Real-time Stabilization of a Falling Humanoid Robot using Hand Contact: An Optimal Control Approach

Shihao Wang[1] and Kris Hauser[2]

***Abstract*— This paper introduces a real-time strategy to stabilize a falling humanoid robot by making hand contact with rails or walls in the environment. It uses an optimal control strategy with a simplified three-link model and finds an optimal hand contact using a direct shooting method. The objective function is designed to maintain stability while minimizing the probability of the contact slipping and minimizing impact that may damage the robot's arm. To achieve real-time performance, the method uses a precomputed database of necessary sticking friction coefficients at the contact points for all possible post-impact states. Validation is performed on a number of simulated falls in several rails and walls.**

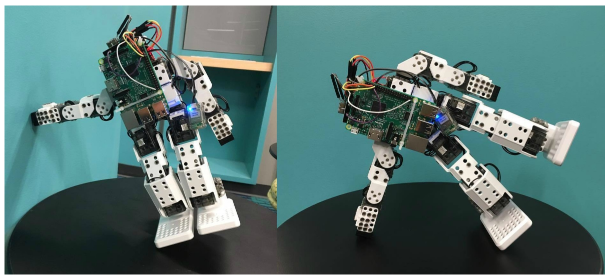


Fig. 1: Illustrating how the hand of a humanoid (ROBOTIS Mini [8]) can be used to stabilize itself on a wall and to reduce falling damage on flat ground.

I. INTRODUCTION

Falling is a common risk for humanoid robots due to the inherent instability of bipedal locomotion [1] and external factors such as uneven terrain and unexpected impulses. Because an unexpected fall can severely damage a robot, many strategies have been proposed to recover stability at the start of a fall [2], [3], or to minimize damage when a fall cannot be recovered [4], [5], [6], [7]. The most widely used technique is protective stepping, which hopes to arrest or stop the robot's falling with additional footsteps. However, stepping might not be feasible in the presence of environmental obstacles, such as tables and walls. It is also challenging to use when the swing foot must cross-over past the stance foot. Many human environments are designed with rails and handles that can be grasped with the hands to maintain stability. In clutter or when rails are available, the robot's hands could be used to prevent falling or to minimize the damage from falls (Fig. 1). However, it remains a major challenge for robots to fully exploit full-body contact with environmental geometry in real-time.

This paper presents an optimal control approach for exploiting hand contact with walls and other environmental obstacles to stabilize a falling robot. Our method assumes that the robot has already detected that it is falling toward obstacles, and has a map of the environment. It then optimizes a point of contact on the environment and a joint trajectory for one arm and hip to stabilize itself after the impact. The objective function tries to achieve a low impulse at the instant of contact as well as low tangential friction forces at the points of contact. The impulse term reduces the shock imparted to the robot structure, while the tangential friction terms reduce the necessary sticking friction coefficients needed to avoid slippage, since a secondary slipping event may be irrecoverable. The objective function measures overall probability of catastrophic failure, and can be tuned to incorporate knowledge of robot structure durability and environmental friction characteristics.

To make the computation tractable, we approximate the robot dynamics in the falling plane as a planar 3-link model. This model is computationally light but complex enough to allow for the use of pre-impact inertia shaping to reduce impact and the use of post-impact compliance to achieve closed loop stability. To optimize the proposed performance criteria, we use a shooting method to handle the pre-impact dynamics and choice of the contact point, and we use feedback linearization to analytically derive a stabilizing post-impact controller. Precomputation of required sticking friction coefficients helps our optimization run quickly. Experiments demonstrate that in a simulated environment, this algorithm stabilizes a falling robot for a wide range of environmental obstacles and initial conditions. It runs in real-time ($< 0.1s$) on a standard PC, and achieves lower risk of damage and secondary slipping compared to heuristic methods.

II. RELATED WORK

The problem of humanoid robot falling has been considered by several authors. Falling and push recovery strategies aim to balance the robot with the use of swing foot to reduce the falling speed, and can be employed in a unified framework with walking [2]. Besides falling in sagittal plane, push recovery strategies have also been studied for lateral plane falling [9] and omni-directional falling [10]. However, these techniques do not consider environmental constraints,

*This work was supported by NSF grant NRI #1527826

[1]Shihao Wang is with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708, USA shihao.wang@duke.edu

[2]Kris Hauser is with the Departments of Electrical and Computer Engineering and Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708, USA kris.hauser@duke.edu

and inside buildings or other crowded environments could run the robot into obstacles.

A first step to fall recovery is fall detection, and our method is designed to be triggered after a fall is predicted. Fall detection and fall direction prediction approaches include machine learning techniques [11], abnormality detection [12], [13] and zero moment point comparison [14].

Fall mitigation strategies reduce the impulse or damage at collision by altering how the robot falls. Fujiwara et. al. [4] allows the robot to land on the robot knees or arms. A similar trajectory planning approach is proposed in [6]. Ha and Liu [15] minimize the falling damage using multiple contact points and optimize a sequence of contacts using a variant of a 2-D inverted pendulum model. Goswami proposes a tripod contact strategy to reduce the robot's kinetic energy during the fall before the impact [7]. However, the issue with each of these techniques is a long computation time, reported to be over 1 s and in some cases 15 s [4], [6], [15], [7]. Our technique uses a simplified 3-link dynamics model and precomputed databases to achieve fast optimization.

Lee and Goswami [16] propose a momentum controller for a robot to fall on a targeted body segment, in their case a backpack assumed to be the impact absorber. In similar work, the same authors and colleagues propose general fall control methods that allow a robot to change the footstep locations to avoid a person or obstacle while falling [17], [18]. These techniques need less time than the fall mitigation strategies but still cannot satisfy the real-time computation requirement. Another practical impact absorber is to use an airbag [19]. Unlike this prior research, we aim to avoid falling altogether by making protective contact with the hand.

Heuristic approaches have been used in past work to achieve real-time performance. Tam and Kottee propose the use of walking sticks to prevent forward falling [20]. Their work uses forward fall detection and a heuristic quadrupedal posture to stop the robot from falling. In Yi et al, a machine learning approach is used to select between predefined push recovery controllers for a small humanoid robot, including use of the ankle, hip, or stepping [3]. A more comprehensive learning approach was taken in Kumar et. al., in which reinforcement learning is used to find a full body fall-recovery policy that can use protective stepping or hands, and this outperforms direct optimization by orders of magnitude [21]. However, all of these approaches work only on flat ground, and unlike our work the shape of the environment is not used by the falling controller.

III. Method

A. Summary

Our method makes the following assumptions:

1) A fall detector predicts the robot has begun to fall towards a wall or other environmental features.
2) The robot's starting configuration and velocities are known.
3) All dynamics are computed in the *falling plane*, which is the plane containing the center of mass velocity and the gravity vector.
4) The shape of the nearby environment in the falling plane is known.
5) Centers of masses and inertia matrices of the robot's links are known.
6) The hand makes contact on the environment at a single point of contact, and this point of contact must stick (rather than slide or bounce) for the robot to avoid further falling.

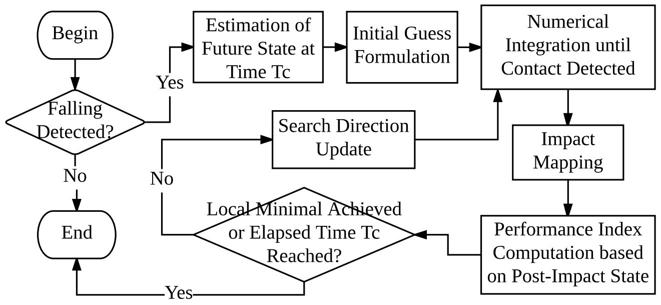

Begin
Falling Detected?
Yes
No
End
Estimation of Future State at Time Tc
Initial Guess Formulation
Numerical Integration until Contact Detected
Search Direction Update
Impact Mapping
No
Local Minimal Achieved or Elapsed Time Tc Reached?
Performance Index Computation based on Post-Impact State
Yes

Fig. 2: Flowchart of the proposed system.

With this information, the method computes a falling trajectory to minimize a combination of two objectives: impact at the hand and the necessary friction coefficients at the hand/foot points of contact. (We note that the definition of "hand" can mean any part of a given arm on the upper torso, so our work could also be applied to contact with the elbow.) Fig. 2 presents the flowchart of the proposed method.

However, due to the time used to compute the trajectory, once the optimizer has finished calculating the trajectory, the robot will not be at the same state as it was at the instant of falling. Instead, we give a fixed time budget T_c for computation and use the robot state predicted after duration T_c as the initial state for the optimal control sequence. Hence, once the optimization is complete the robot will be ready to execute the optimized controls. Using experiments we empirically determined that $T_c = 0.1$ s is a suitable budget.

We divide the motion into two phases: 1) *pre-impact*, where the robot is pivoting about an edge of its foot and its hand is brought to touch the environment, and 2) *post-impact*, where the hand is in contact with the environment and any residual velocity is dampened. Our algorithm is a direct shooting method to find a locally optimal controller that brings the robot hand into contact with the environment and then stabilizes the post-impact dynamics. The optimal contact point is determined via simulation in the inner loop of the optimization.

The performance index for the optimal trajectory estimates the probability that at least one of the following three *catastrophic events* occurs:

Event 1: The impact from collision damages the robot.
Event 2: The foot contact point slips.
Event 3: The hand contact point slips.

The first event occurs when the impact I exceeds the critical amount I_{crit} needed to damage the robot. The second and third events occur, respectively, when the *necessary sticking friction coefficient* at the hands and feet μ_A and μ_E of a hypothetical trajectory exceed the actual environmental friction coefficients μ_{foot} and μ_{hand}. The terms I, μ_A, and μ_E are assumed deterministic functions of a given trajectory, while I_{crit}, μ_{foot}, and μ_{hand} are unknown, normally distributed random variables.

The performance index is chosen to be

$$J(\boldsymbol{q}_{tr}, \dot{\boldsymbol{q}}_{tr}, \boldsymbol{u}_{tr}) = 1 - P(I \leq I_{crit})P(\mu_A \leq \mu_{foot})P(\mu_E \leq \mu_{hand}) \tag{1}$$

where $\boldsymbol{q}_{tr}$ is the configuration trajectory, $\dot{\boldsymbol{q}}_{tr}$ is its time derivative, and the control trajectory is $\boldsymbol{u}_{tr}$. The probability of each event is calculated using the CDF of a normal distribution with some given mean and standard deviation. Specifically, for each of the three values X, $P(y \leq X)$ is 1 minus the value of the CDF $cdf(y)$ of a normal distribution model $X \sim N(Avg(x), Std(x)^2)$ with mean $Avg(x)$ and standard deviation $Std(x)$. Here, $Avg(x)$ and $Std(x)$ can be coarsely estimated by a human engineer (lower values lead to a more conservative controller) or measured empirically.

B. Optimization Algorithm

Since the impact time t_I is not a fixed value, we use a shooting method that integrates the pre-impact dynamics forward with the chosen control sequence until impact is detected. Then an impact mapping is used to get the post-impact state from the pre-impact state. The magnitude of change of the linear momentum is the collision impulse I. For a post-impact trajectory, the necessary sticking friction coefficients μ_A and μ_E are then calculated.

In the shooting method, $\boldsymbol{u}_{tr}$ is taken as a piecewise constant control sequence of length N, with each segment lasting for dt. We determine the time step dt by first picking a value of T within which the robot will fall to the ground, and then set $dt = T/N$. In our experiments, $T = 0.2s$ and $N = 20$. A numerical optimizer with the quasi-Newton method is used to minimize the objective function over the control trajectory. To make this procedure run in real-time, our method uses 1) a simplified three-link inverted pendulum dynamics model, 2) an analytical solution of the post-impact stabilizing controller, 3) a precomputed database of necessary sticking friction coefficients at the contact points, which eliminates the need to integrate the post-impact trajectory.

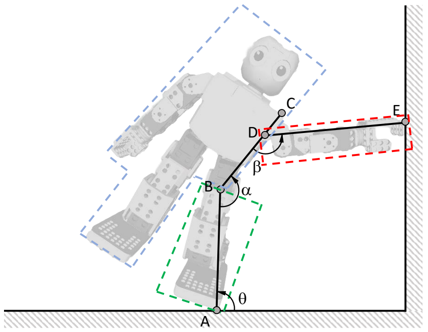


Fig. 3: Three-link model used in this work, illustrated in the post-impact state. The robot is divided into three blocks — stance leg, torso, and contact arm — each of which is modeled as a rigid link. The free arm and swing leg are assumed to be fixed to the torso.

We model the robot as a rigid, planar three-link inverted pendulum that captures only a few essential degrees of freedom in the falling plane. Joints with unessential degrees of freedom are locked to reduce the modeling error from robot to the pendulum. The pendulum rests on a single point contact at the foot, and after impact the hand will make a fixed contact with the wall (Fig. 3). The three DOFs of this model are denoted θ, α, and β, and correspond to the foot center of rotation (which is not assumed actuated), the hip, and the shoulder respectively. The one or two points of contact are assumed to be rigid, point contact, with a Coloumb friction model. The rigidity of contact is assumed so that we can compute the collision impulse and post-impact state using the assumption that the contact forces at the hand and foot are impulsive [22]. Hence, the configuration is $\boldsymbol{q} = [\theta, \alpha, \beta]^T$ and the control is $\boldsymbol{u} = [u_\alpha, u_\beta]$.

This model strikes a balance between simplicity and complexity. Benefits of its simplicity include:

1) Only two actuated links are assumed in the pre-impact dynamics, which reduces the dimensionality of trajectory optimization.
2) The post-impact system becomes a 1 DOF closed-chain system, which ensures a simple analytic expression for the post-impact controller. It also becomes easy to compute the friction forces after impact.
3) It is practical to precompute a 5-D database that saves the necessary sticking friction coefficients of the post-impact controller for all possible post-impact states. Applying the same technique to a 4 link-model would require a 7-D database, which is more time consuming and memory intensive.

It is also sufficiently complex to:

1) Match a wide range of initial conditions.
2) Allow the robot to use inertia-shaping to reduce collision impact (such as by lunging the hand outward).
3) Allow the robot to use compliance during post-impact stabilization, whereas a 2-link model would stop instantaneously.

C. Pre-impact Dynamics and Control Sequence Initialization

Pre-impact, the robot is an underactuated system with Lagrangian dynamics given by

$$D(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + G(\boldsymbol{q}) = B\boldsymbol{u} \tag{2}$$

where $D(\boldsymbol{q})$ is the generalized inertia matrix. $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is a 3×1 vector representing the centrifugal and coriolis forces. $G(\boldsymbol{q})$ is a 3×1 vector representing the generalized gravity. $\boldsymbol{u}$ is a 2×1 vector of joint torques at α and β, and B is a 3×2 full column rank matrix mapping the joint torques to generalized coordinates.

To seed the optimizer, we use a seed control $\boldsymbol{u}_s$ determined through an inverse dynamics approach to reach heuristic reference angles $(\alpha_{ref}, \beta_{ref})$ with the arm extended in the direction of falling. Eq. (2) can be rewritten as

$$\ddot{\boldsymbol{q}} = D^{-1}(B\boldsymbol{u} - C(\boldsymbol{q}, \dot{\boldsymbol{q}}) - G(\boldsymbol{q})) \tag{3}$$

so that the equations of motion for α and β can be fully controlled via a second order linear system with gains

$(k_{P\alpha}, k_{P\beta}, k_{D\alpha}, k_{D\beta})$ as follows:

$$\begin{aligned}\begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix} &= B^T D^{-1}(B\boldsymbol{u} - C(\boldsymbol{q},\dot{\boldsymbol{q}}) - G(\boldsymbol{q})) \\ &= \begin{bmatrix} -k_{P\alpha}(\alpha - \alpha_{ref}) - k_{D\alpha}\dot{\alpha} \\ -k_{P\beta}(\beta - \beta_{ref}) - k_{D\beta}\dot{\beta} \end{bmatrix}\end{aligned} \tag{4}$$

With the desired acceleration, the control $\boldsymbol{u}$ at each point in time is determined by solving this linear system, which has a solution because $B^T D^{-1} B$ is full rank. Then this $\boldsymbol{u}$ is plugged back in to (2) to integrate the pre-impact dynamics. The seed control sequence $\boldsymbol{u}_s$ is then determined by discretizing the pre-impact control trajectory at a given resolution.

D. Impact Mapping

With any given pre-impact control $\boldsymbol{u}_{tr}$, we perform integration of (2) and determine the instant of time at which the hand position $\boldsymbol{r}_E(\boldsymbol{q})$ makes contact with the environment. To make this fast, the environment is represented as an implicit function $\rho(\boldsymbol{x})$ which satisfies $\rho(\boldsymbol{x}) < 0$ in its interior and $\rho(\boldsymbol{x}) > 0$ in its exterior.

For the impact mapping, we use the generally used rigid impact model [22], [23] that assumes the duration of impact is infinitesimal. At this instant, the pre-impact and post-impact joint angles are the same, but the angular velocities change due to impulsive constraint forces at the two contact points. These are modeled using Dirac delta functions, so that at impact (2) becomes:

$$D_e(\boldsymbol{q_e})\ddot{\boldsymbol{q}}_{\boldsymbol{e}} + C_e(\boldsymbol{q_e},\dot{\boldsymbol{q}}_{\boldsymbol{e}}) + G_e(\boldsymbol{q_e}) = B_e\boldsymbol{u} + E_e(\boldsymbol{q_e})^T \boldsymbol{\delta}_{f_{imp}} \tag{5}$$

where $\boldsymbol{q_e} = [\boldsymbol{q}^T, r_{A_x}, r_{A_y}]^T$ is the extended generalized coordinate, r_{A_x} and r_{A_y} describe the horizontal and vertical position of the foot contact point respectively, D_e, C_e, G_e and B_e share the similar form with the corresponding matrix in (2) with dimensions augmented by two, $\boldsymbol{\delta}_{f_{imp}} = [\delta_{f_{Ax}}, \delta_{f_{Ay}}, \delta_{f_{Ex}}, \delta_{f_{Ey}}]^T$ representing the constraint forces at the two contact points at the impact time and $E_e(\boldsymbol{q_e})$ is the Jacobian matrix of the extended constraint equations $\phi_e(\boldsymbol{q_e})$ with respect to the extended generalized coordinate

$$\phi_e^T(\boldsymbol{q_e}) = \begin{bmatrix} \phi(\boldsymbol{q})^T & r_{Ax} - x_a & r_{Ay} - y_a \end{bmatrix} \tag{6}$$

where $\phi(\boldsymbol{q})$ is the hand contact constraint equation (10), and (x_a, y_a) give the fixed foot position.

Joint torques are not impulsive so the integration of (5) is

$$D_e(\boldsymbol{q_e})(\dot{\boldsymbol{q}}_{\boldsymbol{e}}^{+} - \dot{\boldsymbol{q}}_{\boldsymbol{e}}^{-}) = E_e(\boldsymbol{q_e})^T \boldsymbol{p}_{f_{imp}} \tag{7}$$

where $\boldsymbol{p}_{f_{imp}} = \int_{t_i^-}^{t_i^+} \boldsymbol{\delta}_{f_{imp}} d\tau$ denotes the magnitude vector of the impulse over the impact duration time. Another equation set is the extended holonomic constraint equations,

$$E_e(\boldsymbol{q_e})\dot{\boldsymbol{q}}_{\boldsymbol{e}}^{+} = 0 \tag{8}$$

Together with (7), the post-impact state and collision impulsive can be solved [24],

$$\begin{bmatrix} D_e(\boldsymbol{q_e}) & -E_e(\boldsymbol{q_e})^T \\ E_e(\boldsymbol{q_e}) & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{e}}^{+} \\ \boldsymbol{p}_{f_{imp}} \end{bmatrix} = \begin{bmatrix} D_e(\dot{\boldsymbol{q}}_{\boldsymbol{e}})\dot{\boldsymbol{q}}_{\boldsymbol{e}}^{-} \\ 0 \end{bmatrix} \tag{9}$$

The magnitude of $\boldsymbol{p}_{f_{imp}}$ is denoted as I for the performance index evaluation in (1).

E. Post-impact Dynamics and Stabilizing Controller

Fig. 3 illustrates the post-impact system. From this time, the contact point E will be assumed fixed on the support wall. Therefore, two holonomic constraint equations are added to the system dynamics:

$$\phi(\boldsymbol{q}) = \begin{bmatrix} r_{E_x}(\boldsymbol{q}) - x_c & r_{E_y}(\boldsymbol{q}) - y_c \end{bmatrix}^T = \boldsymbol{0} \tag{10}$$

where $\boldsymbol{r}_E = [r_{E_x}, r_{E_y}]^T$ are the coordinates of point E, and (x_c, y_c) are the coordinates where E first makes impact. Hence, the post-impact equations of motion are

$$D(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}}) + G(\boldsymbol{q}) = B' u_\beta + E(\boldsymbol{q})^T \boldsymbol{\lambda} \tag{11}$$

where $E(\boldsymbol{q})$ is the Jacobian of (10) with respect to $\boldsymbol{q}$ and $\boldsymbol{\lambda} \in \mathbb{R}^2$ are Lagrange multipliers.

By parameterizing the solution manifold of (10) by the shoulder angle β, we reduce the dynamics in (11) to be a linear function of the shoulder torque u_β:

$$\ddot{\beta} = f(\beta,\dot{\beta}) + g(\beta,\dot{\beta})u_\beta \tag{12}$$

where $g(\beta,\dot{\beta})$ is the linear term and $f(\beta,\dot{\beta})$ is the constant term. As long as $g(\beta,\dot{\beta}) \neq 0$ (verified via experiment), we can specify a stabilizing controller $\ddot{\beta} = -K\dot{\beta}$ where K is the derivative gain. The gain is chosen such that $\forall|\dot{\beta}| \in [\dot{\beta}_{min}, \dot{\beta}_{max}]$, $|K\dot{\beta}|$ remains within the joint acceleration bound $\ddot{\beta}_{max}$. Here the bounds of joint angular velocities and accelerations are determined based on the configuration of the motor on the robot. This controller also leads to an analytic expression of how the post-impact system evolves

$$\beta(t) = \beta^+ + \dot{\beta}^+(1 - e^{-Kt})/K \tag{13}$$

where β^+ is the post-impact angle and $\dot{\beta}^+$ is the post-impact angular velocity.

F. Necessary Sticking Friction Coefficient

Successful fall recovery requires both contact points to be fixed during the post-impact motion, which requires the friction force to be strictly less than the maximum friction that can be supplied by the environment. For a given trajectory, we can compute the minimum friction coefficients that would be required to keep the robot from slipping, which we call the *necessary sticking friction coefficient*. One challenge to resolve is that at each point in time the contact equations are indeterminate, since there are 4 force variables but only 3 constraints (linear momentum balance and angular momentum balance). In particular, the three-link mechanism can apply a continuum of internal forces via joint torques. We propose a method for analytically determining the internal forces to keep the necessary sticking friction coefficient at a minimum.

The forces at point A and point E (hand-wall contact) (Fig. 3) must respect linear momentum balance

$$\boldsymbol{F_A} + \boldsymbol{F_E} = \sum_{i=1}^{3} m_i \boldsymbol{a_i} - m\boldsymbol{g} \tag{14}$$

and angular momentum balance (taking point A to be the origin)

$$\boldsymbol{r}_E \times \boldsymbol{F_E} = \sum_{i=1}^{3} \boldsymbol{r}_{m_i} \times m_i(\boldsymbol{a}_i - \boldsymbol{g}) + \sum_{i=1}^{3} I_i \ddot{\boldsymbol{\Omega}}_i - \boldsymbol{u_\beta} \tag{15}$$

where $\boldsymbol{F_A} = [F_{Ax}, F_{Ay}]^T$ and $\boldsymbol{F_E} = [F_{Ex}, F_{Ey}]^T$ are the contact forces at point A and point E, m is the total mass of the robot, $\boldsymbol{g}$ is the gravity vector, and for link i, $\boldsymbol{r}_{m_i}$ is the position of its center of mass relative to A, $\boldsymbol{a}_i$ is the acceleration vector of its center of mass, I_i is its moment of inertia and $\ddot{\boldsymbol{\Omega}}_i$ is its angular acceleration vector, and $\boldsymbol{u_\beta} = [0, 0, u_\beta]^T$. We rearrange the above linear equation to the form

$$W\boldsymbol{F} = \boldsymbol{\omega} \tag{16}$$

where W is a 3×4 time-dependent matrix, $\boldsymbol{F}^T = [\boldsymbol{F_A^T}, \boldsymbol{F_E^T}]$, and $\boldsymbol{\omega}$ is a 3×1 vector stacking the right-hand sides of (14) and (15). The solution to (16) is a combination of external force (particular solution) and internal force (general solution) [25],

$$\boldsymbol{F}(t) = \boldsymbol{F}_p(t) + c\boldsymbol{F}_g \tag{17}$$

where c is arbitrary, $\boldsymbol{F}_p(t) = [F_{Ax_p}, F_{Ay_p}, F_{Ex_p}, F_{Ey_p}]^T$ is solved with QR factorization, and $\boldsymbol{F}_g$ is in the null space of W. In our problem the rank of W is 3 so the null space is 1-D. Specifically, $\boldsymbol{F}_g$ consists of two constant equal and opposite 2-D internal forces parallel to $\boldsymbol{r}_E$:

$$\boldsymbol{F}_g = \begin{bmatrix} \boldsymbol{F}_{Ag} \\ \boldsymbol{F}_{Eg} \end{bmatrix} \propto \begin{bmatrix} \boldsymbol{r}_E \\ -\boldsymbol{r}_E \end{bmatrix}. \tag{18}$$

where $\boldsymbol{F}_{Ag} = [F_{Ax_g}, F_{Ay_g}]^T$ and $\boldsymbol{F}_{Eg} = [F_{Ex_g}, F_{Ey_g}]^T$ are the internal force vectors at point A and point E. We normalize this vector so that c gives the magnitude of the internal force.

Letting now $\boldsymbol{F}(t, c)$ be a function both of time t and internal force magnitude c, the friction coefficients at point A and point E are respectively

$$\mu_A(t, c) = \left| \frac{F_{Ax}(t, c)}{F_{Ay}(t, c)} \right|, \mu_E(t, c) = \left| \frac{\boldsymbol{n}_E \times \boldsymbol{F}_E(t, c)}{\boldsymbol{n}_E^T \boldsymbol{F}_E(t, c)} \right| \tag{19}$$

where $\boldsymbol{n}_E$ is the surface normal vector at point E with its polar angle denoted as ψ. Now we choose an optimal value of c to minimize the sum of μ_A and μ_E. Hence, we choose

$$c^*(t) = \arg\min_c \mu_A(t, c) + \mu_E(t, c) \tag{20}$$

and hence the necessary sticking friction coefficient of the post-impact trajectory is

$$\begin{aligned} \mu(\beta^+, \dot{\beta}^+, r_{E_x}, r_{E_y}, \psi) &= \max_t \mu_A(t, c^*(t)) + \mu_E(t, c^*(t)) \\ &= \max_t (\min_c \mu_A(t, c) + \mu_E(t, c)). \end{aligned} \tag{21}$$

Appendix A shows how the optimization of (20) can be performed analytically, which makes (21) easier to evaluate.

G. Friction Database Pre-computation

For a given trajectory, the necessary sticking friction coefficients μ given by (21) could be calculated by integrating (11) over time and computing (20), but this would be inefficient. We accelerate optimization by precomputing a database over the five index variables $(\beta^+, \dot{\beta}^+, r_{E_x}, r_{E_y}, \psi)$. Given the robot kinematics and velocity limits, each variable has a bounded valid range, so we precompute the necessary sticking friction coefficients from all possible post-impact states and save them into a database. A single database is sufficient to cover any initial state of the robot and environment obstacles of any shape. However, a new database is needed if any of the kinematic parameters of the robot change (such as falling in the sagittal plane vs frontal plane).

The precomputation time and database size depend on the resolution of discretization. In our experiments, each of the five variables are discretized into at most 45 quantities. The calculation time is 70 min with OpenMP using four threads. The size for this database is around 120 MB, which easily fits in the RAM of a small embedded PC.

IV. Simulation Experiment

We evaluate the speed and quality of our strategy in a MATLAB planar simulation with 8 scenarios. All tests are on a 64-bit Intel Core i7 2.50GHz CPU with 8GB RAM. The optimal control trajectory is computed with the Dlib library's numerical optimization solver with the stopping strategy to be a comparison between a termination threshold $\varepsilon = 1e^{-7}$ and the change in the objective function from one iteration to the next iteration [26]. The robot parameters used are shown in Table I, and Table II shows the weight coefficients and heuristic values used for the simulation.

The eight scenarios illustrate a variety of initial conditions and environments. **Wall 1** and **2** are vertical walls at distance 0.09 m and 0.18 m, respectively. **Wall 3a** and **3b** are a cubic curve passing through (0.2,0.5), (0.07,-0.25) and (0.135,0.125) in which the derivative of this curve vanishes. **Table** and **Flat** are flat surfaces at height 0.1 m and 0.0 m. **Circle a** and **b** provide a floating circular support whose center is at (0.15 m,0.12 m) and 0.05 m radius. The initial conditions in each simulation is listed in Table III. In all cases except for **Table** and **Flat**, the control is initialized using the method described in Section III-C with heuristic

TABLE I: Model Parameters

	Mass (kg)	Length (m)	Moment of Inertia ($kg \cdot m^2$)
Leg	0.12	0.125	3.1677e-04
Torso	0.53	0.070	8.8277e-04
Arm	0.05	0.100	4.4271e-05

TABLE II: Misc. Coefficients and Heuristic Values

Controller Gains		Target Values		Failure Parameters	
k_{P_α}	250	α_{ref1}	$5\pi/6\,rad$	$Avg(I)$	$0.35\,N \cdot s$
k_{P_β}	250	β_{ref1}	$5\pi/8\,rad$	$Std(I)$	$0.1\,N \cdot s$
k_{D_α}	20	α_{ref2}	$\pi/2\,rad$	$Avg(\mu_A)$	0.5
k_{D_β}	20	β_{ref2}	$3.6\pi/5\,rad$	$Std(\mu_A)$	0.1
K	14			$Avd(\mu_E)$	0.4
				$Std(\mu_E)$	0.1

TABLE III: Simulation Initial Conditions

Problem	θ (rad)	α (rad)	β (rad)	$\dot{\theta}$ (rad/s)	$\dot{\alpha}$ (rad/s)	$\dot{\beta}$ (rad/s)
Wall 1	1.57	3.14	0	-2.0	-2.0	0
Wall 2	1.57	3.14	0	-2.0	-2.0	0
Wall 3a	1.57	3.14	0	-2.0	0.0	0
Wall 3b	1.41	3.14	0	-2.0	0.0	0
Table	1.41	3.14	0.78	-2.0	0.0	0
Flat	1.41	3.14	0.78	-2.0	0.0	0
Circle a	1.57	3.14	0	-2.0	0.0	0
Circle b	1.57	3.14	-0.62	-2.0	0.0	0

target values $\alpha_{ref} = \alpha_{ref1}$ and $\beta_{ref} = \beta_{ref1}$. In **Table** and **Flat**, we use $\alpha_{ref} = \alpha_{ref2}$ and $\beta_{ref} = \beta_{ref2}$ as given in Tab. II.

The optimization results are listed in Table IV. Timing is indicated in the Time column, showing that each example is optimized in under 0.1 s. The Impact, μ_A, and μ_E indicate the objective function components in the optimized trajectory. For comparison, the Heuristic Objective column gives the objective function value using the heuristic seed trajectory (raising the arm) as a fraction of the uncontrolled trajectory, and the Optimized Objective column gives the optimized value. These two columns show that although the heuristic is clearly better than uncontrolled falling, our optimization method can reduce the probability of failure often by several factors.

Traces of these simulations in the 3-link model are shown in Fig. 4 and Fig. 5. In each figure, the left column shows the uncontrolled motion and the right shows the optimized motion. In every case, the optimized velocity of the center of mass at pre-impact is lower than the uncontrolled velocity, showing arm extension and a backward movement of the hip. The post-impact motion then zeroes out the residual velocity. Comparing **Wall 1** to **2**, the robot reaches its hand forward to catch itself against the wall, and reaching is more extreme for further walls. Comparing **Wall 3a** and **3b**, the method chooses a different contact point depending on the initial velocity. For the **Flat** case, the optimized falling reduces the impact velocity with a bent-over pre-impact posture in which the robot's center of mass remains relatively high above the ground. This result is consistent with observations of optimal falling trajectories from prior work [7].

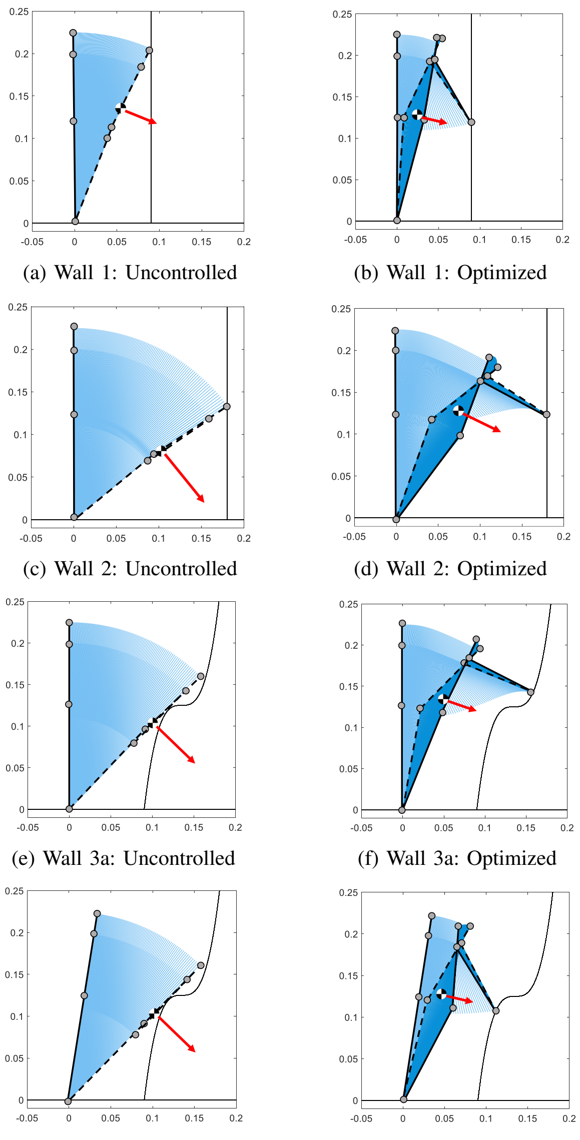
V. CONCLUSIONS

This paper presented a real-time strategy to stabilize a falling humanoid robot with making hand contact with walls or rails in the environment. Our optimal control approach is designed to maintain stability while minimizing the risk of damage to the robot and the risk of contacts slipping. The method works for environmental obstacles of different shapes ranging from vertical walls to flat ground. Simulation results suggest that optimization may yield a many-fold reduction in damage risk compared to heuristic techniques.

There are several directions to explore in future work. Our current method works only in a 2D plane, which requires separate precomputations for different falling directions. Future research will focus on the realization of 3-D stabilization with hand contact. Ongoing experiments are implementing this strategy on a small physical humanoid robot, and we are also investigating the effects of environmental measurement uncertainty on the controller's performance.

TABLE IV: Simulation results. Objective values are shown as a % of the uncontrolled value.

Problem	Time (ms)	Impact ($N \cdot s$)	μ_A	μ_E	Heuristic Objective (%)	Optimized Objective (%)
Wall 1	47	0.199	0.246	0.104	13.03	7.31
Wall 2	87	0.370	0.356	0.093	85.58	61.31
Wall 3a	74	0.282	0.347	0.072	66.47	29.67
Wall 3b	52	0.211	0.260	0.078	55.95	8.96
Table	53	0.170	0.086	0.110	26.36	3.82
Flat	75	0.326	0.232	0.242	99.95	44.75
Circle a	59	0.176	0.231	0.212	26.68	7.32
Circle b	71	0.175	0.193	0.117	25.47	4.33



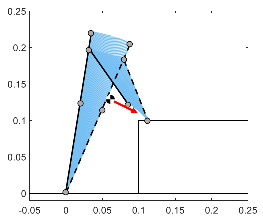
(a) Wall 1: Uncontrolled (b) Wall 1: Optimized (c) Wall 2: Uncontrolled (d) Wall 2: Optimized (e) Wall 3a: Uncontrolled (f) Wall 3a: Optimized (g) Wall 3b: Uncontrolled (h) Wall 3b: Optimized

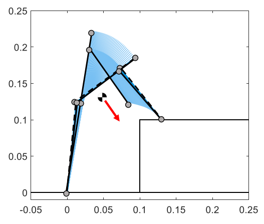
Fig. 4: Simulation of uncontrolled and optimal controllers for examples **Wall 1** to **Wall 3b**. For the controlled motion, we draw the initial state (solid), pre-impact state (dashed) and the steady state (solid). The light blue trace is the pre-impact motion and the dark blue trace is the post-impact motion. The red arrow denotes the velocity of the robot's center of mass at the pre-impact state (scaled by 0.1).

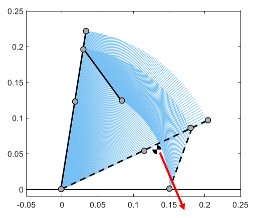
APPENDIX

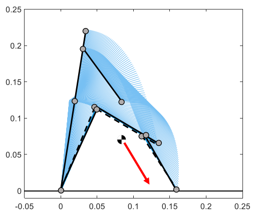
A. Analytic Solution for the Optimal Internal Force

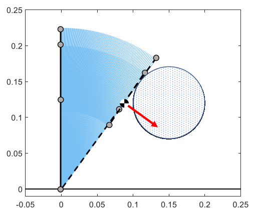
From (20), the optimal internal force c^* brings the sum of the friction coefficients to achieve the minimum. Denote the tangential component of the hand force at E as $F_E^{\parallel} = F_{E_p}^{\parallel} + cF_{E_g}^{\parallel}$ and the normal component as $F_E^{\perp} = F_{E_p}^{\perp} + cF_{E_g}^{\perp}$. Then the explicit expression for the sum of the friction coefficients

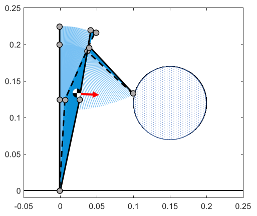

(a) Table: Uncontrolled

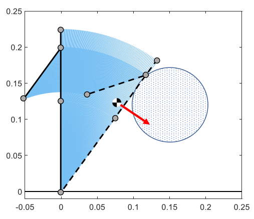

(b) Table: Optimized


(c) Flat: Uncontrolled


(d) Flat: Optimized


(e) Circle a: Uncontrolled


(f) Circle a: Optimized


(g) Circle b: Uncontrolled

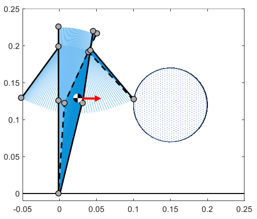

(h) Circle b: Optimized

Fig. 5: Simulation of uncontrolled and optimal controllers, examples **Table** to **Circle b**.

is

$$h(c) = \left| \frac{F_{Ax_p} + cF_{Ax_g}}{F_{Ay_p} + cF_{Ay_g}} \right| + \left| \frac{F_{E_p}^{\perp} + cF_{E_g}^{\perp}}{F_{E_p}^{\parallel} + cF_{E_g}^{\parallel}} \right| \tag{22}$$

To guarantee the positivity of the ground support force at point A, the value of c is bounded from below,

$$F_{Ay_p} + cF_{Ay_g} > 0 \Rightarrow c > -\frac{F_{Ay_p}}{F_{Ay_g}} \tag{23}$$

because F_{Ay_g} is always positive. Then the procedure to find the optimal internal force c^* is as follows:

1) Compute the critical points of function $h(c)$.
2) Compare the value of each candidate point with the lower bound and select those points that are feasible.
3) Set c^* to the candidate point that bring $h(c)$ to the minimal value among all other candidates.
4) If no feasible point is found in step 2), the minimum value of $h(c)$ lies at a boundary point, so c^* is set to infinity.

References

[1] T. McGeer, “Stability and control of two-dimensional biped walking,” in *Technical Report 1.*, Center for Systems Science, Simon Fraser University, Burnaby, B.C., Canada, 1988.

[2] J. Pratt, S. D. J. Carff, and A. Goswami, “Capture point: A step toward humanoid push recovery,” in *IEEE-RAS Int. Conf. Humanoid Robots*, Dec. 2006, pp. 200–207.

[3] S.-J. Yi, B.-T. Zhang, D. Hong, and D. D. Lee, “Online learning of low dimensional strategies for high-level push recovery in bipedal humanoid robots,” in *IEEE Int. Conf. Robotics and Automation*, Karlsruhe, Germany, 2013, pp. 1641–1647.

[4] K. Fujiwara, S. Kajita, K. Harada, K. Kaneko, M. Morisawa, and F. Kanehiro, “Towards an optimal falling motion for a humanoid robot,” in *IEEE-RAS Int. Conf. Humanoid Robots*, Genova, Italy, 2006.

[5] K. Fujiwara, S. Kajita, K. Harada, K. Kaneko, M. Morisawa, F. Kanehiro, S. Nakaoka, and H. Hirukawa, “An optimal planning of falling motions of a humanoid robot,” in *IEEE/RSJ Int. Conf. Intel. Robots and Systems*, San Diego, 2007, pp. 456–462.

[6] J. Wang, E. Whitman, and M. Stilman, “whole-body trajectory optimization for humanoid falling,” in *American Control Conf.*, 2012, pp. 4837–4842.

[7] Y. S and G. A, “Tripod fall: Concept and experiments of a novel approach to humanoid robot fall damage reduction,” in *IEEE Int. Conf. Robotics and Automation*, 2014.

[8] *ROBOTIS Mini*, ROBOTIS INC Std. [Online]. Available: http://www.robotis.us/robotis-mini-intl/

[9] M. Missura and S. Behnke, “Lateral capture steps for bipedal walking,” in *IEEE-RAS Int. Conf. Humanoid Robots*, Bled, Slovenia, 2011.

[10] ——, “Omnidirectional capture steps for bipedal walking,” in *IEEE-RAS Int. Conf. Humanoid Robots*, Atlanta, GA, USA, 2013.

[11] S. Kalyanakrishnan and A. Goswami, “Learning to predict humanoid fall,” *Int. J. Humanoid Robotics*, vol. 8, no. 2, pp. 245–273, Jun. 2011.

[12] Karssen, J.G.D., and M. Wisse, “Fall detection in walking robots by multi-way principal component analysis,” *Robotica*, vol. 27, no. 2, pp. 249–257, 2009.

[13] K. Ogata, K. Terada, and Y. Kuniyoshi, “Falling motion control for humanoid robots while walking,” in *IEEE-RAS Int. Conf. Humanoid Robots*, Pittsburgh,, 2007.

[14] ——, “Real-time selection and generation of fall damage reduction actions for humanoid robots,” in *IEEE-RAS Int. Conf. Humanoid Robots*, Daejeon, Korea, 2008, pp. 233–238.

[15] S. Ha and C. K. Liu, “Multiple contact planning for minimizing damage of humanoid falls,” in *IEEE/RSJ Int. Conf. Intel. Robots and Systems*, 2015.

[16] S. Lee and A. Goswami, “Fall on backpack: Damage minimizing humanoid fall on targeted body segment using momentum control,” in *ASME Int. Design Engineering Tech. Conf.*, 2011, pp. 47 153:1–10.

[17] a. A. G. S.-k. Yun and Y. Sakagami, “Safe fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping,” in *IEEE Int. Conf. Robotics and Automation*, 2009, pp. 781–787.

[18] A. Goswami, S. k. Yun, U. Nagarajan, S.-H. Lee, K. Yin, and S. Kalyanakrishnan, “Direction-changing fall control of humanoid robots: theory and experiments,” *Autonomous Robots*, vol. 36, no. 3, pp. 199–223, 2014.

[19] S. Kajita1, R. Cisneros1, M. Benallegue1, M. M. Takeshi Sakaguchi1, and Shinichiro Nakaoka1, K. Kaneko1, and F. Kanehiro, “Impact acceleration of falling humanoid robot with an airbag,” in *IEEE-RAS Int. Conf. Humanoid Robots*, Cancun, Mexico, 2016.

[20] B. Tam and N. Kottege, “Fall avoidance and recovery for bipedal robots using walking sticks,” in *Australasian Conf. Robotics and Automation*, Brisbane, Dec. 2016.

[21] V. C. Kumar, S. Ha, and C. K. Liu, “Learning a unified control policy for safe falling,” in *IEEE/RSJ IROS*, 2017.

[22] Y. Hurmuzlu and D. B. Marghitu, “rigid body collisions of planar kinematic chains with multiple contact points,” *Int. J. Robot. Res.*, vol. 13, no. 13, pp. 82–92, 1994.

[23] E. Westervelt, C. C. J. Grizzle, J. Choi, and B. Morris, Eds., *Feedback Control of Dynamic Bipedal Robot Locomotion*. Boca Raton: CRC Press, 2007.

[24] a. F. P. Ch. Glocker, “Dynamical systems with unilateral contacts,” *Nonlinear Dynamics*, vol. 3, no. 4, pp. 245–259, 1992.

[25] A. Bicchi and V. Kumar, “Robotic grasping and contact: a review,” in *IEEE Int. Conf. Robotics and Automation*, 2000.

[26] D. E. King, “Dlib-ml: A machine learning toolkit,” *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.