2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Fast Tennis Swing Motion by Ball Trajectory Prediction and Joint Trajectory Modification in Standalone Humanoid Robot Real-time System

Mirai Hattori[1], Kunio Kojima[1], Shintaro Noda[1],
Fumihito Sugai[1], Yohei Kakiuchi[1], Kei Okada[1] and Masayuki Inaba[1]

***Abstract*— In this work, we propose a system for humanoid robot fast motions. When a humanoid robot performs a motion such as a tennis forehand stroke motion, a whole-body fast motion in reaction to visual information is required. There are three problems to tackle. (1) Motion is desired to be quick. (2) Real-time visual processing considering visual noises is needed. (3) Real-time joint angle modification with balance keeping is needed. To solve the problem (1), we used an offline optimization system to enhance the motion speed. To solve the problem (2), we implement a ball trajectory prediction algorithm using the Extended Kalman Filter (EKF). To solve the trade-off between (1) and (3), we propose an offline optimization condition with an estimated balance margin. By using these methods, we achieved a non-step tennis forehand stroke motion with a humanoid robot by predicting a ball's trajectory with stereo cameras on the robot's head.**

I. Introduction

Humanoid robots are designed to work in human environments. Therefore, it is desirable for humanoid robots to perform tasks as fast as humans. In this research, we focus on humanoid robot real-time motions. We humans perform daily tasks using environment recognition and we modify our motions in real time. Sports motions such as running and jumping are good examples of real-time motions. It is easy for humans to modify a motion in real time in reaction to visual information, but it is difficult for humanoid robots because it can easily fall down while modifying a motion. In this study, we propose a standalone humanoid robot system that uses head stereo camera image information while a robot is performing a fast motion. By using the proposed system, we achieved a tennis forehand swing stroke motion of a humanoid robot by predicting a flying tennis ball's trajectory and by modifying a racket's trajectory (Fig. 1).

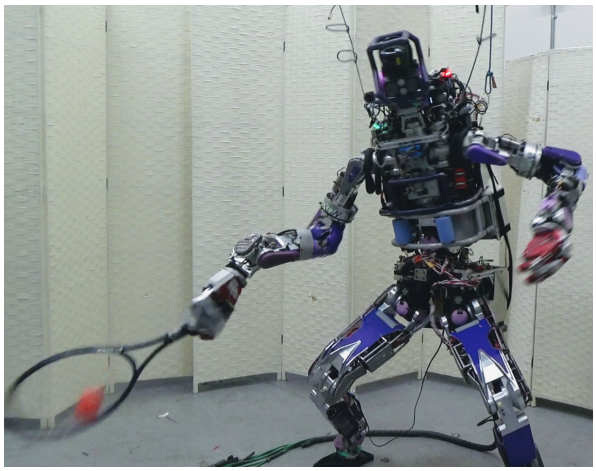


Fig. 1. Humanoid robot JAXON while performing tennis forehand stroke motion

A. Related Work

1) Robot Motion Generation reacted to environmental information: Several works have tackled real-time robot motion generation. A wired multi-link robot fast batting motion was achieved[1]. The robot's joint trajectory was modified to hit the time-changing target position acquired from visual information in real time. Stereo cameras were used in the paper and the distance between those cameras was 0.8[m]. The distance is too long for humanoid robot head stereo cameras, hence the system is not fully applicable to humanoid robots. A lightweight badminton robot was designed to move its joints fast[2]. However, this approach is difficult for regular motor-controlled humanoid robots. Another approach for real-time robot motion generation is continuous motion optimization using a locally reactive controller[3]. The literature focuses on object manipulation and obstacle avoidance using visual perception.

2) Humanoid Robot Real Time Motion Generation: Humanoid robot researchers have been studying real time motion generation considering balance stabilization[4], [5], [6]. Life-sized humanoid robots are necessary to consider balance keeping, which makes locomotion and manipulation difficult. Real time foot trajectory regeneration of a humanoid robot when the robot stumbles was demonstrated[7]. The demonstration uses environmental information using force sensors on the robot's feet during balance keeping. By limiting the discussion to humanoid robot sports motion, a baseball batting motion[8] was achieved. In that study, the motion was generated from human motion capture data. Humanoid soccer robots have been developed over the past few decades[9], but it is still difficult for humanoid robots to perform soccer kick motions as fast as humans. Table tennis hitting motions were performed[10], but its system was not for a standalone humanoid robot system because external cameras were used to predict table tennis ball trajectory.

[1]All the authors are with JSK Laboratory. The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan. [m-hattori, k-kojima, s-noda, sugai, kakiuchi, k-okada, inaba]@jsk.imi.i.u-tokyo.ac.jp

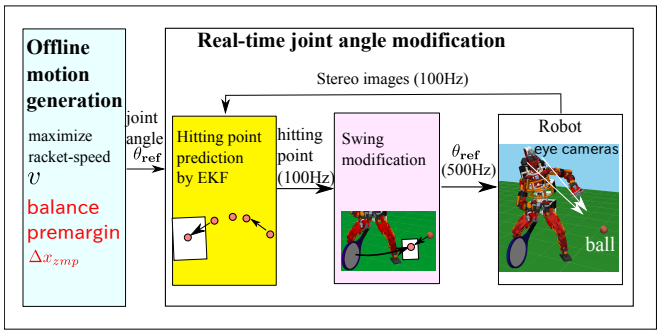


Fig. 2. Whole-body fast motion planning and modification system based on visual information

In our previous study, a humanoid robot's whole body fast sport motion with nonlinear optimization was achieved[11]. However on that study, no environmental information such as visual information was used.

B. Contribution of This Study

The contributions of this study are as follows.

- Motion speed was optimized offline with an estimated balance margin.
- Real-time flying object trajectory prediction by stereo image processing was done using the Extended Kalman Filter (EKF).
- Real-time joint trajectory modification reacted to visual information in a standalone humanoid robot system was achieved.
- By using the proposed system, a humanoid robot's tennis forehand stroke motion was performed without falling down.

II. Real-time Tennis Swing Modification System by Real-time Visual Feedback

In this research, we propose a standalone humanoid robot system for performing fast motions in reaction to visual information. In our proposed system, a tennis forehand swing motion is generated through offline optimization and the motion is modified through real-time optimization.

The whole system in this research is shown in Fig. 2.

The system consists of three subsystems: an offline racket speed optimization system for planning an initial motion (Sec. III), a real-time ball trajectory predicting system using the EKF (Sec. IV) and a real-time joint angle modification system by quadratic programming (Sec. V). In Sec. III, an initial motion is generated offline by using a Sequential Quadratic Programming (SQP) solver. This subsystem includes balancing conditions. The reference Zero Moment Point (ZMP) trajectory is changed in reaction to the ball trajectory prediction output, thus a balance margin is estimated in this subsystem. In Sec. IV, a tennis ball's trajectory is predicted in real time using the images from stereo cameras attached to the robot's head. This process is executed in 100[Hz] frequency. In Sec. V, the joint trajectory is modified in real time based on the predicted ball trajectory. This process is executed in 500[Hz] frequency.

III. Offline Tennis Forehand Swing Motion Planning by Swing Speed Optimization

A. Swing Speed Optimization System

In order to achieve tennis forehand stroke, an initial joint trajectory is generated in this subsystem. The initial motion generation system in this work is based on our previous work[11]. The differences from our previous work in the planning phase are as follows.

1) The desired hitting time is fixed.
2) The security margin for the balancing condition is estimated.
3) The final pose of the motion is not included in the equality conditions.

In this system, the joint trajectories are described as B-Spline curves[12] and the racket speed to the normal direction to the racket surface is optimized through nonlinear optimization. Each planned joint trajectory is reserved as B-Spline curve control points, which is the design variables in the offline optimization problem. A B-Spline curve is described as a recursion The trajectory of a certain joint j is described using uniform n-order B-Spline curves and m control points. Let $p_{j,i}(i = 0 \cdots m-1)$ is a control point i of a certain joint j and $b_{i,n}(t)(i = 0 \cdots m-1)$ is the value of the basis function of n-order B-Spline curve.

$$\theta_j(t) = \sum_{i=0}^{m-1} p_{j,i} b_{i,n}(t) \tag{1}$$

Each control point $p_{j,i}$ is the design variable of the optimization problem.

The optimization problem can be solved with a SQP solver[13]. We used CCSA (Conservative Convex Separable Approximation) algorithm in the solver option[14].

The objective function is the linear combination of the following values:

1) Racket speed to a pre-defined certain direction (Negated because this is a minimization problem)
2) The sum of each integral of the jerk of the joints

Let $w = 1 - 0.000005$ is a weighting variable, $\boldsymbol{n}$ is a unit normal vector to the racket surface, $\boldsymbol{J}(\boldsymbol{\theta_h})$ is the jacobian from the joint angles $\boldsymbol{\theta_h}$ to the racket's sweet spot, t_h is the desired hitting time and t_f is the time when the motion ends, the objective function is formulated as follows:

$$\min_{p_{i,j}} (-w\boldsymbol{n}^T \boldsymbol{J}(\boldsymbol{\theta_h})\boldsymbol{\theta}(\dot{\boldsymbol{t}}_{\boldsymbol{h}})) + (1-w)\int_0^{t_f} ||\dddot{\boldsymbol{\theta}}_t||^2 dt \tag{2}$$

The equality conditions are as follows:

1) Initial pose joint angles and their 1st/2nd order derivatives
2) Hitting pose joint angles
3) 1st/2nd order derivatives of final pose joint angles
4) Foot position and attitude

Let $\boldsymbol{\theta}(t)$ are the joint angles at the time $t = t$, $\boldsymbol{\theta}_0$ are the initial pose joint angles $(t = 0)$, $\boldsymbol{\theta}_h$ are the hitting pose joint angles $(t = t_h)$, $\boldsymbol{r}_{L/R}(t)$ are the left/right foot position and attitude at the time $t = t$ and $\boldsymbol{r}_{L/R0}$ are the initial

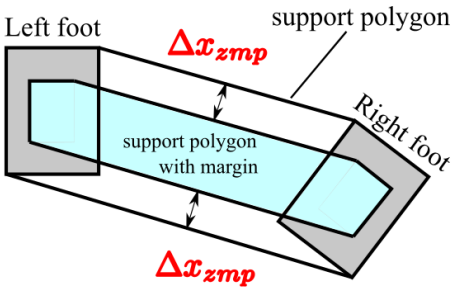


Fig. 3. The support polygon and the ZMP margin
The outer polygon is the support polygon, which is the convex hull of the left and right foot vertices. The inner polygon is the support polygon with an estimated margin Δx_{zmp}. Given the conditions that both the robot's feet does not rotate or step, the reference ZMP trajectory has to be fully inside the support polygon.

left/right foot position and attitude, the equality conditions are formulated as follows:

$$\boldsymbol{\theta}(0) = \boldsymbol{\theta}_0, \dot{\boldsymbol{\theta}}(0) = \mathbf{0}, \ddot{\boldsymbol{\theta}}(0) = \mathbf{0} \quad (3)$$

$$\boldsymbol{\theta}(t_h) = \boldsymbol{\theta_h} \quad (4)$$

$$\dot{\boldsymbol{\theta}}(t_f) = \mathbf{0}, \ddot{\boldsymbol{\theta}}(t_f) = \mathbf{0} \quad (5)$$

$$\boldsymbol{r}_L(t) = \boldsymbol{r}_{L0} \quad (6)$$

$$\boldsymbol{r}_R(t) = \boldsymbol{r}_{R0} \quad (7)$$

The inequality conditions are as follows:

1) Joint angle limitations and joint velocity limitations
2) Collision avoidance
3) Distance between target ZMP and the nearest support polygon edge

The inequality conditions 1 and 2 are calculated under the robot hardware constraints.

The support polygon in the last inequality condition 3 is restricted with a margin Δx_{zmp} (See Fig. 3). The margin is described as ϵ_b in [11], but for a better understanding of the parameter, we call the value as Δx_{zmp}. The value Δx_{zmp} is important because it is a trade-off parameter that controls the volume of the feasible solution space and the safety margin for balance keeping, thus the margin has to be estimated. The estimation process is described in Sec. III-B. Each of the conditions are discretized by a step time $(0.05[\mathrm{s}])$.

The optimization time condition is set so that the motion starts at $t = 0.0[\mathrm{s}]$, the desired hitting time is at $t = 0.7[\mathrm{s}]$ and the motion ends at $t = 1.4[\mathrm{s}]$. After 2 hours of the optimization process, we acquired optimized joint trajectories. The acquired optimized swing motion is shown in Fig. 4.

B. Decision of Balance Margin; Δx_{zmp}

To achieve tennis forehand stroke motion, the robot has to predict a ball's trajectory. It is desirable for a legged robot not to slip or rotate while the robot is predicting a ball's trajectory. Hence, the balancing condition is set so that both the robot feet do not rotate or slip while the robot is moving.

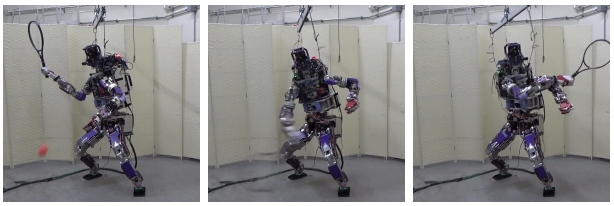


Fig. 4. Initial motion
The three poses are the initial pose (t = 0.0[s]), the hitting pose (t = 0.7[s]) and the final pose (t = 1.4[s]). Note that this motion only plays motion acquired from the offline optimization. The motion does not include the real-time optimization in our proposed system.

In this section, the decision process of the balance margin Δx_{zmp} is explained. The ZMP position on the ground at a certain time $\boldsymbol{x_{zmp}}(t)$ is calculated by inverse dynamics using the planned joint angles and their derivatives. To evaluate the dynamic balancing conditions, the shortest distance between $\boldsymbol{x_{zmp}}(t)$ and the edge set of the support polygon $\boldsymbol{L}$ is defined as $d(\boldsymbol{L}, \boldsymbol{x_{zmp}}(t))$. The function is positive when the $\boldsymbol{x_{zmp}}(t)$ is inside the support polygon, and it is negative when the $\boldsymbol{x_{zmp}}(t)$ is outside the support polygon. By using this function, the balancing condition is described as Eq. 8.

$$d(\boldsymbol{L}, \boldsymbol{x_{zmp}}(t)) \geq \Delta x_{zmp} \quad (8)$$

The ZMP trajectory is changed when the joint angles are modified, so the ZMP margin of the support polygon Δx_{zmp} has to be decided to prevent the robot from rotation and slip. The ZMP margin Δx_{zmp} is decided by the following process. The ZMP position $\boldsymbol{x_{zmp}}(t)$ calculated by inverse dynamics must be fully inside the support polygon. Given that both the robot's feet do not rotate or slip, as shown in Fig. 3, the support polygon is shorter for the robot's front-back direction than its left-right direction. The robot has a racket in its right hand and it modifies its right arm joint trajectories in real time, thus the ZMP position changes mainly to the robot's front-back direction by the right arm joint trajectory modification.

The robot's ZMP is calculated as Eq. 9.

$$\boldsymbol{x_{zmp}} = \frac{\sum_{i=1}^{n_i} m_i \left(x_i(\ddot{z} + g) - z\ddot{x_i}\right) - \sum_{i=1}^{n_i} I_i \dot{\omega_i}}{\sum_{i=1}^{n_i} m_i (\ddot{z} + g)} \quad (9)$$

where i means the ith joint, m_i is the ith link mass, x_i is the ith link position, I_i is the moment of inertia of the ith link, ω_i is the joint velocity of the ith link and z is the center of gravity. The robot dynamics model includes the racket model used in the experiment. The robot modifies its right arm joint angles, hence the biggest contribution to the robot's front-back ZMP position change from the planned motion is the right arm shoulder pitch joint torque difference during modification. The maximum difference of the robot's shoulder-pitch torque from the planned torque is defined as $\Delta\tau$. Using this value, the ZMP change Δx_{zmp} is approximated by Eq. 10. Note that the acceleration differences of the robot's links from the original motion are approximated

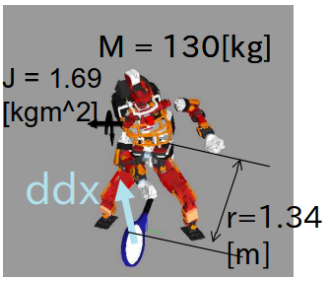


Fig. 5. The variables and their coordinates when estimating the pre-margin

Fig. 6. Stereo cameras attached to the robot's head

as $\mathbf{0}$.

$$\Delta x_{zmp} = \frac{|\Delta\tau|}{\sum_{i=1}^{n_i} m_i g} \tag{10}$$

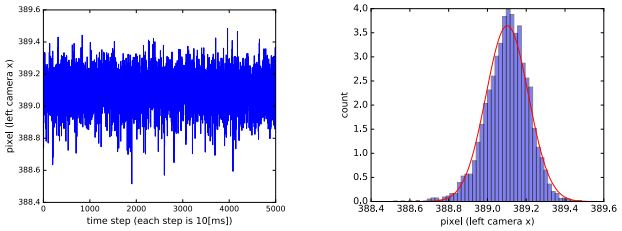
The value $\Delta\tau$ has to be calculated to estimate Δx_{zmp}. This value can be estimated from the hitting point acceleration. A hitting point is the intersection point of the predicted trajectory and the pre-decided hitting plane at a certain time. The hitting point acceleration is calculated as 2nd-order numerical derivatives of time-changing hitting point position in the world coordinates. The mean value of the predicted hitting point trajectory z-directional accelerations $\ddot{r_z}$ is about $82.0[\mathrm{m/s^2}]$. The torque needed to achieve this acceleration can be calculated as follows:

$$\Delta\tau = J\frac{\ddot{r_z}}{r} \tag{11}$$

where $J = 1.69[\mathrm{kgm^2}]$ is the right arm moment of inertia around the shoulder pitch joint axis, $r = 1.34[\mathrm{m}]$ is the distance between the right arm joint axis and the racket sweet spot. The robot's total mass is $130[\mathrm{kg}]$. These variables are illustrated in Fig. 5. Using Eq. 10 and Eq. 11, the value Δx_{zmp} is calculated as $\Delta x_{zmp} = 0.08[\mathrm{m}]$

IV. Real-time Ball Trajectory Prediction Using Stereo Camera Images

In order to achieve a tennis forehand stroke motion, a ball trajectory prediction system is included in the system. A lot of ball trajectory prediction algorithms were proposed. In [1], [15], [16], active stereo camera systems are used. In our work, stereo cameras are attached to the robot's head (See Fig. 6). Although the cameras move in the world coordinates, Thus the cameras are fixed with respect to the robot's head link. Our ball trajectory prediction algorithm is similar to [17] and [18] except that the position and the attitude of the camera base and their derivatives have to be considered.



(a) pixel (b) histogram

Fig. 7. Left camera pixel x values from ball segmentation and gaussian function fit to pixel histogram
The pixels in the figure is from an experiment. In the experiment, we placed a static ball on a desk and a robot looked at the ball from a fixed position.

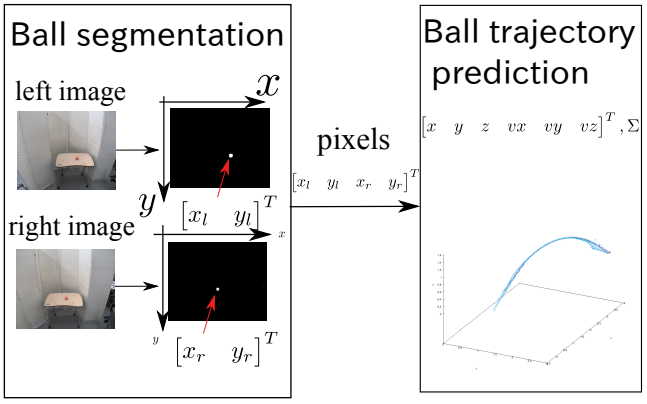


Fig. 8. Ball trajectory prediction overview
A computer inside the robot acquires the left and right camera images. From each image, centroid pixels of the ball region are extracted. The EKF computation node estimates the ball's position and velocity from the left and right pixels extracted.

The specification of the camera is shown in Table. I. To modify the motion quickly, the trajectory prediction system has to be real-time. In this paper, we consider the situation where a ball flies for only around 1.0[s] toward a robot. The distance between the two stereo cameras is about 0.23[m], which is narrower than the 0.8[m] compared with the existing literature[1]. Due to this restriction, the pixel noises from ball segmentation are amplified in the triangulation process and they transform into depth calculation noises. Thus a noise filtering process is needed to hit a ball. The pixel noises behaved nearly in gaussian manners (Fig. 7), thus gaussian-based filtering algorithm can be used. There are various prediction algorithms such as Particle Filter (PF) or Ensemble Kalman Filter (EnKF), but the Extended Kalman Filter (EKF) algorithm was chosen because it is a computationally lightweight analytical algorithm. The images acquired from the camera are processed in 100[Hz] in this work. The ball trajectory prediction system overview is shown in Fig. 8. First, regions of a ball on the stereo cameras are acquired by primitive image processing. Then, the centroid pixels of the regions are extracted. Finally, the location and the velocity of a ball are estimated using the output pixels.

TABLE I
BLACKFLY S (CAMERAS) AND LM5NC1L (LENSES) SPECS

Type	Spec
Model number	BFS -U3 -13Y3C
Max frame rate	170[frame/s]
Used frame rate	100[frame/s]
Shutter Mode	Global Shutter
Resolution (height)	1024[pixel]
Resolution (width)	1280[pixel]
Type	**Spec**
Model number	LM5NC1L
Focal length	4.5[mm]
Shortest photographing distance	0.2[m]
Angle of view	79.0°

A. Primitive Ball Segmentation from Stereo Images

In this subsystem, to increase the stereo image processing calculation frequency, stereo matching algorithms are not used because its computation costs are high. Instead, the centroid pixels where the ball exists are extracted from each of the images.

The segmentation process is as follows:

- Stereo images from the cameras are acquired
- The images are resized in the half size of the original image
- Using thresholding operation based on HSV(Hue, Saturation, Value) color space and the opening closing image manipulation morphological operation, regions where the ball shows are extracted
- Centroid pixels of the ball regions are acquired

Full size image processing takes more than $0.02[\mathrm{s}]$, which means the execution frequency is lower than $50[\mathrm{Hz}]$. Thus the images are resized in the half size of the original image in the software process to make the calculation time shorter. Note that all the calculation sequences are processed in the robot's internal computer, and the segmentation process is vulnerable to outliers. The center of the pixels in the left and right images are used in Sec. IV-B.

B. Ball Trajectory Prediction Using Extended Kalman Filtering

In this subsystem, the ball's 3D positions and velocities from the world origin are estimated by EKF. We assumed that a ball behaves in a parabolic motion, thus the ball trajectory is decided by the ball position and velocity.

The ball state $\boldsymbol{x_k}$ in EKF is described as follows:

$$\boldsymbol{x_k} = \left[x_k, y_k, z_k, \dot{x_k}, \dot{y_k}, \dot{z_k}\right]^T \tag{12}$$

where k is the sequential number for each time step. $\left[x_k, y_k, z_k\right]^T$, $\left[\dot{x_k}, \dot{y_k}, \dot{z_k}\right]^T$ is the world position (in $[\mathrm{m}]$ respectively) and velocity (in $[\mathrm{m/s}]$ respectively) of the ball.

The initial ball state $\boldsymbol{x_0}$ is acquired by the conventional triangulation method. The initial ball position is acquired by the triangulation method and the initial ball velocity is calculated on the hypothesis that the ball reaches the predecided hitting point at a 45 degree angle in a parabolic manner. From our experience, the initial ball velocity do not have to be accurate unless the velocity value is extremely different from the ground truth. At first we calculated the initial velocity from the first two acquired points, but the calculated initial speed of a ball was sometimes extremely fast because of measurement noises.

The state equation in EKF is described as follows:

$$\boldsymbol{x_k} = \begin{bmatrix} I_3 & I_3\Delta t \\ 0_3 & I_3 \end{bmatrix} \boldsymbol{x_{k-1}} + \begin{bmatrix} \frac{1}{2}\boldsymbol{g}(\Delta t)^2 \\ \boldsymbol{g}\Delta t \end{bmatrix} + \boldsymbol{w_k} \tag{13}$$

where $\boldsymbol{g} = \left[0, 0, -9.8\right]^T [\mathrm{m/s^2}]$ is the acceleration of gravity, $\Delta t \sim 0.01[\mathrm{s}]$ is the interval between time step $k-1$ and k, and $\boldsymbol{w_k}$ is the model noise.

The observation $\boldsymbol{z_k}$ is defined as follows:

$$\boldsymbol{z_k} = \left[x_{l,k}, y_{l,k}, x_{r,k}, y_{r,k}\right]^T \tag{14}$$

where $x_{l,k}, y_{l,k}$ is the pixels acquired from the left camera segmentation output, $x_{r,k}, y_{r,k}$ is the right output pixels.

The observation equation in EKF is described as follows:

$$\boldsymbol{z_k} = h(\boldsymbol{x_{k-1}}) + \boldsymbol{v_k} \tag{15}$$

$$\sim \frac{\partial h}{\partial \boldsymbol{x_{k-1}}} \boldsymbol{x_{k-1}} + \boldsymbol{v_k} \tag{16}$$

where h is the nonlinear function that transforms the world position and velocity of the ball $\boldsymbol{x_{k-1}}$ into pixels. $\boldsymbol{v_k}$ is the pixel noises of the segmentation process from each image.

The nonlinear function h is defined as follows: let $R^o_{cam,k}$ and $\boldsymbol{p^o_{cam,k}}$ be the attitude and position of the camera coordinate system from the world origin.

The ball position from the camera coordinate system $\left[X_k, Y_k, Z_k\right]^T$ is calculated as follows.

$$\begin{bmatrix} X_k \\ Y_k \\ Z_k \end{bmatrix} = {R^o_{cam,k}}^T \left(\begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} - \boldsymbol{p^o_{cam,k}} \right) \tag{17}$$

Let P_l and P_r be the projection matrices of left and right cameras.

$$P_l = \begin{bmatrix} P_{l00} & 0 & P_{l02} & 0 \\ 0 & P_{l11} & P_{l12} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{18}$$

$$P_r = \begin{bmatrix} P_{r00} & 0 & P_{r02} & P_{r03} \\ 0 & P_{r11} & P_{r12} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{19}$$

The pixels $x_{l,k}, y_{l,k}, x_{r,k}$ and $y_{r,k}$ are calculated using left and right focal lengths f_l and f_r.

$$\begin{bmatrix} x_{l,k}f_l \\ y_{l,k}f_l \\ f_l \end{bmatrix} = P_l \begin{bmatrix} X_k \\ Y_k \\ Z_k \\ 1 \end{bmatrix}, \begin{bmatrix} x_{r,k}f_r \\ y_{r,k}f_r \\ f_r \end{bmatrix} = P_r \begin{bmatrix} X_k \\ Y_k \\ Z_k \\ 1 \end{bmatrix} \tag{20}$$

As a consequence, we get a transformation matrix.

$$\begin{bmatrix} x_l \\ y_l \\ x_r \\ y_r \end{bmatrix} = \begin{bmatrix} P_{l00}Z_k^{-1}X_k + P_{l02} \\ P_{l11}Z_k^{-1}Y_k + P_{l12} \\ P_{r00}Z_k^{-1}X_k + P_{r02} + P_{r03}Z_k^{-1} \\ P_{r11}Z_k^{-1}Y_k + P_{r12} \end{bmatrix} \tag{21}$$

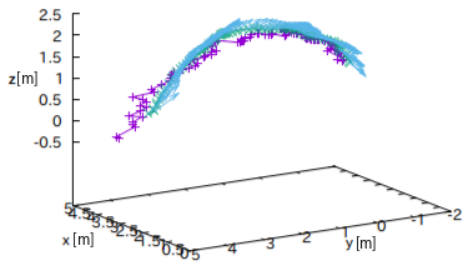


Fig. 9. Predicted ball trajectory
The purple points are acquired by the raw triangulation approach and the blue arrows are acquired by our EKF approach. The directions of blue arrows are directions of a ball's velocity. Our approach can estimate a ball's position and velocity, which means the ball's trajectory can be predicted. Note that our approach considers noises from the ball region segmentation.

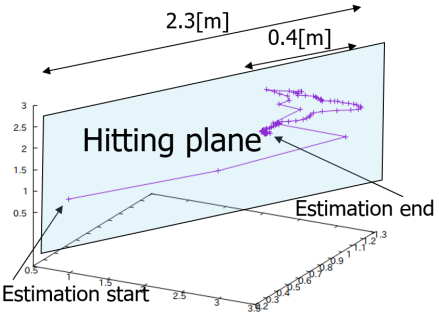


Fig. 10. The time-changing hitting points and the hitting plane.
A hitting point is the intersection point of the predicted trajectory and the pre-decided hitting plane at a certain time.

The function $h(\boldsymbol{x_k})$ is calculated using Eq. 17 and Eq. 21.

One of the extracted trajectory is shown in Fig. 9. Compared with the raw triangulation method (the purple points), the EKF approach in this work (the blue arrows) considers the ball segmentation pixel noises. The raw triangulation method is a method to estimate only the position of a ball, but our approach estimates the velocity of a ball, which means the trajectory of a ball can be predicted.

V. Real-time Joint Angle Modification

In order to hit a ball, every time a ball's trajectory is predicted and the reference joint angle trajectory of a robot is calculated accordingly.

A. Decision of Hitting Point

We set the initial motion generation condition so that the motion starts at $t = 0.0[\mathrm{s}]$, the desired hitting time is at $t = 0.7[\mathrm{s}]$ and the motion ends at $t = 1.4[\mathrm{s}]$. Thus the desired hitting time $t = 0.7[\mathrm{s}]$ is predetermined regardless of the ball trajectory. A robot has to decide the desired hitting point in real time from the predicted ball trajectory. The predicted hitting point is the intersection point of the predicted trajectory and the pre-decided hitting plane. One of the predicted trajectories is shown in Fig. 10. The estimated position and velocity of a ball change with time while a ball is flying. Using EKF algorithms, we obtain the covariance of a ball's state vector. The predicted hitting points are not used for the first $0.2[\mathrm{s}]$ as the prediction result did not converge enough to hit a ball.

B. Real-time Joint Angle Modification

The system is based on our previous work[11]. This base system is implemented in this work in the real-time programming layer and the target hitting point is used to hit the ball with the racket. The system includes a real-time balance stabilization system based on the Linear Inverted Pendulum model[4]. The sweet-spot of the racket must get through the hitting point at the hitting time. The right arm end effector position and attitude are easily calculated from the predicted hitting point position and attitude by static transformation. Every time the predicted hitting point changes, we solve the inverse kinematics problem and the optimization problem for the right arm joint trajectory modification. Note that this system works when the hitting point is in the region where the inverse kinematics calculation can be solved. The sum of squares of the angular difference from the initial joint angles is minimized using Quadratic Programming (QP).

The objective function is as follows:

- The sum of squares of the angular difference from the current joint trajectory

The equality conditions are as follows:

- Current pose joint angles
- Hitting pose joint angles

The inequality conditions are as follows:

- Joint angle limitations
- Joint angular velocity limitations

These conditions are all linear conditions, thus using a QP solver[19], this problem can be solved in real time in 100[Hz] frequency.

The balancing conditions are considered in the offline motion generation phase so that both the robot's feet do not rotate or slip while the robot is modifying its motion, hence it isn't necessary to calculate them in real time. The collision conditions are considered in the offline motion generation phase, but in the real time phase they are not included in the conditions. For the safety, the robot's motion will stop when the robot's internal controller detects self-collisions.

VI. Experiment and Result

The proposed system was applied to the tennis swing motion. The top-right number in Fig. 14 is the elapsed time from when the swing motion started.

We carried out dynamics simulations using the robot's model[20]. The simulation includes simulation of a ball movement (Fig. 11).

Fig. 12 shows the comparison of the planned ZMP trajectory and the actual ZMP trajectory under $\Delta x_{zmp} = 0.08[\mathrm{m}]$.

We conducted an experiment with a real humanoid robot. The horizontal component of the ball speed in the experiment was about $5\mathrm{m/s}$. The racket maximum speed was about $4\mathrm{m/s}$ (See Fig. 13). Fig. 14 shows a real humanoid robot

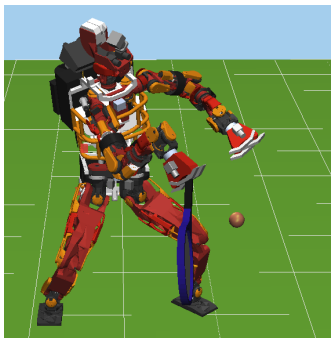
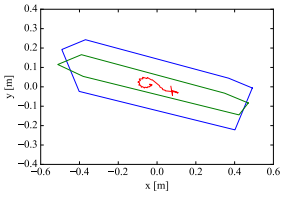
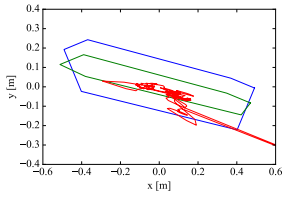


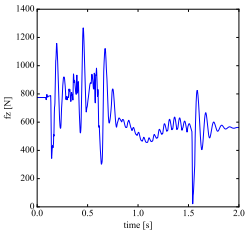
Fig. 11. Dynamics simulation environment
The simulator handles the robot dynamics simulation and the camera images simulation. As a ball is thrown from $4[\mathrm{m}]$ from the robot, the robot performs a motion using our proposed real-time system.



(a) Planned ZMP trajectory



(b) Actual ZMP trajectory in simulation



(c) Right foot force sensor value

Fig. 12. Comparison of planned and actual ZMP trajectories
This Figure shows the comparison of the planned ZMP trajectories and the actual ZMP trajectories under $\Delta x_{zmp} = 0.08[\mathrm{m}]$ in dynamics simulations. The blue polygons in Fig. (a) and Fig. (b) are the support polygons of both the robot's feet. The green polygons are restricted support polygons with the margin $\Delta x_{zmp} = 0.08[\mathrm{m}]$. The orange lines are the planned and actual ZMP trajectories respectively. In Fig. (b), the actual ZMP trajectory is outside of the polygon instantly. This is because the robot's right foot force sensor value was nearly zero (Fig. (c)) and the simulator could not calculate the actual ZMP value properly.

forehand stroke motion without falling down using the ZMP margin value $\Delta x_{zmp} = 0.08[\mathrm{m}]$.

The proposed system has computational latencies and target joint angle following-up latencies. The computational latencies consist of the following three latencies.

1) USB communication latency between the stereo cameras and the computer for visual processing
2) Ball region segmentation and EKF calculation latency
3) Inverse kinematics and QP calculation latency

The cameras support USB3.0 standard. According to the time stamp difference between the cameras and the computer, the USB communication latency was less than $0.01[\mathrm{s}]$. The visual processing and the EKF calculation node was executed at $100[\mathrm{Hz}]$, thus the latency was less than $0.01[\mathrm{s}]$. The Inverse kinematics calculation time was up to $0.003[\mathrm{s}]$ and QP calculation time was up to $0.002[\mathrm{s}]$. The target joint angle following-up latencies are up to $0.008[\mathrm{s}]$ The sum of these latencies was up to $0.033[\mathrm{s}]$. The horizontal component of the ball speed was about $5\mathrm{m/s}$, thus the real ball horizontal position was $0.165[\mathrm{m}]$ ahead of the predicted ball horizontal position in the worst case. The tennis forehand stroke experiment was successful in spite of this latencies because the ball's trajectory and the racket's trajectory are similar except for their direction.

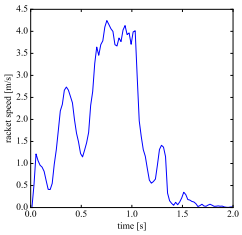


Fig. 13. The racket velocity norm while swinging.
The racket velocity in the graph was acquired by forward kinematics calculation using the experimenal logs. The robot's base position and the velocities of the joints were used in the calculation. Note that the racket speed did not significantly change when hitting the ball ($t = 0.8$).

The proposed system has hitting time prediction errors, hitting point prediction errors and target joint angle following-up errors. The system could predict the hitting time in $0.2[\mathrm{s}]$, and the hitting time prediction error was less than $0.1[\mathrm{s}]$. It takes $0.7[\mathrm{s}]$ to perform the swing motion, thus the robot cannot hit a ball which flies less than $0.9[\mathrm{s}]$ theoretically.

The variance from EKF output in the predicted hitting point was on the order of $0.01[\mathrm{m}]$ around the hitting time, but there were cases where the robot failed to hit the ball because the ball is out of the region where the inverse kinematics calculation fails. Also, the robot failed to hit the ball in cases when the ball hit the ground and bounced because the ball is expected to fly in a parablic manner.

VII. Conclusions and Future Work

When a humanoid robot performs whole-body fast motions based on visual information, the robot motion is desired to be quick and the robot has to modify its joint angles to meet the condition of the visual information while keeping its balance. Estimation of the ZMP trajectory change in the offline planning prevents the robot from falling down in real time and implementation of the real-time joint trajectory modification leads to the achievement of the hitting motion in $0.7[\mathrm{s}]$. Owing to the EKF prediction system considering visual noises, a humanoid robot can modify the racket trajectory to hit a ball in real time. The ball trajectory prediction process and the joint trajectory modification process are achieved in a standalone humanoid robot real-time system. We confirmed that the real-time visual feedback considering noises and the real-time joint trajectory modification can

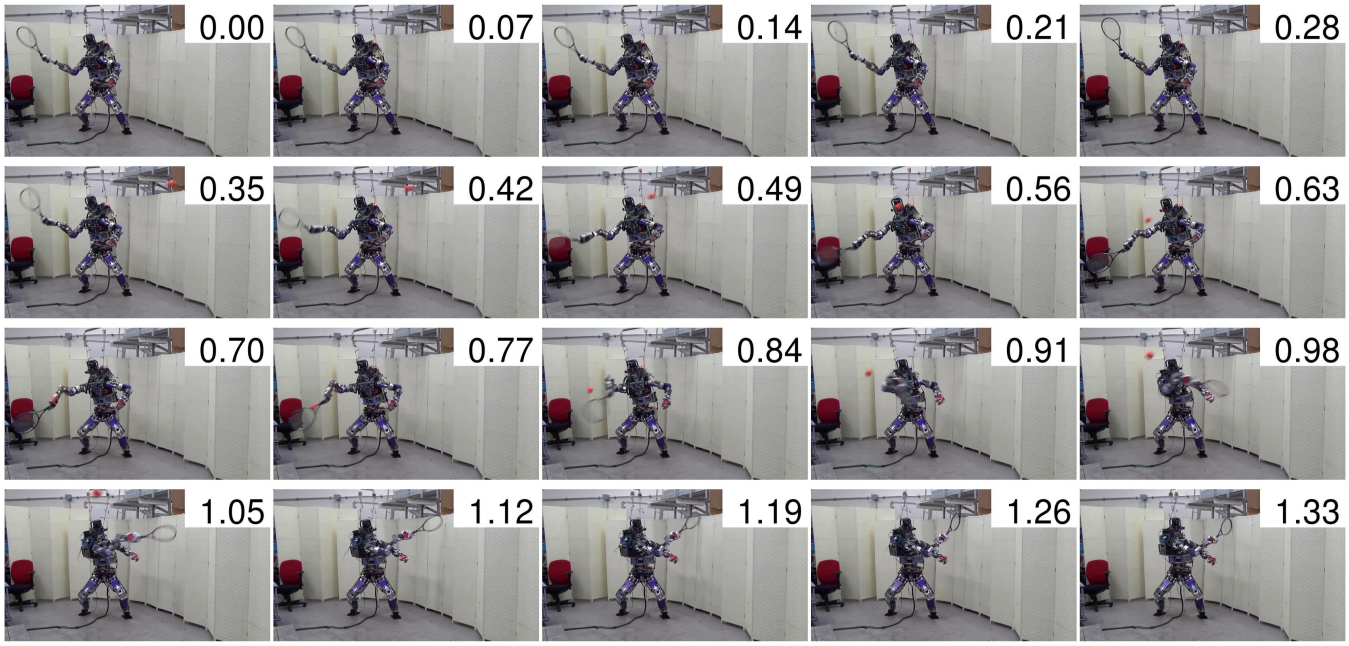


Fig. 14. Tennis forehand real-robot swing motion with modification.The top-right number is the elapsed time from when the swing motion started.

achieve whole-body motion reacting to the visual information.

An important issue to resolve for future work is the fast locomotion generation while a robot is performing fast motions. A ball's trajectory in the world coordinates could be predicted in this paper since we set the conditions so that both a robot's feet do not rotate or slip, which means both the positions of the robot's feet are fixed to the world coordinates.

REFERENCES

[1] T. Senoo, A. Namiki, and M. Ishikawa. High-speed batting using a multi-jointed manipulator. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1191–1196 Vol.2, 2004.

[2] S. Mori, K. Tanaka, S. Nishikawa, R. Niiyama, and Y. Kuniyoshi. High-speed and lightweight humanoid robot arm for a skillful badminton robot. *IEEE Robotics and Automation Letters*, Vol. 3, No. 3, pp. 1727–1734, 2018.

[3] D. Kappler, F. Meier, J. Issac, J. Mainprice, C. G. Cifuentes, M. Wuthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, Vol. 3, No. 3, pp. 1864–1871, 2018.

[4] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 239–246 vol.1, 2001.

[5] T. Takenaka, T. Matsumoto, T. Yoshiike, and S. Shirokura. Real time motion generation and control for biped robot-2(nd) report: Running gait pattern generation. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1092 – 1099, 2009.

[6] S. Hong, Y. Oh, D. Kim, and B. You. Real-time walking pattern generation method for humanoid robots by combining feedback and feedforward controller. Vol. 61, No. 1, pp. 355–364, 2014.

[7] T. Ishikawa, Y. Kojio, K. Kojima, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Bipedal walking control against swing foot collision using swing foot trajectory regeneration and impact mitigation. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4531–4537, 2017.

[8] S.-H. Hyon, J. Moren, and G. Cheng. Humanoid batting with bipedal balancing. In *Proceedings of the 2008 IEEE-RAS International Conference on Humanoid Robots*, pp. 493–499, 2008.

[9] R. Gerndt, D. Seifert, J. H. Baltes, S. Sadeghnejad, and S. Behnke. Humanoid Robots in Soccer: Robots Versus Humans in RoboCup 2050. *IEEE Robotics Automation Magazine*, Vol. 22, No. 3, pp. 147–154, 2015.

[10] R. Xiong, Y. Sun, Q. Zhu, J. Wu, and J. Chu. Impedance control and its effects on a humanoid robot playing table tennis. *International Journal of Advanced Robotic Systems*, Vol. 9, No. 5, p. 178, 2012.

[11] R. Terasawa, S. Noda, K. Kojima, R. Koyama, F. Sugai, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Achievement of dynamic tennis swing motion by offline motion planning and online trajectory modification based on optimization with a humanoid robot. In *Proceedings of the 2016 IEEE-RAS International Conference on Humanoid Robots*, pp. 1094–1100, 2016.

[12] C. d. Boor. On calculating with b-splines. *Journal of Approximation Theory*, Vol. 6, pp. 50–62, 1972.

[13] The nlopt nonlinear-optimization package. `https://github.com/stevengj/nlopt`. Accessed on 2020-02-14.

[14] K. Svanberg. A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization*, pp. 555–573.

[15] M. Shibata and T. Honma. 3d object tracking on active stereo vision robot. In *7th International Workshop on Advanced Motion Control. Proceedings (Cat. No.02TH8623)*, pp. 567–572, 2002.

[16] S. Kao, Y. Wang, and M. Ho. Ball catching with omni-directional wheeled mobile robot and active stereo vision. In *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, pp. 1073–1080, 2017.

[17] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hahnle, and G. Hirzinger. Off-the-shelf vision for a robotic ball catcher. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 3, pp. 1623–1629 vol.3, 2001.

[18] O. Birbach and U. Frese. A Multiple Hypothesis Approach for a Ball Tracking System. In M. Fritz, B. Schiele, and J. H. Piater, editors, *Computer Vision Systems*, pp. 435–444, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[19] J. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, Vol. 6, , 2014.

[20] S. Nakaoka. Choreonoid: Extensible virtual robot environment built on an integrated GUI framework. In *2012 IEEE/SICE International Symposium on System Integration (SII)*, pp. 79–85, 2012.