Mechanical Delay Compensation and Synchronization for Robotic Musicians Using Neural Networks

Higor Camporez[1], Jair Silva[1], Leandro Costalonga[2] and Helder Rocha[1]

[1] Departamento de Engenharia Elétrica – UFES, Vitória - ES, Brasil
[2] Departamento de Computação e Eletrônica – UFES, São Mateus - ES, Brasil
higorcamporez@gmail.com, jair.silva@ufes.br
leandro.costalonga@ufes.br, helder.rocha@ufes.br

Abstract. Technology has influenced musical activities, mainly in the way of human interaction. One case is human-robot interaction where humans can control robotic musicians in real-time performances. However, robots have mechanical delays caused by electromechanical components. These delays need to be compensated to keep the robots in synch. Thus, this paper presents an approach to learn and compensate robots' mechanical delays using neural networks. Also, clock synchronization using the simple network time protocol is applied. As proof of concept, a robotic bongo was developed for tests. The experimental results for the mechanical delay compensation technique showed a reduction of 55.33% in the average delay between two robots.

Keywords: Robots Synchronization, Robotic Musicians, Mechanical Delay Compensation, Neural Networks

1 Introduction

Today's technology has changed the way humans interact with music, where new technologies have created new tools for musical interactions. This approach influenced the creation of research areas like ubimus (ubiquitous music) (Keller, Lazzarini, & Pimenta, 2014) and IoMusT (Internet of musical things) (Turchet, Fischione, Essl, Keller, & Barthet, 2018).

One of the approaches that use music and technology is robotic musicians (Weinberg, Bretan, Hoffman, & Driscoll, 2020; Bretan & Weinberg, 2016). These robots use many types of electronics, for example, percussive robots commonly use solenoids in their implementation (Singer, Feddersen, Redmon, & Bowen, 2004; Kapur, Trimpin, Singer, Suleman, & Tzanetakis, 2007) and brushless direct current motors. The last one can improve speed and musical expressivity in robots (Yang, Savery, Sankaranarayanan, Zahray, & Weinberg, 2020). There are also many forms of interaction such as an autonomous real-time interaction with humans like the robot Shimon (Hoffman & Weinberg, 2010) provides.

Robotic musicians can be controlled by messages. Examples using MIDI protocol are described in (Jordà, 2002; Singer, Larke, & Bianciardi, 2003; McVay, Carnegie, & Murphy, 2015). The control messages can be sent over LAN (local area network) which relates to networked music performance (NMP) (Rottondi, Chafe, Allocchio, & Sarti, 2016). The RoboMus (Camporez, Mota, Astorga, Rocha, & Costalonga, 2018) platform supports real-time robotic musical performances over LAN, using open sound control (OSC) (Wright & Freed, 1997). However, robotic musicians commonly have delays caused by electromechanical components and these delays need to be compensated to keep the robots in synch. According to Pisoni (1977) human beings can detect a time difference between two successive acoustic events (between 500 and $1.500Hz$) of up to 20 milliseconds which can be used as a reference for the delay between robots.

This paper describes improvements for the RoboMus platform, presenting the use of neural networks for learning robots' mechanical delay profile. This paper also presents the construction of a robotic bongo for the tests of synchronization.

2 The RoboMus Platform

The RoboMus platform (Camporez et al., 2018) is a framework that is planned to provide real-time robotic musical performances, providing interaction between robotic musicians and humans through control interfaces. The framework is open source and one of its targets is low-cost robots. The RoboMus can be divided into three parts: a) control interfaces, which provide interaction between humans and robots, b) musical message synchronization server (MMSS), that is a message centralizer between control interfaces and robots, where techniques to compensate the mechanical delays are applied, and c) robotic musicians.

The OSC protocol, used on the platform, has a specific tag for the time that represents the absolute time for a message to be executed. However, this approach requires that all RoboMus components (MMSS, robots, and interfaces) have synchronized clocks. Thus, the MMSS is also a master clock that implements the SNTP (simple network time protocol) (Mills, 2006). Since low-cost is a target for the platform, SNTP was chosen because it requires less computational resources. Also, a message can be delivered for a robot a few milliseconds before the absolute time to solve the network latency.

2.1 Macro-synchronization: Mechanical Delay Learning

Each robot may have its own design and, consequently, its own delay profile. Thus, the MMSS must know the delay profiles to provide synchronization where these profiles can be learned by neural networks. For example, the MMSS sends hundreds of messages to a robot and when this robot finishes an action execution, it sends back a message containing the spent time (delay). Afterward, with all the messages and delays (dataset), the MMSS can train a specific neural network to learn this robot's delay profile. These processes demand some time. However, they can be done in the background.

The RoboMus platform does not define a message format for a robot to return a delay. Thus, we defined a new message format for a spent delay where the OSC address is {MMSS OSC address}/*delay*/{robot OSC address}. The first parameter of the message is an identifier and the second is the delay in milliseconds.

The RoboMus includes an interesting system for the connection of a new robot in a virtual concert hall which is denoted as the handshake. In this approach, a robot sends its specific actions and control parameters. In other words, the robot sends what it is capable to do. The specific action format of the RoboMus handshake message is defined as: $< /action_1; parameter_{11}\_type; ... ; parameter_{1N}\_type > ... < /action_M; parameter_{M1}\_type; ... ; parameter_{MX}\_type >$. This format does not define the set of valid values to each action's parameter. However, this information is important for the MMSS to generate valid messages to train a robot.

We defined a new format for the specific action that includes a set of valid values for each parameter. Thus, at the end of the previous format, we added a set of valid values represented inside parentheses, in which each set is separated by ;. However, the first one is the action name. The new pattern is shown below:

$$< /action_1; parameter_{11}\_type; parameter_{12}\_type; ...; parameter_{1N}\_type > ... < /action_M; parameter_{M1}\_type; parameter_{M2}\_type; ...; parameter_{MX}\_type > (/action_1; set_{11}; set_{12}; ...; set_{1N}) ... (/action_M; set_{M1}; set_{M2}; ...; set_{MX}),$$

where each set can be described in two forms: 1) defining the extremities of the interval following mathematical notation, for example, $]a, b]$, where $]a$ denotes values greater than a and $b]$ denotes values less than or equal to b; or 2) defining a list of elements where the elements can be written inside curly brackets and separated by comma, for example, $\{p1, p2, p3, ..., pn\}$.

A mechanical delay of a robot represents the transition time from a state to another state. Thus, we assume that the last message executed represents the current state and the next message represents the next state. We defined a generic model of inputs and outputs for neural networks used to learn delay profiles. The inputs are the parameters from the last and next action message and two action ids that identify the actions. The output is the mechanical delay. The number of parameters for each message can vary according to action. Thus, the number of network input variables is defined as $2 \times n + 2$, where n is the number of parameters from the action with the highest number of parameters and the part $+2$ represents the two ids. In addition, for actions that contain less than n parameters, the MMSS fills the remaining input variables with zeros.

3 BongoBot Prototype

The BongoBot is a robotic bongo designed to test our macro-synchronization strategies. The robot was built considering the low-cost premise. Thus, it was built with four solenoids (12 Volts and 5 Newtons) and two piezoelectric sensors. Fig. 1 shows the robot's 3D model and its real version.

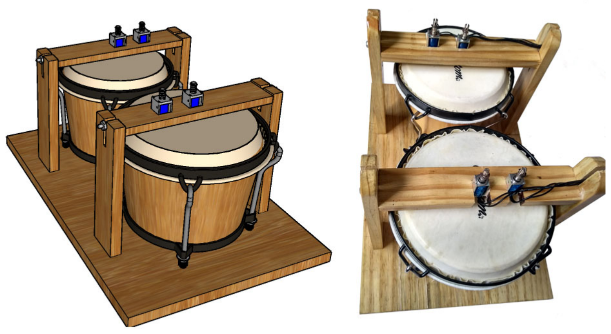

Fig. 1. BongoBot 3D model and real model.

The electromechanical components of the BongoBot were controlled by a Raspberry Pi 3 Model B where all RoboMus communication patterns were implemented. The block diagram of the robot is shown in Fig. 2 (a). However, an auxiliary circuit was necessary to control the solenoids because Raspberry's digital output pins do not supply the voltage and current required by the solenoids (12V and 300 mA). Thus, a driver circuit was implemented for each solenoid where each circuit works as a switcher. Its scheme is shown in Fig. 2 (b). A piezoelectric sensor was placed under each skin of the bongo to capture its sound. This approach allows the Raspberry to detect beats and estimate the robot's mechanical delays. Furthermore, we use an amplifier circuit (Fig. 2 (c)) to allow the Raspberry to read weak beats.

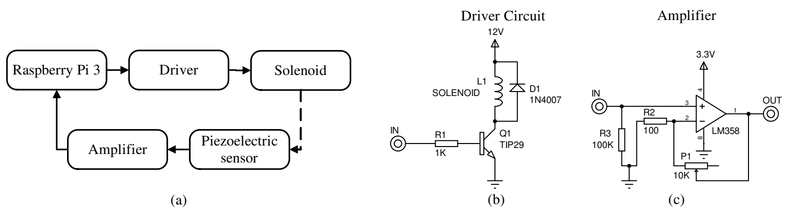

Fig. 2. (a) The robot system diagram, (b) driver circuit, and (c) amplifier circuit.

The BongoBot has two actions following the RoboMus format. The first action is to play the larger drum (*hembra*) with OSC address denoted by */playHembra*. The other is to play the smaller drum (*macho*) where the OSC address is */playMacho*. These actions have the same parameters, which are: *identifier*, the message identification number; *solenoid*: which solenoid to play, where 0 represents the solenoid placed in the center and 1 symbolizes the solenoid fixed on the side; and *intensity*: the beat intensity, where 0 is the lowest and 1 is the highest intensity.

3.1 Methodology for Macro-synchronization Evaluation

Each action message from the MMSS contains an absolute time to be executed. Thus, clock synchronization between robots is the first step to achieve a great performance. We implemented the SNTP algorithm in the MMSS and it worked as the master clock. The setup for the clock synchronization test is shown in Fig. 3 in which two Raspberry Pi 3 model B were used to represent two robots. Thus, the MMSS sent hundreds of messages to the devices (the same messages for both). When the internal clock of a Raspberry matches with the message's time tag, it sends a digital pulse from the output pin. An Arduino Mega and a PC were used only to evaluate the delay between the Raspberry devices. Then, the Arduino reads the pulses from the Raspberry devices and calculates the time difference between the pulses, which consequently represents the difference between their internal clocks.

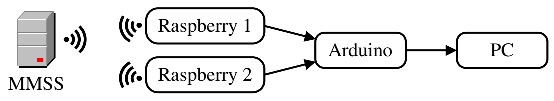

Fig. 3. Experimental setup to measure the difference between devices' clocks.

After clock synchronizations, we evaluated the mechanical delay compensation learned by neural networks. However, at least two robots are necessary for this task. Thus, we divided the BongoBot into two robots with one drum each. One Raspberry controls the larger drum and another Raspberry controls the smaller drum. The test setup is described in Fig. 4. An Arduino Mega and a PC was used only to evaluate the delay. The Arduino reads, separately, signals from each piezoelectric sensor placed under the drum skins and then estimates delay between the robots' beats. Thus, if the two Raspberry Pi devices receive a message from the MMSS with an absolute time of 9:00:00.000, each Raspberry will wait until its internal clock marks that time to send a signal to control the solenoid and play its drum. When a drum is played, the piezoelectric sensor will generate a signal that the Arduino will read and calculate the delay.

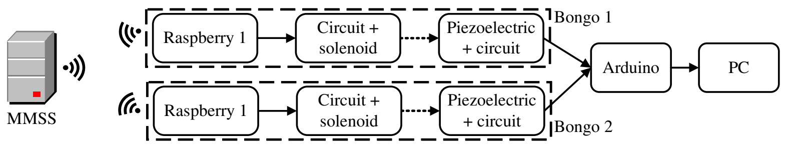

Fig. 4. Experimental setup to measure the delay between the robots' beats.

4 Experimental Results

4.1 Clock Synchronization

Following the setup described in Fig. 3, Fig. 5 (a) shows the results for 1000 messages where the robots synchronized their clocks, using SNTP, only once at the beginning of the test. The delay increment over time is caused by clock inaccuracy. This effect is known as clock drift. Thus, to solve the problem, a new synchronization was implemented every 30 seconds. Fig. 5 (b) depicts a test with 1000 messages where abrupt transitions occur when the SNTP algorithm is executed. However, the maximum delay value of the test was $18.81ms$ and the difference mean value is about $7.82ms$ with the standard deviation equal to $5.55ms$.

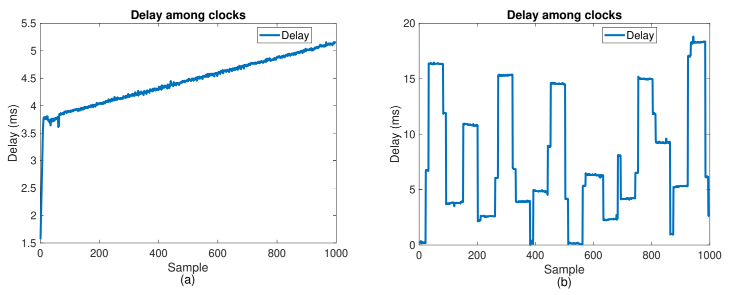

Fig. 5. Delay between two Raspberry Pi devices: (a) shows the results of the test with clocks synchronization, only once, at the beginning, showing the drift clock effects and (b) depicts the synchronization results applying the SNTP from time to time.

4.2 Macro-synchronization between Bongos

As explained in Subsection 3.1, the BongoBot was divided into two robots. Because they use the same solenoid model, they have a similar mechanical delay. Thus, for macro-synchronization studies, we applied strong beats (intensity from 0.8 to 1) for the robot 1 *hembra* which will spend less time to be executed. For the robot 2 *macho*, we implement weak beats (intensity from 0 to 0.2) because the robot will spend more time to executed. Therefore, this approach forces a possible lack of synchronism between the robots. Using the setup described in Fig. 4, two tests with 1000 random messages for each robot were developed and analyzed, where the messages had the same time tag and different intensities.

The first test (test 1) can be seen in Fig. 6 (a). Test 1 presented the average delay equal to $10.88ms$, the standard deviation equal to $5.75ms$, and 8.5% of the beats with delays above $20ms$ (human perception threshold). For test 2 without delay compensation as well, the results are illustrated in Fig. 6 (b) with

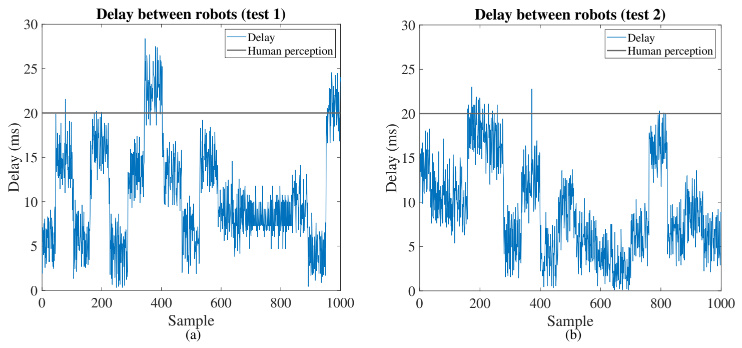

Fig. 6. Delay between two robots without delay compensation: (a) shows the results for test 1 and (b) depicts the results for test 2.

the average delay equal to $9.15ms$ and the standard deviation equal to $5.17ms$, where 2.2% of beats had delays above $20ms$.

Each mechanical delay profile was learned by a perceptron multilayer neural network (Haykin, 2001; Rosenblatt, 1957) which the model is shown in Fig. 7. The same neural network structure was used for both robots due to their similarity. However, each network was trained specifically for each one. The MMSS used the specific protocol to send random messages to a robot and this robot returns the delay spend on each message. Thus, the MMSS can train the neural network automatically. For both robots, the MMSS generated 1500 random messages that represent the dataset. The dataset was divided into test data, with 20% of the dataset, and training data with 80%. Both neural networks were trained with batch size = 32 and epochs = 256. The optimizer chosen was Adam. The neural network for robot 1 (larger drum) reached an RMSE (root mean square error) value equal to 0.82% and robot 2 (smaller drum) obtained 1.2%.

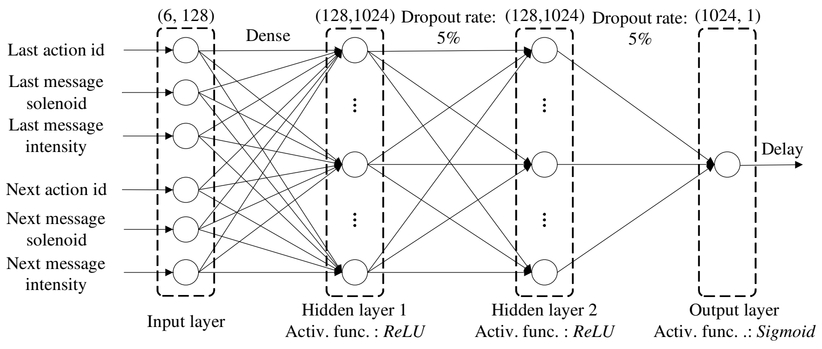

Fig. 7. Specific neural network model used to learn the bongo robots mechanical delays.

Two tests (tests 3 and 4) were performed after training the robots' delay profile. In these tests was applied the delay compensation provided by the neural networks. In other words, a message scheduled to be executed at 10:00:00.000 (sound response time) will be processed by the neural network and the delay result will be subtracted from the scheduled time of the execution. For example, if the predicted delay is $25ms$, the new execution time will be 9:59:59.975. The two tests were done with the same messages used for the tests without delay compensation (tests 1 and 2). However, the mechanical delays were compensated.

The results for test 3 are shown in Fig. 8 (a), where the robots reached an average delay between them equal to $5.58ms$ with the standard deviation of $3.75ms$, and 0% of delay values higher than $20ms$. For test 4, the results are presented in Fig. 8 (b), where all the delay values were lower than $20ms$ and the average delay was equal to $4.86ms$ with the standard deviation of $3.88ms$.

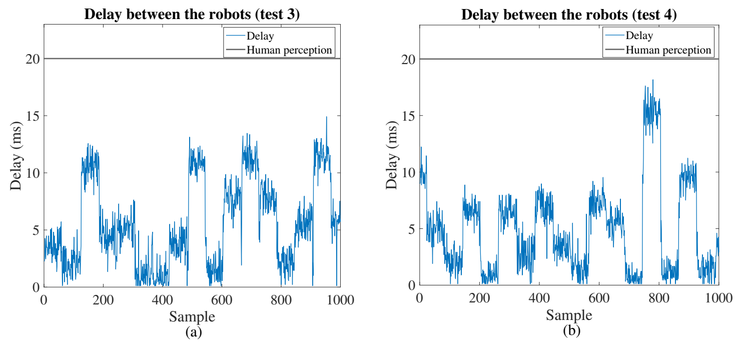

Fig. 8. Delay between the two robots with compensation strategy: (a) illustrates the results for the test 3 and (b) shows the results for the test 4.

As expected, there was an improvement in the average delay, standard deviation, and in the number of beats above $20ms$, once we added the delay compensation. Comparing test 1 with test 4, there was a reduction of 55.33% ($6.02ms$) in the average delay and a reduction of 32.52% ($1.87ms$) in the standard deviation. In addition, the number of beats above $20ms$ was reduced to zero.

5 Conclusions

This paper presented macro-synchronization strategies for the RoboMus platform using robotic bongos in the tests. We implement neural networks to learn the robots' mechanical delay profiles and SNTP (simple network time protocol) for synchronization between clocks. These two approaches showed interesting experimental results where SNTP keeps the delay lower than the human perception and the mechanical delay compensation reduced the average delay between two robots by up to 55.33%.

References

Bretan, M., & Weinberg, G. I. L. (2016). A Survey of Robotic Musicianship. *Communications of the ACM*, *59*(5), 100–109. Retrieved from `http://search.ebscohost.com/login.aspx?direct=true{\&}db=bth{\&}AN=115178367{\&}site=ehost-live{\&}scope=site` doi: 10.1145/2818994

Camporez, H. A. F., Mota, T. S. R., Astorga, E. M. V., Rocha, H. R. O., & Costalonga, L. L. (2018). RoboMus: Uma Plataforma para Performances Musicais Robóticas. In D. Keller & M. H. D. Lima (Eds.), *Aplicações em música ubíqua* (1st ed., pp. 58–93). São Paulo, Brasil: ANPPOM.

Haykin, S. (2001). Perceptron de Múltiplas Camadas. In *Redes neurais - princípios e prática* (2ª ed., pp. 183–282). Porto Alegre: Bookman.

Hoffman, G., & Weinberg, G. (2010). Shimon: An Interactive Improvisational Robotic Marimba Player. In *Chi '10 extended abstracts on human factors in computing systems* (pp. 3097–3102). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/1753846.1753925` doi: 10.1145/1753846.1753925

Jordà, S. (2002). Afasia: the Ultimate Homeric One-man-multimedia-band. In *Proceedings of the international conference on new interfaces for musical expression* (pp. 102–107). Dublin, Ireland. Retrieved from `http://www.nime.org/proceedings/2002/nime2002{\_}102.pdf`

Kapur, A., Trimpin, E., Singer, A., Suleman, G., & Tzanetakis, G. (2007). A Comparison of Solenoid-Based Strategies for robotic drumming. In *Icmc.* Copenhagen.

Keller, D., Lazzarini, V., & Pimenta, M. S. (Eds.). (2014). *Ubiquitous Music* (1st ed.). Springer International Publishing. doi: 10.1007/978-3-319-11152-0

McVay, J., Carnegie, D., & Murphy, J. (2015). An overview of MechBass: A four string robotic bass guitar. In *2015 6th international conference on automation, robotics and applications (icara)* (pp. 179–184). doi: 10.1109/ICARA.2015.7081144

Mills, D. L. (2006). *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI* (No. 4330). RFC 4330. RFC Editor. Retrieved from `https://rfc-editor.org/rfc/rfc4330.txt` doi: 10.17487/RFC4330

Pisoni, D. B. (1977). Identification and discrimination of the relative onset time of two component tones: implications for voicing perception in stops. *The Journal of the Acoustical Society of America*, *61*(May 2013), 1352–1361. doi: 10.1121/1.381409

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton project para.* Cornell Aeronautical Laboratory.

Rottondi, C., Chafe, C., Allocchio, C., & Sarti, A. (2016). An Overview on Networked Music Performance Technologies. *IEEE Access*, *4*, 8823–8843. doi: 10.1109/ACCESS.2016.2628440

Singer, E., Feddersen, J., Redmon, C., & Bowen, B. (2004). LEMUR's Musical Robots. *Proceedings of the International Conference on New Interfaces for Musical Expression*, 181–184. Retrieved from `http://`

www.nime.org/proceedings/2004/nime2004{_}181.pdf doi: 10.1145/1027527.1027569

Singer, E., Larke, K., & Bianciardi, D. (2003). LEMUR GuitarBot: MIDI Robotic String Instrument. *Proceedings of the 2003 Conference on New Interfaces for Musical Expression (NIME-03)*, 188–191.

Turchet, L., Fischione, C., Essl, G., Keller, D., & Barthet, M. (2018). Internet of Musical Things: Vision and Challenges. In *Ieee access* (Vol. 6, pp. 61994–62017). doi: 10.1109/ACCESS.2018.2872625

Weinberg, G., Bretan, M., Hoffman, G., & Driscoll, S. (2020). *Robotic Musicianship: Embodied Artificial Creativity and Mechatronic Musical Expression* (1st ed.). Cham: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-030-38930-7 doi: 10.1007/978-3-030-38930-7

Wright, M., & Freed, A. (1997). Open Sound Control: A New Protocol for Communicating with Sound Synthesizers. *Proc. ICMC 1997*, 101–104.

Yang, N., Savery, R., Sankaranarayanan, R., Zahray, L., & Weinberg, G. (2020). *Mechatronics-driven musical expressivity for robotic percussionists.*