

Soft Landing for Planar Quadruped Robots*

*Code: https://github.com/Atom990/planar_quadruped_landing

Zixin Zhang
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA, USA
zixinz@andrew.cmu.edu

Fan Yang
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA
fanyang3@andrew.cmu.edu

Abstract—Many of legged animals in nature have excellent landing ability in the face of challenging terrain drop. They can effectively cushion the impact away through subtle movements, thus avoiding injury. If legged robots can have such ability, then their ability to cope with complex environments will be significantly enhanced. In recent years, researchers have used trajectory optimization methods to solve many problems related to the motion of legged robots. In this project, we introduce the soft landing problem: how to find a joint trajectory that makes sure the robot reaches the desired final state and minimizes the ground reaction forces of each foot during the touch-down phase of landing. We solve this using a hybrid trajectory optimization approach for a simplified planar quadruped robot in simulation. Some optimal trajectories with different initial conditions are shown.

I. INTRODUCTION

In nature, many animals with legs need to face the challenge of terrain drop. They often need to jump from very high places and make sure they can land softly and without hurting themselves. They have evolved an amazing ability to land very effectively, converting gravitational potential energy into kinetic energy during the landing process, and have evolved many biological structures that enable effective cushioning. Legged robots that move in the real world cannot avoid this challenge. If legged robots could have the ability to land softly, similar to animals in nature, these robots would be significantly more capable of coping with unexpected drops and minimizing damage in challenging environments, and they would also be able to reach more places in the world.

A. Related Work

Compared to the many advances in legged locomotion, there has been relatively little work exploring how to make robots cope with sudden terrain differences and drops. Some work has investigated how to allow the robot to adjust its torso posture during the falling, but has not focused on motion planning during the landing [1], [2].

Some people focus on the landing for hopper and biped robots, but the landing problem they discussed many differences from the process we are discussing. [3] designed a gait for the reduced biped robot that can achieve a soft landing of the feet during walking, i.e., the sole of the supporting foot is completely attached to the ground when landing, and the

angular velocity is 0. [4] introduced soft-landing trajectory for hopping robot which was planned to minimize impulse and maximum force from the ground. A stiffness controller formulation with only the desired body stiffness and damping values is used to realize human-like soft landing motion of an articulated hopping robot in [5]. These works only consider the landing of legs while walking, after jumping, or after dropping from a very low height (10cm). We have no idea the performance of these control methods when larger impacts happen.

Researchers from ETH demonstrated planar landing and airborne orientation control on the SpaceBok robot in a “low-gravity” environment (0.03 g) [6]. They tested their controller for three-dimensional cases in simulation, and for two-dimensional cases in the real world. However, we don’t know the performance of the proposed method in the real world and three-dimensional conditions with more dramatic impacts and inertial effects. [7] presented an optimization-based method to generate a landing trajectory for mini cheetah, and demonstrated 2m-high free drop in the real world. But about the quality of landing, such as the change of GRFs, there are few discussions in the paper.

B. Preliminary Comments

Consider a three-dimensional legged robot released from a given height above the ground. We assume that the robot free falls and its feet touch the ground in an unknown order. Here we define three phases (Fig. 1):

- 1) Aerial: The robot is completely in the air and is free falling.
- 2) Touch-down: This phase begins when the robot makes initial contact with the ground.
- 3) Landed: The robot reaches a “stable” configuration.

We will mainly focus on the touch-down phase, so we will consider “touch-down planning”, which is how to determine a joint trajectory from the start of the touch-down phase to the landed phase.

C. Problem Statement

In this course project, we reduce the problem to two-dimensional world. We consider the soft landing for planar “quadruped” robots. For a mini-cheetah quadruped robot in

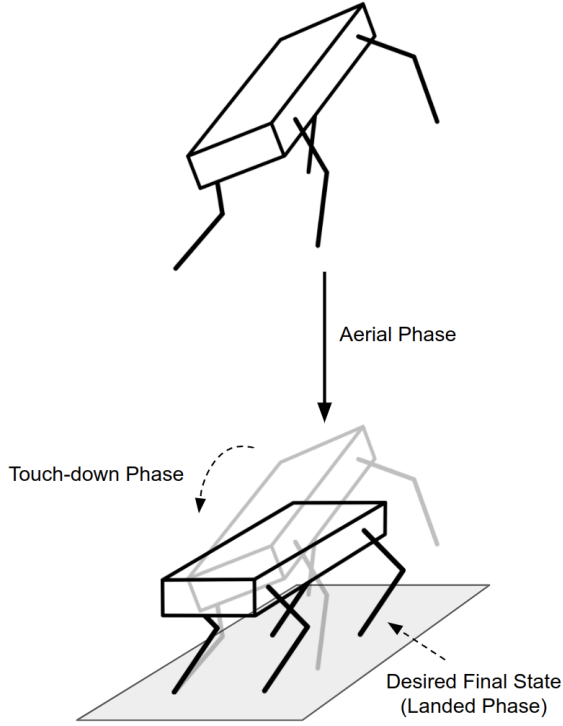


Fig. 1: Landing Phases

three-dimensional space, if we project it onto its side plane, we can get a planar five-link robot.

We place the planar quadruped robot at a certain height from the horizontal flat ground and let it fall freely. In the two-dimensional case, the robot has only one orientation angle. We assume that the initial orientation angle is between -90 and 90 degrees, i.e., the robot's back won't face the ground, and assume that the angular velocity is always 0 during the fall, i.e., the orientation angle remains constant during the fall. The soft landing problem we want to solve is about how to find a joint trajectory that makes sure the robot reaches the desired final state and minimizes the ground reaction forces of each foot during the touch-down phase.

After we simplify the problem, the number of possible contact modes becomes much less compared to the three-dimensional situation, which makes it possible to define the contact sequence. We use a hybrid system to describe the dynamics under different contact modes. In that case, we use hybrid trajectory optimization to solve this simplified problem.

II. APPROACH

A. Simplified Robot Dynamics

The model used in our experiments can be seen in Fig. 3. In the three-dimensional case, the quadruped robot has $4 + C_4^2 + C_4^3 + 1 = 15$ contact modes in the touch-down phase. So many contact modes make it almost impossible to artificially define the appropriate contact sequence. We could probably use the contact-implicit approach to solve it, but it would be

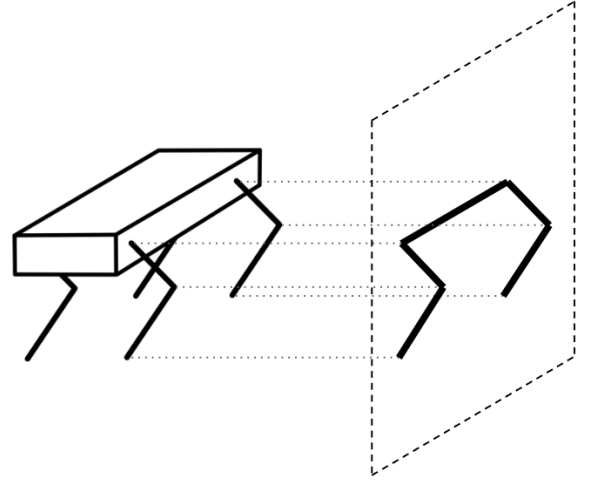


Fig. 2: Planar Quadruped Robot

an extraordinarily complex problem, well beyond the workload and time constraints of the course project.

Therefore, we simplified the problem. Today, several of the most well-known quadruped robots, such as mini-cheetah [8], Spot, and ANYmal [9], have a classical 12-joint configuration, i.e., each leg has a thigh and a calf, as well as three degrees of freedom. As shown in Fig. 2, if we let such a quadruped robot stand stationary, with the plane formed by the thigh and calf of each leg parallel to the side view plane, and all legs have the same joint angle, then its projection on the side view plane will be a planar five-link.

The full dynamics, that is the Lagrangian equations of motion of a five-link robot, are too complex, so it can significantly lengthen the solution time of our hybrid trajectory optimization problem. To avoid this problem, we use the centroidal dynamics model [10], as shown in Fig. 3. This model is widely used for legged robots, especially when the legs and feet are very light compared to the torso because the dynamics of the legs and feet won't affect the motion of the torso much. It can significantly reduce the nonlinearity in the problem and greatly reduce the computation.

A slight difference from the common centroidal dynamics is that we also consider the masses of the feet here. The reason for this is that we need to set the positions of the feet to the decision variables in order to facilitate writing the equation constraints about the contact modes in the optimization problem. In addition to this, we can use inverse kinematics to calculate the angle of each joint based on the torso position and the feet positions, which in turn enables the control of the robot.

For external forces, we only consider gravity and the ground reaction force. The ground reaction force includes the normal force and the friction. In our model, we assume that our robot will have an inelastic collision with the ground, which means the robot will never bounce up after touching the ground. We also assume that the foot that touches the ground first will never move, that is the friction coefficient of the ground is

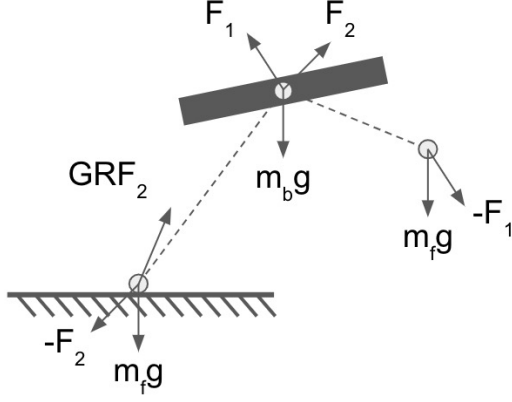


Fig. 3: Simplified robot dynamics. m_b and m_f are the mass of the torso and foot, respectively. $g = -9.81\text{m/s}$. The force between the i th foot and the robot torso is denoted as F_i , and it is usually the force that connects the robot feet and the torso and prevent the robot from falling apart. GRF_i is the ground reaction force on the i th foot. For the feet on the ground, $F_i = GRF_i + m_f g$ because the feet should be stationary.

infinity.

Specifically, the torso of the robot is reduced to a point mass of m_b located in the torso's center of mass. The foot is then considered as a point mass of m_f , and $m_f \ll m_b$. The force between the i th foot and the robot torso is denoted as F_i . F_i is usually the force that connects the robot feet and the torso and prevent the robot from falling apart. Specially, when the robot's feet are on the ground, the feet should be stationary because we assume no bouncing and sliding, so $F_i = GRF_i + m_f g$, where GRF_i is the ground reaction force on the i th foot.

In summary, we have the following simplifications for the robot:

- 1) We only consider planar dynamics (two-dimensional world).
- 2) We only consider the masses of the torso and feet, and use centrodial dynamics model.

B. State Space Model

We formalize the previous model into mathematical equations. The state includes the position of the torso $p_b \in \mathbb{R}^2$, the orientation of the torso $\theta \in \mathbb{R}^1$, the position of the feet $p_1 \in \mathbb{R}^2$ and $p_2 \in \mathbb{R}^2$, the linear velocity of the torso $v_b \in \mathbb{R}^2$, the angular velocity of the torso $\omega \in \mathbb{R}^1$, and the linear velocities of the feet $v_1 \in \mathbb{R}^2$, $v_2 \in \mathbb{R}^2$ and time $t \in \mathbb{R}^1$. Thus, the state

is defined as:

$$\mathbf{x} = \begin{bmatrix} p_b \\ \theta \\ p_1 \\ p_2 \\ v_b \\ \omega \\ v_1 \\ v_2 \\ t \end{bmatrix} \quad (1)$$

The control input includes the force of each foot on the torso $F_i \in \mathbb{R}^2$ and time step $\Delta t \in \mathbb{R}^1$, so it is defined as:

$$\mathbf{u} = \begin{bmatrix} F_1 \\ F_2 \\ \Delta t \end{bmatrix} \quad (2)$$

We add time t into the state vector $\mathbf{x}$ and time step Δt into the control input vector $\mathbf{u}$ for hybrid trajectory optimization purpose, which will be described in Sec. II-C.

The dynamics model is defined as

$$\dot{\mathbf{x}} = \begin{bmatrix} v_b \\ \omega \\ v_1 \\ v_2 \\ \frac{1}{m_b}(F_{1x} + F_{2x}) \\ \frac{1}{m_b}(F_{1y} + F_{2y} + m_b g) \\ \frac{\tau_F}{I_b} \\ -\frac{F_{1x}}{m_f} + g \\ -\frac{F_{1y}}{m_f} + g \\ -\frac{F_{2x}}{m_f} + g \\ -\frac{F_{2y}}{m_f} + g \\ \Delta t \end{bmatrix} \quad (3)$$

where m_f is the mass of the foot, m_b is the mass of the robot torso, $I_b = \frac{m_b L^2}{12}$ is the inertia of the robot torso with length L , and τ_F is the torque generated by F_1 and F_2 on the torso's center of mass:

$$\tau_F = -F_{1x}p_{1y} + F_{1y}p_{1x} - F_{2x}p_{2y} + F_{2y}p_{2x} \quad (4)$$

F_{ix} , F_{iy} , p_{ix} and p_{iy} represent the components of F_i and p_i in the x and y directions, respectively.

C. Hybrid Trajectory Optimization

Though there would be multiple contacts during the falling, it is still straightforward to define the contact modes needed to model the falling procedure. Thus, we can define a feasible contact sequence easily and model our problem as a hybrid trajectory optimization problem.

In our problem, we only consider touch-down planning, so we can define only three contact modes:

- 1) Mode 1: The front foot (foot 1) touches the ground while the rear foot (foot 2) is in the air.
- 2) Mode 2: The rear foot (foot 2) touches the ground while the front foot (foot 1) is in the air.
- 3) Mode 3: Both feet touch the ground.

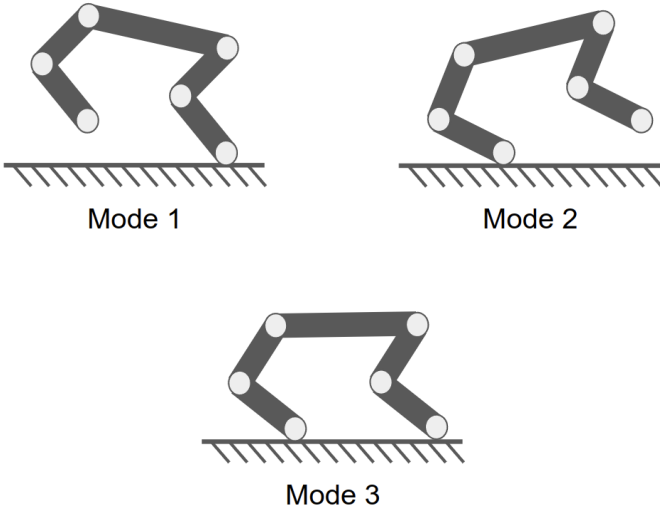


Fig. 4: Three different contact mode

In our problem, we assume that the initial orientation angle of the torso θ is a non-zero value, and there is no bouncing and sliding, so we only consider the mode transition of 1 to 3 and 2 to 3. The contact modes are shown in Fig. 4.

The objective is defined in a classical LQR style:

$$\begin{aligned} \min_{w, b, \xi} \quad & \frac{1}{2} (x_N - \bar{x}_N)^T Q_N (x_N - \bar{x}_N) + \\ & \frac{1}{2} \sum_{k=1}^{N-1} \Delta t_k (x_k - \bar{x}_k)^T Q (x_k - \bar{x}_k) \\ & + (u_k - \bar{u}_k)^T R (u_k - \bar{u}_k) \end{aligned} \quad (5)$$

The constraints are defined as follows:

$$x_1 - x_{init} = 0, \quad (6)$$

$$x_N - x_{goal} = 0, \quad (7)$$

$$f_1(x_k, u_k) - x_{k+1} = 0, \forall k \in M_1 \setminus J_1, \quad (8)$$

$$f_2(x_k, u_k) - x_{k+1} = 0, \forall k \in M_2 \setminus J_2, \quad (9)$$

$$f_3(x_k, u_k) - x_{k+1} = 0, \forall k \in M_3, \quad (10)$$

$$jump_3[f_i(x_k, u_k)] - x_{k+1} = 0, \forall k \in J_1, J_2, \quad (11)$$

$$y_{1,k} = 0, \forall k \in M_1, M_3, \quad (12)$$

$$y_{2,k} = 0, \forall k \in M_2, M_3, \quad (13)$$

$$d_{min} \leq \|p_{b,k} - p_{1,k}\| \leq d_{max}, \forall k \quad (14)$$

$$d_{min} \leq \|p_{b,k} - p_{2,k}\| \leq d_{max}, \forall k \quad (15)$$

$$0 \leq y_{b,k} - \|l_b \sin \theta\|, \forall k \quad (16)$$

M_i is the set of indices of knot points in contact mode i . The jump map sets J_1 and J_2 are the indices where the mode of the next time step is different than the current, i.e. $J_i = \{k+1 \notin M_i | k \in M_i\}$. Constraints 6 and 7 denotes the initial and desired final state should match the desired state. Constraints 8, 9, and 10 are dynamics constraints of different contact modes. Constraint 11 is the dynamics constraint of the jump map. Constraints 12 and 13 make sure that we

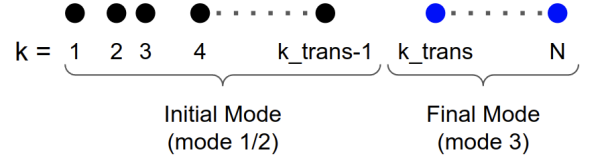


Fig. 5: Contact Sequence

are having the inelastic contact, which means the feet on the ground should always stay on the ground. Constraints 14 and 15 are kinematic constraints, which makes sure that the feet are within a reasonable range of the robot torso. Constraint 16 makes sure that the robot torso will not touch and penetrate the ground.

For hybrid trajectory optimization, it is necessary to define a feasible contact sequence. As we stated before, we assume that the orientation angle θ doesn't change during falling, and there is only one contact transition occurs in the touch-down phase, so we only need to define the index k_{trans} of the knot point where mode 3 starts. Fig. 5 shows how we define a contact sequence. N is the total number of knot points in the trajectory, and it is fixed during solving.

Assume we have actuator at each joints, that is there are 4 actuators. The degrees of freedom of the robot is $4 + 3 = 7$, because the robot torso has additional three degrees of freedom about position and orientation. So this system is underactuated. If we define a very strict contact schedule with a constant time step, the robot's feet may not be able to touch the ground at the moment we want due to the underactuated nature of the system, and the optimization problem would be infeasible. Thus, an important trick we use here is adding time t into our state, and time step Δt into our control input, so we scale the stage cost by Δt_k , as shown in Eq. 5. In this way, the solver can adjust the mode transition time to make the problem feasible without changing anything else.

In the reference state trajectory $\bar{x}_k$, we simply set all the states to be our desired final state. For the reference control input $\bar{u}_k$, since we define the contact schedule, so the contact mode of each knot point is known. When there is only one leg on the ground, we set the force on that foot to be $m_b g$, and when there are two legs on the ground, we set the force on each foot to be half of $m_b g$.

D. Implementation

We have implemented the whole approach using Julia. We choose COIN-OR Interior Point Optimizer (IPOPT) [11] with MathOptInterface (MOI) [12] to solve the hybrid trajectory optimization problem. MOI is an abstraction layer designed to provide a unified interface to mathematical optimization solvers so that users do not need to understand multiple solver-specific APIs. IPOPT is an open source software package for large-scale nonlinear optimization. It can be used to solve

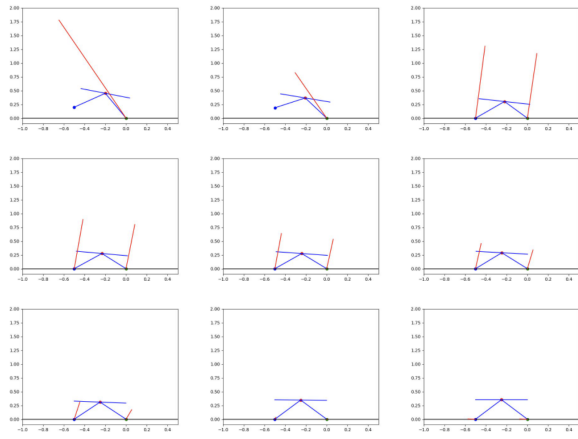


Fig. 6: The landing process when the torso mass is 10 kg and the touchdown velocity is 3 m/s. The red line shows the GRF of each foot.

general nonlinear programming problems of the form:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g^L \leq g(x) \leq g^U \\ & x^L \leq x \leq x^U \end{aligned} \quad (17)$$

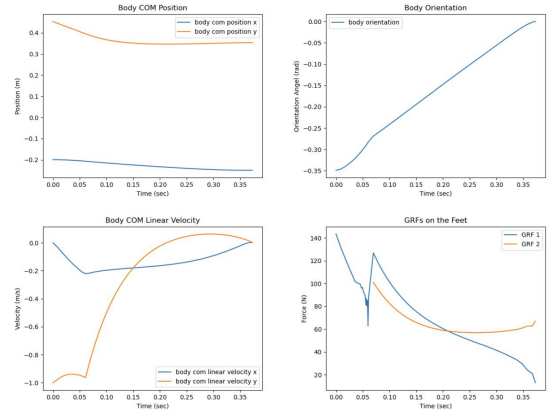
The files in the code framework are:

- 1) *constraints.jl* includes implementation of constraints and the Jacobians of constraints.
- 2) *costs.jl* calculates the objective and the gradient of the objective.
- 3) *moi.jl* implements solving using MOI.
- 4) *nlp.jl* defines a HybridNLP struct used for solving by MOI.
- 5) *planar_quadruped.jl* defines the robot model, calculates dynamics and the Jacobians of dynamics.
- 6) *quadratic_cost.jl* defines a QuadraticCost struct and some function about calculating the objective.
- 7) *ref_traj.jl* generates the reference trajectory.
- 8) *main.ipynb* is the main file we run.

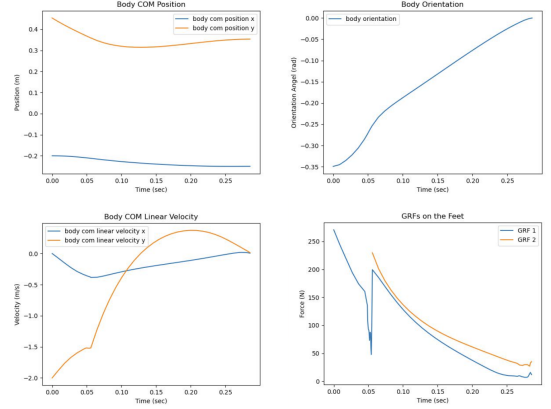
In addition, two Python files are used to generate figures and videos from the .csv file we store from IPOPT.

III. RESULTS

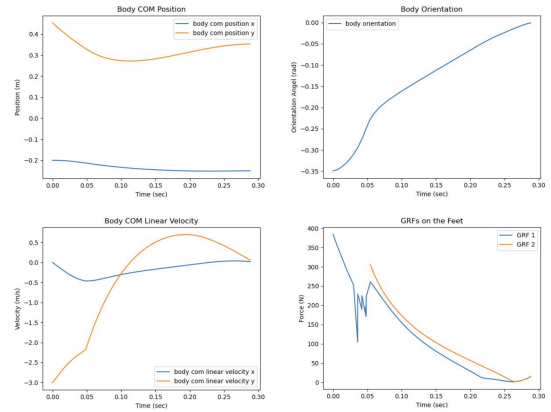
The parameters of the robot model are: torso mass $m_b = 10.0$ kg, foot mass $m_f = 0.1$ kg, torso length $L = 0.5$ m. The lower and upper bound of Δt is set to be 0.0001s and 0.006s. The total number of knot points in the trajectory is 61, and



(a) Initial Touch-down Velocity = 1 m/s



(b) Initial Touch-down Velocity = 2 m/s



(c) Initial Touch-down Velocity = 3 m/s

Fig. 7: The change of some physical quantities during the landing process when the torso mass is 10 kg, including the position and orientation of the torso, the linear velocity of the torso, and the magnitude of GRFs on the feet.

$$R = \textit{Diagonal} \begin{bmatrix} 10.0 \\ 10.0 \\ 1.0 \\ 10.0 \\ 10.0 \\ 10.0 \\ 10.0 \\ 10.0 \\ 10.0 \\ 10.0 \\ 1.0 \\ 10.0 \\ 10.0 \\ 10.0 \\ 10.0 \\ 0.0 \end{bmatrix} \quad (18)$$

When the initial velocity is small, the GRF does not decrease to 0. But when the initial velocity is large, the GRF decreases to close to 0. This is because when the initial velocity is large, the linear velocity of the robot will change from downward to upward, so a reasonable state is that the GRF decreases to close to 0 to minimize the objective function, and then the robot decelerates by its own gravity. Fig. 6 clearly illustrates this process.

The successful landing with an initial velocity of 3 m/s is the best result we have been able to obtain so far. However, this is not a very large initial velocity and corresponds to a free fall from a height of about 45 cm. Therefore, in future work, we can try to improve the performance of the approach by further tuning the parameters. In the meantime, we will try to

- [1] J. T. Bingham, J. Lee, R. N. Haksar, J. Ueda and C. K. Liu, "Orienting in mid-air through configuration changes to achieve a rolling landing for reducing impact after a fall," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 3610-3617, doi: 10.1109/IROS.2014.6943068.
- [2] V. Kurtz, H. Li, P. M. Wensing, and H. Li, "Mini cheetah, the falling cat: A case study in machine learning and trajectory optimization for robot acrobatics", 2021. arXiv: 2109.04424 [cs.RO].
- [3] F. Takemori and A. Kitamura, "Soft landing approach for a reduced biped robot model," Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003), 2003, pp. 753-758 vol.2, doi: 10.1109/AIM.2003.1225437.
- [4] E. Ohashi and K. Ohnishi, "Variable compliance control based on soft-landing trajectory for hopping robot," 30th Annual Conference of IEEE Industrial Electronics Society, 2004. IECON 2004, 2004, pp. 117-122 Vol. 1, doi: 10.1109/IECON.2004.1433294.
- [5] S. Sung, Y. Youm, "Landing Motion Control of Articulated Hopping Robot," International Journal of Advanced Robotic Systems, vol. 4, no. 3, pp. 303-312, Sep. 2007, doi: 10.5772/5686.
- [6] N. Rudin, H. Kolvenbach, V. Tsounis and M. Hutter, "Cat-Like Jumping and Landing of Legged Robots in Low Gravity Using Deep Reinforcement Learning," in IEEE Transactions on Robotics, vol. 38, no. 1, pp. 317-328, Feb. 2022, doi: 10.1109/TRO.2021.3084374.
- [7] S. H. Jeon, S. Kim, and D. Kim, "Real-time Optimal Landing Control of the MIT Mini Cheetah," 2021, arXiv:2110.02799 [cs.RO].
- [8] B. Katz, J. D. Carlo and S. Kim, "Mini Cheetah: A Platform for Pushing the Limits of Dynamic Quadruped Control," 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 6295-6301, doi: 10.1109/ICRA.2019.8793865.
- [9] M. Hutter et al., "ANYmal - a highly mobile and dynamic quadrupedal robot," 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 38-44, doi: 10.1109/IROS.2016.7758092.
- [10] H. Dai, A. Valenzuela and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," 2014 IEEE-RAS International Conference on Humanoid Robots, 2014, pp. 295-302, doi: 10.1109/HUMANOIDS.2014.7041375.
- [11] A. Wächter, L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," in Mathematical programming, vol. 106, pp. 25-57, Mar. 2006, doi: 10.1007/s10107-004-0559-y.
- [12] B. Legat, O. Dowson, J. D. Garcia, M. Lubin, "MathOptInterface: A Data Structure for Mathematical Optimization Problems. INFORMS Journal on Computing," in INFORMS Journal on Computing, vol. 34, No. 2, pp. 672-689, Mar-April 2022, doi: 10.1287/ijoc.2021.1067.