Control of Dynamic Parkour Motions for a Hopping Leg

Maximilian Albracht[1,2]
[1] FH Aachen, [2] Robotics Innovation Center, DFKI GmbH

Motivation

As legged locomotion in nature is very dynamic and rich, this elegant art of movement should also be exploited by robots to enable efficient and unrestricted locomotion even on difficult terrain. As parkour places a strong emphasis on adaptability, creativity, and functional motion, the incorporation of techniques from this discipline into the field of robotics holds substantial potential as well.

Hopping Leg on a Broomstick

The system encompasses four rotational degrees of freedom, with two being actively controllable (Hip and Knee) and two remaining passive (Pitch and Yaw). Quasi direct drives (qdd100 from mjbots [1]) are utilized as actuators, enabling high torque to weight ratio and backdriveability.

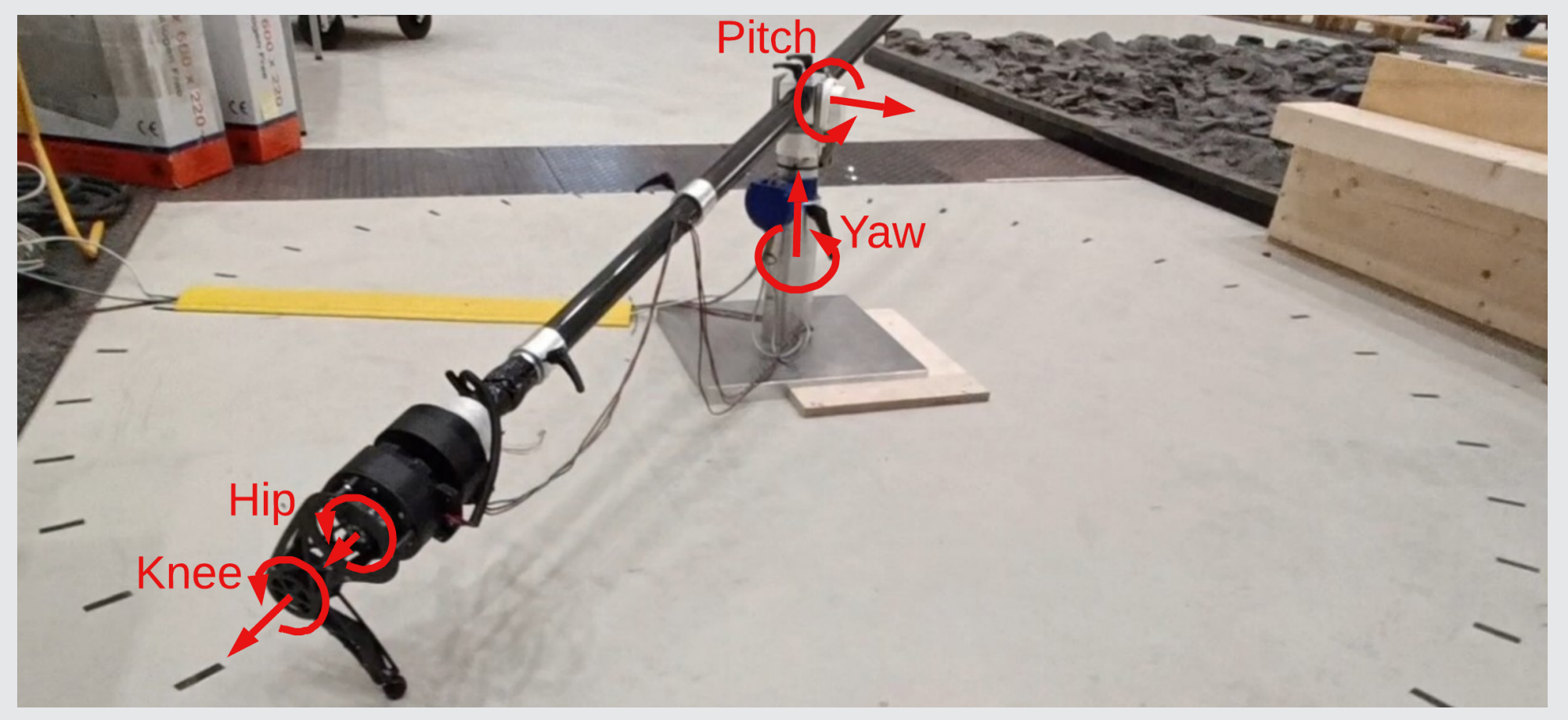

Figure: Degrees of Freedom

Workspace Idealization

The workspace of the system is approximately a thin-walled sphere. However, with joint limits and a very long rod, the workspace is roughly equivalent to a cylindrical surface. This workspace idealization is highly beneficial:

- sufficiently accurate
- reduces computational demands
- clearer visualization

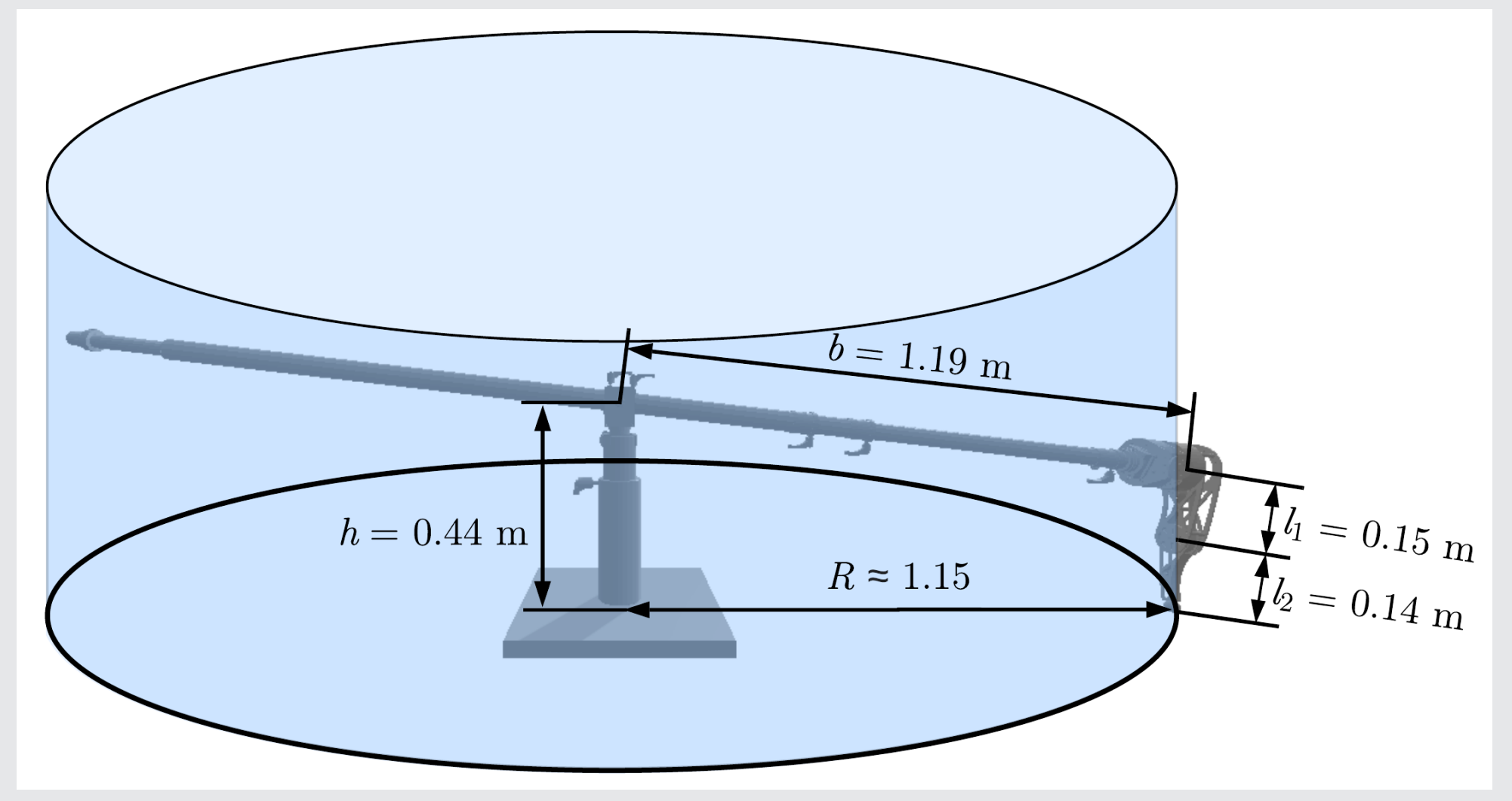

Figure: Idealized Workspace

Energy Based Hopping Control

In flight phase, the legs center of mass is bounded to a ballistic trajectory. The governing equations are considered with the position difference Δx, Δz and the launch angle θ as known variables.

$$x = x_0 + vt\cos\theta$$
$$z = z_0 + vt\sin\theta - \frac{1}{2}gt^2$$

From this, the launch velocity v is calculated.

$$v = \frac{\Delta x}{\sqrt{2\frac{\Delta x\tan\theta - \Delta z}{g}}\cos\theta}$$

The GRF (ground reaction force) λ to be applied over exertion distance d is computed by comparison of the kinetic energy and the work on the hopping legs center of mass.

$$\frac{1}{2}mv^2 = \lambda d \quad \Rightarrow \quad \lambda = \frac{mv^2}{2d}$$

By utilizing the Jacobian matrix $\mathbf{J}$, a mapping between the GRF vector $\boldsymbol{\lambda}$ and motor torques $\boldsymbol{\tau}$ is established.

$$\boldsymbol{\tau} = \mathbf{J}^\intercal \boldsymbol{\lambda}$$

Due to the Jacobian's dependence on joint angles, it needs to be recalculated every time a joint angle of the leg changes.

Optimal Motion Planning

Depending on the desired number of jumps N, the following jump parameters are set as continuous decision variables:

- $\mathbf{t}[N] \equiv$ time per jump
- $\mathbf{v}[N] \equiv$ take-off velocity
- $\theta[N] \equiv$ take-off angle

With those, the landing point $x[n]$ is defined for each jump. A parkour with K obstacles is modeled as a non-continuous function. Thus, motion planning transforms into a mixed-integer optimization problem, necessitating the introduction of binary decision variables:

- $\delta_\mathbf{a}[N, K] \equiv$ obstacle front passed
- $\delta_\mathbf{b}[N, K] \equiv$ obstacle back passed

The idea is to have the variable equal to zero if the landing point is before, and equal to one if the landing point is behind the corresponding obstacle border $A[k]$, $B[k]$.

Optimal Motion Planning (ctnd.)

This is enforced by the following constraints:

$$\delta_\mathbf{a}[n,k](A[k] - \mathbf{x}[n]) \le 0$$
$$(1 - \delta_\mathbf{a}[n,k])(\mathbf{x}[n] - A[k]) \le 0$$
$$\delta_\mathbf{b}[n,k](B[k] - \mathbf{x}[n]) \le 0$$
$$(1 - \delta_\mathbf{b}[n,k])(\mathbf{x}[n] - B[k]) \le 0$$

With this logic, the height of the parkour is mapped and imposed as constraint on the landing height $z[n]$:

$$\sum_{k=1}^{K} H[k](\delta_\mathbf{a}[n,k] - \delta_\mathbf{b}[n,k]) = \mathbf{z}[n]$$

The constraints above are set for $\forall n \in \mathbb{N} \subset [1, N]$ and $\forall k \in \mathbb{N} \subset [1, K]$. Also modeled are margins M_h, M_v and restricted areas in which landing is prohibited.

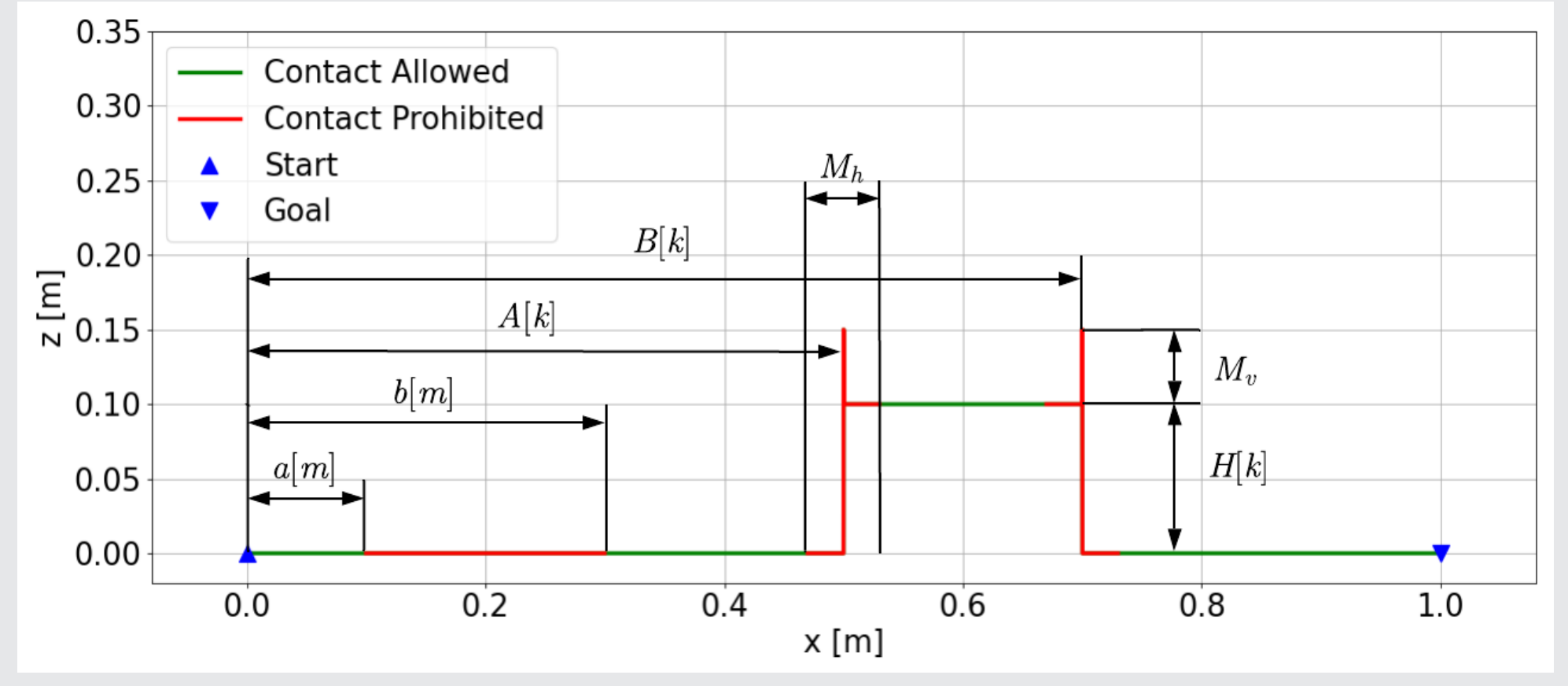

Figure: Parkour Model

The criterion for optimality is the minimum sum of jump times.

$$\min \sum_{n=1}^{N} \mathbf{t}[n], \quad \forall n \in \mathbb{N} \subset [1, N]$$

Extended Sate Machine

During a jump, the hopping leg goes through distinct states, and these states transition through discrete events. To effectively address these challenges, an extended state machine is implemented as the core control structure. A PD control scheme is imposed for the granular segmented jumping phases, with appropriate gains and targets assigned for each task.

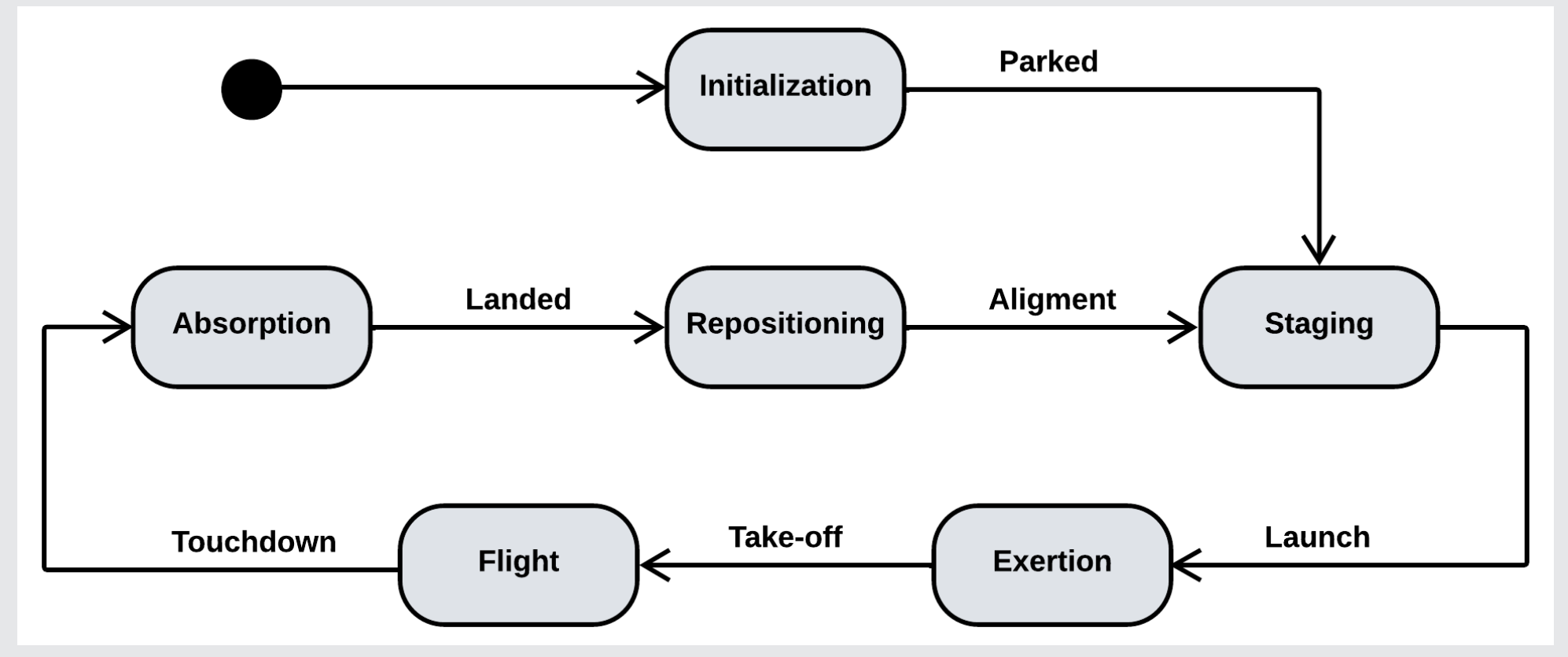

Figure: State Machine Diagram

Extended State Machine (ctnd.)

Table: PD Gains and Target Configuration per State

State	Gains				Target	
	Hip		Knee			
	K_p	K_d	K_p	K_d	r [m]	θ [deg]
Staging	11	0.3	9	0.5	0.14	θ_d
Exertion	0	0	0	0	-	-
Flight	2	0.2	2	0.2	0.17	93
Absorption	0.1	0.5	0.1	0.5	0.15	θ_d
Repositioning	4	0.3	4	0.3	0.14	θ_d

Table: Trigger Conditions per Event

Event	Trigger
Launch	$v_f \le 0.003$ $0.12 \le r \le 0.14$ $\theta_d - 2 \le \theta \le \theta_d + 2$ $n \le N - 1$
Take-off	$r \ge 0.2$
Touchdown	$\tau_h \ge \tau_{th} \vee \tau_k \ge \tau_{th}$ $r \le 0.17$
Landed	$v_{ee} \le 0.1$ $r \le 0.15$
Alignment	$v_{ee} \le 0.1$ $\theta_d - 5 \le \theta \le \theta_d + 5$

Experimental Results

A parkour course with six obstacles and four restricted areas is to be traversed with a total of twelve jumps. The goal region is reached with a complete yaw rotation after 7.2 m.

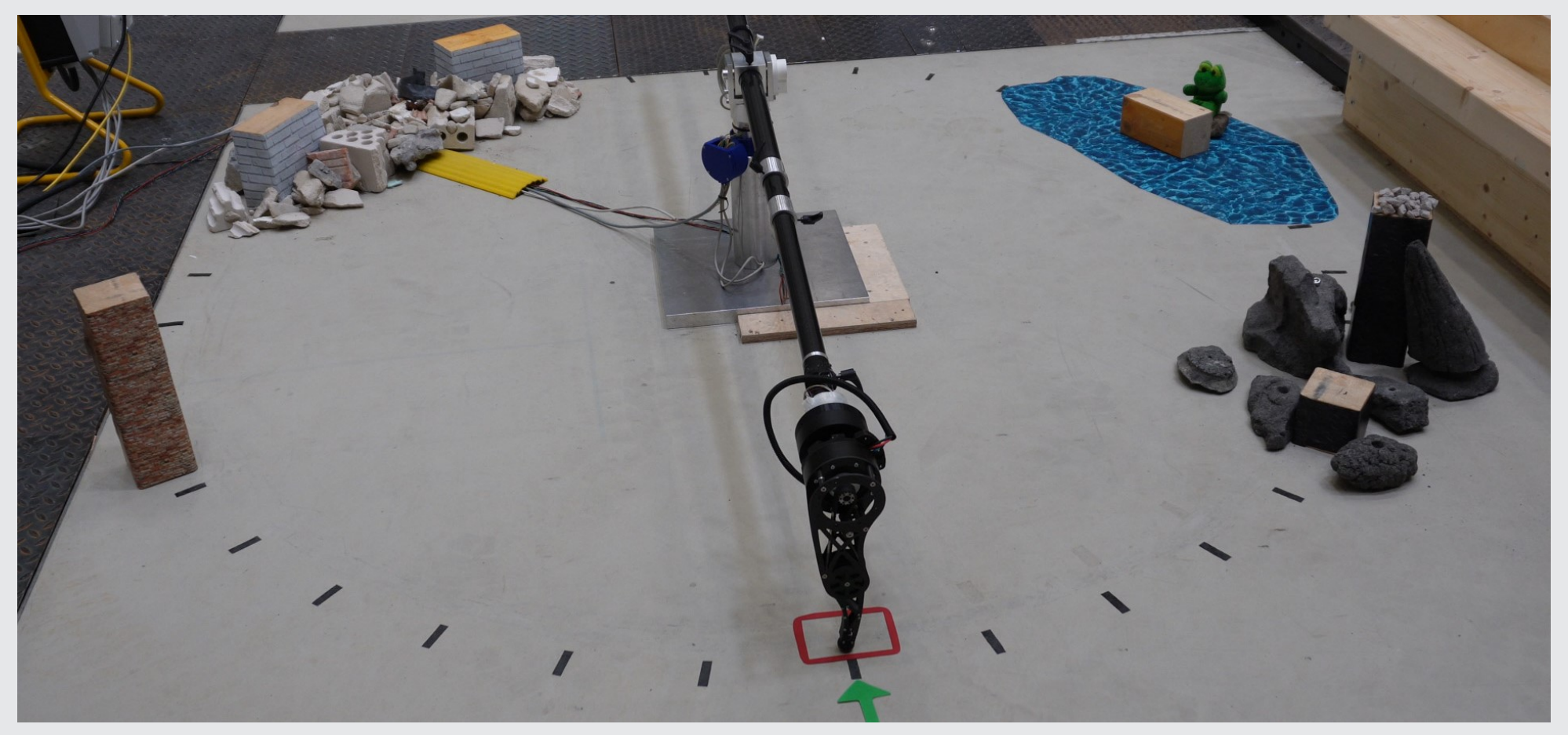
Figure: Experiment Setup

The motion planning is solved with SNOPT [2], leveraging the Mixed Integer Branch and Bound process within Drake [3]. The processor used is an Intel(R) Core(TM) i7-2620M CPU @ 2.70GHz. It takes 11.461 s to find an optimal solution. The planned contact sequence is then executed with a motor control loop frequency of 130 Hz.

Experimental Results (ctnd.)

The measured foot trajectory is kinematically determined with HyRoDyn [4] and projected onto the two-dimensional plane of the idealized workspace. In the final jump, a backflip is executed as a show-off element.

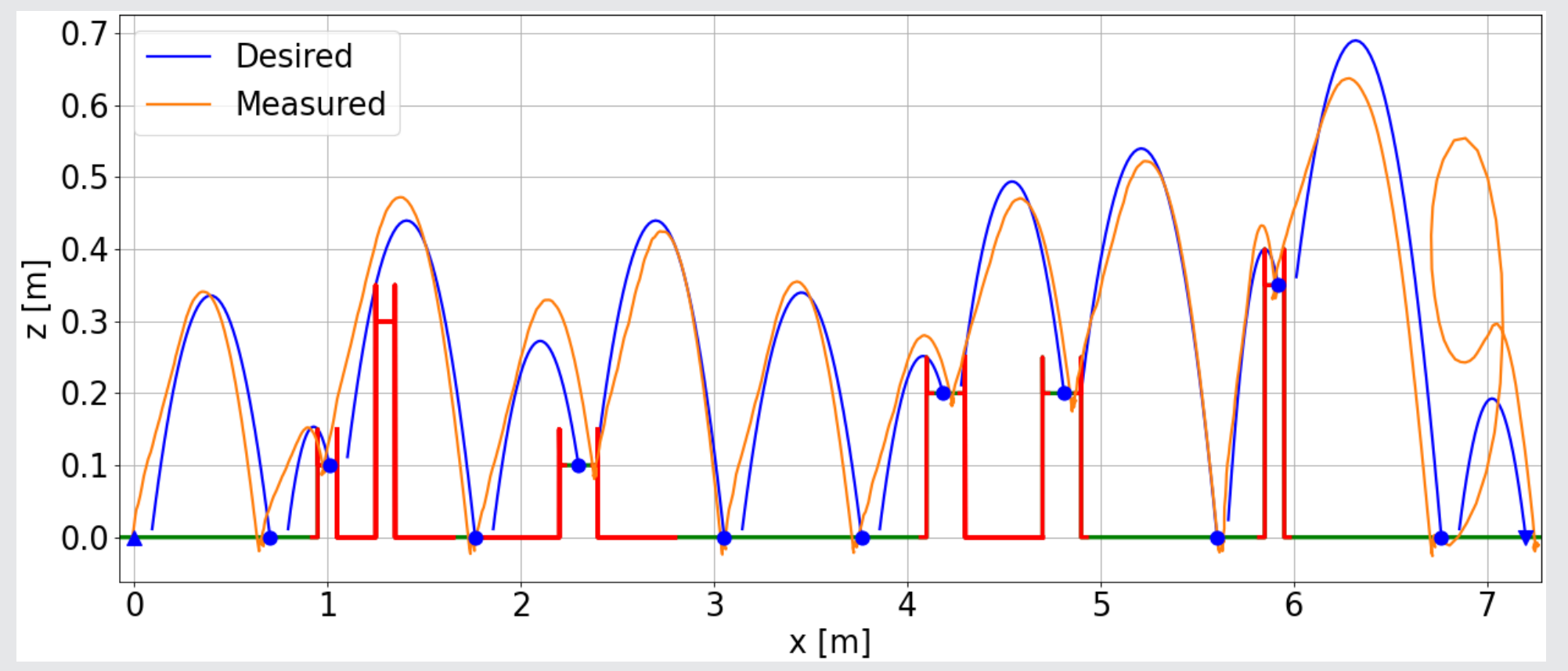

Figure: Planned and Measured Foot Trajectory

The output generated by the controller are the desired actuator torques for each time step.

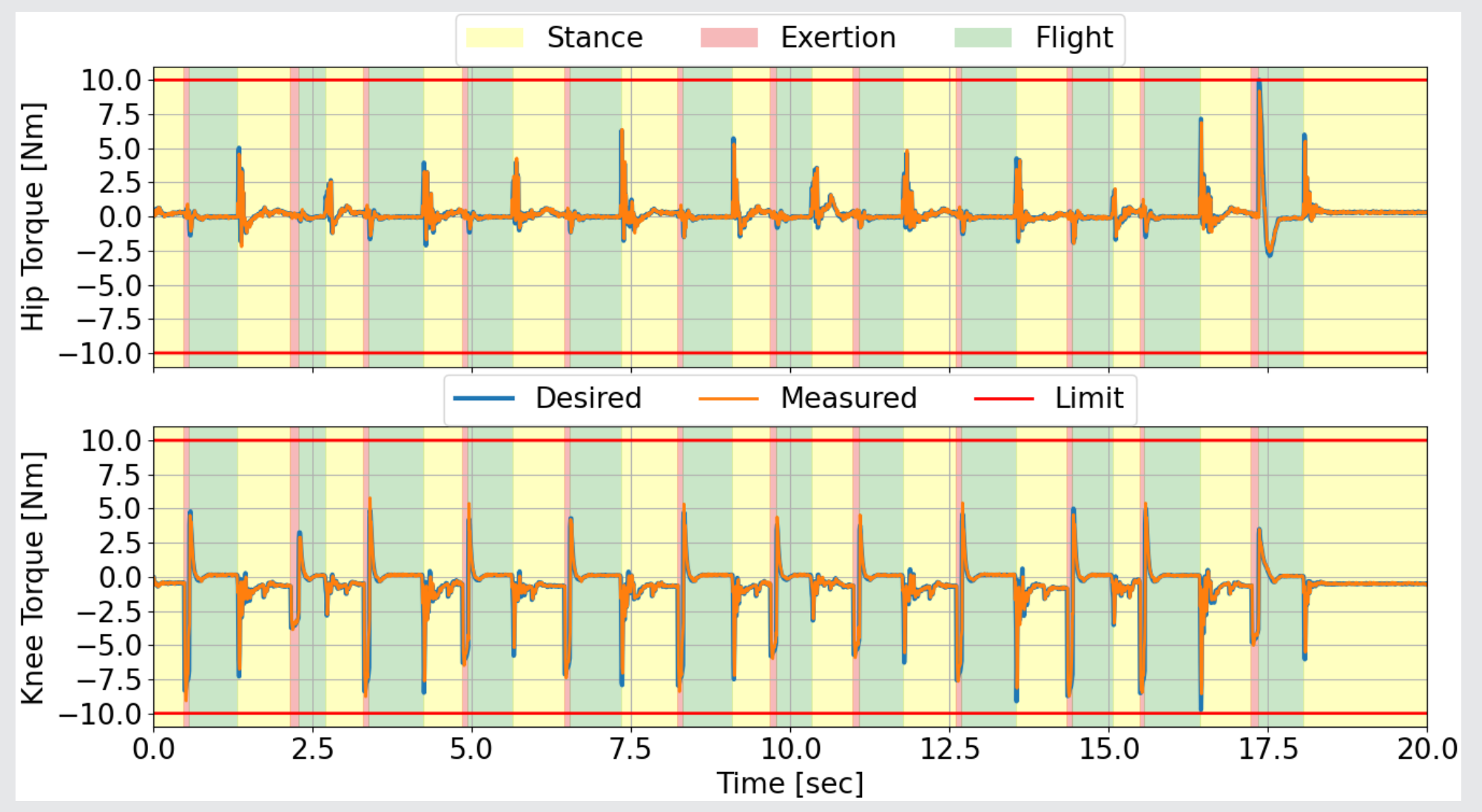

Figure: Torque Trajectory

Acknowledgment

This work was supported by the federal state of Bremen for setting up the Underactuated Robotics Lab under Grant 201-342-04-2/2021-4-1.

References

[1] mjbots. *qdd100 beta 3 servo*. URL: https://mjbots.com/products/qdd100-beta-3.

[2] Philip E. Gill, Walter Murray, and Michael A. Saunders. "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization". In: *SIAM Review* 47 (1 Jan. 2005), pp. 99–131. ISSN: 0036-1445. DOI: 10.1137/S0036144504446096.

[3] Russ Tedrake and the Drake Development Team. *Drake: Model-based design and verification for robotics*. 2019. URL: https://drake.mit.edu.

[4] Shivesh Kumar. "Modular and Analytical Methods for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots". PhD thesis. Bremen, Germany: Universität Bremen, 2019.