Technische Universität München
TUM School of Computation, Information and Technology

Fast Learning-Based Motion Planning and Task-Oriented Calibration for a Humanoid Robot

Johannes Valentin Tenhumberg

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz : Prof. Dr. Achim Lilienthal
Prüfende der Dissertation:

1. Prof. Dr.-Ing. Darius Burschka
2. Prof. Dr.-Ing. Berthold Bäuml
3. Prof. Dr.-Ing. Udo Frese

Die Dissertation wurde am 06.12.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 07.07.2025 angenommen.

Abstract

Dextrous humanoid robots hold immense potential from manufacturing to healthcare. The capability to move and act autonomously in challenging environments is critical for their successful application. Besides understanding the environment, the robot needs an accurate model of itself to navigate safely and perform tasks accurately. The thesis tackles this interplay of *fast* and *accurate* motion planning for humanoid robots.

The first part focuses on accurately modeling the robot's kinematics, including elasticities and mass distribution, through efficient and self-contained calibration. We introduce a general approach that works for diverse robots and calibration setups. Our theory strictly distinguishes between the actual measurement setup used to collect the data and the robot's intended task, which should be improved through calibration. A typical example is using a camera to measure pixels to improve an end-effector's cartesian accuracy. In the trade-off between a minimal and self-contained measurement setup and the full description of all aspects of the task, this work combines a probabilistic view of the data collection with minimizing the intended task error. The methods are demonstrated for the elastic humanoid Agile Justin using its internal RGB camera and for the DLR Hand-II with its tree-like structure using contact measurement, highlighting the viability on complex hardware and the broad applicability of the approach.

Avoiding self-collision and handling obstacles in diverse environments are critical issues for motion planning. For a complex robot in an unstructured world, motion planning has many local minima of various quality and can have drastically different solutions if the input (i.e., world, start, target) changes only slightly. We use an optimization-based technique for its speed and easy extensibility. However, the cost landscape is challenging for gradient-based techniques, as they operate only locally and, thus, depend entirely on the initialization to avoid local minima. The idea is to mitigate this dependence by using neural networks to predict educated initial guesses, which lie in the area of attraction of the global minimum. We discuss two variants for training such networks. A supervised approach which uses an exhaustive dataset of successful motions in challenging environments, and an unsupervised approach which uses the objective function directly to update the network weights. Crucial for training and the ability to generalize to new environments was to encode the worlds with Basis Point Sets. By extending the work to inverse kinematics, additional insight could be achieved into the network structure in the context of the mode switches, which are fundamental for motion planning.

Overall, the work improved the accuracy and planning time of the humanoid Agile Justin with its 19 degrees of freedom. The maximal error at the end-effector was reduced from 6 cm to 0.8 cm, and the motion planning time in self-acquired high-resolution voxel models from up to 10 s to realtime-capable 0.2 s. Both improvements are significant steps towards more dexterity and autonomy and are meanwhile indispensable for the daily work on the research robot in the lab and at multiple fairs and conferences.

Zusammenfassung

Geschickte humanoide Roboter bergen ein immenses Potenzial, von der Fertigung bis zur Gesundheitsfürsorge. Die Fähigkeit, sich in schwierigen Umgebungen autonom zu bewegen, ist zentral für ihren erfolgreichen Einsatz. Neben dem Verständnis der Umgebung benötigt der Roboter ein genaues Modell seiner selbst, um sicher zu navigieren und Aufgaben präzise auszuführen. Diese Arbeit befasst sich mit dem Zusammenspiel von *schneller* und *genauer* Bewegungsplanung für humanoide Roboter.

Der erste Teil konzentriert sich auf die genaue Modellierung der Kinematik durch eine effiziente und in sich geschlossene Roboterkalibrierung. Es wird ein allgemeiner Ansatz beschrieben, der die Wahl des Robotermodells, die Sensitivitätsanalyse und die optimale Versuchsplanung umfasst. Ein besonderer Schwerpunkt liegt auf der Unterscheidung zwischen dem tatsächlichen Messaufbau, der zur Datenerfassung verwendet wird, und der beabsichtigten Aufgabe des Roboters, die durch die Kalibrierung verbessert werden soll. Diese Arbeit kombiniert dabei eine probabilistische Sicht der Datenerfassung mit der Minimierung des beabsichtigten Aufgabenfehlers. Die Methoden werden anhand des humanoiden Roboters Agile Justin mit seinem elastischen Oberkörper und der DLR Hand-II mit ihrer baumartigen Struktur demonstriert, um die Durchführbarkeit auf komplexer Hardware und die breite Anwendbarkeit des Ansatzes zu verdeutlichen.

Die Vermeidung von Selbstkollisionen und die Bewältigung von Hindernissen in unbekannten und vielfältigen Umgebungen sind zentral für Bewegungsplannung. Für einen komplexen Roboter in einer unstrukturierten Welt gibt es bei der Bewegungsplanung viele lokale Minima, und die Ergebnisse hängen stark von den Randbedingungen ab. Diese Kostenlandschaft stellt eine Herausforderung für optimierungsbasierte Verfahren dar, da sie nur lokal arbeiten und daher auf die Initialisierung angewiesen sind, um lokale Minima zu vermeiden. Die Idee ist, diese Abhängigkeit zu verringern, indem neuronale Netze Anfangswerte vorhersagen, die ein Optimierer schnell zu global optimalen Lösungen iterieren kann. Die Verwendung der Zielfunktion zur generierung von Trainingsdaten und zum Steuerung des Trainings hat sich als entscheidend erwiesen. Darüber hinaus half das Training auf zufälligen Simplex-Welten und die Kodierung dieser Welten mit Basispunktmengen bei der Verallgemeinerung auf neue, nicht gesehene Umgebungen. Durch die Ausweitung der Arbeit auf die inverse Kinematik führte zu zusätzliche Erkenntnisse über die Netzwerkstruktur und den Wechsel zwischen Lösungsmoden.

Insgesamt konnte die Arbeit die Genauigkeit und Planungszeit des humanoiden Agile Justin mit seinen 19 Freiheitsgraden verbessern. Der maximale Fehler am Endeffektor wurde von 6 cm auf 0.8 cm und die Planungszeit in ungesehenen Umgebungen von bis zu 10 s auf 0.2 s reduziert. Beide Verbesserungen sind bedeutende Schritte in Richtung mehr Geschicklichkeit und Autonomie und sind mittlerweile ein unverzichtbarer Bestandteil der aktuellen Arbeit am Forschungsroboter.

Preface

This cumulative thesis is based on five peer-reviewed core publications [1; 2; 3; 4; 5] and deals with two key topics for autonomous robot motion:

- **Robot calibration** [1; 2; 3] in Chapter 2. This chapter derives a theoretical framework to tackle self-contained and task-oriented calibration on a wide range of robotic systems by combining a probabilistic view on data collection with the goal of improving a desired task. It is submitted as a advanced journal paper [TBB24] to IEEE Transactions on Robotics and currently under review.
- **Robot motion planning** [4; 5] in Chapter 3. This chapter analyzes supervised and unsupervised learning techniques to speed up optimization-based motion planning and inverse kinematics in challenging unseen environments.

Most publications include accompanying videos in order to visualize the results and especially to showcase their application on actual hardware. Readers who prefer a visual overview of the topics, please refer to the respective video links [3], [4], [5], and [Ten+24]. The written links and all further publication details are listed in the bibliography section.

Contents

1 Introduction

Potential of Humanoid Robots Two prominent applications of humanoid robots are in manufacturing and healthcare. A robot capable of human-like dexterity can mitigate the industry's labor shortage and open new possibilities to design and manufacture products. Moreover, a humanoid robot that can autonomously act in environments as unstructured as people's homes can assist people to live a self-determined life longer. As a step towards these possibilities, the thesis focuses on the interplay of two building blocks for the successful applications of humanoids. First, the robot needs an accurate model of itself. Second, this model must be combined with understanding the world to move and act safely and accurately in unknown and challenging environments and thus enable *fast* and *accurate* motion planning for humanoid robots.

Application on DLR's Agile Justin While the developed methods are general and apply to various robotic systems, DLR's humanoid Agile Justin (see Fig. 1.1) is the benchmark for validating and testing the discussed methods on actual hardware. This robot is a complex mechatronic system with 53 degrees of freedom (DoF). It drives on four wheels and has an upper body with a rope mechanism and a passive joint to redirect torques and keep the chest upright. Agile Justin has two arms. Each consists of a lightweight robot arm (LWR III, 7 DoF) and a DLR-Hand II (12 DoF), which enables it to perform a wide range of dextrous manipulation tasks. In its actuated head is a Kinect camera with RGB and depth sensors, which the robot uses to model its environment. Those self-acquired high-resolution voxel models are the basis for collision-free motion planning of the humanoid. This setting of a complex elastic humanoid robot with many DoF and the self-acquired high-resolution voxel models of the environment provides the context and the application for calibration and motion planning in this thesis.

Robot Calibration As robotic systems become more autonomous, the necessity for an efficient and accurate calibration of the mechatronic system itself is evident. When the robot's operations are not limited to preprogrammed and taught configurations, an accurate system model throughout the whole configuration and task space is crucial and a prerequisite for safe and precise movements. Besides providing an accurate model of the physical robot, calibration should be practical, easy to apply, and designed for repeatability. If the robot uses its internal sensors to gather measurements, it reduces the dependence on external equipment and extends the calibration chain to those sensors. This self-reference makes the calibration more holistic and brings it closer to the final application of the system. When designing a calibration, one must consider the requirements of the robotic task that needs improvement. However, this does not always

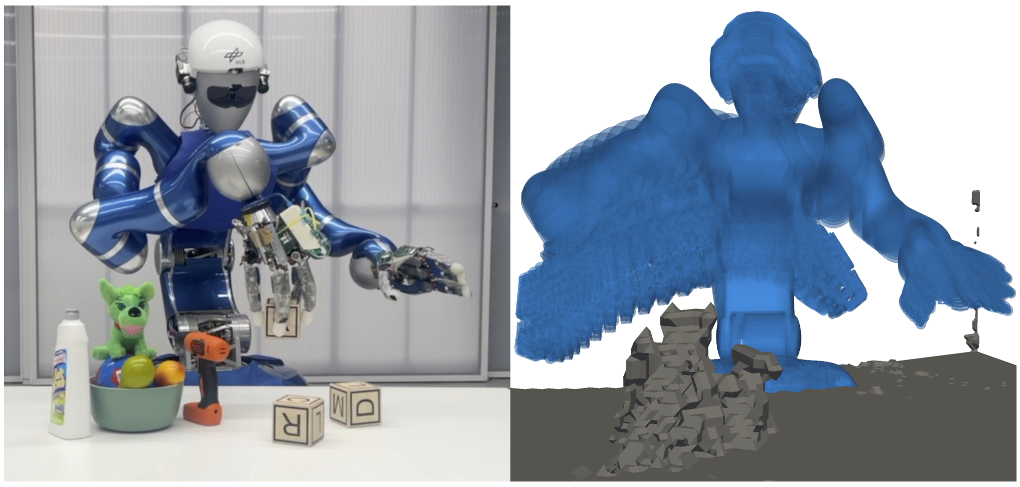


Figure 1.1: Agile Justin is spelling DLR by building a tower out of lettered cubes. Besides the dextrous in-hand manipulation, an accurately calibrated robot (see Chapter 2) and fast motion planning (see Chapter 3) is crucial to perform this combined task. See also the video [Ten+24] for the full experiment.

align with the constraints derived from the measurement setup. Robot calibration must balance accessible measurement collection with utilizing their information optimally to achieve high final accuracy in the task space.

With a sound Bayesian formulation, this work analyzed practical and theoretical aspects of the calibration problem. The task-oriented approach enabled the calibration of the humanoid with all geometric parameters and elasticities using only the internal RGB camera and the contact-based calibration of the robotic DLR-Hand II. The errors could be significantly reduced in both cases and so enable safe and dextrous motions.

Motion Planning The central problem is that robots still have to start planning from scratch for every new task. They have no memory of previous tasks, neither successful nor failed ones. Humans have an image of motion in their mind, a sketch of the necessary actions to reach the goal since they have done this motion or a similar one many times before. The idea is to enable robots to draw from experience, to help in similar as well as in unknown situations and so make its motions faster and more efficient.

To achieve this, this work combined optimization-based motion planning (OMP) with warm starts from neural networks. OMP can handle problems with many parameters, produce smooth motions, and elegantly incorporate additional terms into the objective function. The downside of this approach is that the optimization is always only a local corrective to an initial guess. Therefore, the overall convergence and feasibility depend on the initial guess and are susceptible to local minima. This work moved away from brute force and random multistarts towards more educated and experience-based initial guesses in order to mitigate those drawbacks. Neural networks trained using modern deep-learning techniques significantly reduced the planning speed and enabled fast motion planning in challenging and unknown environments.

2 Task-Oriented Robot Calibration

2.1 Introduction

Robotic systems are continually evolving, becoming more capable and autonomous. Integrating and fusing multiple sensors allows for skillful behavior in diverse settings. As the capabilities rise and robots can perform more complex tasks autonomously, the underlying robot model must also provide accuracy far away from pre-taught poses and default motions. Therefore, an accurate model of the robot and its sensors remains the basis for reliability and dexterity.

There are many factors to consider when calibrating such a robot model. Foremost is the relation to the task that needs improvement or is failing because of an inaccurate kinematic. This task should guide the design of the measurement setup, the sample selection, and the identification of the model parameters. Besides, an easy application, repeatability, and autonomous data collection with minimal external requirements are relevant practical aspects of robot calibration. Extensive measurement setups can readily provide all relevant information for the task but often restrict the easy applicability. This work explores the intersection between task-oriented and practical calibration. Central is the distinction between the function to quantify the task accuracy and the actual measurement setup used to collect samples on the hardware. A typical example is a camera-based calibration that measures pixels that should increase the cartesian accuracy of an end-effector. The goal is to provide a general framework for calibrating robots with different kinematic structures and measurement setups while always focusing on the final robotic task.

Contributions

This work unifies and extends the concepts from three preceding papers, which showed the non-geometric calibration of a humanoid robot kinematics with an external tracking system [1] and an internal RGB camera [2] and the contact-based calibration of a robotic hand [3]. In detail, we present:

- A unified robot calibration, with the forward kinematics as central mapping, over the choice of the measurement setup, sensitivity analysis, the Bayesian formulation of the identification problem, and optimal sample selection
- Formalization of the distinction between the actual calibration and the robot task, with different measurement functions and different sets of robot joint configurations

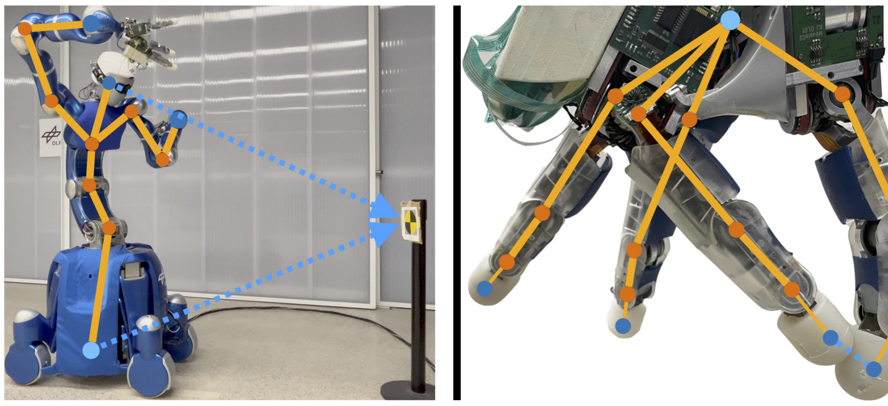


Figure 2.1: Calibration of two complex mechatronical systems; *left*: vison-based calibration of the humanoid Agile Justin [Bäu+14] with internal RGB camera. *right*: contact-based calibration of the DLR-Hand II [J B01].

- Generalization of the concept of sensitivity analysis and OED to mitigate the problems arising from those differences in measurement and task sets and functions [3]
- Introduction of task-oriented weighting (TOW) to minimize the remaining systematic errors in the task space
- Extensive studies in simulation on synthetic data and examples on actual hardware in different challenging settings [1; 2; 3] to analyze and validate the methods

A significant extension of this work is the derivation of TOW. It allows us to combine the full probabilistic model to account for measurement uncertainties with an approximation to a true task function and minimize the remaining systematic errors in the desired space. Fig. 2.2 shows this combined approach's different aspects and acts as a visual table of contents for Section 2.3. Our TOW approach is universal and can be applied to arbitrary calibration and measurement setups (see the Bayesian graphs in Fig. 2.7 and Fig. 2.20).

2.2 Related Work

Roth et al. [RMR87] outline four critical aspects of robot calibration: the robot model, the dataset, the identification of the model parameters, and the compensation [VW88] of this robot model. They also distinguish between identifying geometric parameters and non-geometric calibration, including, for example, joint elasticities and gear backlash. Hollerbach et al. [HW96] also provide a taxonomy and a calibration index based on the motion equation. They emphasize the distinction between open-loop and closed-loop calibration. Open-loop calibration requires external measurement systems to track robot positions or markers in the workspace, while closed-loop calibration relies solely on internal sensors.

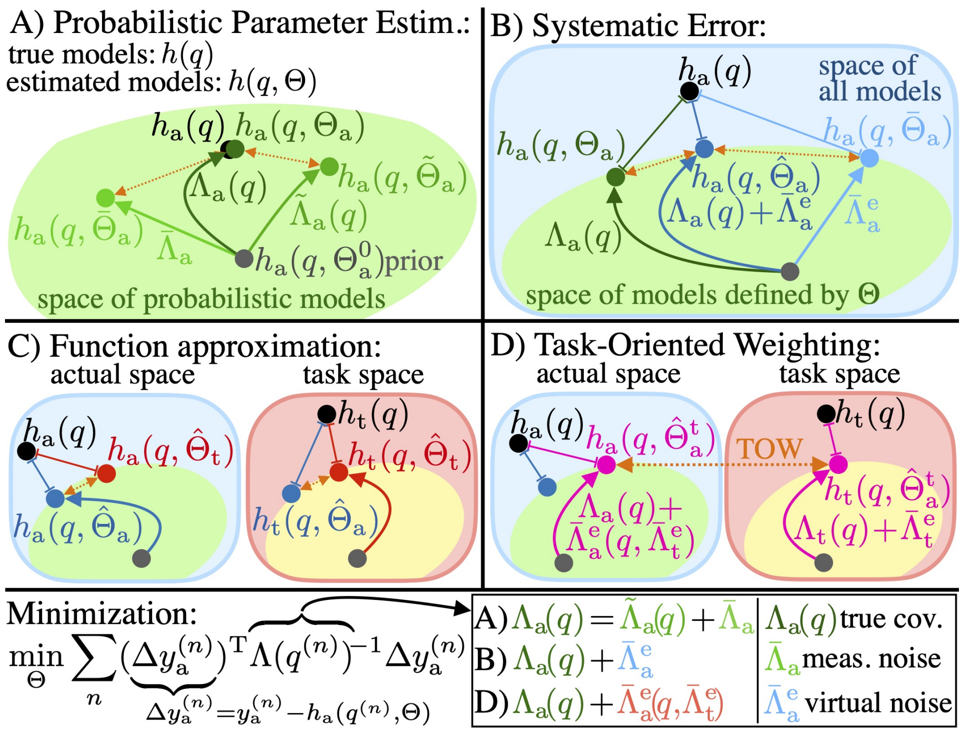

A) Probabilistic Parameter Estim.:
true models: $h(q)$
estimated models: $h(q,\Theta)$
$h_a(q)$
$h_a(q,\Theta_a)$
$h_a(q,\tilde{\Theta}_a)$
$h_a(q,\bar{\Theta}_a)$
$\Lambda_a(q)$
$\tilde{\Lambda}_a(q)$
$\bar{\Lambda}_a$
$h_a(q,\Theta_a^0)$prior
space of probabilistic models
B) Systematic Error:
$h_a(q)$
space of all models
$h_a(q,\Theta_a)$
$h_a(q,\bar{\Theta}_a)$
$h_a(q,\hat{\Theta}_a)$
$\Lambda_a(q)+\bar{\Lambda}_a^e$
$\bar{\Lambda}_a^e$
$\Lambda_a(q)$
space of models defined by Θ
C) Function approximation:
actual space
task space
$h_a(q)$
$h_a(q,\hat{\Theta}_t)$
$h_a(q,\hat{\Theta}_a)$
$h_t(q)$
$h_t(q,\hat{\Theta}_t)$
$h_t(q,\hat{\Theta}_a)$
D) Task-Oriented Weighting:
actual space
task space
$h_a(q)$
$h_a(q,\hat{\Theta}_a^t)$
TOW
$h_t(q)$
$h_t(q,\hat{\Theta}_a^t)$
$\Lambda_a(q)+\bar{\Lambda}_a^e(q,\bar{\Lambda}_t^e)$
$\Lambda_t(q)+\bar{\Lambda}_t^e$
Minimization:
$\min_\Theta \sum_n \underbrace{(\Delta y_a^{(n)})}_{\Delta y_a^{(n)}=y_a^{(n)}-h_a(q^{(n)},\Theta)}{}^T \overbrace{\Lambda(q^{(n)})}^{}{}^{-1}\Delta y_a^{(n)}$
A) $\Lambda_a(q)=\tilde{\Lambda}_a(q)+\bar{\Lambda}_a$
B) $\Lambda_a(q)+\bar{\Lambda}_a^e$
D) $\Lambda_a(q)+\bar{\Lambda}_a^e(q,\bar{\Lambda}_t^e)$
$\Lambda_a(q)$ true cov.
$\bar{\Lambda}_a$ meas. noise
$\bar{\Lambda}_a^e$ virtual noise

Figure 2.2: *A:* Probabilistic view of the calibration problem for three different probabilistic models. Only when the modeled uncertainty matches the true process and each sample is weighted appropriately does the calibration converge correctly. *B:* If the model is imperfect, the systematic error can be modeled by adding virtual noise $\bar{\Lambda}_a^e$. Again, the combined covariance has to match the true data generation process to get a close approximation of the true function outside the model's scope. *C:* When approximating a function outside the model, the space in which we minimize matters. The respective solutions for the parameters Θ are not necessarily good fits for the other measurement space. *D:* The full probabilistic model for h_a is combined with minimizing the systematic error in the task space h_t.

Open-loop Geometric Calibration

As accurate forward kinematics is relevant for most robotic applications, many examples of successful calibrations exist. Often, a robotic arm's geometric model is calibrated with an external tracking system [PD90; GM11; PK11; Xio+17; WFC19]. Many variations exist; for example, a static camera was used to track the reflections of line lasers on metallic spheres [YCX18]. For a robotic hand, a visual approach with an external tracking system and individual markers on each finger was demonstrated by Lee et al. [Lee+13] . Using an electromagnetic tracking system to measure the position and orientation of the fingertips of a soft robotic hand is less prone to occlusions [TGR18]. However, it is still an open-loop calibration, requiring additional equipment.

Open-loop Non-Geometric Calibration

Industrial robots often achieve high accuracy through pure geometric calibration. However, in scenarios involving robots constructed from lightweight components, a purely geometric model may fall short, particularly when elasticities in the joints become a significant source of non-geometric errors. Joint torque sensors can address this issue, as demonstrated for the MIRO robot arm [Klo+11] and an LWR arm [BOG16]. This approach incorporates sensor measurements into the kinematic model and is only suitable for control rather than planning, as it relies on real-time sensor reading. Furthermore, it cannot account for additional effects like lateral elasticities.

An alternative method for handling elasticities involves identifying a mass model of the robot and calculating torques at static equilibrium for each pose. Caenen et al. [CA90] presented a technique for integrating torques into the DH-formalism by introducing torque-dependent linear offsets to the rotational DH parameters. For relatively rigid robots, this linearization works well and can be directly applied [KB19].

While often ignored, the iterative search for the torque equilibrium [Lee13; ZNK14b] is highly relevant for robots with larger elasticities. For very flexible robots, more is needed than including joint elasticities, and one also must account for lateral elasticities or even the mechanical bending of the individual links [KB02]. One can also use a neural network to model the non-geometric errors [NLK19]. While this might not be relevant for simple joint elasticities, this approach can model general unknown effects. However, a challenge for such complex robot models is their efficient compensation.

Closed-loop Visual Calibration

The calibration model should not only accurately represent the true robot, but the calibration process should also be fast and easy, ensuring broad applicability and simple repetition if necessary. In an ideal scenario, the robot employs its internal sensors for self-calibration, ensuring that the same kinematic chain used for performing tasks is used for the calibration process [Ma96]. Hubert et al. [HSB12] demonstrated this by incorporating a Bayesian approach and conducting hand-eye calibration on an anthropomorphic robot using a checkerboard marker. For the humanoid Nao, the joint offsets were calibrated using the robot's internal RGB camera to track the position of four checkerboard markers on its hands and feet [MWB15].

Closed-loop Geometric Calibration

Another calibration approach, sidestepping the need for a camera system, leverages geometric constraints on the kinematic chain. Early work constrained the end-effector motion on a plane to calibrate a robotic arm [IH97]. Bennett et al. [BH91] applied this idea to a robotic hand, calibrating the Utah-MIT hand through the rigid connection of two fingertips with a plate, resulting in a closed-loop kinematic chain. Subsequent works used mechanical fixtures [MD00; Wu+15], relative calibration techniques [SGK09], and precise reference plates [JB15] for the calibration of robotic arms without a vision system.

A specific case of geometric calibration relies on self-contact. Multiple works on this topic focus on the humanoid robot iCub. Roncone et al. [Ron+14] calibrated its DH parameters using self-touch with a tactile skin containing 4200 sensing points. Furthermore, there was a simulation study [SPH19] and experiments on the hardware combining and comparing multiple sensor modes [Ste+22]. Self-contact can also be used for incremental updates to the kinematic scheme [Zen+18] or identifying the layout of an artificial tactile skin [Rus+21].

Optimal Experimental Design

Data collection in robotics is often time-consuming, and one wants to rely on a minimal number of measurements. In addition, the final accuracy often strongly depends on the specific choice of measurement configurations in the dataset. Two key words here are the observability index [HW96] and the optimality alphabet [SH08] from the literature of optimal experimental design (OED). Those terms describe measures of the information in a given sample or a set of samples. The underlying idea is to choose configurations in which the effect of the parameters on the measurement is significant compared to the measurement noise [DP90; HXZ08]. The Noise Amplification Index [NH96] is often less sensitive to measurement noise than the classical optimality alphabet, as demonstrated for the combined calibration of kinematic parameters and the joint stiffness of a robotic arm [ZKR10]. Which of those different optimality measures is the best choice depends on the robotic task [Dan02; Xio+17; KB19; Jia+20], but they are often closely correlated.

A good calibration set makes the identification of those parameters more accessible and robust. However, finding an optimal set is an enormous combinatorial problem, which one can generally only approach approximately using heuristics. Only for the small-scale calibration sets the combinatorial problem can be solved exhaustively [Car+13]. Besides a greedy search, DETMAX [ZNK14a; YCX18] is a common heuristic that iteratively swaps samples starting from an initial guess. Furthermore, genetic algorithms [ZWH97], tabu search [DPM05], and simulated annealing [LW08] were applied to this problem.

Finally, the calibration goal is to improve the robot's performance on a specific task. Therefore, the performance on a calibration set is often the wrong metric. To bridge this mismatch, one needs to consider a task; either only containing a single pose [Kli+12] or a set of relevant task poses [Car+13]. With those task sets, one can compute the desired optimality measures and use them as guidance for constructing a respective calibration set. One can go even further and design the calibration so that the production line in which the robot will be applied is more efficient, which differs from a high cartesian accuracy [LB90].

Parameter Identifaction

Another aspect of robot calibration is to identify the optimal values of the model parameters. Li et al. [LLL21] compare basic least-squares, least-squares with regularization, Levenberg-Marquart (LM), particle filter algorithms, and Bayesian Maximum a posteriori (MAP). They show that the optimal algorithms strongly depend on the type of calibration problem. The consecutive idea is to use a hybrid approach to the parameter identification [Luo+21]. The authors could increase the final accuracy by combining a first optimization step with LM followed by differential evolution. Tohme et al. [TVY20] provides a general Bayesian viewpoint on model calibration with a focus on model evidence. One crucial goal they achieved was that the calibrated model's declared confidence would accurately match the actual measurement data, allowing for safe and educated use.

Bundle Adjustment

A related problem in computer vision is bundle adjustment [Tri+00]. It tackles the simultaneous localization of a scene geometry and camera calibration from images. A pivotal difference to robotic calibration is that the individual measurements are not tightly coupled through a robot kinematic. In its most basic form, only the bundles of light radiating from the different camera positions onto landmarks of the scene are adjusted. This setup with only relative frames differs from robot calibration as no forward model couples the measurements. Without a system, systematic errors are not relevant. However, the optimization problem can become large [Aga+10] and is often split up and tackled iteratively [Ila+17]. There are known ways to include uncertainties of the measurements in the calibration and propagate errors from previous measurements [EL09; Eud+10; Sib+09] to new sets of images. In general, one uses the design matrix, which is the Jacobian of the forward model, to map the uncertainties of the measurements into the parameter space and weigh the different samples and types of measurements.

This idea was also applied to a robotic context. To combine multiple types of measurements into a single calibration, Pradeep et al. [PKB14] projected the measurement uncertainties back to the parameter uncertainties using the Jacobians of the measurement functions for the linearized mapping.

Work and Applications on Agile Justin

The humanoid 19 DoF Agile Justin [Bäu+14] together with its 12 DoF DLR-Hand II [J B01] are complex mechatronic systems built from lightweight and custom components (see Fig. 2.1). In earlier work, the head-mounted cameras, the hand-eye chain, and the IMUs in the head and the base were calibrated [Car+13; BB14; BFB15]. The applications range from moving accurately and safely [4; 5] through its self-acquired voxel models [WFB13] for the upper body to dextrous manipulation [SPB22; Pit+23] and grasping [Win+22; Hum+23] for its DLR-Hands II. We validated our proposed methods on these two distinct systems.

Table 2.1: Calibration Nomenclature

actual meas. fun.	$h_\mathrm{a}(q,\Theta)$	describes the meas. process
task meas. fun.	$h_\mathrm{t}(q,\Theta)$	describes the desired task
true meas. fun.	$h(q)$	true process; approximation goal
systematic error	Δh	arises because model can not fully represent reality
virtual noise	$\bar{\Lambda}^\mathrm{e}$	additional uncertainty to model systematic error

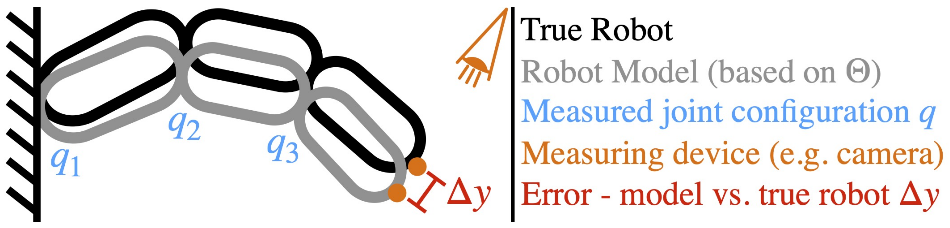

q_1
q_2
q_3
Δy
True Robot
Robot Model (based on Θ)
Measured joint configuration q
Measuring device (e.g. camera)
Error - model vs. true robot Δy

Figure 2.3: The concept robot calibration: By measuring the actual robot and comparing its current model with these measurements, one can adapt the model parameters to enable a good fit between the measured and modeled robot.

2.3 Bayesian Calibration with Task-Oriented Weighting

The central equation in robotics is the forward kinematics $f(q,\Theta_\mathrm{f}) = F$ (see Section 2.5.1). It maps from the generalized joint angles q to the robot's physical pose in the cartesian workspace F. Everything from path planning and checking for obstacle collision or self-collision to grasping depends strongly on an accurate model of the robot's kinematics. The calibration goal is to find parameters Θ_f that accurately model the true robot kinematics in the task context.

Fig. 2.3 shows the idea behind calibration. The true robotic arm is measured and compared to the robot model. Identifying the model parameters means finding a set that minimizes the error between the measured and modeled robot. How the measurement setup looks in detail depends on the robot, the available measurement hardware, and the application. Fig. 2.4 gives an overview of different measurement setups, which are described in Section 2.5.4 in more detail. In general, we distinguish between two measurement functions. The task measurement function h_t describes the robotic task and captures all relevant of the desired application. The actual measurement function h_a describes the data collection and can be applied on the hardware.

2.3.1 Maximum a Posteriori Estimation

While the central parameters of the forward model are the DH parameters Θ_f, often, one must estimate additional parameters jointly with the kinematic parameters. Examples are camera intrinsics or additional frames to close the measurement loop. We denote the combination of all calibration parameters Θ. These parameters can be identified using a dataset $S = \{(q^{(n)}, y^{(n)})\}_{n=1}^{N_\mathrm{S}}$ of corresponding pairs between the robot's config-

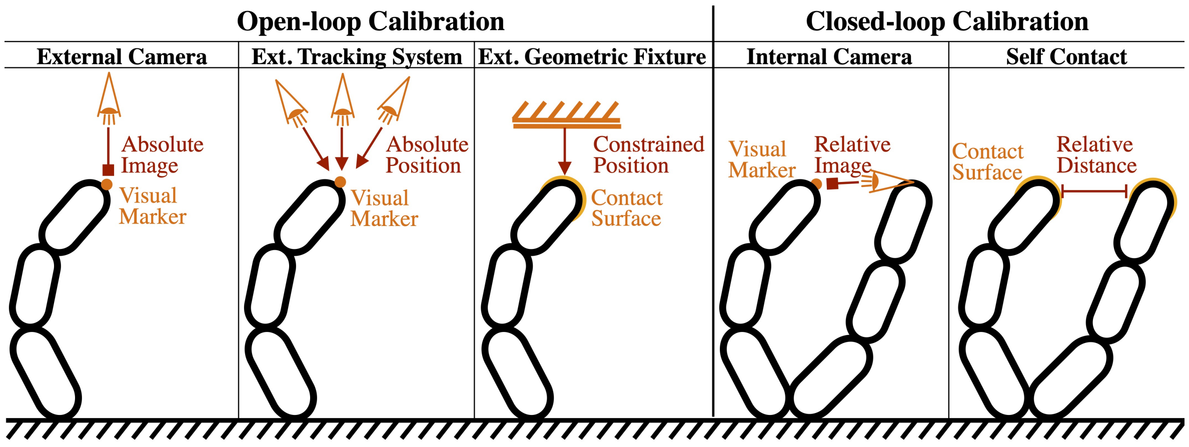


Figure 2.4: Sketches of different open and closed-loop setups, showing the required hardware and the type of measurement. See Section 2.5.4 for details.

uration $q^{(n)}$ and the measurement $y^{(n)}$. Those q and y are at the opposite ends of the measurement function h and are therefore needed to identify it

$$y = h(q, \Theta) = h(f(q, \Theta_\mathrm{f}), \Theta). \tag{2.1}$$

The calibration goal is function approximation, that is, to find a set of parameters Θ^* that minimizes the distance from the model $h(q, \Theta)$ to the true function $h(q)$

$$\Theta^* = \arg\min_\Theta \|h(q) - h(q, \Theta)\|. \tag{2.2}$$

In a fully deterministic setting or for a sufficiently large dataset, the best one can do is to use least squares (LS)

$$\arg\min_\Theta \sum_n (\Delta y^{(n)})^\mathrm{T} (\Delta y^{(n)}) \tag{2.3}$$

$$\text{with } \Delta y^{(n)} = y^{(n)} - h(q^{(n)}, \Theta). \tag{2.4}$$

However, suppose the data collection is noisy, and one wants to optimally use the information of a limited number of uncertain measurements. In that case, it is better to formulate the problem as a probabilistic estimation problem [Bis06]. The idea is to find the maximum of the posterior distribution $p(\Theta|S)$ given the measurements S. The maximum a posteriori (MAP) results in a weighted least squares problem

$$\begin{aligned} &\arg\min_\Theta \sum_n \log(p(y^{(n)}|q^{(n)}, \Theta)) + \log(p(\Theta)) \\ = &\arg\min_\Theta \sum_n (\Delta y^{(n)})^\mathrm{T} \Lambda_\mathrm{y}^{-1} \Delta y^{(n)} + \Delta\Theta^\mathrm{T} \Lambda_\Theta^{-1} \Delta\Theta \\ &\text{with } \Delta\Theta = \Theta_0 - \Theta,\ \Lambda_\Theta = \mathrm{Diag}\,(\sigma_\Theta^2). \end{aligned} \tag{2.5}$$

The central covariance Λ_y describes the uncertainty of the measurements and weighs the samples accordingly. The measurement noise is often modeled as uniform Gaussian $\mathcal{N}(0, \Lambda_\mathrm{y})$ with zero mean and diagonal variance, which is just added to the deterministic forward model. However, the uncertainty model should capture the real process, and depending on the setting, the covariance Λ_y can be more complex and, in general, q-dependent. Furthermore, this approach includes the initial uncertainty in the parameters with a Gaussian distribution around the nominal parameters Θ_0 and a diagonal covariance Λ_Θ. Incorporating this prior regularizes the method and ensures a minimum exists, even if there are redundancies in the robot or measurement model.

This probabilistic view allows us to weigh samples according to their certainty and rely more on measurements with low expected noise and trust measurements with high noise levels less. Standard MAP finds the model parameters Θ, which makes the observed data S the most probable. LS can be viewed as a special case of MAP with uniform measurement covariance Λ_y and infinite large prior Λ_Θ.

2.3.2 Modeling Systematic Error with Virtual Noise

The probabilistic framework, which incorporates the uncertainty of the measurements into the estimation, is robust. However, in general, MAP does not result in a model that minimizes the distance to the true function (2.2) as LS. It gives optimal fits only if the calibration model can fit the truth well (or the uncertainty is uniform). The problem is that one often can not capture all relevant effects, mainly when dealing with complex systems or relying on a simple model for easy applicability of the calibrated model. In such a setting, an error remains between the model and the truth even for the optimal set of parameters Θ. We denote this remaining mismatch of the model systematic error

$$\Delta h = \min_{\Theta} \|h(q) - h(q, \Theta)\|. \tag{2.6}$$

It stems from the limited expressiveness of the model and can not be further reduced by collecting more measurements.

When ignoring the systematic error, MAP overestimates the certainty of samples and tries to fit the model too close to measurements it believes certain, leading to larger errors for uncertain regions. Because the model can not reproduce the measurements everywhere, this leads to an overall suboptimal fit. To tackle this, the systematic error is modeled as additional virtual uncertainty $\bar{\Lambda}_\mathrm{a}^\mathrm{e}$ in the measurements

$$\hat{\Lambda}_\mathrm{a}(q) = \underbrace{\Lambda_\mathrm{a}(q)}_{\tilde{\Lambda}_\mathrm{a}(q) + \bar{\Lambda}_\mathrm{a}} + \bar{\Lambda}_\mathrm{a}^\mathrm{e}. \tag{2.7}$$

It is virtual as it only models the gap to the true function and does not originate from a noisy measurement process.

Fig. 2.5 demonstrates the idea. The true function $h_\mathrm{a}(q) = q^3$ is cubic, and the measurement process has a quadratic covariance $\Lambda_\mathrm{a}(q) = q^2 + 0.01$ with high certainty in the center and less certainty towards the edges. We use this setup to generate measurement

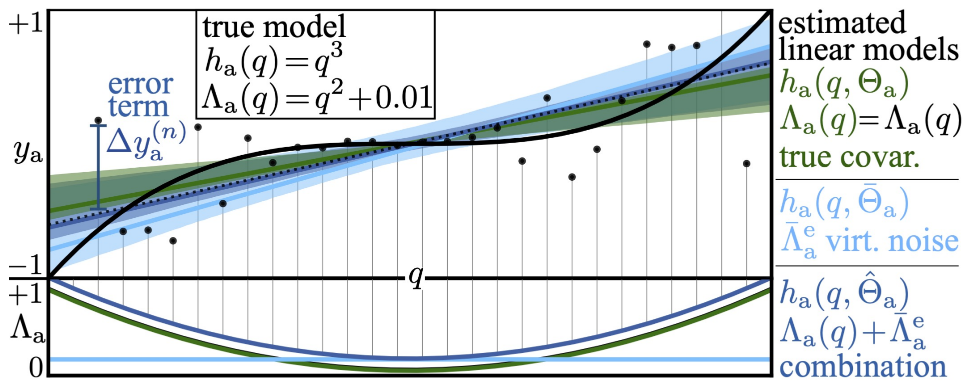


Figure 2.5: Detailed view of panel B in Fig. 2.2. Three linear models with different covariances try to fit the true model using MAP. The inverse covariance matrix acts as weighting for the error terms. Samples with higher covariance are less certain and get weighted less in the minimization (2.5).

data, and we want to approximate the true function with a linear model using MAP and compare three different covariances to weigh the measurements.

The model using the same covariance as the true process (green) neglects that it can not fully represent the true model and, therefore, weights the measurements in the center too heavily, ignoring the edges. The model with a constant covariance (light blue) does not account for the different noise levels of the samples, emphasizing the noisy edges too heavily and making the prediction in this example too steep. This behavior shows the downside of pure LS. Because it weighs all samples uniformly, ignoring their uncertainty, it puts too much emphasis on samples with larger noise at the cost of valuable information in the more certain measurements. For a low number of measurements, this leads to a high variance in the estimated parameters between different rollouts of the data collection. When the true variance and the virtual noise are correctly combined (dark blue), the linear model optimally fits the true function from the noisy measurements. In contrast to the green model, here, a small mismatch for the measurements with high certainty is allowed if this leads to a better global fit of the true function. This approach balances the probabilistic viewpoint with the goal of approximating the mean value of a true function.

Fig. 2.6 shows in the top how the distance to the true function goes down for a higher number of measurements N_S. One can see that the model with the true covariance (green) leads to reasonable estimates for a small number of samples where the different levels of measurement noise have a significant impact but ultimately converges to a suboptimal solution. Pure LS (constant covariance, light blue) converges to the best linear fit (black) with enough data points. Its downside, however, is that it does not efficiently use the information in the noisy measurements, which leads to wrong predictions for small datasets. The dotted lines are variations of the combined covariance $\hat{\Lambda}_a$ with different magnitudes of the virtual noise. All three models exhibit the desired convergence behavior. They use the information about different levels of uncertainty for a smaller number of samples. With more data points, they become closer to pure LS

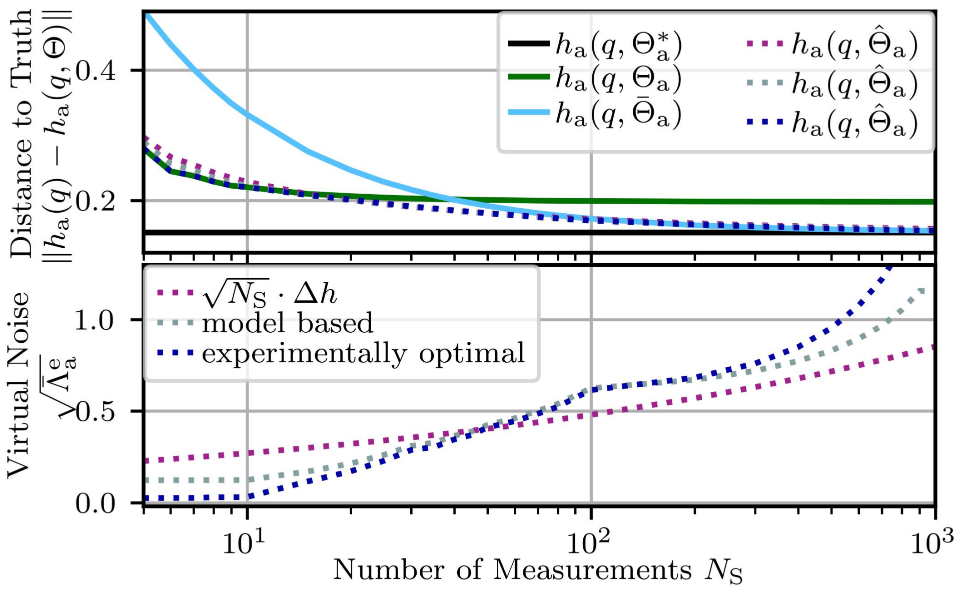


Figure 2.6: Analysis of the example from Fig. 2.5 over different number of measurements. *Top:* Three ways to determine the level of the virtual noise $\bar{\Lambda}_a^e$. *Bottom:* Distance of the different models to the true function as mean over random 10000 rollouts for each dataset size.

and ultimately converge to the optimal fit to the true function. This shows that the additional virtual noise can account for the systematic error in the probabilistic context.

To achieve the wanted behavior, the magnitude of the virtual noise matters (see Fig. 2.6, bottom). While the systematic error Δh for a given model is constant, its description as virtual measurement noise depends on the number of samples. The additional noise is virtual and is only introduced to model the systematic error. Therefore, its effect should not be reduced with more samples like it is the case for measurement errors, which diminish with $\sqrt{N_S}$. Even for a vast dataset, the model can not be more "certain" than the systematic error Δh. Therefore, the additional virtual noise should be chosen so that after calibration, the variance in the predicted measurements is larger by exactly this margin. This condition can be expressed as a difference in variances

$$\begin{aligned} &\mathrm{E}_\Theta\left[\mathrm{E}_{Q,Y}\left[\mathrm{E}_q\left[\Delta V\right]\right]\right] \overset{!}{=} (\Delta h)^2 \qquad (2.8)\\ \text{with } &\Delta V = \mathrm{Var}_{y\sim\hat{p}(y|q,\hat{\Theta}^*)}[y] - \mathrm{Var}_{y\sim p(y|q,\Theta^*)}[y]\\ &\Theta^*(Q,Y) = \arg\max_\Theta p(Y|Q,\Theta)\\ &\hat{\Theta}^*(Q,Y) = \arg\max_\Theta \hat{p}(Y|Q,\Theta)\\ &\text{with } \hat{p}(y|x,\Theta) = [N(\_|0,\bar{\Lambda}^e) * p(\_|x,\Theta)]_y. \end{aligned}$$

It can be computed by first sampling different sets of parameters from the prior distribution to generate an ensemble of datasets. Those measurements can then be used to calibrate two models, one normal and one with the added virtual noise. Finally, one can compare the uncertainty in the predicted measurements and choose $\bar{\Lambda}_{\mathrm{a}}^{\mathrm{e}}$ that the difference in the variance of the models is equal to the systematic error $(\Delta h)^2$. As Fig. 2.6 shows, $\bar{\Lambda}_{\mathrm{a}}^{\mathrm{e}}$ can also be approximated by $(\Delta h)^2\sqrt{N_{\mathrm{S}}}$, which matches the intuition that the systematic error modeled as measurement noise should persist even for large N_{S}. Alternatively, $\bar{\Lambda}_{\mathrm{a}}^{\mathrm{e}}$ can be determined experimentally with a validation set.

2.3.3 Task-Oriented Weighting

While MAP with virtual noise combines the probabilistic model with function approximation, it still minimizes the remaining errors in the actual measurement space Δy_{a}. However, when performing function approximation, the space in which we minimize matters. We aim to find a set of parameters Θ_{t}^{*} that approximates the task measurement function h_{t} well in the presence of systematic errors. Therefore, we extend the concept of virtual noise to the task space.

To achieve this methodologically, we start from the standard posterior for y_{a} and include the relation to the task measurements by expanding and marginalizing over y_{t}

$$p(y_{\mathrm{a}}|q,\Theta) = \int p(y_{\mathrm{a}}|y_{\mathrm{t}},q,\Theta)\, p(y_{\mathrm{t}}|q,\Theta) dy_{\mathrm{t}}. \tag{2.9}$$

To minimize in the task space we alter $p(y_{\mathrm{t}}|q,\Theta)$ and add a virtual noise $\bar{\Lambda}_{\mathrm{t}}^{\mathrm{e}}$

$$\hat{p}(y_{\mathrm{t}}|q,\Theta) = \mathcal{N}(y_{\mathrm{t}}|h_{\mathrm{t}}(q,\Theta), \Lambda_{\mathrm{t}}(q) + \bar{\Lambda}_{\mathrm{t}}^{\mathrm{e}}). \tag{2.10}$$

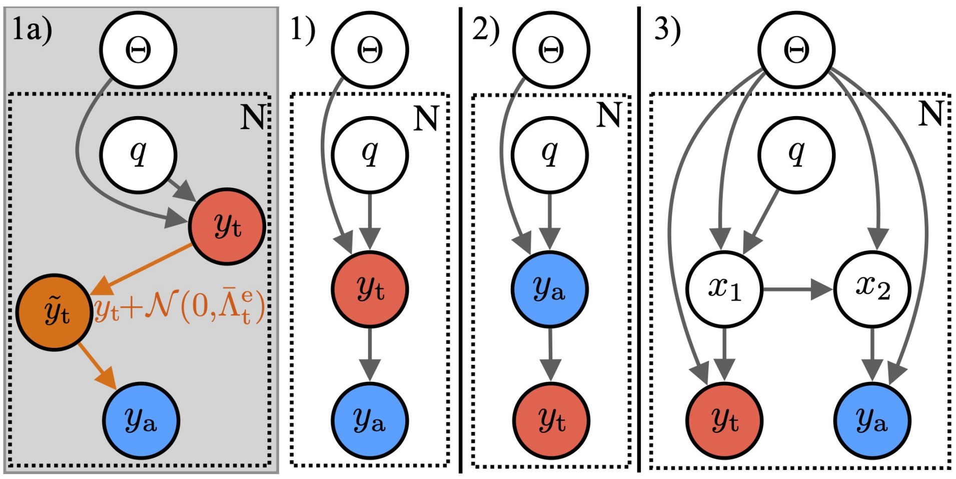
As for the virtual noise in the actual space, this modification happens outside the probabilistic framework and allows us to fit the calibration model defined through the parameters Θ closer to the true function. In this case the approximation is towards $p(y_{\mathrm{t}}|S_{\mathrm{a}})$ instead of $p(y_{\mathrm{a}}|S_{\mathrm{a}})$. To perform the marginalization over y_{t} for the altered version of (2.9), we need to compute $p(y_{\mathrm{a}}|y_{\mathrm{t}},q,\Theta)$. Using the graph of the linear Gaussian system (see (2.14)), we can compute the covariance $\Lambda_{\mathrm{a|t}}$ of the conditional probability. Marginilizing over y_{t} results in the altered posterior distribution $\hat{p}(y_{\mathrm{a}}|q,\Theta) = \mathcal{N}(y_{\mathrm{a}}|h_{\mathrm{a}}(q,\Theta), \hat{\Lambda}_{\mathrm{a}})$ with

$$\hat{\Lambda}_{\mathrm{a}} = \Lambda_{\mathrm{a}} + \Lambda_{\mathrm{at}}\Lambda_{\mathrm{t}}^{-1}\bar{\Lambda}_{\mathrm{t}}^{\mathrm{e}}\Lambda_{\mathrm{t}}^{-1}\Lambda_{\mathrm{ta}}. \tag{2.11}$$

The adjusted covariance is the only change necessary to the standard MAP approach in (2.5) and allows us to combine the full probabilistic model of h_{a} with the function approximation towards h_{t} in a general manner.

Probabilistic Bayesian Graphs

We use Probabilistic Bayesian graphs to derive (2.11) and compute its components. Probabilistic graphs can describe arbitrary measurement setups, and Fig. 2.7 shows three different graphical models of calibration processes with various relations between the nodes. The stochastic variables of the actual measurement y_{a} can be expressed as a function h_{a} of the input nodes q and Θ. In general, calibration is a probabilistic



Task-Oriented Weighting:

1) $\hat{\Lambda}_a = \Lambda_a + {}^{t}J_a \, \bar{\Lambda}_t^e \, {}^{t}J_a^T$

2) $\hat{\Lambda}_a = \Lambda_a + (\Lambda_a {}^{a}J_t)({}^{a}J_t^T \Lambda_a {}^{a}J_t)^{-1} \, \bar{\Lambda}_t^e \, ({}^{a}J_t^T \Lambda_a {}^{a}J_t)^{-1}({}^{a}J_t^T \Lambda_a)$

$\hat{\Lambda}_a = \Lambda_a + ({}^{a}J_t^T)^{-1} \, \bar{\Lambda}_t^e \, ({}^{a}J_t)^{-1}$, only if ${}^{a}J_t$ is invertible

3) $\hat{\Lambda}_a = \Lambda_a + \Lambda_{at}\Lambda_t^{-1} \, \bar{\Lambda}_t^e \, \Lambda_t^{-1}\Lambda_{ta}$, general case

Figure 2.7: Detailed view of panel D inFig. 2.2. Three probabilistic graphs with different structures. They describe how the actual y_a (blue) and the task measurements y_t (red) are computed from the joint configuration q and the model parameters Θ for each of the N samples. For the sequential case in 1), the approach to minimize the error in the task space can be interpreted as adding virtual noise (1a) to the task measurement y_t and propagating it through the graph to y_a. The box provides the TOW for the two sequential and the general case.

process with multiple different sources of noise. To start, the graphical model allows the computation of $p(y_a|q, \Theta)$, which includes how the different noise sources get propagated through the graph and accumulate as uncertainty for the actual measurement. Furthermore, one can use graphical models to compute every conditional probability, specifically $p(y_a|y_t, q)$. This distribution tells us the relation between task y_t and actual measurement (q, y_a) even for case 2) if y_t follows after y_a, or there is no direct connection between y_t and y_a like in case 3). We can always compute all conditional probabilities for any directed acyclic graph [Bis06].

While, in general, the calibration model includes non-linear functions, one can linearize around the measurement points and approximate the graph with a linear system of Gaussians. The edges of the graph can be defined through

$$p(x_i|\mathrm{pa}_i) = \mathcal{N}(x_i| \sum_{j \in \mathrm{pa}_i} W_{ij}x_j + b_i, \Lambda_i). \tag{2.12}$$

Here, $\mathrm{p}a_i$ denotes the parent nodes of a child i. It follows that all conditional and marginal distributions are also Gaussian. To compute them, it is convenient to start from the joint distribution, which can be built iteratively by following the structure of parent and child nodes through the graph

$$\begin{aligned} \mathrm{E}[x_i] &= \sum_{j \in \mathrm{p}a_i} W_{ij} x_j + b_i \\ \mathrm{Cov}[x_i, x_j] &= \sum_{k \in \mathrm{p}a_j} W_{ik} \,\mathrm{Cov}[x_i, x_k] + I_{ij} \Lambda_j . \end{aligned} \tag{2.13}$$

From the joint distribution, one can order and separate variables block-wise to obtain the conditional probabilities

$$\begin{aligned} p(x) &= \mathcal{N}\Big(\begin{bmatrix} x_\mathrm{k} \\ x_\mathrm{l} \end{bmatrix} \Big| \begin{bmatrix} \mu_\mathrm{k} \\ \mu_\mathrm{l} \end{bmatrix}, \begin{bmatrix} \Lambda_\mathrm{k} & \Lambda_\mathrm{kl} \\ \Lambda_\mathrm{lk} & \Lambda_\mathrm{l} \end{bmatrix} \Big) \\ \mu_\mathrm{k|l} &= \mu_\mathrm{k} + \Lambda_\mathrm{kl} \Lambda_\mathrm{l}^{-1} (x_\mathrm{l} - \mu_\mathrm{l}) \\ \Lambda_\mathrm{k|l} &= \Lambda_\mathrm{k} - \Lambda_\mathrm{kl} \Lambda_\mathrm{l}^{-1} \Lambda_\mathrm{lk} . \end{aligned} \tag{2.14}$$

Special Cases

If the actual measurement y_a follows directly from the task measurment y_t (see Fig. 2.7, 1a), our approach results in

$$\hat{\Lambda}_\mathrm{a} = \Lambda_\mathrm{a} + {}^\mathrm{t}J_\mathrm{a} \,\bar{\Lambda}^\mathrm{e}_\mathrm{t} \,{}^\mathrm{t}J_\mathrm{a}^\mathrm{T}, \tag{2.15}$$

with the Jacobian ${}^\mathrm{t}J_\mathrm{a} = \frac{\partial y_\mathrm{a}}{\partial y_\mathrm{t}}$ describing the mapping from task to actual measurement. For example, this known form of the additional covariance arises for camera-based calibration, where the actual image h_a is projected from the cartesian position h_t. In this case, our approach can be viewed as adding virtual noise [2] at the task node in the probabilistic graph (see Fig. 2.7, 1a) and propagating it to the task node.

In case 2) y_t follows after y_a. Here, the relation between the task and actual space can not always be directly resolved

$$\begin{aligned} \hat{\Lambda}_\mathrm{a} &= \Lambda_\mathrm{a} + (\Lambda_\mathrm{a} {}^\mathrm{a}J_\mathrm{t})({}^\mathrm{a}J_\mathrm{t}^\mathrm{T} \Lambda_\mathrm{a} {}^\mathrm{a}J_\mathrm{t})^{-1} \bar{\Lambda}^\mathrm{e}_\mathrm{t} ({}^\mathrm{a}J_\mathrm{t}^\mathrm{T} \Lambda_\mathrm{a} {}^\mathrm{a}J_\mathrm{t})^{-1} ({}^\mathrm{a}J_\mathrm{t}^\mathrm{T} \Lambda_\mathrm{a}) \\ \hat{\Lambda}_\mathrm{a} &= \Lambda_\mathrm{a} + {}^\mathrm{a}J_\mathrm{t}^{-\mathrm{T}} \,\bar{\Lambda}^\mathrm{e}_\mathrm{t} \,{}^\mathrm{a}J_\mathrm{t}^{-1}, \text{ only if } {}^\mathrm{a}J_\mathrm{t} \text{ is invertible.} \end{aligned} \tag{2.16}$$

If the Jacobian ${}^\mathrm{a}J_\mathrm{t}$ is invertible, we can reduce the additional covariance to a familiar form. The general version allows us to compute the available parts of the inverse mapping, even if there happens to be information loss between y_a and y_t.

While those two edge cases in 1) and 2) can also be tackled separately, our method can compute the correct weighting to minimize the desired task space instead of the actual measurement space for all acyclic-directed graphs. Furthermore, the actual and task nodes can be distributed and arbitrarily related (3). The task-oriented weighting allows us to balance between Bayesian estimation and function fitting. For $\bar{\Lambda}^\mathrm{e}_\mathrm{t} = 0$, we have a purely probabilistic view of Bayesian estimation, which provides information on how trustworthy different actual measurements are depending on the noisy forward process that produced them. For $\Lambda_\mathrm{a} = 0$, one is in the domain of purely deterministic function fitting where one tries to find the closest fit for a family of functions and maps everything in the relevant space (see Fig. 2.2).

Alternative Approach - Minimization in Task Space

If we asume that all probabilities are Gaussian and we can transform actual measurements into task measuments using their means, we can derive the TOW also by performing MAP in the task space. This approach is more strict on the choice of probability functions, and the derivation over (2.9) holds in general and only relies on Gaussians only for the easier computability of the different terms.

Forumlating MAP in task space yields

$$\arg\min_{\Theta} \sum_n \log(p(y_\mathrm{t}^{(n)}|q^{(n)},\Theta)) + \log(p(\Theta)) \tag{2.17}$$

$$= \arg\min_{\Theta} \sum_n (\Delta y_\mathrm{t}^{(n)})^\mathrm{T}\, \Lambda_\mathrm{t}^{-1} \Delta y_\mathrm{t}^{(n)} + \Delta\Theta^\mathrm{T} \Lambda_\Theta^{-1} \Delta\Theta \tag{2.18}$$

$$\text{with } \Delta y_\mathrm{t}^{(n)} = y_\mathrm{t}{}^{(n)} - h_\mathrm{t}(q^{(n)},\Theta). \tag{2.19}$$

After iteratively setting up the joint distribution of all variables in the graphical model with (2.13), one can use (2.14) to obtain the task measurement $y_\mathrm{t}^{(n)}$ as mean of $p(y_\mathrm{t}|y_\mathrm{a}, q)$

$$p(y_\mathrm{t}|y_\mathrm{a}, q) = \mathcal{N}(\mu_\mathrm{t} + \Lambda_\mathrm{ta}\Lambda_\mathrm{aa}^{-1}(y_\mathrm{a} - \mu_\mathrm{a}),\, \Lambda_\mathrm{t|a}). \tag{2.20}$$

Here the means μ_a and μ_t are given through the measurement models $h_\mathrm{a}(q,\Theta)$ and $h_\mathrm{t}(q,\Theta)$. This leads to the estimation for the error in the task measurement as mapping from the actual measurement error

$$\Delta y_\mathrm{t}^{(n)} \approx \Lambda_\mathrm{ta}\Lambda_\mathrm{a}^{-1}(y_\mathrm{a}^{(n)} - h_\mathrm{a}(q^{(n)},\Theta)) = \Lambda_\mathrm{ta}\Lambda_\mathrm{a}^{-1}\Delta y_\mathrm{a}^{(n)}. \tag{2.21}$$

The virtual noise to account for the systematic error can be directly added to the task covariance $\Lambda_\mathrm{t} + \bar{\Lambda}_\mathrm{t}^\mathrm{e}$. Combining this adaption with the quadratic error terms to a single covariance in the actual measurement space leads to

$$\hat{\Lambda}_\mathrm{a} = (\Lambda_\mathrm{a}^{-1}\Lambda_\mathrm{at}(\Lambda_\mathrm{t} + \bar{\Lambda}_\mathrm{t}^\mathrm{e})^{-1}\Lambda_\mathrm{ta}\Lambda_\mathrm{a}^{-1})^{-1} \tag{2.22}$$

Finally everything is expressed through the measured quantities $\{(q^{(n)}, y_\mathrm{a}^{(n)})\}_{n=1}^{N_\mathrm{S}}$ Moreover, the error term is in Δy_a but scaled with a transformed covariance. For simple graphs (Fig. 2.7, 1 & 2), one can directly verify the equivalence of the two views derived in (2.11) and (2.22).

Summary

Fig. 2.2 summarizes the different aspects of our approach and is worth studying in closer detail. The goal is to estimate the parameters of model to match a true function as closely as possible. For this, the estimated uncertainty of each sample needs to match the true process that generated the measurements. Otherwise, the weighting of the error terms through the covariance is off and leads to wrong parameters. This weighting is already relevant if the model can fully describe the truth (A). If this is not the case and the model can not capture all aspects of the true function, this gap must be acknowledged. We achieve this by modeling the systematic error as virtual measurement noise. This addition happens outside the probabilistic model and does not represent a noise but an unmodeled error. Adding this term allows the model to be less accurate in regions where the true noise model would predict a high certainty to approximate the true function better overall. Fig. 2.5 demonstrated this behavior for a simple example (B). Furthermore, when we are in the regime of function approximation, the space in which we minimize the remaining model errors matters. A good solution for one metric is not necessarily a good fit for another (C). Finally, given noisy measurements in the actual space, we want a minimal error in the task space. Therefore, we combine the full probabilistic model for the measurements to estimate how informative different samples are with a weighting to perform the function approximation in the task space by mapping the virtual noise from the task space into the actual measurement space (D). The introduction of virtual noise and its extension into the task space lead to corrected calibration models. Without using these adaptions, the calibration will converge to a suboptimal set of parameters - independent of the number of measurement points.

2.4 Sensitivity Analysis and Optimal Experimental Design

2.4.1 Sensitivity Analysis

As we will show in the Section 2.5.4, the measurement functions can differ quite drastically in the spaces they describe and how much information they provide. The first step when designing a calibration setup is determining if a chosen measurement function h leads to any non-identifiable parameters. We can answer this question directly by looking at the jacobians of the different measurement functions

$$J^s = \frac{\partial h(q^s, \Theta)}{\partial \Theta}\bigg|_{\Theta_0}. \tag{2.23}$$

Each measurement is d-dimensional, and concatenating those matrices for each of the N_S measurements leads to the combined jacobian $\mathbf{J} = [J^1, \ldots, J^{N_\mathrm{S}}]$ with dimensions $(N_\mathrm{S} \cdot d) \times N_\Theta$. We can investigate which parameters are identifiable by the measurement function h by looking at the nullspace of $\mathbf{J}^T\mathbf{J}$. The size of the nullspace marks how many of the model parameters Θ cannot be identified. From the eigenvectors corresponding

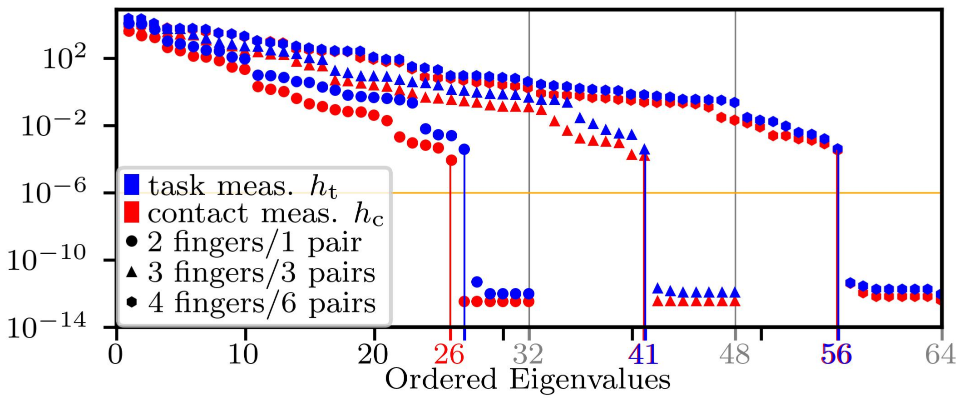


Figure 2.8: This figure shows the ordered eigenvalues for different measurement setups of the DLR-Hand II to analyze the sensitivity. The task measurement function h_t is blue, and our contact measurement function h_s is red. Furthermore, we show three modes. For the one where all the pairs are calibrated simultaneously (⬢), the kernels of both measurement functions have the same size. The same is true for the calibration with three fingers (▲) However, the kernel sizes differ for a single pair (●). The light gray vertical lines indicate the maximal number of parameters for each mode.

to the eigenvalues close to zero, one can identify sets of parameters that one cannot measure.

When one deals with two distinct measurement functions, as we described with the desired task h_t and the actual measurement function h_a, the necessary condition is that

$$\mathrm{Kernel}(\mathbf{J}_a^T \mathbf{J}_a) \subseteq \mathrm{Kernel}(\mathbf{J}_t^T \mathbf{J}_t). \tag{2.24}$$

If this condition is satisfied, one can use the actual measurement function h_a to identify all parameters relevant to the task, defined by h_t. Note that this is less strict than demanding that both kernels are zero and applies in general to distinct measurement functions for calibration and evaluation. We allow for unidentifiable parameters if they do not influence our desired measurement function.

2.4.2 Optimal Experimental Design

Given a set of measurements, we can use the TOW approach to perform the minimization in the task space. However, one can often additionally choose the configurations where the measurements should be collected. OED describes how to do this optimally. The intuition is to collect samples with maximal information for the calibration problem. Following Carrillo et al. [Car+13] , we use task D-optimality to select appropriate samples for measuring. However, we have two key differences in our setup. First, we have a theoretical task measurement function h_t and an actual measurement function h_a that we can apply to the hardware. Second, the actual dataset S_a might drastically reduce the configuration space. Ultimately, we want a good fit for the desired task measurement function h_t on the given task set S_t.

We generalize the optimality framework to account for those mismatches between measurement functions and data distributions. The central equation decouples [Car+13] and the task D-optimality can efficiently be computed by using

$$O_\mathrm{D} = \frac{1}{N_{\mathrm{S_t}}} \det\big(\mathrm{Cov}(\Theta) \sum_{i=1}^{N_{\mathrm{S_t}}} J_\mathrm{t}^{(i)T} J_\mathrm{t}^{(i)}\big). \tag{2.25}$$

The sum over $J_\mathrm{t}^{(i)\mathrm{T}} J_\mathrm{t}^{(i)}$ is constant for a given task set S_t of size $N_{\mathrm{S_t}}$ and a desired task measurement function h_t. The covariance over the calibration measurements can be estimated using the actual measurement function h_a and the actual calibration set S_a. Let $S_\mathrm{a} = \{s_i\}_{i=1}^{N_\mathrm{S}}$ be a subset of a larger calibration set and the combined jacobian $\mathbf{J}_\mathrm{a} = [J_\mathrm{a}^{s_1}, \ldots, J_\mathrm{a}^{s_{N_\mathrm{S}}}]$ corresponding to this subset of measurements. Then, the covariance of Θ is given through this sum

$$\mathrm{Cov}(\Theta) = \mathbf{J}_\mathrm{a}^T \Lambda_\mathrm{a} \mathbf{J}_\mathrm{a} + \mathrm{Diag}(\sigma_\Theta^2). \tag{2.26}$$

Here, the uncertainty in the actual measurements and prior get mapped into an uncertainty in the calibrated parameters.

(2.25) lets us compute how well different actual calibration sets y_a are suited to minimize the error of the task measurement function y_t over a desired task set. This criterion can be used to choose a good set of suitable poses for measuring. Besides reducing the overall size of the necessary calibration set, this selection criterion also counteracts the mismatch in the measurement functions and the calibration and task sets.

2.4.3 Efficient Data Collection

One can use the task D-optimality in the OED framework to select an optimal set of measurement configurations for a given task. However, one needs a large and diverse set for this search. If the underlying set is too small or homogeneous, the final set can not be diverse and informative.

We propose exhaustive rejection sampling to generate a suitable basis set (see Fig. 2.9). The critical challenge is that the actual measurement setup often poses many constraints on the possible configurations. For example, when a camera is used to collect measurements of markers, the markers must be in the camera's field of view, must not be occluded, and must face toward the camera. Another problem is that measurements can become ambiguous if multiple markers are visible simultaneously. The detection is often easier if one ensures that only a single marker is visible. Other measurement setups pose different constraints. For geometric fixtures, those mechanical constraints directly reduce the measurement space. In all cases, one must model and simulate the robot and the respective sensors to check those constraints. Furthermore, all configurations must be feasible regarding joint limits and self-collision for the robot. As the data collection happens before the calibration, one must account for uncertainties with larger safety margins when moving the robot. These precautions are particularly relevant to avoid

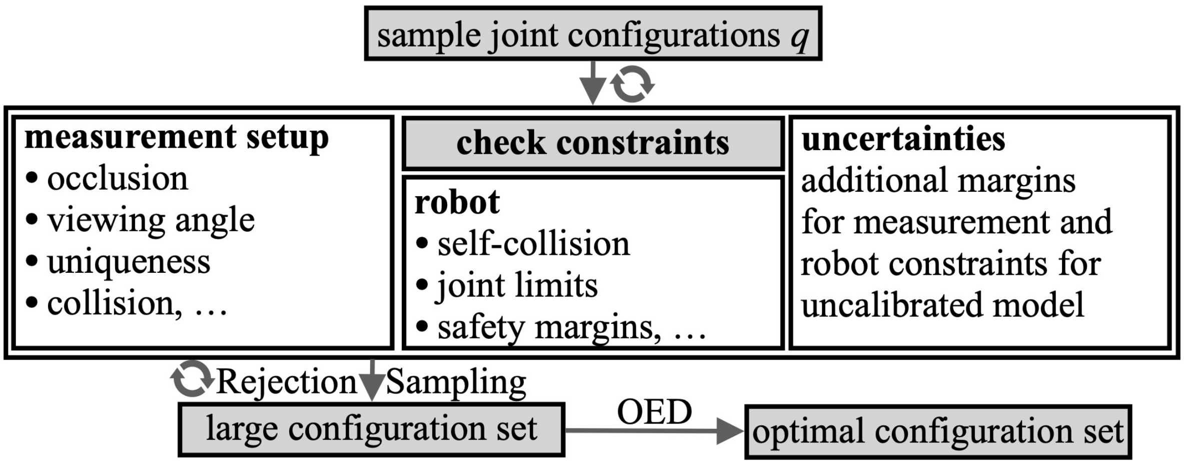


Figure 2.9: Scheme showing the data sample generation. We use rejection sampling to check all constraints on the measurement setup and the robot. Appropriate safety margins for those checks are crucial to account for the possible large uncalibrated errors of the robot kinematics.

invalid measurements and collisions for uncalibrated system with significant errors.

In general, the measurement process collects pairs (q, y) to calibrate the parameters of the measurement function $h(q, \Theta) = y$. For vision-based measurements, one collects the cartesian position $y^{(n)}$ for selected joint angles $q^{(n)}$. A key difference to a geometric calibration method, like contact-based calibration, is that one does not know the exact joint configuration beforehand, but it must be measured. For contact-based measurements, the $y^{(n)}$ is known a priori; the contact is, by definition, $y^{(n)} = 0$. The contact measurement delivers the exact configuration q_i, which leads to contact. Therefore, for geometric measurement, one can not directly move to a predefined configuration, and the setup must also include a search strategy to find the exact configuration that satisfies the geometric constraint.

If hysteresis is not a problem, ordering the configurations for the measurements is beneficial. Reducing the distance between the measurements via a traveling-salesman heuristic can make the data collection more time-efficient, and fast execution is always crucial for robotic applications.

2.5 Robot Model and Measurement Functions

2.5.1 Geometric Robot Model

The function we want to calibrate is the forward kinematics $f(q) = F$. This function is central for robotics and maps from the robots' joints q to the robot's body frames F. A

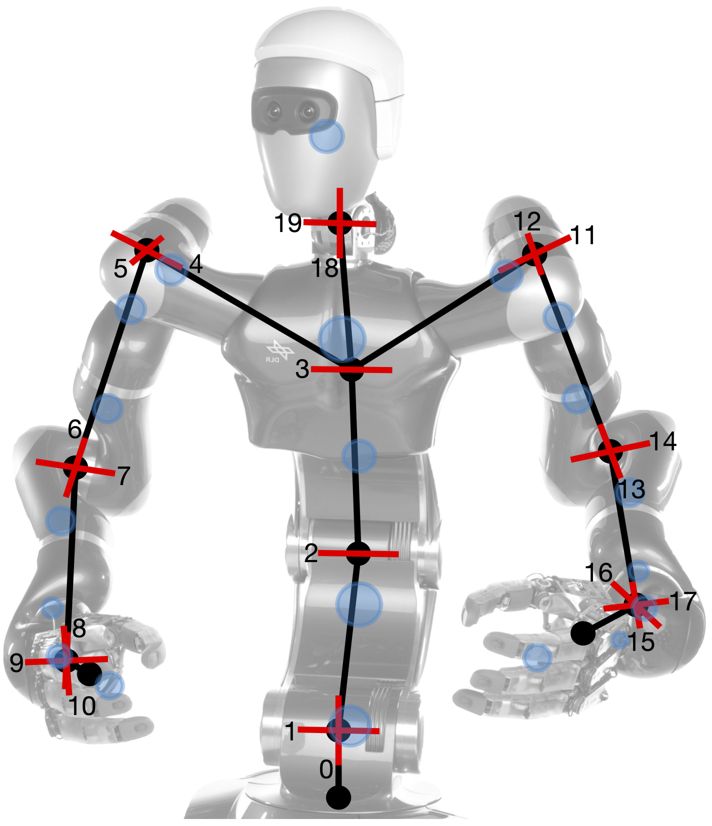


Figure 2.10: The kinematic tree structure of the humanoid Agile Justin with torso, head, and two arms, showing its 19 DoF (red) and the mass model (blue) used for the elastic calibration of the complete robot.

common representation uses the Denavit–Hartenberg (DH) parameters. In this formulation, four values $\rho_i = [d_i, r_i, \alpha_i, \theta_i]$ describe the transformation between two consecutive frames of the robot

$$^{i-1}T_i = \mathrm{Rot}_\mathrm{x}(\alpha_i) \cdot \mathrm{Trans}_\mathrm{x}(r_i) \cdot \mathrm{Rot}_\mathrm{z}(\theta_i) \cdot \mathrm{Trans}_\mathrm{z}(d_i). \tag{2.27}$$

This minimal representation with two translational (r_i, d_i) and two rotational parameters (α_i, θ_i) is enough to describe an arbitrary robot. The joints q_i are treated as offsets to θ_i for rotational joints and as offsets to d_i for prismatic joints.

The frame ^0T_E of an end-effector $(i = E)$ relative to the robot's base frame $(i = 0)$ is calculated by following the kinematic chain and applying the transformations in series

$$^0T_E = {}^0T_1 \cdot {}^1T_2 \cdot \ldots \cdot {}^{E-1}T_E. \tag{2.28}$$

For robots with a kinematic tree structure, like humanoids or robotic hands with multiple end-effectors $E_{k,k=1\ldots N_\mathrm{E}}$, equation (2.28) holds for each branch of the kinematic tree.

For a calibrated robot, not only is an accurate mapping to the end-effector relevant, but the whole chain needs to be modeled adequately. Only then can self-collisions be avoided, and grasps using the whole surface of the hand can be performed stably. So, the forward kinematics maps from joint configurations $q \in \mathcal{Q}$ to *all* the frames F of the robot

$$f(q, \rho) = F = [{}^0T_1, {}^0T_2, \dots, {}^0T_{N_\mathrm{F}}]. \tag{2.29}$$

Each of the N_F frames F_i describes a full 6D pose $(F_{i,x}, F_{i,r}) \in \mathbb{R}^3 \times SO(3)$. $F_{i,x}$ is the translational and $F_{i,r}$ the rotational part of the homogeneous transformation F_i.

2.5.2 Non-geometric Robot Model

Only considering the geometric model can fall short of describing the true robot. Typical cases where non-geometric effects must be modeled are robots with high elasticities or gear backlash. When requiring higher accuracy, it becomes necessary to also include non-geometric effects. Given that the DH parameters can describe any robot, it is practical to incorporate non-geometric effects into this formalism. The idea is to keep the basic structure of the DH parameters, which describe the consecutive mapping along the frames of the kinematic tree. The non-geometric effects act as offset on the geometric DH parameters

$$\rho = \rho(\rho_0, \rho_\mathrm{n}) = \rho_0 + g(\rho_\mathrm{n}). \tag{2.30}$$

In general, g can be an arbitrary function parameterized by the additional non-geometric DH parameters ρ_n. However, aspects like computation speed are essential as the forward kinematics is central in most robotic applications. This work focuses on torques as the primary source of non-geometric effects. Generally speaking, an acting torque will bend the robot and produce a slightly different pose F. Often, a linear dependence is sufficient to model the torque-induced bending of the robot, as Caenen et al. [CA90] first showed for a robotic arm

$$\rho = \rho(\rho_0, \kappa, \tau) = \rho_0 + \kappa\, \tau \tag{2.31}$$

They also neglected any lengthening effects of the links, and the matrix $\kappa = [0, 0, \kappa^\alpha, \kappa^\theta]$ describes only compliance around the respective axes, corresponding to the rotational DH parameters $[\alpha, \theta]$ and the acting torques $\tau = [\tau^x, \tau^z]$.

The non-geometric forward kinematics f now depends via the DH parameters ρ on the torques τ and the elasticity parameters κ:

$$F = f(q, \rho(\rho_0, \kappa, \tau)) \tag{2.32}$$

One option to calculate the forward kinematics is to measure the acting torques in each joint and use those values to compute the offsets to the geometric DH parameters. However, this is not possible if one wants to use this model beforehand for planning, as at this time, no sensor readings are available. Furthermore, torque sensors can only measure the torques acting in the joints (which relate to an offset in θ).

A self-contained approach is required if one wants to use the forward kinematics for planning or also needs to consider lateral elasticities. In the regime of no external forces and reasonably slow, quasi-static motions, the torques originate only from the robot's weight and its specific distribution. For a frame, the pair $\nu_j = [m_j, w_j]$ describes the mass m_j and its position w_j relative to this frame (i.e., ${}^0w_j = {}^0T_j w_j$). The torques produced by this mass due to the gravity vector g around the respective coordinate axes of another frame i with origin ${}^0p_i$ can be calculated [CA90] by[1]:

$$\begin{aligned}\tau^x_{ij} &= (({}^0T_j\, w_j - {}^0p_{i-1}) \times m_j g) \cdot {}^0e^x_{i-1}\\ \tau^z_{ij} &= (({}^0T_j\, w_j - {}^0p_i) \times m_j g) \cdot {}^0e^z_i\end{aligned}$$

The contributions from all masses that act on the respective link must be summed up to calculate the full torque τ_i acting on a link i; that is, the torques from all masses higher up in the robot's kinematic tree.

The torques $\tau = \tau(F, \nu)$ now depend on the weight distribution $\nu = [\nu_1, \nu_2, \ldots]$ of the robot and therefore on the frames F which are depending on the DH parameters ρ:

$$\rho = \rho(\rho_0, \kappa, \tau(F(q, \rho), \nu)). \tag{2.33}$$

This implicit equation for the DH parameters defines the equilibrium between the torques due to gravity and the torques due to the robot's flexion. We define the solution to this equation as the non-geometric DH parameters

$$\rho^* = \rho^*(q, \rho_0, \kappa, \nu). \tag{2.34}$$

One possibility to solve (2.33) is by the following iteration:

$$\rho_n = (1-\lambda)\rho_{n-1} \;+\; \lambda\rho(\rho_0, \kappa, \tau(F(q, \rho_{n-1}), \nu)) \tag{2.35}$$

$$\rho_\infty = \rho^* \tag{2.36}$$

Choosing an appropriate $\lambda \in [0, 1]$ for the weighted sum ensures the convergence of the iteration even for very soft (or strongly non-linear) robots. Interpreting the DH parameters as generalized coordinates and the update rule as the discretized integration over time of a damped dynamical system gives an intuition for the convergence of this iterative update. The robot moves due to gravity and swings into the torque equilibrium. A suitable choice for the start of the iteration are the geometric DH parameters ρ_0 which can be interpreted as a robot with zero gravity or infinite stiffness. Finally, the non-geometric forward kinematics is then given by

$$F = f(q, \Theta_\text{f}) = f(q, \rho^*(q, \underbrace{\rho_0, \kappa, \nu}_{\Theta_\text{f}})),$$

[1]Those torques act on the rotational axes parameterized by α_i and θ_i, described in (2.27). Because they make up a frame by sequential stacking, they do not all share the same frame of reference.

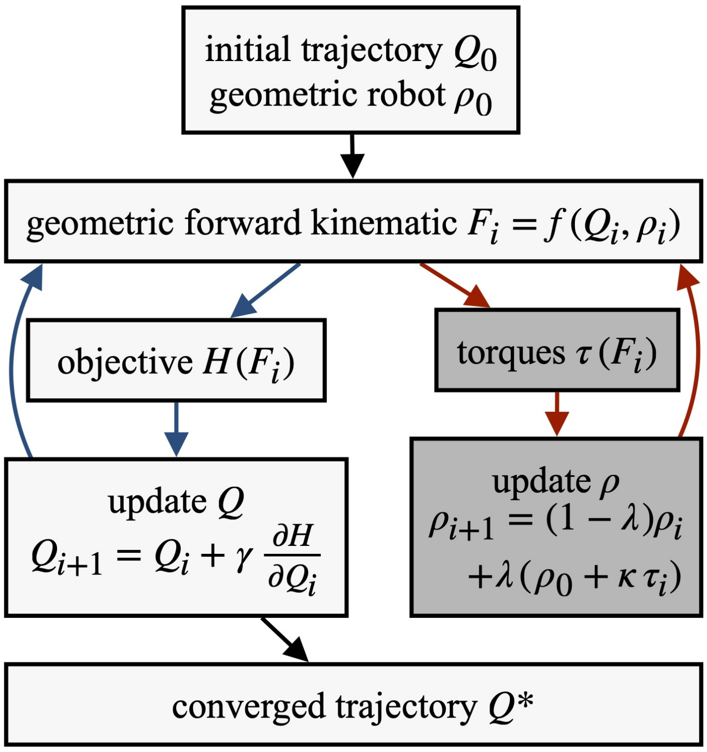

initial trajectory Q_0
geometric robot ρ_0
geometric forward kinematic $F_i = f(Q_i, \rho_i)$
objective $H(F_i)$
torques $\tau(F_i)$
update Q
$Q_{i+1} = Q_i + \gamma \frac{\partial H}{\partial Q_i}$
update ρ
$\rho_{i+1} = (1-\lambda)\rho_i + \lambda(\rho_0 + \kappa \tau_i)$
converged trajectory Q^*

Figure 2.11: Flowchart of the optimization loop in light gray. Dark gray shows the additional loop for the static torque equilibrium. As the outer loop converges, no additional passes through the inner torque loop are necessary.

where ρ^* describes the solution of the non-geometric DH-parameters in torque equilibrium, resulting from the robot's mass distribution ν in configuration q.

2.5.3 Efficient Compensation

The calibration goal is to find a set of parameters Θ which describes the robot as accurately as possible. Nevertheless, the speed and ease of use of the calibration model, e.g., in motion planning, are also crucial. Incorporating the new set of DH parameters ρ is straightforward and works without any changes or additional costs as they replace the old DH parameters. The same holds for masses m and compliances κ. However, to determine the elastic effects, one must find the static equilibrium between acting torques and elasticities described in (2.33). While it might be feasible for calibration (offline procedure) to use the iterative algorithm (2.35), it is more prohibitive for compensation (online). One should carefully evaluate this trade-off between accuracy and simplicity when choosing a calibration model for a robot.

Knowing that we will use the forward kinematics mostly in the framework of an optimization-based path planner has further implications. Such a planner works on paths in configuration space $Q = [q_1, q_2, ..., q_n]$ and performs iterations to get from an initial path Q_0 to a converged path Q^*. For each step, the optimizer considers the

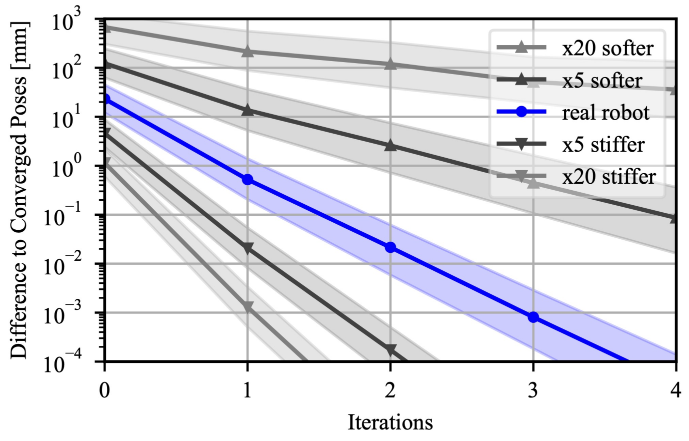


Figure 2.12: Simulated convergence towards the static torque equilibrium for robots of different stiffness. For iteration zero, the elastic effect is ignored. The bands indicate the standard deviation for 1000 different joint configurations.

objective function $H(Q_i)$ and updates the path using the gradient information. The nested structure of the forward kinematics

$$f^*(q) = f(q, \rho(f(q, \rho(f(...))))) \tag{2.37}$$

leads to highly non-linear gradients, making the model more difficult to use. As a solution to this, we separate the torque equilibrium search in a separate loop, as shown in Fig. 2.11. After determining the acting torques accurately and updating the DH parameters accordingly, we assume these to be constant for the gradient calculation $\partial F/\partial q$. This allows us to use the pure geometric forward kinematics, implicitly assuming $\partial \rho/\partial q = 0$. Although we completely omit the torque iterations for the gradients, this approximation does not hinder convergence in our tests.

The second important aspect visualized in Fig. 2.11 is the combination of the optimization loop and the torque equilibrium loop. Even if the updates of the configurations are large at the beginning of the optimization, when the planner converges, the pose updates get small. If those updates are significantly smaller than the offset produced by a torque update, it is unnecessary to search for a new static equilibrium iteratively. In other words, it is sufficient to reuse the already computed frames, calculate the acting torques, and update the DH parameters only once while nevertheless solving the forward kinematics f^* exactly. The outer loop of the converging optimizer makes it unnecessary to perform inner iterations when searching the torque equilibrium.

2.5.4 Types of Measurement Functions

To identify the parameters of the forward kinematics Θ_v, we need a measurement setup that relates the joint configuration to the robot frames. The following section describes different types of measurement functions, both open-loop and closed-loop variants, and gives detailed descriptions and formulas to the sketches in Fig. 2.4.

Cartesian Measurement Function

One common approach to collect measurements is to mount markers on the kinematic tree and use a tracking system to collect cartesian measurements of these markers. Assuming that the markers are fixed at position ${}^{E_k}x$ relative to end-effector E_k, the cartesian measurement function

$$y^k = h_{\mathrm{v}}(q, \Theta)^k = {}^{\mathrm{c}}T_0 \cdot f_{E_k}(q, \Theta_{\mathrm{f}}) \cdot {}^{E_k}x \tag{2.38}$$

describes how a marker moves dependent on the joint configuration q and the parameters Θ_{f}. Suppose one wants to calibrate the forward kinematics jointly for multiple end-effectors. In that case, one can concatenate the measurements for each of the N_{E} markers to a combined measurement function $h_{\mathrm{v}} = [h_{\mathrm{v}}^1, \dots, h_{\mathrm{v}}^{N_{\mathrm{E}}}], h_{\mathrm{v}}^k \in \mathbb{R}^3$. Multiple markers per end-effector also make estimating the frame's orientation possible. The full calibration parameters consist now of the parameters to describe the forward kinematics Θ_{f} and additionally the relative frame of the camera system to the robot base ${}^{\mathrm{c}}T_0$ and the relative positions of the N_{E} markers ${}^{E_k}x$

$$\Theta = [\, \Theta_{\mathrm{f}}, \underbrace{{}^{\mathrm{c}}T_0,\, [{}^{E_k}x]_{k=1}^{N_{\mathrm{E}}}}_{\Theta_{\mathrm{v}}}]. \tag{2.39}$$

While, in general, an external camera can measure without constraining the robot directly, one still needs to account for self-collision and a clear view of the markers. These additional constraints reduce the possible configuration space for the measurements.

Image Measurement Function

One can also use a single camera instead of an entire tracking system. However, one cannot directly measure the marker's cartesian position, only its projection into the image space $U\colon \mathbb{R}^3 \to \mathbb{R}^2$. Here, we use the classical pinhole model with radial distortion to project the 3D position of the marker x into 2D pixel coordinates u [BFB15]. First, the marker's position x must be transformed into the camera frame ${}^0T_{\mathrm{c}}$. Then, the relative position is projected along the z-axis of the camera frame by $P(x)$ and radial distortion $D(u, \xi_{\mathrm{c}})$ is added. The marker's pixels u in the image are then calculated as an offset from the camera's center point c_{c} scaled with the focal length l_{c}:

$$\begin{aligned} u &= U({}^0x, {}^0T_{\mathrm{c}}, \Theta_{\mathrm{ci}}) = c_{\mathrm{c}} + l_{\mathrm{c}} \cdot D(P({}^{\mathrm{c}}T_0\, {}^0x), \xi_{\mathrm{c}}) \\ &\text{with } \; P(x) = \left(\frac{x_{\mathrm{x}}}{x_{\mathrm{z}}}, \frac{x_{\mathrm{y}}}{x_{\mathrm{z}}}\right)^{\mathrm{T}} ;\; D(u, \xi_{\mathrm{c}}) = \frac{u}{1 + \xi_{\mathrm{c}} |u|^2} \end{aligned} \tag{2.40}$$

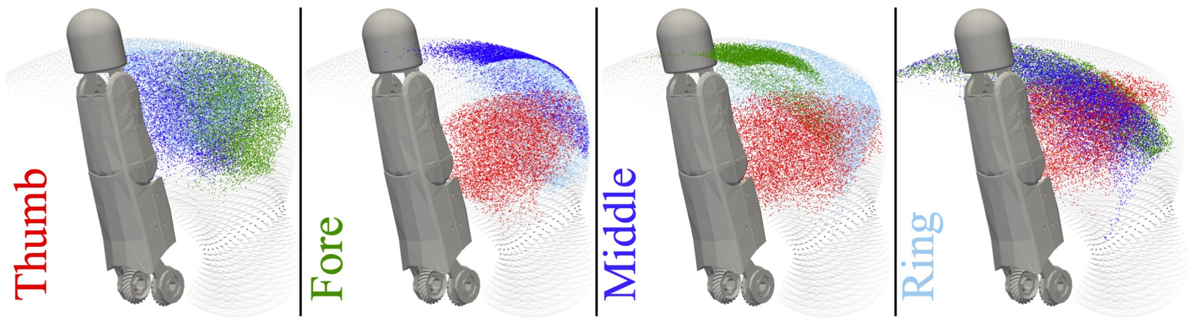


Figure 2.13: Visualization of the constrained workspace of the different fingers of the DLR-Hand II. For each finger, the colored dots indicate the common workspace with the three other fingers. Contact needs to happen in those shared subspaces. In light grey, the total workspace is indicated.

This yields the measurement function for a single marker

$$y^k = h_\mathrm{c}(q,\Theta)^k = U(\underbrace{f_{E_k}(q,\Theta_\mathrm{f})\cdot{}^{E_k}x}_{{}^0x_{E_k}},\ \underbrace{f_\mathrm{c}(q,\Theta_\mathrm{f})\cdot{}^\mathrm{c}T_\gamma}_{{}^0T_\mathrm{c}},\ \Theta_\mathrm{ci}). \tag{2.41}$$

This formulation includes both open and closed-loop setups depending on whether the camera is mounted on the robot or fixed outside the robot and, therefore, relative to the robot's static base. Again, one can concatenate the measurements for each of the N_E image markers to a combined measurement function $h_\mathrm{c} = [h_\mathrm{c}^1, \dots, h_\mathrm{c}^{N_\mathrm{E}}]$, $h_\mathrm{c}^k \in \mathbb{R}^2$. Besides the position of the markers ${}^0x_{E_k} = {}^0T_{E_k}{}^{E_k}x$ and frame of the camera ${}^0T_\mathrm{c} = {}^0T_\mathrm{c}{}^\mathrm{c}T_\gamma$ relative to the forward kinematics F, the intrinsic parameters of the camera Θ_ci can also be part of the calibration. Those additional parameters are combined and denoted as

$$\Theta = [\,\Theta_\mathrm{f},\ \underbrace{{}^\mathrm{c}T_\gamma,\ [{}^{E_k}x]_{k=1}^{N_\mathrm{E}},\ \overbrace{c_\mathrm{c},\ l_\mathrm{c},\ \xi_\mathrm{c}}^{\Theta_\mathrm{ci}}}_{\Theta_\mathrm{c}}\,]. \tag{2.42}$$

As for a tracking system, one must account for self-collision and a clear view of the markers without obstruction or occlusions. Furthermore, the 2D image space differs from the Cartesian space and provides less information per sample.

Contact Measurement Function

Another option for performing the actual measurement is to use geometric information. For example, one could constrain the end-effector's motion on a known plane or to a fixture. These cases are also open-loop, as the location of the additional tools must be known. A more general closed-loop formulation uses the contact information of the robot itself. In other words, the corresponding measurement function measures the distance between different robot parts. To achieve this, one needs the exact geometry of the two bodies to compute the distance d^v between a pair of bodies $v = (E_k, E_l)$ on the kinematic tree. The distance depends then only on the relative position and orientation of the bodies ${}^{E_k}T_{E_l}$, which can be directly computed from the forward kinematics of the two respective frames

$$d^v({}^{E_k}T_{E_l}) = d^v((f_{E_k}(q, \Theta_\mathrm{f}))^{-1} \cdot f_{E_l}(q, \Theta_\mathrm{f})). \tag{2.43}$$

This formalism can also describe a setup with an external fixture. In that case, one body is static and does not depend on the robot's joint configuration. This can be modeled by linking that body to the robot's base.

If the bodies at E_k and E_l have simple geometric forms, one can directly compute the distance d^v. The more general case is that the geometries are given as arbitrary meshes. In this case, one has to use, for example, algorithms like GJK [GJK88] to compute the distance. The contact measurement function for the pair v can now be written as

$$y^v = h_\mathrm{s}(q, \Theta)^v = d^v(f(q, \Theta_\mathrm{f})). \tag{2.44}$$

This function can only measure the scalar distance between two body pairs. With N_E end-effectors on the kinematic tree, there are in total $N_\mathrm{V} = \binom{N_\mathrm{E}}{2}$ pairs. The combined function for all those pairs is $h_\mathrm{s} = [h_\mathrm{s}^1, \dots, h_\mathrm{s}^{N_\mathrm{V}}], h_\mathrm{s}^k \in \mathbb{R}$.

This minimal measurement function depends solely on the robot's kinematics and not on additional external tools. If the shape of the robot's surface and structure is known, there are no additional tools needed, and no additional parameters

$$\Theta = \Theta_\mathrm{f}. \tag{2.45}$$

As only configurations in contact can be measured, the available configuration space is drastically reduced (see Fig. 2.13).

Relative Position Measurement Function

This function is included to show that a task measurement function is not necessarily practical for the real hardware. Instead, it should capture the requirements of the task accurately.

For example, for a robotic hand to move the fingers in a controlled manner and perform dextrous manipulation, an accurate model for the relative (not the absolute) positions of the fingertips to each other is necessary [SPB22; Pit+23].

For N_E end-effectors, $N_\mathrm{E} - 1$ distance vectors define the whole set. One can choose one tip E_1 as the basis and compute the positions relative to it for the remaining ones

$$y^k = h_\mathrm{r}^k(q, \Theta) = f(q, \Theta_\mathrm{f})_{E_k,x} - f(q, \Theta_\mathrm{f})_{\mathrm{E}_1,x} \ . \tag{2.46}$$

Concatenating the measurements for all the end-effector pairs yields the combined function $h_\mathrm{r} = [h_\mathrm{r}^2, \dots, h_\mathrm{r}^{N_\mathrm{E}}], h_\mathrm{r}^k \in \mathbb{R}^3$. This results in $3 \cdot (N_\mathrm{E} - 1)$ data points per robot pose for the three spatial directions and N_E end-effectors. Note that (2.46) is a theoretically desired measurement function, which measures all the information relevant for precise in-hand manipulation. For example, it cannot detect a simultaneous translational offset of all end-effectors like a cartesian tracking system could. However, such a shift does not change the end-effectors' relative behavior and is irrelevant to the task.

Task Test Set

Besides defining a measurement function h_t to describe the task, a suitable task set S_t is crucial to evaluate the calibration quality. Depending on the robot application, this set can vary. Often, one wants a high accuracy across the whole Cartesian workspace. However, setups that focus on smaller subsets are also possible. Generally, this task set S_t differs from the actual calibration set S_a. This difference is especially prominent if the actual measurement setup poses additional constraints on the robot configurations that are not present for the task.

2.6 Experiments

We conducted multiple simulation and hardware experiments to verify the methods introduced in Sections 2.3 and 2.4. Simulations help, especially for OED and our TOW, to validate the results of an extensive number of experiments and to directly compare all relevant metrics for the actual and task measurement functions. For the analysis, we focus on two systems: the humanoid robot Agile Justin and the DLR-Hand II. Both robots have a tree-like structure and many DoF, making them challenging examples for calibration.

2.6.1 DLR-Hand II - Contact-based Calibration

The first robotic system we analyze is the DLR-Hand II (see Figs. 2.1 and 2.14). This four-fingered hand has 12 DoF and is torque-controlled. It is capable of performing dextrous grasping [Win+22; Hum+23] and in-hand manipulation [SPB22; Pit+23]. The small form factor makes visual calibration prone to occlusions and impractical. So, we opt for a purely contact-based calibration. We include all four DH parameters per joint for the forward kinematics model, resulting in $N_\Theta = N_\mathrm{DoF} \times 4$ calibration parameters. Each finger has three active and one passive joint. This results in 16 DH parameters per finger and 64 for the whole hand. Each finger has three parallel joints, and the ring and middle finger are mounted on the base with the same orientation, which leads to redundancies.

Sensitivity Analysis

For grasping and in-hand manipulation, the relative position of the fingers is crucial to interact with different objects dextrously. Therefore, the task measurement function is $h_\mathrm{t} = h_\mathrm{r}$ (see (2.46)). To check if the actual contact measurement function $h_\mathrm{a} = h_\mathrm{s}$ (see (2.44)) can identify all the parameters relevant to the task, we compute the nullspace of

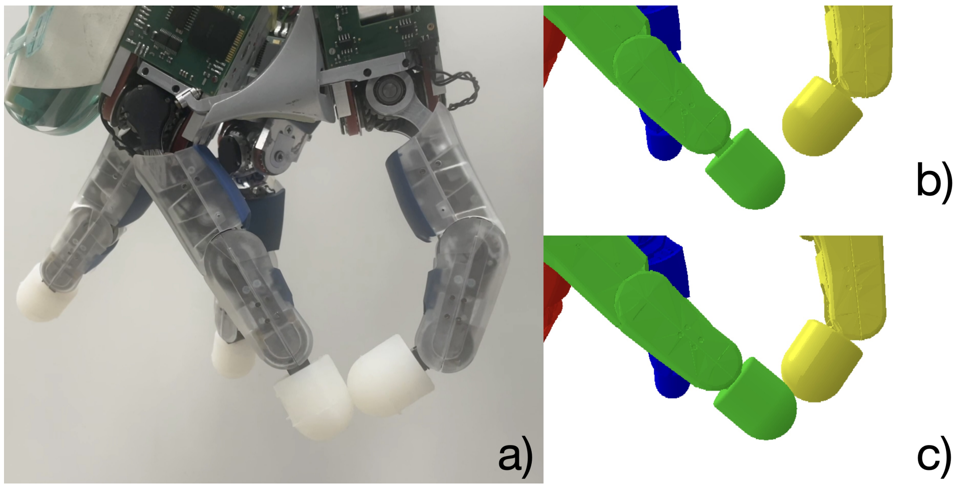


Figure 2.14: The DLR-Hand II in a finger contact pose from the test set *a)* with the measured joint angles q mirrored to a nominal *b)* and a calibrated *c)* model. The model's error is visibly reduced from a distance of 8.3 mm to a penetration of 1 mm, allowing dextrous in-hand manipulation.

$\mathbf{J}^T\mathbf{J}$ as described in Section 2.4.1. We evaluate the Jacobians $\mathbf{J}_\mathrm{a}$ and $\mathbf{J}_\mathrm{t}$ for the nominal kinematic at 100 configurations from the two respective sets.

Fig. 2.8 shows the sensitivity analysis for the DLR-Hand II. The task measurement function h_t is blue, and the actual contact measurement function h_a is red. Higher eigenvalues relate to better sensitivity, and eigenvalues below 10^{-6} indicate that parameters become hard to identify numerically. All the finger pairs are calibrated simultaneously for the rightmost mode (⬢). Both measurement functions have 56 eigenvalues larger than 10^{-6}. Analyzing the eigenvectors confirms that the kernel of our contact measurement function h_s is fully included in the kernel of the task measurement function h_t. Therefore, we can identify all parameters relevant to the task. The parallel axes explain the $8(= 2\times 4)$ unidentifiable parameters for the hand. Each finger has three parallel axes along which a shift can be adjusted without influencing the end-effector. This results in a two-dimensional nullspace and 14 identifiable parameters from the 16 for each finger.

The leftmost mode (●) shows the calibration of just a single pair. Here, from the $32(= 2 \times 16)$ total parameters, $28(= 2 \times 14)$ should be identifiable. However, measuring between two chains yields less information than measuring between three or more chains. The task measurement using the relative positions can identify 27 parameters. However, when restricting the measurements to a single pair, the scalar distance function can measure even less, and only 26 parameters are identifiable. An intuition is that two end-effectors can move on a sphere around each other without changing the scalar distance between them. In that case, one cannot identify all the parameters relevant to the task by pure contact measurements. This invariance generally resolves when adding a third chain (▲) to the picture, favoring a holistic calibration of all parts of the kinematic tree.

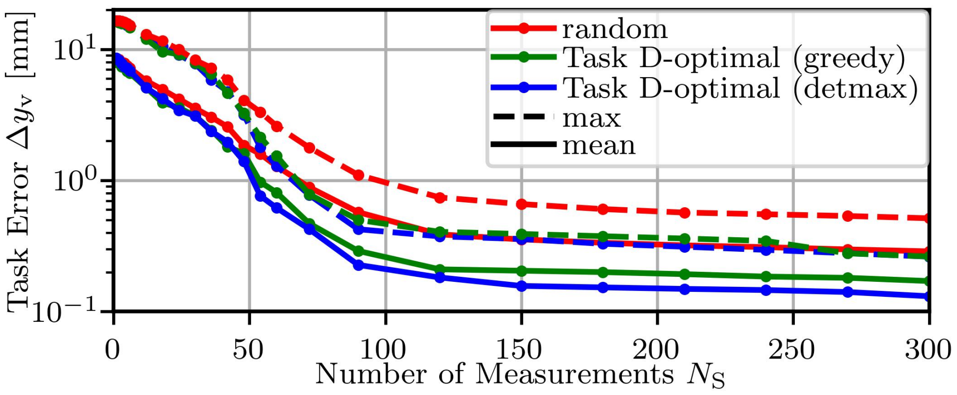


Figure 2.15: Convergence of mean and maximal cartesian error Δy_{v} on the uniform cartesian test set for different calibration sets for the contact measurement function h_{s}. Randomly chosen sets (red) converge slower than the sets designed according to task D-optimality (2.25). The greedy strategy is green, and the detmax[Mit00] strategy is blue. The distribution mismatch between calibration and test set can explain why the gap between random and selected samples remains significant even for larger calibration sets.

Optimal Experimental Design

We conducted extensive simulations to study the influence of different calibration sets on the resulting task accuracy. Using the nominal DLR-Hand II as a basis, noise was applied to the DH parameters to obtain variations of the robot. For the rotational DH parameters α and θ we add uniformly sampled noise in the range of $\pm$ 5° and for the translational parameters DH d and r we used uniform noise $\pm$ 5 mm. In this fashion, we created 100 different kinematics to ensure a broad distribution of robot models. Next, we simulated the data collection step and collected measurements for the actual contact measurement function $h_{\mathrm{a}} = h_{\mathrm{s}}$ and the task measurement function $h_{\mathrm{t}} = h_{\mathrm{r}}$. The mean deviation from those models was 21mm from the nominal kinematic on the uniform cartesian task set.

Fig. 2.15 shows the results of the optimal sample selection introduced in Section 2.4.2. The cartesian error Δy_{v} on the cartesian task set is drawn over the number of contact measurements for different selection strategies. Randomly chosen sets (red) converge slower than the sets designed according to task D-optimality (see (2.25)). We compare a greedy strategy (green), which at each step adds the sample s_i, which improves the task D-optimality most against the DETMAX algorithm [Mit00] (blue). This procedure tries to swap samples in an existing calibration set to improve the task D-optimality. Both selection strategies outperform the random approach, significantly reducing the mean error to 0.1mm for a set of 300 measurements.

The configurations for the contact measurements differ from the uniform distribution in the cartesian workspace on which we evaluate the calibrations. This distribution mismatch explains why the gap between random and optimized samples remains significant even for larger calibration sets. Using task D-optimality as a selection criterion

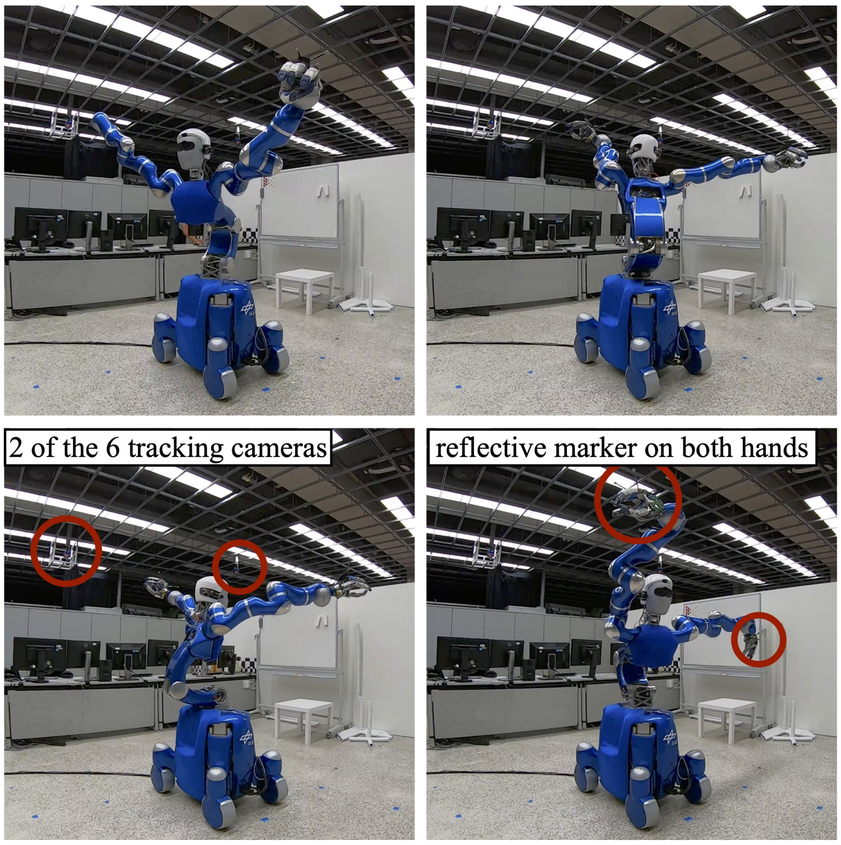


Figure 2.16: We collect measurements using the Vicon tracking system to evaluate our approach in the cartesian space. This external tracking system consists of six cameras mounted on the ceiling that directly track the cartesian position of retro-reflective markers with high accuracy.

directly accounts for this shift and significantly improves contact calibration with its strict constraints on data generation.

Real-world Experiments

After performing the sensitivity analysis and OED, we calibrated the DLR-Hand II using only pairwise contact measurements h_s. For this, we collected 300 samples over all finger pairs to ensure a large enough set for calibration and evaluation. The uncalibrated model has a mean error of 6 mm over all 300 samples and a maximal error of up to 17.7 mm for the scalar distance measurement. Calibrating all the DH parameters leads to a model that has a maximal error of 3.7 mm and a mean error of 0.7 mm. Fig. 2.14 shows the improvement through our proposed contact-based calibration on the real DLR-Hand II.

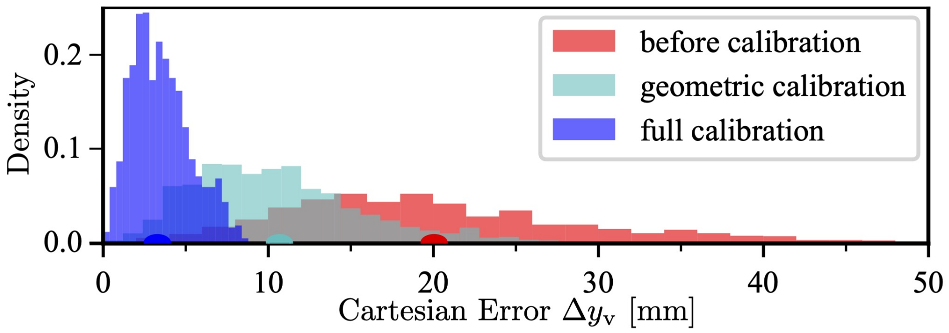


Figure 2.17: Histogram of the residual error at the end-effectors before and after calibration, highlighting the need for non-geometric modeling of Agile Justin [Bäu+14]. The circles on the x-axis mark the mean μ of the error.

Table 2.2: "Leave-one-out analysis": residual Cartesian err. Δy_v [mm]

	full	$-\kappa^\theta$	$-\alpha$	$-\theta$	$-\kappa^\alpha$	$-r$	$-d$
μ	3.12	9.09	6.48	6.35	4.85	4.33	3.46
σ	1.71	4.23	2.93	3.16	2.28	2.03	1.73
max	8.23	24.59	17.25	18.89	13.37	11.09	8.97

2.6.2 Agile Justin - External Tracking System

DLR's Agile Justin is a humanoid robot with two arms and a torso on a mobile platform. Overall, it has 19 DoF, an RGB camera, and a depth sensor mounted on its head. For the actual calibration, we first used an external cartesian tracking system $h_\mathrm{a} = h_\mathrm{v}$ (see (2.38)) to generate ground truth data (see Fig. 2.16). In this case, the actual measurement setup is identical to the desired task $h_\mathrm{t} = h_\mathrm{v}$. The calibration should improve the cartesian accuracy of the end-effectors.

We collected 500 samples with the external tracking system and split them into a calibration set with 300 and a test set with 200 samples. In what follows, we calibrate all the geometric DH parameters plus the stiffness parameters $\Theta = [\rho, \kappa, \Theta_\mathrm{v}]$. The priors used for the calibrations were chosen based on previous experiments with the robot, with uncertainty Λ_Θ in lengths of 0.1 m, angles of 0.2 rad, and elasticities of 0.1 rad/kNm. As the first step, the additional frames parametrized by Θ_v must be determined to close the measurement loop. Only calibrating those frames gives the accuracy of the nominal kinematics before calibration, with a mean residual error of 20 mm over the 200 test poses. In the worst cases, the error was larger than 60 mm. After calibration, the mean residual error of the whole model is reduced to 3.1 mm and the maximum error to 8.2 mm.

Fig. 2.17 compares the nominal model (red), a pure geometric calibration without elasticities (green), and the full non-geometric calibration (blue). This comparison highlights the need for a non-geometric calibration model for Agile Justin. The joints' elasticities and lateral elasticities produce significant errors, which a purely geometric

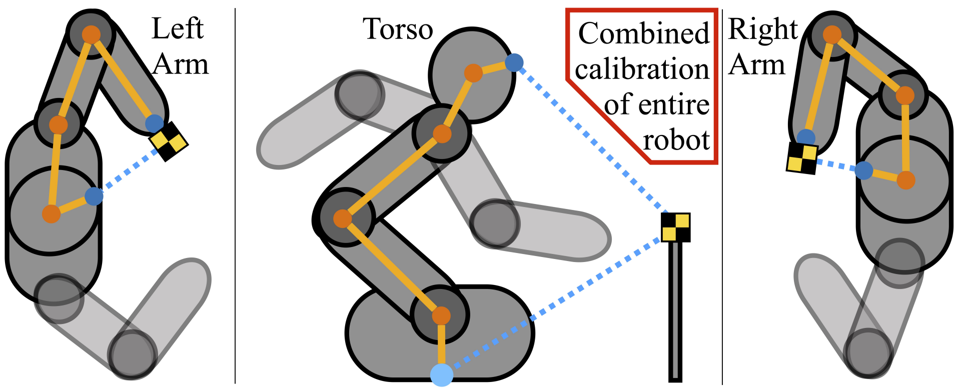


Figure 2.18: With its internal camera, the robot collects RGB images of markers on both hands and a pole in front of it. The blue chains show how forward kinematics plus camera projection close the measurement loop. Even if the arms are not directly involved in the pole measurements, their mass distribution in different positions influences the torso elasticities.

Table 2.3: Error in image and cartesian space for different setups.

Calibrate			Image Error [px]			Cartesian Error [mm]		
on	Θ_{ci}	TOW	μ	σ	max	μ	σ	max
Images	no	no	1.05	0.59	3.76	4.77	2.29	11.75
Images	yes	no	0.97	0.53	3.44	4.65	2.27	11.58
Images	no	yes	1.21	0.70	4.13	4.11	1.87	9.34
Images	yes	yes	1.15	0.62	3.97	3.94	1.83	9.16
Position	-	-	-	-	-	3.12	1.71	8.23

model cannot capture. In Table 2.2, starting from the full model on the left, only a specific parameter type was left out one at a time. This analysis shows how crucial joint and lateral elasticities are for accurately modeling Justin's kinematics. One reason for the prominent joint elasticities is that we model not only the mechanical elasticity of a joint but also the elasticity of the joint position controller. We use a relatively simple joint position controller as it is robust (e.g., it does not rely on the torque sensors, which are notoriously drifting and hard to maintain). However, it results in an additional joint elasticity.

2.6.3 Agile Justin - Internal RGB Camera

To make the calibration more practical, we replace the measurement with the external tracking system through a visual calibration using the robot's RGB camera. For the calibration of the entire kinematic tree, the internal camera tracks two markers at the ends of the chain for the arms and one external marker for the torso. The calibration sketch is shown in Fig. 2.18 and examples of the actual measurements in Fig. 2.19. This setup allows the calibration to be performed autonomously everywhere, not only in a

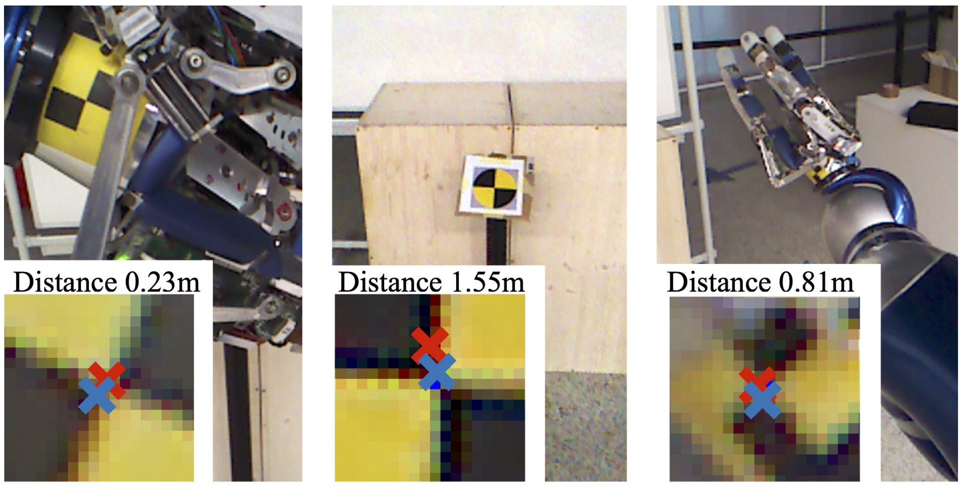


Figure 2.19: DLR's Agile Justin collects measurements to calibrate its non-geometric forward kinematics. The images are from the robots' internal RGB camera with a resolution of 640 × 480, showing examples for the left arm, the pole, and the right arm. The markers' distances to the camera vary between measurements from 0.2 m up to 1.5m. Without a correction (red), the pixel error is uniformly distributed over the distances, leading to more significant cartesian errors for detections further away from the camera as pixels here correspond to a larger area. Our TOW (blue) counteracts this and improves the cartesian accuracy.

laboratory with dedicated measurement equipment. Furthermore, this setup is closer to the robot's intended application. Here, the robot uses its internal cameras to model the environment [WFB13] and localize objects for interacting and grasping. Therefore, an accurate calibration of the hand-eye chain together with the flexible torso is crucial for successful task execution.

The task measurement function stays cartesian $h_\mathrm{t} = h_\mathrm{v}$, but the actual setup measures the markers pixels in the image space $h_\mathrm{a} = h_\mathrm{c}$ (see (2.41)). This discrepancy has implications for the parameter identification. Agile Justin is a complex mechatronic system; even the elastic robot model does not capture all the relevant effects. Without adjustment, the MAP approach tries to distribute those remaining errors equally in the image space, not the cartesian workspace. We use the proposed weighting through the adapted covariance (see Section 2.3.3) to counteract this. This general formulation, which weights the individual samples based on the relevance to h_v, results in a weighting factor $\propto \frac{1}{z^2}$ for the image measurement function h_c. Samples farther away from the camera are weighted stronger, as the same pixel error at a larger distance relates to a more significant cartesian error.

This weighting helps distribute the error evenly in the task-relevant cartesian space (see Table 2.3). While the pixel error increases slightly, the mean and maximal cartesian error gets smaller by over 20 % when correcting for the mapping between the image and task space. Furthermore, we show that it is possible to include the camera intrinsics Θ_ci

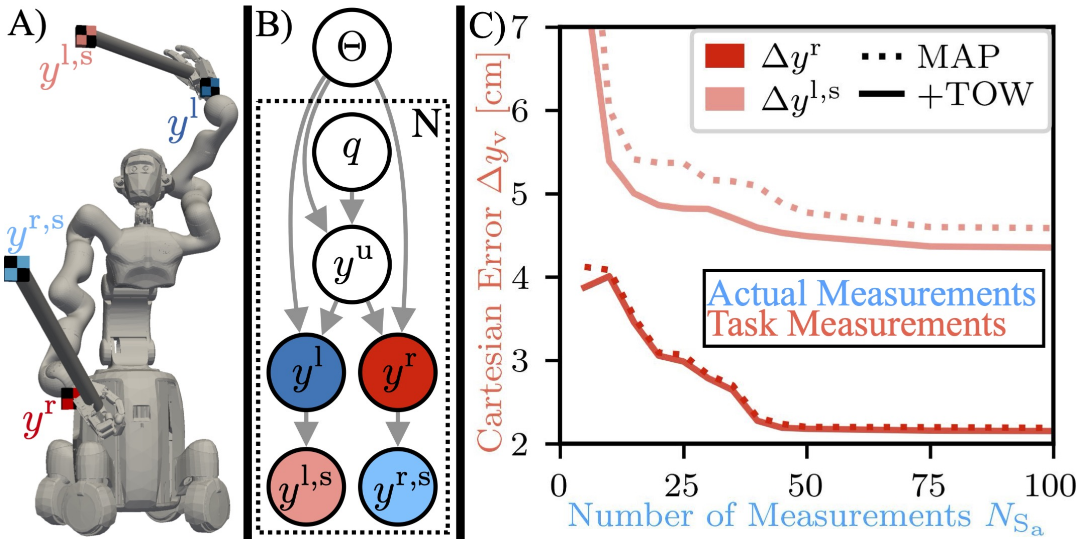


Figure 2.20: Study to show the generality of our TOW approach. *A)* Agile Justin with unmodeled elasticities holding two 0.7 m long sticks. The actual measurements (red) are the position of the left wrist $h_{\mathrm{v}}^{\mathrm{l}}$ and the end of the right stick $h_{\mathrm{v}}^{\mathrm{r,s}}$. The task measurements (blue)are crosswise the right wrist $h_{\mathrm{v}}^{\mathrm{r}}$ and the end of the left stick $h_{\mathrm{v}}^{\mathrm{l,s}}$. *B)* Probabilistic graph of the calibration setup with branching from the upper body y_{u} into the two sides. *C)* Comparison between standard MAP (dotted) and our TOW approach (solid, see Section 2.3.3) on the cartesian task error over different actual calibration set sizes. Especially for the marker on the end of the stick the error reduction with TOW on the task set is significant.

in the calibration, further improving accuracy. The final mean error at the end-effectors is 3.9 mm on average and 9.2 mm in the worst case. These results are comparable to a calibration using the cartesian measurements of an external tracking system, which is reported in the last row.

2.6.4 Agile Justin with Sticks

We simulated a calibration of Agile Justin in a more challenging setting to demonstrate the general applicability of our TOW approach. Fig. 2.20 shows in A) the setup with the robot holding two 0.7 m long sticks and in B) the respective probabilistic graph. The robot is calibrated using an external tracking system $h_{\mathrm{a}} = h_{\mathrm{v}}$. The actual measurements (red) are the position of the left wrist $h_{\mathrm{v}}^{\mathrm{l}}$ and the end of the right stick $h_{\mathrm{v}}^{\mathrm{r,s}}$. The task measurements (blue) are crosswise the right wrist $h_{\mathrm{v}}^{\mathrm{r}}$ and the end of the left stick $h_{\mathrm{v}}^{\mathrm{l,s}}$. Intuitively, focusing on the extended sticks means that with a longer leverage, errors in the orientation outweigh the errors in the position. Using our TOW approach, we can adjust the calibration to find an optimal set of parameters for the task space, even in this convoluted setting.

500 variations of Agile Justin were generated by adding noise to the forward kine-

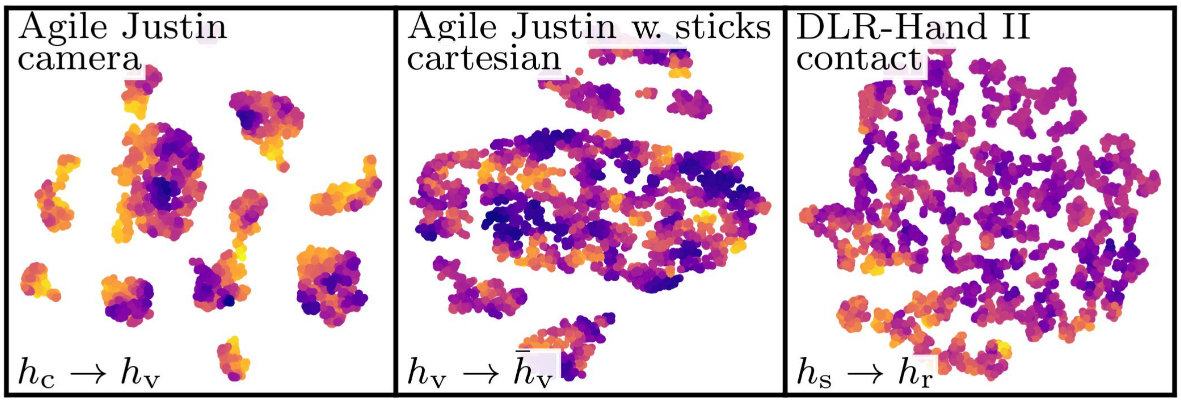


Figure 2.21: This graph visualizes the different calibration sets and the weighting of individual configurations stemming from Section 2.3.3. From left to right, the actual calibration setups are Agile Justin with its internal RGB camera (h_c), Agile Justin with an external tracking system (h_v), and the DLR-Hand II using the contact information (h_s). The high-dimensional configuration spaces were mapped into 2D using standard t-SNE [VH08].

matic's nominal DH parameters. As minimization in the task space is only relevant if the model is imperfect, we generated data including elasticities. However, the calibration model only contains the geometric DH parameters. In the experiments, we analyze how those purposely unmodeled errors get distributed for different measurement functions and weightings. Fig. 2.20 shows in C) that our TOW approach of task space minimization helps increase the task's accuracy. The remaining errors of the task cartesian measurement at the right wrist h_v^r and the left stick $h_\mathrm{v}^\mathrm{l,s}$ are drawn over different calibration set sizes for a standard MAP (dotted) and our additional TOW (solid). The calibration itself is always performed on actual measurements, while the evaluation is done on a different task set of configurations. The errors at the stick ends are significantly larger overall. The longer the leverage is, the more the errors in the parameters become apparent. While TOW performs slightly worse on the actual calibration set, there is an apparent reduction of cartesian task error, especially at the end of the left stick. This improvement over the standard MAP is especially prominent for small calibration sets. However, it is still visible for larger calibration sets, with a final reduction of 10 % from 4.9 cm to 4.4 cm for $h_\mathrm{v}^\mathrm{l,s}$.

Fig. 2.21 shows a visualization of our proposed weighting for the individual measurement configuration. The calibration set for Agile Justin's internal RGB camera is on the left, the setting of Agile Justin with markers on extended sticks is in the center, and on the right is the set for the contact-based calibration of the DLR-Hand II. The dimensionality reduction of the high-dimensional joint spaces to 2D using t-SNE [VH08] shows the sparse and scattered structure of the available subspaces for each actual measurement setup. The projection from cartesian to image space in the left image relates to the smooth and continuous distribution of the sample weighting, as here, the weighting is proportional to the marker's distance from the internal camera. For the contact-based calibration, configurations with different weights can lie closer together, as the relation between a change in the scalar distance between two surfaces and the cartesian distance of the two frames is essential here. Independent of the specific relation between h_t and h_a, as long as both measurement functions can identify all relevant parameters, task-oriented weighting helps to minimize unmodeled errors in the relevant task space.

2.7 Summary

Our work offers a unified approach to robot calibration and tackles the misalignment between the actual measurement setup and the final robotic task. Acknowledging those differences in the measurement functions and the configuration sets allows us to select practical measurement setups while keeping the task's performance in mind. Our contributions formalize these differences and help to mitigate them. Starting with a sensitivity analysis, one can test if the calibration setup can capture all information relevant to the task. Central is the introduction of TOW to combine the full probabilistic view with an additional weighting to reduce the remaining errors in the desired task space for arbitrary measurement setups and select the best model parameters to approximate the desired function. Finally, we discussed selecting an optimal set of measurement configurations for a task defined by a measurement function and a corresponding set of configurations. Extensive simulation studies, as well as real-world experiments on Agile Justin and the DLR-Hand II, demonstrate the soundness and broad applicability of the proposed methods.

Future Work

Future work in robotic calibration could explore advanced modeling approaches, such as polynomial functions or simple neural networks, to effectively capture non-geometric effects. While those models are more expressive, an important aspect will be their efficient compensation and easy applicability. Additionally, a Bayesian evaluation of parameter stability over time is promising. The overall goal is to integrate the information of prior and current measurements more tightly to minimize the effort for recalibrations and make them more efficient. A further route is to combine the kinematic calibration firmer with the calibration of other internal sensors. Examples are torque sensors in the motors or tactile skin on the end-effectors. Moreover, calibrating the entire body with all frames and the surface of all links is highly relevant, instead of just the mapping to the end-effector. Overall, a holistic approach to calibration is beneficial, primarily if the robot uses multiple sensors and body parts, as it is the case for the fingers of a hand when manipulating an object.

3 Learning-Based Motion Planning

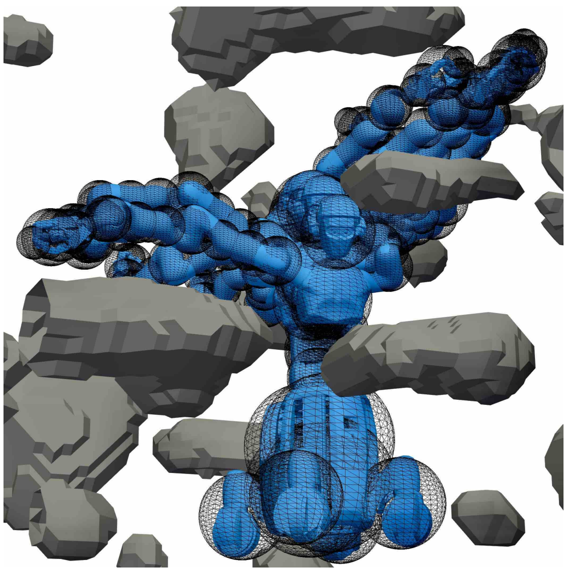
3.1 Introduction

At the core, robotic motion planning is about getting from the start joint configuration to a goal configuration while avoiding obstacles in the environment and self-collision along the path. For the most common robotic tasks, such as picking and placing objects, one does furthermore need the solution to the inverse kinematics (IK) to know how to configure the robots' joints to reach a specific position in the workspace. In general, motion planning and IK are a combined problem. For example, positioning a cup upright on a cluttered table requires motions with cartesian constraints at the end-effector. This joint problem of solving the IK and getting a collision-free trajectory is challenging, so it is often divided into two sub-problems [Sch+14]. First, the IK is solved to find the final configuration for grasping the object, and then the trajectory from the initial configuration to the goal configuration is planned.

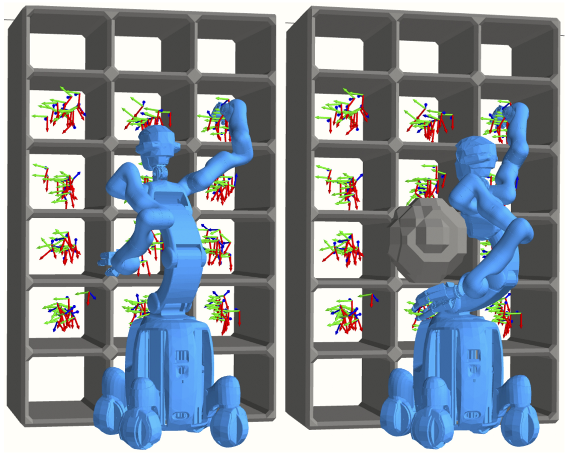
Solving a motion task and IK fast and efficiently does not only mean that the robot spends less time contemplating and more time moving. If a solver can find a feasible solution in a fraction of a second, this opens up the door for more reactive planning and integrating the global planner more tightly into the sensor-action loop. Furthermore, we can tackle more high-level tasks with multiple smaller motion problems if each substep can be solved efficiently.

A promising approach for speeding up motion planning and inverse kinematics is not to solve each planning problem anew but to use experience from having solved related motion tasks before. Important for the applicability of such learning-based planners to real-world problems is that they not only allow for arbitrary start and goal configurations but also for arbitrary environments as input. The goal is to encode the robot's specific kinematic structure and how it relates to blocked and free space in the workspace.

This chapter presents two deep learning-based enhancements for an optimization-based motion planner and IK solver that allows robot planning in high-resolution environments with complex obstacle geometries. We tackle problems of efficient data generation, a suitable encoding of the problem, and the inherent ambiguity of inverse kinematics. For the humanoid robot Agile Justin [Bäu+14] with 19 DoF, feasible trajectories can be computed in only 200 ms on a single CPU core (see Fig. 3.1a). IK solution gets especially hard when incorporating self-collision avoidance and avoiding collisions with obstacles in arbitrary environments (see Fig. 3.1b, right).



(a) **Motion Planning.** Agile Justin is moving along a collision-free path Q in a challenging obstacle environment generated with simplex noise. The sphere model of the 19 DoF humanoid is shown as wireframes.



(b) **Inverse Kinematics.** Agile Justin in a shelf environment. The frames for the IK problem were sampled randomly in the respective compartments of the shelf (*left*). Placing an additional obstacle in the workspace makes the previous solution infeasible, which leads to a different collision-free solution (*right*).

Figure 3.1: Motion Planning and IK for Agile Justin [Bäu+14] during training and application in challenging environments.

3.2 Related Work

The following section gives a combined overview of the related work on robot motion planning and inverse kinematics. The problems are challenging because they involve handling the non-linear structure of the robot's kinematics and understanding free and blocked space in unknown environments. However, a fast and efficient solution to those problems is crucial for safe and dextrous motion and interaction with the environment.

Motion Planning and Inverse Kineamtics

The goal is to have a fast and reactive motion planner on a low level that can plan between arbitrary robot configurations while avoiding collisions with the environment and self-collision [Bel+07]. Two popular but fundamentally different approaches to solving a motion task in robotics are sampling-based planning (SMP) and optimization-based motion planning (OMP).

Sampling-based Motion Planning SMP [LaV06] uses randomness at its core and can guarantee to find a feasible collision-free path globally if there is any. Examples are

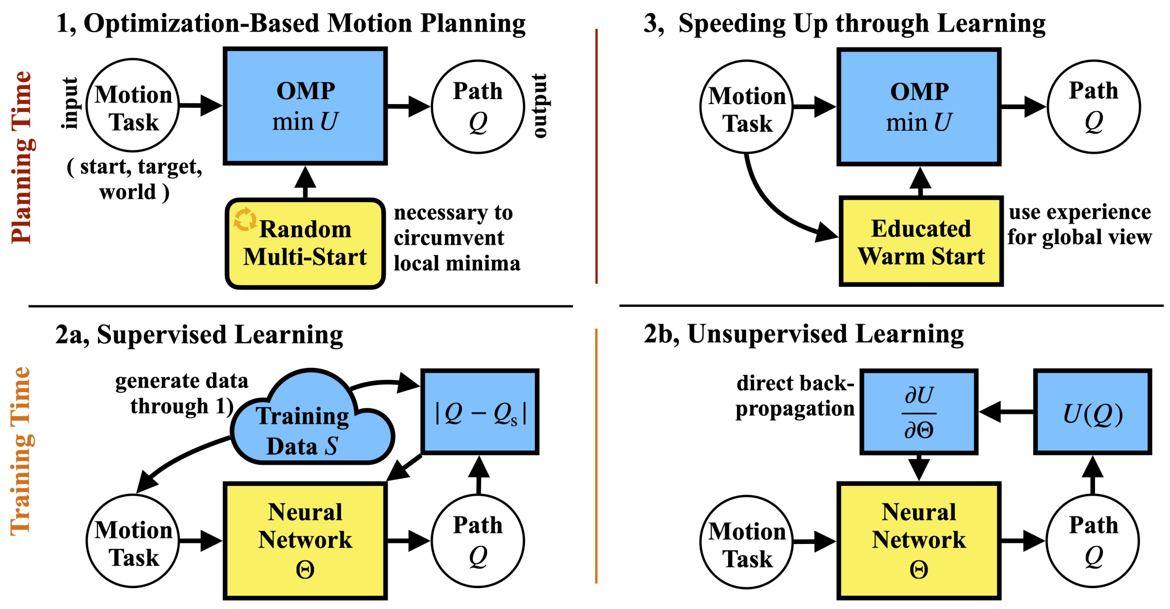


Figure 3.2: The standard OMP approach is on the top left (1, Section 3.3). The local method iterates on an initial guess to converge to a solution. The idea is to replace the random multi-starts through a learning-based guess to speed up the planning (3, Section 3.4). The classical approach in 1) can be used to generate training data for the neural network to encode the experience via supervised learning (2a, Section 3.4.3). Alternatively, one can apply an unsupervised approach and use the objective U directly to update the weights of the network directly through backpropagation (2b, Section 3.4.4). All figures hold for motion planning and inverse kinematics.

Probabilistic Roadmaps [Kav+96] and Rapidly exploring Random Trees (RRT) [KL00], which explore the configuration space iteratively and build a graph of possible configurations until a branch finds the goal. As the search space grows exponentially with the robots' degrees of freedom, vanilla variants of SMP do not scale well to complex robots. Furthermore, SMP often produces suboptimal paths with many kinks, leading to lengthy and dynamically costly motions. Extensions to RRT address those issues partially by using a heuristic to search in an ellipsoid around the direct connection [GSB14], focusing the computations after an initial solution on improving the optimality [Kar+11] or using the implicit geometric structure of a batch of random samples [GSB15].

Optimization-based Motion Planning OMP formulates the motion task as an optimization problem [BKC08], and one can apply various non-convex optimization techniques to find a solution. STOMP [Kal+11] is a gradient-free method that uses stochastic information to make iterative updates on an initial trajectory. ITOMP [PPM12] also uses a stochastic optimization framework for dynamic environments by computing a local bound for each obstacle. CHOMP [Zuc+13] uses a covariant Hamiltonian formulation to make gradient updates perpendicular to the geometry of the current path. TrajOpt [Sch+14] uses an SLSQP approach that sequentially solves quadratic problems with linearized constraints to converge to an optimal solution. Furthermore, newton-based methods [Tou14] and methods that use particle swarm optimization (PSO) [KL15] have been applied to robot motion planning. Another challenging aspect of motion planning includes additional dynamic constraints [CLS16; PM14] or non-differential constraints like contacts or collisions [ZLB21].

The computational complexity of those methods can scale linearly with the DoF and converge quickly to a smooth trajectory. For gradient descent at each iteration, only the gradient computation and its addition to the current configuration are performed. Both operations are linear in the DoF when using automatic differentiation. However, they only find a local minimum and strongly depend on the initial guess.

The objective function usually has many local minima for non-trivial robot kinematics and environments. Therefore, a multi-start approach is needed to find a feasible global solution that will massively slow OMP in complex scenarios. STOMP [Kal+11] and CHOMP [Zuc+13] try to mitigate the strong dependency on an initial guess by introducing stochasticity to the optimization. TrajOpt [Sch+14] uses convex hulls to represent the robot and its environment and thus increases the attraction basin for each optimum. However, OMP still needs multi-starts to find a feasible global solution.

Jacobian-based Inverse Kinematics There exists a multitude of non-learning-based methods to solve the IK problem. Many are based on the Inverse Jacobian method [TS00; SB11; Sug11; CT15]. A popular algorithm is TRAC-IK [BA15], combining a Newton-based algorithm with a Sequential Quadratic Programming (SQP) method. Running both methods in parallel and terminating if one succeeds improves speed and robustness. Quick-IK [Lia+17] also utilizes parallelism and simultaneously tries many speculative initial guesses. Falco et al. [FN11] analyze the general convergence of Inverse Jacobian methods and report the results for an 11 DoF robotic arm.

However, other optimization methods have also been applied to the IK problem. Collinsm et al. [CS17] use PSO for snake-like robots with many DoF. Trutman et al. [Tru+22] describe the IK as a polynomial optimization problem, which they use to find a globally optimal solution for serial 7 DoF robots. Tringali et al. [TC20] use a randomized matrix to weight the pseudo inverse in a Jacobian-based method. This adaptation allows them to improve the convergence to a globally optimal IK solution.

Ferrentino et al. [FSC21] use dynamic programming to solve the IK with obstacles and make it available in ROS. Giamou et al. [Gia+22] formulate the IK problem as a distance-geometric problem, allowing them to use semidefinite programming methods to find a low-rank solution. While the approach is elegant and fast, it can only incorporate spherical obstacles. Zhao et al. [Zha+21] introduce a modern and fast solver that combines Inverse Jacobian methods, SQP, and PSO. Their method can handle dynamic obstacles but is limited to spheres.

Learning for Motion Planning and Inverse Kinematics

Motion Planning Whether the underlying approach is optimization-based or search-based, there are different options for incorporating learning and speeding up the methods efficiently. For SMP, a common idea is to adapt the sampling strategy of new leaves in the search tree. [IHP18] proposes a methodology for non-uniform sampling, whereby a sampling distribution is learned through a conditional variational autoencoder (CVAE) from demonstrations and then used to bias sampling. When sampling a small percentage of the new leaves uniformly, one can keep the completeness guarantees while speeding up the convergence significantly by focusing on relevant regions in the configuration space. For mobile robots in 2D, this non-uniform sampling distribution can be directly represented by a CNN [Wan+20].

An elegant approach to speed up OMP is using experience from previously solved motion tasks to provide an *educated initial guess*. Jetchev et al. [JT13] proposed saving a database of feasible trajectories and looking for a reasonable first guess for a new problem. Merkt et al. [MIV18] improved the idea through more efficient database storage and tested it on a humanoid robot.

Instead of databases, neural networks are often used to encode the experience. The expectation is that they are more memory efficient, encode various solutions implicitly, learn a general understanding of feasible trajectories, and produce valuable predictions in unseen settings. Qureshi et al. [Qur+19] coined the term Motion Planning Network (MPNet) and then improved on the idea [Qur+21]. They used an RRT planner to collect feasible trajectories in 2D and 3D worlds and encode the environment with point clouds. Their method works on the Baxter robot for ten known table scenes with 1000 paths per scene. Strudel et al. [Str+20] showed that they could outperform these results by employing the PointNet [Qi+17] architecture to encode the point clouds of the environments. They achieved good results in 3D with a sphere and a rigid S-shape with three translational and rotational DoFs but did not consider a robotic application. Bency et al. [BQY19] and Lembono et al. [Lem+20] applied variations of the idea successfully to the two humanoids, Baxter and PR2. However, they only used a single fixed environment without generalization to different worlds. Lehner et al. [LA18] trained a Gaussian mixture model to steer the search in a probabilistic roadmap. The approach was demonstrated on a real 7 DoF robot but equally only for one fixed world.

Some works combine the whole pipeline from visual input to the motor actions into a single end-to-end learning problem [Pfe+17]. The visual input can also be used to make data-driven predictions on how a robot's actions will change the environment [FL17]. This visual approach is especially well suited to navigation problems in 2D, as all the tools of computer graphics and image processing can be applied directly [LMT19]. Still, those cannot scale directly to more complex robots in 3D.

Other learning methods have also been applied to this problem. For example, Jurgenson et al. [JT19] used reinforcement learning with convolution layers to process the occupancy map of the world for 2D serial robots. They invoke a classical planner only for cases where random exploration fails to find a feasible solution, so they implicitly use this expert knowledge to guide the training. Pandy et al. [PLC21] introduce a different approach, where the dataset generation is skipped entirely, and the network is directly trained using the objective function of the planning problem as the training loss, i.e., no supervision is used. However, they only use geometric primitives to represent environments with few obstacles, limiting the flexibility. Diffusion models were also applied to robot motion planning [Jan+22]. While their generative capabilities are promising, conditioning them on realistic environments is still challenging.

Inverse Kinematics There are also many learning-based approaches for solving the IK problem more efficiently. One problem all those methods suffer from is the sizeable nullspace of possible configurations [KRS18] and the inherent ambiguity of the IK solution. An early idea to make learning in the vast configuration space faster and more manageable was to use function decomposition [AT05]. Goal babbling [RSG10] permits us to explore this space more efficiently and without the direct need for a large amount of expert knowledge. Structuring and combining predictions can help to tackle ambiguous solutions [Boc+11], and using the underlying symmetries of the IK problem is essential for a good generalization [RSG09]. It can be beneficial to split the robot kinematics into distinct parts and learn the inverse mapping for similar parts separately [AZ19]. To improve the speed and portability of IK methods, Zaidel et al. [Zai+21] introduce a neuromorphic approach that they apply to a 7 DoF robot arm. Combining neural networks and fuzzy hints is another possibility to learn joint-wise inverse kinematics [AWI06] or the complete inverse mapping for the whole workspace [DGD19]. Further complexities arise when the kinematic is not purely geometric, but secondary aspects like elasticities need to be modeled [CL16].

All learned IK methods described so far consider an obstacle-free working environment. Lehner et al. [LRA22] leverage transfer learning between similar robot kinematics in a single environment with obstacles. They use the network predictions to guide a Rapid Random Tree (RRT) motion planner. IK can also be understood as a means to avoid self-collision. The nullspace of redundant robots can be used to move different body parts out of collision [SAL15; HLH24]. Another option is to use Generative Adversarial Networks (GANs) to learn constrained robot configurations [Lem+21]. They use the predictions of those networks as an initial guess for an optimization-based planner to warm-start the IK problem and as samples for an RRT motion planner. They consider the environment for their tasks, but for each new scene, an ensemble of GANs needs to be trained to counteract the mode collapse and produce valuable samples.

Obstacle Collision

Avoiding obstacles in the environment is a central aspect of robot motion planning and IK. The combination of a complex kinematic structure and narrow and cluttered environments subdivides the solution space into many separate valleys. This fragmentation is a challenge for both local optimization-based approaches and search-based methods.

The first step in including obstacle avoidance in motion planning is to encode the environment. Gilbert et al. [GJ85] introduce the concept of distance functions and their applications to robot path planning. Potential Fields [RK92] work similarly and can be used to represent actual obstacles or to model additional virtual constraints of the planning problem.

For learning-based methods, one needs to encode the environment for neural networks. Only then can they learn the spatial relations efficiently and make valuable predictions in unseen and challenging environments. While point clouds are a typical representation coming directly from the robot's sensors, they are not directly usable for an NN because they are not permutation invariant. PointNet [Qi+17] directly tackles this issue by adding an invariant operation at an early high-dimensional feature space. Another option is to fuse the depth information into a binary occupancy map [WFB13]. Especially when working with larger environments with many relevant details at different resolutions, data structures like OctTrees are helpful. OctNet [RUG17] brings this hierarchical concept into the machine learning domain, leading to a more efficient encoding of deep and high-resolution maps.

A different route is not to represent the high-resolution sensor inputs but to find a more compact representation. Composing a map into a collection of convex objects would be especially suitable. Convex objects can be implicitly interpreted as a collection of half-space constraints or support functions, which makes them especially suited for neural networks. On the other hand, a fast and accurate decomposition of an arbitrary map enables well-established methods for distance and collision-checking, which also speeds up motion planning in general [Sch+14]. Deng et al. [Den+20] demonstrate how to learn this decomposition for a diverse set of objects, but not yet on the scale of whole maps of challenging environments.

A robot-centered idea to reduce complexity is to learn the SDF of each link individually [Liu+23]. This separation allows one to directly query the distance from the 6D position of a body part to an arbitrarily placed point in the environment, sidestepping the need to learn the kinematic structure and coupling of the robot's joints.

For complex mechatronic systems, efficient *self-collision avoidance* is crucial to motion planning and IK. In a naive approach, every body part must be checked against all the others, which is expensive, especially for robots with tree structures like humanoids. SCAFoI [Fan+15] predicts and dynamically selects the relevant link pairs to be checked online to improve the computation efficiency. A support vector machine (SVM) predicts the status of the relative positional relationship for each region of interest.

A central concept in collision avoidance is the *swept volume*. This measure describes the workspace volume the robot moves through when following a path. It can be used to avoid self-collision efficiently [TBF11] or to reduce the problem complexity by focusing on the irreducible pace space [Ort+18]. While the swept volume is valuable, its accurate computation for complex robots is often expensive. Therefore, learning-based approaches to predict the swept volume [Bax+20; Chi+21] between two joint configurations are a promising alternative to estimate the covered area of a motion.

Contributions

While many learning-based approaches to motion planning and IK exist, none of them strongly focus on generalizing to arbitrary unseen environments. With the adaption of

Basis Point Sets (BPS) [PLR19] as world encoding into the robotics domain and efficient data generation and training schemes, we introduced a learning-based motion planner [4] and a learning-based IK solver [5] that can handle complex robots in challenging unseen environments. For the humanoid Agile Justin, the high-resolution environment model is generated in real-time from sensor data [WFB13]. We can now use the BPS encoding of this model directly to plan fast and efficiently in those environments. This quick motion and IK planning on the self-acquired voxel maps is the basis to safely combine the different skills of Agile Justin to dextrous grasping and manipulation tasks [Ten+24].

Our main contributions are:

- The introduction of the BPS [PLR19] into learning-based motion planning and inverse kinematics. This memory- and computational-efficient encoding enables training and generalization for complex robots in challenging environments.
- A fast learning-based motion planner between arbitrary configurations for complex previously unseen environments (200 ms for the 19 DoF humanoid Agile Justin on a self-acquired high-resolution voxel grid).
- A learning-based fast and accurate solver for IK (for humanoid Agile Justin an accuracy of 10^{-4} m and 10^{-3} rad in 10 ms).
- Building challenging training and testing environments using Simplex [Gus05] noise. The worlds can be autogenerated with configurable complexity.
- A supervised training scheme, where we combine the network and the objective function as a metric to clean, extend, and boost an initial dataset efficiently.
- An unsupervised training scheme where the objective function is used directly to train the weights of a neural network, skipping data generation entirely.
- Extensive experiments in simulation for different robots ranging from a simple 2D sphere bot to the 19 DoF humanoid Agile Justin in 3D.
- A report of first real-world experiments for Agile Justin, showing the successful Sim2Real transfer of the training on Simplex worlds.
- A detailed analysis of the challenges in learning ambiguous IK with collision avoidance and the resulting network, including a twin-headed architecture, a singularity-free output representation, and boosting.
- The optimal solution to the IK problem varies not smoothly across the workspace. We show that two heads are enough for a network to predict the sharp switches between those regions of different modes.
- A benchmark of the supervised and the unsupervised learning approach shows a ten times faster training time for the latter and a more straightforward training procedure while outperforming the random baseline significantly.

3.3 Optimization-Based Motion Planning on Voxel Models

In this section, motion planning is formalized. The goal is to move fast and collision-free through the world. One can distinguish between moving to a given joint configuration and moving to a cartesian point in the workspace. We humans know the former from dancing or acrobatics, where the body should move into a particular posture. The latter is common when interacting with the world. When grasping an object, the important thing is to reach the object collision-free and not directly to the precise configurations of all joints and body parts.

The problem of finding a robot configuration that reaches a certain point in the workspace is called inverse kinematics (IK). As the combined problem of motion planning plus inverse kinematics is challenging, this work separates it into two distinct problems. First, solving the IK determines the final joint configuration that reaches the wanted point. In the second step, the motion from the current configuration of the robot to this new IK configuration is planned to move collision-free and smoothly through the world. We rely on an optimization-based motion (OMP) to compute collision-free paths [Zuc+13] and solutions to the IK. The basic idea is to formulate the motion problem as an objective function, where finding the optimum of this function is equivalent to finding an optimal path or IK.

Central to the problem formulation is the robot model, calibrated in Section 2.5.1. The forward kinematics maps from joint configurations $q \in \mathbb{R}^{N_\text{DoF}}$ to the all link frames

$$f(q) = \{F_i\}_{i=1}^{N_\text{F}}. \tag{3.1}$$

Each homogenous transformation matrix F_i describes a full 6D pose $(\text{Pos}_i, \text{Rot}_i) \in \mathbb{R}^3 \times SO(3)$. The position $\text{Pos}(F_i) = \text{Pos}_i$ and orientation $\text{Rot}(F_i) = \text{Rot}_i$ of the end-effectors and all frames of the kinematic tree are relevant to check for self-collisions and collisions with the environment. In the following, we formulate the different parts of the objective function for motion planning and the IK.

3.3.1 Path Representation with Substeps

To describe a motion over time Q, we use a discrete set of waypoints

$$Q = [q_1, \dots, q_{N_\text{t}}],\ q_t \in \mathbb{R}^{N_\text{DoF}}, \tag{3.2}$$

where the relevant terms to build a cost function can be computed for each joint configuration along the path. With a finer time resolution of the path, more detailed motions can be described. However, as those waypoints are the variables of the optimization problem, there is a direct trade-off between finer resolution and the optimization problem's dimensionality and, therefore, the required computation time.

In extension to the original CHOMP algorithm [Zuc+13], we subdivide the path be-

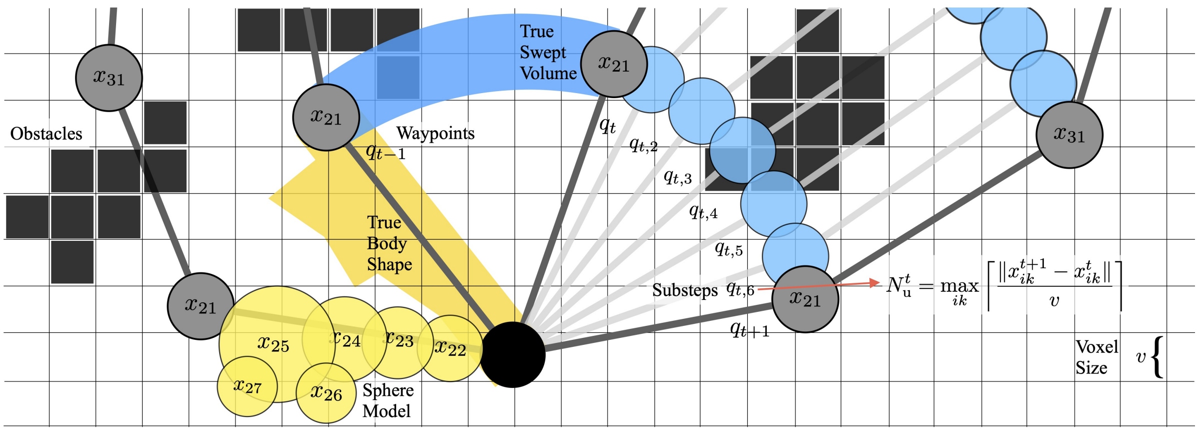


Figure 3.3: Discretization of a robotic arm's motion in space and time. Each robot body part is modeled as a collection of spheres (*yellow*) to detect collision with itself and the environment. The environment is modeled as a voxel model (*black*). To ensure all collisions are detected, we use substeps $q_{t,u}$ between two discrete waypoints q_t and q_{t+1} to explicitly calculate the swept volume of the path in higher resolution (*blue*).

tween two consecutive waypoints into substeps $q_{t,u}$ via linear interpolation (see Fig. 3.3).

$$q_{t,u} = q_t + \frac{u}{N_\text{u}}(q_{t+1} - q_t). \tag{3.3}$$

This way, a collision-free path can be guaranteed when the number of substeps N_u is adjusted to the step length and the resolution of the environment model. Also, other types of interpolation, such as B-splines, are possible. The essential advantage is that this formulation allows for a moderately small number of optimization parameters while still capturing all relevant aspects of the motion through the world. So, the swept volume of each sphere is *explicitly* (see Fig. 3.3) computed instead of the *implict* computation CHOMP performs via a projection of the cartesian velocity vector of the moving spheres.

3.3.2 Objectives for Short, Close, and Collision-Free Paths

Path Length

When ignoring all constraints like collisions, the essential criterion is that the path is short. One can directly express this in the configuration space by computing the squared Euclidian norm between two configurations

$$U_1^\text{q}(q_a, q_b) = \frac{1}{2}\|q_a - q_b\|^2. \tag{3.4}$$

Extending this to a path Q in configuration space, one can compute the length cost

$$U_{\mathrm{l}}^{\mathrm{p}}(Q) = \frac{N_{\mathrm{t}} - 1}{U_{\mathrm{l}}^{\mathrm{q}}(q_{N_{\mathrm{t}}}, q_1)} \sum_{t=1}^{N_{\mathrm{t}}-1} U_{\mathrm{l}}^{\mathrm{q}}(q_{t+1}, q_t), \tag{3.5}$$

which favors short and smooth trajectories. It is convenient to scale the length cost by the minimal possible path length, the direct connection between $q_{N_{\mathrm{t}}}$ and q_1. Then, the shortest possible path always has a cost of one. Scaling the different objective terms in such a fashion makes weighting them in a total objective function easier.

Another possibility to measure the distance between two configurations is in the cartesian workspace

$$U_{\tilde{\mathrm{l}}}^{\mathrm{q}}(q_a, q_b, e) = \frac{1}{2} \| \operatorname{Pos}(f(q_b)_e) - \operatorname{Pos}(f(q_a)_e) \|^2. \tag{3.6}$$

For this, it is convenient to focus on a relevant frame e, like the end-effector carrying a cup of tea, or it is possible to sum up the motion of all body parts. This extends analogous to for a path Q

$$U_{\tilde{\mathrm{l}}}^{\mathrm{p}}(Q) = \frac{N_{\mathrm{t}} - 1}{U_{\tilde{\mathrm{l}}}^{\mathrm{q}}(q_{N_{\mathrm{t}}}, q_1)} \sum_{t=1}^{N_{\mathrm{t}}-1} U_{\mathrm{l}}^{\mathrm{q}}(q_{t+1}, q_t). \tag{3.7}$$

One can also combine the metrics in the configuration space and in the workspace. Another related option is to use an energy-based formulation. Here, the robot should minimize the motion of its mass distribution.

Avoiding Obstacle Collision

To calculate the collision cost between the robot and the environment, we need a model for both. The forward kinematics $F = f(q)$ maps from joint configurations to the link frames F_i, and each link's geometry is described by a set of spheres $\boldsymbol{S}_i = \{x_{ik}, r_{ik}\}_{k=1}^{N_{\mathrm{s},i}}$ with centers and radii. The world is represented by an SDF $D(x)$, which gives the distance to the closest obstacle for each point x in the workspace. The collision cost is then given by the sum of all the collisions of the different body parts

$$U_{\mathrm{w}}^{\mathrm{q}}(q) = \sum_{i=1}^{N_{\mathrm{F}}} \sum_{k=1}^{N_{\mathrm{s},i}} c\Big(D\big(F_i(q) \cdot x_{ik}\big) - r_{ik}\Big). \tag{3.8}$$

When considering a path Q, one needs to sum over all waypoints and the substeps between them

$$U_{\mathrm{w}}^{\mathrm{p}}(Q) = \sum_{t,u}^{N_{\mathrm{t}},N_{\mathrm{u}}} U_{\mathrm{w}}^{\mathrm{q}}(q_{t,u}). \tag{3.9}$$

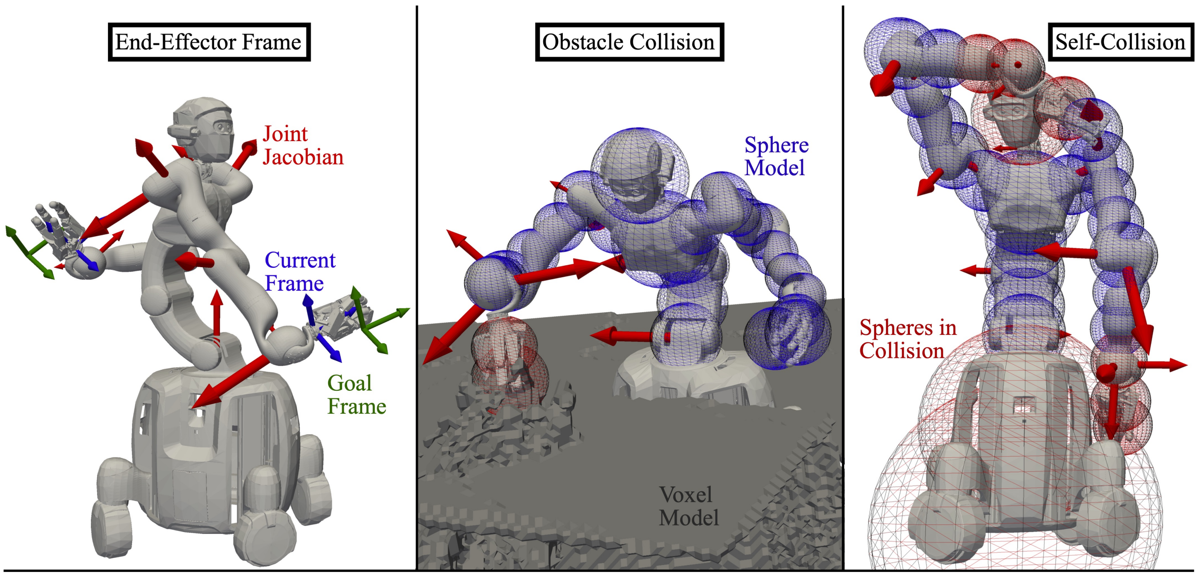


Figure 3.4: The three images show different constraints of motion planning plus the corresponding gradients with respect to the joints. Moving locally in these directions brings the robot closer to fulfilling these constraints.
Left: aligning the end-effectors with desired goal frames $U_{\mathrm{f}}^{\mathrm{q}}$ (3.15);
Center: avoiding obstacle collision with voxel model of the world $U_{\mathrm{w}}^{\mathrm{q}}$ (3.8);
Right: avoiding self-collision with the different body parts $U_{\mathrm{s}}^{\mathrm{q}}$(3.11).

The smooth clipping function c is introduced to transform the inequality into an equality constraint, which is then written as an additional cost term in the objective [Sch+14]

$$c(d) = \begin{cases} -d + \frac{\epsilon}{2} & \text{, if } d < 0 \\ \frac{1}{2\epsilon}(d-\epsilon)^2 & \text{, if } 0 \le d \le \epsilon \\ 0 & \text{, if } \epsilon < d \end{cases} \tag{3.10}$$

It considers only the parts of the robot that are in collision by setting positive distances to zero. Thus, a collision-free solution has a cost of zero.

Avoiding Self-Collision

In addition to collisions with the world, complex robots must also account for self-collision. These constraints become challenging, especially when dextrously working with multiple arms in a small workspace. Using the same sphere model of the robot as for the obstacle collision, the cost sums up all the collisions between the different body pairs

$$U_{\mathrm{s}}^{\mathrm{q}}(q) = \sum_{j>i}^{N_{\mathrm{F}},N_{\mathrm{F}}} \sum_{k,l}^{N_{\mathrm{s},i},N_{\mathrm{s}j}} c\Big(\| F_i(q)\cdot x_{ik} - F_j(q)\cdot x_{jl}\| - (r_{ik} + r_{jl})\Big). \tag{3.11}$$

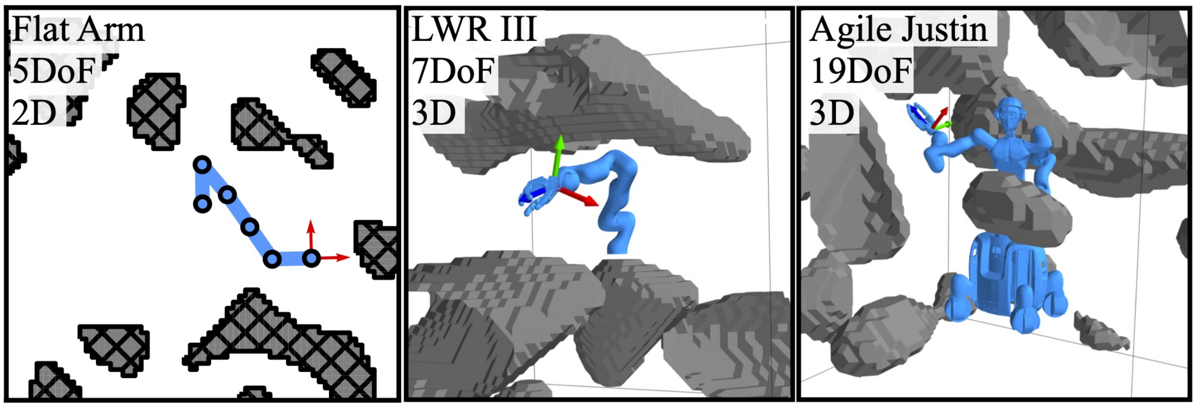


Figure 3.5: The three robots used in the experiments in environments generated with Simplex Noise [Per01]. The Flat Arm in 2D helps to analyze and visualize the IK problem in detail. The LWR III and Agile Justin demonstrate the capabilities of our method for complex robotic systems.

As the number of checks grows quadratic with the number of spheres in the robot model, it is often helpful to model the robot with fewer convex shapes or use multiple layers of accuracy for the collision model. This second approach keeps the computations straightforward but limits the computation of all detailed spheres to cases where the coarser representation already detected a collision. Here again, we use substeps to guarantee collision-free paths by explicitly checking the swept volume along a path Q

$$U_{\mathrm{s}}^{\mathrm{p}}(Q) = \sum_{t,u}^{N_{\mathrm{t}},N_{\mathrm{u}}} U_{\mathrm{s}}^{\mathrm{q}}(q_{t,u}). \tag{3.12}$$

Reaching End-Effector Target

The equality constraint for the IK is that the distance between a specific frame in the chain F_e and a target frame $\bar{F}$ is zero. This can be expressed as an objective function with a translational part

$$U_{\mathrm{p}}^{\mathrm{q}}(q, \bar{F}, e) = \frac{1}{2}\|\operatorname{Pos}(f(q)_e) - \operatorname{Pos}(\bar{F})\|^2 \tag{3.13}$$

and a rotational part

$$U_{\mathrm{r}}^{\mathrm{q}}(q, \bar{F}, e) = \frac{1}{2}(3 - \operatorname{Trace}(\operatorname{Rot}(f(q)_e) \cdot \operatorname{Rot}(\bar{F}^{-1}))). \tag{3.14}$$

The combined objective term is just the weighted sum of those two terms

$$U_{\mathrm{f}}^{\mathrm{q}}(q, \bar{F}, e) = \lambda_{\mathrm{p}} U_{\mathrm{p}}^{\mathrm{q}}(q, \bar{F}, e) + \lambda_{\mathrm{r}} U_{\mathrm{r}}^{\mathrm{q}}(q, \bar{F}, e). \tag{3.15}$$

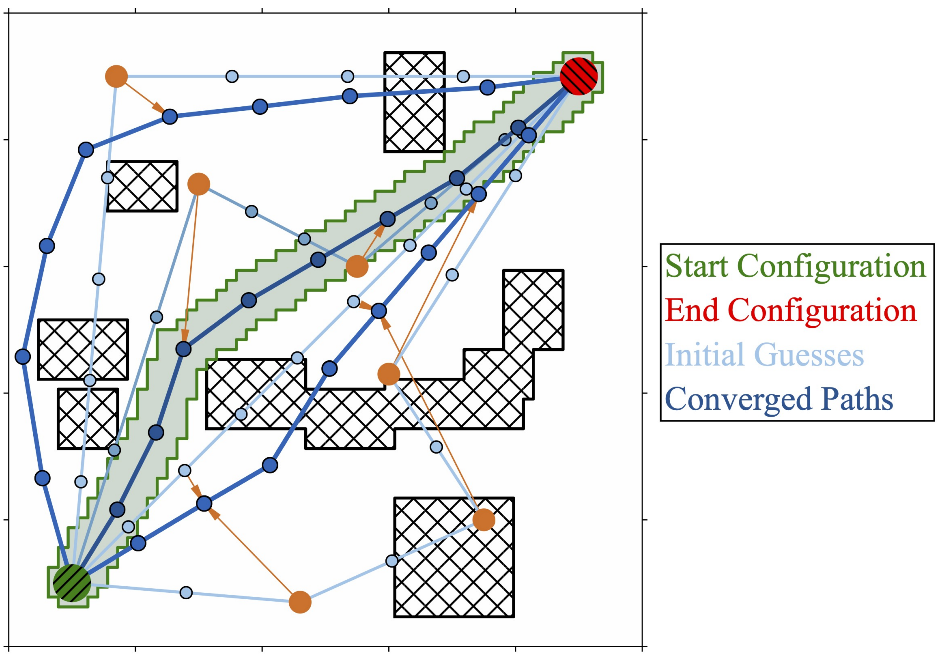


Figure 3.6: Gradient Descent to solve a simple motion Problem. Even in 2D with a sphere robot, multi-starts are necessary to find a global solution.

3.3.3 Combined Objectives for Motion Planning and Inverse Kinematics

Motion Planning

After formalizing the individual aspects of robotic motion planning, we can write the overall objective function as a weighted sum of

$$U^{\text{path}}(Q) = \lambda_{\text{w}} U^{\text{p}}_{\text{w}}(Q) + \lambda_{\text{s}} U^{\text{p}}_{\text{s}}(Q) + \lambda_{\text{l}} U^{\text{p}}_{\text{l}}(Q). \tag{3.16}$$

Note that the weighting factors can be normalized independently of the robot and environment and are mainly to ensure the higher importance of the collision terms over the secondary objectives like length. The advantage of this formulation is that it is easy to incorporate additional goals like energy efficiency or avoiding certain workspace regions.

With this formulation, the optimal path Q^* and solution to the motion task is the one with the lowest objective value

$$Q^* = \arg\min_{Q} U^{\text{path}}(Q). \tag{3.17}$$

A direct approach to find a solution of this optimization problem for iteratively finding a minimum of the objective function, starting with the initial guess Q_0,

$$Q_{i+1} = Q_i - \alpha \frac{\partial U^{\text{path}}(Q_i)}{\partial Q_i}. \tag{3.18}$$

We use standard gradient descent (see Fig. 3.6), as efficiency in the OMP part is not the primary concern of our method, and it allows for easy adaption and parameterization.

Inverse Kinematics

In the overall objective U^{ik} for the IK, one part is concerned with the frame at the end-effector $U_{\mathrm{f}}^{\mathrm{q}}$, and one part accounts for the collisions and additional objectives U_{A}:

$$\begin{aligned} U^{\mathrm{ik}} &= U_{\mathrm{f}}^{\mathrm{q}} + U_{\mathrm{A}} \\ \text{with } U_{\mathrm{f}}^{\mathrm{q}} &= \lambda_{\mathrm{p}} U_{\mathrm{p}}^{\mathrm{q}} + \lambda_{\mathrm{r}} U_{\mathrm{r}}^{\mathrm{q}} \\ \text{and } U_{\mathrm{A}} &= \lambda_{\mathrm{w}} U_{\mathrm{w}}^{\mathrm{q}} + \lambda_{\mathrm{s}} U_{\mathrm{s}}^{\mathrm{q}} + \lambda_{\mathrm{l}} U_{\mathrm{l}}^{\mathrm{q}}. \end{aligned} \tag{3.19}$$

While the mapping from the joint configuration to the end-effector frame is unique, the same does not hold for the inverse mapping. For an over-actuated robot, infinitely many joint configurations can reach a given frame in the workspace. However, one is usually not interested in an arbitrary solution but one which satisfies additional criteria. We introduce an additional term to the objective, namely the closeness $U_{\mathrm{l}}^{\mathrm{q}}(q, \bar{q})$ to a default configuration $\bar{q}$: minimizing $U_{\mathrm{l}}^{\mathrm{q}}$ makes the mapping unique and ensures that the solutions are close to the default configuration, making motion planning to this configuration faster and easier. The optimal configuration q^* and solution to the IK problem is the one with the lowest objective

$$q^* = \arg\min_q U^{\mathrm{ik}}(q). \tag{3.20}$$

Solver with Nullspace Projection

While this formulation as an optimization problem is complete and (3.19) is used to train the unsupervised networks in Section 3.4.4, it is often not efficient to solve this complex cost function jointly. To weaken the impact of competing terms in the objective function, we solve the IK problem in two steps.

First, we solve the pure IK with a projection step to ensure the constraints at the end-effector $U_{\mathrm{p}}^{\mathrm{q}}$ and $U_{\mathrm{r}}^{\mathrm{q}}$ are satisfied. This root search can be solved by iteratively applying the pseudo-inverse of the end-effector constraints:

$$\Delta p = \left[U_{\mathrm{p}}^{\mathrm{q}}(q)\,,\, U_{\mathrm{r}}^{\mathrm{q}}(q)\right] \tag{3.21}$$

$$J = \left[\frac{\partial U_{\mathrm{p}}^{\mathrm{q}}(q)}{\partial q}\,,\, \frac{\partial U_{\mathrm{r}}^{\mathrm{q}}(q)}{\partial q}\right] \tag{3.22}$$

$$q_{i+1} = q_i + J^{\dagger} \Delta p \tag{3.23}$$

For the humanoid Agile Justin, the IK requirements in the real world are to be accurate below 10^{-4} m and 10^{-3} rad. This numerical threshold is one order of magnitude more accurate than the actual accuracy of the calibrated system [1; 2].

In the second step, we apply gradient descent with nullspace projection to satisfy the collision constraints and optimize the additional terms in U_{A}. Each gradient step is

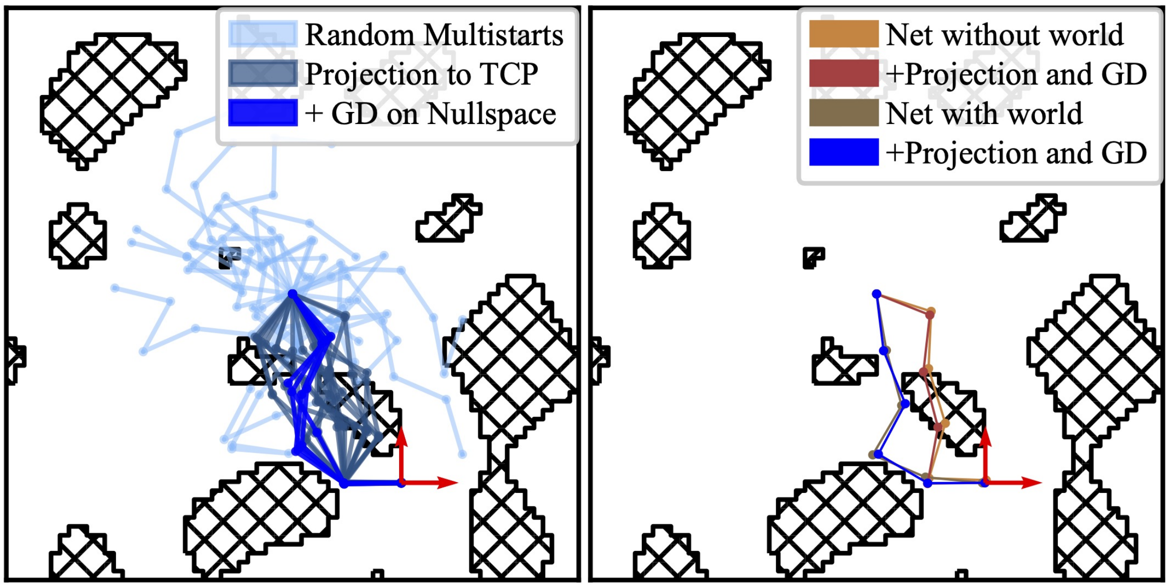


Figure 3.7: The graphic visualizes the two-stage solution process for a collision-free IK. The naive approach with random guesses is on the left, and the sped-up approach using the learned prediction of a neural network is on the right.

again projected on the IK manifold to ensure the constraints at the end-effector stay satisfied

$$q_{i+1} = q_i + (I - J^T(J^T)^\dagger)\frac{\partial U_\mathrm{A}(q)}{\partial q}. \tag{3.24}$$

These update steps push the configuration out of collision and closer to the default pose. While those gradient-based approaches are straightforward to implement and converge quickly for a given sample, they are also susceptible to the initial guess. Especially for complex robots and environments, multiple samples are necessary until a feasible solution is found.

Fig. 3.7 shows the steps of the two-stage IK procedure and compares random multistarts and the prediction of a neural network as initial guesses. In the *left image*, 20 initial random guesses of configurations (light blue) colors are used. These configurations are then projected onto the desired end-effector (red coordinate system) using (3.23), ensuring that translational and rotational constraints are satisfied (steel). Then, we compute the gradient of U_A and apply gradient descent inside the end-effector nullspace (3.24) to move the robot out of collision and closer to the default configuration (dark blue). After these steps, only five feasible solutions remain. This ratio gets even worse for more complex robots in challenging 3D environments, which Table 3.5 analyzes in more detail.

The *right image* showcases the same two-stage process when using the single prediction of an IK network but for two network variants. As the predictions of both networks are close to the desired end-effector, the projection step on the end-effector is not shown

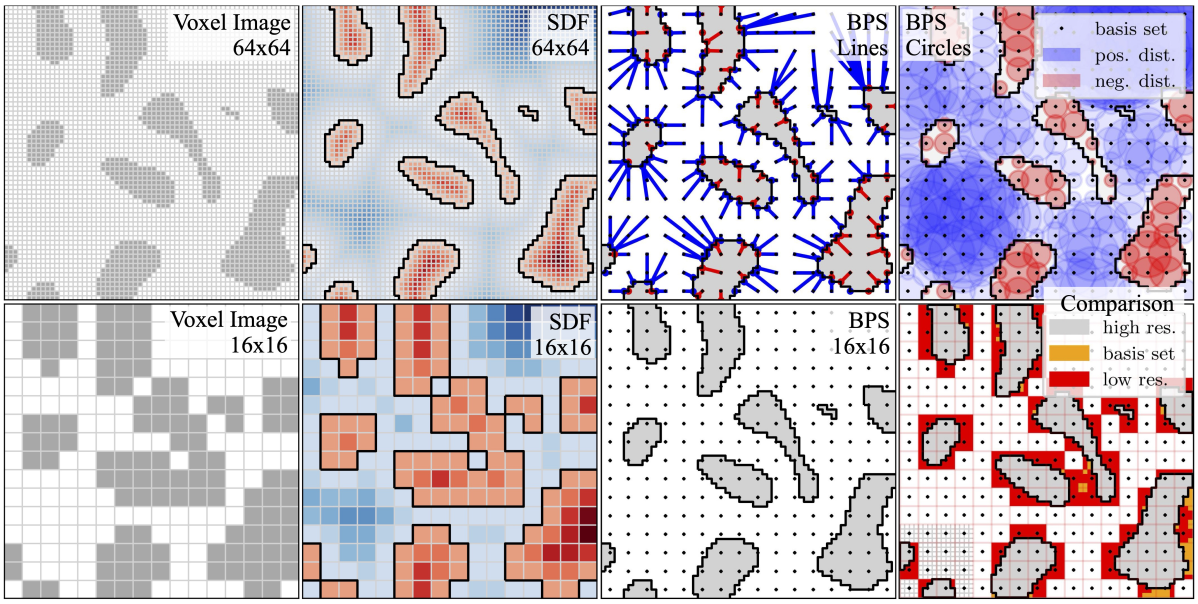


Figure 3.8: Representation capabilities of SDF and BPS for different resolutions.

here. The neural networks eliminate the need for exhaustive multi-start searches and can speed up the computation significantly. The design and training of such networks follow in the next section.

3.4 Learning for Speeding-Up Optimization-Based Motion Planning

The idea is to mitigate the strong dependence on the initial guess and to reduce the susceptibility for local minima by using the prediction of a neural network as a warm-start for the optimization-based solvers. This learning-based initial guess should replace the otherwise random multi-starts and speed up the local planning approach, especially in unknown and challenging situations (see Fig. 3.2).

We discuss two methods for training such neural networks for motion planning and inverse kinematics. The central idea is to use the respective objective functions U^{path} and U^{ik}. One can either use it to generate training data with a non-learning-based OMP algorithm, or one can use an unsupervised regression approach, where the objective function is directly used to update the network weights via backpropagation.

3.4.1 Environment Representation

If a network should produce collision-free predictions for arbitrary unseen environments, it needs to “understand” the scene. Hence, a suitable encoding of the world is essential. Two common environment representations in robotics and computer vision are occupancy grids W_{O} and point clouds W_{P}. Both have their advantages and disadvantages.

While occupancy grids can be processed like images in 2D, their memory inefficiency and the high computational cost for the convolution operations become a burden in 3D. On the other hand, point clouds are a denser representation. Still, they have no fixed size and no inherent ordering, making it hard for a network to learn a permutation invariant mapping.

Prokudin et al. [PLR19] introduced BPS to represent spatial information in computer vision, which is especially suited for deep learning. Choosing a fixed set of basis points once and measuring the distances relative to this set for all new environments does not have the problem of varying permutations and lengths as point clouds have. It is also far more efficient in terms of memory and computation than voxel grids, allowing fast training in high-resolution 3D environments.

The BPS representation can be understood as a subsampled SDF. Formally, the BPS is an arbitrary (see Fig. 3.9) but fixed set of points

$$B = [b_1, \dots, b_{N_\mathrm{b}}],\ \ b_i \in \mathbb{R}^d. \tag{3.25}$$

The feature vector W_B passed to the network consists of the distances to the closest point in the environment for all basis points. If the environment is defined by a point cloud W_P, this can be calculated by

$$W_\mathrm{B} = [\min_{x_i \in W_\mathrm{P}} |b_1 - x_i|, \dots, \min_{x_i \in W_\mathrm{P}} |b_{N_\mathrm{b}} - x_i|]. \tag{3.26}$$

Alternatively, if the environment is given by an occupancy grid or a distance field D like we used for OMP in Section 3.3, one can directly look up the feature vector

$$W_\mathrm{B} = [D(b_i), \dots, D(b_{N_\mathrm{b}})]. \tag{3.27}$$

With the second approach, it is possible to use signed distances. This directly adds a notion of inside and outside to the representation of the world.

Fig. 3.8 visualizes the concept of BPS and shows their representation capabilities in comparison to a simple voxel model. The top left image is the basis; this $64 \times 64 = 4096$ occupancy map should be encoded. On the left side, SDF and voxel images for different resolutions are shown. On the right, a regular BPS with $16 \times 16 = 256$ points and their respective distances to the closest obstacle in a high-resolution occupancy map is shown. Blue areas describe positive distances to obstacles and are guaranteed free of obstructions. Red regions show negative distances and are completely inside barriers. We can draw no direct conclusion from the white areas, but they have to be marked as obstacles to be conservative. The BPS representation for this regular grid is equivalent to subsampling a high resolution 64×64 SDF. The bottom right image shows the reconstruction of the BPS to the full image with the errors marked in orange. A patch of the original grid is added to highlight the difference in resolution. The far more conservative result of reducing the resolution of the occupancy grid by the same factor is highlighted in red. This comparison demonstrates that the basis set preserves more information than conservatively shrinking the occupancy grid by the same factor. A

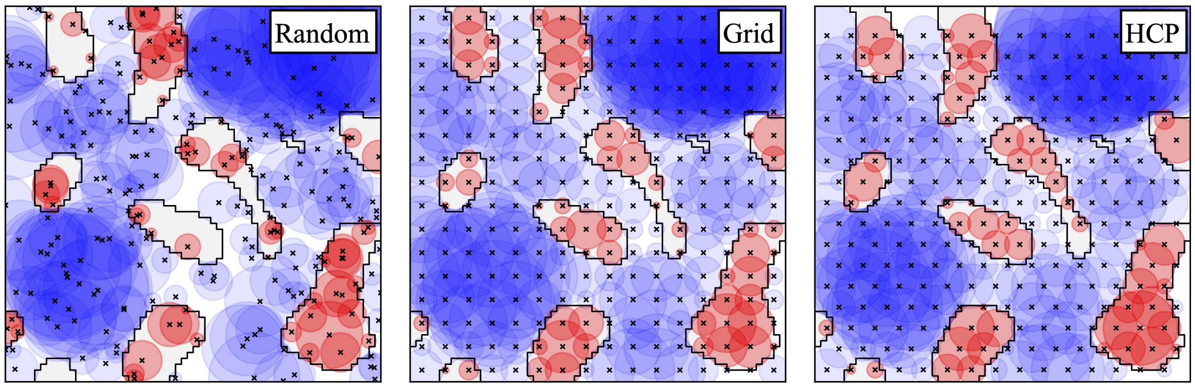


Figure 3.9: The three plots show different modes of generating the BPS. Each set contains 256 points. Randomly choosing the set can lead to large unseen areas in the workspace. Utilizing hexagonal close packing (hcp) ensures that the distance between all points of the set is equal, thereby achieving a uniform coverage of the workspace.

further advantage of the untruncated distances is that, even if there are no nearby data points close to a basis point, they can nevertheless help represent a surface further away. Those attributes make BPS a good choice for encoding the environment for motion planning and IK networks, which are discussed in the following sections.

3.4.2 Robot Representation

The configuration space $\mathcal{Q} \subseteq \mathbb{R}^{N_{\mathrm{DoF}}}$ of a complex robot is vast and generally not sparse. While there are configurations that the robot cannot reach because of joint limits or self-collision, the majority of the space is feasible. The space becomes even larger when not considering just a single configuration but a whole motion $\mathbb{R}^{N_\mathrm{t}} \times \mathcal{Q}$. This path space, however, can be viewed as sparse, as consistent paths are not an arbitrary mess of entirely different configurations. The constraints of timely ordering and a smooth variation of the configurations along the motion reduce the space of actual paths. Nonetheless, configuration space and path space are vast, and to facilitate effective learning, the representation of those spaces for the network's input and output is crucial.

Joint Configuration Encoding

When looking at the IK problem or collision avoidance, the network needs to understand the mapping between configuration space and cartesian workspace. We use a singularity-free representation for the robot's joint configurations using 2D unit vectors instead of the joint values in radians. In the plane, the unit vector is a natural representation of an angle, which inherently corresponds to the directions vector in the workspace. This modification is especially relevant if the joint limits are $[-\pi, +\pi]$ or close to it. Moreover, even in three-dimensional spaces and with stricter joint limits, this encoding choice enables the network to more effectively capture the underlying problem. By utilizing

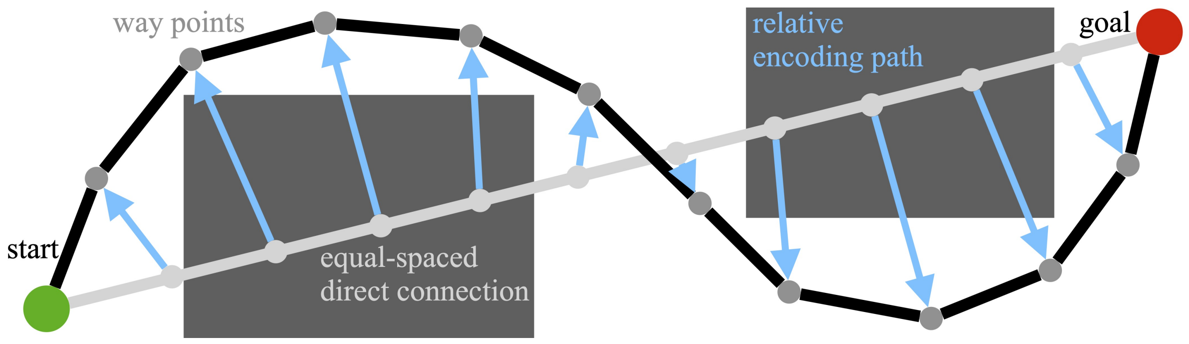


Figure 3.10: The scheme shows two different variants to represent a path. Either use the absolute configurations along the waypoints or the relative difference to the straight connection between the start and goal.

this singularity-free representation, the network eliminates the necessity to internally manage a switch for joint values nearing singularity points. We enforce this encoding scheme by directly mapping the network inputs and outputs onto the unit circle.

Relative Path Encoding

The naive encoding of a path is just the array of waypoints, each a snapshot of the current joint configuration. Expecting a neural network to learn and understand this representation is challenging. While there are definitely similarities and common structures in different motions, they do not become obvious in this geometric representation. As the path space is vast, but the subset of actual paths is much smaller, it is beneficial to choose an encoding that more closely represents the structure of actual paths.

For that matter, a central concern is using invariances and common patterns of motion planning. The idea is to use the first best guess, the direct connection in configuration space between the start and goal, as the basis for the encoding. Instead of using the absolute configuration for every waypoint, the distance to the respective waypoint on the straight connection is used (see Fig. 3.10, blue). This representation is not absolute and always needs a start and end configuration to which it is applied. The advantage is that this path representation is invariant to many different setups. A zero vector as encoding is always identical to the direct connection in absolute terms. Furthermore, notions like the path is curved to the left persist across different pairs of start and goal.

Another benefit of this encoding scheme is that the neural network learns from a reasonable starting point. At the beginning of the training, the network's predictions are just noise. The relative path encoding adds this noise to the direct connection between the start and end points of the given motion task.

Symmetries of Motion Planning Data generation is costly, so one can use symmetries in motion planning to efficiently use the information in each sample. If one has the optimal path from A to B, one also has the solution from B to A. This assumption is no longer valid if terms in objective function break the temporal symmetry. Furthermore,

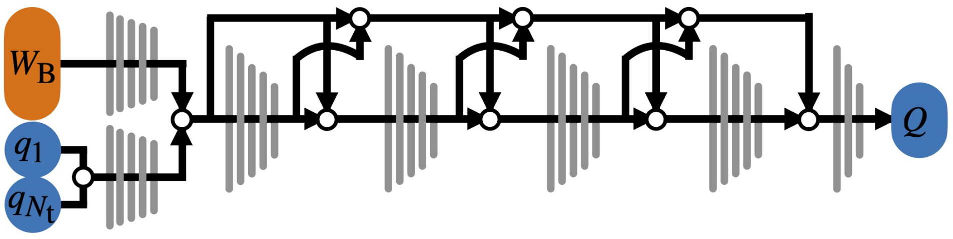


Figure 3.11: Network architecture to map from a given motion task (world, start, and end configuration) to an optimal path. Blocks of tapered Fully Connected Layers (gray) are combined like the DenseNet architecture [Jég+17] via skip-connections and concatenations. See the bottom of Table 3.2 for the number of network parameters used for the different robots.

many robots also have spatial symmetry axes. Often, it is possible to align the first joint of the robot with one axis of the cartesian coordinate system of the environment. Doing so allows us to rotate the world and this first joint simultaneously without changing the optimality of the resulting trajectory in the new world. Chamzas et al. [CSK19] use this spatial symmetry to store paths in their database more efficiently. While ideally, one wants to integrate these symmetries directly in the data representations or the network architecture, we also use them to augment and increase the dataset.

3.4.3 Supervised Learning

First, we look into supervised training for motion planning. The network should encode the experience of successful paths by learning a mapping from a motion task, consisting of a world and a start q_1 and end configuration q_{N_t}, to the intermediate waypoints of an optimal path $Q^* = [q_2, ..., q_{N_t-1}]$. Fig. 3.11 shows the network architecture we use.

Besides encoding the in- and output of the network (see Sections 3.4.1 and 3.4.2), a crucial point for supervised learning is the dataset. The following section discusses several insights into the generation and usage of such a dataset with and for OMP. Our methods substantially increase the final prediction quality of our network and make training more efficient. See Section 3.5.2 for experimental validation of these methods.

Dataset Adaption for Efficient Learning

The idea is to no longer rely on random multi-starts and speed up the planning time by using a neural network to predict an initial guess for OMP. The network should encode the experience of successful paths by learning a mapping from a motion task, consisting of a world and a start q_1 and end configuration q_{N_t}, to the intermediate waypoints of an optimal path $Q^* = [q_2, ..., q_{N_t-1}]$. Fig. 3.11 shows the network architecture we use. Besides encoding the in- and output of the network, a crucial point for supervised learning is the dataset. The following section discusses several insights into generating and using such a dataset with and for OMP. Our methods substantially increase the final

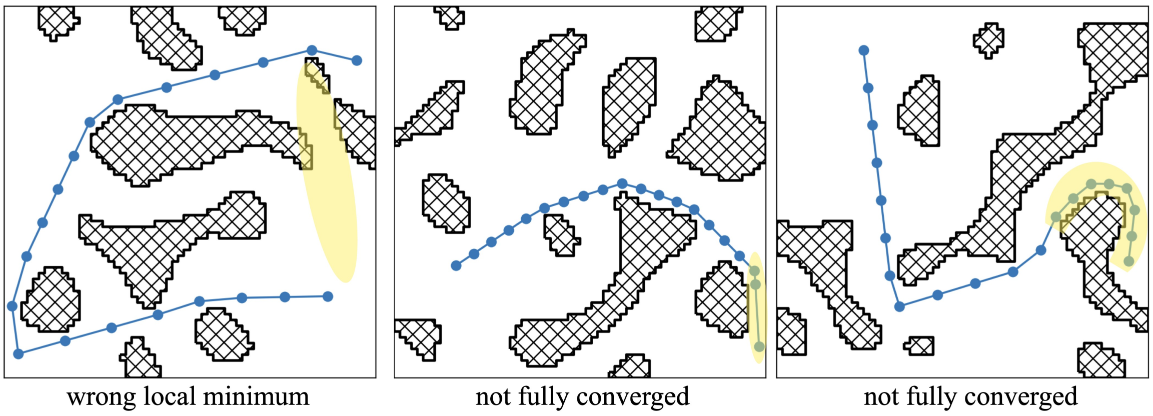


Figure 3.12: The graphic shows three examples where even 100 random multistarts did not converge to the optimal solution. The suboptimal regions are highlighted in yellow.

prediction quality of our network and make training more efficient. See Section 3.5.2 for experimental validation of these methods.

Challenging Samples The training data distribution should represent the actual application and focus on challenging examples. Suppose the dataset is too easy, and direct linear connections from A to B drastically outweigh more complex trajectories. In that case, training the network can quickly get stuck in a local minimum and predict only straight lines, regardless of the given task. One possibility of generating a dataset with more challenging motion problems is to consider only samples where OMP using the direct connection as an initial guess does not converge to a feasible path.

Consistent Samples Besides finding challenging samples, the ambiguity of motion planning (more than one feasible solution) can become a problem for trajectory regression. Even if we assume that we can resolve the ambiguity of the optimal path via the objective for the shortness of the path, there will be "close calls" in the dataset, that is, two paths with almost the same objective values but fundamentally different geometries. Furthermore, the classical planner using a limited number of multi-starts can only produce suboptimal labels (see Fig. 3.12), making it hard for the network to learn consistent mapping. This suboptimality is especially problematic for challenging tasks, where the classical planner often fails and only produces feasible paths in a small fraction of the attempts.

Interplay between Network and Dataset

We propose to use the neural network Θ to correct and enhance its own training data $S = \{(x, y)\}$. This approach is possible whenever synthetic data is used for training, and one has an objective metric to measure the quality of a prediction. The assumption

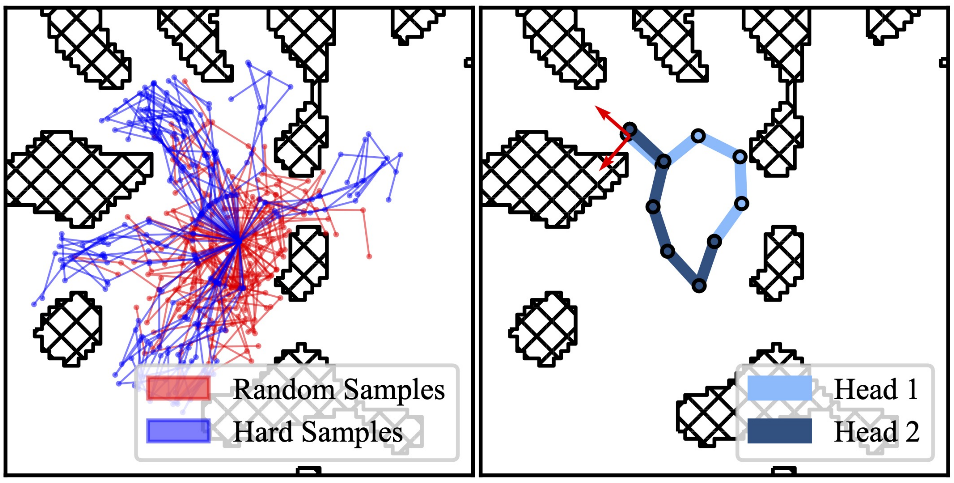


Figure 3.13: *left*: Comparison between 50 random samples and 50 samples in the hard set after the training finished. *right*: The predictions of the twin heads both satisfy the end-effector for a hard position and still show two distinct modes.

is that the network can learn some aspects of the problem even on an imperfect dataset, and its predictions will become better than those of random guessing.

When generating a dataset with and for OMP, we can use the objective $U^{\text{path}}(Q)$ as a universal quality metric. The idea of interweaving the network training closer with the dataset generation and improvement is summarized in Fig. 3.14 and described by Algorithm 1. In what follows, we give a detailed explanation of the different methods.

Clean First, the network can be used as guidance to double-check where the dataset is inconsistent (see Fig. 3.12). If the label and the network's prediction Q_{p} are close, and the respective objective $U^{\text{path}}(Q_{\text{p}})$ is small, no action is necessary. However, if there is a discrepancy between prediction and label, it is worthwhile to use more multi-starts with the OMP to get a better label for the sample. If a prediction has a better objective than the current label, we can replace it without adding any bias to the dataset. Doing so will improve the labels and make the dataset more consistent. This makes it easier for the network to find the underlying patterns.

Boost After some training, the network has learned to predict good paths for the relatively simple samples, but the more challenging outliers are still not solved. Therefore, we use boosting to select challenging samples with higher probability during training, increasing the incentive to learn these samples. We steer this kind of curriculum again by using U^{path} as a metric: the higher the difference between the predicted and actual cost for a given sample, the more challenging it is. In our experiments, the challenging tasks for the network correlated well with the relative path length and the number of obstacles in the scene.

The idea of boosting can also be applied to unsupervised training. In that case, the loss function of the neural network and the total objective $U^{\mathrm{ik}}(q)$ are identical. This measure can directly be used to over-represent complex samples. We define a sample q as hard if its cost $U^{\mathrm{ik}}(q)$ is four times higher than the rolling mean.

The positive effect of boosting can be seen in Fig. 3.13 (left) for a 5 DoF arm in 2D. In the image, 50 random but feasible samples for the robot in the given environment are drawn in red, and in blue, 50 samples that were in the hard set after the training finished. The challenging samples are more extended and fill the narrow passages in the world more completely than the random samples, which tend to cluster towards the center of the workspace.

Extend Lastly, one can use the network to generate new samples. The idea is to use the network's performance on a new sample as a metric for information gain. To improve the network, one wants specifically to add samples where the network performs poorly. To decide this before spending resources to produce a new label using OMP, one can use the objective of the prediction $U^{\mathrm{path}}(Q_{\mathrm{p}})$. If it is small, the network can solve this task already. However, if the objective is significant, the task is challenging for the network, and we include it in the training set.

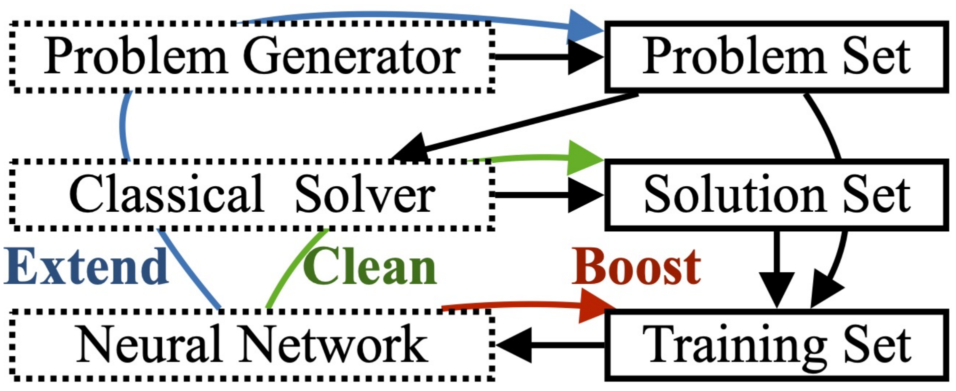


Figure 3.14: Scheme showing the connection between non-learning-based solver, dataset, and neural. The colored arrows indicate the information flow for cleaning, extending, and boosting the dataset with the guidance of the network.
Clean: use the solver to update labels in the solutions set;
Boost: overrepresent hard examples to make training more challenging;
Extend: generate new hard samples and add them to the problem set.
The detailed algorithm is described in Algorithm 1.

Algorithm 1 Improvement of Network and Dateset.
For a visual description, refer to Fig. 3.14.

```
procedure MAIN
    create initial dataset $S = \{(x_i, y_i)\}_{i=1}^{N_S}$ with OMP
    train net $\Theta$ on dataset $S$
    while improvement on testset do
        CleanDataset($S, \Theta$)
        ExtendDataset($S, \Theta, N = |S|/20$)
        BoostDataset($S, \Theta, p_{\text{perc}} = 0.8, p_{\text{ratio}} = 0.9$)
        train net $\Theta$ on dataset $S$
procedure CLEANDATASET($S, \Theta$)                                    ▷ Clean
    for $(x_i, y_i)$ in $S$ do
        $y_p \leftarrow \Theta(x)$
        $y_p^* \leftarrow$ OMP($x, y_p$)
        if $U^{\text{path}}(y_p^*) \leq U^{\text{path}}(y)$ then
            $S$.replace($y \leftarrow y_p^*$)
procedure EXTENDDATASET($S, \Theta, N$)                                ▷ Extend
    for $k \leftarrow 1$ to $N$ do
        $x_i \leftarrow$ sampleNewProblem()
        if $V_p < U^{\text{path}}(\Theta(x_i))$ then
            $y_i \leftarrow$ OMP($x_i$)
            $S$.append($(x_i, y_i)$)
procedure BOOSTDATASET($S, \Theta, p_{\text{perc}}, p_{\text{ratio}}$)                  ▷ Boost
    $V \leftarrow [U^{\text{path}}(\Theta(x_i))$ for $(x_i, y_i)$ in $S]$
    $V_p \leftarrow$ percentile($W, p_{\text{perc}}$)
    for $(x_i, y_i)$ in $S$ do
        if ($U^{\text{path}}(\Theta(x)) < V_p$ and random(0, 1) $< p_{\text{ratio}}$) then
            $S$.remove($(x_i, y_i)$)
```

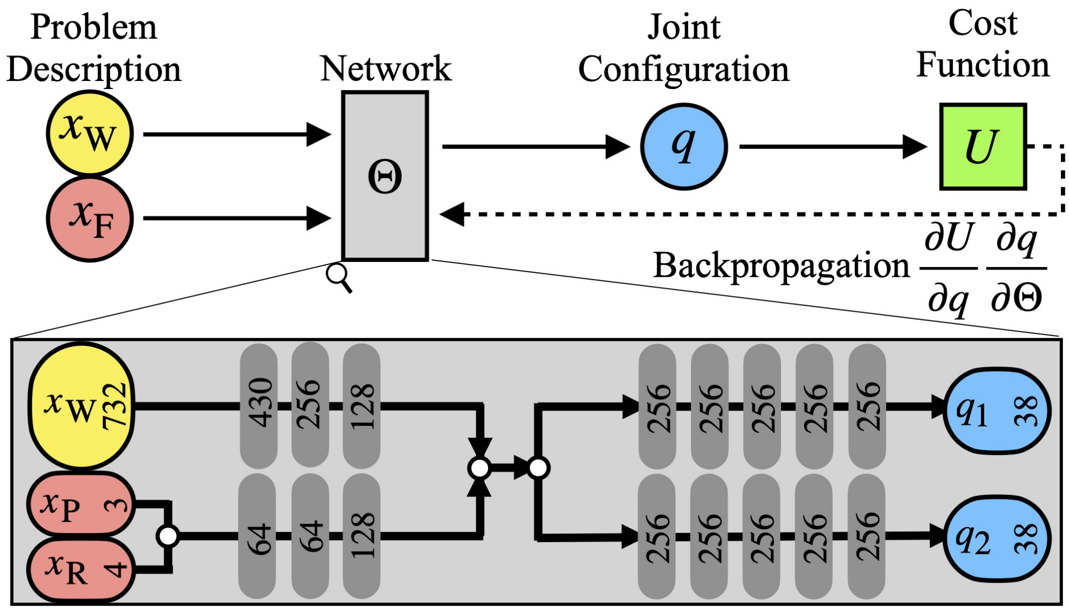


Figure 3.15: The graphic shows the flow of information through the neural network for IK. In the lower half, the detailed network structure for inverse kinematics of Agile Justin (19 DoF) is shown, with two heads and the unit vector representation for the joint angles.

3.4.4 Unsupervised Learning

Alternatively, as the objective function (3.19) holds all the necessary information to quantify a given configuration, it can be directly used as a loss function for training a network. Pandy et al. [PLC21] introduced unsupervised regression networks for robotic motion planning. We adapt the idea and discuss the extensions needed in the context of IK.

For a given problem defined by a world x_W and a frame x_F, one can directly calculate the gradients of (3.19) with respect to the network weights Θ by using the chain rule:

$$\frac{\partial U^{\text{ik}}}{\partial \Theta} = \frac{\partial U^{\text{ik}}}{\partial q}\frac{\partial q}{\partial \Theta}. \tag{3.28}$$

Fig. 3.15 shows the information flow through the network. The IK problem is described by a world x_w, and a frame in the workspace x_f and the network should predict a collision-free joint configuration that satisfies the end-effector. The dotted line indicates the backpropagation during unsupervised training, where the network weights Θ are directly updated according to the gradient of the cost function U^{ik}.

The huge advantage of this unsupervised approach is that no computationally expensive generation of expert data is needed as in supervised learning. Here, different worlds x_W and target frames x_F are sampled randomly, and via backpropagation the resulting gradients can be directly computed. This makes it faster and more straightforward to train.

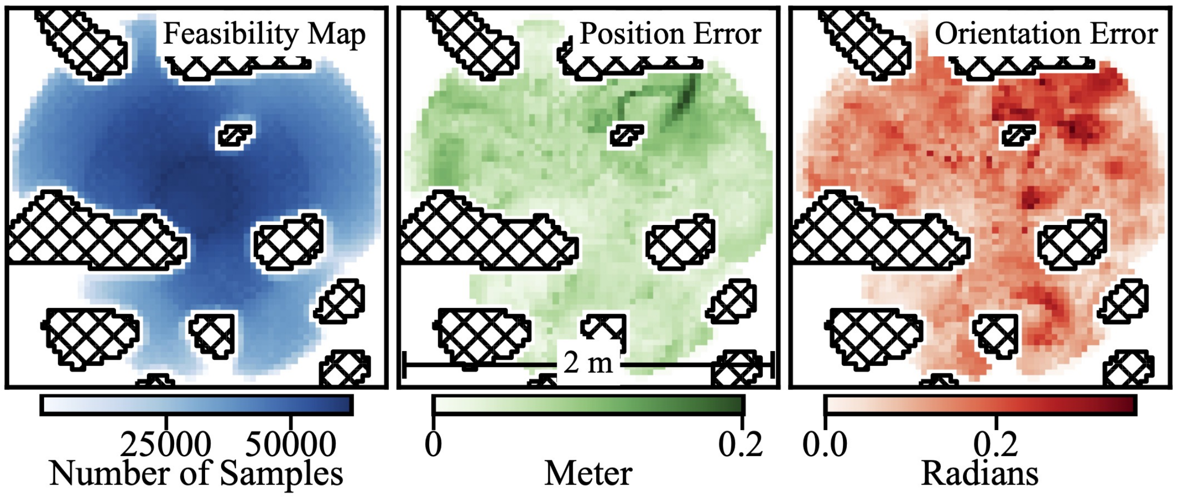


Figure 3.16: Feasibility map *(right/blue)* for the 2D arm with 5 DoF for a specific environment. The maximal position error *(center/green)* and the maximal orientation error *(left/red)* highlight which regions are challenging for the network in more detail. The error maps show the maximal error over all orientations for each 2D position in the image.

To analyze the IK problem in the whole workspace, we generated feasibility and error maps of the robots in the different scenes. Fig. 3.16 shows three maps: feasibility (blue), maximal position error (green), and maximal orientation error (red) for a given position in the workspace. The maps were generated by sampling the whole joint space and collecting which Euclidian targets were reached. Then, the position and the orientation error for each feasible target were computed. In general, the overall number of feasible poses decreases towards the borders of the workspace. Depending on the robot's kinematics, not only the parts of the workspace with obstacles are unreachable, but also areas behind obstacles. This distribution highlights again the benefit of boosting (see Section 3.4.3) to counteract the underrepresentation of challenging samples close to obstacles and the edge of the workspace. Those error maps can assess the network's performance over the whole workspace and are far more detailed than random test sets, which are commonly used. The following sub-sections discuss the insights of this detailed analysis, which gave rise to our network and learning architecture.

Mode Switches and Twin-Headed Networks

While the length cost U_1^{q} ensures, in general, that there is one optimal solution, there are still different modes over the workspace, and the network must switch between those modes to successfully predict *optimal* IK solutions for all possible targets. Fig. 3.17 visualizes the general concept of mode switches between a pair of modes to ensure an optimal solution. Assuming there exist different modes of varying costs U^{ik} across the workspace and the network should make the optimal prediction at each position p, it needs to switch between those modes. The transition regions are complex to represent

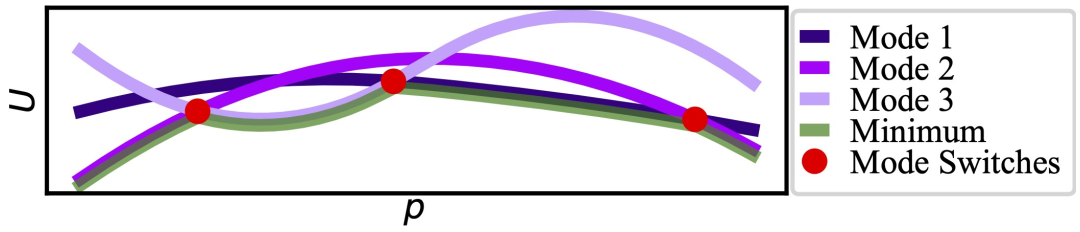


Figure 3.17: The 1D scheme shows the necessity of mode switches over the workspace p to get a globally optimal solution with respect to the cost function U.

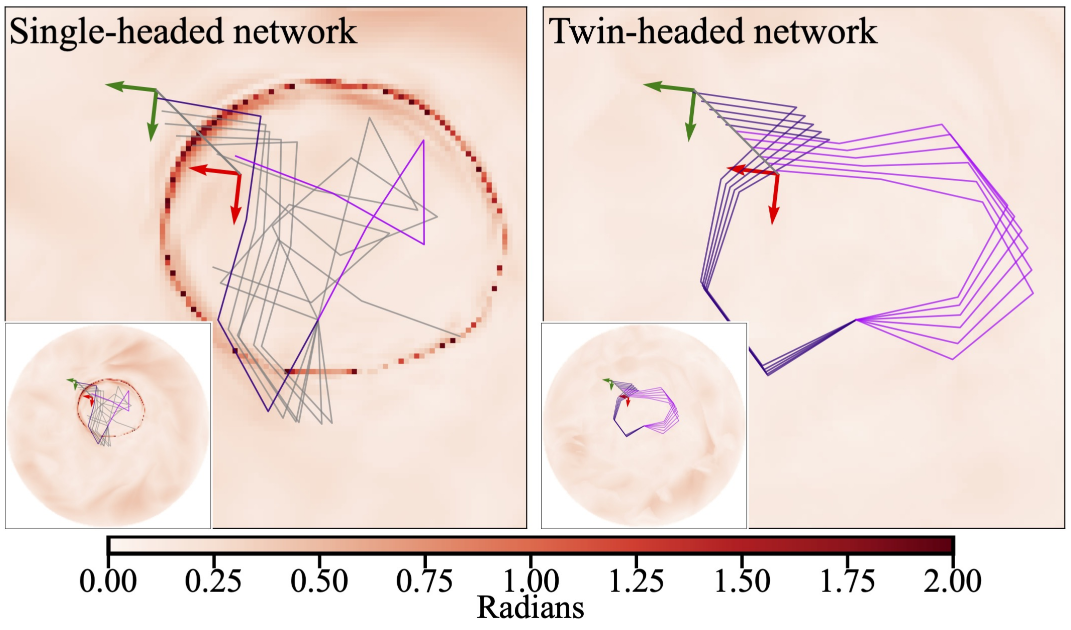


Figure 3.18: comparison between a single-headed network on the left and a twin-headed network on the right for the IK prediction of a 5 DoF robot. The desired end-effector frame follows a line on fixed orientation.

for a neural network and can lead to significant errors.

This behavior plus our solution is visualized in Fig. 3.18. For the 2D arm with 5 DoF, the transition regions can be seen in the heat map. This underlying red heat map indicates the worst orientation error across all 2π possible (discretized with 2880) goal orientations at each position. The distinct circular pattern (left) shows the transition region between two modes, where the prediction of the single-headed network breaks down. The specific position of these transition regions depends even on the initial weights of the network, but each initialization has the same behavior. There are always regions where the network needs to represent the switching between two modes. One can see the prediction breakdown by gradually moving the target frame from outside the ring (green) along a straight path to a position inside the ring (red). In the transition region, the network switches modes and cannot produce valuable predictions.

By adding a second head to the network, which outputs a second prediction, one

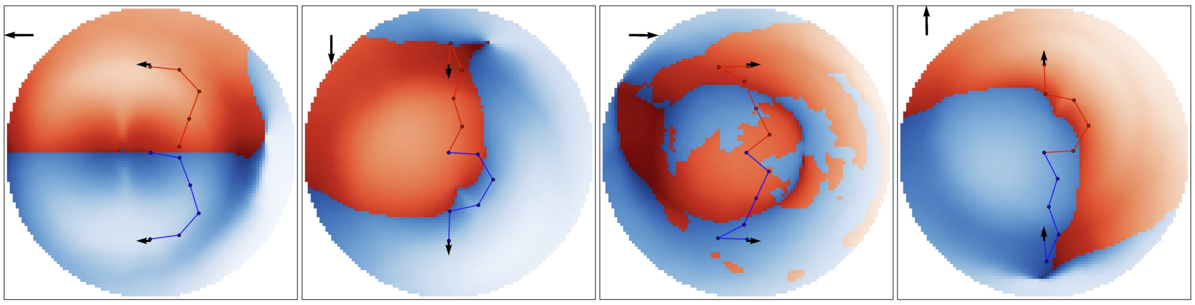


Figure 3.19: The color map shows which head scores better for the objective function over the whole workspace: head one is red, and head two is blue. Visit the website for the full animation.

can overcome this problem. Each head of the twin model also has its own transition regions, but as those two areas do not intersect, one always has a valid and smooth prediction for the configuration. It is essential to add that two heads are enough, even for more complex settings with multiple modes. The two heads do not represent the modes directly but only mask the transition region between pairs of modes.

We introduce an additional loss $U_\mathrm{H} = \|q_\mathrm{a} - q_\mathrm{b}\|$ between the two heads of the network to counteract mode collapse and gain a valuable second guess. Both heads are trained simultaneously via backpropagation. Fig. 3.13 (right) highlights that maximizing the difference in configuration space between those heads produces fundamentally different solution modes. Besides allowing sharp switches between modes, this approach leads to the simplest version of a generative model, with much more accessible training and no need for network ensembles [Lem+21] to prevent mode collapse.

Fig. 3.19 visualizes the predictions of the two heads across the workspace. The goal orientation of the end-effector is in the top left of each image of the series. As for the error maps (Fig. 3.16), the heatmap shows the error over the whole workspace. Darker colors represent more significant errors, and the color indicates which head scores better in the objective function for the specific position. Head one is red, head two is blue, and they perform differently over the workspace. Rotating the goal orientation changes this pattern, especially the symmetries in the kinematics become apparent when following a complete revolution. In addition, the predicted robot configurations are shown at two positions to visualize the different modes.

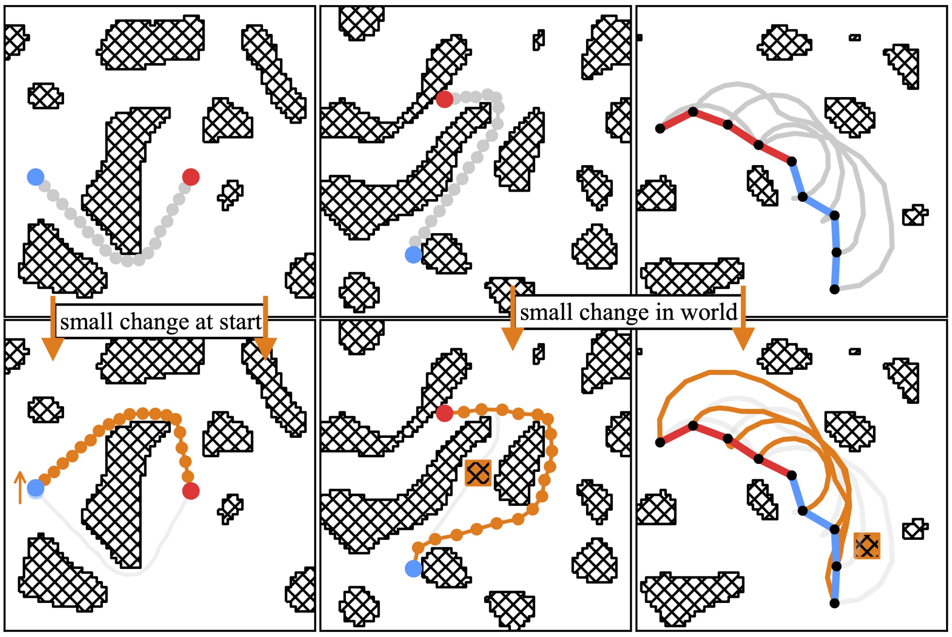


Figure 3.20: Three examples of the networks' capability to make global changes in the prediction for slightly altered problems.

3.5 Experimental Evaluation

This section shows the results of the supervised and unsupervised learning methods for multiple robots with different levels of complexity. To evaluate the networks, we use their prediction as a warm-start for the optimization-based solvers described in Section 3.3 and compare convergence and feasibility rates for unseen test sets. We analyze the proposed methods' influence on the quality of the network predictions and compare them to random baselines. Furthermore, we show some real-world experiments on the humanoid Agile Justin.

Qualitative Analysis

Motion Planning In motion planning, small changes in the problem often lead to fundamentally different solutions. Fig. 3.20 shows a qualitative analysis of the network predictions for motion planning. The changes in the start configuration (*left*) and the environment (*middle, right*) are highlighted in orange. The networks' predictions after optimization for those new problems are shown in the bottom row, indicating the networks can react sharply to small input changes. Our worlds and training were challenging enough that the networks react sharply to small changes in the input, predicting completely different solutions to only slightly altered problems.

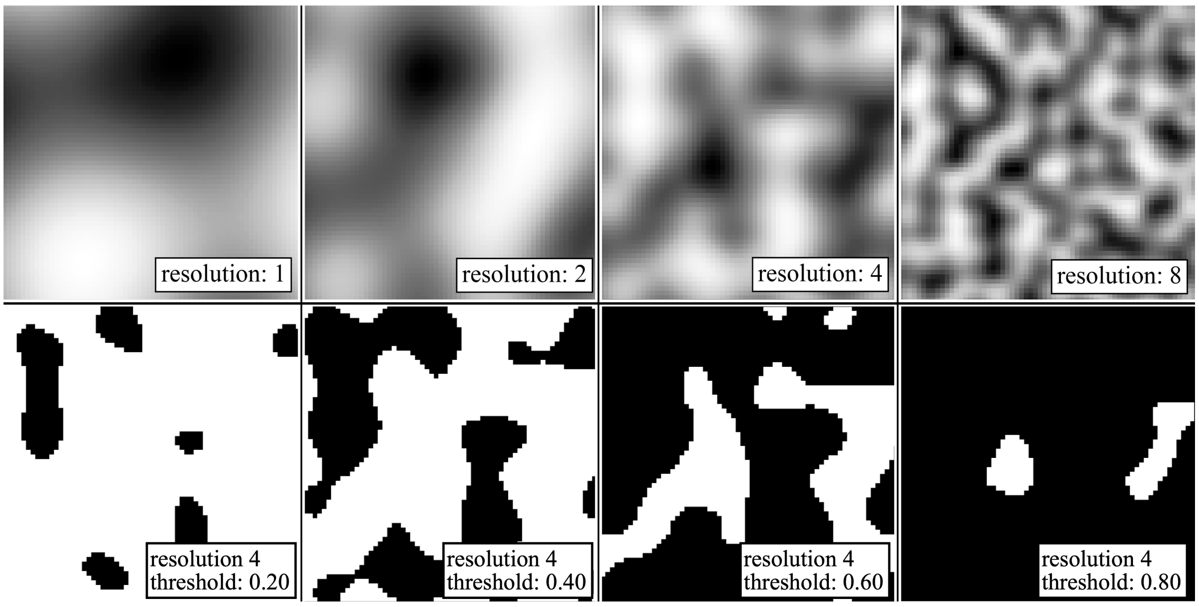


Figure 3.21: Generation of random worlds using Simplex noise [Per01; Gus05]. By changing the resolution (*top*) and the cut-off threshold (*bottom*), the height maps can create a wide range of diverse worlds for training and testing.

3.5.1 Challenging Datasets and Test Environments

We used different types of robots and setups to evaluate our methods. In 2D, we investigated a simple sphere robot with 2 DoF and a serial arm with 4 or 5 DoF to analyze general behavior and obtain detailed visualizations. Furthermore, we used two real 3D robots: the LWR III [Hir+02], a robotic arm with 7 DoF, and humanoid robot DLR Agile Justin [Bäu+14] with 19 DoF distributed over an upper body and two arms.

Challenging Environments To generate diverse and challenging worlds, we used simplex noise (see Fig. 3.21. This gradient noise is used in video games to create random but naturally-looking levels. A typical example is a continuous height map. By changing the cut-off threshold and the resolution of this noise, we can vary the density and form of the obstacles in the binary occupancy grid. To ensure that the environments are not too densely packed with obstacles, at least 200/1000 random robot configurations q must be feasible to include a world in the dataset. Examples of the worlds can be seen in Figs. 3.1a, 3.5 and 3.22.

Table 3.1 compares how suited different environments are for training and testing motion planning and inverse kinematics. We argue that our simplex worlds outperform the austere block or sphere worlds. To validate this, we used an auto-encoder setting. We generated 10000 training worlds for each distribution: block, sphere, and simplex. Additionally, we generated 1000 unseen test worlds for all settings. The worlds we used were 64×64 pixels large, and we used 512 basis points to encode the worlds as input for the neural networks. For the block worlds, the number of blocks was sampled uniformly

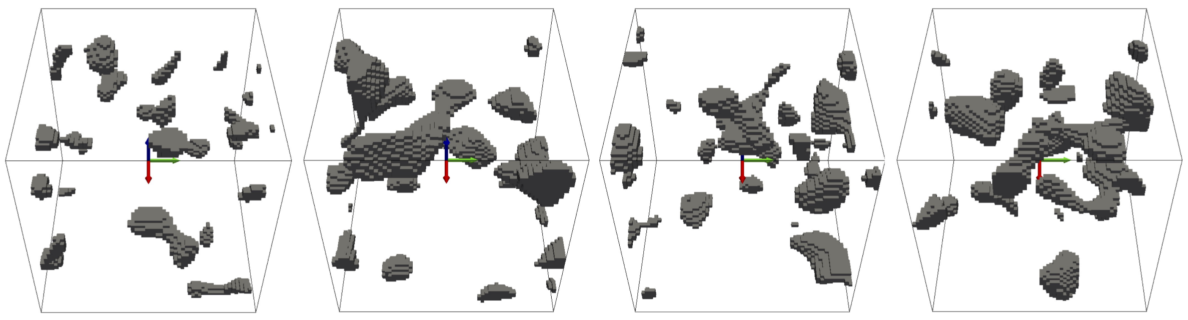


Figure 3.22: Examples of random 3D worlds for our learning framework. Each map has a size of 64^3 pixels and is the ground truth for collision avoidance and safe planning. The worlds were generated using Simplex noise [Per01; Gus05].

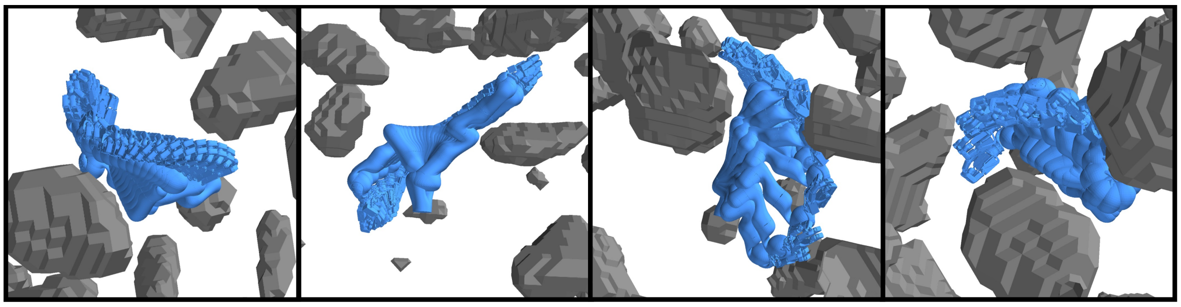


Figure 3.23: Examples of training data of the robotic arm LWR III for supervised learning. Those collision-free paths were generated with an OMP solver using extensive random multi-starts (see Section 3.3) with a focus on the global optimality of the samples for a consistent training signal.

between 1 and 50, and the side lengths of each block were sampled between 1 pixel and 30 % of the whole workspace. For the sphere worlds, we used the same settings for the number of spheres and their radii. We measure the performance of the networks by looking at the reconstruction error. This measurement is done in image space by mapping the BPS encoding to a full occupancy map, as shown in Fig. 3.8. We report the average number of falsely classified pixels.

When contrasting the performance of an autoencoder trained on block worlds or sphere worlds against an autoencoder trained on our simplex worlds, the latter is more generalizable. While the autoencoder trained on block or sphere worlds performs well for a test set from the same distribution, generalization to other settings is poor. However, when training the autoencoder with simplex worlds, the ability to generalize to out-of-distribution worlds is much better.

Motion Planning Dataset To generate the correct labels in the training data for the motion planning networks we used the OMP approach described in Section 3.3.3, with naive gradient descent and fixed step size. The paths consist of $N_{\mathrm{t}} = 20$ waypoints, and to make the dataset challenging, we only included hard tasks. We define a task as hard

Table 3.1: Comparison between simplex worlds and block worlds for autoencoding. Reported is the percentage of incorrectly predicted pixels.

10000 training worlds		Tested on		
1000 unseen test worlds		block worlds	sphere worlds	simplex worlds
Trained on	block worlds	3.7 %	11.2 %	15.6 %
	sphere worlds	6.4 %	3.9 %	12.8 %
	simplex worlds	5.7 %	4.3 %	5.5 %

if a straight line as an initial guess does not converge to a feasible solution. All other paths were discarded as too easy. We used up to 100 multi-starts and always picked the shortest feasible solutions as the correct label. The heuristic for generating the initial guesses was to use one to three random points in the configuration space and connect them linearly with the start and endpoint. See Table 3.2 for an overview of the robots and the datasets.

Overall, we generated 10000 simplex environments for each robot and at least 10 (and up to 1000) paths in each environment. 9000 of those worlds were used for the network's training, and the remaining unseen worlds were for testing. All the motion planning results in Section 3.5 are based on this unseen test set. As a quality measure, we report the feasibility rate ϕ, i.e., the quotient of the number of feasible paths and the size of the test set. Generating those extensive datasets is time-consuming but can be parallelized entirely. To evaluate the methods and the networks online, we measured all timings on computers with Intel i9-9820X @ 3.30 GHz with 32 GB RAM. While all 16 cores are used for training, the online prediction runs only on a single core.

3.5.2 Learning-Based Motion Planning

Network

The last lines of Table 3.2 show the network details in numbers, and Fig. 3.11 displays the general architecture. All the networks were trained purely supervised with a mean squared error between the predicted path Q_p and the label Q as loss function. As the encoding for the path, the deviation from the straight line is used. This representation implies that even an untrained network producing only random noise around zero can make meaningful predictions. For start and end, we use the normalized joint vectors q_1 and q_{N_t} as input. As environment encoding, we use the BPS described in Section 3.4.1 with a hexagonal closed packing and only consider points inside the robots' maximal reach. See Fig. 3.25 for an analysis of the dependency of the prediction quality on the size of the BPS.

Table 3.2: Overview of the motion planning datasets and networks for the different robots.

	Sphere Bot	Flat Arm	LWR III	Agile Justin
DoF	2	4	7	19
World size	$10 \times 10\,\mathrm{m}^2$	$1.0 \times 1.0\,\mathrm{m}^2$	$1.2 \times 1.2 \times 1.2\,\mathrm{m}^3$	$3 \times 3 \times 3\,\mathrm{m}^3$
Grid dimensions	64×64	64×64	$64 \times 64 \times 64$	$64 \times 64 \times 64$
# Worlds	10^4	10^4	10^4	10^4
# Paths	0.6×10^6	6.5×10^6	2.2×10^6	3.7×10^6
Avg. time p. core	0.1 s	0.8 s	3.1 s	8.4 s
# Improvements	0.1×10^6	0.3×10^6	0.2×10^6	0.3×10^5
# Extensions	0.5×10^5	1.5×10^5	3.0×10^5	5.0×10^5
Avg. $U^{\text{path}}(Q)$	1.589	1.7136	1.551	1.483
Avg. Feas. ϕ	67.3 %	32.6 %	54.5 %	44.1 %
Avg. time p. core	0.1 s	0.8 s	3.1 s	8.4 s
Net				
# In → # Out	$516 \rightarrow 39$	$520 \rightarrow 72$	$2062 \rightarrow 126$	$2086 \rightarrow 342$
$\lvert W_{\mathrm{B}} \rvert$	512	512	2048	2048
N_{t}	18	18	18	18
# Parameters	3.4×10^6	7.5×10^6	2.4×10^7	4.1×10^7

Analysis of Dataset Adaptions and Training

Table 3.3 shows the influence of the methods for dataset adaption during training as discussed in Section 3.5. As a metric, we use the feasibility rate ϕ of the predicted paths after further iterations with the OMP as described in Section 3.3.3. First, we compare the hard and the easy datasets. Because the easy dataset consists only of paths produced from straight lines, the overall variance is too slight, and the network does not learn to avoid the obstacles. This network is not able to solve the test set of hard examples. Next, we introduced the different modes of data augmentation to increase the size of the dataset. The temporal and spatial symmetries improve the feasibility rate ϕ without additional computing costs. The number of cleanings describes how often the labels were updated with the help of the neural network. Each iteration brings the labels closer to the optimal solution and makes the dataset more consistent, leading to better results. At this stage, the network performs well with a success rate of over 85 %, but there are still tasks the net cannot solve. We add the boosting technique to overrepresent more challenging samples during training to increase the feasibility further. As the final step, we use the trained network to generate more challenging samples. With this approach, we achieved 100 % feasibility on the hard unseen test set.

Comparison to Random Multi-Start

The capabilities of our method become apparent when we compare the network's prediction to the heuristic with random multi-starts used to create the dataset. In Fig. 3.24,

Table 3.3: Influence of different dataset distributions, dataset extensions, and training methods on the feasibility of predicted paths for Agile Justin.

Dataset		Training		Feasibility ϕ	
# Cleans	Distribution	Aug.	Boost	Network	+OMP
0	easy	no	no	0.042	0.347
0	hard	no	no	0.126	0.653
0	hard	axis	no	0.133	0.691
0	hard	time	no	0.138	0.736
0	hard	both	no	0.143	0.772
0	hard	both	yes	0.171	0.859
1	hard	both	yes	0.196	0.893
2	hard	both	yes	0.217	0.925
3	hard	both	yes	0.223	0.941
3	hard + ext.	both	yes	0.283	1.00

the convergences to a feasible path of different initial guesses are displayed for the LWR III and Agile Justin. Without any experience, the best one can do is try random multi-starts and hope one converges. From the 100 multi-starts we used per task, only 50 % converge to a feasible solution after 50 iterations. Even the lucky initial guess, which converged the fastest for each problem, gets outperformed by the network's prediction. The crucial difference is that our network does not depend on chance but can reliably predict initial guesses that converge after a few iterations to a feasible solution.

The actual speed gain is even more prominent when looking at the distribution over different motion tasks (see bottom of Fig. 3.24). There are problems for the LWR III and Agile Justin where only 10 % or less of all multi-starts converge to a feasible path. If one wants to find a solution to such a problem with 90 % confidence, one would need more than $\log(1-0.1)/\log(1-0.9) > 20$ multi-starts, making the initial guess of the network effectively over 20 times faster.

On our test machine, a single iteration of gradient descent for one path of Agile Justin takes 10 ms on a single core. Using the network's prediction as a warm-start and stopping each sample after convergence leads to an average run time of 182(±29) ms with a worst-case of 334 ms.

3.5.3 Learning-Based Inverse Kinematics

Looking at the IK, we first demonstrate the use of the neural networks inside the collision-free IK solver for a 2D robotic arm. In Fig. 3.7 (**left**), the two-stage approach using many random multi-starts is shown (Section 3.3.3). One can clearly see the advantage when comparing the prediction of a neural network (**right**) against random multi-starts. The same two-stage procedure is used for the two network predictions on the right-hand

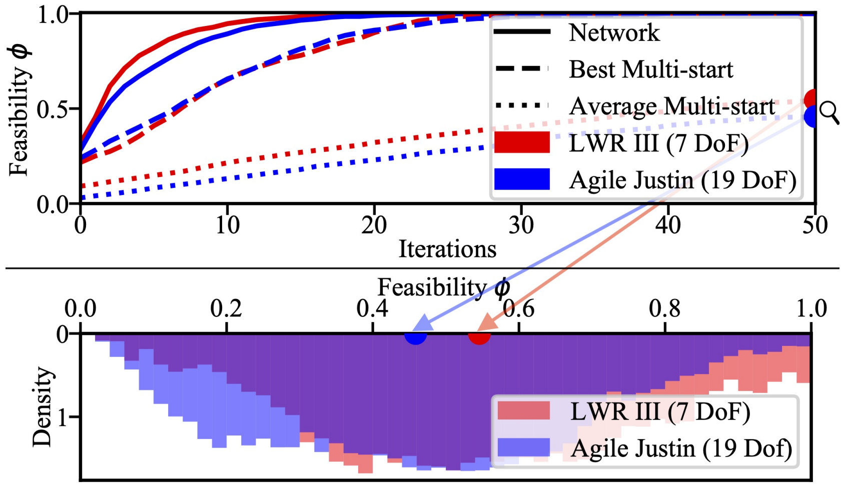


Figure 3.24: Top: Average convergence to feasibility ϕ of OMP for different initial guesses. The network prediction significantly outperforms the average and even the best of 100 multi-starts. Bottom: Distribution of the average feasibility ϕ of the random multi-starts after 50 OMP iterations.

side. One network is trained without the world as a dedicated input, and one uses the world's BPS encoding to predict collision-free IK solutions. First, one can see how close the two predictions are to the desired end-effector. Furthermore, the prediction of the world-aware network is already in the correct narrow passage between the obstacles. Using this prediction as an initial guess eliminates the need for multi-starts in most cases and leads to quicker convergence, as the optimizer only needs a few iterations for a feasible solution.

Table 3.4 shows an ablation study for the learning and network architecture proposed in Section 3.4.4. For the humanoid robot Agile Justin, the test set consisted of 100000 samples from unseen worlds. Because the network prediction is not directly used on the robot but the converged result, we report the feasibility rate after 10 iterations of the solver. The table shows that each architectural component improves the performance of the network. In the extreme case where none of those methods are used, the feasibility rate is only 25 %, while the final performance is close to 100 %. Notably, the boosting does not improve the mean performance but significantly reduces the maximal error of the network's predictions. As this approach over-represents the complex samples with a significant objective U^{ik}, it is designed to improve those worst cases. This design is crucial if one uses those network predictions as a warm-start for an optimization-based solver in challenging scenes: Long searches with many multi-starts slow down the numerical solver for those cases.

The results of comparing the supervised and unsupervised network against a randomly sampled initial guess are summarized in Table 3.5. This evaluation was performed for three robots: A 2D Arm with 5 DoF, the LWR III with 7 DoF, and Agile Justin with 19 DoF (see Fig. 3.5). In 3D, we used a shelf environment like depicted in Fig. 3.1b. Here, 10000 target frames were randomly sampled in the respective boxes of the shelf.

Table 3.4: Ablation study of the IK network predictions for Agile Justin

Training w. Boosting	Twin-Headed Network	Unit Vector Output	Feasibility
Yes	Yes	Yes	0.986
Yes	Yes	No	0.871
Yes	No	Yes	0.695
Yes	No	No	0.596
No	Yes	Yes	0.781
No	Yes	No	0.741
No	No	Yes	0.569
No	No	No	0.248

Table 3.5: Feasibility and Convergence for the different sampling modes for the warm-start of the IK Solver

Robots	DoF	Initial Guess	Mulit-Starts [#]	Feas. (1) [%]	Iterations [#]	Lenght Cost U_1^q [rad]
Flat Arm		Random	13.27 ± 5.41	19.7	12.87 ± 3.78	4.19 ± 0.75
Random World	5	Supervised	3.64 ± 1.29	81.3	9.93 ± 3.67	3.48 ± 0.71
		Unsupervised	3.35 ± 1.37	83.4	8.53 ± 3.29	3.41 ± 0.69
LWR III		Random	17.57 ± 4.53	14.4	15.69 ± 2.89	3.36 ± 0.68
Shelf World	7	Supervised	2.97 ± 0.61	88.7	9.31 ± 3.78	2.91 ± 0.70
		Unsupervised	3.06 ± 0.55	92.6	8.76 ± 4.01	2.85 ± 0.65
Agile Justin		Random	24.52 ± 7.18	8.3	13.88 ± 3.93	6.56 ± 0.93
Shelf World	19	Supervised	4.41 ± 0.94	88.9	6.91 ± 3.66	4.72 ± 0.41
		Unsupervised	4.29 ± 0.97	87.6	7.25 ± 3.61	4.10 ± 0.53

The overall orientation of the target frame was aligned with the shelf, and noise was added to ensure feasible yet challenging samples. The shelf environment is closer to a real-world setting and has notably different attributes than the random worlds the networks were trained on.

Table 3.5 shows that the average feasibility rate of the initial guesses from the networks outperforms the random baseline significantly for a single initial guess (denoted as (1)). Furthermore, the average number of iterations to converge is also decreased. The overall speed advantage can be seen directly from the difference in the necessary iterations. For the humanoid robot Agile Justin (19 DoF), the computation time for a single iteration is 0.8 ms on our testing machine. This leads to an overall solve time of under 10ms for the collision-free IK in unseen environments. The learned warm-starts outperform the random multi-starts in solving time, and the length cost (3.4) is reduced. These solutions are often more convenient and more accessible to integrate into larger motion planning tasks than random solutions.

Besides the improvement of the learning-based approaches over the random multi-start, it can also be seen that supervised and unsupervised training perform similarly well. Overall, this gives an advantage to the unsupervised method, as it requires far less

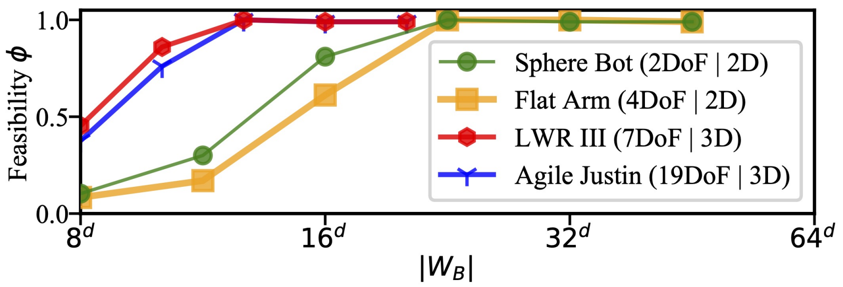


Figure 3.25: Influence of the size of BPS $|W_{\mathrm{B}}|$ on the feasibility ϕ of the network prediction after OMP. The $|W_{\mathrm{B}}|$ on the x-axis is scaled by the world dimension.

time to train, and no prior data generation and cleaning is needed.

3.5.4 Generalization to Self-Acquired Voxel Models

Prokudin et al. [PLR19] demonstrated that the BPS with fully connected layers is superior to occupancy maps with CNNs or point clouds with a PointNet architecture, both in terms of required network parameters and training performance. We can confirm those findings for motion planning. The large memory requirements in 3D made training prohibitively slow and made it hard to iterate on network architecture or training methods. Furthermore, looking towards the application on Agile Justin, the BPS can readily be integrated with the high-resolution SDFs acquired from the robot's depth camera [WFB13].

Fig. 3.25 shows that only a fraction of the number of pixels is enough as size for the basis point set to represent the simple worlds satisfactorily. Too few points cannot adequately represent the obstacles for the network to make valuable predictions. However, the required number of points is significantly smaller than the resolution of the underlying occupancy grid (64^d). Increasing the input layer to 64^3 basis points in 3D was not feasible. The BPS representation and the proposed training scheme on the worlds from simplex noise were robust enough to even generalize to some first results on the real robot (see Fig. 3.26 for motion planning and Fig. 3.27 for IK). Only trained on those random worlds, the networks are able to make valuable predictions from the data collected by Agile Justin's depth camera [WFB13]. For motion planning, the network predictions as warm-start for OMP could solve the unseen motion tasks that needed multi-starts otherwise in under 200 ms.

We also performed experiments to demonstrate the need for collision-free IK. Fig. 3.27 shows two table scenes; the robot should move the right end-effector to the same position in both cases, first without obstacle and then with an additional obstruction. The optimal solution to the IK for the simple scene does collide with the additional obstacle. The whole arm is stuck in the box on the table, and using this solution as a warm-start for our solver does not converge to a collision-free solution. However, using the neural network's prediction as an initial guess produces the correct collision-free solution shown on the right.

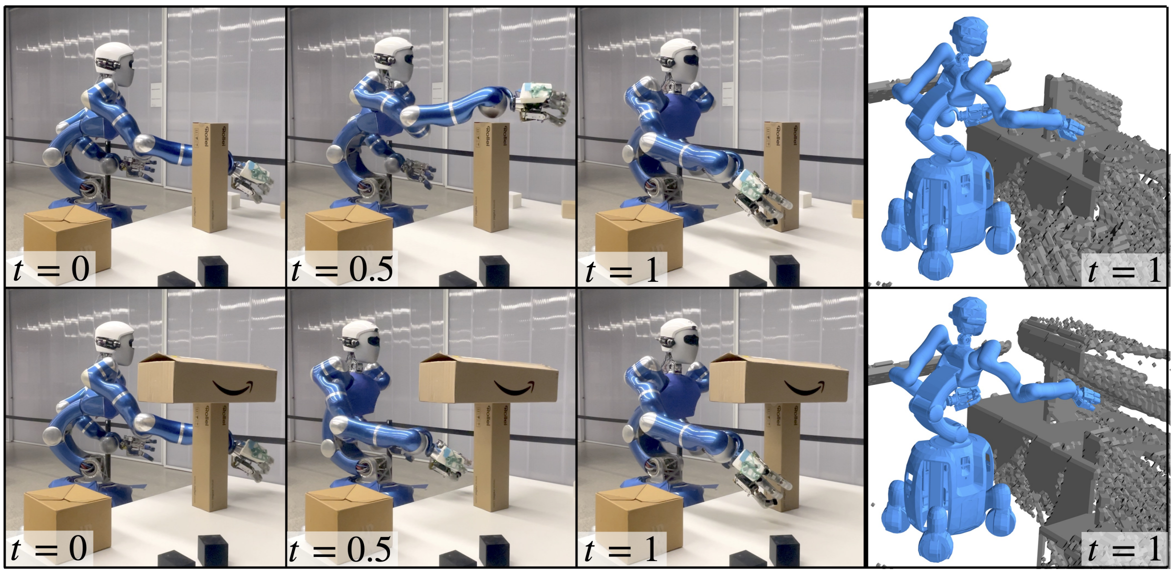


Figure 3.26: The motion of Agile Justin in two table scenes with boxes. The robot applies a different strategy for obstacle avoidance after the top route is blocked. The rendered images show the robot's self-acquired high-resolution voxel models [WFB13] of the scene, which are used as input for the neural networks.

3.6 Summary

We successfully trained motion planning and IK networks using different learning techniques in diverse and challenging settings. The network predictions come close to the global optimum for previously unseen environments. Using these predictions as a warm-start for optimization-based motion planning and IK massively outperforms random multi-start. For the complex robot Agile Justin with 19 DoF, motion planning takes only 200 ms and solving the IK 10 ms on a single CPU core.

One key to success is the basis point set encoding for the environment borrowed from computer vision, which we introduced to motion planning and scales well to high-resolution 3D worlds. For the supervised training, we autogenerated a training dataset of hard examples, i.e., situations for which the standard OMP struggles and for which the trained network should later provide an educated initial guess. We also introduced a scheme to further adapt the dataset during training by cleaning, boosting, and extending the dataset based on a metric defined by the current neural network and the objective function of the OMP. This approach leads to a challenging and consistent dataset on which a network can be trained and improved efficiently. Furthermore, we detailedly analyzed the IK problem with collision avoidance. The focus was on suitable ways to encode the learning problem and how to deal with the necessary mode switches. Multiple ablation studies demonstrated the relevance of those methods.

In the future, we want to combine motion planning and IK more thoroughly to tackle manipulation and grasping tasks more efficiently. The goal is a network that no longer

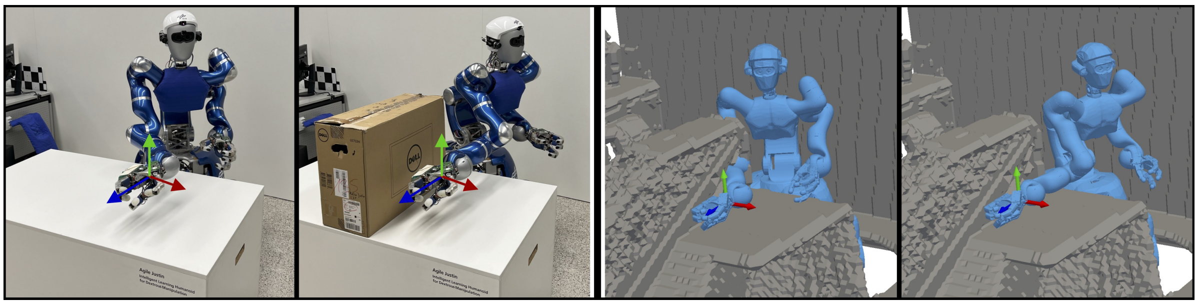


Figure 3.27: Difference between standard IK (left) and collision-free IK (right) for the humanoid robot Agile Justin. The rendered images show the robot's self-acquired high-resolution voxel model [WFB13] of the scene.

requires a goal configuration; only the goal pose of the end-effector has to be provided. We will also further investigate and increase the real-world capabilities of our method. As creating a vast amount of real-world data is expensive, our goal is for our architecture and the autogenerated dataset to allow for a robust transfer of the experience to real scenes.

4 Conclusion

Application on DLR's Agile Justin The importance of an accurately calibrated robot model and fast and efficient motion planning in unknown environments can be seen for the applications of Agile Justin in Tenhumberg et al. [Ten+24] and Fig. 1.1. This video submission demonstrates the state-of-the-art capabilities of the humanoid robot. Two different tasks are shown. The first is grasping of unknown objects [Win+22; Hum+23]. Here, the accuracy of the entire task chain from the internal camera to the hand is crucial for safe and stable grasps. Furthermore, the capability to compute the collision-free IK solution for multiple grasp poses and plan efficiently toward them becomes especially relevant for complex and cluttered scenes. The second task is building a tower out of wooden cubes, which combines efficient motion planning and tactile in-hand manipulation [SPB22; Pit+23] in order to rotate and place the lettered cubes correctly. Again, an accurately calibrated model of the humanoid and the robotic hand is crucial. Only then can this task be performed reliably. A fast motion and IK planner are central to demonstrating these dextrous skills. The results of this work are meanwhile an indispensable part of the daily work with the research robot and were crucial for successful performances at multiple fairs and conferences (most recent CoRL at TUM, November 2024).

Methodology Driven by the real application of this complex system and tackling actual problems that need to be solved, this thesis demonstrated how to achieve *fast* and *accurate* motion planning. Sound and efficient methods could be derived by thoroughly analyzing the problems and understanding the application's context and constraints. By applying first principles and combining well-established methods with modern learning methods, the achieved results could significantly improve upon the state of the art. The first part of this thesis dealt with the efficient, self-contained, and task-oriented calibration of complex mechatronic systems in order to achieve the necessary accuracy. It demonstrated the successful calibration of the tree-like DLR Hand II using only contact information and the calibration of the humanoid Agile Justin using its internal RGB camera. The maximal error at the end-effector could be reduced from 6 cm to 8 mm for the humanoid and from 18 mm to 4 mm for the robotic hand. This achieved precision is detrimental to utilizing efficient planning and learning methods for both systems.

The second part analyzed supervised and unsupervised learning techniques to speed up optimization-based motion planning and inverse kinematics solvers via educated warm-starts. It introduced a hybrid approach of a well-understood and safe optimizer combined with the global understanding of environment and robot kinematics encoded in a neural network. This combination allowed for the efficient use of experience. The robot no longer needs to plan from scratch for each new request and no longer relies on excessive multistarts for challenging situations. With its 19 degrees of freedom for Agile Justin, the planning time in unseen environments could be reduced from multiple seconds to 200 ms. Both improvements are significant steps towards more dexterity and autonomy and are already an indispensable part of the current work on the research robot.

Outlook The capabilities of robots and the areas of their application are increasing. Stemming from the industrial application with a focus on strength and repeatability, new modalities are added. By combining and utilizing multiple sensor modes, from seeing over touching and hearing, robots can build a highly accurate model of themselves and of the objects they manipulate and interact with. Robots can achieve a high level of dexterity when adding those sensing modalities to a mechatronic system with a precision that is similar to or even surpasses the human body. The proposed calibration techniques are a crucial step in this direction.

To create intelligent and dextrous robots, the learning techniques need to become far more data-efficient and adapt better to unseen and challenging situations. The current success of Large Languages Models can be mainly attributed to the vast amount of text available for their training. When comparing those enormous volumes of information to the tiny fraction a human can read and understand in a lifetime, it is apparent how inefficient and superficial the current training techniques are using the provided information. Nonetheless, even relying on those immense datasets and training efforts, they cannot fully generalize to difficult and unseen situations.

So far, there is no such complete database for robotics at hand. Furthermore, as the tasks become more challenging, and the time horizon of planning and the involved degrees of freedom increase, the datasets required would need to grow exponentially. Therefore, the learning techniques must be adapted to efficiently use available information and a high degree of generalization in unseen situations. The hybrid approach in this study provided a first step toward this goal. By encoding aspects of the problem via neural networks and still maintaining all guarantees of classical non-learning-based methods, the proposed approach utilized the inherent strength of the different techniques and its modular structure can be used as a basis for future research.

Bibliography

[Aga+10] Sameer Agarwal, Noah Snavely, Steven M Seitz, and Richard Szeliski. "Bundle Adjustment in the Large". In: *European Conference on Computer Vision (ECCV)*. Springer. 2010

[AT05] V.R. de Angulo and C. Torras. "Speeding Up the Learning of Robot Kinematics Through Function Decomposition". In: *IEEE Transactions on Neural Networks* (2005)

[AWI06] S.F.M. Assal, K. Watanabe, and K. Izumi. "Neural Network-Based Kinematic Inversion of Industrial Redundant Robots Using Cooperative Fuzzy Hint for the Joint Limits Avoidance". In: *IEEE/ASME Transactions on Mechatronics* (2006)

[AZ19] Zainab Al-Qurashi and Brian Ziebart. "Hybrid Algorithm for Inverse Kinematics Using Deep Learning and Coordinate Transformation". In: *IEEE International Conference on Robotic Computing (IRC)*. 2019

[BA15] Patrick Beeson and Barrett Ames. "TRAC-IK: An open-source library for improved solving of generic inverse kinematics". In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2015

[Bäu+14] B. Bäuml, T. Hammer, R. Wagner, O. Birbach, T. Gumpert, F. Zhi, U. Hillenbrand, S. Beer, W. Friedl, and J. Butterfass. "Agile Justin: An Upgraded Member of DLR's Family of Lightweight and Torque Controlled Humanoids". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014

[Bax+20] John Baxter, Mohammad Yousefi, Satomi Sugaya, Marco Morales, and Lydia Tapia. "Deep Prediction of Swept Volume Geometries: Robots and Resolutions". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020

[BB14] Oliver Birbach and Berthold Bäuml. "Calibrating a Pair of Inertial Sensors at Opposite Ends of an Imperfect Kinematic Chain". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2014

[Bel+07] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli E.and Klavins, and G. J. Pappas. "Symbolic Planning and Control of Robot Motion [Grand Challenges of Robotics]". In: *IEEE Robotics and Automation Magazine* (2007)

[BFB15] Oliver Birbach, Udo Frese, and Berthold Bäuml. "Rapid calibration of a multi-sensorial humanoid's upper body: An automatic and self-contained approach". In: *International Journal of Robotics Research* (2015)

[BH91] D.J. Bennett and J.M. Hollerbach. "Autonomous calibration of single-loop closed kinematic chains formed by manipulators with passive endpoint constraints". In: *IEEE Transactions on Robotics and Automation* (1991)

[Bis06] Christopher Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006

[BKC08] Dmitry Berenson, James Kuffner, and Howie Choset. "An Optimization Approach to Planning for Mobile Manipulation". In: *IEEE International Conference on Robotics and Automation (ICRA).* 2008

[Boc+11] Botond Bocsi, Duy Nguyen-Tuong, Lehel Csato, Bernhard Scholkopf, and Jan Peters. "Learning Inverse Kinematics with Structured Prediction". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2011

[BOG16] Pierre Besset, Adel Olabi, and Olivier Gibaru. "Advanced calibration applied to a collaborative robot". In: *IEEE International Power Electronics and Motion Control Conference (PEMC).* 2016

[BQY19] Mayur J. Bency, Ahmed H. Qureshi, and Michael C. Yip. "Neural Path Planning: Fixed Time, Near-Optimal Path Generation via Oracle Imitation". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2019

[CA90] J.L. Caenen and J.C. Angue. "Identification of geometric and nongeometric parameters of robots". In: *IEEE International Conference on Robotics and Automation (ICRA).* 1990

[Car+13] Henry Carrillo, Oliver Birbach, Holger Täubig, Berthold Bäuml, Udo Frese, and José A. Castellanos. "On Task-Oriented Criteria for Configurations Selection in Robot Calibration". In: *IEEE International Conference on Robotics and Automation (ICRA).* 2013

[Chi+21] Hao-Tien Lewis Chiang, John EG Baxter, Satomi Sugaya, Mohammad R Yousefi, Aleksandra Faust, and Lydia Tapia. "Fast deep swept volume estimator". In: *The International Journal of Robotics Research* (2021)

[CL16] Jie Chen and Henry Y. K. Lau. "Learning the Inverse Kinematics of Tendon-driven Soft Manipulators with K-nearest Neighbors Regression and Gaussian Mixture Regression". In: *International Conference on Control, Automation and Robotics (ICCAR).* 2016, pp. 103–107

[CLS16] Jing Chen, Tianbo Liu, and Shaojie Shen. "Online Generation of Collision-Free Trajectories for Quadrotor Flight in Unknown Cluttered Environments". In: *IEEE International Conference on Robotics and Automation (ICRA).* 2016

[CS17] Thomas Joseph Collinsm and Wei-Min Shen. "Particle Swarm Optimization for high-DOF Inverse Kinematics". In: *IEEE International Conference on Control, Automation and Robotics (ICCAR).* 2017

[CSK19] Constantinos Chamzas, Anshumali Shrivastava, and Lydia E. Kavraki. “Using Local Experiences for Global Motion Planning”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019

[CT15] Adria Colome and Carme Torras. “Closed-Loop Inverse Kinematics for Redundant Robots: Comparative Assessment and Two Enhancements”. In: *IEEE/ASME Transactions on Mechatronics* (2015)

[Dan02] David Daney. “Optimal Measurement Configurations for Gough Platform Calibration”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2002

[Den+20] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. “CvxNet: Learnable Convex Decomposition”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020

[DGD19] Jacket Demby’s, Yixiang Gao, and G. N. DeSouza. “A Study on Solving the Inverse Kinematics of Serial Robots using Artificial Neural Network and Fuzzy Neural Network”. In: *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. 2019

[DP90] Morris R. Driels and Uday S. Pathre. “Significance of Observation Strategy on the Design of Robot Calibration Experiments”. In: *Journal of Robotic Systems* (1990)

[DPM05] David Daney, Yves Papegay, and Blaise Madeline. “Choosing Measurement Poses for Robot Calibration with the Local Convergence Method and Tabu Search”. In: *The International Journal of Robotics Research* (2005)

[EL09] Alexandre Eudes and Maxime Lhuillier. “Error Propagations for Local Bundle Adjustment”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 2411–2418

[Eud+10] Alexandre Eudes, Sylvie Naudet-Collette, Maxime Lhuillier, and Michel Dhome. “Weighted Local Bundle Adjustment and Application to Odometry and Visual Slam Fusion”. In: *British Machine Vision Conference*. 2010

[Fan+15] Cheng Fang, Alessio Rocchi, Enrico Mingo Hoffman, Nikos G. Tsagarakis, and Darwin G. Caldwell. “Efficient Self-Collision Avoidance based on Focus of Interest for Humanoid Robots”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2015

[FL17] Chelsea Finn and Sergey Levine. “Deep Visual Foresight for Planning Robot Motion”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017

[FN11] Pietro Falco and Ciro Natale. “On the Stability of Closed-Loop Inverse Kinematics Algorithms for Redundant Robots”. In: *IEEE Transactions on Robotics (T-RO)* (2011)

[FSC21] Enrico Ferrentino, Federico Salvioli, and Pasquale Chiacchio. “Globally optimal redundancy resolution with dynamic programming for robot planning: A ros implementation”. In: *Robotics* (2021)

[Gia+22] Matthew Giamou, Filip Maric, David M. Rosen, Valentin Peretroukhin, Nicholas Roy, Ivan Petrovic, and Jonathan Kelly. “Convex Iteration for Distance-Geometric Inverse Kinematics”. In: *IEEE Robotics and Automation Letters* (2022)

[GJ85] E. Gilbert and D. Johnson. “Distance Functions and Their Application to Robot Path Planning in the Presence of Obstacles”. In: *IEEE Journal on Robotics and Automation* (1985)

[GJK88] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. “A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space”. In: *IEEE Journal on Robotics and Automation* (1988)

[GM11] Luciano Selva Ginani and José Maurício S. T. Motta. “Theoretical and Practical Aspects of Robot Calibration with Experimental Verification”. In: *Journal of the Brazilian Society of Mechanical Sciences and Engineering* (2011)

[GSB14] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. “Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2014

[GSB15] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. “Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015

[Gus05] Stefan Gustavson. *Simplex Noise Demystified*. 2005

[Hir+02] G. Hirzinger, N. Sporer, A. Albu-Schäffer, M. Hähnle, R. Krenn, A. Pascucci, and M. Schedl. “DLR’s Torque-Controlled Light Weight Robot III - are we Reaching the Technological Limits now?” In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2002

[HLH24] Tinghe Hong, Weibing Li, and Kai Huang. “A reinforcement learning enhanced pseudo-inverse approach to self-collision avoidance of redundant robots”. In: *Frontiers in Neurorobotics* (2024)

[HSB12] Uwe Hubert, Jorg Stuckler, and Sven Behnke. “Bayesian Calibration of the Hand-Eye Kinematics of an Anthropomorphic Robot”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2012

[Hum+23] Matthias Humt, Dominik Winkelbauer, Ulrich Hillenbrand, and Berthold Bäuml. “Combining Shape Completion and Grasp Prediction for Fast and Versatile Grasping with a Multi-Fingered Hand”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2023

[HW96] John M. Hollerbach and Charles W. Wampler. “The Calibration Index and Taxonomy for Robot Kinematic Calibration Methods”. In: *The International Journal of Robotics Research* (1996)

[HXZ08] Chenhua Huang, Cunxi Xie, and Tie Zhang. “Determination of Optimal Measurement Configurations for Robot Calibration Based on Observability Measure”. In: *IEEE International Conference on Information and Automation (ICIA)*. 2008

[IH97] M. Ikits and J.M. Hollerbach. “Kinematic calibration using a plane constraint”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 1997

[IHP18] Brian Ichter, James Harrison, and Marco Pavone. “Learning Sampling Distributions for Robot Motion Planning”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018

[Ila+17] Viorela Ila, Lukas Polok, Marek Solony, and Klemen Istenic. “Fast Incremental Bundle Adjustment with Covariance Recovery”. In: *International Conference on 3D Vision (3DV)*. 2017

[J B01] J. Butterfaßand M. Grebenstein and H. Liu and G. Hirzinger. “DLR-Hand II: Next Generation of a Dextrous Robot Hand”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2001

[Jan+22] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. “Planning with Diffusion for Flexible Behavior Synthesis”. In: *International Conference on Machine Learning*. PMLR, 2022

[JB15] Ahmed Joubair and Ilian A. Bonev. “Kinematic calibration of a six-axis serial robot using distance and sphere constraints”. In: *The International Journal of Advanced Manufacturing Technology* (2015)

[Jég+17] Simon Jégou, Michal Drozdzal, David Vázquez, Adriana Romero, and Yoshua Bengio. “The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017

[Jia+20] Zhouxiang Jiang, Min Huang, Xiaoqi Tang, Bao Song, and Yixuan Guo. “Observability index optimization of robot calibration based on multiple identification spaces”. In: *Autonomous Robots* (2020)

[JT13] Nikolay Jetchev and Marc Toussaint. “Fast motion planning from experience: Trajectory prediction for speeding up movement generation”. In: *Autonomous Robots* (2013)

[JT19] Tom Jurgenson and Aviv Tamar. “Harnessing Reinforcement Learning for Neural Motion Planning”. In: *Robotics: Science and Systems* (2019)

[Kal+11] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. “STOMP: Stochastic trajectory optimization for motion planning”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 2011

[Kar+11] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. “Anytime Motion Planning using the RRT*”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 2011

[Kav+96] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. “Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces”. In: *IEEE Transactions on Robotics and Automation* (1996)

[KB02] Wisama Khalil and Sébastien Besnard. “Geometric Calibration of Robots with Flexible Joints and Links”. In: *Journal of Intelligent and Robotic Systems* (2002)

[KB19] Kaveh Kamali and Ilian A. Bonev. “Optimal Experiment Design for Elasto-Geometrical Calibration of Industrial Robots”. In: *IEEE/ASME Transactions on Mechatronics* (2019)

[KL00] James J. Kuffner and Steven M. LaVall. “RRT-Connect: An Efficient Approach to Single-Query Path Planning”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 2000

[KL15] Jeong-Jung Kim and Ju-Jang Lee. “Trajectory Optimization With Particle Swarm Optimization for Manipulator Motion Planning”. In: *IEEE Transactions on Industrial Informatics* (2015)

[Kli+12] Alexandr Klimchik, Anatol Pashkevich, Yier Wu, Benoit Furet, and Stephane Caro. *Optimization of measurement configurations for geometrical calibration of industrial robot.* Tech. rep. Ecole des Mines de Nantes, 2012

[Klo+11] Julian Klodmann, Rainer Konietschke, Alin Albu-Schaffer, and Gerhard Hirzinger. “Static Calibration of the DLR Medical Robot MIRO, a Flexible Lightweight Robot with Integrated Torque Sensors”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2011

[KRS18] Daniel Kubus, Rania Rayyes, and Jochen J. Steil. “Learning Forward and Inverse Kinematics Maps Efficiently”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2018

[LA18] Peter Lehner and Alin Albu-Schäffer. “The Repetition Roadmap for Repetitive Constrained Motion Planning”. In: *IEEE Robotics and Automation Letters* (2018)

[LaV06] Steven M. LaValle. *Planning Algorithms.* Cambridge University Press, 2006

[LB90] S. P. Ladany and D. Ben-Arieh. “Optimal Industrial Robot-Calibration Policy”. In: *The International Journal of Advanced Manufacturing Technology* (1990)

[Lee+13] Sang-Mun Lee, Kyoung-Don Lee, Sang-Hyuek Jung, and Tae-Sung Noh. "Kinematic Calibration System of Robot Hands using Vision Cameras". In: *International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. Ieee, 2013

[Lee13] Bum-Joo Lee. "Geometrical Derivation of Differential Kinematics to Calibrate Model Parameters of Flexible Manipulator". In: *International Journal of Advanced Robotic Systems* (2013)

[Lem+20] Teguh Santoso Lembono, Antonio Paolillo, Emmanuel Pignat, and Sylvain Calinon. "Memory of Motion for Warm-Starting Trajectory Optimization". In: *IEEE Robotics and Automation Letters* (2020)

[Lem+21] Teguh Santoso Lembono, Emmanuel Pignat, Julius Jankowski, and Sylvain Calinon. "Learning Constrained Distributions of Robot Configurations With Generative Adversarial Network". In: *IEEE Robotics and Automation Letters* (2021)

[Lia+17] Shiqi Lian, Yinhe Han, Ying Wang, Yungang Bao, Hang Xiao, Xiaowei Li, and Ninghui Sun. "Dadu: Accelerating Inverse Kinematics for high-DOF robots". In: *ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2017

[Liu+23] Baolin Liu, Gedong Jiang, Fei Zhao, and Xuesong Mei. "Collision-Free Motion Generation Based on Stochastic Optimization and Composite Signed Distance Field Networks of Articulated Robot". In: *IEEE Robotics and Automation Letters* (2023)

[LLL21] Zhibin Li, Shuai Li, and Xin Luo. "An Overview of Calibration Technology of Industrial Robots". In: *IEEE/CAA Journal of Automatica Sinica* (2021)

[LMT19] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. "Conditional Generative Neural System for Probabilistic Trajectory Prediction". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019

[LRA22] Peter Lehner, Máximo A. Roa, and Alin Albu-Schäffer. "Kinematic Transfer Learning of Sampling Distributions for Manipulator Motion Planning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2022

[Luo+21] Guoyue Luo, Lai Zou, Ziling Wang, Chong Lv, Jing Ou, and Yun Huang. "A novel kinematic parameters calibration method for industrial robot based on Levenberg-Marquardt and Differential Evolution hybrid algorithm". In: *Robotics and Computer-Integrated Manufacturing* (2021)

[LW08] Qi Liu and Dongshu Wang. "Optimal Measurement Configurations for Robot Calibration based on Modified Simulated Annealing Algorithm". In: *World Congress on Intelligent Control and Automation*. Ieee, 2008

[Ma96] Sang De Ma. "A self-calibration technique for active vision systems". In: *IEEE Transactions on Robotics and Automation* (1996)

[MD00] M.A. Meggiolaro and S. Dubowsky. “An analytical method to eliminate the redundant parameters in robot calibration”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 2000

[Mit00] Toby J. Mitchell. “An Algorithm for the Construction of ”D-Optimal” Experimental Designs”. In: *Technometrics* (2000)

[MIV18] Wolfgang Merkt, Vladimir Ivan, and Sethu Vijayakumar. “Leveraging Precomputation with Problem Encoding for Warm-Starting Trajectory Optimization in Complex Environments”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2018

[MWB15] Daniel Maier, Stefan Wrobel, and Maren Bennewitz. “Whole-Body Self-Calibration via Graph-Optimization and Automatic Configuration Selection”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 2015

[NH96] Ali Nahvi and J.M. Hollerbach. “The Noise Amplification Index for Optimal Pose Selection in Robot Calibration”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 1996

[NLK19] Hoai-Nhan Nguyen, Phu-Nguyen Le, and Hee-Jun Kang. “A new calibration method for enhancing robot position accuracy by combining a robot model–based identification approach and an artificial neural network–based error compensation technique”. In: *Advances in Mechanical Engineering* (2019)

[Ort+18] Andreas Orthey, Olivier Roussel, Olivier Stasse, and Michel Taix. “Motion planning in Irreducible Path Spaces”. In: *Robotics and Autonomous Systems* (2018)

[PD90] U. S. Pathre and M. R. Driels. “Simulation Experiments in Parameter Identification for Robot Calibration”. In: *The International Journal of Advanced Manufacturing Technology* (1990)

[Per01] Ken Perlin. *Chapter 2 Noise Hardware.* 2001

[Pfe+17] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. “From Perception to Decision: A Data-driven Approach to End-to-end Motion Planning for Autonomous Ground Robots”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 2017

[Pit+23] Johannes Pitz, Lennart Röstel, Leon Sievers, and Berthold Bäuml. “Dextrous Tactile In-Hand Manipulation Using a Modular Reinforcement Learning Architecture”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 2023

[PK11] In-Won Park and Jong-Hwan Kim. “Estimating Entire Geometric Parameter Errors of Manipulator Arm Using Laser Module and Stationary Camera”. In: *Industrial Electronics Conference (IECON).* 2011

[PKB14] Vijay Pradeep, Kurt Konolige, and Eric Berger. "Calibrating a Multi-arm Multi-sensor Robot: A Bundle Adjustment Approach". In: Experimental Robotics ,Springer, 2014

[PLC21] Michal Pandy, Daniel Lenton, and Ronald Clark. "Unsupervised Path Regression Networks". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2021

[PLR19] Sergey Prokudin, Christoph Lassner, and Javier Romero. "Efficient Learning on Point Clouds with Basis Point Sets". In: *International Conference on Computer Vision (ICCV).* 2019

[PM14] Chonhyon Park and Dinesh Manocha. "Fast and Dynamically Stable Optimization based Planning for High-DOF Human-like Robots". In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids).* 2014

[PPM12] Chonhyon Park, Jia Pan, and Dinesh Manocha. "ITOMP: Incremental Trajectory Optimization for Real-Time Replanning in Dynamic Environments". In: *Int. Conf. on Automated Planning and Scheduling (ICAPS).* 2012

[Qi+17] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. "PointNet: Deep learning on point sets for 3D classification and segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2017

[Qur+19] Ahmed H. Qureshi, Anthony Simeonov, Mayur J. Bency, and Michael C. Yip. "Motion Planning Networks". In: *IEEE International Conference on Robotics and Automation (ICRA).* 2019

[Qur+21] Ahmed Hussain Qureshi, Yinglong Miao, Anthony Simeonov, and Michael C. Yip. "Motion Planning Networks: Bridging the Gap between Learning-Based and Classical Motion Planners". In: *IEEE Transactions on Robotics (T-RO)* (2021)

[RK92] E. Rimon and D.E. Koditschek. "Exact Robot Navigation Using Artificial Potential Functions". In: *IEEE Transactions on Robotics and Automation* (1992)

[RMR87] Z. Roth, B. Mooring, and B. Ravani. "An Overview of Robot Calibration". In: *IEEE Journal on Robotics and Automation* (1987)

[Ron+14] Alessandro Roncone, Matej Hoffmann, Ugo Pattacini, and Giorgio Metta. "Automatic kinematic chain calibration using artificial skin: Self-touch in the iCub humanoid robot". In: *IEEE International Conference on Robotics and Automation (ICRA).* 2014

[RSG09] Matthias Rolf, Jochen J. Steil, and Michael Gienger. "Efficient exploration and learning of whole body kinematics". In: *IEEE International Conference on Development and Learning.* 2009

[RSG10] Matthias Rolf, Jochen J. Steil, and Michael Gienger. “Goal Babbling Permits Direct Learning of Inverse Kinematics”. In: *IEEE Transactions on Autonomous Mental Development* (2010)

[RUG17] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. “OctNet: Learning Deep 3D Representations at High Resolutions”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2017

[Rus+21] Lukas Rustler, Bohumila Potocna, Michal Polic, Karla Stepanova, and Matej Hoffmann. “Spatial calibration of whole-body artificial skin on a humanoid robot: comparing self-contact, 3D reconstruction, and CAD-based calibration”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids).* 2021

[SAL15] Matteo Saveriano, Sang-ik An, and Dongheui Lee. “Incremental Kinesthetic Teaching of End-Effector and Null-Space Motion Primitives”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 2015

[SB11] N. Sukavanam and R. Balasubramanian. “An Optimization Approach to Solve the Inverse Kinematics of Redundant Manipulator”. In: *International Journal of Information And Systems Sciences* (2011)

[Sch+14] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. “Motion Planning with Sequential Convex Optimization and Convex Collision Checking”. In: *International Journal of Robotics Research* (2014)

[SGK09] Yunquan Sun, David J. Giblin, and Kazem Kazerounian. “Accurate robotic belt grinding of workpieces with complex geometries using relative calibration techniques”. In: *Robotics and Computer-Integrated Manufacturing* (2009)

[SH08] Yu Sun and John M. Hollerbach. “Observability Index Selection for Robot Calibration”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 2008

[Sib+09] Dieter Sibley, Christopher Mei, Ian D Reid, and Paul Newman. “Adaptive Relative Bundle Adjustment”. In: *Robotics: Science and Systems* (2009)

[SPB22] Leon Sievers, Johannes Pitz, and Berthold Bäuml. “Learning Purely Tactile In-Hand Manipulation with a Torque-Controlled Hand”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 2022

[SPH19] Karla Stepanova, Tomas Pajdla, and Matej Hoffmann. “Robot Self-Calibration Using Multiple Kinematic Chains—A Simulation Study on the iCub Humanoid Robot”. In: *IEEE Robotics and Automation Letters* (2019)

[Ste+22] Karla Stepanova, Jakub Rozlivek, Frantisek Puciow, Pavel Krsek, Tomas Pajdla, and Matej Hoffmann. "Automatic self-contained calibration of an industrial dual-arm robot with cameras using self-contact, planar constraints, and self-observation". In: *Robotics and Computer-Integrated Manufacturing* (2022)

[Str+20] Robin Strudel, Ricardo Garcia, Justin Carpentier, Jean-Paul Laumond, Ivan Laptev, and Cordelia Schmid. "Learning Obstacle Representations for Neural Motion Planning". In: *Conference on Robot Learning (CoRL)*. 2020

[Sug11] Tomomichi Sugihara. "Solvability-Unconcerned Inverse Kinematics by the Levenberg–Marquardt Method". In: *IEEE Transactions on Robotics (T-RO)* (2011)

[TBF11] Holger Täubig, Berthold Bäuml, and Udo Frese. "Real-time Swept Volume and Distance Computation for Self Collision Detection". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 1585–1592

[TC20] Alessandro Tringali and Silvio Cocuzza. "Globally Optimal Inverse Kinematics Method for a Redundant Robot Manipulator with Linear and Nonlinear Constraints". In: *Robotics* (2020)

[TGR18] Ning Tan, Xiaoyi Gu, and Hongliang Ren. "Simultaneous Robot-World, Sensor-Tip, and Kinematics Calibration of an Underactuated Robotic Hand With Soft Fingers". In: *IEEE Access* (2018)

[Tou14] Marc Toussaint. *Newton Methods for k-order Markov Constrained Motion Problems*. Github: MarcToussaint/KOMO. 2014

[Tri+00] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. "Bundle Adjustment - A Modern Synthesis". In: *International Workshop on Vision Algorithms - Theory and Practice*. Springer. 2000

[Tru+22] Pavel Trutman, Mohab Safey El Din, Didier Henrion, and Tomas Pajdla. "Globally Optimal Solution to Inverse Kinematics of 7DOF Serial Manipulator". In: *IEEE Robotics and Automation Letters* (2022)

[TS00] Guarav. Tevatia and Stefan Schaal. "Inverse Kinematics for Humanoid Robots". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2000

[TVY20] Tony Tohme, Kevin Vanslette, and Kamal Youcef-Toumi. "A Generalized Bayesian Approach to Model Calibration". In: *Reliability Engineering and System Safety* (2020)

[VH08] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing Data Using t-SNE". In: *Journal of Machine Learning Research* (2008)

[VW88] W.K. Veitschegger and C.-H. Wu. "Robot Calibration and Compensation". In: *IEEE Journal on Robotics and Automation* (1988)

[Wan+20] Jiankun Wang, Wenzheng Chi, Chenming Li, Chaoqun Wang, and Max Q.-H. Meng. “Neural RRT*: Learning-Based Optimal Path Planning”. In: *IEEE Transactions on Automation Science and Engineering* (2020)

[WFB13] Réne Wagner, Udo Frese, and Berthold Bäuml. “3D modeling, distance and gradient computation for motion planning: A direct GPGPU approach”. In: *IEEE International Conference on Robotics and Automation (ICRA).* 2013

[WFC19] Karl Van Wyk, Joe Falco, and Geraldine Cheok. “Efficiently Improving and Quantifying Robot Accuracy In Situ”. In: *IEEE Transactions on Automation Science and Engineering* (2019)

[Win+22] Dominik Winkelbauer, Berthold Bäuml, Matthias Humt, Nils Thuerey, and Rudolph Triebel. “A Two-stage Learning Architecture that Generates High-Quality Grasps for a Multi-Fingered Hand”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2022

[Wu+15] Yier Wu, Alexandr Klimchik, Stéphane Caro, Benoit Furet, and Anatol Pashkevich. “Geometric calibration of industrial robots using enhanced partial pose measurements and design of experiments”. In: *Robotics and Computer-Integrated Manufacturing* (2015)

[Xio+17] Gang Xiong, Ye Ding, LiMin Zhu, and Chun-Yi Su. “A Product-of-Exponential-based Robot Calibration Method with Optimal Measurement Configurationsptimal Measurement Configurations”. In: *International Journal of Advanced Robotic Systems* (2017)

[YCX18] Chengyi Yu, Xiaobo Chen, and Juntong Xi. “Determination of optimal measurement configurations for self-calibrating a robotic visual inspection system with multiple point constraints”. In: *The International Journal of Advanced Manufacturing Technology* (2018)

[Zai+21] Yuval Zaidel, Albert Shalumov, Alex Volinski, Lazar Supic, and Elishai Ezra Tsur. “Neuromorphic NEF-Based Inverse Kinematics and PID Control”. In: *Frontiers in Neurorobotics* (2021)

[Zen+18] Rodrigo Zenha, Pedro Vicente, Lorenzo Jamone, and Alexandre Bernardino. “Incremental Adaptation of a Robot Body Schema Based on Touch Events”. In: *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob).* 2018

[Zha+21] Liangliang Zhao, Zainan Jiang, Yongjun Sun, Jingdong Zhao, and Hong Liu. “Collision-Free Kinematics for Hyper-Redundant Manipulators in Dynamic Scenes using Optimal Velocity Obstacles”. In: *International Journal of Advanced Robotic Systems* (2021)

[ZKR10] Jian Zhou, Hee-Jun Kang, and Young-Shick Ro. “Comparison of the Observability Indices for Robot Calibration considering Joint Stiffness Parameters”. In: *Advanced Intelligent Computing Theories and Applications.* 2010

[ZLB21] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. "Optimization-Based Collision Avoidance". In: *IEEE Transactions on Control Systems Technology* (2021)

[ZNK14a] Jian Zhou, Hoai-Nhan Nguyen, and Hee-Jun Kang. "Selecting Optimal Measurement Poses for Kinematic Calibration of Industrial Robots". In: *Advances in Mechanical Engineering* (2014)

[ZNK14b] Jian Zhou, Hoai-Nhan Nguyen, and Hee-Jun Kang. "Simultaneous Identification of Joint Compliance and Kinematic Parameters of Industrial Robots". In: *International Journal of Precision Engineering and Manufacturing* (2014)

[Zuc+13] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. "CHOMP: Covariant Hamiltonian Optimization for Motion Planning". In: *The International Journal of Robotics Research* (2013)

[ZWH97] Hanqi Zhuang, Jie Wu, and Weizhen Huang. "Optimal Planning of Robot Calibration Experiments by Genetic Algorithms". In: *Journal of Robotic Systems* (1997)

Publications of the Author

Core Publications

[1] Johannes Tenhumberg and Berthold Bäuml. “Calibration of an Elastic Humanoid Upper Body and Efficient Compensation for Motion Planning”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2021.
DOI: 10.1109/HUMANOIDS47582.2021.9555793.
eprint: https://ieeexplore.ieee.org/document/9555793/.
URL: https://aidx-lab.org/2021-humanoids-elastic/.

Core Publication (1.0)

CRediT: **J. Tenhumberg**: Conceptualization, data curation, investigation, methodology, software, visualization, writing – original draft & review & editing **B. Bäuml**: Conceptualization, formal analysis, methodology, supervision, writing – review & editing

[2] Johannes Tenhumberg, Dominik Winkelbauer, Darius Burschka, and Berthold Bäuml. “Self-Contained Calibration of an Elastic Humanoid Upper Body Using Only a Head-Mounted RGB Camera”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2022.
DOI: 10.1109/Humanoids53995.2022.10000184.
eprint: https://ieeexplore.ieee.org/document/10000184/.
URL: https://aidx-lab.org/2022-humanoids-rgb/.

Core Publication (1.0)

CRediT: **J. Tenhumberg**: Conceptualization, data curation, investigation, methodology, software, visualization, writing – original draft & review & editing **D. Winkelbauer**: Conceptualization, data curation, methodology, software **D. Burschka**: Supervision **B. Bäuml**: Conceptualization, methodology, software, supervision, writing – review & editing

[3] Johannes Tenhumberg, Leon Sievers, and Berthold Bäuml. “Self-Contained and Automatic Calibration of a Multi-Fingered Hand Using Only Pairwise Contact Measurements”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2023.
DOI: 10.1109/Humanoids57100.2023.10375208.
eprint: https://ieeexplore.ieee.org/document/10375208/.
URL: https://aidx-lab.org/2023-humanoids-contact/.
video: https://www.youtube.com/watch?v=dkG9xz1fhOU.

Core Publication (0.5)

CRediT: **J. Tenhumberg**: Conceptualization, data curation, investigation, methodology, software, visualization, writing – original draft & review & editing **L. Sievers**: Conceptualization, data curation, investigation, methodology, software, writing – review & editing **B. Bäuml**: Conceptualization, methodology, supervision, writing – review & editing

[4] Johannes Tenhumberg, Darius Burschka, and Berthold Bäuml. "Speeding Up Optimization-based Motion Planning through Deep Learning". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022.
DOI: `10.1109/IROS47612.2022.9981717`.
eprint: `https://ieeexplore.ieee.org/document/9981717/`.
URL: `https://aidx-lab.org/2022-iros-planning/`.
video: `https://www.youtube.com/watch?v=fKe1_vUNCew`.

Core Publication (1.0)

CRediT: **J. Tenhumberg**: Conceptualization, investigation, data curation, methodology, software, visualization, writing – original draft & review & editing **D. Burschka**: Supervision **B. Bäuml**: Conceptualization, methodology, supervision, writing – review & editing

[5] Johannes Tenhumberg, Arman Mielke, and Berthold Bäuml. "Efficient Learning of Fast Inverse Kinematics with Collision Avoidance". In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2023.
DOI: `10.1109/Humanoids57100.2023.10375143`.
eprint: `https://ieeexplore.ieee.org/document/10375143/`.
URL: `https://aidx-lab.org/2023-humanoids-ik/`.
video: `https://www.youtube.com/watch?v=k96r7l2s384`.

Core Publication (0.5)

CRediT: **J. Tenhumberg**: Conceptualization, investigation, methodology, software, visualization, writing – original draft & review & editing **A. Mielke**: Conceptualization, data curation, software, methodology, writing – review & editing **B. Bäuml**: Conceptualization, methodology, software, supervision, writing – review & editing

Additional Publications

[Ten+24] Johannes Tenhumberg, Leon Sievers, Dominik Winkelbauer, Lennart Röstel, Johannes Pitz, Matthias Humt, Ulrich Hillenbrand, Jörg Butterfass, Werner Friedl, Thomas Gumpert, and Berthold Bäuml. "Intelligent Dextrous Manipulation: Humanoid Agile Justin Meets Learning AI". In: *ICRA IEEE International Conference on Robotics and Automation*. 2024.
URL: `https://aidx-lab.org`.
video: `https://www.youtube.com/watch?v=CZBMXDM1_Tk`
stand-alone video submission with accomanying one-page paper

[TBB24] Johannes Tenhumberg, Darius Burschka, and Berthold Bäuml. "Unified Task-Oriented Robot Calibration". In: *IEEE Transactions on Robotics (T-RO)* (2024). **submitted to IEEE Journal Transactions on Robotics (T-RO) and currently under review (November 2024)**

Publication [1]

Calibration of an Elastic Humanoid Upper Body and Efficient Compensation for Motion Planning

Summary

High absolute accuracy is an essential prerequisite for a humanoid robot to autonomously and robustly perform manipulation tasks while avoiding obstacles. We present for the first time a kinematic model for a humanoid upper body incorporating joint and transversal elasticities. These elasticities lead to significant deformations due to the robot's own weight, and the resulting model is implicitly defined via a torque equilibrium. We successfully calibrate this model for DLR's humanoid Agile Justin, including all Denavit-Hartenberg parameters and elasticities. The calibration is formulated as a combined least-squares problem with priors and based on measurements of the end effector positions of both arms via an external tracking system. The absolute position error is massively reduced from 21 mm to 3.1 mm on average in the whole workspace. Using this complex and implicit kinematic model in motion planning is challenging. We show that for optimization-based path planning, integrating the iterative solution of the implicit model into the optimization loop leads to an elegant and highly efficient solution. For mildly elastic robots like Agile Justin, there is no performance impact, and even for a simulated highly flexible robot with 20 times higher elasticities, the runtime increases by only 30 %.

Contribution

The author of the dissertation contributed 90 % to this paper. He designed the methodology, wrote the software tools to collect the measurements, and performed the full-body calibration, including the lateral elasticities. In this context, he also carried out the experimental validation of the methods. Furthermore, he designed and implemented the algorithm to efficiently compensate the implicit forward kinematics for OMP. Overall, he wrote the original draft and created the graphs and visualizations. Berthold Bäuml supervised the work, helped to derive the equations, and reviewed and edited the paper.

Version

The attached version of the paper is identical to the peer-reviewed, accepted, and published version available under `https://ieeexplore.ieee.org/document/9555793/` Additional information, animations, videos and other material can be found on the paper website `https://aidx-lab.org/2021-humanoids-elastic/`.

```
@inproceedings{Tenhumberg2021elastic,
author = {Johannes Tenhumberg and Berthold Bäuml},
title = {Calibration of an Elastic Humanoid Upper Body and Efficient Compensation for Motion Planning},
booktitle = {IEEE-RAS International Conference on Humanoid Robots (Humanoids)},
year = {2021},
doi = {10.1109/HUMANOIDS47582.2021.9555793}}
```

Publication [2]

Self-Contained Calibration of an Elastic Humanoid Upper Body Using Only a Head-Mounted RGB Camera

Summary

When a humanoid robot performs a manipulation task, it first makes a model of the world using visual sensors and then plans the motion of its body in this model. For this, precise calibration of the camera parameters and the kinematic tree is needed. Besides the accuracy of the calibrated model, the calibration process should be fast and self-contained, i.e., no external measurement equipment should be used. Therefore, we extend our prior work on calibrating the elastic upper body of DLR's Agile Justin by now using only its internal head-mounted RGB camera. We use simple visual markers at the ends of the kinematic chain and one in front of the robot, mounted on a pole, to get measurements for the whole kinematic tree. To ensure that the task-relevant cartesian error at the end-effectors is minimized, we introduce virtual noise to fit our imperfect robot model so that the pixel error has a higher weight if the marker is further away from the camera. This correction reduces the cartesian error by more than 20 %, resulting in a final accuracy of 3.9 mm on average and 9.1 mm in the worst case. This way, we achieve the same precision as in our previous work, where an external cartesian tracking system was used.

Contribution

The author of the dissertation contributed 70 % to this paper. He extended the prior work on calibration to be self-contained and task-oriented using the internal RGB camera. He carried out the experiments on the robot and wrote the software for the calibration and its detailed evaluation. Furthermore, he wrote the original draft and created the graphs and visualizations for this paper. Berthold Bäuml supervised the work and formalized the concept of virtual noise. This general concept allowed a significant error reduction in the relevant cartesian task space.

Version

The attached version of the paper is identical to the peer-reviewed, accepted, and published version available under `https://ieeexplore.ieee.org/document/10000184/` Additional information, animations, videos and other material can be found on the paper website `https://aidx-lab.org/2022-humanoids-rgb/`.

```
@inproceedings{Tenhumberg2022rgb,
author = {Johannes Tenhumberg, Dominik Winkelbauer, Darius Burschka, and Berthold Bäuml},
title = {Self-Contained Calibration of an Elastic Humanoid Upper Body Using Only a Head-Mounted RGB
Camera},
booktitle = {IEEE-RAS International Conference on Humanoid Robots (Humanoids)},
year = {2022},
doi = {10.1109/Humanoids53995.2022.10000184}}
```

Publication [3]

Self-Contained and Automatic Calibration of a Multi-Fingered Hand Using Only Pairwise Contact Measurements

Summary

A self-contained calibration procedure that can be performed automatically without additional external sensors or tools is a significant advantage, especially for complex robotic systems. Here, we show that the kinematics of a multi-fingered robotic hand can be precisely calibrated only by moving the tips of the fingers pairwise into contact. The only prerequisite for this is sensitive contact detection, e.g., by torque-sensing in the joints (as in our DLR-Hand II) or tactile skin. The measurement function for a given joint configuration is the distance between the modeled fingertip geometries, but the actual measurement is always zero. In an in-depth analysis, we prove that this contact-based calibration determines all quantities needed for manipulating objects with the hand, i.e., the difference vectors of the fingertips, and that it is as sensitive as a calibration using an external visual tracking system and markers. We describe the complete calibration scheme, including the selection of optimal sample joint configurations and search motions for the contacts despite the initial kinematic uncertainties. In a real-world calibration experiment for the torque-controlled four-fingered DLR-Hand II, the maximal error of 17.7 mm can be reduced to only 3.7 mm.

Contribution

The author of the dissertation contributed 50 % to this paper. He wrote the software for the entire calibration process. He conceptualized and then carried out the experiments in simulation to compare the different measurement functions and the influence of different calibration sets on the final performance. In addition, he wrote the original draft and he create the graphs and visualizations for this paper. Leon Sievers helped with the first concept of the contact-based calibration and performed the experiments on the actual hardware. His work included the contact detection with high sensitiviy. Berthold Bäuml supervised the work and helped with revieweing and editing the paper.

Version

The attached version of the paper is identical to the peer-reviewed, accepted, and published version available under `https://ieeexplore.ieee.org/document/10375208/` Additional information, animations, videos and other material can be found on the paper website `https://aidx-lab.org/2023-humanoids-contact/`.

@inproceedings{Tenhumberg2023contact,
author = {Johannes Tenhumberg, Leon Sievers, and Berthold Bäuml},
title = {Self-Contained and Automatic Calibration of a Multi-Fingered Hand Using Only Pairwise Contact Measurements},
booktitle = {IEEE-RAS International Conference on Humanoid Robots (Humanoids)},
year = {2023},
doi = {10.1109/Humanoids57100.2023.10375208}}

Publication [4]
Speeding Up Optimization-based Motion Planning through Deep Learning

Summary

Planning collision-free motions for robots with many degrees of freedom is challenging in environments with complex obstacle geometries. Recent work introduced the idea of speeding up the planning by encoding prior experience of successful motion plans in a neural network. However, this "neural motion planning" did not scale to complex robots in unseen 3D environments as needed for real-world applications. Here, we introduce "basis point set", well-known in computer vision, to neural motion planning as a modern compact environment encoding enabling efficient supervised training networks that generalize well over diverse 3D worlds. Combined with a new elaborate training scheme, we reach a planning success rate of 100 %. We use the network to predict an educated initial guess for an optimization-based planner (OMP), which quickly converges to a feasible solution, massively outperforming random multi-starts when tested on previously unseen environments. For the DLR humanoid Agile Justin with 19 DoF and in challenging obstacle environments, optimal paths can be generated in 200 ms using only a single CPU core. We also show a first successful real-world experiment based on a high-resolution world model from an integrated 3D sensor.

Contribution

The author of the dissertation contributed 80 % to this paper. He conceptualized and wrote the software on which this paper is based. First he implemented an optimization-basd motion planner for the 19 DoF robot Agile Justin. The solver can handle passive joints, self-collision, and collision with a hihg-resolution non-convex environment. On top he build a pipeline to generate training data efficiently. Finally he performed the experiments with different neural networks, to evaluate the whole method. Berthold Bäuml first introduced the concept of speeding-up optimization-based planner with the predictions of neural networks. Overall, he supervised the work and helped with revieweing and editing the paper.

Version

The attached version of the paper is identical to the peer-reviewed, accepted, and published version available under `https://ieeexplore.ieee.org/document/9981717/` Additional information, animations, videos and other material can be found on the paper website `https://aidx-lab.org/2022-iros-planning/`.

```
@inproceedings{Tenhumberg2022planning,
author = {Johannes Tenhumberg, Darius Burschka, and Berthold Bäuml},
title = {Speeding Up Optimization-based Motion Planning through Deep Learning},
booktitle = {IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)},
year = {2022},
doi = {10.1109/IROS47612.2022.9981717}}
```

Publication [5]
Efficient Learning of Fast Inverse Kinematics with Collision Avoidance

Summary

Fast inverse kinematics (IK) is a central component in robotic motion planning. For complex robots, IK methods are often based on root search and non-linear optimization algorithms. These algorithms can be massively sped up using a neural network to predict a good initial guess, which can then be refined in a few numerical iterations. Besides previous work on learning-based IK, we present a learning approach for the fundamentally harder problem of IK with collision avoidance in diverse and previously unseen environments. From a detailed analysis of the IK learning problem, we derive a network and unsupervised learning architecture that removes the need for a sample data generation step. Using the trained network's prediction as an initial guess for a two-stage Jacobian-based solver allows for fast and accurate computation of the collision-free IK. For the humanoid robot, Agile Justin (19 DoF), the collision-free IK is solved in less than 10 ms (on a single CPU core) and with an accuracy of 10^{-3} m and 10^{-2} rad based on a high-resolution world model generated from the robot's integrated 3D sensor. Our method massively outperforms a random multi-start baseline in a benchmark with the 19 DoF humanoid and challenging 3D environments. It requires ten times less training time than a supervised training method while achieving comparable results.

Contribution

The author of the dissertation contributed 50 % to this paper. He conceptualized the underlying idea and methodology. He wrote the software for the collision-free IK solver and the supervised training. Furthermore, he wrote the first draft of the paper and did the further edites and improvements. In addition he created the visulizations. Arman Mielke worked on the software related to the unsupervised training, and evaluted and analysed the resluts thourougly. His detailed look into the network behaviour improve the work significantly. Berthold Bäuml supervised the work and helped with revieweing and editing the paper.

Version

The attached version of the paper is identical to the peer-reviewed, accepted, and published version available under `https://ieeexplore.ieee.org/document/10375143/` Additional information, animations, videos and other material can be found on the paper website `https://aidx-lab.org/2023-humanoids-ik/`.

```
@inproceedings{Tenhumberg2023ik,
author = {Johannes Tenhumberg, Arman Mielke, and Berthold Bäuml},
title = {Efficient Learning of Fast Inverse Kinematics with Collision Avoidance},
booktitle = {IEEE-RAS International Conference on Humanoid Robots (Humanoids)},
year = {2023},
doi = {10.1109/Humanoids57100.2023.10375143}}
```

Appendix: Full Text of Publications

This appendix provides the full text of all the core publications on which the cumulative dissertation is based and their respective copyright notes, which state information for reprinting the material.

All publications can be found online using the ORCID iD[1] of the author of the dissertation ORCID: 0000-0002-5090-1259.

In the bibliography of the author, the link to the published version is provided for each publication, and, if applicable, a link to a paper website with additional material is also given. Additionally, the contributions of all authors are stated using the roles defined by CRediT[2] in the bibliography of the author. Finally, that section provides a detailed description of the author's contributions to the publications on one page for each publication.

[1]Open Researcher and Contributor ID, `https://orcid.org`

[2]Contributor Roles Taxonomy, `http://credit.niso.org`

Copyright

Calibration of an Elastic Humanoid Upper Body and Efficient Compensation for Motion Planning

Johannes Tenhumberg and Berthold Bäuml

***Abstract*— High absolute accuracy is an essential prerequisite for a humanoid robot to autonomously and robustly perform manipulation tasks while avoiding obstacles. We present for the first time a kinematic model for a humanoid upper body incorporating joint and transversal elasticities. These elasticities lead to significant deformations due to the robot's own weight, and the resulting model is implicitly defined via a torque equilibrium. We successfully calibrate this model for DLR's humanoid Agile Justin, including all Denavit-Hartenberg parameters and elasticities. The calibration is formulated as a combined least-squares problem with priors and based on measurements of the end effector positions of both arms via an external tracking system. The absolute position error is massively reduced from 21 mm to 3.1 mm on average in the whole workspace. Using this complex and implicit kinematic model in motion planning is challenging. We show that for optimization-based path planning, integrating the iterative solution of the implicit model into the optimization loop leads to an elegant and highly efficient solution. For mildly elastic robots like Agile Justin, there is no performance impact, and even for a simulated highly flexible robot with 20 times higher elasticities, the runtime increases by only 30%.**

I. Introduction

A prerequisite for a humanoid robot to autonomously and robustly perform tasks in the real world is an accurate kinematic model of itself. For grasping, the robot needs to precisely position its tool center point (TCP) relative to objects, and when planning collision-free motions in self-acquired 3D models of the environment [1] the extension of the whole body has to be predicted correctly.

Humanoid robots are complex mechatronical systems built from lightweight components, which often leads to a significant deviation from a straightforward, so-called, *geometric* kinematics and make it necessary to model *non-geometric* effects like elasticities. For our advanced humanoid robot DLR Agile Justin [2], e.g., the deviation from the geometric kinematics leads to a mean error of 21 mm and a worst-case error as large as 61 mm in the whole workspace rendering robust autonomous action almost impossible.

In this paper, we present a kinematic model incorporating joint elasticities and transversal elasticities – to our knowledge for the first time for a complete humanoid upper body. We provide a clear and concise derivation of the kinematic model from the torque equilibrium, explicitly state the resulting exact implicit equation, and show how it can be solved iteratively. We successfully calibrate all Denavit–Hartenberg (DH) parameters and elasticities of this model for the humanoid Agile Justin and discuss contributions of the components of the kinematic model to the error reduction. Finally, we present our elegant approach to efficiently use the complex and implicit model in optimization-based motion planning with almost no performance impact.

The authors are with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Münchenerstr. 20, Wessling, 82234, Germany
{johannes.tenhumberg, berthold.baeuml}@dlr.de

This work was partly funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy, within the projects SMiLE (LABAY97) and SMiLE2gether (LABAY102).

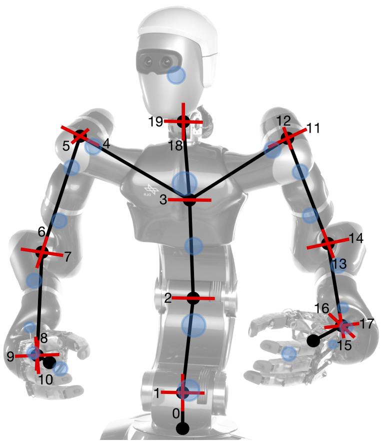


Fig. 1. Kinematic tree structure of the humanoid Agile Justin, showing its 19 degrees of freedom (red) and the mass model used for calibration (blue).

II. Related Work

Roth et al. [3] provide an early overview on robot calibration, where they describe four critical aspects: the calibration model, the dataset, the identification of the model's parameters, and the compensation, i.e., how to use the model in a robotic task. They also describe three different levels of calibration. The first level is only finding the joint offsets. The second is identifying all the robot's geometric parameters. And the third level is the non-geometric calibration, including joint elasticities and gear backlash.

The first two calibration levels have been successfully applied to various robot arms [4, 5, 6]. In all these works, the focus is on algorithmic differences, what sets of DH parameters to use, and the comparison of the speed for calibration, but they ignore the speed of compensation.

Besides robotic arms, also more complex humanoid robots have been calibrated. Maier et al. [7] calibrate the joint offsets for the humanoid robot Nao and Stepanova et al. [8] all the DH parameters for the iCub robot. They also investigate how to combine different internal measurement chains to get a full-body calibration.

Nevertheless, there are cases for which a purely geometric model of the robot is not sufficient. A common source of non-geometric errors is the elasticities in the joints. If the robot is equipped with joint torque sensors, it is convenient to use their measurements in the kinematic model as done in Klodmann et al. [9] for the MIRO robot arm and by Besset et al. [10] for an LWR arm. But as the actual sensor readings are needed, this approach is only feasible for control but not for motion planning. Also, joint torque sensors can not measure any transversal effects.

A different approach for handling elasticities requires identifying a mass model of the robot and computing the torques at their static equilibrium for each pose. Caenen and Angue [11] showed how to incorporate torques into the DH-formalism by adding torque-dependent offsets to the rotational DH parameters. However, they completely neglected to find the torque equilibrium. Others [12, 13, 14] included an iterative search to find the equilibrium, but they only dealt with the joint elasticities and did not include transversal elasticities. Furthermore, they neglected the problem of efficient compensation of such an iterative model.

In the case of our elastic humanoid Agile Justin [2], an efficient compensation of the non-geometric model is crucial as we want to use it in an optimization-based path planner [1]. In previous work, we already automatically calibrated the multi-sensorial head [15, 16] and the IMUs in the head and the base [17]. Instead, in this paper, we provide an accurate calibration and efficient compensation for Agile Justin's full kinematic as needed for whole-body motion planning.

III. Robot Model

A. Geometric Model

The forward kinematics of a robot maps from the configuration space of generalized joint angles to the robots' physical pose in the cartesian workspace. This function is central to robotic path planning. E.g., grasping an object and checking for obstacle or self-collision all strongly depend on an accurate model of the robot's kinematics. A widely used representation of this kinematic model is formulated with the DH parameters. In this formulation, four values $\rho_i = [d_i, r_i, \alpha_i, \theta_i]$ describe the connection between two consecutive frames of the robot:

$$^{i-1}T_i = \mathrm{Rot_x}(\alpha_i) \cdot \mathrm{Trans_x}(r_i) \cdot \mathrm{Rot_z}(\theta_i) \cdot \mathrm{Trans_z}(d_i) \tag{1}$$

The joints q_i are treated as offsets to θ_i or d_i in (2) depending on the type of joint. This minimal representation with two translational and two rotational parameters is enough to describe an arbitrary robot. However, a limit of this formulation shows up for parallel axes (see joints 1 to 3 in Fig. 1). In this case, it is impossible to represent a small variation at the next link with a small variation of the DH parameters. As [11] showed, a solution is to use an additional parameter β representing a rotation around the y-axis, leading to modified DH parameters $\rho_i = [d_i, r_i, \alpha_i, \beta_i, \theta_i]$:

$$\begin{aligned} ^{i-1}T_i = \mathrm{Rot_y}(\beta_i) \cdot \mathrm{Rot_x}(\alpha_i) \cdot \mathrm{Trans_x}(r_i) \cdot \\ \mathrm{Rot_z}(\theta_i) \cdot \mathrm{Trans_z}(d_i) \end{aligned} \tag{2}$$

The frame of the TCP ($i = M$) relative to the robot's base ($i = 0$) is calculated by appling the transformations in series:

$$^0T_M = {}^0T_1 \cdot {}^1T_2 \cdot \ldots \cdot {}^{M-1}T_M \tag{3}$$

For more complex robots with a kinematic tree structure with multiple TCPs M_i, an equation like (3) holds for each branch. We understand the forward kinematics f to map from the robot configuration q not only to the position of the end effector(s) but to all frames of the robot.

$$F = [{}^0T_1, {}^0T_2, \ldots, {}^0T_N] = f(q, \rho) \tag{4}$$

B. Non-geometric Model

However, only considering the geometric model falls short of describing the real robot. This paper focuses on torques as the primary source of non-geometric effects. In general, an acting torque will bend the robot and produce a slightly different position. As the DH parameters can describe an arbitrary robot, it is convenient to integrate the non-geometric effects into this formalism. Caenen and Angue [11] showed this idea by explicitly expressing the influence of torques onto the DH parameters. The simplest possibility is to add a torque-dependent linear to the geometric DH parameters ρ_0:

$$\rho = \rho(\rho_0, \kappa, \tau) = \rho_0 + \kappa\,\tau \tag{5}$$

Note, that the effects of forces on the link lengths are neglected and the matrix $\kappa = [0, 0, \kappa^\alpha, \kappa^\beta, \kappa^\theta]$ describes only compliance around the respective axes, corresponding to the rotational DH parameters $[\alpha, \beta, \theta]$ and the acting torques $\tau = [\tau^x, \tau^y, \tau^z]$. As a consequence, also the forward kinematics now depends via the DH parameters ρ on the torques τ and the elasticity parameters κ:

$$F = f(q, \rho(\rho_0, \kappa, \tau)) \tag{6}$$

In the regime of no external forces and reasonably slow, quasi-static motions, the torques originate only from the robot's weight and its specific distribution. For a frame, the pair $\nu_j = [m_j, w_j]$ describes the mass m_j and its position w_j relative to this frame (i.e., ${}^0w_j = {}^0T_j w_j$). The torques produced by this mass due to the gravity vector g around the respective coordinate axes of another frame i with origin ${}^0p_i$ can be calculated [11] by[1]:

$$\begin{aligned} \tau_{ij}^x &= (({}^0T_j\, w_j - {}^0p_{i-1}) \times m_j g) \cdot {}^0e_{i-1}^x \\ \tau_{ij}^y &= (({}^0T_j\, w_j - {}^0p_{i-1}) \times m_j g) \cdot {}^0e_{i-1}^y \\ \tau_{ij}^z &= (({}^0T_j\, w_j - {}^0p_i) \times m_j g) \cdot {}^0e_i^z \end{aligned} \tag{7}$$

To calculate the full torque τ_i acting on a link i, the contributions from all masses that act on the respective link have to be summed up, i.e., the torques from all masses which are higher up in the robot's kinematic tree.

[1]Those torques act on the rotational axes parameterized by α_i, β_i, and θ_i, which are described in (2). Because they make up a frame by sequential stacking, they do not all share the same frame of reference.

The torques $\tau = \tau(F, \nu)$ now depend on the weight distribution $\nu = [\nu_1, \nu_2, \ldots]$ of the robot and therefore on the frames F which are depending on the DH parameters ρ:

$$\rho = \rho(\rho_0, \kappa, \tau(F(q, \rho), \nu)). \tag{8}$$

This implicit equation for the DH parameters defines the equilibrium between the torques due to gravity and the torques due to flexion of the robot. We define the solution to this equation as the non-geometric DH parameters

$$\rho^* = \rho^*(q, \rho_0, \kappa, \nu). \tag{9}$$

One possibility to solve (8) is by the following iteration:

$$\rho_n = (1-\lambda)\rho_{n-1} + \lambda\rho(\rho_0, \kappa, \tau(F(q, \rho_{n-1}), \nu)) \tag{10}$$
$$\rho_\infty = \rho^* \tag{11}$$

Choosing an appropriate $\lambda \in [0,1]$ for the weighted sum ensures the convergence of the iteration even for very soft (or strongly non-linear) robots[2]. A suitable choice for the start of the iteration are the geometric DH parameters ρ_0 which can be interpreted as a robot in zero gravity or a robot with infinite stiffness. Finally, the non-geometric forward kinematics is then given by

$$F^* = f^*(q, \rho_0, \kappa, \nu) := f(q, \rho^*(q, \rho_0, \kappa, \nu)). \tag{12}$$

IV. Calibration

A. Measurement Model

We use an external camera system from Vicon, consisting of six 16Mpx cameras mounted on the ceiling and designed to track retro-reflective markers with high accuracy. We fixated two such markers on the robot's hands, the last link of the respective kinematic chains (see Fig. 2). To use the markers' positions y for calibrating the forward kinematics f, additional information is necessary. First, we need the robot's base relative to the camera system's world frame ${}^{\mathrm{c}}T_0$. Second, the markers' position on the left and right end effector p_r and p_l must be known. It is not always possible to determine those frames beforehand; therefore, they become part of the calibration problem. Our measurement function h consists of the forward kinematics and the additional frames at the ends of the kinematic chain to close the measurement loop:

$$y = h(q, \Theta) = {}^{\mathrm{c}}T_0 \cdot f^*(q, \rho_0, \kappa, \nu)_{\mathrm{r,l}} \cdot [p_\mathrm{r}, p_\mathrm{l}] \tag{13}$$

The calibration parameters $\Theta = [\rho_0, \kappa, \nu, c]$ are a combination of the DH parameters ρ, the compliances κ, the masses ν, and the parameters $c = [p_\mathrm{c}, o_\mathrm{c}, p_\mathrm{r}, p_\mathrm{l}]$ for the camera base-frame and marker positions to map the forward kinematics to the measurements. The camera frame ${}^{\mathrm{c}}T_0$ is here defined by its position p_c and its orientation o_c.

[2]The convergence can be shown by interpreting the DH parameters as generalized coordinates and the update rule as the discretized integration over time of an damped dynamical system (the robot moving due to gravity).

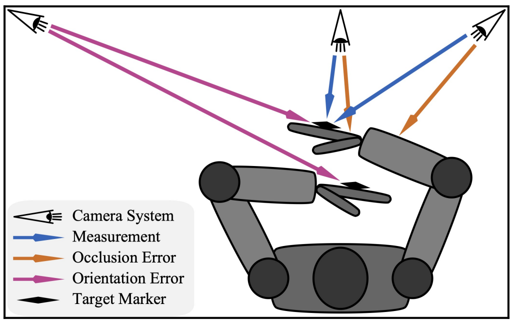


Fig. 2. The measurement setup with cameras tracking markers on the end effectors; showcasing the problem of occlusion and suitable orientation of the marker for a given robot configuration.

B. Identification

The parameters Θ of the measurement function h can be identified using a dataset $D = \{(q^{(n)}, y^{(n)})\}_{n=1\ldots N}$ of pairs of the robot's configuration $q^{(n)}$ and the corresponding marker positions $y^{(n)}$. We formulate the identification as a single combined least-squares problem based on all measurements and all markers to minimize the task space error. To solve for the optimal parameters Θ^*, we use the maximum a posteriori (MAP) approach:

$$\min_{\Theta} \left[\sum_n^N \frac{1}{\sigma_\mathrm{m}^2} |y^{(n)} - h(q^{(n)}, \Theta)|^2 + (\Theta - \Theta_\mathrm{p})^T \Lambda_\mathrm{p}^{-1} (\Theta - \Theta_\mathrm{p}) \right]$$

This approach uses a prior Gaussian distribution with mean Θ_p and a diagonal covariance matrix $\Lambda_\mathrm{p} = \operatorname{diag} \sigma_\mathrm{p}^2$, where the vector σ_p describes the uncertainty of the different calibration parameters. For the measurement noise, we use the usual assumption of a Gaussian distribution with zero mean and standard deviation σ_m. The prior acts as regularization and guarantees the existence of a minimum, even in the presence of redundancies in the measurement/kinematic model. Setting the prior is not critical, but it should be set conservatively to plausible values. Because of the highly nonlinear measurement function, it might be necessary to use multistart for the initial guess to find the global minimum.

C. Configuration Selection and Efficient Sample Collection

We selected the measurement poses randomly by sampling from the configuration space to ensure good overall accuracy and not only close to some standard configurations. Nevertheless, the poses must be feasible for our experimental setup. Besides avoiding self-collision while measuring, the markers imposed additional constraints, as they must be well visible to the cameras. We used rejection sampling to ensure that at least four of the six cameras had a clear view of one marker. We checked for occlusion with simple ray tracing, a straight line from each camera to the marker's position. A sphere model of the robot, which we also use for collision checking, was used to test if the robot blocks any ray. Combining all those constraints, only $1/10000$ configurations were feasible.

After determining a set of feasible configurations in simulation, we now need to measure those poses. The robot has

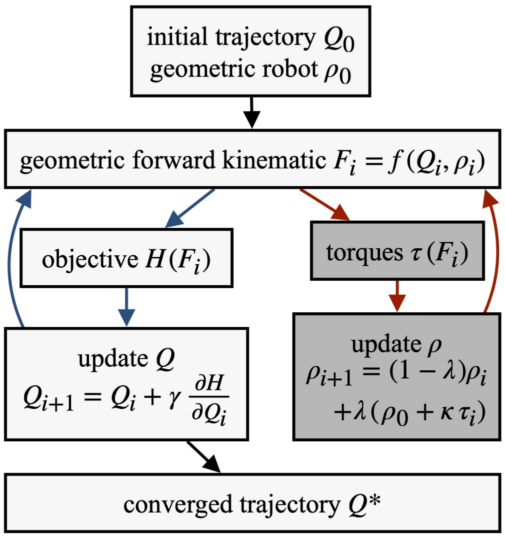

initial trajectory Q_0
geometric robot ρ_0
geometric forward kinematic $F_i = f(Q_i, \rho_i)$
objective $H(F_i)$
torques $\tau(F_i)$
update Q
$Q_{i+1} = Q_i + \gamma \frac{\partial H}{\partial Q_i}$
update ρ
$\rho_{i+1} = (1-\lambda)\rho_i + \lambda(\rho_0 + \kappa\tau_i)$
converged trajectory $Q*$

Fig. 3. Flowchart of the optimization loop in light gray. Dark gray shows the additional loop for the static torque equilibrium. As the outer loop converges, no additional passes through the inner torque loop are necessary.

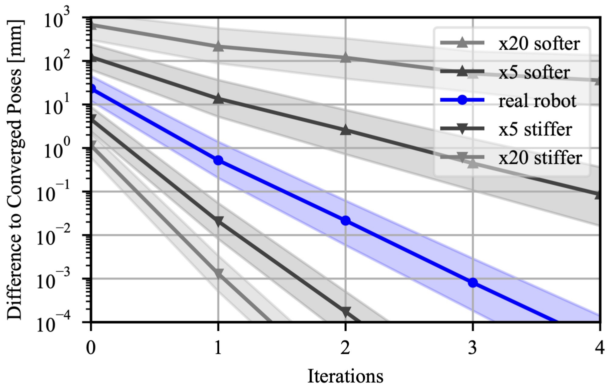


Fig. 4. Simulated convergence towards the static torque equilibrium for robots of different stiffness. For iteration 0 the elastic effect is ignored. The bands indicate the standard deviation for 1000 different joint configurations.

to move to each configuration so that the camera system can collect the corresponding marker positions. One crucial concern regarding experiments with robots is always the time involved. To perform short and collision-free paths from pose to pose, we use an optimization-based path planner; the same planner we want to make more accurate through calibration, but with extended safety margins for the nominal kinematic.

To reduce the time further, we ordered the randomly selected poses to minimize the distance between them. By solving a traveling salesman problem on batches of 100 samples, we could reduce the time by a factor of two. Now the average time to collect one sample is ten seconds. From this, nine seconds fall on the robot performing the trajectory, and one second is added as a pause for the measurement.

V. Compensation

The calibration goal is to find a set of parameters Θ which describes the robot as accurately as possible. Nevertheless, also the speed and ease of use of the calibration model, e.g. in motion planning, is crucial. Incorporating the new set of DH parameters ρ is straightforward and works without any changes or additional costs as they replace the old DH parameters. The same holds for masses m and compliances κ. However, to determine the elastic effects, one must find the static equilibrium between acting torques and elasticities described in (8). While it might be feasible for calibration (offline procedure) to use the iterative algorithm (10), it is more prohibitive for compensation (online). One should carefully evaluate this trade-off between accuracy and simplicity when choosing a calibration model for a robot.

Knowing that we will use the forward kinematics mostly in the framework of an optimization-based path planner has further implications. Such a planner works on paths in configuration space $Q = [q_1, q_2, ..., q_n]$ and performs iterations to get from an initial path Q_0 to a converged path Q^*. For each step, the optimizer considers the objective function $H(Q_i)$ and updates the path using the gradient information. The nested structure of the forward kinematics

$$f^*(q) = f(q, \rho(f(q, \rho(f(...))))) \tag{14}$$

leads to highly non-linear gradients, making the model more difficult to use. As a solution to this, we separate the torque equilibrium search in a separate loop, as shown in Fig. 3. After determining the acting torques accurately and updating the DH parameters accordingly, we assume these as constant for the gradient calculation $\partial F/\partial q$. This allows us to use the pure geometric forward kinematics, implicitly assuming $\partial\rho/\partial q = 0$. Although we completely omit the torque iterations for the gradients, this approximation does not hinder convergence in our tests.

The second important aspect visualized in Fig. 3 is the combination of the optimization loop and the torque equilibrium loop. Even if the updates of the configurations are large at the beginning of the optimization, when the planner converges, the pose updates get small. If those updates are significantly smaller than the offset produced by a torque update, it is unnecessary to search for a new static equilibrium iteratively. In other words, it is sufficient to reuse the already computed frames, and calculate the acting torques and update the DH parameters only once, while nevertheless solving the forward kinematics f^* exactly. The outer loop of the converging optimizer makes it unnecessary to perform inner iterations when searching the torque equilibrium.

VI. Experimental Evaluation

A. Calibration

We collected $N_{\text{all}} = 500$ samples with the procedure described in Section IV-C and split them into a calibration set with $N = 300$ and a test set with $N_{\text{test}} = 200$ samples. This data, as well as an overview of the nominal and the calibrated parameters, are provided online[3].

In what follows, we calibrate the parameters $\Theta = [\rho_0, \kappa, c]$, i.e., not ν (the mass m and the center of mass w) because

[3]https://dlr-alr.github.io/dlr-elastic-calibration

TABLE I

CALIBRATION RESULTS: RESIDUAL TCP ERRORS [MILLIMETERS].

	Frames	**DH parameter**		**Compliance**	
	c	$..+\theta$	$..+d,\ r,\ \alpha,\ (\beta)$	$..+\kappa^{\theta}$	$..+\kappa^{\alpha},\ (\kappa^{\beta})$
μ	21.33	18.18	10.5	5.64	3.12
σ	9.71	7.8	5.74	2.57	1.71
max	63.41	45.9	36.37	12.83	8.23

before calibration | full calibration

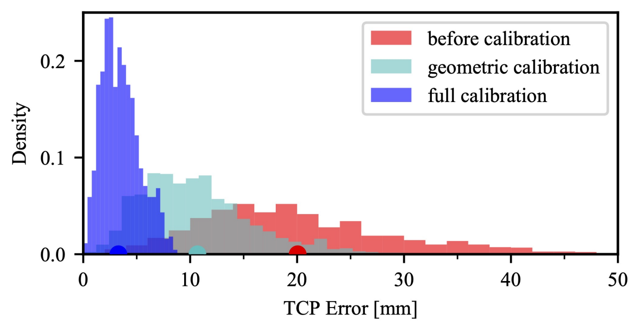


Fig. 5. Histogram of the residual error at the end effectors before and after calibration, highlighting the need for non-geometric modeling of Justin. The circles on the x-axis mark the mean μ of the error (see also Table I).

they could be adopted from the CAD-files of the robot. In a test, we found that adding ν to the calibration parameters did not improve the residual error but significantly increased the runtime of the optimization algorithm. The priors used for the following calibrations were chosen based on previous experiments with the robot, with uncertainty σ_{p} in lengths of 0.1 m, angles of 0.2 rad, and elasticities of 0.1 rad/kNm.

In all experiments, we used multi start for the optimization with initial guesses Θ_0 randomly sampled from the prior distribution. All optimization runs converged to the same optimum, showing the stability of our calibration approach.

As the first step, the additional frames parametrized by c must be determined to close the measurement loop. Only calibrating those frames gives the accuracy of the nominal kinematics before calibration, with a mean residual error of 20 mm - averaged over the 200 test poses and both arms. In the worst cases, the error was larger than 60 mm. After calibration, the mean residual error of the full model is reduced to 3.1 mm and the maximum error to 8.2 mm. Those two cases are reported in Table I in the left- and rightmost columns and a detailed distribution of the errors is shown in Fig. 5. Here the result of a pure geometric calibration without elasticities is also shown, which highlights the need for a non-geometric calibration model for this robot.

We conducted further experiments to evaluate the significance of the different model parameters. Table I reports the residual error when making the model more expressive by adding more parameters step by step, starting with the uncalibrated model (leftmost column) up to the full model (rightmost column). The order in which we added the different types of parameters (always for all joints at once) followed "standard practice" with joint offsets as first and transversal elasticities as the last addition.

TABLE II

"LEAVE-ONE-OUT ANALYSIS": RESIDUAL TCP ERRORS [MILLIMETERS].

	full	$-\kappa^{\theta}$	$-\alpha$	$-\theta$	$-\kappa^{\alpha}$	$-r$	$-d$
μ	3.12	9.09	6.48	6.35	4.85	4.33	3.46
σ	1.71	4.23	2.93	3.16	2.28	2.03	1.73
max	8.23	24.59	17.25	18.89	13.37	11.09	8.97

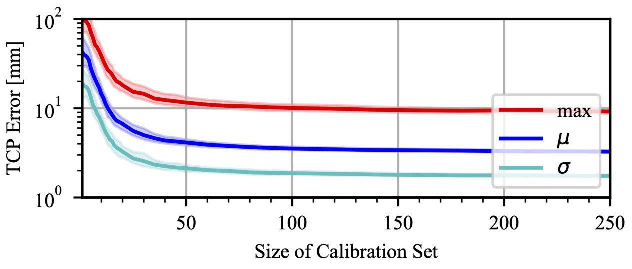


Fig. 6. Test error over the calibration set size N. The bands show the standard deviation over 1000 different calibration sets of a given size.

That this "standard ordering" does not represent the actual influence of the parameters on the residual error can be seen in Table II. Here, starting from the full model (rightmost column), only a specific parameter type was left out one at a time. From this, it can be seen how crucial joint and transversal elasticities are for modeling Justin's kinematics. One reason for the joint elasticities being that prominent is that we not only model the mechanical elasticity of a joint but also the elasticity of the joint position controller [4].

Fig. 6 shows how the test error decreases when more samples N are used for calibrating the full model. Furthermore, one can see the influence of which robot configurations (joint angles) are in a set. The light bands show the standard deviation when selecting 1000 different sets of a given size. While there is still some minor improvement beyond $N = 100$ samples, the main effect of the calibration happens with as little as $N = 50$ samples (those could be collected in less than ten minutes). This information is especially beneficial for recalibrating the robot with a subset of the parameters in the future. A smaller calibration model corresponds to a smaller set necessary for calibration, making the effect of selecting an optimal set of configurations [15] more significant.

B. Compensation

Fig. 4 visualizes the non-geometric effects in more detail by showing the convergence towards a static equilibrium for solving the kinematics via iteration according to (10). We show results for the real and virtual versions of Agile Justin with lower and higher elasticities. For the real robot, the difference after just one iteration is already considerably below our calibration accuracy. But if the robot is softer or the accuracy requirements are stricter, one needs more iterations to converge to the equilibrium configuration.

In addition, we tested the algorithm described in Fig. 3 on these versions of Agile Justin. While for the real or stiffer

[4]We use a relatively simple joint position controller as it is robust (e.g., it does not rely on the torque sensors, which are notoriously drifting and hard to maintain). However, it results in an additional joint-level elasticity.

versions no additional iterations were necessary to converge to the grasping poses, the procedure also works for soft robots with stiffnesses in the range of $\sim 100\,\mathrm{N/rad}$. In this case, the number of iterations increased by 30%. In all cases, the wall clock time for one loop of the optimizer increases by only one percent. Because no additional calculation of the forward kinematics is necessary, and one pass through the torque loop is enough.

VII. Conclusions

In this paper – to our knowledge for the first time – a calibration model including all DH parameters, joint and transversal elasticities was formulated and successfully calibrated for a humanoid robot.

For DLR's Agile Justin, the average error could be significantly reduced from prohibitively large 21 mm to 3.1 mm in the whole workspace. We provided a clear and concise derivation of the implicit kinematic model with elasticities from a torque equilibrium. We showed that this complex implicit model can be used in optimization-based motion planning without any performance impact (for mildly elastic robots like Agile Justin). And even for a simulated soft robot with 20 times higher elasticities, the slowdown was only 30%. We achieved this by tightly integrating the iterative solver for the implicit kinematic model into the optimization planner loop of the planner. Finally, we provided an in-depth discussion of the influence of the individual components of the model on the residual position error.

In the future, we want to make the calibration completely automatic and self-contained by replacing the use of an external tracking system by using the robot's head-mounted cameras, so combining this work with our previous work on automatic camera calibration [16].

References

[1] R. Wagner, U. Frese, and B. Bauml, "3D modeling, distance and gradient computation for motion planning: A direct GPGPU approach," in *2013 IEEE International Conference on Robotics and Automation*, no. Iii. IEEE, 5 2013, pp. 3586–3592.

[2] B. Bauml *et al.*, "Agile Justin: An upgraded member of DLR's family of lightweight and torque controlled humanoids," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2014, pp. 2562–2563.

[3] Z. Roth, B. Mooring, and B. Ravani, "An overview of robot calibration," *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 377–385, 10 1987.

[4] U. S. Pathre and M. R. Driels, "Simulation experiments in parameter identification for robot calibration," *The International Journal of Advanced Manufacturing Technology*, vol. 5, no. 1, pp. 13–33, 2 1990.

[5] L. S. Ginani and J. M. S. T. Motta, "Theoretical and practical aspects of robot calibration with experimental verification," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 33, no. 1, pp. 15–21, 3 2011.

[6] I. W. Park and J. H. Kim, "Estimating entire geometric parameter errors of manipulator arm using laser module and stationary camera," *IECON Proceedings (Industrial Electronics Conference)*, pp. 129–134, 2011.

[7] D. Maier, S. Wrobel, and M. Bennewitz, "Whole-body self-calibration via graph-optimization and automatic configuration selection," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2015-June, no. June. IEEE, 5 2015, pp. 5662–5668.

[8] K. Stepanova, T. Pajdla, and M. Hoffmann, "Robot Self-Calibration Using Multiple Kinematic Chains-A Simulation Study on the iCub Humanoid Robot," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1900–1907, 2019.

[9] J. Klodmann, R. Konietschke, A. Albu-Schaffer, and G. Hirzinger, "Static calibration of the DLR medical robot MIRO, a flexible lightweight robot with integrated torque sensors," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 9 2011, pp. 3708–3715.

[10] P. Besset, A. Olabi, and O. Gibaru, "Advanced calibration applied to a collaborative robot," in *2016 IEEE International Power Electronics and Motion Control Conference (PEMC)*. IEEE, 9 2016, pp. 662–667.

[11] J. Caenen and J. Angue, "Identification of geometric and nongeometric parameters of robots," in *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, 1990, pp. 1032–1037.

[12] W. Khalil and S. Besnard, "Geometric calibration of robots with flexible joints and links," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 34, no. 4, pp. 357–379, 2002.

[13] B.-J. Lee, "Geometrical Derivation of Differential Kinematics to Calibrate Model Parameters of Flexible Manipulator," *International Journal of Advanced Robotic Systems*, vol. 10, no. 2, p. 106, 2 2013.

[14] J. Zhou, H.-N. Nguyen, and H.-J. Kang, "Simultaneous identification of joint compliance and kinematic parameters of industrial robots," *International Journal of Precision Engineering and Manufacturing*, vol. 15, no. 11, pp. 2257–2264, 11 2014.

[15] H. Carrillo *et al.*, "On task-oriented criteria for configurations selection in robot calibration," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 5 2013, pp. 3653–3659.

[16] O. Birbach, U. Frese, and B. Bäuml, "Rapid calibration of a multi-sensorial humanoid's upper body: An automatic and self-contained approach," *International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 420–436, 2015.

[17] O. Birbach and B. Bauml, "Calibrating a pair of inertial sensors at opposite ends of an imperfect kinematic chain," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, no. Iros. IEEE, 9 2014, pp. 422–428.

Copyright

Self-Contained Calibration of an Elastic Humanoid Upper Body Using Only a Head-Mounted RGB Camera

Conference Proceedings:
2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)

Author: Johannes Tenhumberg

Publisher: IEEE

Date: 28 November 2022

Copyright © 2022, IEEE

IEEE Requesting permission to reuse content from an IEEE publication

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Self-Contained Calibration of an Elastic Humanoid Upper Body Using Only a Head-Mounted RGB Camera

Johannes Tenhumberg[1,2] Dominik Winkelbauer[1] Darius Burschka[3] Berthold Bäuml[1,2]

***Abstract*— When a humanoid robot performs a manipulation task, it first makes a model of the world using its visual sensors and then plans the motion of its body in this model. For this, precise calibration of the camera parameters and the kinematic tree is needed. Besides the accuracy of the calibrated model, the calibration process should be fast and self-contained, i.e., no external measurement equipment should be used. Therefore, we extend our prior work on calibrating the elastic upper body of DLR's Agile Justin by now using only its internal head-mounted RGB camera. We use simple visual markers at the ends of the kinematic chain and one in front of the robot, mounted on a pole, to get measurements for the whole kinematic tree. To ensure that the task-relevant cartesian error at the end-effectors is minimized, we introduce virtual noise to fit our imperfect robot model so that the pixel error has a higher weight if the marker is further away from the camera. This correction reduces the cartesian error by more than 20 %, resulting in a final accuracy of 3.9 mm on average and 9.1 mm in the worst case. This way, we achieve the same precision as in our previous work [1], where an external cartesian tracking system was used.**

I. Introduction

When the humanoid robot Agile Justin performs a task, it first uses its internal camera to make a model of the world, including the poses of objects. It then plans how to move its body in this world model to reach an object without obstacle collison or self-collisions. The success of this look-and-move approach depends highly on the calibration of its cameras and the whole kinematic tree. In the case of Agile Justin, the deviation from the nominal geometric kinematics is as large as 61 mm. This significant error makes it necessary to add safety margins for the collisions, and robust and precise manipulation is almost impossible.

In our previous paper [1], we derived a model with elasticities for the humanoid and showed how to use it efficiently inside an optimization-based planner. The calibration of the model was based on the cartesian measurements of an external tracking system where tracking targets were mounted on the two end effectors. Using an external tracking system poses two main problems. First, calibration is only possible when in the lab. Second, the internal camera is not incorporated in the calibration, although it is used when performing manipulation tasks, limiting the accuracy.

In this paper, we show that the elastic model of a humanoid robot can be calibrated using only its head-mounted RGB camera and simple markers on the two end effectors and one in front of the robot. The main contributions are:

- We perform a self-contained robot calibration using only the head-mounted 640×480 RGB camera, i.e., without any external measurement equipment.
- The complete kinematic tree, including the torso, left and right arms, the neck, and the camera, are calibrated. The model has 129 free parameters, including the DH parameters, joint and lateral elasticities, and the extrinsic and intrinsic parameters of the camera.
- We show that directly minimizing the error between measured and reprojected pixel coordinates of the markers results in non-optimal cartesian precision (which is relevant for performing tasks) when dealing with imperfect models, as in the case of our complex humanoid robot. Therefore, we introduce a virtual noise term to compensate for the mapping between the image and cartesian space. This correction reduces the cartesian error at the end effectors by 20 %.
- We validate the calibration results on the real robot. For this evaluation, we use an external tracking system. The final cartesian error at the end effectors is 3.9 mm on

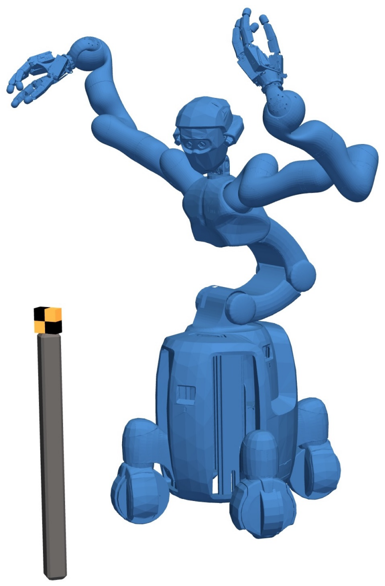


Fig. 1. DLRs's Agile Justin [2] collecting measurements using its head-mounted RGB camera for the calibration of its elastic forward kinematics as well as the camera's intrinsic and extrinsic parameters. Only simple markers on both hands as well as the depicted marker mounted on a pole are used. As described in Section V, we select filtered random configurations to identify the robot in its whole work space.

[1]DLR Institute of Robotics & Mechatronics, Germany; [2]Deggendorf Institute of Technology, Germany; [3]Technical University of Munich, Germany
Contact: johannes.tenhumberg@dlr.de

This work was partly funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy, within the projects SMiLE (LABAY97) and SMiLE2gether (LABAY102).

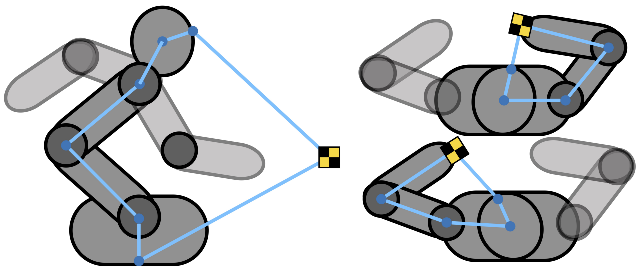


Fig. 2. Sketch of the calibration setup. The robot collects images of markers on both of its hands and a pole in front of it. The blue chains show how forward kinematics plus camera projection close the measurement loop. Even if the arms are not directly involved in the pole measurements, their mass distribution in different positions influences the torso elasticities.

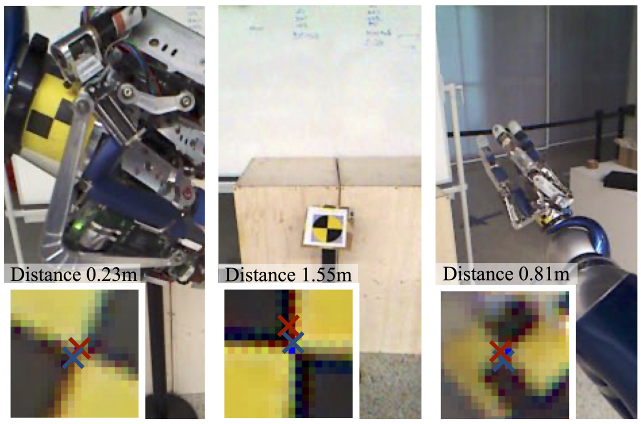


Fig. 3. DLR's Agile Justin collects measurements to calibrate its non-geometric forward kinematics. The images are from the robot's internal RGB camera with a resolution of 640×480, showing examples for the left arm, the right arm, and the pole. The markers' distances to the camera vary between measurements from 0.2 m up to 1.5m. Without a correction (red), the pixel error is uniformly distributed over the distances, leading to more significant cartesian errors for detections further away from the camera as they correspond to a larger area. The correction (blue) counteracts this and improves the cartesian accuracy by 20 %.

average and 9.2 mm in the worst case.

- The procedure of collecting measurements with the robot's internal RGB camera and performing the calibration takes under 30 minutes. Required for the speed is a method to select the poses accounting for a clear view of the markers while allowing for a wide variety of joint configurations.
- The dataset, as well as a Python package to calibrate a general elastic robot, are provided[1]. The tool allows a combination of non-geometric forward kinematics with different custom measurement functions.

II. Related Work

An accurate forward kinematics is relevant for most robotic applications; therefore, there are a lot of examples of successful calibrations. Most of the time, the robotic arm's geometric model is calibrated with an external tracking system [3, 4, 5, 6]. For an overview of the calibration and compensation of elastic robots, we refer to the related work in the preceding paper [1].

However, the calibration model must not only be expressive enough to match the real robot well. Another important aspect is a fast and easy calibration process to make it broadly applicable and easy to repeat if necessary. Ideally, the robot uses its internal sensors to calibrate itself. This choice also ensures that precisely the same chain is calibrated the robot uses to perform its task. Sang De Ma [7] introduced a self-calibration technique for active vision systems. Hubert et al. [8] added a bayesian approach and performed hand-eye calibration of an anthropomorphic robot using a checkerboard marker.

Maier et al. [9] calibrated the joint offsets for the humanoid robot Nao by following four checkerboard markers on both of its hands and feet with its RGB camera. Finally, Stepanova et al. [10] used a combination of visual and tactile self-observing to calibrate all the DH parameters for the iCub robot. However, they did this only in simulation.

While using a single camera for calibration is convenient, minimizing the error in image space is not the same as minimizing the relevant error in the cartesian task space. Reprojection terms have been used for calibration with RGB-D cameras [11, 12]. But they did not introduce virtual noise to account for an imperfect robot model and did not use it to transform the error in the image space to the task-relevant cartesian space. They only assumed a real noise for the joints, which can often be measured quite accurately and does not need to be handled as unknown noise.

We have already tackled the problem calibration for the humanoid Agile Justin [2]. Because this complex mechatronic system is built from lightweight components, it is especially susceptible to torque-dependent elasticity effects. Furthermore, its autonomous motion has strict accuracy requirements for its sensors and its forward kinematics. In previous work, the multi-sensorial head [13], the IMUs in the head and base [14], and the eye-hand chain [15] were calibrated. While using the internal sensors, they ignored the torso for calibration, even as this chain has a significant error.

In the preceding paper, we described the non-geometrical model of the humanoid Agile Justin and showed how to calibrate it with an external camera system [1]. Furthermore, we introduced an efficient technique to compensate for the implicit model, which was crucial as we wanted to use it in an optimization-based path planner [16].

The main drawback of this previous work was the dependence on the external tracking system. This dependence not only limits the general applicability of the approach. Using the external system, we do not calibrate the relevant chain between the robot camera and body, which is pertinent to perform tasks autonomously. Instead, this consecutive paper provides a fast and accurate auto-calibration for Agile Justin's entire kinematic chain. Relying only on the measurements collected by its internal RGB camera, we calibrate exactly the chain needed for whole-body motion planning.

[1]You can find the dataset, videos of the measurements, additional details to the methods, and the code to calibrate an arbitrary elastic robot at https://dlr-alr.github.io/2022-humanoids-calibration.

III. Robot Model

A. Forward Kinematics

We use the same elastic forward kinematics as in [1] to model the robot. To describe how the robots physical pose F in the cartesian workspace changes with joint angles q in the configuration space we us the DH formalism with the geometric parameter ρ. The forward kinematics f maps not only to the position of the end effector(s) but to all frames of the robot.

$$F = [{}^0T_1, {}^0T_2, \dots, {}^0T_N] = f(q, \rho) \tag{1}$$

Following Caenen and Angue [17] we integrate the non-geometric effects from elasticities κ by explicitly expressing the influence of torques τ onto the DH parameters.

$$\rho = \rho(\rho_0, \kappa, \tau) = \rho_0 + \kappa\, \tau \tag{2}$$

The non-geometric forward kinematics is then given by

$$F = f(q, \Theta_\mathrm{f}) = f(q, \rho^*(q, \underbrace{\rho_0, \kappa, \nu}_{\Theta_\mathrm{f}})).$$

where ρ^* describes the solution of the non-geometric DH-parameters in torque equilibrium, resulting from the robots mass distribution ν in configuration q. For more details on the derivation of those equations, as well as an algorithm to compensate and use this implicit model efficiently see Tenhumberg and Bäuml [1].

B. Camera Model

The forward kinematics describes the cartesian position of the different body parts. If one wants to use a camera to measure those positions, one needs a model to project from the cartesian into the image space $U\colon \mathbb{R}^3 \to \mathbb{R}^2$. Here we use the classical pinhole model with radial distortion to project a 3D point of the marker x into 2D pixel coordinates u [15]. First, the detected point x must be transformed into the camera frame ${}^0T_\mathrm{C}$. After this, the 3D point is projected along the z-axis of the camera frame with $P(x)$ and radial distortion $D(u, \xi_\mathrm{C})$ is added. The pixel coordinates of the image u are then calculated as an offset from the cameras center point c_C scaled with the focal length f_C:

$$U({}^0x_\mathrm{M}, {}^0T_\mathrm{C}, \Theta_\mathrm{C}) = c_\mathrm{C} + f_\mathrm{C} \cdot D(P({}^\mathrm{C}T_0\ {}^0x_\mathrm{M}), \xi_\mathrm{C}), \tag{3}$$

$$\text{with } P(x) = \left(\frac{x_\mathrm{x}}{x_\mathrm{z}}, \frac{x_\mathrm{y}}{x_\mathrm{z}}\right)^\mathrm{T};\ D(u, \xi_\mathrm{C}) = \frac{u}{1 + \xi_\mathrm{C}|u|^2} \tag{4}$$

Besides the position of marker ${}^0x_\mathrm{M} = {}^0T_\mathrm{i}{}^\mathrm{i}x_\mathrm{M}$ and frame of the camera ${}^0T_\mathrm{C} = {}^0T_\mathrm{j}{}^\mathrm{j}T_\mathrm{C}$ relative to the forward kinematics F, the intrinsic parameters of the camera are also part of the calibration. Those additional parameters are combined and denoted as $\Theta_\mathrm{C} = [{}^ix_\mathrm{M}, {}^jT_\mathrm{C}, c_\mathrm{C}, f_\mathrm{C}, \xi_\mathrm{C}]$.

IV. Calibration Problem

The goal of calibration is to find the set of parameters $\Theta = [\Theta_\mathrm{f}, \Theta_\mathrm{C}]$ which best fit the kinematic model and the camera model defined in the last section.

A. Maximum a Posteriori Estimation

As usual, we formulate the calibration as a probabilistic estimation problem [18]. In Fig. 4 on the left, the probabilistic model is depicted using the functions introduced in Section III to connect the input joint angles q with the pixel coordinates of the markers u depending on the parameters Θ. The goal is to find the maximum of the posterior distribution $p(\Theta|S)$ given the measurements $S = \{(u^{(n)}, q^{(n)})\}_N$. For simplicity of notation, in what follows, we do not discern between the three different markers but assume that S includes all measurements.

As Fig. 4 shows, the stochastic variable of a markers pixel coordinates u can be expressed as a function (the so-called measurement function h) of the two input nodes q and Θ

$$h(q, \Theta) = U(f(q, \Theta_\mathrm{f})_i{}^ix_\mathrm{M}\,,\ f(q, \Theta_\mathrm{f})_j{}^jT_\mathrm{C}\,,\ \Theta_\mathrm{C})$$
$$u = h(q, \Theta) + \eta_u, \quad \eta_u \sim N(0, C_u), \quad C_u = \sigma_u^2 I.$$

When the data is collected with the camera, there is real measurement noise on the pixels. We model this as gaussian noise η_u with zero mean diagonal variance σ_u. The maximum a posteriori (MAP) problem assuming a diagonal Gaussian prior $p(\Theta)$ then results in a non-linear least squares problem

$$\min_\Theta \sum_n \log p(u^{(n)}|q^{(n)}, \Theta) + \log p(\Theta) =$$
$$= \min_\Theta \sum_n (\Delta u^{(n)})^\mathrm{T} C_\mathrm{m}^{-1} \Delta u^{(n)} + \Delta\Theta^\mathrm{T} C_\mathrm{p}^{-1} \Delta\Theta, \tag{5}$$
$$\text{with } \Delta u^{(n)} = u^{(n)} - h(q^{(n)}, \Theta), \quad C_\mathrm{m} = C_u, \tag{6}$$
$$\text{and } \Delta\Theta = \Theta - \Theta_\mathrm{p}, \quad C_\mathrm{p} = \operatorname{diag} \sigma_\mathrm{p}^2.$$

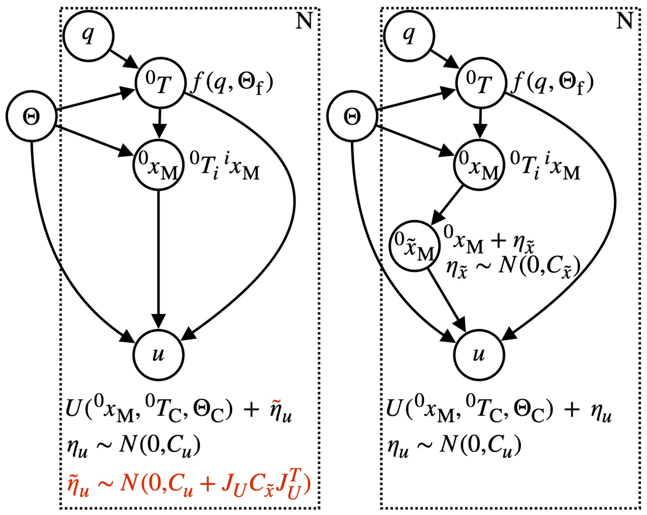
In Section VI we report the results of solving this optimization problem.

B. Virtual Cartesian Noise

The MAP approach in (5) minimizes the difference between the measured marker and the reprojection of the marker in pixel coordinates. This method usually gives reasonable estimates for the parameters Θ if the final pixel error gets very small, i.e., when the measurement model can fit the real robot well. But when fitting an imperfect model, i.e., a model which can not wholly reproduce all aspects of the real robot, the vanilla MAP approach still would minimize the overall pixel error by equally distributing the error between all measurements in (5). Intuitively, this is not what we expect from a good fit: we want a fit that minimizes the error in cartesian space. A camera measures angles. This means a pixel further away corresponds to a larger area in physical space than a pixel closer to the camera. Therefore a pixel error for a marker far from the camera should count more than a pixel error close to the camera.

To achieve this in a methodological sound way, we introduce an additional node in the graphical probabilistic model (Fig. 4, right graph). This addition explicitly models the (actually deterministic) imperfection as additional *virtual noise* in the marker's 3D position.

$${}^0\tilde{x}_\mathrm{M} = {}^0x_\mathrm{M} + \eta_{\tilde{x}}, \quad \eta_{\tilde{x}} = N(0, C_{\tilde{x}}), \quad C_{\tilde{x}} = \sigma_{\tilde{x}}^2 I.$$



N
q
Θ
${}^0T$
$f(q,\Theta_f)$
${}^0x_M$
${}^0T_i\,{}^ix_M$
u
$U({}^0x_M, {}^0T_C, \Theta_C) + \tilde{\eta}_u$
$\eta_u \sim N(0,C_u)$
$\tilde{\eta}_u \sim N(0,C_u + J_U C_{\tilde{x}} J_U^T)$
N
q
Θ
${}^0T$
$f(q,\Theta_f)$
${}^0x_M$
${}^0T_i\,{}^ix_M$
${}^0\tilde{x}_M$
${}^0x_M + \eta_{\tilde{x}}$
$\eta_{\tilde{x}} \sim N(0,C_{\tilde{x}})$
u
$U({}^0x_M, {}^0T_C, \Theta_C) + \eta_u$
$\eta_u \sim N(0,C_u)$

Fig. 4. The probabilistic graph of the calibration problem includes the camera and robot model from Section III. It describes how the markers pixel coordinates u are computed from the joint configuration q and the model parameters Θ for each of the N samples. *Left (w/o red parts)*: In the original mapping, the real pixel measurement noise η_u is the only source of stochasticity. *Right*: An additional virtual cartesian noise node is added to compensate for the imperfect (actually deterministic) kinematic model. *Left (with red parts)*: As shown in Section IV-B, the virtual noise can be incorporated into the original model, resulting in an effective pixel noise with a $\tilde{\sigma}_u$ depending on the distance of the marker to the camera ($\propto 1/z^2$).

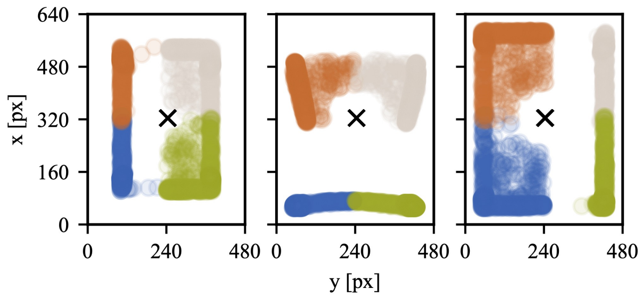


Fig. 5. The different marker positions in the image for the left arm, the pole on the floor and the right arm. We move the pan-tilt joints of the robot's neck to get a good coverage of the image over all markers.

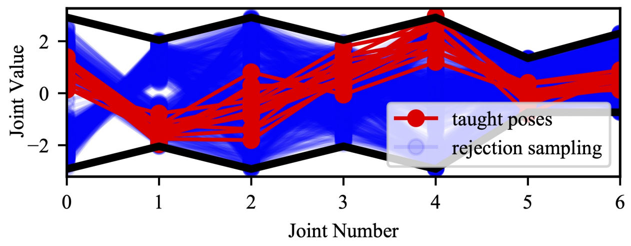


Fig. 6. The different distributions in configuration space of the right arm with 7 joints. In red are the taught poses [19, 13, 15]. In blue are the configurations resulting from the rejection sampling approach described in Section V leading to a broader distribution. The joint limits are black.

It is important to note that the noise is added in base coordinates as we want the model error to be distributed equally in the world (and not, e.g., relative to some moving frame of the robot). The pixel coordinates of this noisy marker position now depend on this additional noise term

$$u = U({}^0x_M + \eta_{\tilde{x}}, {}^0T_C, \Theta_C) + \eta_u.$$

By marginalizing over the new variable ${}^0\tilde{x}_M$, we get the effect of the additional virtual noise on the distribution of the pixel coordinates explicitly

$$p(u|{}^0x_M, {}^0T_C, \Theta) = \int p(u|{}^0\tilde{x}_M, {}^0T_C, \Theta)p({}^0\tilde{x}_M|{}^0x_M)\mathrm{d}{}^0\tilde{x}_M.$$

Due to non-linearities in U, the resulting distribution is non-Gaussian. However, we can approximate it with a Gaussian distribution by linearizing U for the noise and using Gaussian arithmetic. This results in almost the same form as before except for a new effective covariance $\tilde{C}_u = \tilde{C}_u({}^0x_M, {}^0T_C, \Theta)$ which now also depends on the marker's coordinates.

$$u \approx U({}^0x_M, {}^0T_C, \Theta_C) + \tilde{\eta}_u, \quad \tilde{\eta}_u \sim N(0, \tilde{C}_u),$$
$$\tilde{C}_u({}^0x_M, {}^0T_C, \Theta) = C_u + J_U C_{\tilde{x}} J_U^T,$$
$$J_U({}^0x_M, {}^0T_C, \Theta) = \frac{\partial U(x, {}^0T_C, \Theta_C))}{\partial x}\bigg|_{x={}^0x_M}$$

Assuming that the camera distortion can be neglected for calculating the effective noise distribution, we finally get

$$\tilde{C}_u \approx \sigma_u^2 I + \sigma_{\tilde{x}}^2 \left(\frac{f_C}{x_z}\right)^2 \begin{pmatrix} 1 + \frac{x_x^2}{x_z^2} & \frac{x_x x_y}{x_z^2} \\ \frac{x_x x_y}{x_z^2} & 1 + \frac{x_y^2}{x_z^2} \end{pmatrix}, \tag{7}$$

where $x = {}^Cx_M = {}^0T_C^{-1}({}^0x_M)$.

The resulting MAP problem looks exactly the same as the original one (5), except that the weighting of the individual measurement errors is changed to $C_m = \tilde{C}_u({}^0x_M, {}^0T_C, \Theta)$. Looking at (7), this result means that markers further away are more critical and scaled with z^2 – just as we intuitively expected.

V. Efficient Sample Collection

One goal of selecting measurement poses is to cover the whole configuration space. Sampling uniformly in the configuration space ensures that the calibrated model works well even if the robot moves autonomously and uses its full range of motion far away from taught standard configurations.

Nevertheless, the configurations must be feasible for the measurement setup. When the robot uses its camera to collect measurements of the markers, it imposes strict constraints. The markers must be in the camera's field of view, must not be occluded by the robot's own body, and must face toward the camera. We check for occlusion with simple ray tracing and a sphere model of the robot by drawing a straight line from the camera to the marker and ensuring it does not collide with any of the spheres. Checking if the camera and the marker face each other can be done by simply calculating the scalar product between their relative position and viewing direction. Furthermore, we want to ensure that only a single marker is visible to the robot at any configuration. In the case of Agile Justin, we needed over 10 million configurations to find 100 feasible measurement configurations for a marker.

As all those calculations happen before the calibration,

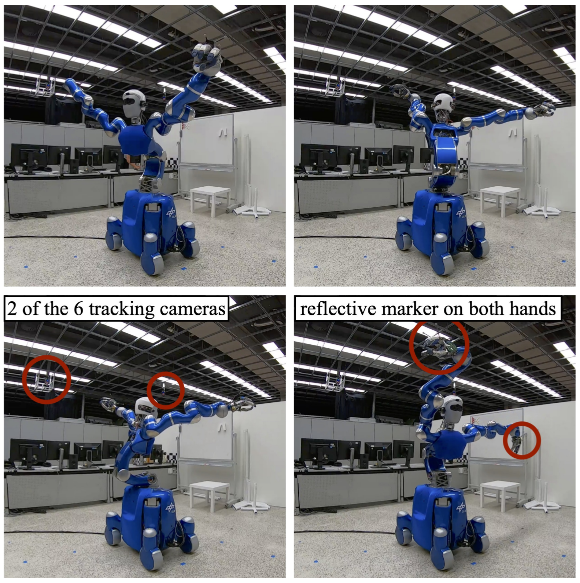


Fig. 7. We also collect measurements using the Vicon tracking system described in [1] to evaluate our approach in the cartesian space. This external tracking system consists of six cameras mounted on the ceiling and directly tracks the cartesian position of retro-reflective markers with high accuracy.

one must account for uncertainties with larger thresholds and safety margins. Fig. 6 shows that this general rejection sampling approach (blue) is better suited to get an even distribution over the configuration space. In contrast, in red are the taught poses used in prior works. This comparison shows that even for experts, it is hard to choose unbiased configurations for a complex humanoid.

We collect multiple measurements per robot configuration to speed up the calibration procedure. In the case of Agile Justin, the camera is mounted on a pan-tilt joint, allowing us to adjust the marker's position in the image easily. Assuming that in the initial configuration, the marker is roughly in the center of the camera's field of view, we move the camera to collect four additional measurements where the tag is in one of the corners of the image each time.

Fig. 5 shows the marker positions in the image when adjusting the head to collect multiple measurements per configuration. The main reason for only moving the head while keeping the rest of the body fixed is to increase the number of samples quickly. Furthermore, good image coverage makes calibrating the camera intrinsics easier. To ensure that, we used neck joints to move the projection of the marker toward the corners while keeping a safety margin to the image center and borders.

As in Tenhumberg and Bäuml [1], we solve a traveling salesman problem to order the configurations we want to measure and reduce the time for calibration. Furthermore, we use an optimization-based path planner to perform short and collision-free paths between the measurement configurations.

VI. Experimental Evaluation

With this approach, we collected measurements for 50 configurations per marker, with five head positions each,

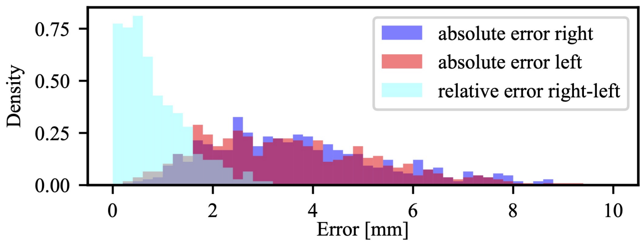


Fig. 8. Absolute cartesian error of the left and right arm after the full calibration. The relative error between those two is significantly smaller, indicating that the main part of the remaining error comes from the torso.

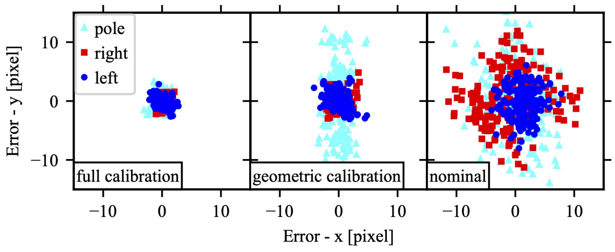


Fig. 9. Distribution of the errors in image space for the three markers at the pole, the left, and the right wrist. Calibrating only the geometric parameters (center) does not explain the elasticities in the torso seen in the pole marker. The entire calibration (left) with geometric and non-geometric parameters distributes the remaining errors uniformly over the three markers.

resulting in $50 \times 5 \times 3 = 750$ samples for the three markers. We split the set (equally for each marker) into 500 samples for calibration and 250 samples for evaluation. The virtual noise is set to $\sigma_{\tilde{x}} = 1\,\text{cm}$ and the pixel noise to $\sigma_u = 0.2$ (sub-pixel detection accuracy). It takes roughly 8 minutes to collect the measurements for the markers on the hands and an additional 13 minutes to make the measurements for the pole. The latter takes more time as the joint configurations are further apart in this setting, as the whole body is involved and not only one arm. Together with performing the calibration itself (3 minutes), the whole procedure takes 32 minutes. For comparison, the procedure with the external tracking system takes 25 minutes, which is a little faster as no separate configurations for each marker are necessary. However, there is an additional overhead for the setup of the tracking system.

For comparison, we also used the external cartesian tracking system to perform a new calibration using the method from Tenhumberg and Bäuml [1]. For this, we recorded recorded 100 different configurations (see Fig. 7). We also use this cartesian tracking data to evaluate the calibration based only on the head-mounted camera. For this, the position of the reflective targets and the tracking systems frame relative to the robot are recalibrated.

Fig. 9 shows the error in image space for the markers on the left and right arm and the pole for different calibration models. On the right-most image is the nominal forward kinematics, and in the center is the geometric model with the DH parameters. The left image shows the full calibration, including joint and lateral elasticities and the camera intrinsics. One can see that the torso chain is responsible for significant non-geometric errors due to the large acting

TABLE I

ERROR IN THE IMAGE AND THE CARTESIAN SPACE FOR DIFFERENT CALIBRATION MODELS WITH AND W/O VIRTUAL NOISE (VN), WITH AND W/O INTRINSIC CAMERA PARAMETERS (Θ_C) AND USING ONLY THE CAMERA (IMAGE) OR TRACKING SYSTEM (POINTS).

Calibrate			Image Error [px]			Cartesian Error [mm]		
on	Θ_C	VN	μ	σ	max	μ	σ	max
Images	no	no	1.05	0.59	3.76	4.77	2.29	11.75
Images	yes	no	0.97	0.53	3.44	4.65	2.27	11.58
Images	no	yes	1.21	0.70	4.13	4.11	1.87	9.34
Images	yes	yes	1.15	0.62	3.97	3.94	1.83	9.16
Points	-	-	-	-	-	3.12	1.71	8.23

torques and its mechanical design with ropes. In Fig. 8 we further analyzed the influence of the different body parts. The absolute cartesian errors of the right and left arm are red and blue, respectively. Here the torso chain is part of the measurements; therefore, its error is included. However, the torso is excluded if we look at the distance between the left and right target and compare it against the measured length. This reduced error in the relative arm positions indicates that the remaining error mainly comes from the torso chain.

The results also emphasize the need for our virtual noise term to cope with the imperfect model of the robot. Even the elastic robot model does not capture all the relevant effects, and a significant error remains. That the virtual noise helps to distribute the error evenly in the task-relevant cartesian space can be seen in Table I. While the pixel error increases slightly, the mean and maximal cartesian error gets smaller by over 20 % when correcting for the mapping between the image and task space. Furthermore, we show that it is possible to include the camera intrinsics Θ_C in the calibration, further improving the accuracy. The mean final error at the end effectors is 3.9 mm on average and 9.2 mm in the worst case. These results are comparable to a calibration using the cartesian measurements of an external tracking system, which is reported in the last row.

VII. CONCLUSION

The main advantage over the previous work from Tenhumberg and Bäuml [1] is that the new approach does not rely on an external camera system and calibrates the same chain used when performing manipulation tasks. However, to achieve comparable accuracy in the cartesian task space, it is not enough to minimize the pixel error in the image of the single RGB camera. We correct this mapping by introducing virtual cartesian noise. This way, it is ensured that the remaining error of our imperfect model is minimized in the task-relevant cartesian space and not the pixel space. We show that this simple, self-contained approach leads to a similar good precision as using an external tracking system, reducing the error to 3.9 mm on average.

In the future, we want to reduce the number of samples needed by optimizing the used kinematic configurations – similarly to the work of Carrillo et al. [13], but for an elastic robot model and based on the here presented generic configuration generation scheme. We also want to improve the kinematic model by introducing an additional non-linear term for the torso chain, as we found here that this sub-chain is the primary source of the remaining error.

REFERENCES

[1] J. Tenhumberg and B. Bäuml, "Calibration of an Elastic Humanoid Upper Body and Efficient Compensation for Motion Planning," in *IEEE-RAS International Conference on Humanoid Robots*, vol. 2021-July, 2021.

[2] B. Bäuml *et al.*, "Agile Justin: An upgraded member of DLR's family of lightweight and torque controlled humanoids," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, 5 2014, pp. 2562–2563.

[3] L. S. Ginani and J. M. S. T. Motta, "Theoretical and practical aspects of robot calibration with experimental verification," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 33, no. 1, pp. 15–21, 3 2011.

[4] I. W. Park and J. H. Kim, "Estimating entire geometric parameter errors of manipulator arm using laser module and stationary camera," *IECON Proceedings (Industrial Electronics Conference)*, pp. 129–134, 2011.

[5] G. Xiong, Y. Ding, L. M. Zhu, and C. Y. Su, "A product-of-exponential-based robot calibration method with optimal measurement configurations," *International Journal of Advanced Robotic Systems*, vol. 14, no. 6, pp. 1–12, 2017.

[6] K. Van Wyk, J. Falco, and G. Cheok, "Efficiently Improving and Quantifying Robot Accuracy In Situ," *arXiv*, 8 2019.

[7] Sang De Ma, "A self-calibration technique for active vision systems," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 114–120, 1996.

[8] U. Hubert, J. Stuckler, and S. Behnke, "Bayesian calibration of the hand-eye kinematics of an anthropomorphic robot," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 11 2012, pp. 618–624.

[9] D. Maier, S. Wrobel, and M. Bennewitz, "Whole-body self-calibration via graph-optimization and automatic configuration selection," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2015-June, no. June. IEEE, 5 2015, pp. 5662–5668.

[10] K. Stepanova, T. Pajdla, and M. Hoffmann, "Robot Self-Calibration Using Multiple Kinematic Chains-A Simulation Study on the iCub Humanoid Robot," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1900–1907, 4 2019.

[11] V. Pradeep, K. Konolige, and E. Berger, "Calibrating a Multi-arm Multi-sensor Robot: A Bundle Adjustment Approach," in *Springer Tracts in Advanced Robotics*, 2014, vol. 79, pp. 211–225.

[12] M. Ferguson and N. Arora, "Robust and Efficient Calibration of Mobile Manipulators," Fetch Robotics, Tech. Rep., 2015.

[13] H. Carrillo *et al.*, "On task-oriented criteria for configurations selection in robot calibration," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, 5 2013, pp. 3653–3659.

[14] O. Birbach and B. Bauml, "Calibrating a pair of inertial sensors at opposite ends of an imperfect kinematic chain," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, no. Iros. IEEE, 9 2014, pp. 422–428.

[15] O. Birbach, U. Frese, and B. Bäuml, "Rapid calibration of a multi-sensorial humanoid's upper body: An automatic and self-contained approach," *International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 420–436, 2015.

[16] R. Wagner, U. Frese, and B. Bauml, "3D modeling, distance and gradient computation for motion planning: A direct GPGPU approach," in *2013 IEEE International Conference on Robotics and Automation*, no. Iii. IEEE, 5 2013, pp. 3586–3592.

[17] J. Caenen and J. Angue, "Identification of geometric and nongeometric parameters of robots," in *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, 1990, pp. 1032–1037.

[18] C. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[19] O. Birbach *et al.*, "Automatic and self-contained calibration of a multi-sensorial humanoid's upper body," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 5 2012, pp. 3103–3108.

Copyright

Self-Contained and Automatic Calibration of a Multi-Fingered Hand Using Only Pairwise Contact Measurements

Conference Proceedings:
2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)

Author: Johannes Tenhumberg; Leon Sievers; Berthold Bäuml

Publisher: IEEE

Date: 12-14 Dec. 2023

Copyright © 2023, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Self-Contained and Automatic Calibration of a Multi-Fingered Hand Using Only Pairwise Contact Measurements

Johannes Tenhumberg*[1,2,3]   Leon Sievers*[1,2,3]   Berthold Bäuml[1,2]

***Abstract*— A self-contained calibration procedure that can be performed automatically without additional external sensors or tools is a significant advantage, especially for complex robotic systems. Here, we show that the kinematics of a multi-fingered robotic hand can be precisely calibrated only by moving the tips of the fingers pairwise into contact. The only prerequisite for this is sensitive contact detection, e.g., by torque-sensing in the joints (as in our DLR-Hand II) or tactile skin. The measurement function for a given joint configuration is the distance between the modeled fingertip geometries, but the actual measurement is always zero. In an in-depth analysis, we prove that this contact-based calibration determines all quantities needed for manipulating objects with the hand, i.e., the difference vectors of the fingertips, and that it is as sensitive as a calibration using an external visual tracking system and markers. We describe the complete calibration scheme, including the selection of optimal sample joint configurations and search motions for the contacts despite the initial kinematic uncertainties. In a real-world calibration experiment for the torque-controlled four-fingered DLR-Hand II, the maximal error of 17.7 mm can be reduced to only 3.7 mm.**
Web: https://dlr-alr.github.io/2023-humanoids-contact/

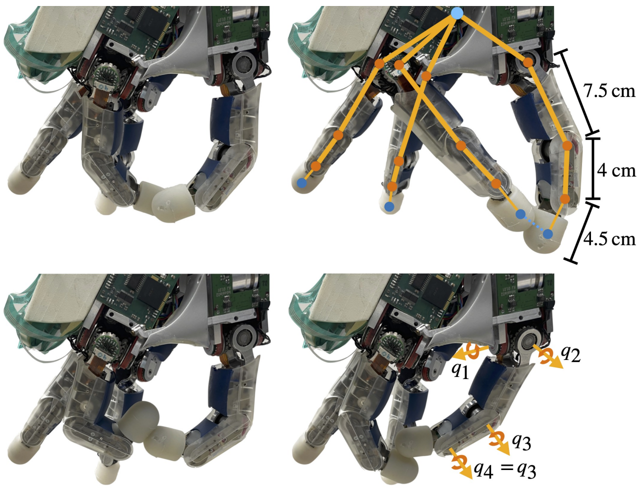


Fig. 1. DLR-Hand II [6] with thumb and index finger in contact for different poses. The kinematic tree of the whole hand is indicated in orange, and the three active joints plus the fourth passive joint are drawn together with the dimensions of the fingers. The other two fingers are far extended to allow for a large shared workspace of the current pair. For the whole calibration, all six finger pairs are measured and calibrated jointly.

I. Introduction

Autonomous robots that robustly perform dextrous manipulation tasks in the real world generally require precise models of the system's kinematics. Regarding multi-fingered robotic hands, one crucial class is planning algorithms for finding an optimal grasp for a given 3D model of an object [1, 2]. This task depends on the precise placement of the fingers on the object's surface. Another class is methods for dextrous in-hand manipulation. Here, only recent modern deep reinforcement learning algorithms trained in simulation have enabled dexterity close to human performance. Primarily when performed in a purely tactile setting [3, 4, 5], i.e., without cameras, where only joint angles (and tactile measurements, e.g., via torque-sensing) are used, robust zero-shot sim2real transfer requires a precise kinematics model with a maximal error of a few millimeters.

It is desirable to have a quick calibration procedure that runs entirely automatically and is self-contained, i.e., it does not need any additional external sensors or tools to obtain a precise kinematic model.

A. Related Work

A classical and often-used approach for calibrating kinematic chains and trees is visual tracking of the end-effector(s). For a robotic hand, this visual approach with an external tracking system was demonstrated by Lee et al. [7]. However, adding visual markers to the many end-effectors is pretty cumbersome, especially as more than one marker is usually required per fingertip due to the mutual occlusions of the many fingers in the small workspace of a hand. Using an electromagnetic tracking system to measure the position and orientation of the fingertips [8] is less prone to occlusions. However, the requirement for additional hardware hinders the ease of use of such an approach.

Another calibration procedure that avoids the need for a tracking system uses geometric constraints on the kinematic chain. An early work constrained the motion of an end-effector on a plane to calibrate a robotic arm [9]. Since then, there have been examples of using mechanical fixtures [10], relative calibration techniques [11], and precise reference plates [12] to calibrate a robotic arm without a vision system. Bennett and Hollerbach [13] applied these ideas to a robotic hand and calibrated the multi-finger Utah-MIT hand by rigidly connecting two fingertips with a plate, resulting in a closed-loop kinematic chain. A downside of all those approaches is that the mounting procedure can damage the fingers and takes time, especially if it needs to be repeated for each finger pair.

A particular case of the relative geometric calibration

* First two authors contributed equally.
[1]DLR Institute of Robotics & Mechatronics, Germany
[2]Deggendorf Institute of Technology, Germany
[3]Technical University of Munich, Germany
Contact: johannes.tenhumberg@dlr.de, leon.sievers@dlr.de

techniques does not require mechanical fixtures to enforce the constraints but relies on self-contact. The humanoid robot iCub was intensely used for such studies. First in simulation [14] and then on the actual hardware together with other sensing modes [15]. Roncone et al. [16] calibrated the full DH parameters of the iCub robot using self-touch with a tactile skin with 4200 sensing points. Besides identifying the parameters of a given kinematic chain, self-contact can also be used to incrementally update the kinematic scheme [17] or identify the layout of an artificial tactile skin [18].

B. Contributions

In prior work, an external camera system was first used to calibrate an elastic kinematic model of the humanoid Agile Justin [19]. Then, this approach became more accessible by using the robot's internal RGB camera to calibrate the complete kinematic tree [20]. This paper extends the calibration to the DLR-Hand II [6, 21] (see Fig. 1) of the robot and introduces a contact-based approach that does not rely on anything other than the kinematic tree structure itself. The main contributions of this work are:

- We show, to our knowledge for the first time, that the kinematics of a multi-fingered hand can be calibrated using only contact measurements of (all) finger pairs, i.e., without any external tools like cameras or markers.
- An in-depth theoretical analysis of the calibration problem with redundant parameters, including proof that the quantities relevant for manipulation tasks, i.e., the difference vectors between the fingertips, are entirely determined by our contact measurement scheme.
- A comparison of calibration by contact measurements with calibration using a visual tracking system (theoretical analysis and simulation experiments).
- A complete scheme for executing a contact-based calibration for a general multi-fingered hand, including optimal selection of finger configurations and planning of search motions to deal with the initial uncertainties in the kinematics.
- Real-world experiments with the contact calibration of the DLR-Hand II resulting in a reduction of the maximal error from 17.7 mm for the nominal kinematic to only 3.7 mm for the full DH parameters calibration.

II. Robot Model

The forward kinematics of a robot maps from the configuration space of generalized joint angles to the robots' physical pose in the cartesian workspace. Both for robotic arms and hands, an accurate model of the robot's kinematics is essential for precise and collision-free motions. A well-fitted model is also required to perform challenging grasping and manipulation tasks.

DH parameters are widely used to describe this kinematic model of the robot. In this formulation, four values $\rho_i = [d_i, r_i, \alpha_i, \theta_i]$ describe the connection between two consecutive frames of the robot:

$$^{i-1}T_i = \mathrm{Rot}_\mathrm{x}(\alpha_i) \cdot \mathrm{Trans}_\mathrm{x}(r_i) \cdot \mathrm{Rot}_\mathrm{z}(\theta_i) \cdot \mathrm{Trans}_\mathrm{z}(d_i) \quad (1)$$

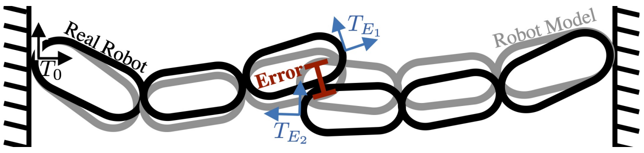


Fig. 2. The scheme shows the actual contact measurement on the hardware (black) and the same joint configuration applied to the robot model (gray). The fingers are in penetration for the current set of calibration parameters. This error (red) should be minimized over the calibration process.

The joints q_i are treated as offsets to θ_i or d_i in (1) depending on the type of joint. This minimal representation with two translational and two rotational parameters is enough to describe an arbitrary robot.

The frame of the end-effector $(i = E)$ relative to the robot's base $(i = 0)$ is calculated by applying the transformations in series:

$$^0T_E = {}^0T_1 \cdot {}^1T_2 \cdot \ldots \cdot {}^{E-1}T_E \quad (2)$$

For robots with a kinematic tree structure with multiple end-effectors $E_k, k = 1 \ldots N_\mathrm{E}$, equation (2) holds for each branch. The forward kinematics maps from joint configurations $q \in \mathcal{Q}$ to all the N_F frames of the robot

$$f(q, \rho) = F = [{}^0T_1, {}^0T_2, \ldots, {}^0T_{N_\mathrm{F}}]. \quad (3)$$

Each frame F_i describes a full 6D pose $(F_{i,x}, F_{i,r}) \in \mathbb{R}^3 \times SO(3)$.

III. Calibration for In-hand Manipulation

A. Identification

The goal of the calibration is to identify this kinematic model. In our case, the central parameters of the forward model are the DH parameters ρ. Nevertheless, in general, there can be additional parameters that have to be estimated jointly with the kinematic parameters. Examples are camera intrinsics or additional frames to close the measurement loop. We denote the combination of all calibration parameters Θ. These parameters of the measurement function h can be identified using a dataset $D = \{(q^{(n)}, y^{(n)})\}_{n=1\ldots N}$ of corresponding pairs between the robot's configuration $q^{(n)}$ and the measurement $y^{(n)}$. We formulate the identification as a single combined least-squares problem based on all measurements and all body parts to minimize the error $\epsilon = y - h(q, \Theta)$ between measurements and model predictions. To find the optimal parameters Θ^*, we use the maximum a posteriori (MAP) approach:

$$\min_\Theta \left[\sum_n^N \frac{1}{\sigma_\mathrm{m}^2} \left\| y^{(n)} - h(q^{(n)}, \Theta) \right\|^2 + \Delta\Theta^T \Lambda_\mathrm{p}^{-1} \Delta\Theta \right]$$
$$\text{with } \Delta\Theta = \Theta - \Theta_\mathrm{p}$$

This approach employs a Gaussian distribution beforehand, characterized by a mean Θ_p and a diagonal covariance matrix $\Lambda_\mathrm{p} = \mathrm{diag}\,\sigma_\mathrm{p}^2$. The uncertainty of the calibration parameters is modeled via σ_p. Furthermore, we assume the usual Gaussian distribution with a mean of zero and a standard deviation of σ_m for the measurement noise. By

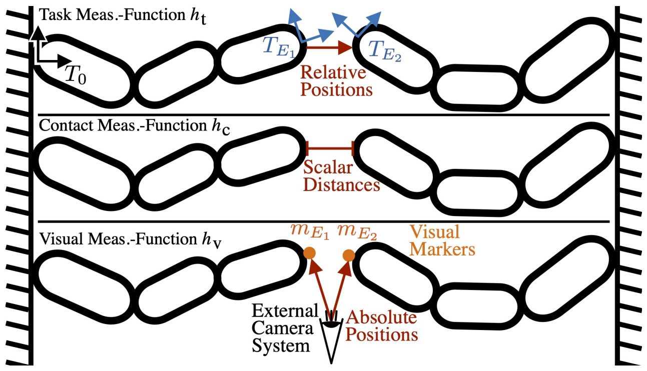


Fig. 3. The scheme shows the three measurement functions discussed in Section III. *Top (Section III-B):* The task measurement function h_t measures the relative positions of the end-effectors. *Middle (Section III-D):* The contact measurement function h_c measures the scalar distance between two end-effectors. *Bottom (Section III-C):* The cartesian measurement function h_v uses an external tracking system to measure the absolute position of the end-effectors.

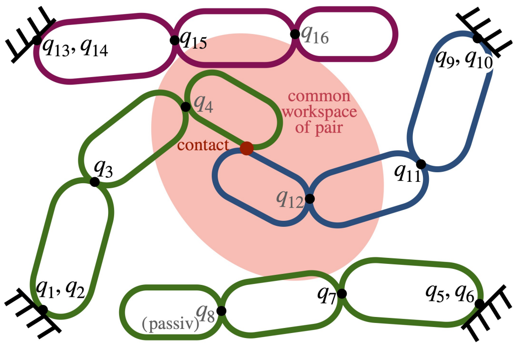


Fig. 4. The graphic shows the contact calibration approach for a four-fingered hand. For pairwise contact, two fingers always need to move out of the shared workspace of the current finger pair to ensure that self-collisions are avoided and the available configuration space for contact detection is well used.

incorporating the prior, this method is regularized, ensuring a minimum exists, even if there are redundancies in the measurement or kinematic model.

B. Task Measurement Function

We aim to calibrate this kinematic model of the hand for dextrous in-hand manipulation. To move the fingers in a controlled manner, as demonstrated by Pitz et al. [4] with the cube, an accurate model for the relative positions of the fingertips to each other is necessary. To relate closely to this task, we chose our desired measurement function to measure the relative positions directly. The calibration process should then minimize the error of this function.

For N_{E} fingers, $N_{\mathrm{E}}-1$ distance vectors define the whole set. One can choose one fingertip E_1 as the basis and compute the positions relative to it for the remaining fingers

$$y^k = h_{\mathrm{t}}^k(q,\rho) = f(q,\rho)_{E_k,x} - f(q,\rho)_{\mathrm{E}_1,x} \tag{4}$$

One can then concatenate the measurements for the pairs to get the combined measurements $h_{\mathrm{t}} = [h_{\mathrm{t}}^2, \dots, h_{\mathrm{t}}^{N_{\mathrm{E}}}], h_{\mathrm{t}}^k \in \mathbb{R}^3$. This results in $3 \cdot (N_{\mathrm{E}} - 1)$ data points per robot pose for the three spatial directions and N_{E} end-effectors.

Note that (4) is our theoretically desired measurement function, which measures all the information we care about for precise in-hand manipulation. For example, we can not detect a translational offset of all end-effectors like a cartesian tracking system could. However, such a shift does not change the fingers' relative behavior and is irrelevant to our task.

Task Test Set: Besides the task measurement function, the test set is central to evaluating the calibration quality. We want a high accuracy across the whole cartesian workspace. Therefore, the test set for evaluating all our experiments should be uniformly distributed in the cartesian workspace. We sample random joint configurations for each finger and map them to the end-effector positions via the forward kinematics f. In the cartesian workspace, we create a fine grid and draw one of the grid cells uniformly, and in the second step, we draw one joint configuration that lies in this grid cell. This sampling strategy ensures a uniform distribution over the workspace, which does not hold for configurations sampled randomly in the configuration space.

After defining a suitable measurement function and test set for our task, we discuss the actual measurement methods that can be applied to the hardware. First, an external camera system that tracks the cartesian position of visual markers followed by our pairwise contact measurements. Note that there are also additional considerations, especially the independence of particular infrastructure and the low complexity of the measurement setup, speaking in favor of contact-based measurements.

C. Cartesian Measurement Function

One option to perform the actual measurements is to mount visual markers on the kinematic tree and use an external tracking system to collect cartesian measurements of these markers. Assuming that the markers are fixed at position m_{E_k} relative to end-effector E_k, the cartesian measurement function

$$y^k = h_{\mathrm{v}}(q,\Theta)^k = {}^{\mathrm{c}}T_0 \cdot f(q,\Theta)_{E_k} \cdot m_{E_k} \tag{5}$$

describes how a marker moves dependent on the joint configuration q and the calibration parameters Θ. Suppose one wants to calibrate the forward kinematics jointly for multiple end-effectors. In that case, one can combine the measurements for each of the N_{E} markers to a combined measurement function $h_{\mathrm{v}} = [h_{\mathrm{v}}^1, \dots, h_{\mathrm{v}}^{N_{\mathrm{E}}}], h_{\mathrm{v}}^k \in \mathbb{R}^3$. While, in general, the measurements with an external camera can be made without constraining the robot directly, one still needs to account for self-collision and a clear view of the markers. These additional constraints reduce the possible space in which measurements can be collected.

D. Contact Measurement Function

Another option to perform the actual measurement is to use contact information. We describe this procedure in more detail in Section V. In general, the corresponding

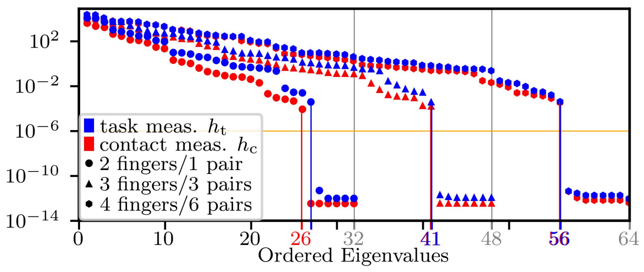


Fig. 5. This figure shows the ordered eigenvalues for different measurement setups to analyze the sensitivity. The evaluation was done with the nominal kinematic of the DLR-Hand II; for a more generic hand, see the right plot. The task measurement function h_t is blue, and our contact measurement function h_c is red. Furthermore, we show three modes. For the one where all the pairs are calibrated simultaneously (⬢), the kernels of both measurement functions have the same size. The same is true for the calibration with three fingers (▲) However, the kernel sizes differ when just a single pair (●) is considered. The light gray vertical lines indicate the maximal number of parameters for each mode.

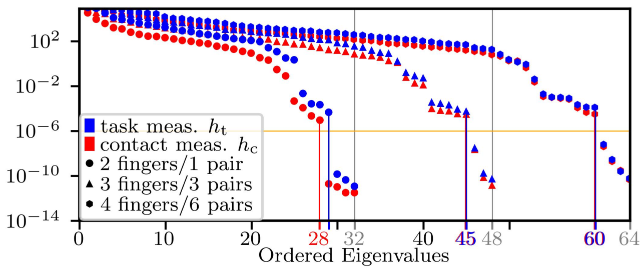


Fig. 6. The plot is structured equivalently to Fig. 5 but for a more generic hand kinematic without parallel axes or mounting frames. Here, in theory, all 64 parameters for the entire hand (⬢) can be identified. However, when only two fingers are considered (●), the contact measurement function h_c is still less sensitive than the task measurement function h_t. This speaks in favor of a holistic calibration of the whole kinematic tree.

measurement function measures the distance between parts of the robot. One needs the exact geometry of the two bodies to compute the distance d^u between a pair of bodies $u = (E_k, E_l)$ on the kinematic tree. The distance is only dependent on the relative position and orientation of the bodies

$$d^u = d^u({}^{E_k}T_{E_l}) = d^u(f(q,\rho)_{E_k}^{-1} \cdot f(q,\rho)_{E_l}) = d^u(q,\rho). \tag{6}$$

This relative frame ${}^{E_k}T_{E_l}$ is directly computable from the forward kinematics $f(q,\rho)$.

If the bodies at E_k and E_l have simple geometric forms, one can directly compute the distance d_m. In the case of the DLR-Hand II, the fingertips are perfect capsules in the contact area and, therefore, straightforward to compute. The more general case is that the geometries are given as arbitrary meshes. In this case, one has to use, for example, algorithms like GJK to compute the distance.

The contact measurement function for the pair u can now be written as

$$y^u = h_c(q,\Theta)^u = d^u(f(q,\rho)). \tag{7}$$

This function can only measure the scalar distance between two body pairs. With N_E end-effectors on the kinematic tree, there are in total $N_U = \binom{N_E}{2}$ pairs. The combined measurement function is in this case $h_c = [h_c^1, \ldots, h_c^{N_U}], h_c^k \in \mathbb{R}$

The contact measurement adds hard constraints to the data collection (see Section V-A). As only configurations in contact can be measured, the available configuration space is drastically reduced. Therefore, the remaining subspace can be quite different from the cartesian task test set described in Section III-B. The following section discusses how those different measurement functions relate and how many parameters they can identify. Furthermore, we discuss the influence of the different calibration and test sets on the sensitivity.

IV. Problem Analysis

A. Sensitivity Analysis

A single contact measurement (7) measures less information than a cartesian tracking system (5). The tracking system can measure the absolute position of the markers in the workspace. The contact can only measure the relative distance between two parts of the kinematic chain. In other words, the contact can measure the joint angles where the distance between the two bodies is zero.

The question is whether this reduced information in the measurement function leads to non-identifiable parameters. We can answer this question directly by looking at the jacobians of the different measurement functions

$$J^s = \frac{\partial h(q^s,\Theta)}{\partial \Theta}\bigg|_{\Theta_0}. \tag{8}$$

Assuming that each measurement is d-dimensional, concatenating those matrices for each measurement leads to the combined jacobian $\mathbf{J} = [J^1, \ldots, J^{N_D}]$ with dimensions $(N_D \cdot d) \times N_\Theta$. We can investigate which parameters are identifiable by the measurement function h by looking at the nullspace of $\mathbf{J}^T\mathbf{J}$. The size of the nullspace marks how many of the model parameters Θ can not be identified. Furthermore, from the eigenvectors corresponding to the eigenvalues close to zero, one can identify the parameters (or sets of parameters) which can not be measured.

When one deals with two distinct measurement functions, as we described with the desired task h_t and our actual contact measurement function h_c, the necessary condition is that

$$\text{kernel}(\mathbf{J}_c^T\mathbf{J}_c) \subseteq \text{kernel}(\mathbf{J}_t^T\mathbf{J}_t). \tag{9}$$

If this condition is satisfied, one can identify all the relevant parameters to the task, defined by h_t. Note that this is less strict than demanding that both kernels are zero and applies in general to distinct measurement functions for calibration and evaluation. We allow for unidentifiable parameters if they do not influence our desired measurement function.

B. Optimal Experimental Design

Following Carrillo et al. [22], we use task D-optimality to select appropriate samples for measuring. However, we have two key differences in our setup. First, we have a theoretical desired task measurement function h_t and an actual measurement function h_c we can apply on the hardware. Ultimately, we want a good fit for the desired measurement function. Second, the contact measurement reduces the configuration space quite drastically. Still, we want a good fit and high accuracy across the whole workspace. Therefore we also have two sets here. The desired test set is uniformly sampled across the whole cartesian test set and one actual test set, which can be measured via contact detection. We generalize the optimality framework to account for those mismatches between measurement functions and distributions. Carrillo et al. [22] showed that the central equation decouples and the task D-optimality can efficiently be computed with

$$O_\mathrm{D} = \frac{1}{l}\det\left((\mathrm{cov}(\Theta)\sum_{s=1}^{N_{\bar{\mathrm{D}}}} J_\mathrm{t}^{s\,T} J_\mathrm{t}^{s}\right). \tag{10}$$

The mean over $J_i^T J_i$ is constant for a given test set of size $N_{\bar{\mathrm{D}}}$ and a desired task measurement function h_t. The covariance over the calibration measurements can be estimated using the actual chosen measurement function and the specific calibration set. Let $S = \{s_i\}_{i=1}^{N_\mathrm{D}}$ be a subset of a larger calibration set and $\mathbf{J}_\mathrm{c} = [J_\mathrm{c}^{s_1}, \dots, J_\mathrm{c}^{s_{N_\mathrm{D}}}]$ the combined jacobians corresponding to those measurements. Then the covariance is given by

$$\mathrm{cov}(\Theta) = \mathbf{J}_\mathrm{c}^T \,\mathrm{diag}(\sigma_\mathrm{m}^2)\mathbf{J}_\mathrm{c} + \mathrm{diag}(\sigma_\mathrm{p}^2). \tag{11}$$

and transforms the uncertainty in the measurements σ_m and the priors σ_p into an uncertainty in the parameters.

(10) lets us compute how well different calibration sets are suited to minimize the error of the desired measurement function over a desired test set. This criterion can be used to choose a good set of suitable poses for measuring. Besides reducing the overall size of the necessary calibration set, this selection criterion also counteracts the mismatch in the measurement functions and the calibration and test sets. We report the results of the sensitivity analysis and the optimal selection criterion in Section VI-B.

V. Contact Measurement Procedure

A. Sample Generation

Our approach is to collect pairwise contact measurements for the tree structure of the hand. A key difference for contact-based calibration is that one does not know the exact joint configuration, which will be measured beforehand. Generally, the measurement process collects pairs (q, y) to calibrate a function $g(q, \theta) = y$. For vision-based measurements, one collects the cartesian position y_i for selected joints q_i. On the other hand, for contact-based measurements, the y_i is known a priori; the contact is, by definition, $y_i = 0$. The contact measurement delivers the exact configuration q_i, which leads to contact.

Therefore, contact measurement must also include a search to find the exact configuration in which the contact happens. We tackle this problem by generating for each measurement point a trajectory along which contact will probably happen.

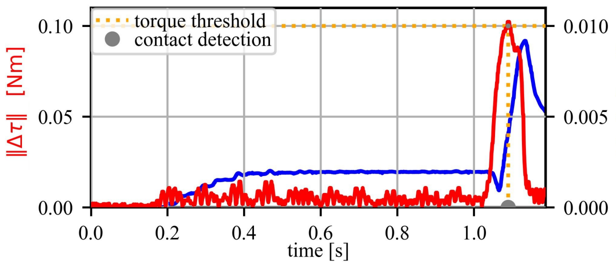


Fig. 7. Red: L^2 norm of $\Delta\tau$. $\Delta\tau$ denotes the difference between the torque signal measured by the passive finger before the approach started, τ_0, and the currently measured torque. Blue: L^2 norm of Δq. Δq denotes the difference between the position signal measured by the passive finger before the approach started and the currently measured position. The contact is detected when the torque threshold $\tau_t = 0.1\,\mathrm{Nm}$ is exceeded. Note how the passive finger's joint angles can change after the active finger starts to drive causing vibrations in the system. The torque sensors only measure noise until the contact occurs.

We define this path by its endpoints. The start configuration is far from contact with the nominal robot model, and the end configuration is deep in penetration. Between those endpoints, we then detect contact for the specific finger pair.

Our approach to generating these search trajectories for the contact measurements consists of multiple steps. The first step is to find the volume in the workspace which both end-effectors can reach. We save a large (n=100000) set of configurations that fall in this intersection for both fingers. The next step is to randomly choose one configuration for finger A and check for which configurations of finger B the tips collide. The threshold for collision one chooses here strongly depends on the robot model's initial uncertainty. Continuing from this pair of configurations, we choose which finger should be static and which should move into contact. For the moving finger, we then sample an additional random configuration as the start point of the measurement drive.

We repeat this procedure for all six finger pairs. For each pair, the rest of the kinematic structure should move as far away as possible from the combined workspace of the current end-effector pair (see Fig. 4). The goal is to obstruct the measurements as little as possible and allow for diverse joint configurations in this constrained setting.

One additional problem is the small form factor of a robotic hand, especially compared to the errors of the uncalibrated system. For a robotic arm with a total reach of one meter, an error of a few centimeters does not change the measurement setup. However, in our case, the DLR-Hand II with its four fingers, the ratio between error in the forward kinematics and the robot's actual size is much larger. We measured uncalibrated errors up to $17.7\,\mathrm{mm}$. That equals roughly 10% of the workspace and is also about the fingertip size. The consequence is that even if, in simulation for the nominal kinematic, the two fingers touch close to the center, it is still possible that the measurement fails on the hardware. Therefore one needs to account for this high uncertainty while generating samples and safe trajectories to collect those samples. Furthermore, robust contact detection is crucial for reliable measurements on the actual hardware.

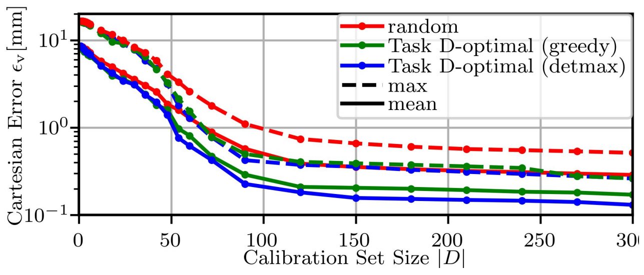


Fig. 8. Convergence of mean and maximal cartesian error ϵ_v on the uniform cartesian test set for different calibration sets for the contact measurement function h_c. Randomly chosen sets (red) converge slower than the sets designed according to task D-optimality (10). The greedy strategy is green, and the detmax[23] strategy is blue. The distribution mismatch between calibration and test set can explain why the gap between random and selected samples remains significant even for larger calibration sets.

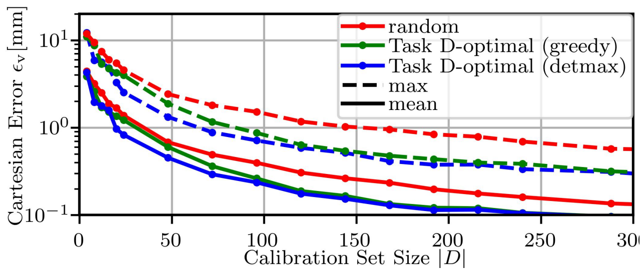


Fig. 9. The plot is designed equivalently to Fig. 8, but for the cartesian measurement function h_v. The overall convergence of the error ϵ_v is quicker because the absolute cartesian position yields more information per sample. Furthermore, the difference between the random approach and the dedicated selection criteria is smaller in this setting. One reason is that the calibration set for the external tracking system is closer to the task test set.

B. Contact Detection

It is essential to detect the precise joint angles in which the contact force between the fingertips is as small as possible. The DLR-Hand II is equipped with an output side torque sensor for each of the twelve active degrees of freedom. Before the passive finger gets approached, the offset, τ_0, is set to the measured torque, $\tau_m(t)$. When the active finger moves and the change in torque is larger than the threshold τ_t, a contact is detected (see Fig. 7).

VI. Experiments

As the model of the forward kinematics for the DLR-Hand II, we include all four DH parameters per joint, resulting in $N_\Theta = N_{\text{DoF}} \times 4$ calibration parameters. The hand has four fingers. Each finger has three active and one passive joint. This results in $16 (= 4 \times 4)$ DH parameters per finger and $64 (= 4 \times 16)$ parameters for the whole hand. Furthermore, each finger has three parallel joints, and the ring and middle finger are mounted with the same orientation.

A. Sensitivity Analysis

To answer the question of whether our contact measurement function h_c can identify all the parameters relevant for the task measurement function h_t, we compute the nullspace of $\mathbf{J}^T\mathbf{J}$ as described in Section IV-A. We evaluate the respective Jacobians $\mathbf{J}_c$ and $\mathbf{J}_t$ for the nominal robot kinematic. For the actual measurement function h_c, we use 100 configurations per pair, which were generated as described in Section V-A. For the task measurement function h_t, we use 100 configurations per finger drawn from the cartesian test set.

Fig. 5 shows the results of this analysis for the DLR-Hand II. The task measurement function h_t is drawn in blue, and our contact measurement function h_c is in red. All the pairs are calibrated simultaneously for the rightmost mode (⬢). Both measurement functions have 56 eigenvalues larger than $1 \cdot 10^{-6}$. Analyzing the eigenvectors confirms that the kernel of our contact measurement function h_c is wholly included in the task measurement function h_t kernel. Therefore, we can identify all parameters relevant to the task. The parallel axis of the fingers can explain the $8 (= 2 \times 4)$ unidentifiable parameters. Each finger has 3 parallel axes along which a shift can be adjusted without influencing the end-effector, resulting in an increased size of nullspace by 2 per finger and 14 identifiable parameters per finger.

The leftmost mode (●) shows the calibration of just a single pair. Considering the calibration of one pair, from the $32 (= 2 \times 16)$ total parameters, $28 (= 2 \times 14)$ should be identifiable. However, the critical insight is that measuring between two chains yields less information than measuring between three or more chains. This holds for the actual scalar distance measurement (red / h_c), where 26 parameters are identifiable, and the task measurement of the relative positions (blue / h_t), where one can identify 27 parameters. However, one can measure even less with the scalar distance function when restricting the measurements to a single pair. An intuition is that two end-effectors can move on a sphere around each other without changing the scalar distance between them. Therefore, in this case, one can not identify all the parameters relevant to the task by pure contact measurements.

This invariance generally resolves when adding a third chain (▲) to the picture, favoring a holistic calibration of the full kinematic tree. Fig. 5 shows the same analysis but for a generic four-fingered hand without parallel axes or mounting frames. In theory, all parameters can be identified for the entire hand. However, in praxis, eigenvalues below 10^{-6} still indicate poor sensitivity.

B. Optimal Experimental Design

We conducted an extensive simulation study to verify our analysis in Section IV. We use the DH formalism to parametrize the forward kinematics as described in Section II and apply noise on the nominal DH parameters to obtain new robot models. For the rotational DH parameters α and θ we applied sampled uniformly $\pm\ 5°$ and for the translational parameters DH d and r we used uniform noise $\pm\ 5\,\text{mm}$.

In this fashion, we created 100 different kinematics to ensure a broad distribution of models. Next, we simulated the data collection step and collected measurements for the actual contact measurement function h_c, the actual cartesian measurement function h_v, and the task measurement function h_t. On average, the models deviated 21mm from the nominal kinematic on the uniform cartesian test set.

TABLE I

CALIBRATION RESULTS IN MM

	Actual Meas. Fun.			Task Meas. Fun.
	mean	std	max	mean
nominal	6.07	3.90	17.70	8.01
calibrated joint offsets	1.04	0.82	5.13	1.36
calibrated full DH	0.72	0.58	3.69	0.89

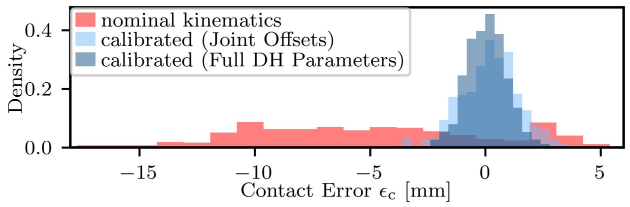


Fig. 10. Comparison of contact error ϵ_c distribution for the different calibration models. The calibration with all DH parameters can reduce the maximal error to $3.7\,\text{mm}$(see also Table I)

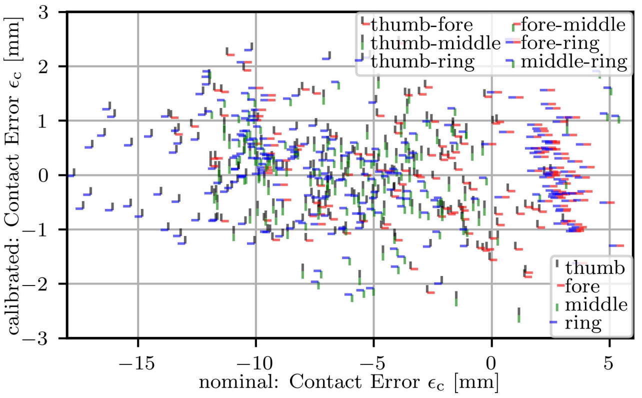


Fig. 11. This plot analyses the contact errors ϵ_c of the individual fingers and pairs before and after the calibration. Each finger has an individual marker defined by a color and an orientation. A measurement of a specific pair is then an overlay of those two finger markers. The x-axis shows the signed distance error before and the y-axis after the calibration, giving detailed insight into the error distribution of the hand.

Fig. 8 show the results of the optimal sample selection introduced in Section IV-B. The error on the cartesian test set is drawn over the calibration set size for different selection strategies. Randomly chosen sets (red) converge slower than the sets designed according to task D-optimality (10). We compare a greedy strategy (green), which at each step adds the sample s_i, which improves the task D-optimality most against the DETMAX algorithm [23] (blue). This procedure tries to swap samples in an existing calibration set to improve the task D-optimality. Both selection strategies outperform the random approach, significantly reducing the mean error to 0.1mm with a set of 300 measurements.

The distribution of configurations for the contact measurements differs from the uniform distribution in the cartesian workspace on which we evaluate the calibrations. This distribution mismatch can explain why the gap between random and selected samples remains significant even for larger calibration sets. Our approach using the task D-optimality as a selection criterion does account for this shift directly and improves the contact calibration with its strict constraint for data generation significantly.

Fig. 9 shows the same analysis but for the cartesian measurement function h_v with an external tracking system. The overall convergence is quicker because the absolute cartesian position yields more information per sample. Furthermore, the difference between the random approach and the dedicated selection criteria is smaller in this setting. One reason is that the calibration set for the external tracking system is closer to the task test set. While proving overall that the selection over task D-optimality is suited to select good calibration sets, those results show that this approach is particularly viable when there is a substantial mismatch between the task measurement function and its test set versus the actual measurement function and its corresponding calibration set.

C. Calibration of the real DLR-Hand II

Following our methodology described in Section IV and Section V-A, we calibrated the DLR-Hand II via pairwise contact measurements. Overall, we collected 300 samples to ensure we have a large enough set for calibration and evaluation. The training-test split was 80/20, and we used cross-validation to get the distribution over the whole dataset. Our findings revealed that 150 samples are sufficient for an accurate calibration. Collecting that data takes 9 minutes, making this automatic calibration procedure easy and quick to use.

The uncalibrated nominal forwards kinematic has a mean error of $6\,\text{mm}$ over all 300 samples and a maximal error of up to $17.7\,\text{mm}$ for the scalar distance measurement. The full error distribution over the signed distance function is shown in red in Fig. 10. Light blue is the calibrated model using only the joint offsets as calibration parameters, reducing the maximal error to $5.1\,\text{mm}$. The entire calibration model, including all DH parameters, is dark blue. This model reduces the maximal error further to $3.7\,\text{mm}$.

Fig. 11 analyses the errors of the individual fingers before and after the calibration. Each finger has an individual marker, and a measurement of a specific pair overlaps those two finger markers. The x-axis shows the signed distance error before and the y-axis after the calibration. Besides the significant overall error reduction, one can see that different finger pairs have other error distributions before and after the calibration. For example, the pair fore-ring is often slightly apart for the nominal model, while the thumb-ring pair is often in deep penetration.

Table I shows the detailed results of the different models. Additionally, the errors in the task space are given. After calibration, one can use (11) to transform the resulting calibration errors of the contact measurement function h_c into the errors of the task measurement function h_t. Starting from the calibration errors σ_m one can first apply $\mathbf{J}_c$ to map to the uncertainties of the calibration parameters $\operatorname{cov}\Theta$ and

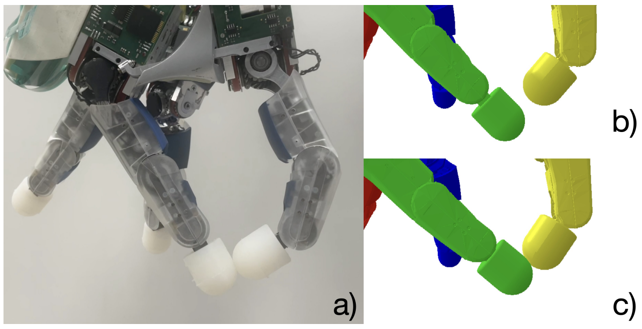


Fig. 12. The DLR-Hand II in a finger contact pose from the testset *a)* with the measured joint angles q mirrored to a nominal *b)* and a calibrated *c)* model. The model's error is visibly reduced from a distance of $8.3\,\mathrm{mm}$ to a penetration of $1\,\mathrm{mm}$, allowing dextrous in-hand manipulation.

then apply in a second step $\mathbf{J}_\mathrm{t}$ to map to the task space. This results in a reduction of the mean error in the task space form $8\,\mathrm{mm}$ for the uncalibrated model to $0.9\,\mathrm{mm}$ after calibration.

Finally, Fig. 12 shows the improvement through our calibration procedure on the real DLR-Hand II. When mirroring a joint configuration q, which is in contact, onto the nominal and the calibrated robot model, one can see the sizeable uncalibrated error and the good fit after the calibration, enabling dextrous grasping and in-hand manipulation.

VII. Conclusions and Future Work

Using a pairwise contact measurement approach, we calibrated the complex robotic DLR-Hand II with 12 active and 4 passive joints. This calibration approach has minimal requirements on the robotic hardware and needs no additional tools, making it easy to apply in different setups. One only needs a method to contact contacts (e.g., torque sensors in our case) and a mathematical model to describe the distance between the body pairs.

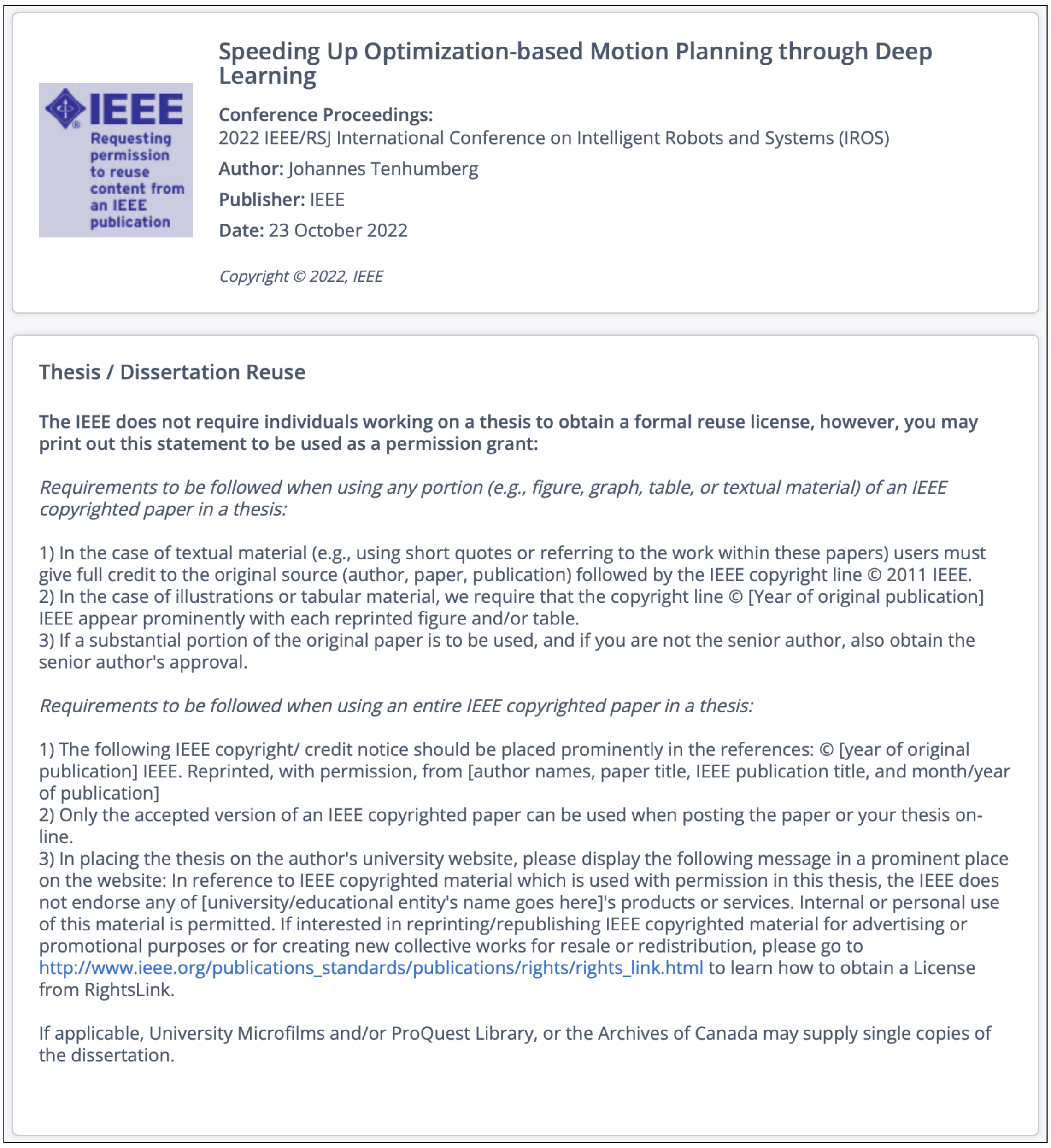
From an uncalibrated distance error of up to $17.7\,\mathrm{mm}$ (roughly equivalent to 10% of the overall size of the workspace), we could reduce maximal error to $3.7\,\mathrm{mm}$ and the average error to $0.7\,\mathrm{mm}$. Looking at the desired task measurement function relevant to dextrous in-hand manipulation, we made a sensitivity analysis to confirm that the scalar distance information between individual finger pairs is enough to identify all relevant DH parameters. Furthermore, we showed that performing a joint calibration of multiple finger pairs yields more information than looking at just a single pair. We used task D-optimality to counteract the mismatch between our desired task measurement function and its corresponding cartesian test set versus the less informative, more constrained contact measurement approach. An exhaustive simulation study verifies this selection approach's effectiveness in balancing the calibration and the desired task.

In future work, we want to extend the calibration approach to the robot structure's elasticities, especially in the drivetrain and the fingertips. For this, forces are applied via the torque-controlled joints while in contact.

References

[1] D. Winkelbauer *et al.*, "A Two-stage Learning Architecture that Generates High-Quality Grasps for a Multi-Fingered Hand," in *International Conference on Intelligent Robots and Systems*, 2022.
[2] ——, "A two-stage learning architecture that generates high-quality grasps for a multi-fingered hand," in *Proc. Int. Conf. Intelligent Robots and Systems*, 2022.
[3] L. Sievers, J. Pitz, and B. Bäuml, "Learning Purely Tactile In-Hand Manipulation with a Torque-Controlled Hand," in *International Conference on Robotics and Automation*, 2022.
[4] J. Pitz, L. Röstel, L. Sievers, and B. Bäuml, "Dextrous Tactile In-Hand Manipulation Using a Modular Reinforcement Learning Architecture," in *International Conference on Robotics and Automation*, 2023.
[5] L. Röstel, J. Pitz, L. Sievers, and B. Bäuml, "Estimator-coupled reinforcement learning for robust purely tactile in-hand manipulation," in *Proc. IEEE-RAS International Conference on Humanoid Robots*, 2023.
[6] J. Butterfaß, M. Grebenstein, H. Liu, and G. Hirzinger, "DLR-Hand II: Next generation of a dextrous robot hand," in *International Conference on Robotics and Automation*, 2001.
[7] S.-M. Lee, K.-D. Lee, S.-H. Jung, and T.-S. Noh, "Kinematic Calibration System of Robot Hands Using Vision Cameras," in *International Conference on Ubiquitous Robots and Ambient Intelligence*, 2013.
[8] N. Tan, X. Gu, and H. Ren, "Simultaneous Robot-World, Sensor-Tip, and Kinematics Calibration of an Underactuated Robotic Hand With Soft Fingers," *IEEE Access*, 2018.
[9] M. Ikits and J. Hollerbach, "Kinematic calibration using a plane constraint," in *International Conference on Robotics and Automation*, 1997.
[10] M. Meggiolaro, G. Scriffignano, and S. Dubowsky, "Manipulator Calibration Using A Single Endpoint Contact Constraint," in *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2000.
[11] Y. Sun, D. J. Giblin, and K. Kazerounian, "Accurate Robotic Belt Grinding of Workpieces with Complex Geometries using Relative Calibration Techniques," *Robotics and Computer-Integrated Manufacturing*, 2009.
[12] A. Joubair and I. A. Bonev, "Kinematic Calibration of a Six-Axis Serial Robot Using Distance and Sphere Constraints," *The International Journal of Advanced Manufacturing Technology*, 2015.
[13] D. J. Bennett and J. M. Hollerbach, "Closed-loop Kinematic Calibration of the Utah-MIT Hand," *Experimental Robotics I*, 1990.
[14] K. Stepanova, T. Pajdla, and M. Hoffmann, "Robot Self-Calibration Using Multiple Kinematic Chains—A Simulation Study on the iCub Humanoid Robot," *IEEE Robotics and Automation Letters*, 2019.
[15] K. Stepanova *et al.*, "Automatic Self-Contained Calibration of an Industrial Dual-Arm Robot with Cameras Using Self-Contact, Planar Constraints, and Self-Observation," *Robotics and Computer-Integrated Manufacturing*, 2022.
[16] A. Roncone, M. Hoffmann, U. Pattacini, and G. Metta, "Automatic Kinematic Chain Calibration Using Artificial Skin: Self-touch in the iCub Humanoid Robot," in *International Conference on Robotics and Automation*, 2014.
[17] R. Zenha, P. Vicente, L. Jamone, and A. Bernardino, "Incremental Adaptation of a Robot Body Schema Based on Touch Events," in *International Conference on Development and Learning and Epigenetic Robotics*, 2018.
[18] L. Rustler *et al.*, "Spatial Calibration of Whole-Body Artificial Skin on a Humanoid Robot: Comparing Self-Contact, 3D Reconstruction, and CAD-Based Calibration," in *International Conference on Humanoid Robots*, 2021.
[19] J. Tenhumberg and B. Bäuml, "Calibration of an Elastic Humanoid Upper Body and Efficient Compensation for Motion Planning," in *International Conference on Humanoid Robots*, 2021.
[20] J. Tenhumberg, D. Winkelbauer, D. Burschka, and B. Bäuml, "Self-Contained Calibration of an Elastic Humanoid Upper Body Using Only a Head-Mounted RGB Camera," in *International Conference on Humanoid Robots*, 2022.
[21] Haiying Hu *et al.*, "Calibrating Human Hand for Teleoperating the HIT/DLR hand," in *International Conference on Robotics and Automation*, 2004.
[22] H. Carrillo *et al.*, "On task-oriented criteria for configurations selection in robot calibration," in *International Conference on Robotics and Automation*, 2013.
[23] T. J. Mitchell, "An Algorithm for the Construction of "D-Optimal" Experimental Designs," *Technometrics*, 2000.

Copyright



Speeding Up Optimization-based Motion Planning through Deep Learning

Conference Proceedings:
2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Author: Johannes Tenhumberg

Publisher: IEEE

Date: 23 October 2022

Copyright © 2022, IEEE

IEEE
Requesting permission to reuse content from an IEEE publication

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Speeding Up Optimization-based Motion Planning through Deep Learning

Johannes Tenhumberg[1,2], Darius Burschka[3] and Berthold Bäuml[1,2]

***Abstract*— Planning collision-free motions for robots with many degrees of freedom is challenging in environments with complex obstacle geometries. Recent work introduced the idea of speeding up the planning by encoding prior experience of successful motion plans in a neural network. However, this “neural motion planning” did not scale to complex robots in unseen 3D environments as needed for real-world applications. Here, we introduce “basis point set”, well-known in computer vision, to neural motion planning as a modern compact environment encoding enabling efficient supervised training networks that generalize well over diverse 3D worlds. Combined with a new elaborate training scheme, we reach a planning success rate of 100 %. We use the network to predict an educated initial guess for an optimization-based planner (OMP), which quickly converges to a feasible solution, massively outperforming random multi-starts when tested on previously unseen environments. For the DLR humanoid Agile Justin with 19 DoF and in challenging obstacle environments, optimal paths can be generated in 200 ms using only a single CPU core. We also show a first successful real-world experiment based on a high-resolution world model from an integrated 3D sensor.**

I. Introduction

At the core, robotic motion planning is about getting from the start joint configuration to a goal configuration while avoiding obstacles in the environment and self-collision along the path. Solving a motion task fast and efficiently does not only mean that the robot spends less time contemplating and more time moving. If a solver can find a feasible solution in a fraction of a second, there opens up the door for more reactive planning and integrating the global planner more tightly into the sensor/vision-action loop. Furthermore, we can tackle more high-level tasks with multiple smaller motion problems if each substep can be solved efficiently.

An interesting approach for speeding up motion planning is not to solve each planning problem anew but to use experience from having solved related motion tasks before. Important for the applicability of such experience-based planners to real-world problems is that they not only allow for arbitrary start and goal configurations but also for arbitrary environment geometries as an input. So, “related tasks” only means that the robot geometry is the same.

This paper presents a deep learning enhancement for an optimization-based planner that allows robot motion planning in high-resolution environments with complex obstacle geometries. For the DLR’s humanoid robot Agile Justin [1] with 19 DoF, feasible trajectories can be computed in only 200 ms on a single CPU core (see Fig. 1).

[1]DLR Institute of Robotics & Mechatronics, Germany; [2]Deggendorf Institute of Technology, Germany; [3]Technical University of Munich, Germany
Contact: `johannes.tenhumberg@dlr.de`

Supported by the Bavarian Ministry of Economics, project SMiLE2gether.

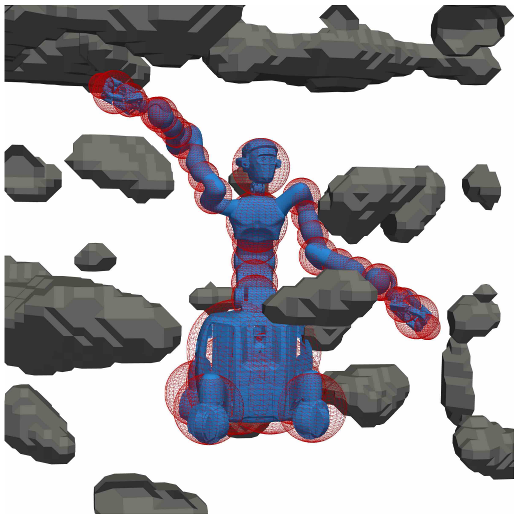


Fig. 1. DLR’s Agile Justin [1] in a challenging obstacle environment, we generate with simplex noise for training and testing. The sphere model of the 19 DoF humanoid is shown as red wireframes. For videos of the motions, visit https://dlr-alr.github.io/2022-iros-planning.

A. Related Work

Two popular but fundamentally different approaches to solving a motion task in robotics are sampling-based planners (SMP) and optimization-based motion planners (OMP). SMP [2] use randomness at their core and can guarantee to globally find a feasible, i.e., collision-free, path (if there is any). Examples are rapidly exploring random trees (RRT) and their extensions, which explore the configuration space iteratively and build a graph of possible configurations until a branch finds the goal. As the search space grows exponentially with the robots’ degrees of freedom (DoF), vanilla variants of SMP don’t scale well to complex robots.

In OMP, the motion task is formulated as an optimization problem [3, 4, 5] and gradient-based techniques are used for non-convex optimization to find a solution. The computational complexity of those methods can, in principle, scale linearly with the DoF[1] and converge quickly to a smooth trajectory. However, they only find a local minimum and strongly depend on the initial guess.

For non-trivial robot kinematics and environments, the objective function usually has many local minima. Therefore, a multi-start approach is needed to find a feasible global solution, massively slowing down OMP in complex scenarios. STOMP [4] and CHOMP [3] try to mitigate the strong

[1]Think of gradient descent where for each iteration only the computation of the gradient and its addition to the current joint angels are performed – both operations linear in the DoF when using, e.g., automatic differentiation.

dependency on an initial guess by introducing stochasticity to the optimization. TrajOpt [5] uses convex hulls to represent the robot and its environment and increases the attraction basin for each optimum. But still, OMP needs multi-starts to find a feasible global solution.

OMP can be sped up by using experience from previously solved motions tasks by providing an *educated* initial guess. Jetchev and Toussaint [6] first proposed to save a database of feasible trajectories and look up a reasonable first guess for a new problem. Merkt et al. [7] improved the idea through more efficient database storage and tested it on a humanoid robot. But for both methods, only results for fixed or only slightly varying environments are shown.

Instead of databases, neural networks are often used to encode the experience. The expectation is that they are more memory efficient, encode various solutions implicitly, learn a general understanding of feasible trajectories, and produce useful predictions in unseen settings. Qureshi et al. [8, 9] coined the term Motion Planning Network (MPNet) and then improved on the idea. They used an RRT planner to collect feasible trajectories in 2D and 3D worlds and encode the environment with point clouds. Their method works on the Baxter robot over ten known table scenes with 1000 paths per scene. Strudel et al. [10] showed that they could outperform these results by employing the PointNet [11] architecture to encode the point clouds of the environments. They achieved good results in 3D with a sphere and a rigid S-shape with three translational and rotational DoFs but did not consider a robotic application. Bency et al. [12] and Lembono et al. [13] applied variations of the idea successfully to the two humanoids, Baxter and PR2. However, they only used a single fixed environment without generalization to different worlds. Lehner and Albu-Schäffer [14] trained a Gaussian mixture model to steer the search in a probabilistic roadmap. The approach was demonstrated on a real 7 DoF robot but equally only for one fixed world.

Also, other learning methods have been applied to this problem. For example, Jurgenson and Tamar [15] used reinforcement learning with convolution layers to process the occupancy map of the world for 2D serial robots. They invoke a classical planner only for cases where random exploration fails to find a feasible solution, so they implicitly use this expert knowledge to guide the training. Pandy et al. [16] introduces a different approach, where the dataset generation is skipped entirely, and the network is directly trained using the objective function of the planning problem as the training loss, i.e., no supervision is used. However, they only use geometric primitives to represent environments with few obstacles, limiting the flexibility.

For a more detailed overview of the literature, also refer to Surovik et al. [17]. They introduce the term "data-driven trajectory initialization" (DDTI) to generalize the ideas of "trajectory prediction" or "memory of motion". To our best knowledge, up to now, there is no experienced-based method for speeding up motion planning that can predict feasible paths for a complex robot in a large set of challenging and previously unseen 3D words.

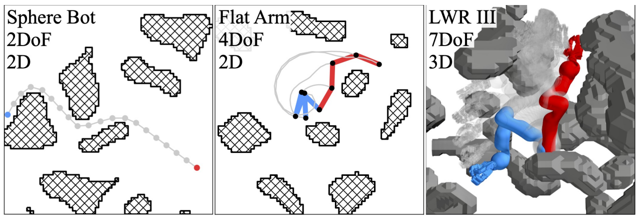


Fig. 2. Three different robots in random 2D and 3D environments, generated with simplex noise. A motion problem is described by the world (hatched), the start configuration (blue), and the goal configuration (red). The solution, the shortest feasible path from start to goal, is drawn in gray. See Table I for an overview of the different robots and worlds in numbers.

B. *Contributions*

We propose a neural network that encodes the experience of optimal trajectories over various tasks and worlds. We train the network supervised to predict the correct path for a given motion problem consisting of a world (obstacle environment) and start and end configurations. We then use the prediction of the network to warm-start an OMP, which ensures feasibility and smoothes the path. Our main contributions are:

- Introducing substeps into CHOMP[3] to explicitly calculate the swept volume to facilitate that no collisions are missed between discrete waypoints.
- The introduction of the basis point sets (BPS), described by Prokudin et al. [18], into learning-based motion planning. This memory- and computational-efficient encoding enables training and generalization for complex robots in challenging environments.
- Building new demanding datasets of motion tasks and optimal solution paths (via OMP) for training and testing. The dataset includes autogenerated random worlds with configurable complexity.
- A training scheme, where we combine the network and the objective function as a metric to efficiently clean, extend and boost an initial dataset.
- Extensive experiments in simulation for different robots ranging from a simple 2D sphere bot up to the 19 DoF humanoid Agile Justin in 3D. For Agile Justin, we also report a first real-world experiment, showing that Sim2Real transfer is working.

II. CHOMP-like Motion Planning

For generating the training samples for our network as well as for online post-processing, we use OMP. Ultimately, the neural network should do the primary workload in solving a motion task. Therefore the runtime efficiency of the OMP is not of utmost importance. We implemented an OMP similar to CHOMP [3], where the robot is modeled as spheres and the environment by a signed distance field (SDF). A significant extension to CHOMP is that we introduce additional substeps interpolating between the time steps. This way, the swept volume can be computed to any arbitrary accuracy, i.e., guaranteeing to miss no obstacle, without increasing the number of optimization variables (see Fig. 3).

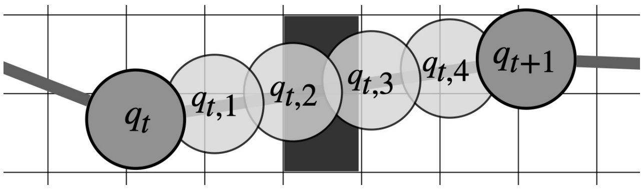


Fig. 3. Substeps $q_{t,u}$ between two discrete waypoints q_t and q_{t+1} to explicitly calculate the swept volume of the path in higher resolution.

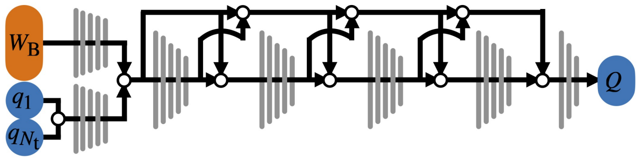


Fig. 4. Network architecture to map from a given motion task (world, start and end configuration) to an optimal path. Blocks of tapered Fully Connected Layers (gray) are combined like the DenseNet architecture [19] via skip-connections and concatenations. See the bottom of Table I for the number of network parameters used for the different robots.

OMP formulates the motion task of getting from point A to point B as an optimization problem with the desired path as the optimum. The path Q consists of a discrete set of waypoints of joint configurations

$$Q = [q_1, \dots, q_{N_\mathrm{t}}],\; q_t \in \mathbb{R}^{N_\mathrm{q}}, \tag{1}$$

and the task is encoded by an objective function $U(Q)$ measuring the quality of a given path. While, in general, the specific objective can be chosen freely, the two usually used terms to get a collision-free and short path are

$$U(Q) = U_\mathrm{C}(Q) + \lambda U_\mathrm{L}(Q). \tag{2}$$

The length cost U_L is given by

$$U_\mathrm{L}(Q) = \frac{N_\mathrm{t} - 1}{|q_{N_\mathrm{t}} - q_1|^2} \sum_{t=1}^{N_\mathrm{t}-1} |q_{t+1} - q_t|^2, \tag{3}$$

which favors short and smooth trajectories. It is often convenient to scale the length cost by the minimal possible path length, the direct connection $|q_{N_\mathrm{t}} - q_1|^2$. Then, the shortest possible path always has a cost of one.

To calculate the collision cost between the robot and the environment, we need a model for both. The forward kinematics $F = f(q)$ maps from joint configurations to the link frames F_i and each link's geometry is described by a set of spheres $\boldsymbol{S}_i = \{x_{ik}, r_{ik}\}_{k=1}^{N_{si}}$ with centers and radii. The world is represented by a SDF $D(x)$ which gives the distance to the closest obstacle for each point x in the workspace. The collision cost is then given by the sum of all the collisions of the different body parts along the path

$$U_\mathrm{C}(Q) = \sum_{t,u}^{N_\mathrm{t},N_\mathrm{u}} \sum_{i=1}^{N_\mathrm{f}} \sum_{k=1}^{N_{si}} c\Big(D\big(F_i(q_{t,u}) \cdot x_{ik}\big) - r_{ik}\Big), \tag{4}$$

$$q_{t,u} = q_t + \frac{u}{N_\mathrm{u}}(q_{t+1} - q_t). \tag{5}$$

In extension to the original CHOMP algorithm [3], we subdivide the path between two waypoints into substeps $q_{t,u}$ via linear interpolation (see Fig. 3). This way, a collision-free path can be guaranteed, when the number of substeps N_u is adjusted to the step length and the voxel size of the SDF. So, the swept volume of each sphere is *explicitly* computed instead of the *implict* computation CHOMP performs via a projection of the cartesian velocity vector of the moving spheres.

The smooth clipping function only considers the parts which are in collision by setting positive distances to 0. Thus, a collision-free path has a collision cost of 0.

$$c(d) = \begin{cases} -d + \frac{\epsilon}{2} & \text{, if } d < 0 \\ \frac{1}{2\epsilon}(d - \epsilon)^2 & \text{, if } 0 \le d \le \epsilon \\ 0 & \text{, if } \epsilon < d \end{cases} \tag{6}$$

In addition to collisions with the world, complex robots must also account for self-collision. Again, the cost sums up all the penetrations between the different body pairs

$$U_\mathrm{s}(Q) = \sum\nolimits_{t,u}^{N_\mathrm{t},N_\mathrm{u}} \sum\nolimits_{j>i}^{N_\mathrm{f},N_\mathrm{f}} \sum\nolimits_{k,l}^{N_{\mathrm{s}i},N_{\mathrm{s}j}} c\Big(\big|F_i(q_{t,u}) \cdot x_{ik} - F_j(q_{t,u}) \cdot x_{jl}\big| - r_{ik} - r_{jl}\Big). \tag{7}$$

Here again, we use substeps to guarantee collision-free paths via explicitly checking the swept volume. To our knowledge, no other OMP-based planner is doing this.

With this formulation, the optimal path and solution to the motion task is the one with the lowest objective

$$Q^* = \operatorname*{argmin}_{Q} U(Q). \tag{8}$$

We use gradient descent for iteratively finding a minimum of the objective function, starting with the initial guess Q_0,

$$Q_{i+1} = Q_i - \alpha \frac{\partial U(Q_i)}{\partial Q_i}. \tag{9}$$

We use vanilla gradient descent as efficiency in the OMP part is not the primary concern of our method and it allows for easy adaption and parameterization (only α)

III. Dataset Adaption for Efficient Learning

The idea is to no longer rely on random multi-starts and speed up the planning time by using a neural network to predict an initial guess for OMP. The network should encode the experience of successful paths by learning a mapping from a motion task, consisting of a world and a start q_1 and end configuration q_{N_t}, to the intermediate waypoints of an optimal path $Q^* = [q_2, ..., q_{N_\mathrm{t}-1}]$. Fig. 4 shows the network architecture we use. Besides encoding the in- and output of the network, a crucial point for supervised learning is the dataset. The following section discusses several insights into the generation and usage of such a dataset with and for OMP. Our methods substantially increase the final prediction quality of our network and make training more efficient. See Section VI-A for experimental validation of these methods.

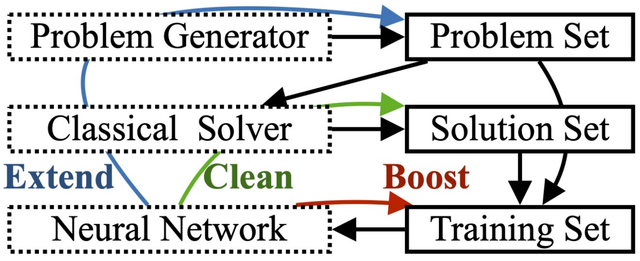


Fig. 5. Scheme showing the connection between non-learning-based solver, dataset, and neural network. The colored arrows indicate the information flow for cleaning, extending, and boosting the dataset with the guidance of the network. Cleaning: use the solver to update labels in the solutions set; Boost: overrepresent existing hard examples in training; Extend: generate new hard examples for the network and add them to the problem set.

A. Challenging Samples

The training data distribution should represent the actual application and focus on challenging examples. Suppose the dataset is too easy, and direct linear connections from A to B drastically outweigh more complex trajectories. In that case, training the network can quickly get stuck in a local minimum and predict only straight lines, regardless of the given task. One possibility of generating a dataset with more challenging motion problems is to consider only samples where OMP using the direct connection as an initial guess does not converge to a feasible path.

B. Consistent Samples

Besides finding challenging samples, the ambiguity of motion planning (more than one feasible solution) can become a problem for trajectory regression. Even if we assume that we can resolve the ambiguity of the optimal path via the objective for the shortness of the path, there will be “close calls” in the dataset (almost the same objective values but fundamentally different paths). Furthermore, the classical planner using a limited number of multi-starts can only produce suboptimal labels, making it hard for the network to learn consistent mapping. This is especially true for challenging tasks where the classical planner often fails and only produces feasible paths in a small fraction of the tries.

C. Symmetries of Motion Planning

Data generation is costly, so one can use symmetries in motion planning to efficiently use the information in each sample. If one has the optimal path from A to B, one also has the solution from B to A. This assumption is no longer valid if terms in objective function break the temporal symmetry. Furthermore, many robots also have spatial symmetry axes. Often, it is possible to align the first joint of the robot with one axis of the cartesian coordinate system of the environment. Doing so allows us to rotate the world and this first joint simultaneously without changing the optimality of the resulting trajectory in the new world. Chamzas et al. [20] use this spatial symmetry to store paths in their database more efficiently. While ideally, one wants to integrate these

Algorithm 1 Improvement of network and dateset

```
procedure MAIN
    create initial dataset $G = \{(x_i, y_i)\}_{i=1}^{N_G}$ with OMP
    train net $\Theta$ on dataset $G$
    while improvement on testset do
        CleanDataset($G, \Theta$)
        ExtendDataset($G, \Theta,\ N = |G|/20$)
        BoostDataset($G, \Theta,\ p_{\text{perc}} = 0.8,\ p_{\text{ratio}} = 0.9$)
        train net $\Theta$ on dataset $G$
procedure CLEANDATASET($G$, $\Theta$)                    ▷ Clean
    for $(x_i, y_i)$ in $G$ do
        $y_p \leftarrow \Theta(x)$
        $y_p^* \leftarrow \text{OMP}(x, y_p)$
        if $U(y_p^*) \leq U(y)$ then
            $G$.replace($y \leftarrow y_p^*$)
procedure EXTENDDATASET($G$, $\Theta$, $N$)             ▷ Extend
    for $k \leftarrow 1$ to $N$ do
        $x_i \leftarrow$ sampleNewProblem()
        if $V_p < U(\Theta(x_i))$ then
            $y_i \leftarrow \text{OMP}(x_i)$
            $G$.append($(x_i, y_i)$)
procedure BOOSTDATASET($G$, $\Theta$, $p_{\text{perc}}$, $p_{\text{ratio}}$) ▷ Boost
    $V \leftarrow [U(\Theta(x_i))$ for $(x_i, y_i)$ in $G]$
    $V_p \leftarrow$ percentile($W, p_{\text{perc}}$)
    for $(x_i, y_i)$ in $G$ do
        if $(U(\Theta(x)) < V_p$ and random(0, 1) $< p_{\text{ratio}})$ then
            $G$.remove($(x_i, y_i)$)
```

symmetries directly in the data representations or the network architecture, we used them to augment and increase the dataset.

D. Interplay between Network and Dataset

We propose to use the neural network Θ to correct and enhance its own training data $G = \{(x, y)\}$. This approach is possible whenever synthetic data is used for training, and one has an objective metric to measure the quality of a prediction. The assumption is that the network can learn some aspects of the problem even on an imperfect dataset and its predictions become better than random guessing.

When generating a dataset with and for OMP, we can use the objective $U(Q)$ as a universal quality metric. The idea of interweaving the network training closer with the dataset generation and improvement is summarized in Fig. 5 and described by Algorithm 1. In what follows, we give a detailed explanation of the different methods.

1) Clean: First, the network can be used as guidance to double-check where the dataset is inconsistent. If the label and the network's prediction Q_{p} are close, and the respective objective $U(Q_{\text{p}})$ is small, no action is necessary. However, if there is a discrepancy between prediction and label, it is worthwhile to use more multi-starts with the OMP to get a better label for the sample. If a prediction has a better objective than the current label, we can replace it without adding any bias to the dataset. Doing so will improve the

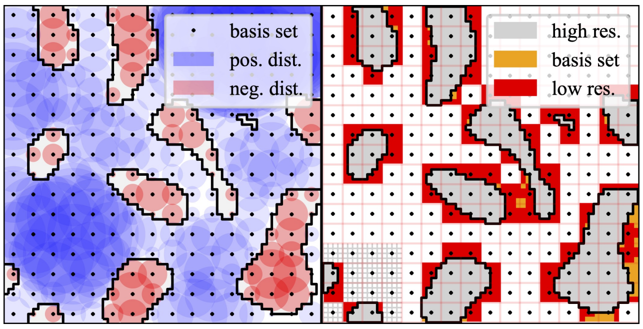


Fig. 6. Visualization of the representation capabilities of basis point sets. On the left, a regular BPS with $16 \times 16 = 256$ points and their respective distances to the closest obstacle in an occupancy map with $64 \times 64 = 4096$ pixels is shown. Blue areas describe positive distances to obstacles and are guaranteed free of obstructions. Red regions show negative distances and are completely inside barriers. We can draw no direct conclusion from the white areas, but they have to be marked as obstacles to be conservative. The BPS representation for this regular grid is equivalent to subsampling a high resolution 64×64 SDF. The right image shows the reconstruction of the BPS to the full image with the errors marked in orange. A patch of the original grid is added to highlight the difference in resolution. The far more conservative result of reducing the resolution of the occupancy grid by the same factor is highlighted in red. This comparison demonstrates that the basis set preserves more information than conservatively shrinking the occupancy grid by the same factor. A further advantage of the untruncated distances is that even if there are no nearby data points close for a basis point, it can nevertheless help in representing a surface further away.

labels and make the dataset more consistent. This makes it easier for the network to find the underlying patterns.

2) Boost: After some training, the network has learned to predict good paths for the relatively simple samples, but the more challenging outliers are still not solved. Therefore, we use boosting to select challenging samples with higher probability during training, increasing the incentive to learn these samples. We steer this kind of curriculum again by using U as a metric: the higher the difference between the predicted and actual cost for a given sample is, the more challenging it is. In our experiments, the challenging tasks for the network correlated well with the relative path length and the number of obstacles in the scene.

3) Extend: Lastly, one can use the network to generate new samples. The idea is to use the network's performance on a new sample as a metric for information gain. To improve the network, one wants specifically to add samples where the network performs poorly. To decide this before spending resources to produce a new label using OMP, one can use the objective of the prediction $U(Q_\text{P})$. If it is small, the network can solve this task already. However, if the objective is significant, the task is challenging for the network, and we include it in the training set.

IV. Basis Point Set as Efficient World Encoding

If a network should speed up the OMP approach with a valuable warm-start, it needs to "understand" the robot motion in arbitrary unseen environments. Hence, a suitable encoding of the world is essential.

Two central environment representations in robotics and computer vision are occupancy grids W_O and point clouds W_P. Both have their advantages and disadvantages. While occupancy grids can be processed like images in 2D, their memory inefficiency and the high computational cost for the convolution operations become a burden in 3D. On the other hand, point clouds are a denser representation. Still, they have no fixed size and no inherent ordering, making it hard for a network to learn a permutation invariant mapping.

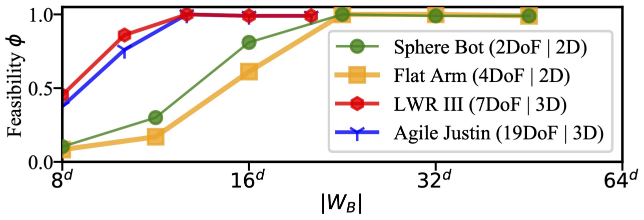


Fig. 7. Influence of the size of BPS $|W_\text{B}|$ on the feasibility ϕ of the network prediction after OMP. The $|W_\text{B}|$ on the x-axis is scaled by the world dimension. Too few points can't represent the obstacles in enough detail for the network to make useful predictions. However, the required number of points is significantly smaller than the resolution of the underlying occupancy grid (64^d). It was not feasible to increase the input layer to 64^3 basis points in 3D.

Prokudin et al. [18] recently introduced a new idea to represent spatial information in computer vision, which is especially suited for deep learning. Choosing a fixed set of basis points once and measuring the distances relative to this set for all new environments does not have the problem of varying permutations and lengths as point clouds have. It is also far more efficient in terms of memory and computation than voxel grids, allowing fast training in high-resolution 3D environments.

Formally, the BPS is an arbitrary but fixed set of points

$$B = [b_1, \dots, b_{N_\text{b}}],\ \ b_i \in \mathbb{R}^d. \tag{10}$$

The feature vector W_B passed to the network consists of the distances to the closest point in the environment for all basis points. If the environment is defined by a point cloud W_P this can be calculated by

$$W_\text{B} = [\min_{x_i \in W_\text{P}} |b_1 - x_i|, \dots, \min_{x_i \in W_\text{P}} |b_{N_\text{b}} - x_i|]. \tag{11}$$

Alternatively, if the environment is given by an occupancy grid or a distance field D like we used for OMP in Section II, one can directly look up the feature vector

$$W_\text{B} = [D(b_i), \dots, D(b_{N_\text{b}})]. \tag{12}$$

With the second approach, it is possible to use signed distances. This adds directly a notion of inside and outside to the representation of the world.

Before using this representation to train a neural network, in Fig. 6 we analyze its properties with regard to efficiency and safety in a setting without learning.

Although an occupancy grid directly represents the cartesian workspace, the mapping to a robot's configuration space can be complex. But the planning network has to understand this mapping to connect movements in joint space with its implications in the world to be able to plan collision-free paths. In Section VI, we show in experiments that the basis

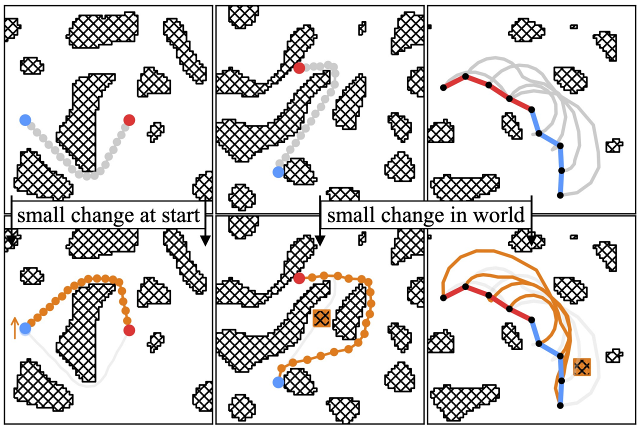


Fig. 8. Three examples of the networks' prediction for only slightly altered problems. The changes in the start configuration (left) and the environment (middle, right) are highlighted in orange. The networks' predictions after optimization for those new problems are shown in the bottom row, indicating they can react sharply to small input changes. For easier visibility in 3D, we refer to animations for the other robots on the website

point set enables to learn that connection for complex robot kinematics and diverse environments.

V. Experimental Setup

A. Dataset

We constructed extensive datasets for different robots (see Figs. 1, 2 and 8) to validate our methods. For 2D, we investigated a simple sphere robot with 2 DoF and a serial arm with 4 DoF. Furthermore, we used two real 3D robots: the LWR III [21], a robotic arm with 7 DoF, and humanoid robot DLR Agile Justin [1] with 19 DoF distributed over an upper body and two arms.

To generate diverse and challenging worlds, we used simplex noise [22]. This gradient noise is used in video games to create random but naturally-looking levels. A typical example is a continuous height map. By changing the cut-off threshold and the resolution of this noise, we can vary the density and form of the obstacles in the binary occupancy grid. To ensure that the environments are not too densely packed with obstacles, at least 200/1000 random robot configurations q must be feasible to include a world in the dataset. Examples can be seen in Figs. 1 and 2.

To generate the labels for the samples, we used the OMP approach described in Section II, with naive gradient descent and fixed step size. The paths consist of $N_\mathrm{t} = 20$ waypoints, and to make the dataset challenging, we only included hard tasks that were not solvable starting from a straight line as initial guess. All other paths were discarded as too easy. We used up to 100 multi-starts and always picked the shortest feasible solutions as the correct label. The heuristic for generating the initial guesses was to use 1 to 3 random points in the configuration space and connect them linearly with the start and endpoint. See Table I for an overview of the robots and the datasets[2]. The table also shows initially

[2]All datasets plus additional information on their generation and use in training are provided at https://dlr-alr.github.io/2022-iros-planning.

TABLE I

Overview of the datasets and nets for the different robots.

	Sphere Bot	Flat Arm	LWR III	Agile Justin
DoF	2	4	7	19
World size	$10 \times 10\,\mathrm{m}^2$	$1.0 \times 1.0\,\mathrm{m}^2$	$1.2 \times 1.2 \times 1.2\,\mathrm{m}^3$	$3 \times 3 \times 3\,\mathrm{m}^3$
Grid dimensions	64×64	64×64	$64 \times 64 \times 64$	$64 \times 64 \times 64$
# Worlds	10^4	10^4	10^4	10^4
# Paths	0.6×10^6	6.5×10^6	2.2×10^6	3.7×10^6
Avg. time p. core	0.1 s	0.8 s	3.1 s	8.4 s
# Improvements	0.1×10^6	0.3×10^6	0.2×10^6	0.3×10^5
# Extensions	0.5×10^5	1.5×10^5	3.0×10^5	5.0×10^5
Avg. $U_\mathrm{L}(Q)$	1.589	1.7136	1.551	1.483
Avg. Feas. ϕ	67.3 %	32.6 %	54.5 %	44.1 %
Avg. time p. core	0.1 s	0.8 s	3.1 s	8.4 s
Net				
# In $\rightarrow$ # Out	$516 \rightarrow 39$	$520 \rightarrow 72$	$2062 \rightarrow 126$	$2086 \rightarrow 342$
$\lvert W_\mathrm{B} \rvert$	512	512	2048	2048
N_t	18	18	18	18
# Parameters	3.4×10^6	7.5×10^6	2.4×10^7	4.1×10^7

mislabeled paths and paths specifically included to challenge the network.

B. Network

The last lines of Table I show the network details in numbers and Fig. 4 displays the general architecture. All the networks were trained purely supervised with a mean squared error between the predicted path Q_p and the label Q as loss function. As the encoding for the path, the deviation from the straight line is used. This representation implies that even an untrained network producing only random noise around zero can make meaningful predictions. For start and end, we use the normalized joint vectors q_1 and q_{N_t} as input. As environment encoding, we use the BPS described in Section IV with a hexagonal closed packing and only consider points inside the robots' maximal reach. See Fig. 7 for an analysis of the dependency of the prediction quality on the size of the BPS.

VI. Evaluation Results

From the 10000 environments we generated for each robot, we used 9000 for the network's training and the remaining unseen worlds for testing. All the results in Section VI are based on this unseen test set with 10000 hard motion tasks. As a quality measure we report the feasibility rate ϕ, i.e., the quotient of the number of feasible paths and the size of the test set.

A. Dataset Adaption

Table II shows the influence of the methods for dataset adaption during training as discussed in Section III. As metric we use the feasibility rate ϕ of the predicted paths after further iterations with the OMP as described in Section II. First, we compare the hard and the easy dataset. Because the easy dataset consists only of paths produced from straight lines, the overall variance is too slight, and the network does not learn to avoid the obstacles. This network is not able to solve the test set of hard examples. Next, we introduced the different modes of data augmentation to increase the size of the dataset. The temporal and spatial symmetries improve the feasibility rate ϕ without additional computing costs. The

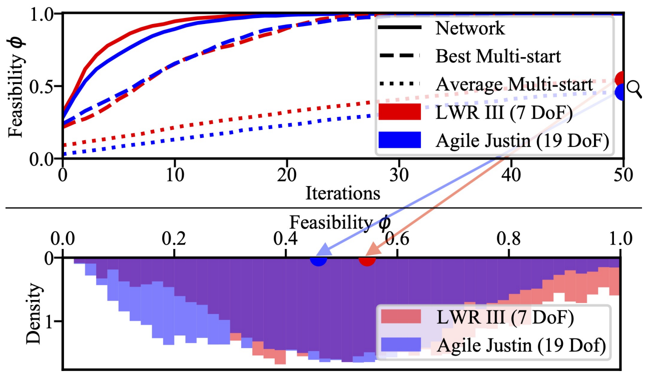


Fig. 9. Top: Average convergence to feasibility ϕ of OMP for different initial guesses. The prediction of the network outperforms the average and even the best out of 100 multi-starts significantly. Bottom: Distribution of the average feasibility ϕ of the random multi-starts after 50 OMP iterations.

TABLE II
ANALYSIS OF DIFFERENT DATASET DISTRIBUTIONS AND EXTENSIONS & TRAINING METHODS FOR AGILE JUSTIN.

Dataset		Training		Feasibility ϕ	
# Cleans	Distribution	Aug.	Boost	Network	+OMP
0	easy	no	no	0.042	0.347
0	hard	no	no	0.126	0.653
0	hard	axis	no	0.133	0.691
0	hard	time	no	0.138	0.736
0	hard	both	no	0.143	0.772
0	hard	both	yes	0.171	0.859
1	hard	both	yes	0.196	0.893
2	hard	both	yes	0.217	0.925
3	hard	both	yes	0.223	0.941
3	hard + ext.	both	yes	0.283	1.00

number of cleanings describes how often the labels were updated with the help of the neural network. Each iteration brings the labels closer to the optimal solution and makes the dataset more consistent, leading to better results. At this stage, the network performs already well with a success rate of over 85 %, but there are still tasks the net can not solve. We add the boosting technique to overrepresent more challenging samples during training to increase the feasibility further. As the final step, we use the trained network to generate more challenging samples. With this approach, we achieved 100 % feasibility on the hard unseen test set.

B. Comparison to Random Multi-start

The capabilities of our method become apparent when we compare the prediction of the network to the heuristic with random multi-starts used to create the dataset. In Fig. 9 the convergences to a feasible path of different initial guesses are displayed for the LWR III and Agile Justin. Without any experience, the best one can do, is to try random multi-starts and hope one of them converges. From the 100 multi-starts we used per task, only 50 % converge to a feasible solution after 50 iterations. Even the lucky initial guess, which converged the fastest for each problem, gets outperformed by the network's prediction. The crucial difference is that our network does not depend on chance but can reliable predict initial guesses that converge after a few iterations to a feasible solution.

The actual speed gain is even more prominent when looking at the distribution over different motion tasks (see bottom of Fig. 9). There are problems for the LWR III and Agile Justin where only 10 % or less of all multi-starts converge to a feasible path. If one wants to find a solution to such a problem with 90 % confidence, one would need more than $\log(1-0.1)/\log(1-0.9) > 20$ multi-starts, making the initial guess of the network effectively over 20 times faster.

On our test machine (Intel i9-9820X @ 3.30 GHz, 32 Gb RAM), a single iteration of gradient descent for one path of Agile Justin takes $10\,\mathrm{ms}$ on a single core. Using the network's prediction as a warm-start and stopping each sample after convergence leads to an average run time of $182(\pm 29)\,\mathrm{ms}$ with a worst-case of $334\,\mathrm{ms}$.

C. World Encoding

Prokudin et al. [18] demonstrated that the BPS with fully connected layers is superior to occupancy maps with CNNs or point clouds with a PointNet architecture, both in terms of required network parameters and training performance. We can confirm those findings for motion planning. The large memory requirements in 3D made training prohibitively slow and hard to iterate on network architecture or training methods. Furthermore, looking towards the application on Agile Justin, the BPS can readily be integrated with the high-resolution SDFs acquired from the robot's depth camera [23].

The BPS representation and the proposed training scheme on the worlds from simplex noise were robust enough to even generalize to some first results on a real robot (see Fig. 10). Only trained on those random worlds, the network was able to make valuable predictions from the data collected by Agile Justin's depth camera [23]. The predictions as warm-start for OMP could solve the unseen motion tasks which needed multi-starts otherwise in under 200 ms.

Learning-based motion planning for such a complex robot was not tackled before in unseen environments, so we only compare it against simpler problems. MPNetPath, for example, takes $0.59\,s$ to plan for the 7 DoF Baxter arm in

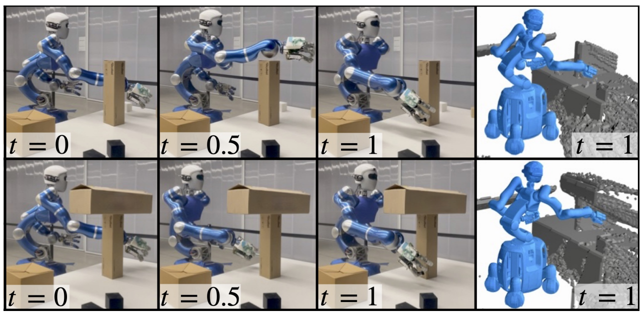


Fig. 10. Agile Justin in two table scenes with boxes. The robot applies a different strategy for obstacle avoidance after the top route is blocked. The rendered voxel model on the right shows the input for the neural network.

a known table scene [9]. Non-learning-based methods like CHOMP take multiple seconds for similar scenes [5].

Fig. 8 shows a qualitative analysis of the network predictions. In motion planning, small changes in the problem often lead to fundamentally different solutions. Our worlds and training were challenging enough that the networks react sharply to small changes in the input, predicting completely different solutions to only slightly altered problems.

VII. Conclusions

We successfully trained motion planning networks using supervised learning on diverse and challenging datasets that predict paths close to the global optimum for previously unseen environments. Using this prediction as a warm-start for optimization-based motion planning massively outperforms random multi-start. For the complex robot Agile Justin with 19 DoF, planning takes only 200 ms on a single CPU core. This shows for the first time that learning-based motion planning works in previously unseen environments for such a complex robot.

One key to success is the basis point set encoding for the environment borrowed from computer vision which we introduced to motion planning and scales well to high-resolution 3D worlds. In addition, we autogenerate a training dataset of hard examples, i.e., situations for which the vanilla OMP struggles and for which the trained network should later provide an educated initial guess. We also introduced a scheme to further adapt the dataset during training by cleaning, boosting, and extending the dataset based on a metric defined by the (current) neural network and the objective function of the OMP. This approach leads to a challenging and consistent dataset on which a network can efficiently be trained and improved.

In the future, we extend the planning problem towards manipulation and grasping by incorporating the inverse kinematics so that no longer a goal configuration but only the goal pose of the end-effector has to be provided. We will also further investigate and increase the real-world capabilities of our method. As it is expensive to create a vast amount of real-world data, the goal is that our architecture and the autogenerated dataset allow for a robust transfer of the experience to real scenes.

References

[1] B. Bäuml *et al.*, "Agile justin: An upgraded member of DLR's family of lightweight and torque controlled humanoids," in *Proc. IEEE International Conference on Robotics and Automation*, 2014.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[3] M. Zucker *et al.*, "CHOMP: Covariant Hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 8 2013.

[4] M. Kalakrishnan *et al.*, "STOMP: Stochastic trajectory optimization for motion planning," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4569–4574, 2011.

[5] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[6] N. Jetchev and M. Toussaint, "Fast motion planning from experience: Trajectory prediction for speeding up movement generation," *Autonomous Robots*, vol. 34, no. 1-2, pp. 111–127, 2013.

[7] W. Merkt, V. Ivan, and S. Vijayakumar, "Leveraging Precomputation with Problem Encoding for Warm-Starting Trajectory Optimization in Complex Environments," in *International Conference on Intelligent Robots and Systems (IROS)*, 10 2018, pp. 5877–5884.

[8] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion Planning Networks," in *International Conference on Robotics and Automation (ICRA)*, vol. 2019-May, 2019, pp. 2118–2124.

[9] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion Planning Networks: Bridging the Gap between Learning-Based and Classical Motion Planners," *IEEE Transactions on Robotics*, vol. 37, pp. 48–66, 2021.

[10] R. Strudel *et al.*, "Learning Obstacle Representations for Neural Motion Planning," in *Conference on Robot Learning (CoRL)*, 2020.

[11] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 77–85, 2017.

[12] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural Path Planning: Fixed Time, Near-Optimal Path Generation via Oracle Imitation," in *International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[13] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of Motion for Warm-Starting Trajectory Optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2594–2601, 2020.

[14] P. Lehner and A. Albu-Schäffer, "The repetition roadmap for repetitive constrained motion planning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3884–3891, 2018.

[15] T. Jurgenson and A. Tamar, "Harnessing Reinforcement Learning for Neural Motion Planning," in *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, 6 2019, pp. 1–13.

[16] M. Pandy, D. Lenton, and R. Clark, "Unsupervised Path Regression Networks," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 9 2021, pp. 1413–1420.

[17] D. Surovik *et al.*, "Learning an Expert Skill-Space for Replanning Dynamic Quadruped Locomotion over Obstacles," in *Conference on Robot Learning*, 2020.

[18] S. Prokudin, C. Lassner, and J. Romero, "Efficient Learning on Point Clouds with Basis Point Sets," in *International Conference on Computer Vision (ICCV)*, 8 2019, pp. 4332–4341.

[19] S. Jégou *et al.*, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," *CoRR*, vol. abs/1611.09326, 2016.

[20] C. Chamzas, A. Shrivastava, and L. E. Kavraki, "Using Local Experiences for Global Motion Planning," in *International Conference on Robotics and Automation (ICRA)*, vol. 2019-May. IEEE, 5 2019, pp. 8606–8612.

[21] G. Hirzinger *et al.*, "DLR's torque-controlled light weight robot III - are we reaching the technological limits now?" in *Proc. IEEE International Conference on Robotics and Automation*, 2002, pp. 1710–1716.

[22] S. Gustavson, "Simplex noise demystified," Linkoeping University, Sweden, Tech. Rep., 2005.

[23] R. Wagner, U. Frese, and B. Bäuml, "3D modeling, distance and gradient computation for motion planning: A direct GPGPU approach," in *2013 IEEE International Conference on Robotics and Automation*, 2013.

Copyright

IEEE
Requesting permission to reuse content from an IEEE publication

Efficient Learning of Fast Inverse Kinematics with Collision Avoidance

Conference Proceedings:
2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)
Author: Johannes Tenhumberg; Arman Mielke; Berthold Bäuml
Publisher: IEEE
Date: 12-14 Dec. 2023

Copyright © 2023, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Efficient Learning of Fast Inverse Kinematics with Collision Avoidance

Johannes Tenhumberg[1,2,3]   Arman Mielke[1,3]   Berthold Bäuml[1,2]

***Abstract*— Fast inverse kinematics (IK) is a central component in robotic motion planning. For complex robots, IK methods are often based on root search and non-linear optimization algorithms. These algorithms can be massively sped up using a neural network to predict a good initial guess, which can then be refined in a few numerical iterations. Besides previous work on learning-based IK, we present a learning approach for the fundamentally harder problem of IK with collision avoidance in diverse and previously unseen environments. From a detailed analysis of the IK learning problem, we derive a network and unsupervised learning architecture that removes the need for a sample data generation step. Using the trained network's prediction as an initial guess for a two-stage Jacobian-based solver allows for fast and accurate computation of the collision-free IK. For the humanoid robot, Agile Justin (19 DoF), the collision-free IK is solved in less than $10\,\mathrm{ms}$ (on a single CPU core) and with an accuracy of $1\times10^{-3}\,\mathrm{m}$ and $1\times10^{-2}\,\mathrm{rad}$ based on a high-resolution world model generated from the robot's integrated 3D sensor. Our method massively outperforms a random multi-start baseline in a benchmark with the 19 DoF humanoid and challenging 3D environments. It requires ten times less training time than a supervised training method while achieving comparable results.
(https://dlr-alr.github.io/2023-humanoids-ik/)**

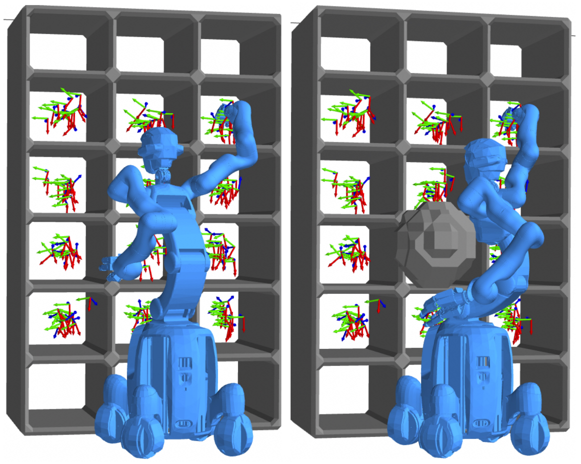


Fig. 1. DLR's Agile Justin [3] in a shelf environment. The frames for the IK problem were sampled randomly in the respective boxes of the shelf (left). On the right, the previous solution was blocked and made infeasible by placing an additional obstacle in the workspace, leading to a different collision-free solution. Details about the training datasets, networks, and videos can be found on the paper's website.

I. Introduction

A solution to inverse kinematics (IK) while avoiding collisions is fundamental for getting joint configurations in the most common robotic tasks, such as picking and placing objects. Still, it can also be used in the context of motion planning: For example, positioning a cup upright on a cluttered table requires motions with cartesian constraints at the end-effector. This joint problem of solving the IK and getting a collision-free trajectory is challenging, so it is often divided into two sub-problems [1]. First, the IK is solved to find the final configuration for grasping the object, then the trajectory from the initial configuration to the goal configuration is planned. Similarly, some problems require a path in which the frame of the end-effector is constrained at some intermediate steps. Solving the IK problems along the path and initializing the motion planner with the solutions can benefit these problems. Therefore, quickly computing a collision-free IK is crucial for real-time grasping and manipulation tasks.

In this paper, we deal with the problem of speeding up the IK with collision avoidance via learning for complex robots like DLR's humanoid robot Agile Justin with 19 degrees of freedom (DoF) as depicted in Fig. 1. As we will show, learning the inherently ambiguous inverse mapping from a frame of a robot's TCP (tool center point) to its joint configuration poses several challenges (other than, e.g., in speeding up motion planning in configuration space [2]). Learning an IK solution gets esp. hard when incorporating self-collision avoidance and avoiding collisions with obstacles in arbitrary environments (see Fig. 1, right).

This paper presents and compares two learning-based approaches to the IK problem: a supervised learning approach that relies on a separate data generation step and an unsupervised approach that does not need time-consuming data generation.

A. Related Work

There are many non-learning-based methods to solve the IK problem. They are often based on the Inverse Jacobian method [4, 5, 6]. A popular algorithm is TRAC-IK [7], combining a Newton-based algorithm with a Sequential Quadratic Programming (SQP) method. Running both methods in parallel and terminating if one succeeds improves speed and robustness.

Other optimization methods have been applied to the IK problem, too. Collinsm and Shen [8] use Particle Swarm Optimization (PSO) for snake-like robots with many degrees of freedom (DoF). Trutman et al. [9] describe the IK as a polynomial optimization problem, which they use to find a globally optimal solution for serial 7 DoF robots. Tringali and

[1]DLR Institute of Robotics & Mechatronics, Germany; [2]Deggendorf Institute of Technology, Germany; [3]Technical University of Munich, Germany

This work was partly funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy, within the projects SMiLE2gether (LABAY102).

Cocuzza [10] leverage the Inverse Jacobian method but use a randomized matrix to weight the pseudo inverse. This adaptation allows them to improve the convergence to a globally optimal IK solution. However, none of these methods take into consideration collisions with the environment.

Ferrentino et al. [11] uses dynamic programming to solve the IK with obstacles and make it available in ROS. Giamou et al. [12] formulate the IK problem as a distance-geometric problem, allowing them to use semidefinite programming methods to find a low-rank solution. While the approach is elegant and fast, it can only incorporate spherical obstacles. Zhao et al. [13, 14] introduce a modern and fast solver that combines Inverse Jacobian methods, SQP, and PSO. Their method can handle dynamic obstacles but is limited to spheres.

There are also many learning-based approaches for solving the IK problem more efficiently. One problem all those methods suffer from is the inherent ambiguity of the IK solution. Bocsi et al. [15] tackles ambiguous solutions by using structured prediction. Kim et al. [16] uses the graph structure of robot kinematics to learn the entire nullspace with a Graph Neural Network (GNN). However, the nullspace gets exponentially large with the DoFs of the robot, making it crucial that the learning of the mapping between forward and inverse kinematics is efficient [17]. To improve the speed and portability of IK methods, Zaidel et al. [18] introduce a neuromorphic approach that they apply to a 7 DoF robot arm.

All IK methods described so far consider an obstacle-free working environment. Lehner et al. [19] leverage transfer learning between similar robot kinematics in a single environment with obstacles. They use the network predictions to guide a Rapid Random Tree (RRT) motion planner. Lembono et al. [20] use Generative Adversarial Networks (GANs) to learn constrained robot configurations. They use the predictions of those networks as an initial guess for an optimization-based planner to warm-start the IK problem and as samples for an RRT motion planner. They consider the environment for their tasks, but for each new scene, an ensemble of GANs needs to be trained to counteract the mode collapse and produce valuable samples.

Until now, no learning-based approach to the IK problem incorporates collision avoidance for arbitrary, previously unseen environments. Moreover, for autonomous robots, the environment model is generated in real-time from sensor data and, hence, is a high-resolution, e.g., voxel-based model [21]. For those challenging worlds, no fast and efficient collision-free IK solver exists.

B. Contributions

We tackle the problem by formulating the IK with collision avoidance as an optimization problem similar to CHOMP [22] and use a combination of Jacobian-based projections and gradient descent to solve it numerically. We use the predictions of a neural network as warm-starts to speed up the optimizer. Those networks are trained with the Basis Point Set (BPS) [23] as encoding for the environment to incorporate collision avoidance. The BPS encoding has already been successfully used for planning robot motions in configuration space [2].

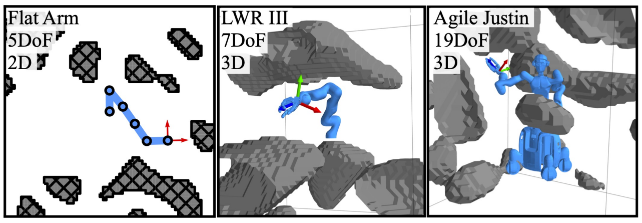


Fig. 2. The three robots used in the experiments in environments generated with Simplex Noise [24]. The Flat Arm in 2D helps to analyze and visualize the IK problem in detail. The LWR III and Agile Justin demonstrate the capabilities of our method for complex robotic systems.

Our main contributions are:

- A learning-based fast and accurate solver for IK with collision avoidance for complex previously unseen environments (for the 19 DoF humanoid Agile Justin on a high-resolution voxel grid an IK solution with an accuracy of $1 \times 10^{-3}\,\mathrm{m}$ and $1 \times 10^{-2}\,\mathrm{rad}$ in $5\,\mathrm{ms}$).
- A detailed analysis of the challenges in learning ambiguous IK with collision avoidance and the resulting network, including a twin-headed architecture, a singularity-free output representation, and boosting.
- The optimal solution to the IK problem varies not smoothly across the workspace. We show that two heads are enough for a network to predict the sharp switches between those regions of different modes.
- A benchmark of the supervised and the unsupervised learning approach shows a ten times faster training time for the latter and a more straightforward training procedure while outperforming the random baseline significantly.

II. Optimization-based Inverse Kinematics

A. Objective Terms

We formulate the IK problem, including avoiding collision with the environment, and self-collision, as an optimization problem. These hard constraints are taken into account as weighted terms in an overall objective function. Central to the problem formulation is the model of the robot. The forward kinematics maps from joint configurations $q \in \mathbb{R}^{N_{\text{DoF}}}$ to the link frames $\{F_i\}_{i=1}^{N_{\text{f}}} = f(q)$. Each frame F_i describes a full 6D pose $(p_i, R_i) \in \mathbb{R}^3 \times SO(3)$.

The equality constraint for the IK is that the distance between a specific frame in the chain F_i and a target frame $\bar{F}_i$ is zero. In the objective, this results in a translational part

$$U_{\text{P}}(q, \bar{F}_i) = \frac{1}{2} \|p_i(q) - \bar{p}_i\|^2 \tag{1}$$

and a rotational part

$$U_{\text{R}}(q, \bar{F}_i) = \frac{1}{2} (3 - \text{Trace}(\, R_i(q)\, \bar{R}_i^{-1}\,)). \tag{2}$$

The goal is to obtain a collision-free IK. The collision inequality constraint between the robot and the environment is also incorporated as a term in the objective using the following robot and environment models. We describe the robots geometry of each link F_i by a set of spheres $\boldsymbol{S}_i = \{x_{ik}, r_{ik}\}_{k=1}^{N_{si}}$ with centers and radii. For the world, we use a Signed Distance Field (SDF) $D(x)$, which gives the distance to the closest obstacle for each point x in the workspace. The collision cost is then given by the sum of all the collisions of the different body parts

$$U_{\mathrm{W}}(q) = \sum_{i=1}^{N_{\mathrm{f}}} \sum_{k=1}^{N_{si}} c\Big(D\big(F_i(q) \cdot x_{ik}\big) - r_{ik}\Big). \tag{3}$$

In addition to collisions with the world, complex robots must also account for self-collision. Again, the cost sums up all the penetrations between the different body pairs

$$U_{\mathrm{S}}(q) = \sum_{j>i}^{N_{\mathrm{f}},N_{\mathrm{f}}} \sum_{k,l}^{N_{si},N_{sj}} c\Big(\big\|F_i(q)\cdot x_{ik} - F_j(q)\cdot x_{jl}\big\| - r_{ik} - r_{jl}\Big). \tag{4}$$

The smooth clipping function c is introduced to transforms the inequality into an equality constraint which is then written as a additional cost term in the objective [1, 2]. It considers only the parts of the robot which are in collision by setting positive distances to zero. Thus, a collision-free solution has a cost of zero.

While the mapping from the joint configuration to the end-effector frame is unique, the same does not hold for the inverse mapping. For an over-actuated robot, infinitely many joint configurations can reach a given frame in the workspace. However, one is usually not interested in an arbitrary solution but one which satisfies additional criteria. We introduce an additional term to the objective, namely the closeness U_{L} to a default configuration $\bar{q}$:

$$U_{\mathrm{L}}(q) = \frac{1}{2} \sum_{i=1}^{N_{\mathrm{DoF}}} (q_i - \bar{q}_i)^2. \tag{5}$$

Minimizing U_{L} makes the mapping unique and ensures that the solutions are close to the default configuration, making motion planning to this configuration faster and easier.

In summary, in the overall objective U, one part is concerned with the frame at the end-effector U_{F} and one part accounts for the collisions and additional objectives U_{A}.

$$U = U_{\mathrm{F}} + U_{\mathrm{A}} \tag{6}$$
$$\text{with } U_{\mathrm{F}} = U_{\mathrm{P}} + \lambda_{\mathrm{R}} U_{\mathrm{R}},\ U_{\mathrm{A}} = \lambda_{\mathrm{W}} U_{\mathrm{W}} + \lambda_{\mathrm{S}} U_{\mathrm{S}} + \lambda_{\mathrm{L}} U_{\mathrm{L}}. \tag{7}$$

Note that the weighting factors can be normalized independent of the robot and environment and are mainly to ensure higher importance of the collision terms over the secondary objectives like length. With this formulation, the optimal configuration q^* and solution to the IK problem is the one with the lowest objective

$$q^* = \underset{q}{\operatorname{argmin}}\, U(q). \tag{8}$$

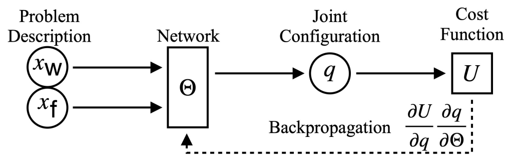


Fig. 3. The flow of information through the neural network. The IK problem is described by a world x_{w}, and a frame in the workspace x_{f} and the network should predict a collision-free joint configuration that satisfies the end-effector. The dotted line indicates the backpropagation during unsupervised training, where the network weights Θ are directly updated according to the gradient of the cost function U.

B. Solver with Nullspace Projection

While this formulation as an optimization problem is complete and (6) is used to train the unsupervised networks in Section III-C, it is often not efficient to solve this complex cost function jointly. To weaken the impact of competing terms in the objective function, we solve the IK problem in two steps.

First, we solve the pure IK with a projection step to ensure the constraints at the end-effector U_{P} and U_{R} are satisfied. This root search can be solved by iteratively applying the pseudo-inverse of the end-effector constraints:

$$\Delta p = [U_{\mathrm{P}}(q), U_{\mathrm{R}}(q)] \tag{9}$$
$$J = [\frac{\partial U_{\mathrm{P}}(q)}{\partial q}, \frac{\partial U_{\mathrm{R}}(q)}{\partial q}] \tag{10}$$
$$q_{i+1} = q_i + J^{\dagger} \Delta p \tag{11}$$

For the humanoid Agile Justin, the IK requirements in the real world are to be accurate below 10^{-3} m and 10^{-2} rad.

In the second step, we apply gradient descent with nullspace projection to satisfy the collision constraints and optimize the additional terms in U_{A}. Each gradient step is again projected on the IK manifold to ensure the constraints at the end-effector stay satisfied

$$q_{i+1} = q_i + (I - J^T (J^T)^{\dagger}) \frac{\partial U_{\mathrm{A}}(q)}{\partial q}. \tag{12}$$

These update steps push the configuration out of collision and closer to the default pose. While this approach is straight forward to implement and converges quickly for a given sample, it is susceptible to the initial guess. Especially for complex robots and environments, multiple samples are necessary until a feasible solution is found.

III. Learning the Inverse Kinematics

The idea is to mitigate the strong dependence on the initial guess by using the prediction of a neural network as a warm-start for the optimization-based IK solver. In this work, we compare two different learning approaches: First, a supervised learning approach that relies on training data generated by the solver described in Section II-B. Furthermore, we introduce an unsupervised regression approach, where the objective function given by (6) is directly used to update the network weights via backpropagation (see Fig. 3). We use the same overall architecture for the supervised and unsupervised networks to compare the approaches.

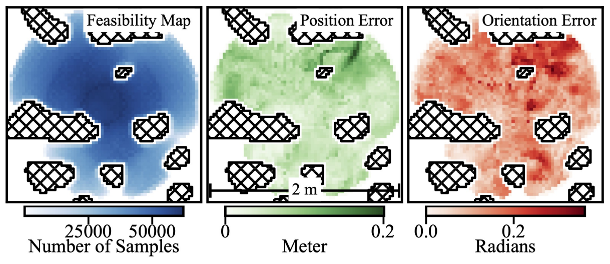


Fig. 4. Feasibility map (right/blue) for the 2D arm with 5 DoF for a specific environment. Depending on the robot's kinematics, not only the parts of the workspace with obstacles are not reachable but also areas behind obstacles. The overall number of feasible poses decreases towards the borders of the workspace. The maximal position error (center/green) and the maximal orientation error (left/red) highlight which regions are challenging for the network in more detail.

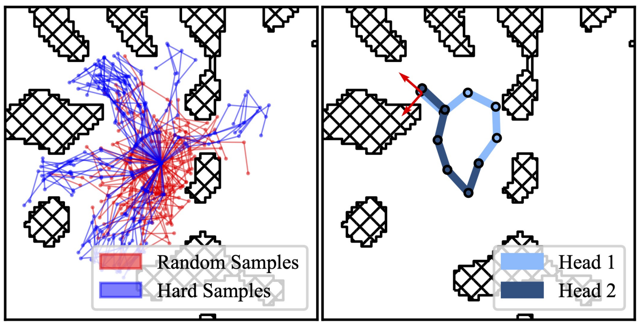


Fig. 5. In the left image, 50 random but feasible samples for the robot in the given environment are drawn in red, and in blue, 50 samples that were in the hard set after the training finished (see Section III-D.1). The challenging samples are more extended and fill the narrow passages in the world better than the random samples. In the right image, the predictions of the twin heads for a random sample are shown. While satisfying the end-effector, the two configurations show two distinct modes.

A. *Environment Representation*

As collision avoidance with the environment is a central aspect of the problem, the following section describes how to generate challenging training worlds and encode the scene to feed it into the networks. The worlds were generated using Simplex noise [24], as described by Tenhumberg et al. [2] for motion planning. By adjusting the noise frequency and the threshold, we can create diverse and challenging environments for the robots. Examples of the different worlds can be seen in Fig. 2.

To encode the environment for the networks, we use the BPS [23], which was already successfully used for robotic motion planning [2] between join configurations. The advantage of this representation over occupancy grids and point clouds is that it is more memory-efficient, computationally efficient, and inherently permutation invariant. The BPS representation can be understood as a subsampled SDF. Formally, the BPS is an arbitrary but fixed set of points in the workspace $B = \{b_i\}_{i=1}^{N_\mathrm{b}}$. The feature vector for the world x_W passed to the network consists of the distances to the closest point in the environment for all basis points. If a distance field D describes the environment, one can directly look up the feature vector

$$x_\mathrm{W} = [D(b_i), \ldots, D(b_{N_\mathrm{b}})]. \tag{13}$$

B. *Supervised Learning*

We use the algorithm described in Section II for the sample generation. Exhaustive multi-starts guarantee that a feasible solution is found, even in challenging scenes. The supervised training relies on consistent data, implying the labels are all globally optimal. Ensuring this requires a lot of computational resources. We use an efficient and generic cleaning method that uses the objective U and the current network to ensure all labels are close to the global optimum. This cleaning is explained and analyzed for motion planning between joint configurations in Tenhumberg et al. [2]. After the data generation, we use a standard Mean Squared Error (MSE) loss to train the network supervised on the ground truth labels.

C. *Unsupervised Learning*

Alternatively, as the objective function (6) holds all the necessary information to quantify a given configuration, it can be directly used as a loss function for training a network. Pandy et al. [25] introduced unsupervised regression networks for robotic motion planning. We adapt the idea and discuss the extensions needed in the context of IK.

For a given problem defined by a world x_W and a frame x_f, one can directly calculate the gradients of (6) with respect to the network weights Θ by using the chain rule:

$$\frac{\partial U}{\partial \Theta} = \frac{\partial U}{\partial q}\frac{\partial q}{\partial \Theta}. \tag{14}$$

In Fig. 3, the information flow through the network and the updates via backpropagation are shown. The huge advantage of this unsupervised approach is that no computationally expensive data generation step is needed as in supervised learning. Here, different worlds x_W and target frames x_f are sampled randomly and via backpropagation the resulting gradients can be directly computed.

D. *Learning and Network Architecture*

To analyze the IK problem in the whole workspace, we generated feasibility maps and error maps of the robots in the different scenes. Fig. 4 shows three maps: feasibility (blue), maximal position error (green), and maximal orientation error (red) for a given position in the workspace. The maps are generated by sampling the whole joint space and collect which euclidian targets were reached. Then the position and the orientation error for each feasible target is computed. Those maps can assess the network's performance over the whole workspace, and are far more detailed than random test sets which are commonly useed The following sub-sections discuss the insights of this detailed analysis which gave rise the our network and learning architecture.

1) Boosting: Fig. 4 shows that the challenging samples close to obstacles are underrepresented if sampled randomly. Random sampling tends to cluster in the central region and under-represent extreme positions which the robot can only

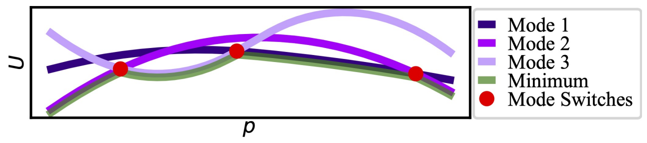


Fig. 6. The 1D scheme of the optimization-based IK problem shows the necessity of mode switches over the workspace to get a globally optimal solution. Different modes exist with varying costs U over the workspace. If the network should make the optimal prediction at each position p, it needs to switch between those modes. The transition regions are hard to represent for a neural network and can lead to significant errors (see Fig. 7).

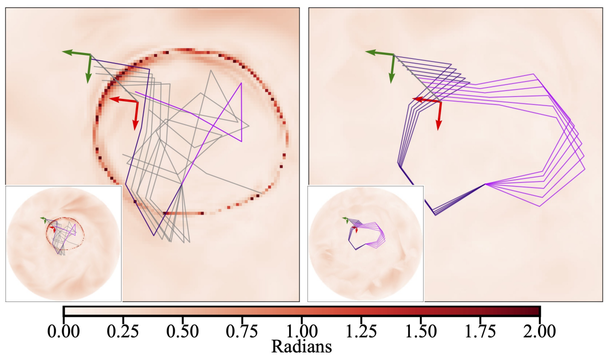


Fig. 7. This comparison is between a single-headed network (left) and a twin-headed network (right) for the IK prediction of a 5 DoF robot. The underlying red heat map indicates the worst orientation error across all 2π possible (discretized with 2880) goal orientations at each position. The distinct circular pattern (left) shows the transition region between two modes, where the prediction of the single-headed network breaks down. Moving the target frame at a specific orientation between those two regions leads to entirely wrong predictions. Each head of the twin model also has switching points, but as those two regions do not intersect, it can always predict valid and smoothly changing configurations (right). Visit also the website for additional visualizations of the mode switches.

reach fully extended. We introduce a boosting technique to overcome this and produce reasonable initial guesses in challenging situations. The idea is to have a set of challenging samples from which the training samples are chosen periodically. Similar to the method described by Tenhumberg et al. [2], we use the objective function U to over-represent complex samples. We define a sample q as hard if its cost U(q) is four times higher than the rolling mean.

The effect of boosting can be seen in Fig. 5. Here, 50 samples are shown in blue, which were in the hard set after the training ended. In contrast, in red, 50 randomly sampled configurations are shown. Those are more clustered towards the center of the world. This behavior can also be seen in Fig. 4(left), where only a tiny fraction of the samples in the configuration space reach the borders of the workspace.

2) Unit Vector Output: We use a singularity-free representation for the networks' output using 2D unit vectors instead of the joint values in radians. In the plane, the unit vector is a natural representation of an angle, which inherently corresponds to the directions vector in the workspace. This modification is especially relevant if the joint limits are $[-\pi, +\pi]$ or close to it. However, also in 3D and with stricter joint limits, the network can easier represent the underlying problem when choosing this encoding. The singularity-free representation omits the need for the network to internally represent a switch for joint values close to the singularity.

3) Twin-Headed Network: While the length cost U_L ensures that there is one optimal solution, there are still different modes over the workspace, and the network must switch between those modes to successfully predict optimal IK solutions for all possible targets. Fig. 6 visualizes the general concept of mode switches between a pair of modes to ensure an optimal solution.

However, the network's prediction can become entirely wrong in these transition regions. This behavior plus our solution is visualized in Fig. 7. For the 2D arm with 5 DoF, the transition regions can be seen in the heat map of the maximal orientation error. Where these transition regions lie depends even on the initial weights of the network, but each initialization has the same overall behavior. In each case, there are regions where the network needs to represent the switching between two modes. One can see the prediction breakdown by gradually moving the target frame from a position outside the ring (green) along a straight path to a position inside the circle (red). The network switches modes and cannot produce valuable predictions in this region.

By adding a second head to the network, which outputs a second prediction, one can overcome this problem. Each head of the twin model also has its own transition regions, but as those two areas do not intersect, one always has a valid and smooth prediction for the configuration. It is essential to add that two heads are enough, even for more complex settings with multiple modes. The two heads do not represent the modes directly but only mask the transition region between pairs of modes. We introduce an additional loss $U_\mathrm{H} = \|q_\mathrm{a} - q_\mathrm{b}\|$ between the two heads of the network to counteract mode collapse and gain a valuable second guess. Fig. 5 (right) highlights that maximizing the difference in configuration space between those heads produces fundamentally different solution modes. Besides allowing sharp switches between modes, this approach leads to the simplest version of a generative model, with much more accessible training and no need for network ensembles [20] to prevent mode collapse.

IV. Results

First, we demonstrate the effectiveness of our approach for the problem of collision-free IK in the case of a 2D robotic arm. Fig. 8 shows the steps of our IK procedure and compares it to using simple random sampling for generating initial guesses. Only five of the initial 20 configurations are feasible after both optimization steps. This ratio gets even worse for more complex robots in challenging 3D environments, which Table III analyzes in more detail.

The right-hand figure shows the same two-stage procedure for two network predictions. One trained without the world as a dedicated input and one which uses the BPS of the world to predict collision-free IK solutions. One can see clearly how close the two predictions are to the desired TCP. Furthermore, the prediction of the world-aware network is already in the correct narrow passage between the obstacles.

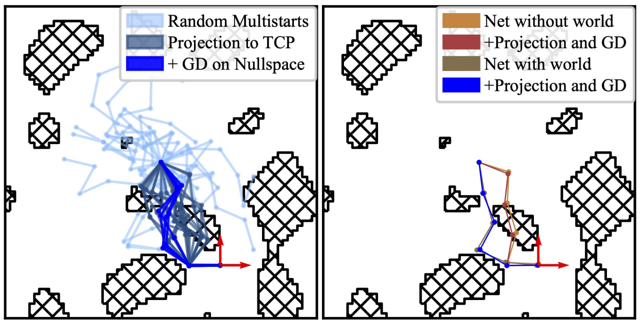


Fig. 8. Visualization of the collision-free IK solution process. In the **left image**, 20 initial random guesses of configurations (see legend for used colors) are used. These configurations are then projected onto the desired TCP (red coordinate system) using (11), ensuring that translational and rotational constraints are satisfied. Then, we compute the gradient of U_A and apply Gradient Descent inside the TCP-nullspace (12) to move the robot out of collision and closer to the default configuration. After these steps, only five feasible (i.e., collision-free) solutions remain. The **right image** showcases the same two-stage process when using the (single) prediction of an IK network but for two network variants. One network is trained without the world as an input, while the other network incorporates the BPS of the world to predict collision-free IK solutions. As the predictions of both networks are close to the desired TCP, the pure projection step on the TCP is not shown here. However, only the prediction of the world-aware network converges to a feasible solution while the other gets stuck in collision.

Using this prediction as an initial guess eliminates the need for multi-starts in most cases and leads to quicker convergence, as the optimizer only needs a few iterations for a feasible solution.

A. Experiments

This section shows the results for the supervised and unsupervised learning methods for robots with different complexity. All timings are measured on a computer with Intel i9-9820X @ 3.30 GHz with 32 GB RAM. All 16 cores are used for training, whereas online prediction runs only on a single core. To evaluate the networks, we use their prediction as a warm-start for the optimization-based solver described in Section II-B and compare convergence and feasibility rates.

Table II shows an ablation study for the learning and network architecture proposed in Section III-D. For the humanoid robot Agile Justin, the different networks were trained on 300 random worlds and evaluated on 20 unseen worlds drawn from the same distribution. The size of the test set was 100000 samples. Because not the network prediction directly is used on the robot but the converged result, we report the feasibility rate after 10 iterations of the solver. The table shows that each architectural component improves the performance of the network. In the extreme case where none of those methods are used, the feasibility rate is only 25%, while the final performance is close to 100%. Notably, the boosting does not improve the mean performance but significantly reduces the maximal error of the network's predictions. As this approach over-represents the complex samples with a large objective U, it is designed to improve those worst cases. This design is crucial if one uses those

TABLE I
TRAINING TIMES FOR THE DIFFERENT NETWORKS

Robots	Supervised Data Generation	Supervised Training	Unsupervised Training
Flat Arm	34.6 h	2.1 h	2.6 h
LWR III	71.3 h	2.7 h	3.0 h
Agile Justin	95.4 h	5.4 h	6.9 h

TABLE II
ABLATION STUDY OF THE NETWORK PREDICTION FOR AGILE JUSTIN

Training w. Boosting	Twin-Headed Network	Unit Vector Output	Feasibility
Yes	Yes	Yes	0.986
Yes	Yes	No	0.871
Yes	No	Yes	0.695
Yes	No	No	0.596
No	Yes	Yes	0.781
No	Yes	No	0.741
No	No	Yes	0.569
No	No	No	0.248

network predictions as a warm-start for an optimization-based solver in challenging scenes: Long searches with many multi-starts slow down the numerical solver for those cases.

The results of comparing the supervised and unsupervised network against a randomly sampled initial guess are summarized in Table III. This evaluation was performed for three robots: A 2D Arm with 5 DoF, the LWR III with 7 DoF, and Agile Justin with 19 DoF (see Fig. 2). In 3D, we used a shelf environment like depicted in Fig. 1. Here 10000 target frames were randomly sampled in the respective boxes in the shelf. The overall orientation of the target frame was aligned with the shelf, and noise was added to ensure feasible yet challenging samples. The shelf environment is closer to a real-world setting and has notably different attributes than the random worlds the networks were trained on.

Table III shows that the average feasibility rate of the initial guesses from the networks outperforms the random baseline significantly for a single initial guess (denoted as (1)). Furthermore, the average number of iterations to converge is also decreased. The overall speed advantage can be seen directly from the necessary iterations difference. For the humanoid robot Agile Justin (19 DoF), the computation time for a single iteration is 0.5 ms on our testing machine. This leads to an overall solve time of under 10ms for the collision-free IK in unseen environments. The learned warm-starts outperform the random multi-starts in solving time, and the length cost (5) is reduced. These solutions are often more convenient and easier to integrate into larger motion planning tasks than random solutions.

Besides the improvement of the learning-based approaches over the random multi-start, it can also be seen that supervised and unsupervised training perform similarly well. Overall, this gives an advantage to the unsupervised method, as it requires far less time to train as no prior data generation and data cleaning [2] is needed as Table I shows.

TABLE III

FEASIBILITY AND CONVERGENCE FOR THE DIFFERENT SAMPLING MODES FOR THE WARM-START OF THE IK SOLVER

Robots	DoF	Initial Guess	Avg. Mulit-Starts [#]	Feasibility (1) [%]	Avg. Iterations [#]	Avg. Length Cost U_L [rad]
Flat Arm Random World	5	Random	13.27 ± 5.41	19.7	12.87 ± 3.78	4.19 ± 0.75
		Supervised	3.64 ± 1.29	81.3	9.93 ± 3.67	3.48 ± 0.71
		Unsupervised	3.35 ± 1.37	83.4	8.53 ± 3.29	3.41 ± 0.69
LWR III Shelf World	7	Random	17.57 ± 4.53	14.4	15.69 ± 2.89	3.36 ± 0.68
		Supervised	2.97 ± 0.61	88.7	9.31 ± 3.78	2.91 ± 0.70
		Unsupervised	3.06 ± 0.55	92.6	8.76 ± 4.01	2.85 ± 0.65
Agile Justin Shelf World	19	Random	24.52 ± 7.18	8.3	13.88 ± 3.93	6.56 ± 0.93
		Supervised	4.41 ± 0.94	88.9	6.91 ± 3.66	4.72 ± 0.41
		Unsupervised	4.29 ± 0.97	87.6	7.25 ± 3.61	4.10 ± 0.53

B. Real-World Experiment on the Humanoid Agile Justin

We present real-world results on the humanoid Agile Justin to show the need for collision-free IK. Fig. 9 shows two table scenes; the robot should move the right TCP to the same position in both cases, first without obstacle and then with an additional obstruction. The rendered images in the bottom row show the self-acquired high-resolution voxel model [21]. The optimal solution to the IK for the simple scene does collide with the additional obstacle. The whole arm is stuck in the box on the table, and using this solution as a warm-start for our solver does not converge to a collision-free solution. However, using the neural network's prediction as an initial guess produces the solution shown on the right. The BPS representation and the proposed training scheme were robust enough to generalize to high-resolution voxel models collected by Agile Justin's depth camera [21], even if the training was only on random simplex worlds.

V. CONCLUSIONS AND FUTURE WORK

We introduced an unsupervised training method for learning the IK with collision avoidance. It works even for a humanoid robot with 19 DoF in challenging and diverse environments sensed with its integrated 3D sensor. An IK solution with an accuracy of $1 \times 10^{-3}\,\mathrm{m}$ and $1 \times 10^{-2}\,\mathrm{rad}$ is computed in only $10\,\mathrm{ms}$ on a single CPU core. Our method trains ten times faster than supervised training by avoiding the generation of an exhaustive training data set. It massively outperforms a multi-start baseline, as we showed in an elaborate benchmark with the humanoid and simpler robots in challenging environments. Based on a detailed analysis of the IK problem with collision avoidance, we derived our network and learning architecture with boosting to enable rare-case performance and dual-heads to handle the necessary switching between different configuration modes. In an ablation study, the relevance of this architecture is demonstrated.

Separating the task of grasping a specific object in a given scene into the subtasks of finding a stable grasp, getting the end configuration via IK, which allows this grasp, and then planning from a start point to that configuration is not always possible. Future work will integrate the IK tighter into the related grasping and path-planning problems. Ideally, grasping a specific object in a given scene must be solved jointly, as this guarantees the feasibility of the complete task and allows us to find globally optimal solutions.

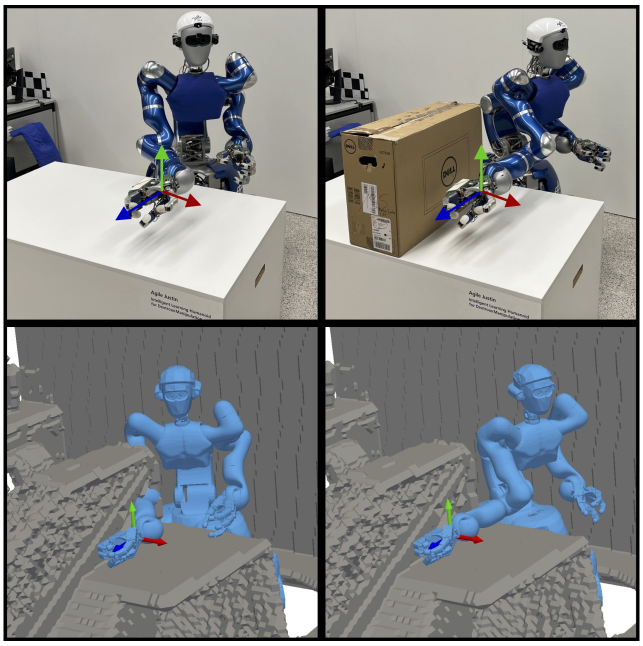


Fig. 9. Difference between standard IK (left) and collision-free IK (right) for the humanoid robot Agile Justin in a real table scene. The rendered images show the robots' self-acquired high-resolution voxel model [21] of the scene. This conservative occupancy map was encoded with BPS and used as input for the neural network. While only trained on random worlds, its prediction for this unseen world converges to the collision-free solution on the right.

REFERENCES

[1] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[2] J. Tenhumberg, D. Burschka, and B. Bäuml, "Speeding Up Optimization-based Motion Planning through Deep Learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[3] B. Bäuml *et al.*, "Agile Justin: An upgraded member of DLR's family of lightweight and torque controlled humanoids," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, 5 2014, pp. 2562–2563.

[4] N. Sukavanam and R. Balasubramanian, "An Optimization Approach to Solve the Inverse Kinematics of Redundant Manipulator," *International Journal of Information And Systems Sciences*, vol. 6, pp. 414–423, 2011.

[5] T. Sugihara, "Solvability-Unconcerned Inverse Kinematics by the Levenberg–Marquardt Method," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 984–991, 10 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5784347/

[6] A. Colome and C. Torras, "Closed-Loop Inverse Kinematics for Redundant Robots: Comparative Assessment and Two Enhancements," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 944–955, 4 2015. [Online]. Available: http://ieeexplore.ieee.org/document/6832645/

[7] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 11 2015, pp. 928–935. [Online]. Available: http://ieeexplore.ieee.org/document/7363472/

[8] T. J. Collinsm and W.-M. Shen, "Particle Swarm Optimization for high-DOF inverse kinematics," in *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, 4 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/7942651/

[9] P. Trutman, M. S. E. Din, D. Henrion, and T. Pajdla, "Globally Optimal Solution to Inverse Kinematics of 7DOF Serial Manipulator," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6012–6019, 7 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9745328/

[10] A. Tringali and S. Cocuzza, "Globally Optimal Inverse Kinematics Method for a Redundant Robot Manipulator with Linear and Nonlinear Constraints," *Robotics*, vol. 9, no. 3, p. 61, 7 2020. [Online]. Available: https://www.mdpi.com/2218-6581/9/3/61

[11] E. Ferrentino, F. Salvioli, and P. Chiacchio, "Globally Optimal Redundancy Resolution with Dynamic Programming for Robot Planning: A ROS Implementation," *Robotics*, vol. 10, no. 1, p. 42, 3 2021. [Online]. Available: https://www.mdpi.com/2218-6581/10/1/42

[12] M. Giamou *et al.*, "Convex Iteration for Distance-Geometric Inverse Kinematics," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1952–1959, 4 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9677911/

[13] L. Zhao, J. Zhao, H. Liu, and D. Manocha, "Collision-Free Kinematics for Redundant Manipulators in Dynamic Scenes using Optimal Reciprocal Velocity Obstacles," *ArXiv*, 11 2018. [Online]. Available: http://arxiv.org/abs/1811.00600

[14] L. Zhao *et al.*, "Collision-Free Kinematics for Hyper-Redundant Manipulators in Dynamic Scenes using Optimal Velocity Obstacles," *International Journal of Advanced Robotic Systems*, vol. 18, no. 1, p. 172988142199614, 1 2021. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1729881421996148

[15] B. Bocsi *et al.*, "Learning Inverse Kinematics with Structured Prediction," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 9 2011, pp. 698–703. [Online]. Available: http://ieeexplore.ieee.org/document/6094666/

[16] J. T. Kim, J. Park, S. Choi, and S. Ha, "Learning Robot Structure and Motion Embeddings using Graph Neural Networks," *arXiv preprint arXiv:2109.07543*, 9 2021. [Online]. Available: http://arxiv.org/abs/2109.07543

[17] D. Kubus, R. Rayyes, and J. J. Steil, "Learning Forward and Inverse Kinematics Maps Efficiently," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 10 2018, pp. 5133–5140. [Online]. Available: https://ieeexplore.ieee.org/document/8593833/

[18] Y. Zaidel *et al.*, "Neuromorphic NEF-Based Inverse Kinematics and PID Control," *Frontiers in Neurorobotics*, vol. 15, 2 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2021.631159/full

[19] P. Lehner, M. A. Roa, and A. Albu-Schaffer, "Kinematic Transfer Learning of Sampling Distributions for Manipulator Motion Planning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2022, pp. 7211–7217.

[20] T. S. Lembono, E. Pignat, J. Jankowski, and S. Calinon, "Learning Constrained Distributions of Robot Configurations With Generative Adversarial Network," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4233–4240, 4 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9385935/

[21] R. Wagner, U. Frese, and B. Bauml, "3D modeling, distance and gradient computation for motion planning: A direct GPGPU approach," in *2013 IEEE International Conference on Robotics and Automation*, no. Iii. IEEE, 5 2013, pp. 3586–3592.

[22] M. Zucker *et al.*, "CHOMP: Covariant Hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 8 2013.

[23] S. Prokudin, C. Lassner, and J. Romero, "Efficient Learning on Point Clouds with Basis Point Sets," in *International Conference on Computer Vision (ICCV)*, 8 2019, pp. 4332–4341.

[24] K. Perlin, "Noise Hardware," *Real-Time Shading SIGGRAPH Course Notes*, vol. 6, 2001.

[25] M. Pandy, D. Lenton, and R. Clark, "Unsupervised Path Regression Networks," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 9 2021, pp. 1413–1420. [Online]. Available: https://ieeexplore.ieee.org/document/9636818/