

2007-49

## Combined Controllers that Follow Imperfect Input Motions for Humanoid Robots

Authors: Gazihan Alankus, O. Burchan Bayazit

Corresponding Author: [gazihan@cse.wustl.edu](mailto:gazihan@cse.wustl.edu)

**Abstract:** Humanoid robots have the potential to become a part of everyday life as their hardware and software challenges are being solved. In this paper we present a system that gets as input a motion trajectory in the form of motion capture data, and produces a controller that controls a humanoid robot in real-time to achieve a motion trajectory that is similar to the input motion data. The controller expects the input motion data not to be dynamically feasible for the robot and employs a combined controller with corrective components to keep the robot balanced while following the motion. Since the system can run in real-time, it can be thought of a candidate for teleoperation of humanoid robots using motion capture hardware.

Type of Report: Other



# Combined Controllers that Follow Imperfect Input Motions for Humanoid Robots

Gazihan Alankus, O. Burchan Bayazit  
Washington University in St. Louis  
{gazihan, bayazit}@cse.wustl.edu

**Abstract**—Humanoid robots have the potential to become a part of everyday life as their hardware and software challenges are being solved. In this paper we present a system that gets as input a motion trajectory in the form of motion capture data, and produces a controller that controls a humanoid robot in real-time to achieve a motion trajectory that is similar to the input motion data. The controller expects the input motion data not to be dynamically feasible for the robot and employs a combined controller with corrective components to keep the robot balanced while following the motion. Since the system can run in real-time, it can be thought of a candidate for teleoperation of humanoid robots using motion capture hardware.

## I. INTRODUCTION

The advances in humanoid robot hardware in recent years made them closer to being a part of everyday life in which they can assist humans in many ways. As the hardware emerged there has been a growing interest in creating better algorithms for their efficient use. The inherent difficulty in controlling humanoid robots limits the variety of motion trajectories that can be realized on them. The ultimate goal is to be able to control humanoid robots so that they can achieve most of the tasks that humans can do, as the hardware permits.

The control problem on humanoids is twofold. Firstly, the humanoid must be able to keep its body upright on its feet while being controlled. This is the low level control problem that is tightly coupled with dynamics. The high level problem is to be able to reason in the environment that the humanoid is in and act according to a certain goal and react to the changes in the environment. Since the high level control depends on the low level control, it is crucial to solve the latter to even start considering the former. If the low level control is efficient and successful enough, humans can supply the high level control. As an example, Robonaut by NASA is an interesting humanoid with very sophisticated actuators that closely resemble human hands [1]. Robonaut is designed to be controlled by astronauts via teleoperation for extra-vehicular activities such as repair and maintenance. Since it does not have legs, the control problem that it needs to solve is rather easy, therefore the astronaut can use it in real-time. One of the goals of humanoid robotics research is to have the same teleoperation capabilities for bipedal humanoid robots requiring balanced full-body motions. With the emergence of such a robust system, bipedal humanoid robots can be teleoperated just like NASA's Robonaut to achieve many tasks for humans, such as bomb detonation, hazardous area

activities and other tasks that are potentially dangerous for human presence while requiring the kinematic capabilities of a human. Such a reliable and robust system could be the first major boost in extending humanoid robots to a wider audience. In this paper we address this need by presenting a real-time algorithm that can control a humanoid with the use of the standard motion capture technology.

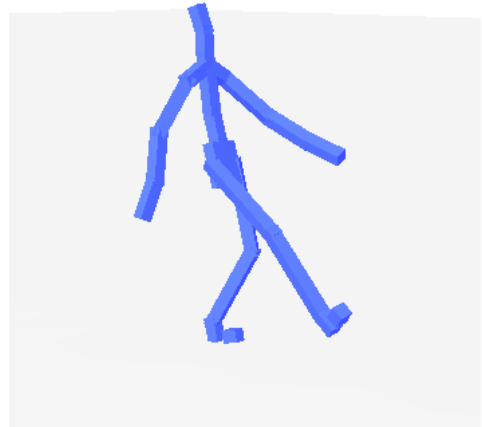


Fig. 1. A humanoid robot created and controlled using motion capture data

In the next section, we present a summary of the related work. In section III, we define the problem with its motivating goals and challenges. Section IV gives detailed information on the physical properties of the robot. In Section V, we present the controller strategies that are composed of simple controller modules. Section VI presents the implementation and experimental results and section VII concludes our paper.

## II. RELATED WORK

There have been a number of researchers that are interested in using motion capture data to control a biped humanoid robot. [2] uses captured human dances to generate whole body motions. [3] learns the motion capture data using predictive control in order to apply it to a humanoid later. [4] uses Gaussian processes to imitate motion capture data with a robot. [5] uses an offline optimization algorithm to modify motion capture data so that it is feasible to be played on a humanoid robot. [6] solves the planning problem in a low dimensional space using optimization and then modifies

the motion to obey physical constraints. While this is not an exhaustive list, the algorithms published are not geared towards real-time operation.

Some terms related to balancing biped robots, such as zero moment point (ZMP) and center of pressure (CoP) have been used loosely in the literature, often extending or limiting their definitions. Publications like [7] and [8] corrected these mistakes by defining and studying these terms carefully. There have been many publications about creating balanced motions for humanoids. [9] gives algorithms for creating fine-tuned humanoid motions for humanoid robot operations. [10] and [11] present algorithms for generating balanced humanoid motions and [12] provides a posture modification method for humanoids.

### III. PROBLEM DEFINITION

We can define our problem as retargeting a dynamically imperfect full-body input motion to a target biped humanoid robot in real-time. There is strong motivation and major challenges for this purpose, as detailed in the following subsections.

#### A. Goals

Ultimately, we envision a system in which a human actor performs a full body motion such as walking and a humanoid robot follows that motion at the same time in parallel. Since this needs to happen in real-time, the algorithms for achieving this need to be very efficient so that the human that is controlling the robot via teleoperation can have a natural feeling of control.

We argue that it is possible for a system to have motion captured from the human actor in real-time, transferred to the robot and used to control the robot, which supplies feedback to the actor via video cameras. For the purpose of capturing the motion, existing motion capture systems can be sufficient. Motion capture systems record the motion of the actor via position markers attached to the body of the actor, and are mostly used in the entertainment industry to supply natural motion to artificial characters. The technology is mature and is widely used by animators worldwide.

In this paper, we present a system that uses stored motion capture data to control the robot in real-time, without doing any preprocessing on the input motion. Once such a system is mature enough for field use, a motion capture device can supply motion data to it in real-time, enabling us to control biped humanoid robots via teleoperation.

#### B. Challenges

In such a system since the human actor and the humanoid robot performer have different physical properties. The input motion is imperfect, i.e., it is not dynamically correct for the target humanoid robot because of differences in fundamental physical properties such as geometry and mass distribution. In addition, the positioning of the motion capture markers on the actor's body is also very important, because most of the time it does not represent the geometric properties of

the actor, which makes it even harder to be used in a robot directly.

Simple attempts to play the motion in the robot fails with the robot falling down. The robot may have a good design, but making it follow dynamically incorrect motion trajectories eventually results in the falling of the robot. In fact, in our simulation experiments we create a humanoid robot that resembles the motion capture data on the fly, without taking into account engineering principles to make it easy for the robot to stay in balance. Such a robot is seen on Figure 1 and is usually harder to keep in balance compared to most of the humanoids that were engineered with balance in mind.

## IV. PHYSICAL PROPERTIES

### A. Dynamics of a Humanoid Robot

A humanoid robot is an articulated robot without a fixed base. There are a large number of links on a humanoid robot, most of which are connected using joints with more than one degree of freedom. The number of DoF on a humanoid robot is large for algorithms that do not scale well with increasing DoF. This makes fast motion planning for humanoid robots rather difficult.

The main factors defining the dynamic behavior of a humanoid robot are gravity, contacts and inertia. On ideal conditions, every link of the robot is subject to a constant gravity force that pulls the robot close to the ground. Gravity affects the robot in a number of ways. It can act for or against the actuation of a joint, therefore the joint can underactuate or overshoot depending on the configuration of the robot. Gravity will make the robot tip and fall if it is not balanced with contact forces or inertia.

The dynamics of a biped robot has been studied extensively [7], [8]. Since it can easily get confusing, there have been a number of vague and incorrect publications on this area. Recently, a number of publications addressed this issue and cleared the analysis with detailed explanations [7], [8]. Here is a simple and easy to understand breakdown of important points when analyzing the dynamics of a biped robot. This is by no means a complete analysis and is presented here for clarity of discussion.

A biped robot in static equilibrium is hardly interesting, so we analyze the robot in its non-equilibrium state. "Dynamically stable" or "dynamically balanced" have frequently been descriptions of the state of a humanoid robot. Often they were used without a very precise definition and analysis, except for publications such as [7]. Dynamic balance refers to the state of the robot where the only resultant torque acting on the robot is around the ground perpendicular axis. Since this is a special case of the non-equilibrium state, this is irrelevant to the general dynamic analysis of the non-equilibrium state.

A system of rigid bodies in space may have many forces and torques acting on different parts of it, which have different linear and angular velocities. Nevertheless, such a system can be simplified to a net force  $\mathbf{F}_{net}$  with a certain application point and that is  $\mathbf{F}_{net} = m\mathbf{a}_{net}$ , and a net torque

that is  $\tau_{net} = I_{net}\alpha_{net}$ , where  $m$  is the mass,  $\mathbf{a}_{net}$  is the acceleration,  $\tau_{net}$  is the torque,  $I_{net}$  is the inertia tensor and  $\alpha_{net}$  is the angular acceleration of the system as a whole. Note that  $m$  is constant and  $I_{net}$  changes with the geometric configuration of the system. The system also has a linear velocity  $\mathbf{v}$  and an angular velocity  $\mathbf{w}$ . At any instant, any two such systems with the same  $F$ ,  $m$ ,  $\tau$ ,  $I$ ,  $\mathbf{v}$  and  $\mathbf{w}$  are equivalent, regardless of the configuration of their bodies. The force acting on a rigid body is equal to the rate of change of the linear momentum of the body  $F = \dot{p}$ . Similarly the torque acting on a body is equal to the rate of change of the angular momentum of the body  $\tau = \dot{H}$ . Therefore, one can find the forces and torques of the robot simply by observing its state.

The forces and torques acting between the internal bodies of the system are created as pairs that cancel out each other, therefore to analyze the behavior of the system it is enough to consider the external forces and torques. The external forces acting on the robot are (1) gravity acting on the center of mass (CoM) of the robot, (2) support forces from the ground to the supporting feet. The external torques are created by these two unless there is a special torque exerting contact. The resultant of these two forces produces the acceleration of CoM, which can be considered as an inertia force when calculating the forces that the robot exerts the ground. When the robot has any contact with the ground, the point where gravity plus inertia forces act (ZMP) exactly where the ground reaction force acts CoM [7].

### B. Balance Maintenance

A common practice in literature is to consider a biped robot balanced when the zero moment point (ZMP) is always inside of the support polygon, as opposed to on the edges or corners of the support polygon. The basic requirement of this constraint is to have a support polygon composed of at least three points. By definition, ZMP can never leave the support polygon [7] and this constraint requires ZMP not to be on the edges and corners of the support polygon. Since ZMP is equivalent to center of pressure (CoP), it is the point where the force exchange between the robot and ground is done, assuming a flat ground surface. To create a tipping moment, the gravity plus inertia forces have to act on the edges of the support polygon, or become zero and not act at all, which both violate the constraint of ZMP being inside the support polygon, therefore, this constraint guarantees the robot to stay balanced and not tip over.

On the other hand, it is rare for natural human motion to have the whole foot as support at all times. When changing support feet, it is common for a human wearing hard shoes to have only the tip of the shoe on the ground for a short period of time. Therefore even though it is a good measure, ZMP inside support polygon constraint by itself is not the best way to go for adapting human motions to biped robots. In general, we consider balance as not tipping the robot over, keeping the body upright and not letting any other link other than the robot feet touch the ground. Given the nature of

biped robots, it is easy to lose balance while doing a full body motion.

### C. Control

Although the external forces play the main role in balance and its loss, there is no direct control over them in a biped robot. All one can control are the relative torques between robot links connected with revolute joints. Therefore controlling bipeds without losing balance is far from trivial. Controlling torques on robot's joints change inertial and contact forces immediately. They also have long term effects such as changing the projection of the CoM which changes the net torque.

Since we are interested in using imperfect input motions, balancing the robot gets more complex than simply making sure ZMP is inside the support polygon. In this paper we present a controller strategy similar to a combined PD controller that aggregates a number of factors driving the robot to a balanced motion in real-time.

A PD controller is a torque based feedback controller that drives the output signal to the desired signal with its proportional component, while making sure the output signal converges fast enough to the desired signal with its derivative component.

$$\tau = K_p \tilde{q} + K_v \dot{\tilde{q}}, \quad \tilde{q} = q_d - q, \quad \dot{\tilde{q}} = \dot{q}_d - \dot{q}$$

Here,  $\tau$  is the torque output of the controller,  $q_d$  is the desired angle,  $q$  is the current angle,  $\dot{q}_d$  is the desired angle rate and  $\dot{q}$  is the current angle rate. The parameters  $K_p$  and  $K_v$  are the gain values for the proportional and the derivative component respectively.

In a PD controller, the derivative component inhibits the torque when the output signal is closing in too fast and increases the torque when it is too slow, therefore is a good fit for controlling a complex system in a simple and effective way in real-time.

In our combined controller, we calculate the combined  $q_d^c$  and  $\dot{q}_d^c$ , and then use the PD control law to control the joints. Note that some of the controllers we use do not have a desired  $q$ , they only have a desired  $q_d$ . This will be explained in section V-F along with the combination method.

## V. CONTROLLER STRATEGIES

This section presents the individual controller types that are used to build the combined controller.

### A. Input Motion Following Controller

This is the simplest of the controllers we use. It ignores the motion of the robot in the world and only tries to control the joint angles of the robot so that the values of the joint angles are as close as possible to the ones in the input motion. It calculates  $q_d$  and  $\dot{q}_d$  to be used in a PD controller.

Figure 2 shows a sample case where this controller is the only one acting on the robot. Because of the factors we mentioned in section III-B, the input motion is not perfect and the robot shortly falls down when being actuated only by this controller. Had the input motion been perfect for the

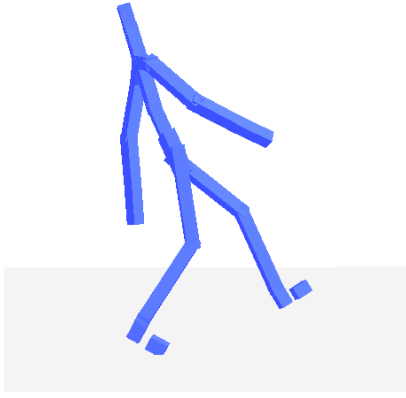


Fig. 2. Input motion following controller failing when used by itself

robot, this controller would likely control the robot close to perfect.

### B. Posture Correcting Controller

This controller is an extension to the input motion following controller based on a simple observation. Most of the time when the input motion following controller fails, it fails because of the feet. As discussed in section III-B, the feet geometry of the robot is usually different than the actor of the original motion. Since the feet are the closest thing to a fixed base a biped robot has, they play a major role in the overall pose of the robot. This is the main controller for playing the input motion and the rest are corrective controllers.

If we were to suppress the ankle joint of the input motion and control it so that the link above the ankle joint gets the orientation that the input motion intended it to have, then we would eliminate a major factor that diverges the body posture from the original motion. With this idea in mind, we created the posture correcting controller. As discussed above, it suppresses the angle inputs for the ankle joint above the support foot when necessary and instead finds the angles that would correct the orientation of the link above, hence the whole robot.

Note that this is not a definite solution since the effects of the foot on the orientation of the body is severely limited.

### C. Link Target Position and Orientation Controller

Since the robot is following the input motion data in real-time, it may miss important events that were supposed to happen in the input motion. One such event is the timing of the foot support. The leg motions of a walking input motion data are very important as they are planted on the ground to pull and push the whole body forwards. If the robot fails to plant the foot on time, the leg motion that the input data has can get wasted before the foot can be planted. This can either result in under-motion of the whole body, or total balance loss because of the whole body orientation not being adjusted properly.

To overcome this behavior, we created the link target position and orientation controller. We use this controller to

direct the foot to desired positions in desired orientations so that footplant constraints are satisfied. This controller works as follows:

The time derivative of the position and orientation of an end effector in a kinematic chain are linear with the time derivative of the joint angle vector.

$$v = J_v \dot{\theta}, w = J_w \dot{\theta}$$

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} J_v \\ J_w \end{bmatrix} \dot{\theta}, J = \begin{bmatrix} J_v \\ J_w \end{bmatrix}$$

Inverting the Jacobian we have,

$$\dot{\theta} = J^\dagger \begin{bmatrix} v \\ w \end{bmatrix}$$

Note that  $J$  is usually not a square matrix, so we need to use the pseudo inverse. To eliminate singular cases, we use the reduced SVD decomposition to calculate the pseudo inverse.

We calculate the Jacobian matrices for both feet with respect to the hip of the robot. These Jacobians can be used to move the feet with respect to the hip. We calculate the desired position and orientations for the feet by assuming the orientation of the hip to be equal to the one in the input motion data. Then we can use  $J^\dagger$  to calculate the angle velocities  $\dot{\theta}$  necessary and use them in the combined controller.

Note that despite we use this controller only for feet placement, it can also be used for other body parts, such as hands for pushing a button in the environment.

### D. CoM Manipulating Controller

As described in challenges, the mass distribution of the robot is most likely different than the actor of the input motion. Also, the relative positions of the motion markers can be misleading for the mass distribution of the robot. Therefore, even though the robot perfectly executes the input motion, it is likely to lose balance because of the differences in total body mass, inertia matrix, and the most important of all, CoG where the gravity forces are applied to the robot. This is very important especially when the robot is acting like an inverted pendulum on the corner of the foot, it relies on the gravity forces to create the desired change in angular momentum. When the CoG is misplaced, the robot fails its motion.

To correct this behavior, we use a CoM manipulating controller which adjusts joint angle rates to change the location of the CoM. We achieve this using the CoM Jacobian, as described in [10]. Note that all the links of the robot have a positional Jacobian  $J_i$ . Columns of  $J_i$  denote the contribution of each angle to the displacement of the link. The number of columns that  $J_i$  has is the number of joint angles between the link and the base, in this case the hip bone. We can align the columns on different  $J_i$ s and find their mass weighted averages to find the  $J_{CoM}$  as follows.

$$\mathbf{J}_{CoM} = \frac{\sum m \mathbf{J}_i}{\sum m}$$

We use this to move the com so that it matches the input data.

### E. ZMP Manipulating Controller

Apart from CoM, we are also interested in the location of ZMP. Not only because we would like to contain it within the inside of the support polygon, but also it defines the rotation of the body conforming to an inverted pendulum [10]. Therefore if the ZMP and CoM are not in the right places, the desired rate of change in angular momentum cannot be achieved.

The way we manipulate the ZMP is through manipulation of the second derivative of CoM position, similar to the method in [10], but more generally the ZMP can be computed as[7]:

$$\mathbf{p}_{ZMP} = \frac{\mathbf{n} \times \mathbf{M}_O^{g^i}}{\mathbf{R}^{g^i} \cdot \mathbf{n}}$$

where  $\mathbf{n}$  is the ground plane normal vector,  $\mathbf{M}_O^{g^i} = m(\mathbf{p}_{CoM} \times \mathbf{g}) - m(\mathbf{p}_{CoM} \times \mathbf{a}_{CoM}) - \dot{\mathbf{H}}$  is the moment of the gravity plus inertia forces with respect to the origin,  $\mathbf{R}^{g^i} = m\mathbf{g} - m\mathbf{a}_{CoM}$  is the resultant gravity plus inertia forces that the robot exerts to the ground,  $\mathbf{p}_{CoM}$  is the location of CoM,  $\mathbf{g}$  is the gravitational acceleration and  $\mathbf{a}_{CoM}$  is the acceleration of the CoM. When we assume  $\mathbf{n}$  and  $\mathbf{g}$  to be strictly vertical, do the necessary substitutions and solve for  $a_x$  and  $a_y$  we get

$$a_{CoM_x} = \frac{-\frac{\dot{H}_y}{m} + (g_z - a_{CoM_z})(p_{ZMP_x} - p_{CoM_x})}{p_{CoM_z}}$$

$$a_{CoM_y} = \frac{-\frac{\dot{H}_x}{m} + (g_z - a_{CoM_z})(p_{ZMP_y} - p_{CoM_y})}{p_{CoM_z}}$$

This means that to achieve the desired ZMP position, we need to calculate the set the horizontal components of the  $\mathbf{a}_{CoM}$  with respect to the vertical component  $a_{CoM_z}$ . There are a number of ways that this can be achieved, such as using the current  $a_{CoM_z}$ , a desired value for  $a_{CoM_z}$  or simply zero. In this paper we use the latter, but this is a point that is open to further research for creating interesting recovery motions from falling.

Once we decide on  $a_{CoM_x}$  and  $a_{CoM_y}$ , we can use the current velocity of the CoM to calculate the desired CoM velocity, which in turn we use in the PD controller. We use this technique to move the ZMP so that it does not fall back when the robot is in full body displacement motion.

### F. Combined Controller

With all these controllers addressed for different issues, we have the necessary tools to imitate the input motion in this biped robot. There could be a number of ways for merging these controllers. For example, we could calculate

torques independently and create a weighted sum of them. Instead, we chose to calculate the individual desired angles and angle rates and make a weighted sum of them that will determine the inputs to the combined PD controller. The reason for choosing this approach becomes clear in the following paragraphs.

Let us take a look at the proportional and derivative components of a PD controller. Unless we constantly give higher values to the derivative component, eventually the proportional component is designed to win and take the output signal to the desired signal. The derivative component, however, plays an important role on defining how the trajectory will look like and has a major role on the speed of the output signal. This resembles our goals as defined in section III-A. We want the robot to move in such a way that keeps it balanced, but ultimately we want the robot to be where the input motion is, just like the output signal of a PD controller. Therefore we let the proportional component of the PD controller to be dominated by following the input motion. The derivative component would then include the derivative component of the PD controller following the input motion, plus the derivative components of all the other corrective controllers. The corrective controllers will influence the motion trajectory, but the robot will end up in the trace of the input motion.

To achieve this, we let only the input motion following controller be a full fledged PD controller and we let all other corrective controllers have only the derivative component of a PD controller. Furthermore, we do not just find the torques per controller and get their weighted sum, but we find the desired angles and the sum of the desired angle rates and then put them together as a PD controller. The reason behind this is that in the former approach, the proportional component of the input motion following controller gets weighted away in the total sum of the controllers. We would need to change its gain depending on how many other corrective controllers we have. Instead, we keep it intact for the final PD controller. Likewise, none of the corrective controllers lose significance with respect to others. Since they only act when correction is needed, they should not be tuned down when there are other controllers working alongside them. Here is what our combined controller looks like:

$$\boldsymbol{\tau} = \mathbf{K}_p \tilde{\mathbf{q}} + \mathbf{K}_v \dot{\tilde{\mathbf{q}}}, \quad \tilde{\mathbf{q}} = \mathbf{q}_d - \mathbf{q}, \quad \dot{\tilde{\mathbf{q}}} = \dot{\mathbf{q}}_d - \dot{\mathbf{q}}$$

$$\mathbf{q}_d = w_a \mathbf{q}_{d_a}$$

$$\dot{\mathbf{q}}_d = \sum w_i \dot{\mathbf{q}}_{d_i}$$

Where  $\mathbf{q}_{d_a}$  is the desired angle vector of the only controller with the desired angle vector,  $\dot{\mathbf{q}}_{d_i}$ s are the desired angle rates of all the controllers and  $w_i$ s are the weights of different controllers that are being combined.

Note that the weight parameters per controller play an important role on the performance of this combined controller. In this study, we manually tuned the weights to create good controllers. In the future we are planning to make weight discovery automatically. A good future study can be

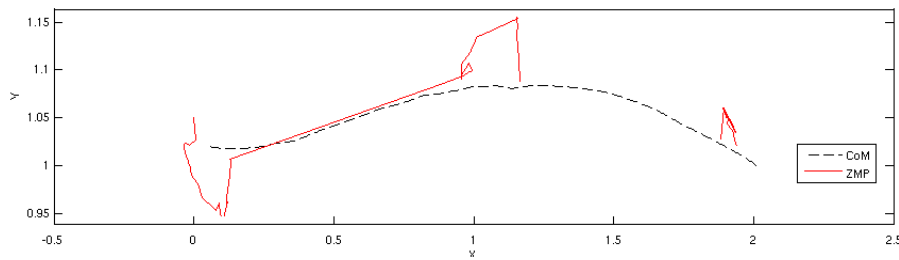


Fig. 3. Experimental results: ZMP and CoM trajectories of the motion generated

to have the weights change adaptively in time depending on the circumstances such as balance being lost, tilted or rough terrain, etc.

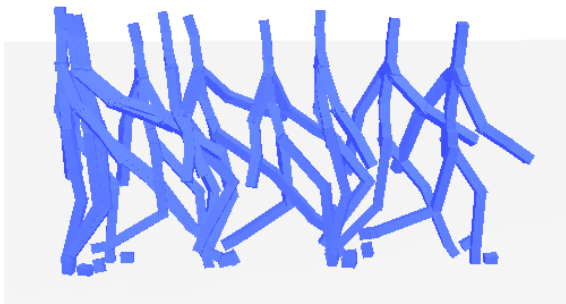


Fig. 4. Experimental results: motion created by the controller being played on the humanoid robot in simulation

## VI. IMPLEMENTATION

For the implementation of our approach, we created an articulated robot simulation framework in C++ on top of Open Dynamics Engine (ODE) [13]. ODE is a real-time rigid body dynamics library that enables us to have realistic physical simulations. Since it is mainly designed for video games, it is very fast, which enabled us to have a real-time running prototype of our system. On the other hand, because of the same reason it is not scientifically accurate since accuracy is expensive to compute and is not required by video games. This was a challenge on our design and prevented us from making detailed predictions in our controllers. Nevertheless, since our aim was to have a real-time controller, keeping it as simple as possible was one of our goals.

For the motion input, we used a walking motion capture data in BVH format. Our system automatically creates a robot according to the joints given by the motion capture data. Every joint connecting two links has three degrees of freedom in order to match the DoF that the motion capture data has. We ran our algorithm in real-time and after adjusting the weights of the controller we gained the results in Figure 3 and 4. The example robot has 24 links and 23 joints. There are 69 controllable angles in the robot, which makes a total of 75 DoF.

## VII. CONCLUSION

In this paper we proposed a real-time controller for humanoid robots based on a combined PD controller. The

algorithm gets a motion capture data as input and produces a controller that controls the biped humanoid robot in real-time to achieve a similar motion trajectory to the input motion while keeping the robot in balance. Since it does not do any preprocessing on the input data, such a system can be used for real-time teleoperation of humanoid robots, which can help humanoids become a part of everyday life.

## REFERENCES

- [1] C.G. Ambrose, R.O. Ambrose, "Robot models for space environments," *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 2, pp. 2113–2118, 1995.
- [2] 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 *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, 2003, pp. 3905–3910.
- [3] R. Chalodhorn, D.B. Grimes, G.Y. Maganis, and R.P.N. Rao, "Learning dynamic humanoid motion using predictive control in low dimensional subspaces," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, 2005, pp. 214–219.
- [4] A.P. Shon, K. Grochow, and R.P.N. Rao, "Robotic imitation for human motion capture using gaussian processes," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, 2005, pp. 129–134.
- [5] N. Naksuk, C.S.G. Lee, and S. Rietdyk, "Whole-body human-to-humanoid motion transfer," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, 2005, pp. 104–109.
- [6] A. Safonova, J. K. Hodgins, and N. S. Pollard, "Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces," in *Proc. ACM SIGGRAPH*, 2004, pp. 514–521.
- [7] P. Sardain and G. Bessonnet, "Forces acting on a biped robot. center of pressure-zero moment point," in *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 2004, pp. 630–637.
- [8] M. B. Popovic, A. Goswami, and H. Herr, "Ground reference points in legged locomotion: Definitions, biological trajectories and control implications," in *The International Journal of Robotics Research*, Vol. 24, 2005, pp. 1013–1032.
- [9] L. Sentis and O. Khatib, "A whole-body control framework for humanoids operating in human environments," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 2006, pp. 2641–2648.
- [10] T. Sugihara, Y. Nakamura, and H. Inoue, "Real-time humanoid motion generation through zmp manipulation based on inverted pendulum control," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 2002, pp. 1404–1409.
- [11] T. Sugihara and Y. Nakamura, "Gravity compensation on humanoid robot control with robust joint servo and non-integrated rate-gyroscope," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, 2006, pp. 194–199.
- [12] D. Wollherr and M. Buss, "Posture modification for biped humanoid robots based on jacobian method," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 2004, pp. 124–129.
- [13] "Open Dynamics Engine," <http://ode.org/>.