MotionFlow: Time-axis-based Multiple Robots Expressive Motion Programming

Qiuyu Lu
Tsinghua University
Beijing, China
lqy17@mails.tsinghua.edu.cn

Yejun Liu
Tsinghua University
Beijing, China
yjl14@mails.tsinghua.edu.cn

Haipeng Mi
Tsinghua University
Beijing, China
haipeng.mi@acm.org

ABSTRACT

Robots have been gradually weaving into the fabric of many areas, including children's education and new media arts. Such combination promotes children's creativity and provides new artistic expression paradigms for artists. However, for robot systems that contain many degrees of freedom, the motion programming can be very complicated.

This paper presents a time-axis-based computer aided design tool for multiple robots expressive motion programming, MotionFlow. It allows users who have no prior coding experience to easily add and edit motion clips modules on the time axis. In the meanwhile, it provides a rendered animation preview of the robots' motions, which enables users to evaluate the programming result without repeatedly testing the program on the hardware. In this way, users can perform rapid optimization and adjustment of the motion design. According to the user study, MotionFlow is very easy to use and very efficient for motion editing. It can greatly lower the threshold of multiple robots motion programming and allows users to focus more on creative work.

CCS CONCEPTS

• Human-centered computing~Human computer interaction (HCI)~Interactive systems and tools • Social and professional topics~Professional topics~Computing education~K-12 education

KEYWORDS

human-computer interaction, computer aided design, robot programming; education; robotic art

ACM Reference format:

Qiuyu Lu, Yejun Liu and Haipeng Mi. 2020. MotionFlow: Time-axis-based Multiple Robots Expressive Motion Programming. In *Proceedings of 2020 3rd International Conference on Computer Science and Software Engineering (CSSE' 20). Beijing, China, 5 pages.* https://doi.org/10.1145/3403746.3403919

1 Introduction

With the development of technology and the intersection of disciplines, robots are gradually weaving into the fabric of people's daily lives. In terms of education, many schools have provided robot-related courses for adolescents and even children. Many of these courses (e.g., animatronics show class) require knowledge and skills in multiple fields, including screenwriting, stage art, voice performance, mechanical design, electronic engineering, etc. Such courses blur traditional lines between art and engineering, demonstrate the universality of creativity across disciplines, and can inspire students to see new career possibilities [1-3]. Also, more and more artists are trying to combine robots with art, such as robotic painting, sculpture, and orchestra performance [4-6]. Artists are translating the robotics technologies into artistic language.

In the above examples, the programming of various expressive actions of the robot is essential. However, as the robot's degree of freedom increases, robot motion programming will become very complicated. Further, if multiple robots are moving simultaneously, the relative positions and interactions of the robots will make motion programming even more difficult. Non-professionals often struggle with this kind of programming.

To deal with this problem, when designing children's robot education products, there is often a compromise between the number of robot's degrees of freedom and the difficulty of programming. For example, strictly limiting the number of degrees of freedom, minimizing the number of robots, and reducing the interaction between/among robots. Correspondingly, the programming tools are usually based on graphical programming, where users can program the motions by connecting pre-set graphical instruction modules without learning to code [7-9]. However, such programming method is inefficient when it comes to editing a large number of motions, and it cannot directly reflect the time and space relationship between different robot motions.

In addition, work like Topobo [10] and NAO [11] further simplified the programming of robot motions by allowing users to program via directly manipulating the corresponding 3D model virtually or the robot itself physically. Although such programming methods are straightforward, and can better support robot systems with lots of degrees of freedom; the programmed motions are like being uploaded into a black box, which means it

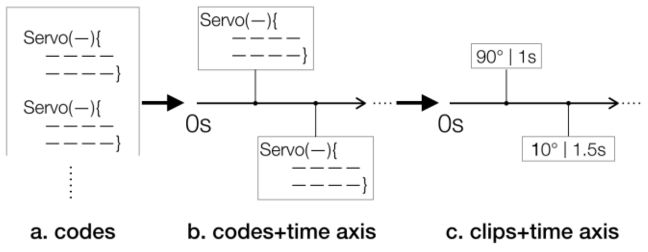

Figure 1: Beyond programming motion codes, toward editing motion clips

is difficult for users to modify them. Such method is more suitable for relatively short, random motion programming.

Unlike education-oriented robot(s), artists usually have to use a robot with a lot of degrees of freedom for artwork, or even many such robots to cooperate and achieve the desired artistic effect when creating robot performances. Therefore, artists generally need to cooperate with professionals to complete motion editing through code programming.

Essentially, robot motion programming is to control each servo to start moving at a certain time point and move to a certain angle over a certain period. Although such codes can be complicated (Figure 1a), if the codes for different servos are arranged along a time axis, the sequential relationship can be clear at a glance (Figure 1b). Further, by packaging these codes into instruction modules (clips), the user can set the servo by simply changing the values of the angle and the duration of the rotation, making the motion programming very simple and easy (Figure 1c). Based on this concept, we propose a multi-robot expressive motion programming tool based on the time axis. Users can directly drag the motion clip to the time axis and can adjust its duration and insertion point freely to program robot's motions quickly. At the same time, the tool can simulate and provide a preview of the programmed motions, allowing users to evaluate the programming result and make rapid optimization without running the hardware. Our work mainly provides the following contributions: 1). The concept of time-axis-based robot motion programming. 2). The development of an example design tool based on this concept. 3). User studies to evaluate such a design tool.

2 Example of A Multi-robot System

ZOOO is a multi-robot stage, and its overall system framework is shown in Figure 2a. Similar to many other multi-robot systems, users can program motions on the computer and upload the program to the main control board (e.g., an Arduino board). All the servos and speakers are controlled by the main control board and, if necessary, the drive board(s).

As shown in Figure 2b, ZOOO's mechanical hardware includes three branches (small robots), each branch contains 5 degrees of freedom (mouth opening and closing, head tilt, left and right arm rotation, body rotation), and a speaker. Besides, each branch is eccentrically attached to a small rotatable round panel. The large stage base is also rotatable. By combining rotation of the branches,

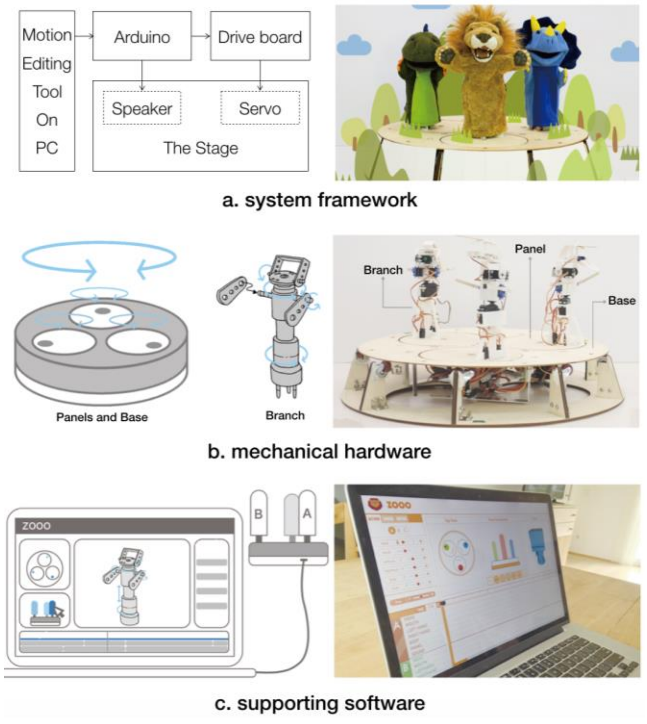

Figure 2: An example of multi-robot system – ZOOO

panels, and base, various forms of interaction among the robots can be realized. For this system, the large number of degrees of freedom and the various positional relationships among robots make programming motion for it very complicated.

Taking this multi-robot system as an example, the design, development, and evaluation of a time-axis-based multi-robot programming tool MotionFlow (Figure 2c) will be discussed below.

3 Time-axis-based Robot Motion Programming

MotionFlow is developed with web languages such as HTML5 and JavaScript. MotionFlow runs on the computer and communicates with the Arduino board through the serial port to control the robot system's hardware.

MotionFlow takes the advantages of HTML5 front-end development to build the graphical user interface. At the same time, it uses JavaScript's flexible event response capabilities to enable users to edit motions quickly and efficiently on the time axis. In addition, the preview function is implemented by WebGL (three.js), the audio insertion function is achieved via the Audio API, and the local operation of MotionFlow is enabled through a web runtime environment node-webkit which is based on Chromium and Node.js.

The core design concept of MotionFlow is to transform code-based robot motion programming into time-axis-based robot motion graphical programming so that users can quickly and easily implement robotic "choreographies".

When designing MotionFlow, we first ensure that the users can quickly and accurately program repetitively editable robot motion clips. And then, we make sure that the clips can be edited freely on the time axis. Lastly, we ensure that users can preview, save, upload the programming result at any time. Based on these basic principles, we divide the interface of MotionFlow into four functional areas: clip editing area, clip selecting/inserting area, motion preview area, and other functions area (Figure 3).

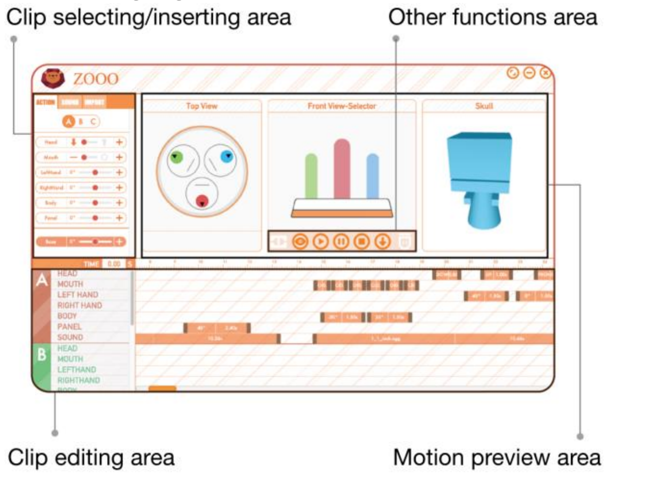

Figure 3: The overall interface of MotionFlow

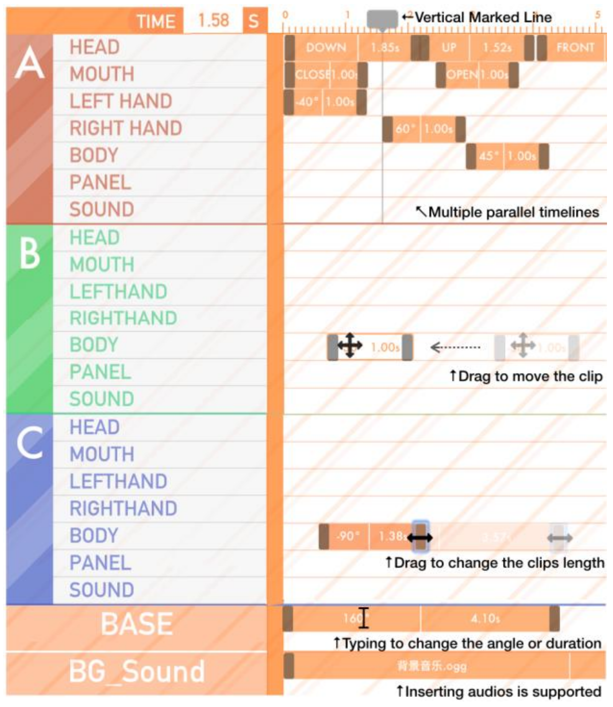

Figure 4: Editing motion and audio clips on the time axis

3.1 Edit Motion Clip on Time Axis

The clip-editing area is the core function area of MotionFlow (Figure 4). Users can quickly program the robots by adding or editing motion clips or audio clips to the time axis. The time axis contains multiple timelines, each timeline corresponds to a degree of freedom (one servo) or a audio track (one speaker). All timeline run in parallel and do not interfere with each other. In order to facilitate the user to distinguish between different timelines, the timelines are grouped based on their ownership (branch A/B/C or base), and named according to the corresponding "body" parts (mouth, head, arms, etc.). Users can add motion clips or audio clips to each timeline through the clip-selecting and inserting tool (described in detail in the next section), or by directly copying and pasting existing motion clips or audio clips on the timeline. Each motion clip contains two parameters: the absolute rotation angle and the rotation duration of the servo. The user can drag the ends of the clip to change the duration of motion clip or trim the audio clip, or drag the entire clip to change its position on timeline. The absolute rotation angle and duration can also be modified by typing in desired value. The clips do not have to be connected end to end. The user can customize the time interval between the clips. The clip editing area also contains a vertical marker line. The marker line mainly provides two functions: 1. Indicating the progress when preview or actual run the robots; 2. Indicating where the clip will be inserted when adding a clip.

3.2 Select and Insert Various Clips

The clip selecting/inserting area provides three different functional tabs for robot programming. (Figure 5).

The motion clip tab (Figure 5.a) allows users to select and insert various motion clips. Since the moving range of each part of the robot is different, the maximum rotating angles of each part are limited in advance. Among them, the mouth part only provides two stages of motion, opening and closing. After the angle is determined, users can click the plus sign to add the motion clip to the corresponding timeline.

The audio clip tab (Figure 5.b) provides audio clips insertion function. After importing the audio into the program, the user can then insert the audio into the selected timeline. On the Import tab

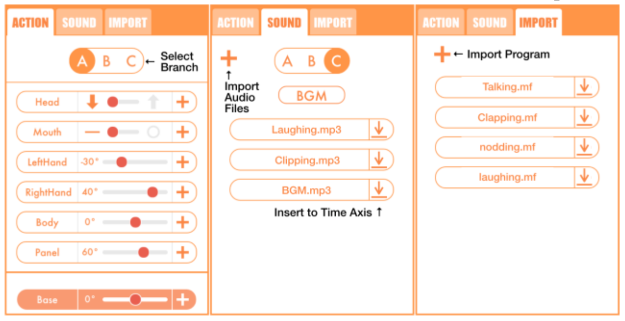

Figure 5: Tabs for selecting and inserting clips

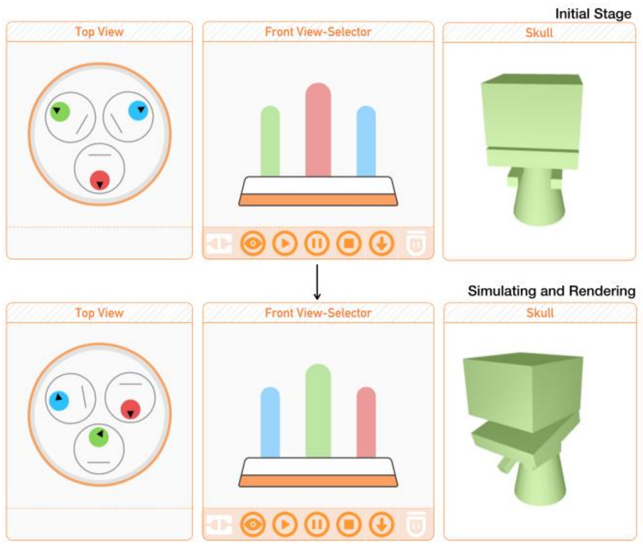

Figure 6: The simulation and preview area

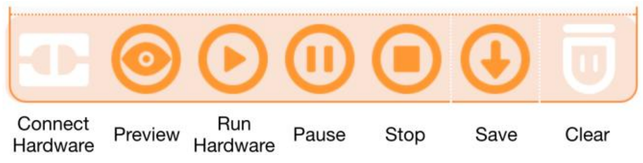

Figure 7: Interface for other functions

(Figure 5.c), users can import the previously saved programming file.Some preset motion combinations (e.g., repeated opening and closing mouth) are also provided here.

3.3 Robot Motion Preview

The preview area simulates and renders the motion programming results in the form of animation, including the top view/front view of the whole robot system and a preview of a single robot (Figure 6). The preview allows users to quickly check the programming result without actually uploading the program and running the hardware. Therefore, MotionFlow can simplify the iterative modification process of motion programming, providing users a smooth programming experience. The top view window and the front view window abstractly present the orientation of the robots and their spatial position relationship with each other. The single robot preview provides a detailed preview of the specific motion of the selected robot.

3.4 Other Functions

The other functions area provides controls for connecting the hardware, running/pausing/stopping the hardware/preview, saving the program, and clearing the time axis (Figure 7). The running hardware function is only available after connecting the computer with the robot system. After the user clicks to run the hardware, MotionFlow will compile the user's programming and upload it to Arduino. Then, the hardware will run. The preview is always available, even when the computer is not connected to the hardware. The user can preview the programming result at any time. In addition, the user can drag the vertical marker line to select the starting time point of preview.

4 User Study

We conducted a user study on MotionFlow to evaluate how such a time-axis-based motion editing tool could potentially ease multi-robot system programming. Three paid participants were recruited for this study. Two of them were fifth-graders (User A and User B), and the remaining one is 31 years old artists. Except that user B has certain robot programming experience using Arduino, neither participant A nor C has any relevant experience.

Before beginning the study, we introduced the ZOOO hardware system to all the participants. Then, the study was carried out with only one user at a time. In the first half of the study, we used 10 minutes to introduce MotionFlow and trained the participant to use it. Then, we asked the participant to use MotionFlow to program a dance show within 15 minutes. In the second half of the study, we first taught Arduino's programming knowledge related to servos and speakers in 10 minutes. And then asked the participant to code to make a dance show in 15 minutes. Moreover, the participant was told that they were free to make a new dance show or try to reproduce the dance show he/she created with MotionFlow before. In order to avoid the time-wasting caused by recording audio, we provided some sound effects materials in advance for the participants and told them that they did not have to record audio by themselves. The result is shown in Table 1.

In the first half of the study, participant A programmed a dance with some random motions. Two robots were involved. There were 62 motion clips in total, and 1 audio clip was used as background music. Participant B programmed an "ice dance" show, which involved all three robots and used a total of 135 motion clips and 1 audio clip as background music. About half of the action clips were programmed to control the rotation of the panels to simulate skating. Participant C programmed a mechanical dance show, which involved all three robots, 217 motion clips, and 13 audio clips. Among the audio clips, 1 was background music, 12 were short sound effects that match the motion. Besides, participant C also tried to put different robots under the "spotlight" (frontmost) through the rotation of the base.

Table 1: User Study Result

User	Clip Type	Number of Clips	
		MotionFlow	Coding
A	Motion	62	7
	Audio	1	0
B	Motion	135	42
	Audio	1	1
C	Motion	217	27
	Audio	12	2
Average	Motion + Audio	143	25

Both participant B and participant C used copying and pasting to create motions quickly. They also fine-tuned pasted motion clips to avoid repetition.

In the second half of the study, participant A programmed a dance with 7 motion clips. Only one robot was involved, and no audio clip was used. Participant B programmed a dance with 42 motion clips and successfully inserted 1 audio clip as background music. Two robots were involved. Participant C programmed a dance with 27 motion clips, successfully inserted 1 background audio clip, and 1 sound effect audio clip. All three robots were involved.

In the following user interviews, all participants stated that "programming on the time axis is very intuitive and easy; the preview function is very convenient, reducing the time for repeated modification; inserting audio in MotionFlow is much easier than in Arduino..." In addition, participant C said that "MotionFlow has some similarities with non-linear video editing software. I have lots of experience in video editing. MotionFlow is very easy to get started for me". Participant B, who had coding experience with Arduino, considered "when there are many servos, the code will get complicated. It becomes difficult to distinguish which piece of the code is controlling which servo, and it is even more difficult to figure out the sequence of them. Even though I want to improve the programming efficiency by copying the code, I often do not know where to paste the code. However, all these operations are very clear on the time axis in MotionFlow". Also, we found that because participants A and C were not familiar with coding, they often encountered compilation failures after trying to change the code, resulting in a lot of time wasted. Moreover, when using code to program, the participants often complained that the motion started time did not match their expectations, and the short sound effect appeared at the wrong time.

The user study results show that, regardless of whether the user has programming experience or not, time-axis-based multi-robot motion programming tools are more straightforward and easier to use than coding, allowing users to focus more on creative work.

4 Discussion & Future Work

While there are many exciting potentials, there are also limitations and space for improvement. For example, the current software mainly focuses on achieving single-line motion performance and does not allow the robots to interact with the surroundings and the audiences. In future research, we would like to try to add support for sensors to increase the interactivity and flexibility of motion editing. Adding such interaction means that the robot's performance will no longer be a single storyline. To deal with this problem, we can introduce multiple secondary time axes and the concept of interruption. When the robot system detects any input signals, it can pause the motions on the current time axis and switch to the corresponding secondary time axis. In addition, limited by the platform rendering capabilities, the preview is currently split into three views and contains a detailed rendering of only one robot. Later we will try a new development framework to enable detailed preview rendering of the entire robot system. Lastly, we believe that users can be more imaginative and creative in their work by freeing them from labor-intensive parts like coding. In the future, we will conduct more user study, such as evaluating whether software tools like MotionFlow can enhance children's creativity in animatronics courses.

5 Conclusion

This paper presents the time-axis-based graphical programming concept for multi-robot expressive motion editing, and introduces a programming tool MotionFlow based on a specific example. User study show that the tool developed based on this concept can greatly reduce the threshold and improve the effectiveness for multi-robot action programming.

ACKNOWLEDGMENTS

We would like to thank Chengpeng Mao and Qiuheqi Zhong for the brainstorming, the hardware prototyping, and the software developing.

REFERENCES

[1] Jennifer Ginger Alford, Lucas Jacob, Paul Dietz. (2013). Animatronics Workshop: A Theater + Engineering Collaboration at a High School. IEEE Computer Graphics & Applications, 33(6):9-13.

[2] Mose Sakashita, Tatsuya Minagawa, Amy Koike, Ippei Suzuki, Keisuke Kawahara, and Yoichi Ochiai. 2017. You as a Puppet: Evaluation of Telepresence User Interface for Puppetry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17). ACM, New York, NY, USA, 217–228.* DOI: https://doi.org/10.1145/3126594.3126608

[3] Will Bosley, Dave Culyba, Brenda Harger, Sabrina Haskell, Andy Hosmer, TJ Jackson, Seema Patel, Christine Skarulis, Peter Stepniewicz, Jim Valenti, Salim Zayat, Eugenia Leu, and Jichen Zhu. 2005. Interbots initiative: an extensible platform for interactive social experiences with an animatronic character. In *ACM SIGGRAPH 2005 Emerging technologies (SIGGRAPH '05). ACM, New York, NY, USA, 9–es.* DOI: https://doi.org/10.1145/1187297.1187307

[4] Eduardo Kac. (1997). Foundation and development of robotic art. Journal of Art Journal, 56(3):60-67.

[5] Meng Liu, Yanlin Liu. (2003). Research on the Application of Robot to Carving Making. Journal of Machine Tool & Hydraulics, 05:110-112.

[6] Jiayin Li, Tianjian Hu, Shenghua Zhang, and Haipeng Mi. 2019. Designing a musical robot for Chinese bamboo flute performance. In *Proceedings of the Seventh International Symposium of Chinese CHI (Chinese CHI '19). ACM, New York, NY, USA, 117–120.* DOI: https://doi.org/10.1145/3332169.3332264

[7] Jiasi Gao. 2018. Research on Design of Tangible Programming Toolkit for Children. Ph.D. Dissertation. Tsinghua University, Beijing.

[8] Eva-Sophie Katterfeldt, David Cuartielles, Daniel Spikol, and Nils Ehrenberg. 2016. Talkoo: A new paradigm for physical computing at school. In *Proceedings of the The 15th International Conference on Interaction Design and Children (IDC '16). ACM, New York, NY, USA, 512–517.* DOI: https://doi.org/10.1145/2930674.2935990

[9] Ayah Bdeir and Ted Ullrich. 2010. Electronics as material: littleBits. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction (TEI '11). ACM, New York, NY, USA, 341–344.* DOI: https://doi.org/10.1145/1935701.1935781

[10] Hayes Solos Raffle, Amanda J. Parkes, and Hiroshi Ishii. 2004. Topobo: a constructive assembly system with kinetic memory. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04). ACM, New York, NY, USA, 647–654.* DOI: https://doi.org/10.1145/985692.985774

[11] E. Pot, J. Monceaux, R. Gelin, and B. Maisonnier. 2009. Choregraphe: a graphical tool for humanoid robot programming. In *Proceedings of RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication. IEEE, New York, NY, USA, 46-51.* DOI: http://dx.doi.org/10.1109/roman.2009.5326209