Synchronized Human-Humanoid Motion Imitation

Antonin Dallard, Mehdi Benallegue, Fumio Kanehiro, Abderrahmane Kheddar

Synchronized Human-Humanoid Motion Imitation

Antonin Dallard, Mehdi Benallegue, Fumio Kanehiro and Abderrahmane Kheddar, *Fellow, IEEE*

***Abstract*— We present a tele-operation control framework that (i) enhances the upper motion synchrony between a user and a robot using the minimum-jerk model coupled with a recursive least-square filter, and (ii) synchronizes the walking pace by predicting user's stepping frequency using motion capture data and a deep learning model. By integrating (i) and (ii) in a task-space whole-body controller, we achieve full-body synchronization. We assess our humanoid-to-human whole-body synchronized motion model on the HRP-4 humanoid robot in forward, lateral and backward walks with concurrent upper limbs motions experiments.**

I. Introduction

Synchronized human motion such as those witnessed in artistic swimming or dancing choreography performances come after an intensive technical training. Those motions are not only precisely predefined but they are strictly timed with auditive or visual rhythms to ensure synchrony. Much looser forms of synchrony occur in other social contexts [1].

On the other hand, making robots produce synchronized motions is 'simpler'. This has been demonstrated with a large number of small size humanoids[1] or quadrupeds[2] or both[3]. Moreover, it can be done even without predefined trajectories with good network communication protocol and dedicated controllers, see e.g., [2], [3].

Yet when it comes to synchronizing motions between a human and a (humanoid) robot, the problem becomes more challenging, e.g., [4]. When the trajectory is predefined, we need to account for various discrepancies such as dissimilar anthropomorphic features of all kind (i.e., in terms of size, range of motion, degrees of freedom, torque limitations, etc.). Finally the human has to be able to perform motions perfectly to ensure synchrony during the execution, see the example of the dance performance reported in [5], [6].

However, to our best knowledge none existing work has tackled the problem of whole body *online* motion synchronization, which is certainly a type of imitation but where the start and end of critical motion timing prevails and motion lag of both actors is to be brought close to zero. Yet, we report in Sec. II few exceptions having links to this problem.

We suggest that human-humanoid synchronized motion imitation requires the following ingredients:

- knowledge on the human motion, which can be provided by any low-lag human tracking technology[4,5];
- anticipating on the start of a new motion, i.e., from a zero velocity state limbs; because of the causality of the motion, it is impossible to bring the start-motion lag between the human and the robot to zero, unless an early intention of the motion can be observed, e.g., using brain computer interfaces [7];
- anticipating on the end-time of an ongoing motion; here a good prediction can theoretically put the lag in the motion synchronization to nil.

To realize human-humanoid synchronized motion, we devise two separate imitation-mapping strategies. The first deals with reproducing arm and attitudes synchronized imitation, Sec. III. The second focuses on steps synchronized motion, that is to say, humanoid footstep and walking are to be synchronized to those of the human, Sec. IV. Experiments with the HRP-4 humanoid robot assess the effectiveness of our approach together with its current limitations, see Sec. V.

II. Background and Problem Statement

A. Whole Body Imitation

Achieving motion imitation between a human and a robot relies first on obtaining a similar posture between both entities. Kinematics retargeting, in the tasks space or in the joint space (or hybrid) [8] allows a humanoid robot to mimic a human posture. For floating base robots, balance must be considered and applied in the control scheme, either as a balance constraint [9] or as an imitation of the human state. In [10], walking imitation is modeled by estimating the dynamics of a demonstrator from measurements to match the foot location; the driving idea is that the demonstrator and imitator walks use similar controller which input (for the robot pace) is determined from observing the same controller output (from the human). Recently, [11] also used this assumption by analyzing a simplified dynamics of a human in order to make a robot track it.

B. Motion Prediction

The motion of the human has been extensively studied [12] and allowed to derive from the motion of the humans limbs some invariant properties. One of those properties gives models for hands trajectory generation such as the minimum jerk model. This latter state that the hand motion always tend to minimize the mean square jerk (derivative of acceleration).

In addition to those properties, the motion prediction has also been tackled as a whole body motion generation using the past movement. This kind of prediction can rely on Deep Learning methods using time-sensitive model [13] which are commonly used in robotics to plan a path [14] or adapt it for safety interactions [15].

Finally, if the motions to be predicted tend to be more specific and can be pre-recorded offline as sets of trajectories,

[1] https://www.youtube.com/watch?v=InYsET-v7wM
[2] https://www.youtube.com/watch?v=7atZfX85nd4
[3] https://www.youtube.com/watch?v=G9p9jdmJQOQ
[4] https://neuronmocap.com/
[5] https://www.xsens.com/motion-capture

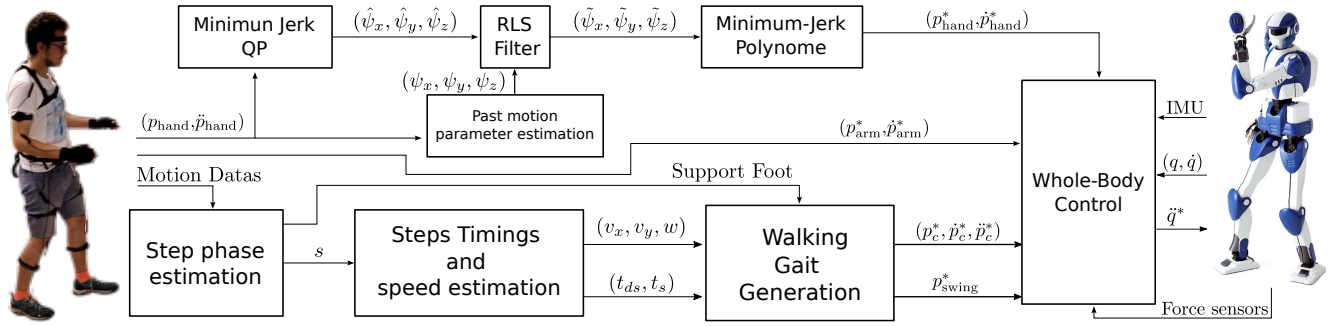


Fig. 1: Proposed control scheme for human-humanoid synchronized motion; the superscript $*$ refer to references forwarded to the robot. *Motion Data* refer to the recorded data detailed in Fig. 4; p_c is the center of mass reference for the robot; p_{swing} represents the swing foot trajectory.

it is possible to convert each set into a probability distribution and therefore create a so called "motion primitive" [16]. This solution allows to anticipate motions that belong to the recorded set over a long period of time [17].

C. Contribution

Our control scheme is summarized in Fig. 1; it aims at imitating and anticipating an operator's intention at two levels:

- The upper body motion by anticipating the hand motion of the operator in the cartesian space to achieve synchrony, this motion prediction algorithm requires only the past and present motion data of the user limb;
- The locomotion by evaluating and anticipating the operator walking pace and forward it to the robot considering balance constraints.

These anticipatory strategies are integrated in a task-space whole-body controller, with experiments using the HRP-4 humanoid robot.

III. Arm Motion Synchronization

A. Trajectory estimation model

We estimate the hand motion using a polynomial interpolation that match a minimum jerk model and correct it through an RLS (Recursive Least Square) filter.

We can derive from the minimal jerk model an analytic solution [18]. It states that to go from an initial point $P_0 = (p_{0x}, p_{0y})$ at t_0 to a point $P_2 = (p_{2x}, p_{2y})$ at t_2 passing by an intermediate point $P_1 = (p_{1x}, p_{1y})$ at t_1, the trajectory can be expressed as a fifth-order polynomial functions of time:

$$\begin{aligned} x(t) &= \sum_{k=0}^{5} c_{kx} t^k + p_{1x}(t-t_1)_+^4 + p_{2x}(t-t_1)_+^5 \\ (t-t_1)_+ &= \begin{cases} t-t_1 \text{ if } t \geq t_1 \\ 0 \text{ if } t < t_1 \end{cases} \end{aligned} \tag{1}$$

And similarly for $y(t)$. [19] extended, by analyzing human catching movements, the minimum Jerk model to 3D. Therefore we propose to apply the model formulated in [18] to 3D.

Using this assumption, the hand motion can be interpolated between P_0 and P_2 through an interval $[t_0;t_2]$ with a waypoint P_1 at t_1 through the parameters $C = (c_0, \ldots, c_5, p_1, p_2)^T$ for each axis. Therefore, we aim to identify the parameters (C_x, C_y, C_z) which define the polynomial regressing the trajectory in the interval $[t_0;t_2]$.

We evaluate those parameters by using the past hand trajectory considered to be in the interval $[t_0;t_1]$. For each axis, C can be determined on the basis of the position, velocity, and acceleration at P_0 and P_2, and the position and velocity at P_1. Therefore, identifying C is equivalent to identifying $\psi = (x_0, \dot{x}_0, \ddot{x}_0, x_1, \dot{x}_1, x_2, \dot{x}_2, \ddot{x}_2)^T$. C can then be computed using a 8×8 matrix A such that $\psi = AC$.

We can compute ψ by formulating an optimization problem that minimizes, within $[t_0;t_1]$, the norm between:

- The polynomial and the recorded position;
- The second derivative of the polynomial and the recorded acceleration.

The recorded trajectories are obtained from a motion capture system composed of IMUs which records angular velocity and linear acceleration that are exploited to reconstruct a user's posture. Motion data are provided at fixed frequency f_m. We record b_{pos} as the past position and b_{acc} as the past acceleration of the operator's hand of a duration $t_1 - t_0$. Setting $t_0 = 0$ and $t_2 > t_1$, we formulate the problem as:

$$\begin{aligned} \hat{\psi} &= \text{argmin}_{\psi} \|b_{\text{pos}} - H_{\text{pos}}\psi\|_2^2 + \|b_{\text{acc}} - H_{\text{acc}}\psi\|_2^2 \\ &\psi_{\min} \leq \psi \leq \psi_{\max} \end{aligned} \tag{2}$$

with : $H_{\text{pos}} = T_0 A^{-1}$ and $H_{\text{acc}} = T_1 A^{-1}$ and

$$T_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1/f_m & 1/f_m^2 & 1/f_m^3 & 1/f_m^4 & 1/f_m^5 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & t_1 & t_1^2 & t_1^3 & t_1^4 & t_1^5 & 0 & 0 \end{pmatrix}$$

$$T_1 = \begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6/f_m & 12/f_m^2 & 20/f_m^3 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 2 & 6t_1 & 12t_1^2 & 20t_1^3 & 0 & 0 \end{pmatrix}$$

$\psi_{\min}$ and $\psi_{\max}$ are set as safety margin.

Symbol $\hat{\psi}$ represent an initial guess of the predicted trajectory, using a RLS filter, we estimate a parameter $\theta \in \mathbb{R}^{64}$ to guess a more accurate value $\tilde{\psi}$ following:

$$\tilde{\psi} = \Psi\theta + \omega \tag{3}$$

where: Ψ is a 8×64 diagonal block matrix of the vector $\hat{\psi}^T$ and ω is a zero-mean noise with covariance R. The parameter θ can be estimated online with the past data; it is updated using the real value ψ with the following two equations:

$$P^- = P + R; \qquad K = P^- \Psi^T (\Psi P^- \Psi^T + R). \tag{4}$$

We then update P, R and θ as follows:

$$P \leftarrow (I - K\Psi)P^-; \quad \theta \leftarrow \theta + K(\psi - \tilde{\psi}); \quad R \leftarrow (\lambda^{-1} - 1)P;$$

with $\lambda \in [0;1]$ often refereed as a "forgetting factor" and provides an approximated exponential decaying weighting on the past data [20]. P defines the estimation covariance and K the estimator gain matrix.

We train this filter on past known data and use $\tilde{\psi}$ to obtain a motion prediction of the user hand. Then, we can compute from $\tilde{\psi}$ any set of position acceleration and velocity in the interval $[t_1;t_2]$. However, we rather use $(x_2, \dot{x}_2, \ddot{x}_2)$ as they represent the furthest prediction available for a value of Δt.

Finally, we can forward the estimated $(x_2, \dot{x}_2, \ddot{x}_2)$ to an inverse kinematics control scheme using an acceleration based tracking law:

$$\ddot{p}_d = -K_p(p - p_r) - K_d(\dot{p} - \dot{p}_r) + \ddot{p}_r, \tag{5}$$

where p_r represent the reference end-effector pose and $\ddot{p}_d$ the robot desired end-effector acceleration. The gain K_d is set to the critical value $K_d = 2\sqrt{K_p}$, therefore, K_p reflects the tracking behavior w.r.t the input target.

B. Determining the tracking latency to compensate

The main external parameters of the presented model are the timings t_1 and t_2 (having set $t_0 = 0$); t_1 is the duration of the input trajectory sequence and $t_2 - t_1$ is how far ahead in time we estimate $(x_2, \dot{x}_2, \ddot{x}_2)$.

Therefore, the optimal value $t_2 - t_1 = \Delta t_{\text{opt}}$ should compensate two sorts of delays:

1) The first delay is inherent to communication lag between the instant the user performs a motion and the instant the measured position is transmitted to the robot. This delay can be estimated using communication average delays;
2) The second delay comes from the robot control itself. The robot motion has an unfixed latency to reach the target depending on the control parameters.

We can measure the delay between a user motion and the robot using the cross correlation between two discrete signals $f(t)$ and $g(t)$ using:

$$f * g(n) = \sum_{m=-\infty}^{\infty} f(m)g(m+n)$$

A value of Δ_{opt} can then be estimated beforehand using pre-recorded values of the robot hand velocity magnitude v_r and of the user v_h, and applying: $\Delta t_{\text{opt}} = \arg\max_{t \in \mathbb{R}} v_h * v_r(t)$.

However, it is important to keep in mind that the prediction error increases with higher values of $t_2 - t_1$, so Δt cannot always be set at will.

To illustrate this, the proposed model is evaluated offline using the recorded motion of a user's hand. Figure 3 shows a prediction sample in one axis with two values of $\Delta t = t_2 - t_1$, and Fig. 2 shows the average error with the range 2σ depending of Δt. The limit of the model are presented from the choice of Δt. We note in Fig. 2 that the predicted trajectory has an increasing average error during uniform motion in the interval $[1.0, 3.0]$ as Δt increases. Moreover, notice that stroke motions induce a high prediction error due to two reasons:

- A stroke motion is not 'grasped' by the minimum jerk model;
- As the estimation is performed from past data, a stroke motion cannot be anticipated.

The limit of the prediction of the stroke motions can be seen in Fig. 3a where the estimated pose during the stroke motion is seen 0.1 seconds ahead instead of 0.2 seconds. The lack of precision can also be evaluated by monitoring the filter estimation covariance matrix P; Δt is chosen to account for the control delay and the limit of the model itself.

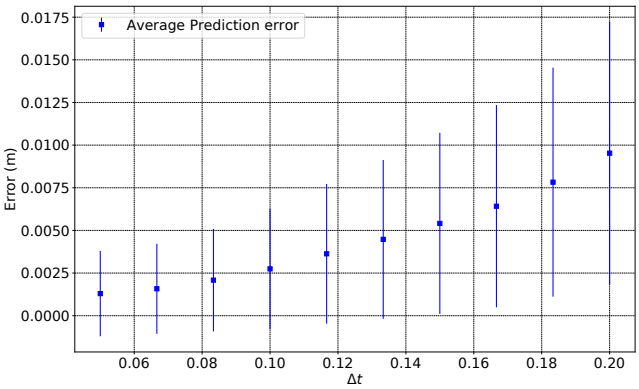


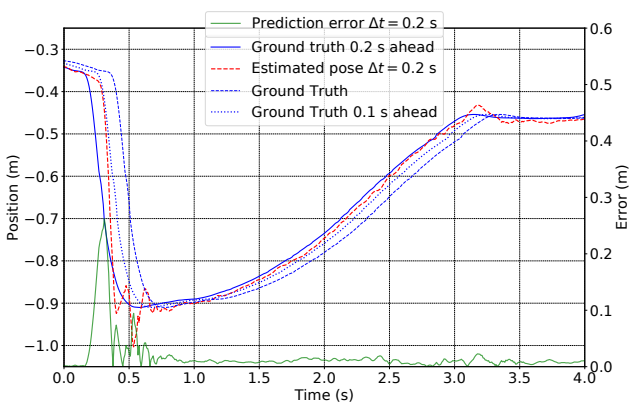
Fig. 2: Average error between the estimated x_2 and the real state depending of Δt for the motion in Fig. 3 for $t \in [1;4]$ using $t_1 - t_0 = \Delta t$, $f_m = 120$ Hz.

IV. Walking Imitation

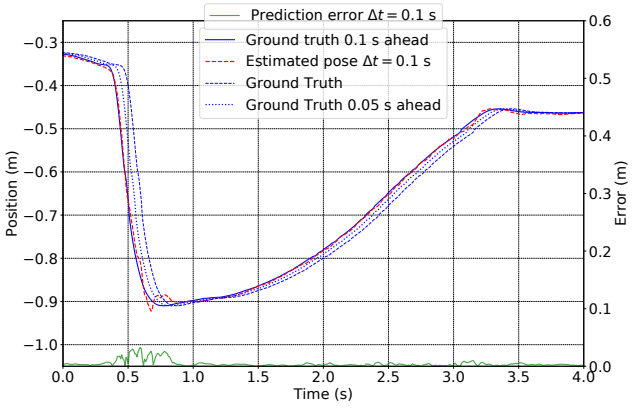
Whereas arm motion synchrony can be achieved by imitating the kinematics of the human limbs, walking synchrony requires to meet balance constraints that are paramount for both humans and humanoids. Besides, one must estimate relevant human walking parameters to be conveyed to the humanoid robot in order to achieve a synchronized walking. There are of two kinds:

- The kinematics that represent the support foot and the future step location;
- The timings which indicate the take off and landing time of the swing foot. We believe this is the most critical feature to reach a high level of embodiment [21]. It is important that human's and humanoid's legs take-off and land synchronously [10].

In the case of teleoperated humanoid locomotion, it is very likely that the operator will have a limited space to control the humanoid robot; it is important to consider also the case where the operator uses on-site stepping.



(a) Estimated x_2 for $\Delta t = 0.2$ s



(b) Estimated x_2 for $\Delta t = 0.1$ s

Fig. 3: Evaluation of the prediction model in one axis depending on $\Delta t = t_2 - t_1$. Using $t_1 - t_0 = \Delta t$, $f_m = 120$ Hz.

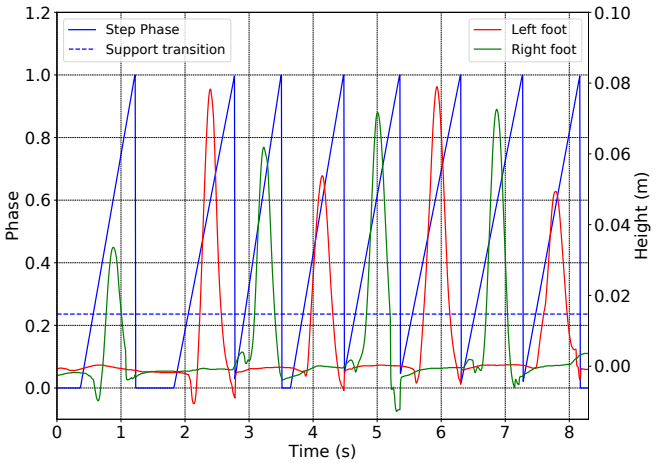


Fig. 4: Example of step phase profile correlated with the step height.

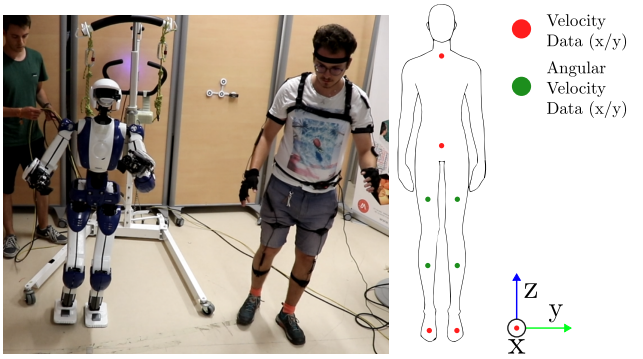


Fig. 5: Experimental set-up: operator walking with the robot (left) and location of sensor and measured data type (right).

A. *Walking frequency estimation*

Synchronizing the user's steps with the humanoid means providing to the robot the timing of foot lift-off and landing ahead of time. We also need to tackle both cases where the user is stepping in place and actually walking. For that, we propose to identify those stepping parameters using a Deep Learning model in order to extract from the user's motion data the stepping phase $s(t)$ defined as:

$$s(t) = \frac{t - t_{st}}{T_s - t_{st}} \tag{6}$$

where t_{st} is the double support starting time and T_s is the single support ending time, we define (t_{st}, T_s, s_{ds}) as the *step phase parameters*.

As it can be unclear to define t_{st} (especially when the user's motion starts), we set t_{st} such that s reaches a value s_{ds} when the user starts its single support phase, hence:

$$t_{st} = \frac{s_{ds} T_s - t_{ds}}{s_{ds} - 1} \tag{7}$$

In fact, s_{ds} can be set in order to match the human double support period during one step which is $s_{ds} = \frac{1}{\phi^3} \approx 0.24$, where ϕ is the golden ratio [22].

1) Dataset: We use as a dataset the motion data of a user walking recorded with the `Neuron` motion capture mockup. We record the angular velocity of the lower body parts and the velocity of the upper body parts. Figure 5 maps the sensor location and the recorded data where the displayed frame is attached to one of the user limb (in our case the hip center). We do not apply an additional filter to the built-in software. The proprietary software can provide the contact state of each foot with the floor using the acceleration data that we use to compute the user step phase parameters.

We split the data in sequences containing the user's motion data as inputs and sequence of the future user step phase as expected output. The sequences on which $s(t)$ transitions from a constant 0 value to non-zero indicate the transition from a standing phase to a walking motion in a close future. However this sequence cannot be predicted and therefore must be discarded. We also include motion data where the user is static or is doing random motions with the upper body, e.g., rotating, bending forward... where $s(t)$ is always equal to 0. This is to reduce, as much as possible, false positive estimations of steps. Figure 4 shows an example of a motion data and the output step phase.

2) Deep learning model using Recurrent Neural Networks: When dealing with temporal data, our model is expected to see each time values of a sequence as a consequence of the previous ones. Recurrent Neural Networks

(RNN) models proved high performance in various problems such as language translation [23]. A RNN cell process each elements of a sequence one by one and use the previous computation of the sequence input to get the next sequence value. Therefore, if we set as an input a temporal sequence $(x_1,\cdots,x_n)^T$, then a RNN layer will compute a sequence $(h_1,\ldots,h_n)^T$ and an output sequence $(y_1,\ldots,y_n)^T$ as follows:

$$h_i = \sigma(W_h x_i + U h_{i-1}) \quad (8) \qquad y_i = W_y h_i + b_y \quad (9)$$

$(h_1,...,h_n)^T$ is called the hidden states, W_h, W_y, b_y, U are the RNN weights and σ is a smooth, bounded function such as a hyperbolic tangent or a logistic sigmoid function.

The concept of RNN has been extended to more complex models to handle the long term dependencies in the input sequence. Long Short Term Memory (LSTM) or Gated Recurrent Unit (GRU) are two RNN models that belongs to the group of Gated Recurrent Neural Network and are widely used in the deep learning community. Their differences lies in the method to control the long term past data in the input sequence. [24] showed some empirical equivalence between both models, however, as GRU requires less parameters to be trained. Therefore, we use GRU for our training model.

A GRU [25] updates the hidden state value h_i using the previous hidden state h_{i-1} and a candidate hidden state $\tilde{h}_i$ such as:

$$h_i = (1 - z_i) h_{i-1} + z_i \tilde{h}_i \quad (10)$$

with the "update gate" z_i and the candidate hidden state $\tilde{h}_i$:

$$z_i = \sigma(W_z x_i + U_z h_{i-1}) \quad \tilde{h}_i = \tanh(W_h x_i + U(r_i \odot h_{i-1})) \quad (11)$$

with $\odot$ the element-wise multiplication and r_i the reset gate:

$$r_i = \sigma(W_r x_i + U_r h_{i-1}) \quad (12)$$

Our model tends to extract from a motion data sequence the future step phase sequence of the user. It can therefore be set as a Seq2Seq (Sequence to Sequence architecture). [23], [26] presented Seq2Seq which is divided into two parts: the first, *Encoder*, is a RNN layer that converts the input sequence into an internal representation of the motion, a context, (which will be the last element of the hidden state). The last part is the *Decoder* that converts the computed context into the expected sequence using a second RNN layer. Then, we apply Eq. 9 on the output hidden state to match the sequence output shape.

This decoder is set as the next input to the previous decoder output. In the model proposed in [26], the context is set as the first decoder hidden value h_1. This model is illustrated in Fig. 6.

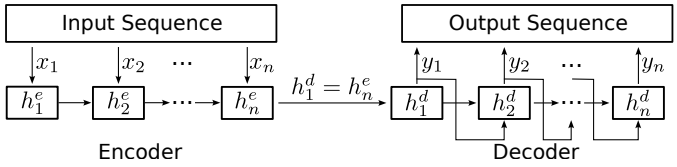


Fig. 6: Seq2Seq model in [26].

3) Learning: We train our model using the Seq2Seq model [26], the loss function is the mean absolute error between the prediction and the model output which is optimized by the Adam (Adaptive Moment Estimation) algorithm. The dataset characteristics are shown in Table I.

TABLE I: Model parameters

Data stream frequency f_d	60 Hz
Model frequency f_{md}	30 Hz
Input sequence length	30
Output sequence length n_s	10
Number of hidden layers	128
Motion data component	8

TABLE II: Training results

Trained samples	7811
Batch size	125
Number of epochs	50
Training data loss	0.0023
Training data mean absolute error	0.0206
Validation data loss	0.0052
Validation data mean absolute error	0.0385

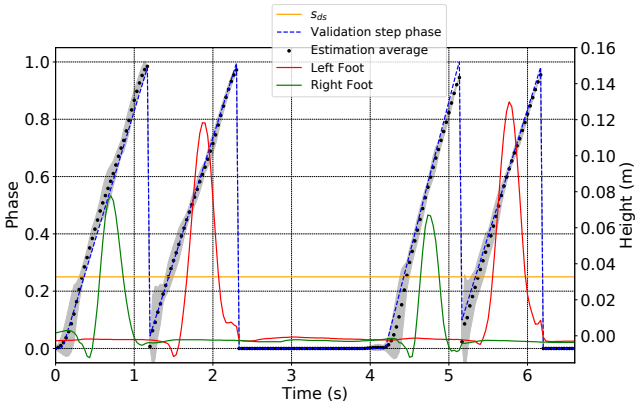


Fig. 7: Phase estimation with validation data compared to the ground truth value, the grey area represent the confidence interval of 95%.

The samples in Fig. 7 represent the average estimation of the stepping phase on validation data. We obtain this average by computing the predicted step phase sequence at the model frequency. We show that the model is able to anticipate the user's single support phase but it represents the prediction with the biggest variance, whereas the single support phase appears to be properly identified. It appears necessary, especially in double support, to use multiple estimations of the model to provide a more accurate value of the stepping parameters.

4) Prediction: Our model outputs a prediction sequence of $s = (s_0,\cdots,s_{n_s-1})^T$ supposed to be linear if we do not consider sequences corresponding to an end and start of new steps. Therefore, for each prediction we can extract a predicted slope and offset $x = (a,b)^T$ using a linear regression such that:

$$x = \text{argmin}_x \|s - Tx\|_2$$
$$T = \begin{pmatrix} 0 & 1/f_{md} & \cdots & (n_s-1)/f_{md} \\ 1 & 1 & \cdots & 1 \end{pmatrix}^T \quad (13)$$

And then compute the step phase parameters

$$t_{st} = -b/a \qquad (14) \qquad\qquad T_s = \frac{1-b}{a} - t_{st} \qquad (15)$$

Finally, using multiple computations of the step phase parameters we can forward an average estimation to the robot and use the standard deviation of this latter as a safety metric for false positive step detection or unreliable estimations.

B. Steps location

When the user is performing a walk-in-place mode, we can estimate an equivalent forward walking speed that will match the user step frequency. The relation between reference velocity and the steps frequency is set from empirical observation of the human walking which lead to Dean's relation [27] linking the human's step frequency f_s, speed magnitude $|v|$ and height h as follows:

$$|v| = \left(\frac{f_s}{0.157} \times \frac{h}{172} \right)^2 \qquad (16)$$

with h in cm and $|v|$ in cm.s^{-1}. We can therefore use the estimated step frequency to generate steps location that will match the computed speed under the humanoid robot kinematics constraints.

V. Experiments and Results

We assessed whole-body synchronized walking using the HRP-4 humanoid from Kawada Robotics. The human operator is equipped with the Neuron Perception 2 IMU whole-body tracking system. The precision of Neuron has been evaluated w.r.t the VICON motion capture system in [28]. Performances shows a RMSE below 4 degrees for mosts joint angles that are in the same range of a the wearable multi-modal system in [29].

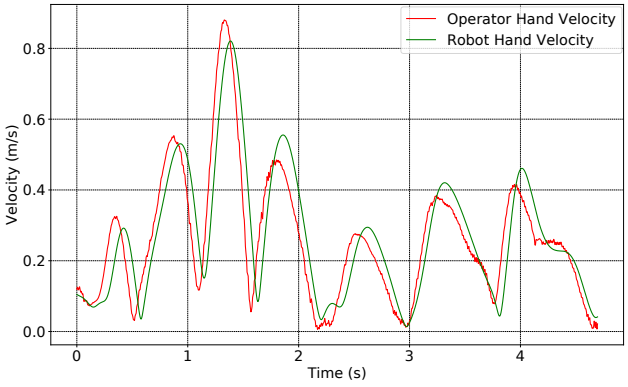
We processed the experiment as follows: (i) we use the Neuron tracking system to collect data from which human walking features prediction is learned; (ii) we integrate upper-body and walking synchronizations using the task-space `mc_rtc` quadratic programming framework [30], [31], that is used also to control the robot; (iii) we assess the walking and terminal point synchronization on simulation using Chorenoid; finally (iv) `mc_rtc` switches into real-time whole body walking with motions of hands and head. We show a human-humanoid side-by-side synchrony and a human-humanoid face-to-face mirror synchrony motions. The computation of the commanded angles and the logged data are done at a frequency of 200 Hz.

A. Experimental constraints

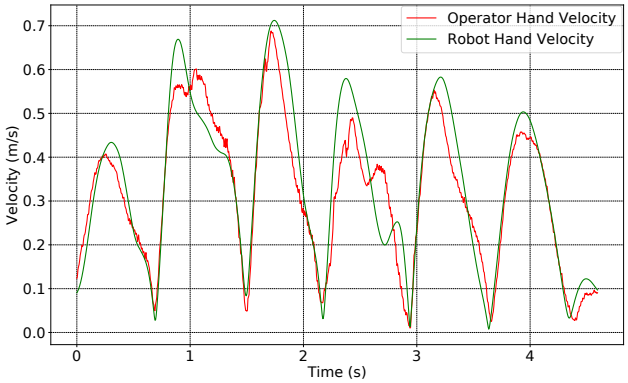
Unlike simulations where torque and joint speed limits can be set at will, the real HRP-4 robot has limited capabilities; e.g., velocity limits and range of motions, lower degrees of freedom, less graceful motions for the available degrees of freedom, and lower range of motion for the existing DoFs as compared to their human counterpart. This is the reason (also considering safety) why the following restrictions have to be considered in real humanoid experiments:

- the learning process applies to a specific human, because of the anthropomorphic and gait differences, the learning process does not generalize to any human operator. Expanding the learning to a large range of human operator is possible, but requires a huge amount of data which is not the core part of our work;
- in order for the synchrony to occur, the operator is asked to perform slow motions to match the speed capacity of the robot. This also allows to achieve experiments without tethering the robot;
- the robot is powered through an outside connected cable and controlled from a wired external computer, therefore, the range of motion of the robot is limited and the reachable space are limited;
- an additional game-joystick is used by another operator for safety purposes. It allows to trigger *on* and *off* the terminal points for synchronized tracking on demand.

B. User hand control



(a) Using measured hand position and velocity, measured delay: 58 ms.



(b) Using predicted hand position and velocity. $t_2 - t_1 = 0.1s$, $t_1 - t_0 = 0.1s$, $f_m = 60$ Hz, measured delay: 3 ms.

Fig. 8: Robots and operator's hands linear velocity magnitude and the computed delay using cross-correlation.

We apply the model developed in Sec. III such as the robot's hands track the operator's predicted hands' Cartesian position and velocity and the current hands orientation and rotational speed, that are updated at a rate of 60 Hz.

However, task space retargeting leads to dissimilarity between the human and the robot posture because of the human/robot shape and degrees of freedom mismatch. Moreover, if the robots arm has kinematics redundancy, the arm posture, tracking only a 6d reference could differ from the user's. Hence, to achieve a convincing synchrony and motion between the human and the robot arm motion, we must ensure that the robot's links between the shoulder and the hand are also following an orientation similar to the user.

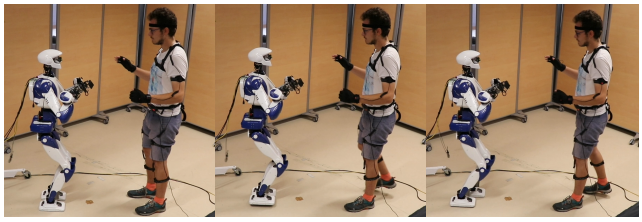
Therefore, each of the robot's two upper limbs are controlled using two tasks. The first is a position task on the robot hand. The second is a task that tracks the operator's upper arm orientation, this has the benefit to provide a similar posture between the operator at the cost of the precision on the hand tracking.

The accuracy of the predicted hand position is shown in Figs. 2 and 3, therefore to assess the motion prediction performance in terms of synchrony, we compare the operator and the robot linear velocity magnitude. More specifically, as the robot's hand motion is also constrained by the arm orientation task, the velocity between the user and the robot will not be identical, we focus on the instant when user's and robot's hand speed reach a velocity close to zero. We show in Fig. 8 the velocity magnitude of the user and the robot during the experiment with and without using the predicted hand position and velocity. Note that anticipating the motion of the operator and forwarding it to the controller enables the robot motion to be synchronized to that of the operator w.r.t the instant when the user's hand comes to a stop. Finally, Fig. 8 shows that the measured delay is reduced from 58 ms to 3 ms (a typical robot control loop is 5 ms) which is under our data logging frequency. However, to obtain this behavior, the model has been set for a value $\Delta t = t_2 - t_1 = 100$ ms which is almost twice the measured anticipation. Nevertheless, this is not a problem as Δt is a tunable parameter that can be empirically set to minimize the delay.

C. Walking parameters online estimation

The robot is using a walking pattern generator that provides a center of mass reference trajectory (position, velocity, acceleration), see [32], and that requires timings parameters and a reference walking velocity.

Therefore we must provide: (i) the single and double support duration; (ii) the walking direction; (iii) the user standing foot. The two first inputs come from the steps timing estimator described in Sec. IV using the same frequencies described in Table I. During the experiments we provide a double support duration time only once; whereas the single support duration is updated regularly during the swing foot phase. On the other hand to estimate the walking direction, one solution could be to use the measured velocity from the IMU on the user's hip; however, if this data can be used for forward walking it cannot be reliable when it comes to side walking as we can't distinguish sway from locomotion. Therefore we rather rely on the velocity measured at the user's swing foot, the drawback of this estimation is that it is available only during the single support phase, therefore, we notice a kinematic discrepancy between the robot and the human swing phase as the robot will swing its leg once the walking direction will be estimated (see, Fig. 9).



(a) End of double support (b) During single support (c) End of single support

Fig. 9: Delay in the walking reference velocity. The user has already placed his feet backward meanwhile the robot is vertically lifting the feet.

To estimate the user's support foot, we rely on the sign of the user's hip velocity in the y-axis (cf. Fig. 5) during the double support phase. We assume that the user does not use the same support twice in a row, therefore during a walking sequence, we estimate the first support foot only once when the walking sequence starts. Figure 10 gathers the contact state of the operator (also obtained from the mocap software) and the robot with the current foot height of both entities. We show that the robot is able to follow the user's pace and swing foot, we still notice some errors in the phase estimation which can be due to a bad estimation from the model itself and from the implemented safeties that can prevent the robot to change its step-timing when it is close from landing w.r.t the previous input step timing.

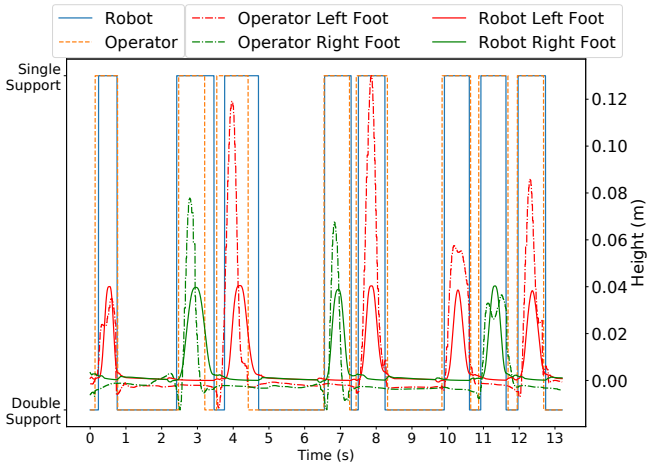


Fig. 10: Support state of the user and the robot with the vertical position of the swing foot; the estimations errors can be seen at the instants when the support state change.

VI. Conclusion

We proposed a novel control framework that enables a humanoid robot to synchronize its steps and arms motions with those of a human in real-time. Using a whole-body motion capture system base on IMUs, we used an invariant of the human hand motion, the minimum jerk model, coupled with a RLS filter to anticipate the hand's cartesian motion. Using recorded data of human walking, we trained a deep

learning model capable of predicting the human duration of double and single support phase. Those predicted data are integrated in a task-space whole-body controller with constraints handling to synchronize the robot with the operator. Our approach is successfully assessed in real experiments with the HRP-4 humanoid robot synchronizing motions with an operator.

The training of the model was performed offline and therefore dependent of the past training data, it would be interesting to switch to a self-learning model approach which might also require an improvement in the detection of the contact state. Another major improvement could be to use a brain computer interface to anticipate on the motion instead of relying solely on the body motions [7] (EEG detects intentional motion about 300 ms ahead of time).

We used the minimum jerk model to anticipate Cartesian translation of the hands of an operator. However a similar method cannot be applied on the operator's hand orientation since (i) to our knowledge there is no equivalent model that study the human hands orientation; (ii) our prediction model relies on the acceleration data of the user's hand, our device fail in estimate properly the hand angular acceleration. Exploring the behavior of the hand's and head orientation in future work would open perspectives in embodiment [21].

REFERENCES

[1] S. S. Wiltermuth and C. Heath, "Synchrony and cooperation," *Psychological Science*, vol. 20, no. 1, pp. 1–5, 2009, pMID: 19152536.

[2] S.-J. Chung and J.-J. E. Slotine, "Cooperative robot control and concurrent synchronization of lagrangian systems," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 686–700, 2009.

[3] P. Bechon and J.-J. E. Slotine, "Synchronization and quorum sensing in a swarm of humanoid robots," *CoRR*, vol. abs/1205.2952, 2012.

[4] Y. Miyake, "Interpersonal synchronization of body motion and the walk-mate walking support robot," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 638–644, 2009.

[5] A. Nakazawa, S. Nakaoka, K. Ikeuchi, and K. Yokoi, "Imitating human dance motions through motion structure analysis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2002, pp. 2539–2544.

[6] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, H. Hirukawa, and K. Ikeuchi, "Learning from observation paradigm: Leg task models for enabling a biped humanoid robot to imitate human dances," *International Journal of Robotics Research*, vol. 26, no. 8, pp. 829–844, 2007.

[7] P. Gergondet, S. Druon, A. Kheddar, C. Hintermüller, C. Guger, and M. Slater, "Using brain-computer interface to steer a humanoid robot," in *IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 192–197.

[8] K. Darvish, Y. Tirupachuri, G. Romualdi, L. Rapetti, D. Ferigo, F. J. A. Chavez, and D. Pucci, "Whole-body geometric retargeting for humanoid robots," in *IEEE-RAS International Conference on Humanoid Robots*, 2019, pp. 679–686.

[9] L. Penco, N. Scianca, V. Modugno, L. Lanari, G. Oriolo, and S. Ivaldi, "A multimode teleoperation framework for humanoid loco-manipulation: An application for the iCub robot," *IEEE Robotics and Automation Magazine*, vol. 26, no. 4, pp. 73–82, 2019.

[10] M. Benallegue, P.-B. Wieber, A. Kheddar, and B. Espiau, "A computational model for synchronous motion imitation by humans: The mirror controller applied on stepping motions," in *IEEE-RAS International Conference on Humanoid Robots*, 2010, pp. 322–327.

[11] J. Ramos and S. Kim, "Dynamic locomotion synchronization of bipedal robot and human operator via bilateral feedback teleoperation," *Science Robotics*, vol. 4, no. 35, 2019.

[12] T. Flash, Y. Meirovitch, and A. Barliya, "Models of human movement: Trajectory planning and inverse kinematics studies," *Robotics and Autonomous Systems*, vol. 61, no. 4, pp. 330–339, 2013.

[13] K. Lyu, H. Chen, Z. Liu, B. Zhang, and R. Wang, "3d human motion prediction: A survey," *Neurocomputing*, vol. 489, pp. 345–365, 2022.

[14] K. Kutsuzawa, S. Sakaino, and T. Tsuji, "Sequence-to-sequence model for trajectory planning of nonprehensile manipulation including contact model," *IEEE Robotics and Automation Letters*, vol. 3, pp. 3606–3613, 2018.

[15] Z. Liu, Q. Liu, W. Xu, Z. Liu, Z. Zhou, and J. Chen, "Deep learning-based human motion prediction considering context awareness for human-robot collaboration in manufacturing," *Procedia CIRP*, vol. 83, pp. 272–278, 2019.

[16] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013.

[17] L. Penco, J. Mouret, and S. Ivaldi, "Prescient teleoperation of humanoid robots," *CoRR*, vol. abs/2107.01281, 2021.

[18] P. Viviani and T. Flash, "Minimum-jerk, two-thirds power law, and isochrony: converging approaches to movement planning," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 21, no. 1, pp. 32–53, 1995.

[19] N. Fligge, J. McIntyre, and P. van der Smagt, "Minimum jerk for human catching movements in 3D," in *IEEE-RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2012, pp. 581–586.

[20] D. Simon, *Optimal State Estimation: Kalman, H-Infinity, and Nonlinear Approaches*. USA: Wiley-Interscience, 2006.

[21] L. Aymerich-Franch, D. Petit, G. Ganesh, and A. Kheddar, "The second me: Seeing the real body during humanoid robot embodiment produces an illusion of bi-location," *Consciousness and Cognition*, vol. 46, pp. 99–109, 2016.

[22] M. Iosa, A. Fusco, F. Marchetti, G. Morone, C. Caltagirone, S. Paolucci, and A. Peppe, "The golden ratio of gait harmony: repetitive proportions of repetitive gait phases," *BioMed research international*, vol. 2013, 2013.

[23] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS Workshop on Deep Learning*, 2014.

[25] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014.

[26] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[27] G. A. Dean, "An analysis of the energy expenditure in level and grade walking," *Ergonomics*, vol. 8, no. 1, pp. 31–47, 1965.

[28] C. Z. Choo, J. Y. Chow, and J. Komar, "Validation of the perception neuron system for full-body motion capture," *PLOS One*, vol. 17, no. 1, 2022.

[29] B. Fang, F. Sun, H. Liu, D. Guo, W. Chen, and G. Yao, "Robotic teleoperation systems using a wearable multimodal fusion device," *International Journal of Advanced Robotic Systems*, vol. 14, no. 4, pp. 1–11, 2017.

[30] K. Bouyarmane and A. Kheddar, "On weight-prioritized multi-task control of humanoid robots," *IEEE Transactions on Automatic Control*, vol. 63, no. 6, pp. 1632–1647, 2018.

[31] K. Bouyarmane, K. Chappellet, J. Vaillant, and A. Kheddar, "Quadratic programming for multirobot and task-space force control," *IEEE Transactions on Robotics*, vol. 35, no. 1, p. 6477, feb 2019.

[32] N. Scianca, D. De Simone, L. Lanari, and G. Oriolo, "MPC for humanoid gait generation: Stability and feasibility," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1171–1188, 2020.