Controlling Posture of Jumping Articulated Robot for Stable Landing

Hotae Lee[1]

***Abstract*— We propose a new control framework of a jumping articulated robot for a stable landing. We derive dynamics of a hybrid system which consists of a flight phase and a stance phase by connecting them through an inelastic impact model of Formalsky. We assume a flight phase is a nonholonomic Chaplygin system and a stance phase is a fully-actuated system. Based on this dynamics, we propose new time-varying control with considerations for features of jumping such as joint angle limit, short duration of flight. It can make a robot get the desired angle within a specific range at the moment of landing. In addition, we find an optimal control to return a robot to an upright pose based on gain tuning. Simulations using a 4-link robot are also performed to show this visually. The motion from new control framework performs in the limit of joints other and requires less torque than conventional controls without a given trajectory.**

Fig. 1. Jumping motion to land on the ground from a high place

I. INTRODUCTION

Many bipedal robots have been developed to be able to move on various terrain. For example, Boston dynamics robot named Atlas [1] can walk through rough terrains like snowy mountain roads. Although bipedal robots have become adept at walking, they still have limitations on fast dynamical activities such as jumping. This behavior is hard to be performed precisely because the flight phase is common in these behaviors when the bipedal robots lose contact with the ground. In these cases, the systems are considered under-actuated systems and it is not easy to control these systems [2]. However, jumping is useful to travel to places of different heights to clear some high obstacles.

So, many researchers have developed novel jumping mechanisms and control frames for jumping highly and efficiently [3], [4]. But, most studies focused on only jumping ability. Although the robots they have built can jump with good performances, it cannot control its midair posture once leaping in to the air. Then, it cannot continue to next tasks and great jumping loses its meaning. Safe landing posture, however, can ensure the robot to protect it from damage and to be able to move continuously.

To this end, researchers tried to solve this problem through robust design which can resist the impact. For instance, Mowgli robot [5] can land without falling due to its large feet and Salto-1P [6] can resist an impact due to its linkage-spring design. But, this approach based on design has limitations which is not applicable to general bipedal robots.

Therefore, posture control in flight phase should be considered. There are already posture control frames for 2-link flying acrobot [7]. Also, there are approaches to control aerial maneuver of robot's body based on tail [8], [9]. Moreover, three link space robot's attitude can be controlled toward the desired configuration angle [10]. However, there is no control frame for a jumping bipedal multi-joint robot in flight phase, considering various features of jumping motions. After an impact, how to control the joint angles without given trajectory is also important to return to an upright pose, which is an initial pose.

In this paper, new posture control frame for a 4-link robot which includes torso, arm, upper leg, lower leg is developed. In a flight phase, this control frame is based on a partial feedback linearization with time-varying feedback. This system is nonholonomic system and nonholonomic constraint is angular momentum conservation. Our contribution is the suggestion of new time-varying control which considered limitations of joint angles and a short duration of flight. In this control framework, we change the order of steps from the previous posture control. To do this, we develop algorithm to find the required input for a given angle and upper and lower bound of torso angle that we can get. Moreover, our contribution also includes the suggestion of optimal control to return a robot to an upright pose, which considered an impact model.

The rest of this paper is organized as follows. In Sec. II, we derived dynamics of a flight phase and a stance phase by using Lagrange dynamics. Also, we decomposed the dynamics into internal rotation and transition. The inner-loop control and the outer-loop control, new control law of a flight phase, is established in Sec.III. Algorithm to find the required input for new control framework is presented in Sec.IV. Optimal control to return a robot to an upright pose is revealed in Sec.V. All simulation results are presented in

*This work was supported by SNU Undergraduate Research Program

[1]Hotae Lee is with the Department of Mechanical and Aerospace Engineering, Seoul National University, Seoul, South Korea hotae319@snu.ac.kr

Sec.VI, and Sec.VII concludes the paper.

II. DYNAMICS MODEL OF FOUR LINK ROBOT

A. Lagrange dynamic model

Studied bipedal robot evolves in the sagittal plane as shown in Fig. 2, where the four link robot is composed of the torso, arm, upper leg and lower leg. Especially, we suppose legs and arms are bilaterally symmetric when considering only jumping and free flying. It means one leg and one arm are enough to analyze the movement in the flight phase. The inertial coordinate frame x-y is attached and the position of end point is indicated by (x, y). The absolute angles and relative angles are defined as shown in Fig. 2. The biped's absolute angles of the torso, arm, upper leg, lower leg are q_0, q_1, q_2, q_3 and they are defined as angles with respect to x-axis. The relative angles are $\theta = [\theta_1;\theta_2;\theta_3]$, which is known as shape variables [11]. The parameters such as the mass and the length of torso, arm, upper leg, and lower leg are indicated with $\lambda_i (i = 1, 2, 3, 4)$, respectively. Generalized coordinates can be $[x; y; q_0; q_1; q_2; q_3]$ or $[x; y; q_0; \theta_1; \theta_2; \theta_3]$. We can divide landing process into three steps, flight phase, impact and stance phase. The robot is said to be in the flight phase when there is no contact with the ground. In the flight phase, while this robot has 4 DoF(Degree of Freedom), it has 3 input torques. The actuators are provided at each joints and ankle joint's actuator does not act in the flight phase. After the robot contacts the ground, the stance phase starts. In the stance phase, it has 4 input torques because the ankle joint actuator can work. The Euler-Lagrange method is used to obtain the dynamic model of this four link robot [12]. For this method, the Lagrangian is defined as $L = K - V$, and the equations are determined as below.

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau \tag{1}$$

There are two dynamics equations depending on the phase. In the flight phase, the generalized coordinates are $q_f = [x; y; q_0; \theta_1; \theta_2; \theta_3]$ and the dynamics are described as below.

$$M_f(\theta)\ddot{q}_f + C_f(\dot{q}_f)\dot{q}_f + g(q_f) = \tau_f \tag{2}$$

$$\tau_f = [0; 0; 0; -T_1; T_2; T_3] \tag{3}$$

where $M_f \in \Re^{6\times6}$ is the inertia matrix, $C_f \in \Re^{6\times6}$ is Coriolis and centrifugal matrix, g is the gravity vector. In the stance phase, we do not need to include x, y in the generalized coordinates. So, $q_s = [q_0; \theta_1; \theta_2; \theta_3]$ and the dynamics equations are described as below.

$$M_s(\theta)\ddot{q}_s + C_s(\dot{q}_s)\dot{q}_s + g(q_s) = \tau_s \tag{4}$$

$$\tau_s = [-T_1; T_2 - T_4; T_3 - T_4; T_4] \tag{5}$$

B. Passive decomposition of 4-link robot system

Applying the passive decomposition to (2), we can decouple the 4-link system dynamics into the transitional CoM(Center of Mass)dynamics and the internal rotational dynamics [13]. Since the (6) is about a simple projectile motion, we can only tackle the rotational motion for the flight phase.

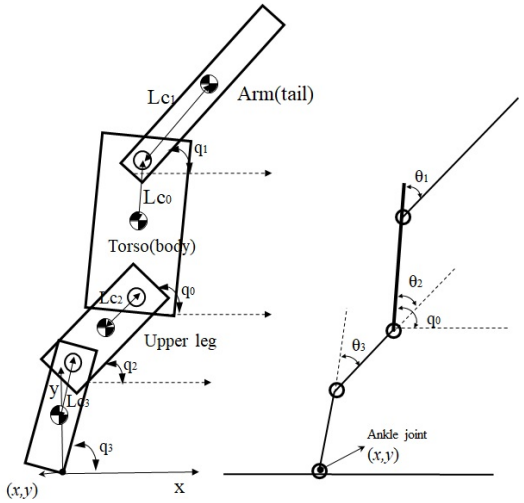

Fig. 2. Dynamics model of a 4-link robot

$$m_L\ddot{p}_L + g_L = \tau_L \tag{6}$$

$$M_E\ddot{q}_s + C_E\dot{q}_s = \tau_E \tag{7}$$

$$\tau_E = [0; -T_1; T_2; T_3] \tag{8}$$

C. Impact model

In the impact model [14], the impact of articulated robot is regarded as an inelastic impact. Thus, the contact point's velocity becomes zero and acts like an ideal pivot, which is actuated by ankle joint. When we assume the contact point is the end point of the lower leg, the position of the CoM is denoted as $f_{cm} = [x_{cm}; y_{cm}]$and the contact point is denoted as $[x; y]$. Also, $h(q)$ is the position of CoM expressed in terms of the contact point. It is determined by the parameters of robot.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{cm} \\ y_{cm} \end{pmatrix} - h(q_s) \tag{9}$$

After an impact, zeroing the velocity of the end point is described as (before is denoted as -, after is denoted as +)

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \dot{x}_{cm}^{+} \\ \dot{y}_{cm}^{+} \end{pmatrix} - \frac{\partial h(q_s)}{\partial q_s}\dot{q_s}^{+} \tag{10}$$

At that time, the Hurmuzlu's impact model is described as

$$\begin{pmatrix} M_E & 0 \\ 0 & m_L I_{2\times2} \end{pmatrix} \left[\begin{pmatrix} \dot{q}_s \\ \dot{x}_{cm}^{+} \\ \dot{y}_{cm}^{+} \end{pmatrix} - \dot{q}_f^{-} \right] = \begin{pmatrix} \frac{-\partial h(q_s)^T}{\partial q_s} \\ I_{2\times2} \end{pmatrix} I_R \tag{11}$$

where I_R is the ground reaction impulse. Substituting (10) into (11), we can get I_R as

$$I_R = m\left(\frac{\partial h(q_s)}{\partial q_s}\dot{q_s}^{+} - \begin{pmatrix} \dot{x}_{cm}^{-} \\ \dot{y}_{cm}^{-} \end{pmatrix} \right) \tag{12}$$

Finally, we can get the velocity after the impact as

$$\dot{q_s}^{+} = \left[M_E + m\frac{\partial h(q_s)^T}{\partial q_s}\frac{\partial h(q_s)}{\partial q_s} \right]^{-1} \left[M_E \middle| m\frac{\partial f_2(q_s)^T}{\partial q_s} \right] \dot{q_f}^{-} \tag{13}$$

III. CONTROL LAW DEVELOPMENT -FLIGHT PHASE

A. Nonholonomic Chaplygin system

There is no generalized torque in the direction of q_0 in the flight phase. Thus, angular momentum is conserved shown as below. $A_{(i,:)}$ means the i^{th} row vector of the matrix A, $A_{(i,j)}$ means the element of i^{th} row, j^{th} column.

$$M_{E(1,:)}\dot{q_s} = \sigma_{cm} = const. \tag{14}$$

Except for $M_{E(1,1)} = 0$, $\dot{q_0}$ is affine in the θ .

$$\dot{q_0} = \frac{\sigma_{cm}}{M_{E(1,1)}} - \frac{M_{E(1,2)}}{M_{E(1,1)}}\dot{\theta}_1 - \frac{M_{E(1,3)}}{M_{E(1,1)}}\dot{\theta}_2 - \frac{M_{E(1,4)}}{M_{E(1,1)}}\dot{\theta}_3 \tag{15}$$

Because M_E is also function of only θ, q_0 is called a cyclic coordinate for the system. The system with this structure is referred to as a nonholonomic Chaplygin systems [15]. Since the goal of jumping is to move, it would be favorable that every input energy is used for a transitional movement. Although real jumping movements can be different from this, we consider the angular momentum as zero to simplify the problem in this paper.

B. Partial Feedback Linearization-Inner loop control

Based on dynamics, this system is under-actuated (4 DoF, 3 Input torque) and we cannot control all state variables independently. Thus, we should surrender to control one state variable directly. In a nonholonomic Chaplygin system, it is general to use shape variables to control cyclic coordinates. In this paper, we apply partial feedback linearization to this dynamics. Shown as below, $\theta = [\theta_1; \theta_2; \theta_3]$ shape variables can be controlled by transforming equations and applying feedback linearization to partial elements.

$$\ddot{q_s} + M_E^{-1}C_E\dot{q_s} = M_E^{-1}[0 \quad T_1 \quad T_2 \quad T_3]^T \tag{16}$$

When $A_{(i_1:i_2,j_1:j_2)}$ is the submatrix which consists of i_1 to i_2 rows, j_1 to j_2 columns of A, we consider only 2, 3, 4 elements as next,

$$\begin{pmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\\ \ddot{\theta}_3\end{pmatrix} + M_E^{-1}C_E\dot{q_s}_{(2:4,1)} = (M_E^{-1})_{(2:4,2:4)}\begin{pmatrix}T_1\\ T_2\\ T_3\end{pmatrix} \tag{17}$$

$$\therefore \begin{pmatrix}T_1\\ T_2\\ T_3\end{pmatrix} = ({M_E^{-1}}_{(2:4,2:4)})^{-1}(u + M_E^{-1}C_E\dot{q_s}_{(2:4,1)}) \tag{18}$$

In order to control $\ddot{\theta}$ with an outer loop control u such as $\ddot{\theta} = u$, we designed the controller as above (18). So, we can control shape variables as wanted by designing u. Torque input also can be determined as above if we choose u.

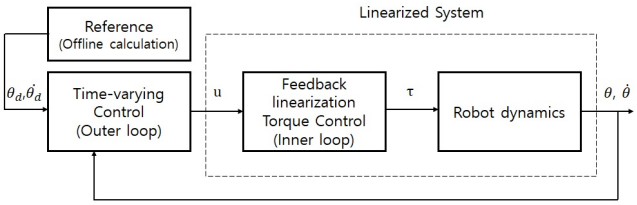

Fig. 3. Simple controller diagram

C. Time-varying feedback law based on a cycle path

Conventional nonholonomic Caplygin system uses cyclic coordinate control based on holonomy angle for reconfiguration. Holonomy angle is a joint drift of a cyclic coordinate after one cycle Γ of u [16], [17] in a nonholonomic Chaplygin system. We called this drift angle and it is parameterized by path size(d). If we know how much the drift will occur with respect to the cyclic path size, we can choose the required path size to drive q_0 to the desired q_0. In kinematics, we can apply directly cyclic velocity input which follows the cyclic path. In this paper, we use the feedback input instead of feedforward input to drive q_0 to q_{0d} because we consider the dynamics, not kinematics. As you can see in Fig. 4, We can reach the desired shape within finite time in step 1. After we have the desired shape, we apply torque input to follow the cyclic path in the shape configuration space to get the desired drift angle. We chose isosceles right triangle (side length = d) cyclic path on θ_1, θ_2 in step 2. It can be shown as below.

Step 1 (to get the desired shape variables)

$$u = \begin{pmatrix}-c_d\dot{\theta_1} - k_d(\theta_1 - \theta_{1d})\\ -c_d\dot{\theta_2} - k_d(\theta_2 - \theta_{2d})\\ -c_d\dot{\theta_3} - k_d(\theta_3 - \theta_{3d})\end{pmatrix} (t_0 < t \leq t_1) \tag{19}$$

Step 2 (to get the desired drift angle)

$$u = \begin{cases}\begin{pmatrix}-c\dot{\theta_1} - k(\theta_1 + d - \theta_{1d})\\ -c\dot{\theta_2} - k(\theta_2 - d - \theta_{2d})\\ -c\dot{\theta_3} - k(\theta_3 - \theta_{3d})\end{pmatrix} & (t_1 < t \leq t_2)\\ \begin{pmatrix}-c\dot{\theta_1} - k(\theta_1 - \theta_{1d})\\ 0\\ -c\dot{\theta_3} - k(\theta_3 - \theta_{3d})\end{pmatrix} & (t_2 < t \leq t_3)\\ \begin{pmatrix}0\\ -c\dot{\theta_2} - k(\theta_2 - \theta_{2d})\\ -c\dot{\theta_3} - k(\theta_3 - \theta_{3d})\end{pmatrix} & (t_3 < t \leq t_4)\end{cases} \tag{20}$$

The motions derived from this input is not natural and exceeds the limit angle of joint. In addition, it needs huge magnitude of torque in order to apply several steps of input within short times. So, we suggest new time-varying control, whose order of step is changed.

D. New Time-varying feedback law-outer loop control

We changed the order of step. We tried to get the drift angle in step 1 and get to the desired shape variables in

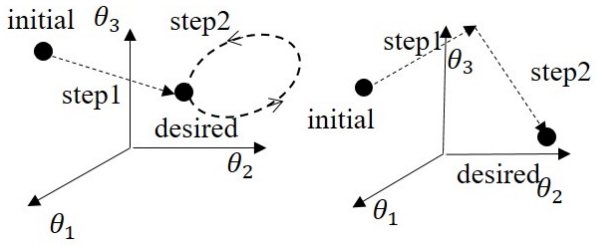

Fig. 4. Procedures on a shape variable space of cyclic control (left) and new time-varying control (right)

step 2. Although we can choose cyclic path to get the required additional angle, we decide to apply input control just once. Input control is applied in step 1 to go toward the desired shape directly. From now on, it is based on the displacement, not a cycle path size. It drives shape variables by the displacement Δ_i, which is in the direction of θ_i to get the desired drift of q_0. We called the point after moving by the displacement the intermediate point. Because a robot has joint angle limits, the intermediate point should be obtained in the range of joint angles. If the initial angular velocity of joints exists, we can settle down through initialization step. However, most jumping motions include the stretched behavior, which means the angular velocity of the joints are almost zero.

Step 1 (to get the desired drift angle)

$$u = \begin{pmatrix} -c\dot{\theta_1} - k(\theta_1 - (\theta_{1i} + \Delta_1)) \\ -c\dot{\theta_2} - k(\theta_2 - (\theta_{2i} + \Delta_2)) \\ -c\dot{\theta_3} - k(\theta_3 - (\theta_{3i} + \Delta_3)) \end{pmatrix} (t \leq t_1) \tag{21}$$

Next, damped PD control input to drive the specific shape into the desired shape is applied in step 2.

Step 2 (to get the desired shape variables)

$$u = \begin{pmatrix} -c_d\dot{\theta_1} - k_d(\theta_1 - \theta_{1d}) \\ -c_d\dot{\theta_2} - k_d(\theta_2 - \theta_{2d}) \\ -c_d\dot{\theta_3} - k_d(\theta_3 - \theta_{3d}) \end{pmatrix} (t_1 < t \leq t_2) \tag{22}$$

This path on shape configuration space generates the drift of q_0 in this system. If we calculate this drift with respect to displacement, we can find required input regardless of q_0 state. This calculation is from integration of (15)

We want to find the required displacement to get the desired drift angle offline. To do so, we need to know whether we can get the desired drift angle or not. Thus, we should find the maximum and minimum of the drift angle. Also, the drift angle is a continuous function of the displacements $\Delta_1, \Delta_2, \Delta_3$. Therefore, if we find maximum and minimum value of drift angle, we can find the required displacement to satisfy the desired drift from *intermediate value theorem*, which states if a continuous function f, with an interval $[a, b]$, as its domain, takes values $f(a)$ and $f(b)$ at each end of the interval, then it also takes any value between $f(a)$ and $f(b)$ at some point within the interval. In a next Section, the algorithm to find the required displacement to achieve the desired drift angle is introduced.

IV. ALGORITHM TO FIND THE REQUIRED DISPLACEMENT

In order to find the required displacement, we developed algorithm based on search on shape configuration space. As mentioned above, we should find the each angle's displacement and it means that we need to find the required intermediate point. Also, the limits of joint angle are described as below. These limitations are represented as a cuboid in a shape configuration space. So, we need to choose the intermediate point inside this cuboid. As mentioned earlier, we have to find the extreme value of drift angle first. We call this range [min, max] a reachable range. After we find it, we can find the required displacement due to *intermediate value theorem*.

A. Limits of joint angle

Limits of joint angle are based on human's body structure. In fact, a human can rotate his arm, while other joints cannot be turned around wholly. However, a human rotates his arm only once when we observed a human's jumping motion. So, we assume that robot does not rotate its arm several times. Thus, the possible range of joint angles is assumed as next.

$$\begin{aligned} -\frac{11}{6}\pi \leq & \theta_1 \leq \frac{\pi}{6} \\ -\frac{\pi}{6} \leq & \theta_2 \leq \pi \\ -\pi \leq & \theta_3 \leq 0 \end{aligned} \tag{23}$$

B. Algorithm to find a reachable range

Algorithm 1 To find a reachable range

$\gamma(\theta_1, \theta_2, \theta_3)$ = drift angle of q_0 when the intermediate point is $(\theta_1, \theta_2, \theta_3)$, Discretize cuboid as $n \times n \times n$ states
Input: Initial and desired shape variables
Initialization: $M \leftarrow max\ \gamma(\theta)$ ($\theta \in$ 8 cubiod vertices)
$\theta = \arg\max_\theta \gamma(\theta)$
while $M - M_{pre} > \varepsilon$ **do**
$M \leftarrow max_{\theta_1}\ \gamma(\theta)$ s.t. θ_2, θ_3 is fixed at previous stage
Iterate search along θ_2, θ_3 axis same as above
end while
Iterate same procedures of 'while' for a min $\gamma(\theta)$
Get a reachable range with θ $(\arg\max\gamma, \arg\min\gamma)$
end
End

If we choose the desired shape configuration, our control method can make shape variables reach to any values in the possible range of joint angles because shape variables are controlled directly. So, we need to decide q_0 drift's reachable range, given the final shape variable. A drift of q_0 does not depend on q_0 state like angular velocity and initial value. Then, we developed algorithm to find maximum drift angle and minimum drift angle inside the possible range of joint angles. For convenience, we searched the maximum and minimum value of the sum of initial angle and drift angle, which means the actual torso angle. The algorithm

also needs the desired shape variables. Given that, algorithm starts to search for maximum and minimum value of drift angle. Using KKT condition and gradient descent method is a strict method, but it requires a long computation time. Since we just need an approximate range to get the required displacement, we discretize the shape configuration space and set the greedy algorithm about each axis. We search along the axis while changing axis direction like $x \rightarrow y \rightarrow z \rightarrow x$ each step. We choose the maximum value on each step's axis and it can converge to approximate maximum value.

C. Algorithm to find the required displacement

If we confirm that the desired torso angle is in the reachable range, we should find the required displacement to reach the desired torso angle. This is based on intermediate value theorem. Bisection search narrows the range which include the solutions by comparing the present value with the desired value.

Algorithm 2 To find the required displacement

Input: Initial and desired q_0 and shape variables θ, $\gamma_{desired}$
  Initialization: $\theta_+ = \arg\max\gamma$, $\theta_- = \arg\min\gamma$
  while $|\gamma_{desired} - \gamma(\theta_{cur})| > \varepsilon$ **do**
    $\theta_{cur} = (\theta_+ + \theta_-)/2$
    if $\gamma(\theta_{cur}) < \gamma_{desired}$ **then**
      $\theta_- = \theta_{cur}$
    else
      $\theta_+ = \theta_{cur}$
    end if
  end while
  $\Delta = \theta_{cur} - \theta_{ini}$
End

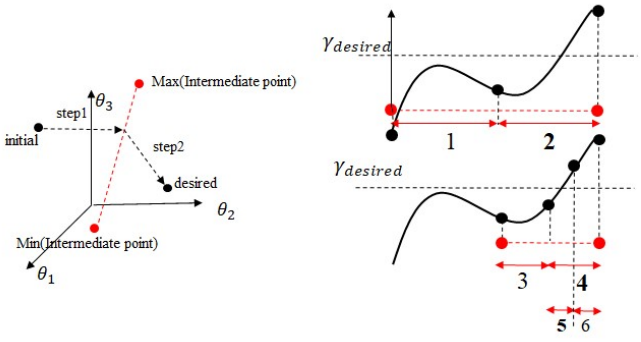

Fig. 5. Branch and Bound algorithm descriptions

V. OPTIMAL CONTROL-STANCE PHASE

After the impact, we assume that 4-link robot system is fully-actuated system. So, the controllability is assured. For a stable landing, a robot has to return to an upright pose. If any torque limits or energy limits are not given, there always exist controls to get the upright pose although the robot land at an arbitrary angle. In some previous researches, controls for landing were tracking controls which track a stable trajectory we already knew. Since we wanted to get the most stable landing process without the given trajectory, we proposed the optimal control which minimizes the energy use as a landing control. There are a lot of ways to solve optimal control problems such as the minimum-effort or the minimum-energy.

$$min \int_{t_0}^{t_f} u^2 dt \quad s.t \quad \dot{x} = f(x,u,t) \tag{24}$$

In this paper, we just assumed that the control is determined as the feedback linearization control and changed the control gain. We just compared the energy use and landing trace through a simulation in Sec. VI.

$$\tau = M_s(\theta)(-c_d\dot{q}_s - k_d(q_s - q_d)) + C_s(\dot{q}_s)\dot{q}_s + g(q_s)$$
$$q_d = [\pi/2; -\pi; 0; 0] \ (upright\ pose) \tag{25}$$

VI. SIMULATION RESULTS

Salto-1P already showed a great ability of jumping and landing with its great structure. So, we assume it can be a great simulation model. As a result, our simulation parameters are based on the Salto-1P's size and human body structure. We combined the human's ratio of mass and length with Salto-1P's specification. Also, aerial time and settling time are same as a tailbot's thing and gain is set critically damped. The simulator is MATLAB and Simulink. We set the state as $[q_0; \theta_1; \theta_2; \theta_3; \dot{q}_0; \dot{\theta}_1; \dot{\theta}_2; \dot{\theta}_3]$. Initial state and desired state are given as below. These are based on human's jumping motion snapshots. In this simulation, we assumed that human jumps down to the ground from a high place. The posture when it touched ground is critical to an impact and generated angular velocity after landing. Every simulation runs for 0.4 seconds with fixed step-size 0.002s and solvers are auto. We simulated three controls as next.

Initial state : [2.356;-0.785;0.262;-1.047;0;0;0;0]
Desired angle: [1.920;-5.236; 1.047;-0.349]

A. Control without time-varying feedback

We first set the uncontrolled elements into torso which has the largest inertia moment and mass. It can prevent the singularity. We just add the desired angle of arm based on the human's jumping motion. We choose control gain such as c_d=48, k_d=496. So, we get the final angle and the results as Fig. 6.

TABLE I
ROBOT PARAMETERS IN SIMULATION

Part	Simulation robot parameter		
	Length(m)	Mass(kg)	Inertia CoM($10^{-5} kg * m^2$)
Torso	0.12	0.0425	5.10
Arm	0.14	0.0111	1.81
Upper	0.08	0.01955	1.19
Lower	0.08	0.0119	0.63

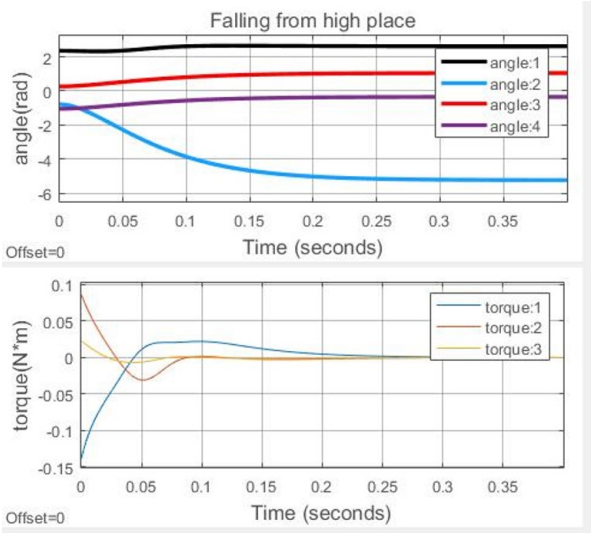

Fig. 6. The results of shape variables control without time-varying feedback (angle, torque)

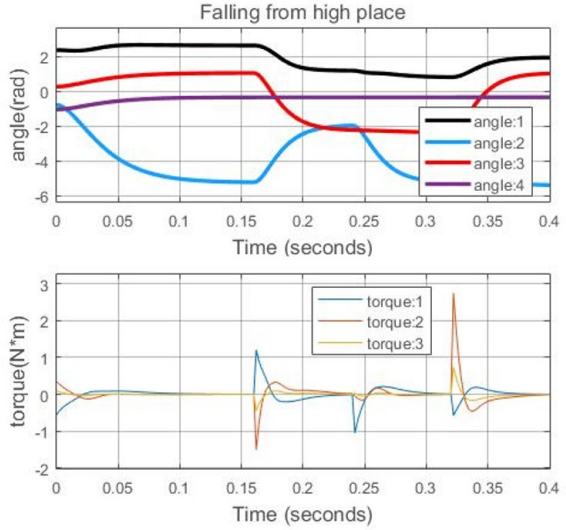

Fig. 7. The results of control based on a cycle path (angle, torque)

Final angle: [2.621;-5.233; 1.047;-0.349]

Based on human's jumping motion, the desired torso angle is determined as 1.920 rad. However, this control cannot make the torso angle reach to the desired value.

B. Control with time-varying feedback

1) Feedback law based on a cyclic path: In order to get the desired torso angle, we use time-varying feedback control based on a cyclic path. We set k=7104, c=168.6, k_d=2304, c_d=96, t_1=0.16, t_2=0.24, t_3=0.32, and t_4=0.40. Offline computation reveals that the suitable size of path is 3.3. We get the final angle and the results as Fig. 7.

Final angle: [1.926;-5.377; 1.010;-0.348]

Since the controls along the path are applied within short time intervals, the magnitude of torque inevitably increases a lot. Also, joint angles exceed the limits of joint angle and it was unreasonable.

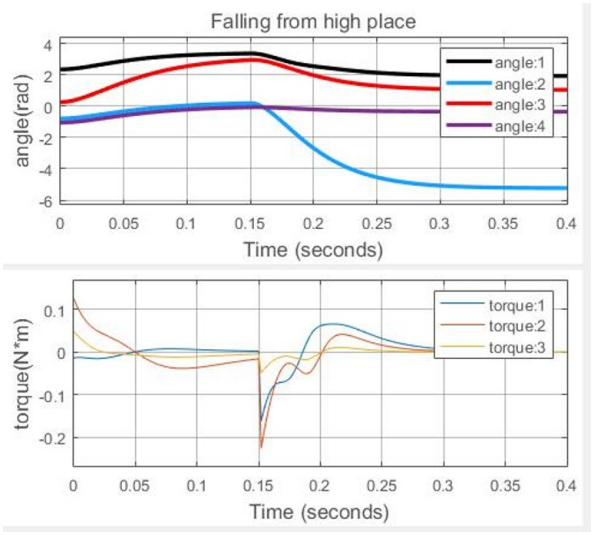

Fig. 8. The results of new time-varying control(angle, torque)

2) New feedback control with one displacement: So, we apply new time-varying feedback control as mentioned above. We set k=888, c= 60, c_d=72, k_d=1296, t_1=0.15, t_2=0.4. First, we start to find the range of a drift angle of q_0 by algorithm to find the reachable range.

Reachable range: [1.868, 2.672]

After we verify that the desired torso angle is included in the reachable range, we apply algorithm to find the required input. So, we can find the required displacement as next.

d= [0.26; 3.14; 0]

Using this displacement, we can get the final angle as below.

Final angle: [1.920;-5.236; 1.047;-0.349]

Not only shape variables are controlled to the desired shape, but the torso angle also reaches to the desired torso angle.

C. Comparative Anaylsis

We can see that first control without considerations for a torso angle yields about 0.7 rad shift from the desired angle. If a robot lands like this, it will fall down the ground. Cyclic feedback control makes a robot reach the desired torso angle, however, its motion looks so complicated and it breaks away from the limits of joint angle, as you can see in Fig. 7. On the other hand, new time-varying control makes a robot reach the desired angle for a stable landing. The motion from this control occurs in the possible range of joint angles. It also needs less torque than cyclic path control.

D. Landing Control with optimal gain

After an impact, we assume that the robot land on the ground with the desired angle. The angular velocity after the impact is obtained as [15.8; 16.6; 46.0;-48.2] through the impact model. Also, we can find out the angular velocity

Fig. 9. Snapshots of the 4-link robot under three controls (Left: without time-varying feedback, Middle: with time-varying feedback on cyclic path, Right: with time-varying feedback with a displacement) the starting moment is dark color and the end moment is light color

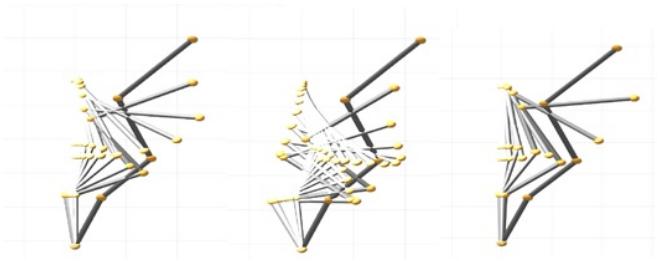

Fig. 10. Snapshots of the 4-link robot's landing under three controls (Left: c_d=42(optimal), Middle: c_d=21, Right: c_d=63)

after the impact depends much on the velocity of CoM, rather than the angular velocity before the impact.

$$\dot{q_s}^+ = \begin{pmatrix} -2.00 & -6.85 & 1.02 & -0.03 & -0.19 & -0.03 \\ -2.28 & -7.19 & -0.01 & 0.98 & -0.18 & -0.01 \\ -6.87 & -19.48 & -0.14 & -0.06 & 0.60 & 0.01 \\ 19.75 & 14.78 & 2.53 & -0.05 & -1.39 & -0.03 \end{pmatrix} \dot{q_f}^- \tag{26}$$

With this angular velocity, we find the optimal gain to minimize the used energy. Every simulation runs for 1.0 seconds with fixed step-size 0.005s and solvers are auto. As a result, we get optimal gain c_d=42, k_d=441 to minimize the energy use. It is not an actual optimal control, but we figure out that simply slow stabilization to absorb a shock does not minimize the energy use.

VII. CONCLUSIONS

In this paper, we proposed the posture control framework of a jumping articulated robot for a stable landing. We assumed a bipedal robot in a sagittal plane can be treated as a 4-link robot. Also, we considered various features of a jumping motion. We established Lagrange dynamics of a hybrid system which is combined with a flight phase and a stance phase. Flight phase was regarded as a Nonholonomic Caplygin system (under-actuated) and Stance phase was regarded as a fully-actuated system because the impact model assumes an inelastic impact and the end point acts as an ideal pivot. Based on this dynamics, we presented new time-varying control by taking joint angle limits, short duration of flight, and torque magnitude into account. It could help a robot get the desired shape variables and the desired torso angle within a possible angle range and with a small torque. Moreover, after an impact, optimal landing control based on gain tuning could yield natural motions to return to an upright pose without any trajectory. This new control framework in a hybrid system can help a bipedal robot to jump and land safely, similar to human beings. Some future researches include rigorous solutions for landing optimal control problem, finding optimal landing angle, consideration for some joint flexibilities, and implementation of the proposed control frame on a real 4-link robot system.

REFERENCES

[1] B. dynamics. (2016) Atlas, the world's most dynamic humanoid. [Online]. Available: https://www.bostondynamics.com/atlas

[2] C. Chevallereau, E. Westervelt, and J. Grizzle, "Asymptotically stable running for a five-link, four-actuator, planar bipedal robot," *The International Journal of Robotics Research*, vol. 24, no. 6, pp. 431–464, 2005.

[3] M. G. Pandy, F. E. Zajac, E. Sim, and W. S. Levine, "An optimal control model for maximum-height human jumping," *Journal of biomechanics*, vol. 23, no. 12, pp. 1185–1198, 1990.

[4] N. Shiraishi, Y. Kawaida, Y. Kitamura, S. Nakaura, and M. Sampei, "Vertical jumping control of an acrobat robot with consideration of input timing," in *in SICE, Proceedings of 2002 SICE Annual Conference on*, vol. 4. IEEE, 2002, pp. 2531–2536.

[5] R. Niiyama, A. Nagakubo, and Y. Kuniyoshi, "Mowgli: A bipedal jumping and landing robot with an artificial musculoskeletal system," in *in Robotics and Automation(ICRA), Proceedings of 2007 IEEE International Conference on*. IEEE, 2007, pp. 2546–2551.

[6] D. W. Haldane, J. K. Yim, and R. S. Fearing, "Repetitive extremeac-celeration (14-g) spatial jumping with salto-1p," in *Submitt. to IEEE Int. Conf. Intell. Robot. Syst*, 2017.

[7] X. Xin, T. Mita, and M. Kaneda, "The posture control of a 2-link free flying acrobot with initial angular momentum," in *Decision and Control, Proceedings of 2002 IEEE Conference on*, vol. 2. IEEE, 2002, pp. 2068–2073.

[8] T. Libby, A. M. Johnson, E. Chang-Siu, R. J. Full, and D. E. Koditschek, "Comparative design, scaling, and control of appendages for inertial reorientation," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1380–1398, 2016.

[9] J. Zhao, T. Zhao, N. Xi, M. W. Mutka, and L. Xiao, "Msu tailbot: Controlling aerial maneuver of a miniature-tailed jumping robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 6, pp. 2903–2914, 2015.

[10] G. C. Walsh and S. S. Sastry, "On reorienting linked rigid bodies using internal motions," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 1, pp. 139–146, 1995.

[11] R. Olfati-Saber, "Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[12] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. Wiley New York, 2006, vol. 3.

[13] H. Yang and D. Lee, "Dynamics and control of quadrotor with robotic manipulator," in *Robotics and Automation (ICRA), Proceedings of 2014 IEEE International Conference on*. IEEE, 2014, pp. 5544–5549.

[14] Y. Hurmuzlu and D. B. Marghitu, "Rigid body collisions of planar kinematic chains with multiple contact points," *The international journal of robotics research*, vol. 13, no. 1, pp. 82–92, 1994.

[15] J. Angeles and A. Kecskemethy, *Kinematics and dynamics of multi-body systems*. Springer, 2014, vol. 360.

[16] J.-M. Godhavn, A. Balluchi, L. Crawford, and S. Sastry, "Path planning for nonholonomic systems with drift," in *American Control Conference, 1997. Proceedings of the 1997*, vol. 1. IEEE, 1997, pp. 532–536.

[17] A. De Luca, R. Mattone, and G. Oriolo, "Steering a class of redundant mechanisms through end-effector generalized forces," *IEEE transactions on Robotics and Automation*, vol. 14, no. 2, pp. 329–335, 1998.