mathrm{E}[t_1]$ is $\mathrm{E}[t_1] \ge \left(\frac{50}{\theta_2+1}\right)\mathrm{E}[t_3]$. For the measurements in Figure 19, if the delay variances are large (i.e., $\mathrm{V}[t_2]/\mathrm{E}[t_2]^2 = \mathrm{V}[t_3]/\mathrm{E}[t_3]^2 = 1$), we have $\mathrm{E}[t_1] > 2.1$ seconds for the local dancer and $\mathrm{E}[t_1] > 0.083$ seconds for the remote dancer. In the real case, we have controlled the delay variances to be less than 0.2 such that the average delay between two actions of a local (remote) performer is longer than 0.1 seconds, then the probability of out-of-sequence actions is less than 0.01%.

X. CONCLUSION

We proposed CATtalk to create and maintain interactive artworks. The novel idea of CATtalk is to treat all art devices in an interactive artwork as IoT devices. Through the CATtalk GUI, a new artwork can be easily created with the existing art devices. The art devices developed independently by individual artists can be quickly integrated in CATtalk through the DSgen and the ModelGen mechanisms.

By using CATtalk's no-code and low-code mechanisms, the artists can manipulate CATtalk with little or no programing efforts. To our knowledge, no platforms like CATtalk have been reported in the literature. CATtalk has a built-in mechanism so that when a smartphone scans a CATtalk generated QR code, the smartphone becomes an art device. Therefore, any person in the audience can play with an interactive artwork with his/her smartphone.

We also conducted analytic analysis, simulation and measurements to ensure that the interactive art performance in cross-country remote stages are not affected by the communications delays. In our measurements, the average local and remote communication delays are about 0.01 and 0.05 seconds, respectively. If the art performance is designed such that the average delay between two actions of a local (remote) performer is longer than 0.1 seconds, then the probability of out-of-sequence actions is less than 0.01%. Such delay analysis for remote interactive art performance has not been conducted in the literature.

CATtalk can be extended to accommodate intelligent interactions among the artworks themselves. Specifically, we are building multiple robot arms (one or more) in multiple locations in Taiwan (Hsinchu, Taichung and Tainan). This CATtalk network is sponsored by famous international manufactures including Quanta, Accton, Chunghwa Telecom, NCHC, AiQ and HIWIN. In this scenario, a negotiation mechanism is required to support intelligent interactions among the artworks. An example of the negotiation platforms is given in [28]. Our future work will investigate the negotiation policy among robot arms based on the study in [28].

REFERENCES

[1] I. Kanaya, M. Imura, and M. Kanazawa, "Interactive art to go," in *Proc. 11th Conf. Adv. Comput. Entertainment Technol.*, New York, NY, USA, Nov. 2014, pp. 1–4, doi: 10.1145/2663806.2663871.

[2] K. Ladenheim, R. McNish, W. Rizvi, and A. LaViers, "Live dance performance investigating the feminine cyborg metaphor with a motion-activated wearable robot," in *Proc. ACM/IEEE Int. Conf. Hum.-Robot Interact.*, New York, NY, USA, Mar. 2020, pp. 243–251, doi: 10.1145/3319502.3374837.

[3] R. Hong and H. He, "Interference and consultation in virtual public space: The practice of intermedia art in metaverse," in *Proc. 17th Int. Conf. Mobility, Sens. Netw. (MSN)*, Dec. 2021, pp. 792–797, doi: 10.1109/MSN53354.2021.00124.

[4] A. Jorjin. (2022). *A Pioneer of the AR Smart Glasses Industry*. Jorjin Technologies Inc. [Online]. Available: https://www.jorjin.com/products/ar-smart-glasses/odm-reference-design/j6f/

[5] B. Spittle, M. Frutos-Pascual, C. Creed, and I. Williams, "A review of interaction techniques for immersive environments," *IEEE Trans. Vis. Comput. Graph.*, early access, May 12, 2022, doi: 10.1109/TVCG.2022.3174805.

[6] Y.-B. Lin, Y.-W. Lin, C.-M. Huang, C.-Y. Chih, and P. Lin, "IoTtalk: A management platform for reconfigurable sensor devices," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1552–1562, Oct. 2017, doi: 10.1109/JIOT.2017.2682100.

[7] M. P. Woźniak, P. Sikorski, M. Wróbel-Lachowska, N. Bartłomiejczyk, J. Dominiak, K. Grudzień, and A. Romanowski, "Enhancing in-game immersion using BCI-controlled mechanics," in *Proc. 27th ACM Symp. Virtual Reality Softw. Technol.*, New York, NY, USA, Dec. 2021, pp. 1–6, doi: 10.1145/3489849.3489862.

[8] K. Moriya, T. Iio, Y. Shingai, T. Morita, F. Kusunoki, S. Inagaki, and H. Mizoguchi, "Playing with invisible animals: An interactive system of floor-projected footprints to encourage children's imagination," *Int. J. Child-Comput. Interact.*, vol. 32, Jun. 2022, Art. no. 100407, doi: 10.1016/J.IJCCI.2021.100407.

[9] A. Brennecke, M. Traber, S. Stimberg, and B. Stockleben, "Interactive image driven sound," in *Proc. Conf. Mensch Comput.*, New York, NY, USA, 2020, pp. 85–89, doi: 10.1145/3404983.3410004.

[10] H. Ahn, J. Kim, K. Kim, and S. Oh, "Generative autoregressive networks for 3D dancing move synthesis from music," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3501–3508, Apr. 2020, doi: 10.1109/LRA.2020.2977333.

[11] T. Bi, P. Fankhauser, D. Bellicoso, and M. Hutter, "Real-time dance generation to music for a legged robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1038–1044, doi: 10.1109/IROS.2018.8593983.

[12] S. F. Langa, M. M. Climent, G. Cernigliaro, and D. R. Rivera, "Toward hyper-realistic and interactive social VR experiences in live TV scenarios," *IEEE Trans. Broadcast.*, vol. 68, no. 1, pp. 13–32, Mar. 2022, doi: 10.1109/TBC.2021.3123499.

[13] Y.-B. Lin, M.-T. Yang, and Y.-W. Lin, "Low-cost four-dimensional experience theater using home appliances," *IEEE Trans. Multimedia*, vol. 21, no. 5, pp. 1161–1168, May 2019, doi: 10.1109/TMM.2018.2876043.

[14] (2022). *VR-All-Art: VR Art—Create & Explore VR Exhibitions*. [Online]. Available: https://vrallart.com/

[15] Z. Cooper. (2022). *The Future of Art: 8 Digital Installations and Interactive Spaces*. [Online]. Available: https://architizer.com/blog/inspiration/collections/digital-art-projection-installations/

[16] D. Restif. (2022). *Top Free Sites for Creating Digital Art, Tech & Learning*. [Online]. Available: https://www.techlearning.com/tl-advisor-blog/584

[17] Y.-B. Lin, H. Luo, C.-C. Liao, and Y.-F. Huang, "PuppetTalk: Conversation between glove puppetry and Internet of Things," *IEEE Access*, vol. 9, pp. 6786–6797, 2021, doi: 10.1109/ACCESS.2020.3048697.

[18] AiQ. (2022). *Cutting Edge Motion Capture*. [Online]. Available: https://www.aiqsmartclothing.com/

[19] J. White. (May 2022). *Three Ways Unity Makes Money, Seeking Alpha*. [Online]. Available: https://seekingalpha.com/article/4515216-ways-unity-makes-money

[20] Y.-W. Lin, Y.-B. Lin, C.-Y. Liu, J.-Y. Lin, and Y.-L. Shih, "Implementing AI as cyber IoT devices: The house valuation example," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2612–2620, Apr. 2020, doi: 10.1109/TII.2019.2951847.

[21] W.-E. Chen, Y.-B. Lin, T.-H. Yen, S.-R. Peng, and Y.-W. Lin, "DeviceTalk: A no-code low-code IoT device code generation," *Sensors*, vol. 22, no. 13, p. 4942, Jun. 2022, doi: 10.3390/S22134942.

[22] Y.-W. Lin, Y.-B. Lin, M.-T. Yang, and J.-H. Lin, "ArduTalk: An Arduino network application development platform based on IoTtalk," *IEEE Syst. J.*, vol. 13, no. 1, pp. 468–476, Mar. 2019, doi: 10.1109/JSYST.2017.2773077.

[23] Y.-B. Lin, M.-Z. Shieh, M.-F. Shih, and C.-C. Cheng, "EduTalk: An IoT environment for learning computer programming and physics," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21946–21957, Nov. 2022, doi: 10.1109/JIOT.2022.3182280.

[24] (2022). *MediaPipe*. [Online]. Available: https://google.github.io/mediapipe/

[25] (2022). *MusicTalk*. [Online]. Available: https://www.youtube.com/watch?v=VrRIy9KPCEw

[26] S.-R. Yang and Y.-B. Lin, "Performance evaluation of location management in UMTS," *IEEE Trans. Veh. Technol.*, vol. 52, no. 6, pp. 1603–1615, Nov. 2003.

[27] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance, Computer System Analysis Using Queuing Network Models*. Upper Saddle River, NJ, USA: Prentice-Hall, 1984.

[28] R. Rajavel and M. Thangarathanam, "Agent-based automated dynamic SLA negotiation framework in the cloud using the stochastic optimization approach," *Appl. Soft Comput.*, vol. 101, Mar. 2021, Art. no. 107040.

YI-BING LIN (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Washington, Seattle, USA, in 1990. He is currently the Winbond Chair Professor at the National Yang Ming Chiao Tung University (NYCU), the Chair Professor at the National Cheng Kung University and China Medical University, and an Adjunt Research Fellow at Academia Sinica. From 1990 to 1995, he was a Research Scientist with Bellcore. He joined the National Chiao Tung University (NCTU), where he became a Senior Vice President of NCTU, in 2011. From 2014 to 2016, he was the Deputy Minister at the Ministry of Science and Technology, Taiwan. He is the coauthor of the books *Wireless and Mobile Network Architecture* (Wiley, 2001), *Wireless and Mobile All–IP Networks* (John Wiley, 2005), and *Charging for Mobile All–IP Telecommunications* (Wiley, 2008). He is an AAAS fellow, an ACM fellow, and the IET fellow.

HELIN LUO (Member, IEEE) received the master's degree from the Graduate School of Arts and Technology, Taipei National University of the Arts, and the doctoral degree from the Graduate Institute of Networking and Multimedia, National Taiwan University. He specializes in creating inter-disciplinary works using art and technology. For his creations, he draws from his personal experience of being extremely addicted to online games during middle and upper school to explore the power of virtual worlds, the thrill of speed, and other variations during this era of technology. Furthermore, his works are centered around the concept of ''immigrant illness'' amidst this generation of digital immigrants. His works have been recognized at many electronic and contemporary art festivals both domestically and internationally. These include the Digital Arts Award Taipei (2008, 2010, 2011, and 2015), the FILE–Electronic Language International Festival (2009, 2010, and 2013), and the other Competitions. He is the first Taiwanese artist to create works using a four-axis drone. His works have won first prize for the Performance Award at the Digital Art Festival Taipei. He was also specially invited to Arts Electronica Festival to present his inter-disciplinary works made through drones.

CHEN-CHI LIAO received the B.S. and M.S. degrees in computer science and information engineering from the National Ilan University, Yilan, Taiwan, in 2016 and 2018, respectively. He is currently pursuing the Ph.D. degree in computer science and engineering with the National Yang Ming Chiao Tung University, Hsinchu, Taiwan.

• • •