Abstract—This paper reports on our first attempt toward a baseball batting demo on a human-sized biped humanoid robot to demonstrate its integrated performance of perception, control and learning. Real-time whole-body motion control and visual perception are integrated to make the robot predict the ball and hit it. A ball is thrown by a human and recognized by the eye cameras. For the prediction, we propose a simple sequential estimator to predict the arrival time and position of the ball. For the control, fast and smooth batting trajectories are superposed on a whole-body force controller with its bipedal balancing taken into account. A trajectory-based iterative learning framework is introduced to incorporate with the ballistic motions. This paper also demonstrates partial but promising simulation and experimental results of the proposed framework. So far, we succeeded in the timing estimation and the fast swinging motion up to 300 ms from start to stop without falling. The experimental and simulation videos are supplemented.

Index Terms—Humanoid robots, Motor learning, Whole-body motion control, Visual perception, Baseball.

I. INTRODUCTION

A. Motivation

Understanding human motor control is useful for development of novel human-friendly assistive devices such as exoskeleton systems. This has been motivated us to explore biologically-consistent motor control strategies for human-sized humanoid robots. At JST ERATO Dynamic Brain Project (1996-2001), various motor control experiments including imitation learning have been conducted successfully with SARCOS Dexterous Arm system and SARCOS DB, upper body humanoid robot [1][2]. Wherein, the torque-controllability coming from the hydraulic actuation with the force feedback was mainly utilized for compliance, not primarily used for load bearing task. From 2005 we have been working on full-body motor control for our full-sized hydraulic humanoid platform, [3][4] (Fig. 1), which was developed by SARCOS for JST ICORP Computational Brain Project, where we have been focusing on bipedal balancing and walking. For that purpose, we implemented a precise torque controller and a model-based whole-body force controller, and succeeded in various compliant full-body balancing task [5]. This serves as a “basic” motion controller, which can be incorporated into various higher-level motor tasks with learning [6].

This paper extend our results into integrated motion task including perception and prediction – baseball batting. Fig. 1 shows one of our preliminary results, in which the robot could hit the incoming ball, “by chance”, based on the prediction of the arrival timing (not including the arrival positions). There are many related studies where externally-mounted stereo camera was used for batting or catching (see, for example, [7] for the batting with high-speed camera). Different from those studies, we don’t use any external cameras, but use the robot’s “eye”, and the robot is balancing with its leg while swinging – like humans do. As shown in Section IV, the method used in this particular demo is rather bold. However, if we fail in this demo, we cannot proceed any more. In this sense, the success in this demo is our first milestone.
B. A batting task covered in this paper

A baseball batting is one of the “extremal” of human motor control: 1) control, 2) perception and 3) learning. Every subtasks are subject to severe constraints (time, power, computation, etc). Therefore, finding plausible solution is quite difficult and deserves a long-term exploration [8][9].

This paper presents our first attempt toward this goal. Being in the initial stage, our focus is on how to make the robot hit the ball (hit the ball to a target area is left for the future work). Herein, the primary task is to estimate the arrival time and position of the ball thrown by a human (or humanoid) pitcher, while swinging a bat toward the presumed position with a specified momentum and attitude.

We have two learning processes here; one is for the ball prediction, the other is for the bat swinging. The need for the ball prediction is clear. Without this, the robot cannot hit the ball with sufficient momentum. The learning for bat swinging is necessary because it is a “ballistic” and coordinated “full-body” motion. Instead of relying on brute-force numerical optimization, we try to make the robot learn such skillful motor control by itself through try-and-error.

Based on the timing prediction, the swing starts before the ball arrives. Since the arrival time is less than 1s, the motion speed is very fast (ballistic). Such a motion control is difficult to be modified during one sequence. Naturally, the learning should be done episodically, based on the final evaluation of the task performance: does it hit the ball? otherwise, how far was it? Crucial thing is the robot must be able to evaluate the control action that has been just taken. This is also true for the ball prediction because a number of uncertainties disturb the estimation, which could be recognized only after the end of the trial.

C. Humanoid platform CB-i

In this study we use a humanoid platform, CB-i, shown in Fig. 1 and Fig. 2 developed by SARCOS for JST-ICORP project [3]. The robot has torque controllability, which we have been pursuing during hardware and software development over the past several years [4]. Its DoF configuration is shown in Fig. 3(a). The arms and legs have 7 DoFs: there are 3 DoFs on the neck; the torso has 3 DoFs. There are total of 50 DoFs including the fingers (12 DoF) and active eye cameras (4 DoF). Except for the hand and eyes, joints are driven by hydraulic servo actuators. The overall height is 1.58 m and the hip height is 0.82 m in an upright posture. Typically the robot is attached with the cubic sole with width 0.12 m, length 0.25 m and depth 0.02 m. Currently, the total weight is about 88 kg.

The robot is installed with position sensors (potentiometers), force sensors (load cells / strain gauges) and attitude sensors (integrated gyro). The actuators are controlled by low-level joint controllers (small-sized digital controllers) supporting position and force feedback at 5 kHz. The primary motor control task is processed on ART-Linux installed on the on-board PC104Plus computer (Intel Pentium-M processor, 1.4 GHz). This computer communicates with the low-level joint controllers via Ethernet at 1 kHz. The computer also communicates with external host computers and vision servers via (wireless) LAN. For the visual system, see Section II-A.
D. Overall architecture and paper organization

The overall control architecture together with the section organization is shown in Fig. 2. The high-level task-space planning center detects an incoming ball by the visual processing system and estimate the arrival timing and positions at each time instance. We use a simple linear time-varying model as shown in Section II.

The estimated timing and positions are used to form desired motion trajectories $\hat{y}(t)$. For the trajectory generation, we use minimum-jerk spline as shown in Section III-A.

The task-space control center computes basic whole-body joint-torques $\tau_L$ (load-bearing torque) with the postural balancing taken into account. This computation requires joint angle, gyro information, and possibly the ground reaction forces.

Section III-B briefly describes the controller.

The given joint trajectories are superposed onto the task-space control torque via appropriate stiffness. This local stiffness also serves as a simple balancing controller. Section argues how this integration by superposition is valid from a simple musculo-skeletal system.

To predict the ball trajectory and evaluate the swinging, the vision system has to keep watching the ball stably even during the fast motions. Section III-C shows a head orientation controller implemented for such purpose. Section III-D shows simulations and experimental results on fast swinging motion with bipedal balancing.

Furthermore, the task-space control center learns compensative joint trajectories to improve overall performances based on self-experience [6]. Although not applied so far, this additional learning for repetitive motions is briefly summarized in Section III-E to make the scope of this study clear.

Finally, Section IV shows the details of the preliminary but integrated experimental results of batting task (Fig. 1). Section V summarizes the results.

II. VISUAL PERCEPTION AND PREDICTION OF BALL

A. Overview

The visual system of CB-i consists of four i1394 cameras with 640x480 pixel resolution at 30 fps, pairwise mounted in each eye, with one providing wide-angle vision and one providing a narrow-angle high resolution foveal view. These are easily connected to either a vision cluster system or to single desktop PC:s for less compute-intensive applications. See [4] for the details.

From a perceptual point of view the baseball-batting task is fairly restricted and domain-specific, and at the same time not latency-tolerant so the general cluster-based vision system was rejected in favor of adapting a fast, efficient tracker system [10] to the task.

The perceptual task in short is primarily to find and track the ball (performed by the base tracker system) and to detect the onset of a throw; to estimate the ball arrival time and to estimate its position relative to the robot at the time of arrival.

In biological systems, approach detection and arrival time is done by Tau-margin estimation [11] in combination with contextual and size cues and with binocular disparity when applicable [12]. For a directly approaching object, its time of arrival $\tau$ is proportional to the rate of dilation in the visual field: $\tau \approx v / \delta v$, when optical angle $v$ is small (see [13] for a full account). This dilation can be computed in just a couple of frames; does not need object segmentation or prior attention on the object; and is computationally efficient. The estimation does require high resolution, dependable edge position or optic flow rate detection. With the constraints for our system, however, the direct use of Tau-margin estimation was found to be marginal, with frequent false approach detection positives. Instead, we employ straightforward disparity-based distance detection.

B. Sequential prediction of timing and position

The 3D position of the ball is acquired in the visual coordinate, and then transformed into $(x(t), y(t), z(t)) \in \mathbb{R}^3$ in the world coordinate frame $\Sigma_W$ based on the posture measurement at video rate (30 fps). We assume the robot hit a ball at some point in a cross section – hyperplane. Without loss of generality, we define the cross section as a subspace of XZ-plane. A ball crosses a position $(x^*, z^*)$ on the section at time $t = t^*$. We introduce a “time-to-go” $\tau(t) = t^* - t$ before the ball crosses the section (i.e. a down counter). This is estimated by the following model.

$$\hat{\tau}(t) = \frac{y(t)}{y(t) - y(t-1)} \Delta t + \delta t,$$

where $\delta t$ is a compensation for the unknown timing estimation error. This error naturally occurs due to the un-modeled external disturbance (fair/head wind) which causes the acceleration. We assume this error can be detected from the ball crossing the section (positional measurement when the ball is close to the robot is assumed to be “repeatable” during the baseball

![Fig. 4](image.png)

Fig. 4. Sequential prediction of the time-to-go and final position. For simplicity, the cross section toward which the ball penetrates is defined as a subspace of XZ-plane, where the robot identifies the arrival. Note that the measured and estimated positions at the final time should coincide with each other even if the measurement is precise or not.
The learning is simply done episodically by

$$\delta t_{k+1} = \delta t_k - \hat{\tau}(t^*)$$  \hspace{1cm} (2)

with $\delta t_1 = 0$, where $k \geq 1$ is the trial number.

The vertical position of the ball is estimated by the following model.

$$\ddot{z}(t) = w_{z1}(t)z(t) + w_{z2}(t)\frac{z(t) - z(t-1)}{\Delta t} \hat{\tau}(t) + w_{z3}(t)\left(-\frac{1}{2}g\hat{\tau}(t)^2\right)$$  \hspace{1cm} (3)

$$= w_z(t)^T\phi_z(t)$$  \hspace{1cm} (4)

where $w_z(t)$ is the time-varying parameter and $\phi_z(t)$ is the input vector which contains the estimated arrival timing $\hat{\tau}(t)$ and the position measurements (current and previous time instant). We decided to have $w_z(t)$ time-varying so that it can reflect time-dependent or distance-dependent uncertainty in the position measurement and actual un-modeled disturbances (e.g. upward/downward wind).

However, this setting requires identifying the timing of throwing ($t = 0$) because one need to synchronize the each time stamp of the parameters for each trial. A possible remedy will be replacing the time stamp with some strictly increasing scalar value, which is insensitive to the throw timing. On the other hand, identifying the throwing timing is more or less necessary to make the robot pay attention to throw and prepare for swing. For this reason, we adopted this time-based estimation here. If the ball is too fast, the robot can start swinging based on the prediction at earlier timing. If the ball is slow, it is better to adopt the prediction at later timing because it is more accurate.

The parameter is updated episodically by using sequential least-square method:

$$S_{k+1}(t) = S_k(t) - \frac{S_k(t)\phi_{k+1}(t)\phi_{k+1}(t)^T S_k(t)}{1 + \phi_{k+1}(t)^T S_k(t)\phi_{k+1}(t)}$$  \hspace{1cm} (5)

$$w_{k+1}(t) = w_k(t) - S_{k+1}(t)\phi_{k+1}(t)\phi_{k+1}(t)^T w_k(t) - z(t^*))$$

for every time step $t > 0$ with $w_1(\cdot) = [1, 1, 1]$ and $S_1(\cdot) = \text{diag}(1, 1, 1)$. In the same way, we estimate the frontal position of the ball $\hat{x}(t^*)$ at each time instance. Note that (5) is TD(1)-learning for delayed reinforcement problems [14]. We don’t use a discount factor $\gamma$ in TD($\gamma$)-learning purely because we have no idea whether it is useful for our setup.

It should be noted that since we estimate mixed parameters $\delta t$ and $w$ ($z^*(t)$ depend on $\tau^*(t)$), we should estimate $\delta t$ first to avoid slow convergence or divergence. If converged sufficiently, then move onto the update of $w$.

Fig. 5 shows one of the simple test. The estimator predicts the time-to-go and the final height under the condition shown in the caption. Due to the head wind, the arrival time was extended to $t^* = 1.41$ from 1.0. The total time step was 48 at 30 fps. After 10 trials, the time error was estimated as $\delta t = -0.47199$, and weighting factor was estimated as $w(1) = (1.0007, 1.1129, 0.88685)$ and $w(48) = (0.99981, 1.0001, 1)$. The prediction error in $\delta t$ is mainly due to the poor cross detection with the fixed camera rate (300 ms).

We can see the prediction becomes precise after a few trials. As the time goes, the precision improves.

III. BAT SWINGING CONTROL

A. Motion teaching/capturing for swinging form

Clearly, a batting form is a coordinated whole-body motion. To facilitate the learning, we divided it into two parts: pre-programmed one and induced one by other controllers. The upper body was attributed to the former. Both parts are improved through the trial-and-error.

Before learning, some initial swinging forms have been taught directly via external forces from a human operator, or captured from humans using a real-time motion capture system (MOCAP). We utilized the VisualEyes Technology, using a triangle of three mutually calibrated camera units. This motion capture system can provide the Cartesian coordinates of uniquely identified markers at a rate of 100Hz, and a typical accuracy of 1-5mm. The acquired trajectories are fitted to minimum-jerk splines with via-points to facilitate learning (terminal-point or via-point learning) and synthesis (interpolation or extrapolation) as in imitation learning paradigm [15][16][16].

Based on the MOCAP data analysis, we begin with only the initial and final position (without via-point) for four arm joints (Shoulder F/E, Shoulder A/A, Humeral ROT, Elbow F/E) and Torso ROT joint (see Fig. 3(a)). Table I shows the initial and final joint angles, which are used in the later simulations and experiments.

B. Whole-body force controller for balancing and trajectory tracking

Balancing control is important to achieve stable batting. We use our whole-body balancing controller proposed in [17]. We do not show the details but give only the simplest form here.
Let us define the coordinate as Fig. 3(b). Herein, \( r_C = [x_C, y_C, z_C]^T \in \mathbb{R}^3 \) is the position vector of CoM in world coordinate frame \( \Sigma_W, q \in \mathbb{R}^n \) is the joint angles, and \( \phi \in \mathbb{R}^3 \) is the attitude of the head. As long as the robot is touching the ground, there is a point called the center of pressure (CoP) whose position vector from CoM is indicated by \( r_P = [x_P, y_P, z_P]^T \in \mathbb{R}^3 \) that lies within the supporting region.

The force controller computes the whole-body joint torque by

\[
\tau_L = J_P(\phi, q)^T(f_u + Mg) - D\dot{q}
\]

(6)

where \( J_P(\phi, q) \in \mathbb{R}^{3 \times n} \) is the Jacobian from CoM to the “desired” CoP and \( f_u = \begin{bmatrix} f_{ux}, f_{uy}, f_{uz} \end{bmatrix}^T \in \mathbb{R}^3 \) is certain new force input. If we set \( f_u = 0 \), the robot is completely gravity-compensated and accept any external forces. The resultant motions and internal motions[18] due to the redundancy [19] are damped by the second term of (6) according to passivity property [20]. If we give \( f_u \) as a feedback controller like

\[
f_u = -K_P x_C - K_D \dot{x}_C
\]

(7)

with the feedback gains \( K_P, K_D \geq 0 \), then the robot can still move within a manifold \( \{ q | x_C = 0 \} \). Thus, any externally-, or internally-imposed motions such as upper body motions given by Section III-A are compensated by other joints. We utilize this property to achieve balancing form.

Furthermore, if we know the CoM does not move too much, we can “downsize” the original controller (6) + (7) into:

\[
\tau = J_P(q)^T Mg - D\dot{q} - K(q - q_d),
\]

(8)

where \( \tau(t) \) is a target joint trajectory and \( K \) is the stiffness matrix. From computational point of view, this controller is not only cheap, but also effective in the sense that it is free from CoM velocity feedback, which makes the robot fluctuate even in the equilibrium due to the noise and delay. The schematic illustration is shown in Fig. 6. We think this noise-free controller is suitable for batting during the fast swinging motion in order to stabilize the vision (but not suitable for recovery from large disturbances).

C. Stabilization of head and orientation control of bat

To predict the ball trajectory and evaluate the swinging, the vision system has to keep watching the ball stably even during the fast motions. The controller is given by a sum of a feedback and a feed-forward term. The feedback control is done in local. That is, the robot controls its neck joints based on the (fast) local feedback from the gyro sensor mounted in the head. The feedback controller, in particular, makes sense because the stabilized head could be the basis for the whole-body posture computation.

Let \( e = [e_1, e_2, e_3] \in SO(3) \) be the orientation matrix of the head and \( \bar{\tau} \) be the desired one. The feedback controller is implemented on the torque controller for the head joints:

\[
\tau_e = J_e^T K_e \omega_e
\]

(9)

with the head joint Jacobian \( J_e(q) \), the velocity feedback gain \( K_e \) and the velocity error

\[
\omega_e = \frac{1}{2}(e_1 \times \bar{e}_1 + e_2 \times \bar{e}_2 + e_3 \times \bar{e}_3).
\]

(10)

In our case, \( e \) can be directly obtained from our gyro sensor.

The feed-forward control is simply counteraction against the head motions due to the “known” body motions. The computation is exactly the same except for that a time sequence \( e(t) \) is given apriori.

Also, we control the bat (hand) orientation using the same formula:

\[
\tau_h = J_h^T K_h \omega_h
\]

(11)

with the hand joint Jacobian \( J_h(q) \), velocity feedback gain \( K_h \) and the velocity error \( \omega_h \)

This is because we want to make the inverse kinematics (IK) simple, which is required in the trajectory learning. Specifically, we need to compute new via points in the joint space from desired position on the cross section (Fig. 4). By introducing the real-time feedback controller (9), we can exclude the hand joints from the IK computation. We skip the details of the trajectory learning. But see [16].

D. Swing control test

We applied the controller (8) to the fast swinging motion. The parameters for the upper-body trajectories are given in Section III-A. Fig. 7 (Video no.1) shows the experimental result of a fast swing. The CoM and some joint trajectories are plotted in Fig. 8. The desired swinging duration (start-to-stop) is 300 ms, but actual swing duration was increased to 450 ms with delay 60 ms. The actual peak joint speeds were 460, 350, 490 deg/s respectively. Since the mass of the arm is about 5.5 kg, the result is not so bad for realistic batting. Note that in this experiment, the balancing controller does not compensate the upper body motions in feedforward manner.
Therefore, the deviation in CoM (top graph) can be further reduced. 

To see how fast the swinging could be, we also tried to combine waist-twisting motions. We also applied the feedback control of the head orientation (9). Fig. 9 and Video no. 2 shows the synthesized swing. In the simulations, surprisingly, fast motions were achieved while keeping the head orientation constant. The waist-twisting motions are achieved by prescribed minimum-jerk trajectories for rotational joints in the lower body: Thigh ROT and Tibia ROT (see Fig. 3(a)), which are similarly superposed onto the balancing controller using (8).

However, if we applied the same controller to the actual robot, it could not keep the stable contact with the ground (without toe-off or heel-off). Nevertheless, even without perfect torque controllability, we expect that the robot can achieve fast motions if we apply the iterative learning framework described in Section III-E (left for the future work).

E. Improving swinging motions

Apart from the learning the “given” upper-body trajectories via IK computation, it is also necessary to improve the compensative trajectories for the lower-body joints. Otherwise, the lower-body joints are constrained to a “fixed” initial posture via stiffness, which may not be useful for fast motions. The underlying principle is “begin with slow motion and learn fast motion”, which has been proposed in [6]. The learning and synthesis is summarized as follows.

(P1) Apply the whole-body balancing controller (see Section III-B)

(P2) Superpose some pre-planned joint trajectories, or “directly teach” the motions by human.

(P3) Record “all” joint motions generated by the previous control trial.

(P4) Encode the recorded motions into some small set of parameters such as via points and frequency.

(P5) Superpose the acquired whole-body joint trajectories with “increased speed”

(P6) Repeat (P3)-(P6)

In this way, fast motions can be learned iteratively. Learning Cartesian trajectory is also possible if the measurement is accurate enough.

IV. PRELIMINARY RESULT: BATTLING WITH SIMPLE TIMING PREDICTION

Here we will show a preliminary result where the robot could hit the ball by successfully estimating the arrival timing. Since the 3D visual perception is under construction, so far we could do only a simple timing prediction, not position estimation. Nevertheless, the result shows a possibility of the proposed control/learning framework to be fully implemented on our robotic platform.

In this demo we used only the two foveal cameras, set at 2.5 ms shutter speed. The target is a yellow foam ball 7cm in diameter, thrown at a distance of between 4 and 5 meters. The ball image from the throwing point is about 8 pixels in
diameter. We use only one swinging form whose time duration is fixed to 300 ms, which is given in Fig. 8.

A procedure used in this particular demo is set as follows:

- The robot stand at the rest posture with the active balancing mode.
- A human standing 4.5m ahead of the robot shows a ball, while the robot vision system on the visual server is looking for the ball in its visual field.
- Once the ball has been detected, the eye controller moves the cameras to center the ball, and the robot enters into a “ready-mode” (activate a simple balancing mode, moves the arms and torso to the initial posture).
- If the disparity between the right and left camera images is over a threshold, assume the ball is approaching (start timer).
- If the counter is under approx 150 ms the robot start swinging. The eye cameras make succade to a visual area where the ball is supposed to pass.
- During the swinging motion, the cameras counteract to the torso rotation in a feedforward manner so that the cameras do not loose the ball image. The images are sequentiliy recorded into vision server for later evaluation.

The result is shown in Fig. 1 (and Video no.3). Of course, the robot could hit the ball “by chance” because we do not estimate the position in this particular demo. Fig. ?? shows the head, eye and torso movement during the motion. Fig. 10 shows the raw images from the wide camera to be used for sequential learning proposed in Section II-B. Note that parameter update can be done even after the one trial. The images are fed into a time-consuming visual processing to extract the time-history of the ball, which is our ongoing work.

V. CONCLUSION

This paper presented on our first attempt toward a baseball batting demo on a human-sized biped humanoid robot to demonstrate its integrated performance of perception, control and learning. Real-time whole-body motion control and visual perception were integrated to make the robot predict the ball and hit it. A ball thrown by a human was recognized by the eye cameras. We proposed a simple sequential estimator to predict the arrival time and position of the ball. For the control, fast and smooth batting trajectories were superposed on a whole-body force controller with the bipedal balancing taken into account. A trajectory-based iterative learning framework was introduced to incorporate with the ballistic motions. We demonstrated partial but promising simulation and experimental results of the proposed framework. So far, we succeeded in the timing estimation and the fast swinging motion up to 300 ms from start to stop without falling.

ACKNOWLEDGEMENT

The authors wish to thank Dr. Gordon Cheng for his support.

REFERENCES


APPENDIX

Let us briefly discuss the superposition in (8) from a biological point of view. Here we adopt a simple musculoskeletal system model proposed by Hogan [21]. With this model, we can compute desired motor command (neural activity) for isometric muscles \( \tau_1, \tau_2 \) when some desired one-parameter joint trajectory \( \bar{q}(t) \in \mathbb{R}^n \) (angle displacement from the rest posture) as well as the desired load torque \( \tau_L(t) \) are given. In this case, the motor command should meet

\[
\tau_L(t) = T_0(\tau_1 - \tau_2) - K(\tau_1 + \tau_2)\bar{q}(t),
\]

where \( K \) is the angular stiffness, and \( T_0 \) is the maximum torque at the rest position (see [21]).
For humanoids, there are many possible joint postures during full-body motor control. Furthermore, the required load is different from task to task. For example, the anti-gravitational torque to support its own weight will be the most fundamental load torque for humanoid robots. The desired loads and postures are supposed to be given by the task-space control center, then the lower-level control center generates necessary motor commands as shown in the right block in Fig. 2. In our robot, the force controller is implemented as a high-speed force feedback control (Section I-C).

We can solve the optimal motor command $\pi$ from (12), which is commanded to the model in a feedforward manner. Then, the actual applied torque becomes

$$\tau = T_0(\pi_1 - \pi_2) - K(\pi_1 + \pi_2)q$$  \hspace{1cm} (13)$$

When $q(t) = \bar{q}(t)$, this satisfy $\tau = \tau_L$. Otherwise, the joint torque becomes

$$\tau = \tau_L + K(\pi_1 + \pi_2)\Delta q$$  \hspace{1cm} (14)$$

where $\Delta q = q - \bar{q}$ is the joint angle error.

This results in a superposition of feedforward and proportional feedback control law for target joint trajectory. This observation motivates the implementation of superposition in whole-body force controller given by (8).