Humanoid Robot Dance Control over Real-Time Music Stimulation

AU332 Project Final Report

Jingwei Zhao, Zhengbao He
Shanghai Jiao Tong University, Shanghai, China
{zhaojw, lstefanie}@sjtu.edu.cn

Abstract: In this project, we implement a novel dance control system for humanoid robot over real-time music stimulation. We generate dance motions for the robot from RGB human dance videos by estimating 3D human pose, which are further analytically mapped to corresponding robot joint angles to actuate the robot with the motions. With a motion base established in this way, we choreograph for the robot by selecting motions following a Markov Chain rule. To synchronize the motions to the music beats, we apply beat tracking algorithms in real time to extract beat claims as reference, and adjust the dance speed by feeding the motion-beat error into a PID controller. We test and evaluate our control system with the virtual robot Nao in the CoppeliaSim platform, which not only demostrates varied and visually pleasing dance motion sequences while keeping itself stable and ballanced, but also displays a tight synchronization between the motions and music beats.

Key Words: Humanoid Robot, Dance Control, Real-Time Music Stimulation

1 Introduction

With the emergence of humanoid robots, there has been more and more instances where robots imitate human activities and interact with humans. Related work includes facial expression imitation[1], human pose imitation[2–4], and human motion imitation[5–7], etc. Since dance, as a classical performance art form, has been part of human social interaction for multiple millennia, enabling robots to imitate human dance has also become an actual field of research.

There have been various contributions to the field of dancing robots. Early work such as [8, 9] actuate their robots to display impressive dancing capabilities by precisely pre-programming the sequence and timing of dancing motions (i.e., choreography) to a set of fixed musical pieces. However, they cannot respond to external musical stimulation as humans do. As for [10–13], their robots dance to external auditory or visual sitimulations, but are limited with single, periodic dance motions.

Besides motion design, the hardware and software delay presents another difficulty in controlling robot dance. Specifically, the robot's motion output tends to lag the input due to delays in arithmetic operations and data transmission. To stress this problem, [14] combines a feedback PID controller with a feedforward lookup-table-based controller to synchronize the key frames of dancing motions to the music beats. This work also implements non-programmed choreography in correspondence with real-time music stimulation, but each of its motion primitives is still pre-programed, and the robot used here, namely, ANYMal[15], is not humanoid.

Our goal with this project is to enable a humanoid robot dance to real-time music stimulation, with non-programmed dancing motions which can be easily generated, and which are both varied and visually pleasing. The robot we select here is Nao produced by Softbank Robotics. Algorhithms are tested in the CoppeliaSim[16] platform, which is the newest version of Vrep[17].

To let the robot dance in a non-programmed way, we utilize 3D human pose estimation to extract 3D human pose of dancers from videos. The 3D pose is later mapped to corresponding robot joint angeles in an analytical way so that the robot can imitate the dancing motions. Since information about ankle joints are not recorded in the 3D human pose, we further analytically keep the robot's feet always parallel to the ground. This also keeps the robot in static stability, thus we can let it imitate more sophisticated motions.

We set a motion base for the robot by extracting dancing poses from a number of videos. Each motion is consisted of a number of motion frames, which corresponds to the video frames from which the motions are extracted. To let the robot dance with real-time music, we follow [14] and take advantage of the beat tracking algorhthms implemented by [18] to extract real-time music beats. To choreograph, we select dancing motions in an improvised way from the motion base with a Markov Chain rule. To synchronize the motions with the beats, we implement a PID controller in a feedback path to let the robot follow the music tightly.

In order to further stress the influence of time delay due to data transmission, we pack our whole control system into three mutually communicating processes, each taking charge of specific tasks. These processes are, namely, Beat Extractor, Motion Selector, and Speed Controller and Motion Actuator, respectively. An overall diagram is shown in Figure 1.

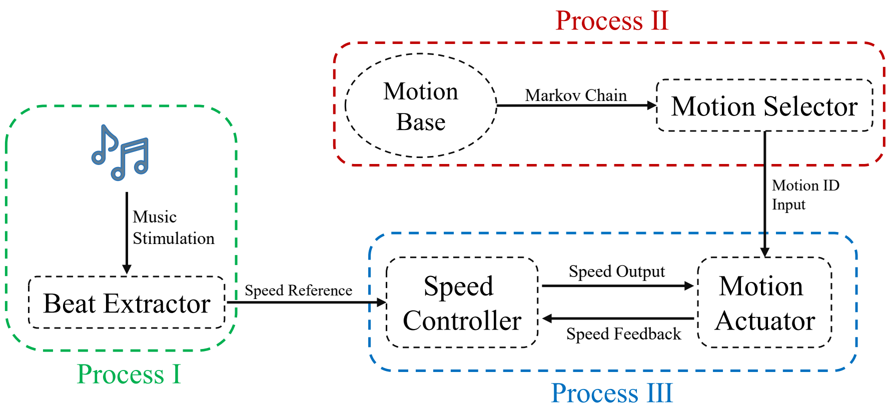


Fig. 1: Overall Diagram of the Robot Dance Control System

The rest of this report is structured as follow. We introduce our work in Section 2. Specifically, motion generation, motion selection, and speed control are introcuced in Section 2.1, 2.2, and 2.3 respectively. In Section 3, we integrate each part, and evaluate the overall performance of our dance control system with robot Nao in the CoppeliaSim platform. Finally, we discuss our contribution and weakness in Section 4.

2 Our Work

2.1 Motion Generation

In this part, we introduce the way we generate dance motions for the robot. We first estimate 3D human pose from dancing videos, and then map the pose locations into robot joint angles. A motion base is established by applying this process to many dance video clips. Each motion consists of a number of motion frames which are corresponding to the video frames from which they are extracted. The merit of our method is that, once the pipline is built, few codes are needed for the generation of new motions. Details are illustrated as follow.

2.1.1 3D Human Pose Estimation

We estimate 3D human pose from RGB videos, most of which are solo dance episodes collected from YouTube. Specifically, for each video frame, we first crop the dancer out with Yolo v3[19] by detecting the 'person' label, and then feed the detected image patch to ResPoseNet raised by [20] for 3D human pose estimation.

The reason why we use Yolo to detect the dancer before pose estimation is that it helps ResPoseNet to concentrate on the dancer by putting the dancer in a whole image patch. Thus the 3D pose we obtain can be more accurate. With this method, we can also select certain dancer from multi-person dance videos to estimate poses, because Yolo v3 supports multi-target detection. However, We haven't tried such multi-person videos because we have found a number of good-quality solo dance ones. Thus we just input these videos, obtain the most confident framewise detection patch from Yolo, and then feed the detection to ResPoseNet for 3D human pose estimation. This further helps keeping a higher estimation accuracy. The whole procedure for this part is illustrated in Figure 2.

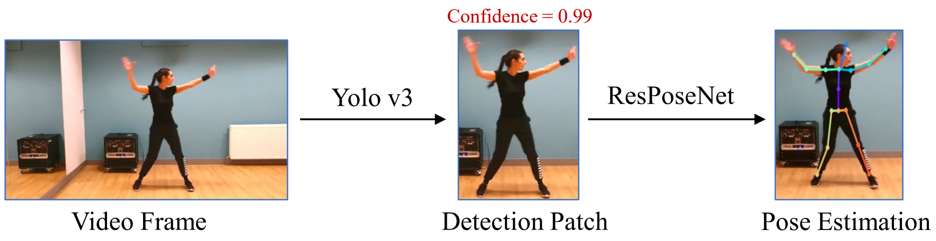


Fig. 2: Procedure of 3D Human Pose Estimation

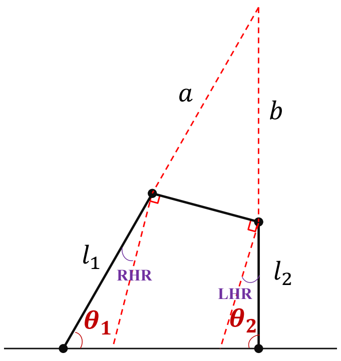
2.1.2 Mapping from Pose to Angle

Obtaining the pose information for each dance motion frame only partly solves our problem, because the 3D pose cannot be directly used to actuate the robot. In our project, to actuate Nao, we have to map the pose to Nao's joint angles. Nao has totally 25 free joints throughout its body. Since what we focus on is mapping dance motions, we neglect the joints for the neck, wrist, and fingers for simplicity. However, the rest work is still a rather huge systematic procudure.

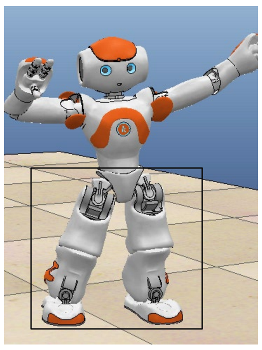
For mapping to the angles of Nao's joints of shoulders, elbows, hips, and knees, we follow [3] and implement an anylitical mapping method. Specifically, for each motion frame, each pose key point on the limbs can find its corresponding joint on Nao. By expressing the links as vectors from one key point to another, and applying dot and cross multipication with each other, the angles between two adjacent links can be obtained, and this angle can easily find its mapping to Nao's body joint angles. In practice, we found that the mapped results for the shoulder pitch angles tend to be very large. After confirming that the mapping method was correctly implemented, we concluded that this phenomenon should have something to do with ResPoseNet's own features. As a result, we multiply a factor smaller than one to these angles in order to keep visually pleasing dance motions.

Another problem to tackle is that, the information for ankel joint angles are not recorded in the 3D human pose. Fortunately, we have the prior heuristic that the robot should keep stable and ballanced when making each dance motion. As a result, we try to keep its feet always parallel to the ground. To do so, we first simplify and model Nao's lower limbs analytically.Because it is rather difficult to do the modeling in 3D space, we map the 3D space into a vertical and a horizontal 2D plane for simplification. This mapping is kind of approximation, and the value we obtain could deviate from the true value, but the results we get in practice is quite feasible.

For example, in the instance of Figure 3(b), we model Nao's lower limbs (inside the black box), and get the approximation in Figure 3(a). l_1 and l_2 are the distances from Nao's hip joints to ankle joints projectd to the vertical plane, and RHR and LHR are the angles of *RightHipRoll* and *LeftHipRoll*, respectively.



(a) Analytical Model



(b) Nao's Lower Limbs

Fig. 3: Analytical Model for Ankle Joint Mapping

To control Nao's ankle joints while keeping it stable and ballenced, we need to obtain the value of θ_1 and θ_2 in Figure 3(a). And analytically, we have:

$$\theta_1 = \arctan\left(\frac{\sin(LHR - RHR)}{\frac{a+l_1}{b+l_2} - \cos(LHR - RHR)}\right) \tag{1}$$

$$\theta_2 = \arctan\left(\frac{\sin(LHR - RHR)}{\frac{b+l_2}{a+l_1} - \cos(LHR - RHR)}\right) \tag{2}$$

With θ_1 and θ_2, we can calculate corresponding values for the joint angle *RightAnkleRoll* and *LeftAnkleRoll*. A similar method can be applied to getting another pair of angles, namely, *RightAnklePitch* and *LeftAnklePitch*. In this way, we keep Nao's feet always parallel to the ground. Thus Nao can maintain stability while making each dance motion.

In practice, probabily due to the deviation brought by our approximation, the robot could still fall down when the motion is fierce. So we limit the range of angle *RightKneePitch* and *LeftKneePitch*, and let Nao dance mainly along the vertical 2D plane. In this way, we can let Nao make both visually pleasing and stable and ballanced dance motions.

2.2 Motion Selection

With the process introduced in the last part, a motion base is established by applying the algorithms to many dance video clips. Each motion consists of a number of motion frames which are corresponding to the video frames from which they are extracted.

While we try to make our robot dance with varied motions, the motion sequence or choreography it follows should be logical in some sense. For example, certain motions may have a symmetric counterpart, and a left motion should be followed by a corresponding right motion in this case. Also, it may be resaonable to repeat certain motions several times, while not reasonable for other motions. As a result, specific features should be concidered for each motion when deciding which motion to choose next from the motion base.

In our practice, for each motion with ID n, we set the following features:

- BPS_n: BPS is abbreviation for $BeatPerSecond$. This feature is an interval which designates the bound of music tempo at which motion n can run. Currently, the music clips we use to test our algorithms are of similar tempi, which are between this interval for all the motions. So we have not brought this feature into use yet. In our future work, we will rely on this feature to filter potential motions which could detereorate the robot's stability at certain tempo intervals.

- $Repeat_n$: This feature is the probability for which the next motion will repeat the current one. It ranges from 0 to 1, where 0 indicates that motion n will never repeat itself, and 1 indicates that motion n will always repeat itself once trigered. Currently, most of our motions have this value ranging from 0 to 0.5.

- $Symmetric_n$: This feature could be an integer number or the bollean value $False$. $False$ indicates that motion n does not have a symmetric counterpart. An integer number indicates motion n is symmetrc, and the number is extactly the ID of the corresponding symmetric motion. In practice, we follow the rule that left is prior to right, and we always assign odd ID to left motions and even ID to right motions in order to tell different cases. We let a left motion (n is odd) jump to its right counterpart $Symmetric_n$ with probability $= 1$ ($Repeat_n = 0$), and let a right motion (n is even) jump back to its left counterpart $Symmetric_n$ with probability $= Repeat_n$.

- FPB_n: FPB is abbreviation for $FramePerBeat$. This is an positive number which indicates the average number of frames between two consecutive key motion frames, where the 'key motion frames' are defined as the motion frames aligned with the beat of the soundtrack of its originated video. This feature is used to control the motion actuating speed, and will be illustrated in the next part.

Considering the features shown above, we establish our motion selector as a probabilistic model. Such a model does not pre-program the choreography with candidate motions, but generate motion sequence in an immprovised way. In this way, our robot can dance in varied patterns. Specifically, the motion selection process is designed as a Markov Chain. A Markov Chain is a sequence of discrete random variables X_1, X_2, X_3, … with transition probabilities of the next state given by the current one, i.e.,

$$P(X_{n+1} = x|X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = P(X_{n+1} = x|X_n = x_n) \tag{3}$$

The possible values x_1, x_2, …, x_i of X_n are candidates of motion IDs for the n^{th} motion. The transition probability $P(X_{n+1} = x_i|X_n = x_j)$ for the n^{th} motion is determined by the features BPS_n, $Repeat_n$, $Symmetric_n$, and the current tempo of the music BPS_{music}. Conceptually, We impliment our motion selector as shown in Algorithm 1, where the motionBuffer Q is a data trasmission queue from Motion Selector to Motion Actuator. While the former puts selected motion ID in, the latter takes it out when a new motion is wanted.

Algorithm 1: Algorithm of Motion Selector

```
Input:  motionBase B, initialMotion ID n = 0, motionBuffer Q
Result: Iterate and update n
1  Q ← n;
2  while Dance not terminated do
3      if Q is empty then
4          $BPS_n, Symmetric_n, Repeat_n \leftarrow features$(motion n);
5          if $Symmetric_n \neq False$ and $Symmetric_n$ is divisible by 2 then
6              n ← $Symmetric_n$;
7          end
8          if $Symmetric_n \neq False$ and $Repeat_n \neq 0$ then
9              n ← $Symmetric_n$ with probability $Repeat_n$;
10         end
11         if $Symmetric_n = False$ and $Repeat_n \neq 0$ then
12             n ← n with probability $Repeat_n$;
13         end
14         if n is not updated yet (including by itself) then
15             Randomly choose motion m from B with $Symmetry_m$ divisible by 2;
16             n ← m;
17         end
18         Q ← n;
19     end
20 end
```

With the algorithm implemented above, our robot dances with real-time music in an improvised fashion, while consecutive dance motions are logically reasonable. The choreography it follows is not pre-programmed, but generated in real time according to features of the previous motion and the current music tempo. In this way, the dancing capability of our robot is further enhanced.

2.3 Speed Controller

Till now, we have established the motion base with 3D human pose estimation and analytical mapping. We have also implemented a Markov Chain based motion selecting algorithm with which varied and logical motions are selected as choreography. The next problem we must take care is that, the robot should actuate each dance motion with accurate synchronization to the music beats.

To settle this problem, we first need to obtain the true beat information as reference. Following [14], our real-time beat extractor is based on the method proposed in [18], where we use the full implementation provided by the madmom [21] library. Given an audio input, an RNN network is first introduced to handle recursively cutted audio frames, and produce beat activations which indicate how possible an audio frame corresponds to a music beat. And then a dynamic Bayesian Network is utilized to handle the whole activations to finally make the music beat claims. The audio frames the madmom beat tracker processes are 10 ms long, which results in an update rate of 100 Hz. This is short enough not to violate the real-time constraint, while still long enough to keep sufficient data for frequency analysis.

With the beat extracting algorithm, besides extracting real-time music beats, we can also extract soundtrack beats of the dance videos along with the motion generating process. In order to take advantage of the soundtrack, we add beat information to the motion frames in this way. Specifically, each motion frame records the distance (in number of frames) to the next beat and to the last beat respectively. These two indices are named $beat_{i,n}^{next}$ and $beat_{i,n}^{last}$ for frame i in motion n. If frame i is the closesd to one certain beat, then it has $beat_{i,n}^{next} = beat_{i,n}^{last} = 0$

When we actuate the robot, the main factor which lags the motion from the beat is the time delay in data transmission, which takes place when we send joint angle information to the CopelliaSim platform. Although we can record this delay and eliminate its influence by introducing compensating terms, there could still be unpredicted causes which deviates our dance motion from the music beat. As a result, real-time speed control mechanism for the dance motions is direly needed.

Since the robot is actuated by sequential motion frames, the dance speed is controlled by the time sleep between consecutive motion frames. Initially, for motion n, the time sleep τ_n is set as:

$$\tau_n = \frac{1}{BPS_{music} \times FPB_n} - \delta \tag{4}$$

where BPS_{music} is the tempo of the music, calculated by the duration between consecutive music beats; FPB_n is the value of the features of motion n, which is introduced in Section 2.2; δ is the compensating term to eliminate the influence of data transmission.

We further utilize a PID controller to synchronize our motion to the beat. The reference input is the beat claims generated by our beat extracting algorithm. When the K^{th} beat is claimed, $beat_{i_k,n}^{next}$ and $beat_{i_k,n}^{last}$ of the current frame i_k of current motion n is recorded. Then the error term e_{i_k} is determined as follow:

$$e_{i_k} = \begin{cases} beat_{i_k,n}^{next} & beat_{i_k,n}^{next} \leq beat_{i_k,n}^{last} \\ -beat_{i_k,n}^{last} & beat_{i_k,n}^{next} > beat_{i_k,n}^{last} \end{cases} \tag{5}$$

In this way, a feedback compensation term γ is calculated as:

$$\gamma_{i_k,n} = K_p e_{i_k} + K_i \sum_{i=1}^{K} e_{i_k} + K_d(e_{i_k} - e_{i_k-1}) \tag{6}$$

where K_p, K_i, and K_d are the controller parameters.

By calculating and employing this term at each beat claim, our dance control system is no more sensitive to the unpredicted time delays.

3 Performance evaluation

Till now, we have implemented the basic componets we need to build our dance control system for humanoid robot over real-time music stimulation. In this part, we will integrate these components into the whole system, and evaluate its performance.

3.1 Component Integration

With our work introduced in Section 2, we can establish three basic components for our system, namely, Beat Extractor, Motion Selector, and Speed Controller. Also, a Motion Base is built to provide candidate dance motions. To actuate the robot, a new component, Motion Actuator, is required to transmit each motion frame to the CoppeliaSim platform in order to really actuate the robot.

Ideally, to avoid time delay due to inefficient cooperation, the four components mentioned above (except the motion base, which is not an operating component) should be implemented in four different processes, working individually at the same time. However, to realize it, a father process is further needed, leading to totally five processes working individually, which is beyond the capability of our four-core laptop CPU. To compromise, we combine Speed Controller and Motion Actuator in a single process, and implement three processes working individually at the same time (in fact, totally four processes if the father process is counted). The overall system diagram is illustrated in Figure 1. With this configuration, we evaluate our system's performance by simulating a dance control trial with Nao in CoppeliaSim.

3.2 System Performance

3.2.1 Motion and Choreography

A sequence of screenshots of one improvised dance instance is displayed in Figure 4. A demo video is also attached to the portfolio submitted together with this report. All of them are the true results of our dance control system. In Figure 4, We can see that the improvised choreography satisfies our expectation. Specifically:

- The humanoid robot, Nao, keeps its feet parallel to the ground for each motion of the leg. In this way, Nao stays stable and ballanced during the whole dance.
- The choreography shows certain rationality. In the two middle frames in Figure 4, a right motion is executed after its left counterpart. This symmetric pattern is reasonable and common in human choreography, which accords with the inherent aesthetic logic of the art form, dance. By this way, we make the robot dancer more similiar to real human dancers.
- The robot shows varied dance motions, instead of single periodic motions as in the early works. In fact, the dance instances are quite visually pleasing.

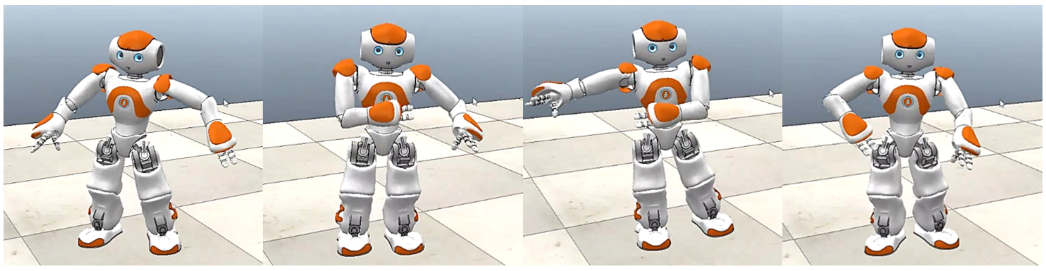


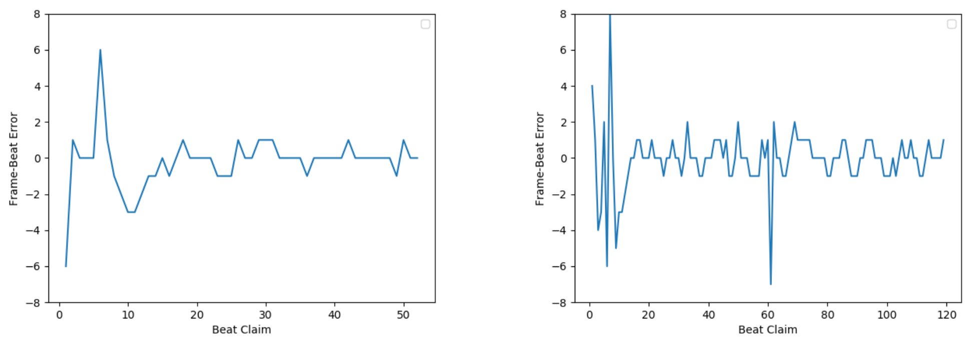
Fig. 4: An Improvised Dance Instance

3.2.2 Synchronization to the Music Beats

Besides the visual experience in the dance instances, it is also important to evaluate the motion-beat synchronizing capability of our system. We expect our system to control the time sleep between consecutive motion frames properly, and align key motion frames with the beat claims as accurately as possible.

We first evaluate the speed response when a dance instance starts. Specifically, We play a 120-bpm music piece and let the robot dance with it, during which we record the frame-beat error e_{i_k} introduced in Section 2.3 for motion frame i_k which is recorded at k^{th} beat claim, $k = 1, 2, \ldots$. The result is shown in Figure 5(a). We can see that after the music goes off, the frame-beat error is rather large (beyond ± 5). However, after around 10 beat claims, which is approximately equivalent to 5 seconds, the error is successfully suppressed, and stays within the range between -1 and $+1$. That is to say, with only occasional ± 1 beat deviations, our key motion frames are almost precisely aligned with the beat claims.

Then we also test our robustness to music tempo change. We first play a 120-bpm music piece, and then switch to a different 150-bpm piece, during which the robot keeps dancing, and we record the frame-beat error e_{i_k} which produces the results in Figure 5(b). During the first piece, it still takes around 10 beat claims for the frame-beat error to stablize with ± 1 range. The tempo switch takes place at around the 60^{th} beat claim. Because the tempo actually increases, we can see that the frame-beat error suddenly drops deeply negative. This indicates that the current frame lags the beat claim. However, we can see that our system responds quite quickly to this disturbance, and within 5 beats ahead, it returns back to the stable and low-error level. This result illustrates that the synchronization mechanism of our system is robust to disturbance brought by music tempo change. Hence our robot dance performance generalizes well to less strictly organized music.



(a) Real-Time Error

(b) Response to Tempo Change

Fig. 5: Test Results for Motion-Beat Synchronization

The experiments discussed above demostrates two merits of our system. First, our system generate both varied and visually pleasing dance motions while keeping the robot stable and ballanced. Second, our system enables the robot to follow the music tightly, by accurately synchronizing key motion frames to the music beats.

4 Conclusion

In this project, we implement a novel dance control system for humanoid robot over real-time music stimulation. We use the robot Nao in the CoppeliaSim platform to test and evaluate the performance of our system, and obtain excellent results.

Our contributuin in this project is that, firstly, to our knowledge, we are the first to utilize 3D human pose estimation methods to generate robot dance motions; also, our Markov Chain based choreography method and PID based speed control method realize satisfied effects, and thus can be further studied and improved in future work.

However, we also see some limitations in our work. The performance of our work is greatly dependant on the performance of pose estimation algorithms and beat tracking algorithms. Thus our work cannot generalize to every dance video and music piece, because those algorithms are only trained over specific datasets. Also, the analytical method we take to keep the robot stable and ballanced is limited because it limits the robot's motions.

In our future work, we will select more easily-generalized dependency algorithms to further boost the robustness of our dance control system. Besides, we will try utilizing Reinforcement Learning to keep the stability of the robot in a more efficient and accurate way. We look forward to our work promoting robots' dance capabilities, as well as their interation with human beings.

Acknowledgement

This article is the project report by student Jingwei Zhao and Zhengbao He for AU332 Aritificial intelligence tutored by Professor Yue Gao. During the working process of this project, Jingwei Zhao mainly takes charge of motion extraction, beat extraction, multiprocess implementation, and writing of the final report, while Zhengbao He sees to analytical mapping, motion selection, speed control, and revision of the final report. We sincerely cooperate with each other, and finally obtain some fruitful results. This report is formatted with the article template of China Conference on Sound and Music Technology (CSMT).

We are extremely grateful for Professor Gao's help and guidance to our program, as well as in the AI lectures. It is Professor Gao who lead us into the gate of Artificial Intelligence, and let our vision extend further into the vast future. We are also greatly thankful to TA Ming Sun and Summeryuki. We would not have learnt so far without their contributions. Finally, we are especially thankful to Mr. Buqing Nie, who kindly offered us with a great deal of help when we initially dived into the unknown field of this project.

Reference

[1] S. S. Ge, C. Wang, and C. C. Hang, "Facial expression imitation in human robot interaction," in *RO-MAN 2008-The 17th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 213–218, IEEE, 2008.

[2] Z. Zhang, Y. Niu, Z. Yan, and S. Lin, "Real-time whole-body imitation by humanoid robots and task-oriented teleoperation using an analytical mapping method and quantitative evaluation," *Applied Sciences*, vol. 8, no. 10, p. 2005, 2018.

[3] J. Lei, M. Song, Z.-N. Li, and C. Chen, "Whole-body humanoid robot imitation with pose similarity evaluation," *Signal Processing*, vol. 108, pp. 136–146, 2015.

[4] Z. Zhang, Y. Niu, L. Kong, S. Lin, and H. Wang, "A real-time upper-body robot imitation system,"

[5] S. Kim, C. Kim, B. You, and S. Oh, "Stable whole-body motion generation for humanoid robots to imitate human motions," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2518–2524, IEEE, 2009.

[6] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa, and K. Ikeuchi, "Generating whole body motions for a biped humanoid robot from captured human dances," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 3, pp. 3905–3910, IEEE, 2003.

[7] J. Koenemann, F. Burget, and M. Bennewitz, "Real-time imitation of human whole-body motions by humanoids," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2806–2812, IEEE, 2014.

[8] L. Geppert, "Qrio, the robot that could," *Ieee Spectrum*, vol. 41, no. 5, pp. 34–37, 2004.

[9] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, and K. Ikeuchi, "Task model of lower body motion for a biped humanoid robot to imitate human dances," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3157–3162, IEEE, 2005.

[10] C. Crick, M. Munz, and B. Scassellati, "Synchronization in social tasks: Robotic drumming," in *ROMAN 2006-The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 97–102, IEEE, 2006.

[11] S. Bock, F. Krebs, A. Durand, S. Poll, and R. Balsyte, "Robod: a real-time online beat and offbeat drummer," 2016.

[12] K. Murata, K. Nakadai, K. Yoshii, R. Takeda, T. Torii, H. G. Okuno, Y. Hasegawa, and H. Tsujino, "A robot uses its own microphone to synchronize its steps to musical beats while scatting and singing," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2459–2464, IEEE, 2008.

[13] A. Schöllig, F. Augugliaro, S. Lupashin, and R. D'Andrea, "Synchronizing the motion of a quadrocopter to music," in *2010 IEEE International Conference on Robotics and Automation*, pp. 3355–3360, IEEE, 2010.

[14] T. Bi, P. Fankhauser, D. Bellicoso, and M. Hutter, "Real-time dance generation to music for a legged robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1038–1044, IEEE, 2018.

[15] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 38–44, IEEE, 2016.

[16] E. Rohmer, S. P. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework,"

[17] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321–1326, IEEE, 2013.

[18] S. Böck, F. Krebs, and G. Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles.," in *ISMIR*, pp. 603–608, 2014.

[19] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[20] G. Moon, J. Y. Chang, and K. M. Lee, "Camera distance-aware top-down approach for 3d multi-person pose estimation from a single rgb image," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 10133–10142, 2019.

[21] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "Madmom: A new python audio and music signal processing library," in *Proceedings of the 24th ACM international conference on Multimedia*, pp. 1174–1178, ACM, 2016.