Go Fetch! - Dynamic Grasps using Boston Dynamics Spot with External Robotic Arm

Simon Zimmermann[1], Roi Poranne[2], Stelian Coros[1]

***Abstract*— We combine Boston Dynamics Spot® with a light-weight, external robot arm to perform dynamic grasping maneuvers. While Spot is a reliable, robust and easy-to-control mobile robot, these highly desirable qualities come with the price that the control access granted to the user is restricted. Consequently Spot's behavior must largely be treated as a black box, which causes difficulties when combined with a moving payload such as a robotic arm. We overcome the arising challenges by building a model of the combined platform, fitting the corresponding model parameters using experimental data and a straight-forward optimization framework. We use this model to generate control commands for the physical platform using trajectory optimization. We demonstrate that even with a simple model, and control trajectories deployed in a feed-forward manner, the combined platform is capable of executing grasping tasks in a dynamic fashion. Furthermore, we show how the platform can use the additional degrees of freedom of the legs to extend the reachability of the arm.**

I. Introduction

The field of robotics will soon enter a new era where robots are no longer confined to production halls, but are prevalent in our day to day lives. Highly publicized demonstrations of companies such as *Boston Dynamics* paint a picture of a future where robots also trot around in our neighborhoods. Beside that, the academic community has invested vast amounts of resources into developing mobile platforms such as *quadrupeds* [1], [2], [3]. In contrast to their wheeled counterparts, they can safely and robustly navigate through not only flat, but also rough and challenging terrains [4]. This extended locomotive versatility comes with the price that these platforms are notoriously difficult to control and stabilize. While academia has been making significant contributions to an open-sourced quadruped ecosystem (e.g. [5]), industry decisively competes in an effort to commercialize new technologies as well. In particular, *Boston Dynamics* has been pioneering legged robotic platforms for decades, and recently made one of their platforms commercially available: The Spot® robot [6] is a mid-sized quadruped with 360° vision and integrated obstacle avoidance. Its primary intended use is to perform autonomous inspection missions of industrial plants, construction sites or other facilities that pose a rather high level of danger to humans. Spot is designed to be stable, robust and easy to operate, allowing it to successfully navigate through diverse environments. This ability arguably makes it to one of the most sophisticated legged platforms currently available. However, being

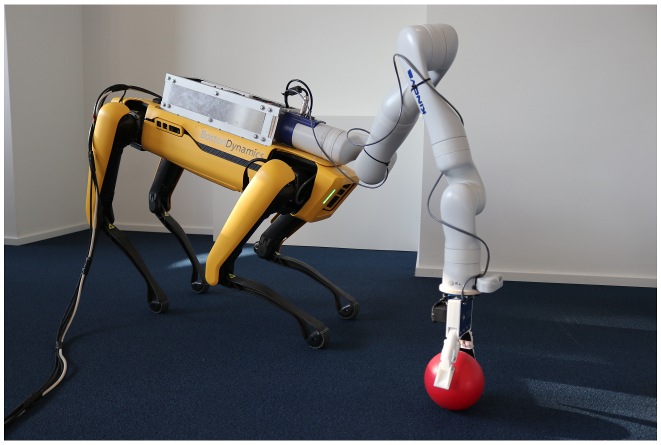


Fig. 1: Boston Dynamics Spot® equipped with an external, light-weight robotic arm and an under-actuated gripper. The combined platform called *Spova* can use both arm and legs to comfortably reach objects lying on the ground.

a commercial product, some of the technology built into Spot remains a trade secret, and is inaccessible to the user. Control commands, for example, are limited exclusively to body pose and velocity, while commands to individual legs or motors are impermissible. This design relieves the user from the burden of motion optimization and provides a safeguard from reckless operation. However, direct leg control is not enabled, and consequently, users have no means of regulating foot placement. In other words, the behavior of Spot must largely be treated as a black box.

In this paper, we study Spot as an augmentable research platform and discuss the impact of its restricted control access on its capabilities. Since Boston Dynamics' own proprietary arm is not yet available, we equip Spot with an external, separately controlled robotic arm and gripper (depicted in Figure 1), and explore the challenges and abilities of such a setup. We argue that combining public and proprietary hardware is becoming highly valuable in the field of robotics [7], [8]. To this end, we present a generic optimization-based framework that first optimizes for model parameters based on real-world measurements, and then uses these results to generate optimal nominal control trajectories. We then apply this methodology to our specific platform combination. Since it is comprised of a Kinova® arm mounted on Spot, we will refer to it as *Spova* for convenience purposes.

The major challenge stems mainly from Spot's nature of operation: a highly reliable mobile robot, but with restricted control access. We suggest a solution by formulating a simple

[1] The authors are with the Department of Computer Science, ETH, Zurich, Switzerland. `simon.zimmermann@inf.ethz.ch; roi.poranne@inf.ethz.ch; scoros@gmail.com`

[2]Department of Computer Science, University of Haifa, Haifa, Israel

model of the dynamics of Spova, which we evaluate on a large variety of real-world experiments. While reliably grasping objects is arguably still challenging for *statically* mounted robotic arms [9], we further increase the difficulty of the problem by investigating how Spova can *dynamically* pickup objects. By doing so, we are working towards the goal of making mobile robots more efficient in performing manipulation tasks. Furthermore, we demonstrate how the arm can benefit from the additional degrees of freedom of the legged body in order to expand its reach and perform more flexible grasping motions. We show that even for a simple model, fast and dynamic maneuvers can be achieved while benefiting from short computation times.

II. Related Work

The literature on legged, mobile manipulators is vast, and we only mention a small selection of relevant work [10]. Particular examples are IIT's HyQ robot combined with a hydraulic manipulator [7] and ANYmal with a collaborative arm [8], [11]. [7] proposed a stabilizing control procedure that treats body disturbances caused by the arm by optimizing for the ground reaction forces generated from the legs. [8] presented an optimization-based framework for a torque-controlled platform relying on an inverse dynamics model while focusing on robustness against external forces. [11] used a similar hardware setup, and presented a method to explicitly incorporate a metric for robustness into the motion planning process. Another related physical platform is the CENTAURO [12]. [13] uses this centaur-like humanoid robot to push heavy objects with the help of its environment. Other approaches considered cases where robots use their legs to perform manipulation tasks, either with an additionally attached gripper [14], or by using their feet [15], [16].

As common for a legged robots [17], we model Spova as an arm on a free-floating base. However, in stark contrast to previous work, we cannot rely on a faithful dynamical model based on forces and torques due to the nature of the restricted control access of Spot. Consequently, we use kinematics to formulate a simple, *empirical* model. To ensure that this model predicts reality well, we execute a *parameter identification* procedure. We refer the interested reader to [18], [19] for an overview of different methodologies in this area. Our approach is closely related to [20], which computed material parameters by minimizing the difference between measured and simulated data in a least-squares sense.

An alternative avenue to build black box models for real-world robotic systems control are *learning-based* approaches [21], [22], [23], [24]. While these techniques typically generalize better and can be applied to a wider range of scenarios, they often require large data sets and extended training sessions. In contrast, our simple optimization-based approach benefits from short computation times, which allows for fast prototyping of different empirical models. Overall, we see our core contribution in the demonstration that even with the use of a simple model and well-known, straightforward techniques, a variety of dynamic maneuvers can be successfully executed on the combined hardware platform.

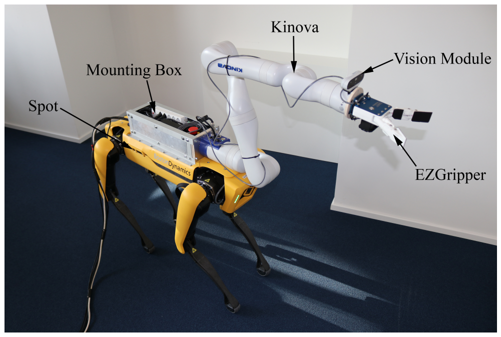


Fig. 2: Hardware setup of Spova: Spot® is carrying the Kinova® arm, which is equipped with the SAKE EZGripper™. All communications to an external computer are currently enabled by cables.

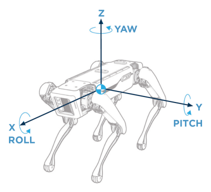
III. Setup

A. Hardware

Boston Dynamics showcased Spot with its own customized robotic arm [25] several years ago, but it is not commercially available as of yet. Spot can carry payloads of up to 14 kg, which limits the range of arms that can be mounted on it. In this work, we employ the Kinova® Gen3 [26] with 7 degrees of freedom and integrated vision module as our manipulator. With a weight of 8.2 kg, maximum payload of 4.0 kg, and a reach of 902 mm, it is an adequate off-the-shelf option for Spot to carry. We mount Kinova on Spot using a custom-made metal box, fabricated from standard aluminium angle profiles and plates, which can be screwed onto Spot's mounting rails (see Fig. 2). As for the gripper, we choose the underactuated EZGripper™ Gen2 from SAKE Robotics [27]. The mounting box is spacious enough for the power supply of both arm and gripper. In our current setup, cables are used for both communication and power supply, meaning that both Spot and the arm receive commands from an external computer via Ethernet, while the gripper currently receives commands via USB. Our future plans include setting up a cable-free configuration, since all platforms are in fact capable of Wifi communication, and arm as well as gripper can be powered using batteries. Nevertheless, our method is agnostic to the mode of communication.

B. Communication Interface

We send control commands to the hardware using an external PC that runs our framework written in C++. Boston Dynamics provides a Python API to communicate with Spot [28], which we interface with using *pybind11* [29]. This API allows reading the current state of the robot including its body pose, which is computed internally using information from its onboard sensors, and can be expressed relative to a coordinate system determined during boot up. While commands can only be directed to the body, the API allows to select predefined gait patterns. Out of the possible gaits, the *trot* was observed to be the most appropriate, since it results in fast, agile and stable motions. Depending on whether the robot stands or walks, the body reacts to commands in a slightly different manner. In *stand* mode, the full body pose can be set, subject to predefined ranges based on the physical limitations. The individual angles are shown in the inset image, taken from the *Spot User Guide*.



In *walk* mode, the roll and yaw angles are more restricted, but the behaviour of pitch and body height remain similar to stand mode. Position or velocity commands can additionally be sent to trigger motion, where the command reference frame can be specified by the user. Our experiments showed that sending velocity commands in body frame (i.e. forward, sideways and rotational velocity) exhibit the best performance in terms of smoothness and flexibility. The API also allows to configure a *payload*, i.e. a load with a predefined mass, center of mass and moment of inertia. This information is used by the robot's internal locomotion controllers. Currently, and to the best of our knowledge, it is only possible to configure a *static* payload; changing payload information during operation in not permissible according to Boston Dynamics [30]. This is the main challenge to address when Spot is carrying an external moving arm as a payload, which we discuss in Sec. IV.

To communicate with the Kinova arm, we use the Kortex$^{\text{TM}}$ API in C++ [31] provided by the manufacturer. It allows sending velocity commands for the individual joints, and read the current joint angles from the robot. The SAKE EZGripper uses a Dynamixel motor manufactured by ROBOTIS, enabling the use of the Dynamixel SDK in C++ [32] to set gripper finger positions and gripping force.

In our framework, the communication to each of these platforms runs on an independent thread. To accurately track the targets set by our motion planner presented in section IV, and avoid drift and offsets resulting from sending velocity commands, we use PID control [33] within each of these control loops. To do so, we leverage the onboard pose estimation provided by each of the hardware platforms.

IV. Methods

A. Methodology

Our approach is to emulate the platform's behaviour using a simplistic, parameterized dynamics model, which we discuss in Sec. IV-B. As such, our main goal boils down to the generation of optimal control trajectories for this system. In contrast to the aforementioned previous work, we cannot rely on a finely tuned model, but as we demonstrate, our simple model performs sufficiently well after a parameter identification process. We formulate this procedure as follows: Let $\mathbf{x}$ and $\mathbf{u}$ represent the entire *time-discretized* state and control trajectories, and let $\mathbf{p}$ denote a set of model parameters. We can express $\mathbf{x}$ as a function of an initial state $\mathbf{x}_0$, $\mathbf{u}$ and $\mathbf{p}$, that is $\mathbf{x} = \mathbf{x}(\mathbf{u}, \mathbf{x}_0, \mathbf{p})$. We wish to identify $\mathbf{p}$ based on measurements. To this end, we run several simulations with different control inputs and initial states, denoted by $\bar{\mathbf{u}}^j, \bar{\mathbf{x}}_0^j$. We compare these simulations to the real, *measured* states $\hat{\mathbf{x}}^j = \hat{\mathbf{x}}(\bar{\mathbf{u}}^j, \bar{\mathbf{x}}_0^j)$ obtained by applying the same control sequences on the real system starting from the same initial conditions. The ideal $\mathbf{p}$ would minimize the difference between simulation and reality. We pose this as an optimization problem, searching for $\mathbf{p}$ that minimizes

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \sum_{j=1}^{n_d} \left\| \mathbf{x}(\bar{\mathbf{u}}^j, \bar{\mathbf{x}}_0^j, \mathbf{p}) - \hat{\mathbf{x}}^j \right\|^2 \tag{1}$$

for n_d experiments. Using $\mathbf{p}^*$, we can then solve the trajectory optimization problem

$$\min_{\mathbf{u}} \mathcal{O}(\mathbf{x}(\mathbf{u}, \mathbf{x}_0, \mathbf{p}^*)), \tag{2}$$

where we optimize for the control sequence $\mathbf{u}$ with respect to a certain objective $\mathcal{O}$, which we discuss in Sec. IV-C.

B. Model

Since we can only send body commands to Spot, we choose to model Spova as a kinematic arm with floating base. Therefore, the state is represented by the stacked position and orientation of the body and the arm's joint angles, denoted by q^k for the k^{th} degree of freedom. Consequently, the state $\mathbf{x}_i$ at time step i is parameterized by $\mathbf{x}_i = (q_i^1, \ldots, q_i^{n_q+6}) \in \mathbb{R}^{n_q+6}$, where $n_q = 7$ is the number of the arm's joints. We express the body orientation in Euler Angles, and note that we do not run into Gimbal lock issues as long as we choose the Euler axes such that the lock configuration is not within the body constraints imposed by Spot's physical limits. The control input $\mathbf{u}_i$ at time step i represents the velocity commands that we send to the individual platforms. Given these components, we can formulate a basic dynamical system

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h \cdot \mathbf{u}_i, \tag{3}$$

where h is the step size. While this simple model appears to well-approximate the behavior of the body and the arm when they are operated independently, it fails when combined together, due to the additional forces that the arm applies to the body. Spot's API does allow configuring customized payloads, and in particular, it allows specifying details about the mounting position, total mass, center of mass (CoM) and moment of inertia (MoI) of the payload, which are likely used in its internal controllers. However, Spot's API does currently not allow to change the payload's configuration *during* operation [30], and since we naturally wish the arm to move, this poses a limitation. Without any payload updates, Spot starts to drift when the arm is moved, causing it to deviate from its planned route. This becomes especially problematic when performing dynamic maneuvers with the arm when the quadruped is in fast motion. We address this problem with the following strategy: We attempt to correct the model using an unknown correction term $\mathcal{B}$. It aims to reduce the discrepancy between the nominal state $\bar{\mathbf{x}}_{nom}$, and the actual one, as well as to express the interaction between the body and the arm using the current and past positions of the CoM of the arm. Specifically, we add $\mathcal{B}$ to the *positional* part of (3), and therefore extend it to

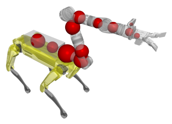
$$\mathbf{x}_{i+1}^{pos} = \mathbf{x}_i^{pos} + h \cdot \mathbf{u}_i^{pos} + \mathcal{B}(\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{x}_{i-2}), \tag{4}$$

where the superscript pos is understood as taking only the positional part of state (e.g. q_i^1, q_i^2, q_i^3) and control. The past states serve the purpose of approximating the velocity and acceleration of the CoM by first order discretization. The term $\mathcal{B}$ is defined as

$$\begin{aligned}\mathcal{B}(\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{x}_{i-2}) = & \, p_p(\mathcal{C}(\mathbf{x}_i) - \mathcal{C}(\bar{\mathbf{x}}_{nom})) \\ & + \frac{p_v}{h}(\mathcal{C}(\mathbf{x}_i) - \mathcal{C}(\mathbf{x}_{i-1})) \\ & + \frac{p_a}{h^2}(\mathcal{C}(\mathbf{x}_i) - 2\mathcal{C}(\mathbf{x}_{i-1}) + \mathcal{C}(\mathbf{x}_{i-2})),\end{aligned} \tag{5}$$

where $\mathcal{C}(\mathbf{x}_j)$ is the CoM of the arm at state $\mathbf{x}_j$, To estimate the CoM, we use $n_m = 13$ point masses (see inset below). Then,

$$\mathcal{C}(\mathbf{x}_i) = T \cdot \frac{1}{\sum_{j=1}^{n_m} m_j} \sum_{j=1}^{n_m} m_j \mathcal{K}(\mathbf{x}_i, \mathbf{l}_j), \tag{6}$$



where m_j are the masses, $\mathcal{K}$ is the forward kinematics function, $\mathbf{l}_j$ are the local coordinates of the masses, and T transforms the CoM into the body frame. Eq. (5) contains three model parameters p_p, p_v and p_a, which weight the influence of the CoM's position, velocity and acceleration on the body, respectively. The position is computed as an offset from the nominal state $\bar{\mathbf{x}}_{nom}$, which is also used to configure a static payload using Spot's API (configuration as shown in the inset).

C. *Trajectory Optimization*

Following real-world measurements and parameter optimization (1), we use the resulting model to generate optimal control trajectories for Spova. We do so by optimizing the complete trajectory at once

$$\min_{\mathbf{u}} \quad \mathcal{O}(\mathbf{x}(\mathbf{u}, \mathbf{x}_0, \mathbf{p}^*)) \tag{7a}$$

$$\text{s.t.} \quad \mathbf{g}(\mathbf{x}(\mathbf{u}, \mathbf{x}_0, \mathbf{p}^*)) \leq 0, \tag{7b}$$

$$\mathbf{h}(\mathbf{x}(\mathbf{u}, \mathbf{x}_0, \mathbf{p}^*)) = 0 \tag{7c}$$

using the objective $\mathcal{O}$ and inequality and equality constraints $\mathbf{g}$ and $\mathbf{h}$, respectively.

Equality Constraints are used to specify pick and place targets. Given an object with a global pose $\mathbf{z}$, we look for a state $\mathbf{x}_i$ at a predefined time i such that the object is reachable by the gripper. This can be done via the IK constraint

$$\mathcal{K}(\mathbf{x}_i, \mathbf{e}) - \mathbf{z} = 0, \tag{8}$$

where $\mathbf{e}$ is the gripper pose in local coordinates. Equality constraints are also used to specify the return to a predefined state $\bar{\mathbf{x}}_i$, i.e.

$$\mathbf{x}_i - \bar{\mathbf{x}}_i = 0. \tag{9}$$

Inequality Constraints are used to specify hardware limitations, i.e. joint or body pose limits, as well as limits on velocities and accelerations. We enforce these limits on each degree of freedom q using box constraints $q_{\min} \leq q \leq q_{\max}$, where $q_{\min}$ and $q_{\max}$ are the lower and upper bounds, respectively. Similarly, we model velocity and acceleration limits by first order discretizations. Note that we need to enforce these constraints for all time steps of the trajectory.

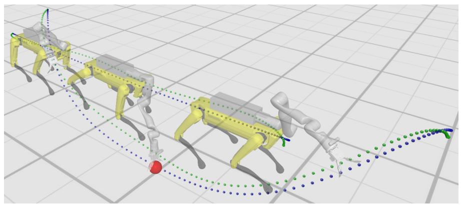
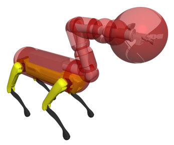


Fig. 3: Trajectory planning in simulation: Spova snatches a ball off the ground while walking by. The framework generates control inputs (in green) for an optimal state target trajectory (in blue). Spova is shown at three different instances. Note how the control trajectory of the body compensates for the disturbance created by the arm's movements based on the optimized model.



Inequality constraints are also used for collision avoidance, i.e. to ensure that arm and body do not collide with themselves, each other, or the environment. To this end, we approximate both body and arm with collision primitives as depicted in the inset, where we use a combination of spheres and capsules. To avoid self-collision, we add a constraint for each pair of collision primitives of non-consecutive links in the kinematic chain. We formulate such a constraint as

$$d(\mathcal{K}(\mathbf{x}_i, \mathbf{c}^j), \mathcal{K}(\mathbf{x}_i, \mathbf{c}^k)) \geq r^j + r^k, \tag{10}$$

where $d(\mathcal{K}(\mathbf{x}_i, \mathbf{c}^j), \mathcal{K}(\mathbf{x}_i, \mathbf{c}^k))$ denotes the shortest distance function between two collision primitives j and k with radius r^j and r^k, and local coordinates $\mathbf{c}^j$ and $\mathbf{c}^k$, respectively.
Besides self-collision, it is important that Spova does not collide with its environment. To this end, external obstacles are approximated by the same collision primitives, and can therefore be handled by adding more constraints in the same manner. Equivalently, floor and walls are modelled as individual planes and added to the constraint list as well.

Objective. The resulting nominal trajectory should be as smooth as possible. This is achieved by minimizing the acceleration of the root pose, as well as the individual joint angles, which can be formulated with the objective

$$\mathcal{O}(\mathbf{x}) = \sum \|\mathbf{x}_i - 2\mathbf{x}_{i-1} + \mathbf{x}_{i-2}\|^2. \tag{11}$$

Solver. We solve both (1) and (7) using Newton's method. (1) is solved by computing the gradient using the chain rule

$$\frac{\mathrm{d}\mathcal{O}}{\mathrm{d}\mathbf{p}} = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{p}}^T \frac{\partial \mathcal{O}}{\partial \mathbf{x}} + \frac{\partial \mathcal{O}}{\partial \mathbf{p}}. \tag{12}$$

To solve (7), we avoid computing the jacobian $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{u}}$ by leveraging that the function $\mathbf{x}(\mathbf{u}, \mathbf{x}_0, \mathbf{p}^*)$ is invertible with respect to $\mathbf{u}$. We do so by expressing all objectives and constraints in terms of $\mathbf{x}$, and solve the optimization problem directly for $\mathbf{x}$. Afterwards we compute the corresponding

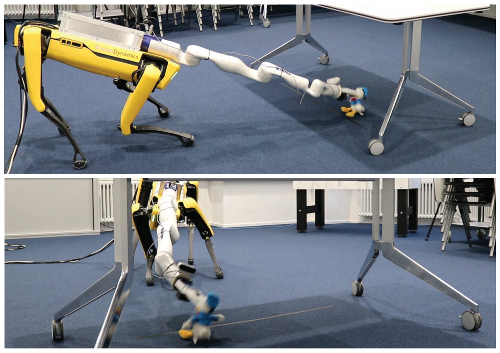


Fig. 4: Spova is leveraging the additional degrees of freedom of the body to dynamically pickup a toy without colliding with the table. The two images show the same scene from two different perspectives.

optimal $\mathbf{u}$ as a function of $\mathbf{x}$ by rearranging (4) for $\mathbf{u}$. Since Newton's method does not take constraints explicitly into account, we convert them into soft constraints using barrier functions as presented in [34]. A graphical example of a optimized trajectory in simulation is given in Figure 3.

V. Results

We evaluate the efficacy of our method by running several real-world experiments. To this end, Kinova's integrated vision module combined with standard vision techniques is used to detect the position of individual objects. All conducted experiments and demonstrations are shown in the accompanying video.

Follow. Spova interactively follows a visual fiducial marker wielded by the user, which is detected using Aruco Marker Detection [35]. The video shows how Spova uses the degrees of freedom of both the body and the arm to track the target with the onboard camera. It simplifies the task of keeping the marker within sight while not having to perform aggressive maneuvers. This demonstration is implemented in a *receding horizon* fashion: At each iteration of the high-level control cycle, the trajectory is replanned for a short time horizon based on Spova's and the target's current measured position. Then, the first control action of this new trajectory is given to the lower-level control cycles of the individual robots, where PID controllers provide accurate tracking.

Reach. Spova is tasked with retrieving an object that is placed under a table, making it difficult for the arm to reach it without risking collisions. The motion planner comes up with a solution that leverages the additional degrees of freedom of the body, enabling Spova to safely snatch the object off the ground and place it in the basket on the table. To generate a collision-free motion plan, the shape of the table is approximated by collision primitives as discussed in section IV. Fig. 4 and the video illustrate the scenario from two different view points.

Go Fetch is another experiment conducted, involving Spova to independently and repeatedly retrieve an object from previously unknown locations: First, the user places a ball somewhere in the room. Spova executes a simple search pattern to detect it with its camera using Circular Hough Transform [36]. When spotted, it computes the ball's position in world coordinates using the camera intrinsics, and heads out to retrieve it. After a successful grasp, it returns it to the basket, which remains at a fixed, known location. After returning to the initial configuration, the game starts again. Fig. 5 showcases these different stages, and the video shows several runs. The framework independently switches between the different stages using a simple state machine, such that the user does not have to intervene during the game. We note that the deployed ball detection method can lead to inaccurate position estimates when there is a large distance between ball and camera. Instead of applying a more sophisticated detection method, we leverage the fact that the pose estimation's accuracy increases with decreasing distance to the target. We do so by updating the motion plan based on the new detection data while Spova is moving towards the ball. This is realized by re-solving the same optimization problem that was used to create the first command trajectory at each high-level control iteration. In addition, we add a regularization objective that matches Spova's pose at the closest time step of the planned trajectory with its current *measured* pose. It supports smooth control commands generated by the new motion plan.

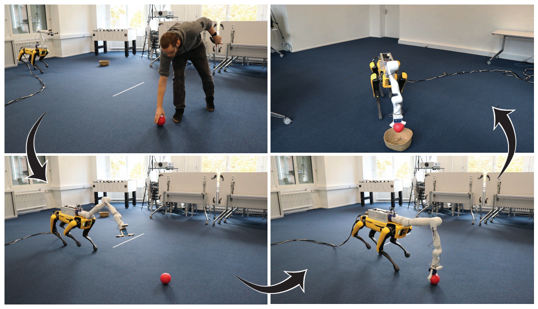


Fig. 5: User is playing "Go Fetch" with Spova . First, a ball is placed somewhere in the room (top left image). Spova independently detects it (bottom left), picks it up (bottom right), and returns it to the basket (top right).

Snatch. The goal of this experiment is to dynamically snatch an object off the ground while walking by. Fig. 6 and the video demonstrate how Spova executes this task. The experiment is conducted several times in a row, where the object is shifted along a straight line to create different scenarios, testing the robustness and reliability of the presented method. Spova first detects the object's position on the line, plans its path, and after execution returns to the initial configuration, where it waits for the object to be placed again by the user. We note that the timing of the different hardware components is crucial to successfully grasp the object, especially for the gripper. The gripper's *close* command is implemented as a discrete event, and therefore needs to be triggered before the end-effector reaches its target. This means that the command needs to be administrated at a slightly earlier state of the trajectory. Since we plan the complete trajectory beforehand, we can precompute the end-effector position where the gripper needs to be activated by taking the closing

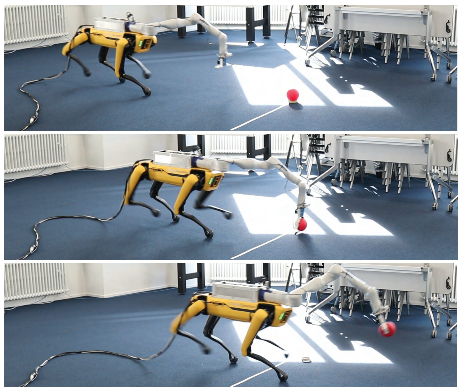


Fig. 6: Spova snatching the ball dynamically off the ground while walking by. The images show the execution before (top), during (middle) and after (bottom) the grasp.

velocity of its fingers into account. Using the pose estimation of the two robots, the trigger is sent as soon as Spova is sufficiently close to this precomputed position. The success of this approach depends on the assumption that Spova can track the trajectory accurately, but makes it more flexible regarding time delay.

VI. Discussion and Future Work

We analyze the robustness and limitations of our setup by running several tests on the *Snatch* experiment presented in section V. The task for Spova is to repeatedly snatch a ball off the ground from different positions on a straight line while walking by. To find optimal model parameters, we use nine different data sets, where we run the experiment from the same initial condition for different time frames and ball positions. We compute control trajectories with all model parameters set to zero, apply it to the physical platform, and collect the measured states from both hardware components at each high-level control cycle. These data points are then used to fit the model parameters in a least-squares manner by solving (1). Using the resulting model to generate control trajectories, we show in the accompanying video that the experiment can be successfully repeated several times in a row. Some quantitative performance measurements for these experiments can be found in Table I. Fig. 7 illustrates that the optimized model is crucial for a successful grasp of the ball. The plots show the position of the body (upper plot) and the gripper (lower plot) in world coordinates in the lateral direction of Spova's body. The body is supposed to track a straight line, while the arm swings sideways to pick up the ball while walking by. We first conduct the experiment with all model parameters set to zero. In this case, the control trajectory coincides with the target state trajectory (in blue). When sending these commands to Spova, the body drifts sideways due to the forces induced by the arm's movements (in red). The control trajectory (in green) generated from the previously fitted model counteracts this behavior, leading to a closer tracking of the target state trajectory (in pink). This brings the gripper close enough to the ball such that Spova can successfully grasp it.

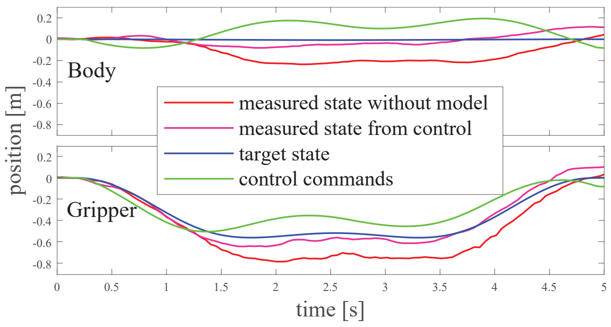


Fig. 7: Plots showing measurements from the *Snatch* experiment. They depict data of the position of the body (upper) and the gripper (lower) in world coordinates in the lateral direction of Spova's body. When sending the target state (blue) as control commands to Spova, the body starts to drift sideways due to the arm's movements (red). The control commands generated from the model (green) counteract this behavior, which brings the gripper close enough to the ball to successfully grasp it (pink).

TABLE I: Quantitative measurements conducted for the *Snatch* experiment using an Intel Core i7-7709K 4.2Ghz PC. They have been averaged over several different runs with varying ball positions. N is the number of discrete steps.

Parameter fitting time (9 runs, in total 16796 data points)	$0.9307s$
Trajectory optimization time ($N = 100$)	$0.7596s$
Average body correction magnitude: $\sum_{i=0}^{N} \frac{1}{N} \lvert\lvert \mathcal{B}_i \rvert\rvert$	$0.1008m$
Max body correction magnitude: $\max_i \lvert\lvert \mathcal{B}_i \rvert\rvert$	$0.1997m$
Gripper tracking error: $\sum_{i=0}^{N} \frac{1}{N} \lvert\lvert \mathcal{K}(\mathbf{x}_i, \mathbf{e}) - \mathcal{K}(\hat{\mathbf{x}}_i, \mathbf{e}) \rvert\rvert$ *with* / *without* model fitting	$0.0494m$ / $0.1062m$
Success rate ball grasps	80%

In the video, we also show cases where Spova fails to grasp the ball. These cases typically arise due to inaccurate ball position estimation, or if Spova starts the trajectory execution with some offset from the planned initial condition. Due to the fact that we are currently deploying the high-level control trajectory in an open-loop fashion, Spova is not able to make up for these inaccuracies during the execution. A solution would be to introduce an additional high-level feedback loop, in particular in form of a model predictive control approach [37]. We started to investigate this topic, and will continue to do so in order to further increase the reliability and robustness of the approach. Another reason why Spova sometimes misses the ball is due to a more complex behavior of Spot's internal controller that are currently not captured by our simple model. For example, Spot sometimes slows down due to the disturbances created by the arm's movements, causing a delay that brings the individual components out of sync. This behavior could be reflected by a more sophisticated model. An alternative to our current approach would be to generate more data through physical measurements and use it to train a more comprehensive model using machine learning. Nevertheless, with our current setup, Spova is able to grab the ball successfully four out of five times.

References

[1] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ – a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, Sept. 2011, publisher: IMECHE.

[2] S. Seok, A. Wang, M. Y. Chuah, D. J. Hyun, J. Lee, D. M. Otten, J. H. Lang, and S. Kim, "Design Principles for Energy-Efficient Legged Locomotion and Implementation on the MIT Cheetah Robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1117–1129, June 2015.

[3] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "ANYmal - A Highly Mobile and Dynamic Quadrupedal Robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 38–44.

[4] C. D. Bellicoso, M. Bjelonic, L. Wellhausen, K. Holtmann, F. Günther, M. Tranzatto, P. Fankhauser, and M. Hutter, "Advances in real-world applications for legged robots," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1311–1326, 2018.

[5] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[6] "Spot® | Boston Dynamics." [Online]. Available: https://www.bostondynamics.com/spot

[7] B. U. Rehman, M. Focchi, J. Lee, H. Dallali, D. G. Caldwell, and C. Semini, "Towards a multi-legged mobile manipulator," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3618–3624.

[8] C. D. Bellicoso, K. Kramer, M. Stauble, D. Sako, F. Jenelten, M. Bjelonic, and M. Hutter, "ALMA - Articulated Locomotion and Manipulation for a Torque-Controllable Robot," in *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE, May 2019, pp. 8477–8483.

[9] O. Z. Lviv, GDG, "Failure to grasp." [Online]. Available: http://failtograsp.github.io

[10] B. U. Rehman, D. G. Caldwell, and C. Semini, "Centaur robots - a survey," in *Human-Centric Robotics*. World Scientific, Aug. 2017, pp. 247–258.

[11] H. Ferrolho, W. Merkt, V. Ivan, W. Wolfslag, and S. Vijayakumar, "Optimizing Dynamic Trajectories for Robustness to Disturbances Using Polytopic Projections," *arXiv:2003.00609 [cs]*, Aug. 2020.

[12] "H2020 Project CENTAURO." [Online]. Available: https://www.centauro-project.eu

[13] M. P. Polverini, A. Laurenzi, E. M. Hoffman, F. Ruscelli, and N. G. Tsagarakis, "Multi-Contact Heavy Object Pushing With a Centaur-Type Humanoid Robot: Planning and Control for a Real Demonstrator," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 859–866, Apr. 2020.

[14] G. Heppner, T. Buettner, A. Roennau, and R. Dillmann, "Versatile - high power gripper for a six legged walking robot," in *Mobile Service Robotics*. World Scientific, July 2014, pp. 461–468.

[15] J. Whitman, S. Su, S. Coros, A. Ansari, and H. Choset, "Generating gaits for simultaneous locomotion and manipulation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2017, pp. 2723–2729, iSSN: 2153-0866.

[16] W. Wolfslag, C. McGreavy, G. Xin, C. Tiseo, S. Vijayakumar, and Z. Li, "Optimisation of Body-ground Contact for Augmenting Whole-Body Loco-manipulation of Quadruped Robots," *arXiv:2002.10552*, Feb. 2020.

[17] O. E. Ramos, "Step-by-Step Kinematic Modeling and Control of a Robot with a Free-Floating Base," in *2018 IEEE ANDESCON*, Aug. 2018.

[18] J. Wu, J. Wang, and Z. You, "An overview of dynamic parameter identification of robots," *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 5, pp. 414–419, Oct. 2010.

[19] R. Mahnken, "Identification of Material Parameters for Constitutive Equations," in *Encyclopedia of Computational Mechanics Second Edition*. American Cancer Society, 2017, pp. 1–21.

[20] D. Hahn, P. Banzet, J. M. Bern, and S. Coros, "Real2Sim: visco-elastic parameter estimation from dynamic motion," *ACM Transactions on Graphics*, vol. 38, no. 6, pp. 236:1–236:13, Nov. 2019.

[21] A. S. Polydoros and L. Nalpantidis, "Survey of Model-Based Reinforcement Learning: Applications on Robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, May 2017.

[22] K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, and J. Mouret, "Black-box data-efficient policy search for robotics," Sept. 2017, pp. 51–58, iSSN: 2153-0866.

[23] K. Chatzilygeroudis and J. Mouret, "Using Parameterized Black-Box Priors to Scale Up Model-Based Policy Search for Robotics," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 5121–5128, iSSN: 2577-087X.

[24] J. K. Gupta, K. Menda, Z. Manchester, and M. Kochenderfer, "Structured Mechanical Models for Robot Learning and Control," in *Learning for Dynamics and Control*. PMLR, July 2020, pp. 328–337.

[25] "Boston Dynamics SpotMini 2018," Feb. 2018. [Online]. Available: https://www.youtube.com/watch?v=fUyU3lKzoio&feature=youtu.be

[26] "Kinova Robotics: Kinova Gen3." [Online]. Available: https://www.kinovarobotics.com/en/products/gen3-robot

[27] "Sake Robotics: EZGripper Gen. 2." [Online]. Available: https://sakerobotics.com/

[28] "boston-dynamics/spot-sdk," Oct. 2020. [Online]. Available: https://github.com/boston-dynamics/spot-sdk

[29] "pybind/pybind11," Sept. 2020. [Online]. Available: https://github.com/pybind/pybind11

[30] Boston Dynamics, Private Communication, 2020.

[31] "Kinovarobotics/kortex," Oct. 2020. [Online]. Available: https://github.com/Kinovarobotics/kortex

[32] "ROBOTIS-GIT/DynamixelSDK," Oct. 2020. [Online]. Available: https://github.com/ROBOTIS-GIT/DynamixelSDK

[33] A. O'Dwyer, "PID control: the early years," 2005, publisher: Technological University Dublin. [Online]. Available: https://arrow.tudublin.ie/engscheleart/86

[34] S. Zimmermann, G. Hakimifard, M. Zamora, R. Poranne, and S. Coros, "A Multi-Level Optimization Framework for Simultaneous Grasping and Motion Planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2966–2972, Apr. 2020.

[35] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, June 2014.

[36] M. Rizon, Y. Haniza, S. Puteh, M. S. Ali Yeon, S. Abdul Rahman, M. Sugisaka, Y. Sazali, M. M. Rozailan, and M. Karthigayan, "Object detection using circular hough transform," 2005, accepted: 2011-03-22T02:32:25Z Publisher: Science Publications.

[37] V. Adetola and M. Guay, "Robust adaptive MPC for constrained uncertain nonlinear systems," *International Journal of Adaptive Control and Signal Processing*, vol. 25, no. 2, pp. 155–167, 2011.