Crossmodal learning for dexterous manipulation

Dissertation

zur Erlangung des akademischen Grades

Dr. rer. nat.

an der Fakultät
für Mathematik, Informatik und Naturwissenschaften
der Universität Hamburg

eingereicht

2024

beim Fachbereich Informatik

von

Philipp Sebastian Ruppel

Erstgutachter:
Prof. Dr. Jianwei Zhang

Zweitgutachter:
Prof. Dr. Sören Laue

Vorsitzender der Prüfungskommission:
Prof. Dr. Stefan Wermter

Datum der Disputation:
26.6.2025

Zusammenfassung

In dieser Dissertation werden Ansätze zum Erlernen von Objektmanipulationsfähigkeiten durch Roboter unter Ausnutzung von Synergien zwischen verschiedenen Sensormodalitäten und physikalischen Gesetzmäßigkeiten entwickelt. Beim Trainieren von künstlichen neuronalen Netzen in simulierten Umgebungen können durch Erweiterung der Robotersteuerung um Vorhersage von Kontaktkräften und -punkten lokale Minima überwunden werden. Hierdurch können gewisse Manipulationsaufgaben, für die konventionelle bestärkende Lernverfahren auf zufälliges Ausprobieren zurückgreifen müssten, direkt als einstufige Ausgleichsprobleme gelöst werden. Für multimodales Lernen in der echten Welt werden neuartige taktile Matrixsensoren entwickelt. Diese können in dreidimensional gewölbten Formen und komplett aus hochelastischen Materialien gefertigt werden. Auf Basis dieser Sensoren werden taktile Sensorhandschuhe hergestellt. Es werden Verfahren entwickelt, durch welche Roboter Objektmanipulationsaufgaben aus wenigen Demonstrationen mit menschlichen Händen lernen können. Aus den Datenaufzeichnungen werden dreidimensionale multimodale Rekonstruktionen berechnet. Diese werden zum Trainieren künstlicher neuronaler Netze verwendet. Durch Einbeziehung taktiler Messwerte können nicht nur Bewegungen sondern auch Kontaktkräfte direkt aus Demonstrationsdaten gelernt werden. Zur Anpassung an neue Situationen mit veränderten Objektanordnungen wird der Lernvorgang teilweise als Entrauschproblem formuliert. Objektinteraktion wird ebenfalls aus den Datenaufzeichnungen gelernt. Durch überspezifizierte multimodale Vorhersage und physikalische Konsistenzbedingungen zur Laufzeit können Entrauschproblem und Objektdynamik zu einem einstufigen überwachten Lernproblem zusammengefasst werden. Zum gemeinsamen Anlernen und Ausführen wird eine integrierte Roboterarbeitszelle entwickelt. Diese ist mit zwei Endeffektoren und einem Multikamerasystem ausgestattet. Gelernte Objektmanipulationsfähigkeiten werden auf humanoiden Hand-Arm-Systemen sowie dem neu entwickelten Roboter erfolgreich ausgeführt.

Abstract

This dissertation presents approaches for robot learning of object manipulation tasks using synergies between multiple sensor modalities and physical laws. When training artificial neural networks in simulated environments, generating contact point and contact force predictions in addition to robot commands can help overcome local minima. This allows some manipulation tasks, for which conventional reinforcement learning algorithms would have to rely on exploration through random trial and error, to be solved directly through gradient-based optimization. In order to facilitate multimodal robot learning and teaching in the real world, novel tactile matrix sensors are developed. These can be made in three-dimensional multi-curved shapes and produced completely from highly stretchable materials. The sensors are integrated into sensor gloves. Methods are developed for teaching robots through small numbers of real-world demonstrations with human hands. Data recordings are pre-processed into three-dimensional multimodal reconstructions. These are used to train artificial neural networks. Integration of tactile measurements enables teaching not only motions but also forces through demonstration. Adaptation to new situations with previously unseen object arrangements is modeled as a denoising problem. Human demonstration data is also used to learn object dynamics. Through overspecified multimodal predictions and physical consistency constraints at runtime, denoising and object dynamics can be combined into a single-level supervised learning problem. An integrated workcell is developed that supports both teaching and robot execution. The system is equipped with two end-effectors and a multi-camera system. Learned behaviors are successfully executed on the newly developed robot and on humanoid hand-arm systems.

Contents

1. Introduction

Humans make use of multiple different sensor modalities when manipulating objects with their hands. This includes vision for identifying and locating objects, proprioception for sensing and controlling body and hand motions, and touch for regulating contact forces and relative positioning under occlusion. These are not only measuring redundant copies of the same state information, but can complement each other.

Before grasping an object, the shape, color, texture and pose may be observed visually, but can not be felt haptically. While grasping it, the object may be fully or partially occluded, but can be perceived through the tactile modality. Even if the hand and the object can be seen, the observed positions could often correspond to an infinite number of different forces, such as opposing grasping forces. Conversely, when controlling only the position of a robot end-effector and moving the robot against a hard surface or object, the resulting force can be unknown. Assuming an inelastic rigid-body model, the same poses could correspond to any force magnitude from zero to infinity. If a robot with non-zero finite stiffness would perfectly imitate the shape of a human hand's skin while attempting to grasp an object from above under impedance control, the mechanical force could be zero and the grasp could fail even if the robot is perfectly replicating the human hand pose for exactly the same object pose.

Likewise, known forces do not imply known positions. When sensing no contact force, the manipulator could be almost touching the object or be far away. For non-zero contact forces, the object could often still be in an infinite number of different poses. When controlling forces without considering positions, the position could be undefined or subject to ever increasing drift.

In simple cases, it can be tempting to overlook such questions. When picking an object with a two-finger gripper or a humanoid power grasp, one could simply set the robot to position control, deliberately miscalibrate the robot with small position biases and scaling factors to push into the object during grasping and hope for a finite control stiffness to generate a suitable grasping force. However, these hardcoded parameters would be task-dependent and not necessarily be suitable for other tasks, objects and robots. For dexterous manipulation beyond grasping, such approaches can generally be assumed to fail. If a finger is not only applying force along a single direction, even a perfectly tuned position offset for one grasp can be unsuitable when a different force is required along a different direction. Furthermore, when using a high control stiffness for accurate position control, small position errors could always lead to high force errors.

Multimodality can also be of fundamental importance during the learning process itself. When already close to or touching an object, current learning methods, such as deep reinforcement learning or gradient-based policy optimization with differentiable physics, can quickly explore the object's response to small incremental motions. However, if the object is placed further away from the hand, it does not react to small hand motions. Moreover, if the desired direction of motion for the object is close to the direction towards the hand, most, and initially all, contact events during random exploration would push the object away from the hand, and it would be reasonable for a unimodal agent to deduce that touching an object would be counter-productive for grasping it.

The complimentary nature of the different sensor modalities involved in human object manipulation abilities, especially vision, proprioception and touch, suggests that multimodality could be of central importance for robot learning of dexterous manipulation tasks.

1.1. Research questions

1. Learning for object manipulation can be difficult, even in simulations where complete mathematical descriptions of environments and task goals already exist. Could giving an artificial neural network additional senses to access or manipulate the inner workings of a simulator it inhabits simplify learning?

2. If the simulator already stores all relevent information, it should be possible to treat the entire learning and simulation problem as a single optimization. How should the problem be formulated and how should it be solved?

3. Humans may want to delegate repetitive work to robots. However, programming simulators requires different skills than most physical manipulation tasks. Some applications could therefore benefit from minimizing differences between robot programming and working with human hands directly. Could working with human hands be used as a multimodal programming language for robots?

4. Conditions such as initial object poses may not always be the same during teaching and execution. How could behavior learned from few demonstrations be generalized to new situations?

5. Hands and grippers interact with objects through contact forces. Would it be possible to teach not only motions but also contact forces with human hands?

6. Teaching contact forces may require sensors around human hands while interference with human dexterity should be minimized. Soft skin-like rubber gloves might be ideal. Could thin tactile matrix sensors be created in organic multi-curved shapes? Could the sensors, electrodes and conductors be entirely made from rubber-like stretchable materials?

7. Hardware setup for robotic manipulation can require significant time and effort. Could facilities for teaching and robot execution be integrated into a single self-contained desktop device?

1.2. Thesis structure

Learning for robotic manipulation is first investigated in simulation (chapter 2). Initial experiments with differential physics examine opportunities and challenges of using traditional policy gradients. In order to address the observed shortcomings, a new contact-based multi-modal learning method is developed. The work also reveals opportunities to further simplify the computations, which are addressed through the development of a suitable automatic differentiation framework (chapter 3). Chapter 4 investigates tactile sensing for real-world multimodal learning and develops double-curved sensor matrices, a new type of highly stretchable sensor skin and sensor gloves for teaching by demonstration. These are combined with multi-camera systems and the data is processed to obtain multi-modal reconstructions of human demonstrations (chapter 5). Facilities for teaching and robot execution are combined into a single workcell (chapter 6). A multimodal learning framework for teaching by human demonstration is developed (chapter 7) and combined with components from previous chapters. The system is tested in multiple different robot experiments.

2. Contact-based neural policy optimization in simulation

When searching for robot behavior automatically to solve manipulation tasks, contact dynamics can introduce additional local minima and plateaus. Before touching an object, the object may not respond to small exploratory robot motions at all. Even when touching an object, the robot may only be able to push the object along specific directions, which might not include any of the desired responses.

This can even be the case for seemingly easy problems such as grasping. A cube-shaped object is placed on a table and a robot should grasp and lift the object. A reward function measures the distance towards a goal pose, rewarding the agent if the position and orientation approach the goal and further punishing the agent if the object is moved or turned away from it. Close to the start state, small robot motions would not result in any object motions or changes in the reward function. Most larger robot motions would likely push the object away from the desired goal position or goal orientation. If the reward function would only consider positions but ignore orientations, a larger random motion could eventually flip the object, thus moving the center of the object upwards, and generate a reward. However, if the reward function considers both position and orientation, such cases may also lead to decreased reward.

Autonomous policy search can, however, be surprisingly successful for favorably designed tasks. If the robot already touches an object, even small hand and finger motions can immediately move the object. In some cases, when learning in simulation, and if even infinitesimally small robot motions can lead to increased rewards, random exploration may not even be necessary at all and the learning problem can be solved directly through gradient-based optimization with differentiable physics.

In other cases, it may be possible to accelerate the learning through human demonstration, curriculum learning or reward shaping. A human programmer develops a set of instructions, encoded as additional rewards or punishments, that guide the agent towards solving the task. However, the programmer then has to dictate a correct approach instead of letting the learning algorithm find the solution. If a human wants to teach a task by demonstration, it may be preferable for the human teacher to simply demonstrate the desired behavior with their own hands in the real world and let the robot learn the task completely from real-world demonstration data (see chapter 7). In turn, if a human programs a simulator and a reward function for autonomous policy discovery, programming of shaped rewards or curricula, or recording of additional demonstration data, may be redundant. It may even be counter-productive, if the motivation is to let the computer find a solution autonomously, or if the redundant specification introduces contradictions. In both cases, a human programmer or teacher would essentially have to specify the task twice. Instead, given a simulator and a reward function, it should be possible to compute a solution directly from the model itself and general physically-based abstractions.

Contact response can be notoriously hard to simulate. Instead of naively computing contact forces from positions and then integrating the forces, state-of-the-art physics engines typically solve contact response as optimization problems. Some formulations introduce additional slack variables for contact forces or impulses. Similar approaches can also be used for motion planning. Instead of only solving for a sequence of robot poses, additional contact variables are introduced, which allow the planner to explicitly reason about contact states and find motion plans for contact-rich tasks while overcoming local minima, plateaus and discontinuities.

It would be desirable to use similar approaches for accelerating robot learning. However, in contrast to motion planning or optimizing a single trajectory, learned behavior should adapt to observations. This can, for example, include object positions or initial robot states, or sensor data encoding such information. During training, the conditions would typically be randomized. This implies that the values of the contact variables would also have to be different during each training step. When optimizing contact information as free variables together with the policy, previous solutions for the contact variables would always immediately be invalidated by different randomized conditions during the next training step.

2.1. Related Work

A large number of works apply reinforcement learning to robotics-inspired scenarios [210, 142, 211, 151, 39, 116]. The training is often performed in simulation, which can be faster, cheaper and safer than autonomous learning on a real robot. During reinforcement learning in simulation, the robot interacts with a virtual environment and, by learning from experience and through exploration by trial and error, should maximize a reward function. Such approaches have been especially successful in cases where meaningful rewards are available immediately. A policy for steering a simulated car along a road can be learned by applying continuously increasing punishments the further the car is away from the center of the road and certain grasping problems can be learned by gradually luring a robot towards a specific object and then towards a target with continuously increasing hardcoded rewards [210]. If a cube is placed inside a robot hand with the palm facing upwards, such that a single finger, wrist or thumb motion can easily affect the orientation of the cube, it is possible to automatically explore these interactions and train an artificial neural network to turn the cube. However, the experiment required 50 hours of training time on 6144 CPU cores and 8 GPUs, corresponding to more than 100 years of simulated experience [142]. The approach could be extended by programming a simulated environment that gradually increases task difficulty and through further parallelization. On a cluster of 12800 CPU cores, a similar cube reorientation task could be learned in approximately one day. After retraining for several months on 29440 CPU cores, the system could also learn motions to turn the top-facing side of a Rubik's Cube when triggered by a hard-coded higher-level policy [211].

State-of-the-art physics engines typically solve contact dynamics as constrained optimization problems, which can, in some but not all formulations, involve solving for forces or impulses at contacts and joints [132, 16, 184, 212]. Some motion planning methods solve for contact variables across multiple time steps in addition to motion trajectories [129, 128, 185]. These can be applied to find physically plausible solutions to robotic manipulation problems [128, 185]. In physics simulation, the contacts are typically determined by a separate collision detection step [132, 16, 212, 64]. The collision response solver may only have to solve for one modality, such as positions [132, 16] or impulses [212]. Contact-based motion planning can involve solving for contact points, contact forces and motions simultaneously [16, 185].

In some cases, a need for random exploration during robot control or learning can be avoided or reduced through differentiable physics. Gradient information can be propagated directly through the physics simulator by extending the simulation engine itself or through automatic differentiation [184, 162, 7, 197, 150, 93, 3, 86, 32, 31, 77, 11, 84, 85, 213, 214]. Gradient information can be used during learning through direct policy optimization, solving the entire learning problem as a single optimization problem, which can be especially prone to suffer from local minima, or by integrating analytical gradients into existing reinforcement learning algorithms [214].

Main themes of this thesis might in part also be reminiscent of the transition to physically-based rendering in 3D graphics. Mathematically simple but physically incorrect shading equations [68, 149] can be replaced using physical attributes such as conductivity and roughness, surface structure and physical laws including conservation of energy. While originally intractable, the behavior can be estimated using closed-form approximations [40, 78]. Tone mapping [2] and linear gamma [71] allows simulating color perception via light energy. AI raytracing estimates illumination by interpolating between small numbers of physically correct but computationally expensive samples using artificial neural networks [12, 182].

2.2. Overview

This chapter proposes to solve robot learning problems with contact dynamics in a single optimization by learning contact variables together with robot control. An artificial neural network is trained to generate not only robot commands but also contact points and contact forces. The learning process simultaneously optimizes the network weights for task goals and physical consistency. Instead of having to treat the environment and underlying physics as a black box, the neural network can essentially tap directly into the physics engine. This can allow for immediate learning progress even before touching an object. The learning problem is solved directly through gradient-based policy optimization.

A first preliminary experiment investigates the efficacy of traditional differentiable physics or simply propagating control gradients through a simulator. Simply replacing a hard contact condition with a smooth falloff enables efficient solutions to a two-fingered toy problem but fails for dexterous manipulation with a multi-fingered robot hand. In order to overcome the observed shortcomings, a different approach is introduced, which treats contact points and contact forces as additional control variables. Contact variables are connected to a neural policy network, which also controls the robot joints. The methods are validated in multiple dexterous manipulation scenarios and different solution methods to the policy optimization problem are compared.

2.3. Task definitions

The methods presented in this chapter are studied in the context of 8 different manipulation tasks and three different robot setups (see figure 2.1 and table 2.1).

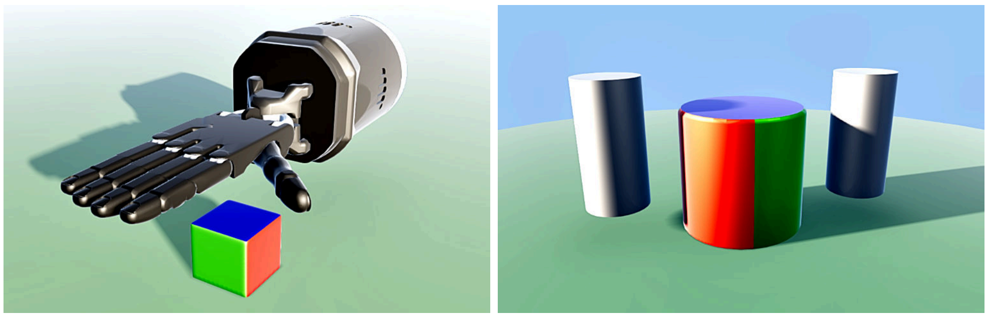


Figure 2.1.: Shadow C6 hand with a cube-shaped object (left), cylinderical object with two cylindrical fingers (right).

Task	Object	Goal	Robot	Joints	DoF
1	Cube	Turn	Two fingers	4	4
2	Cylinder	Turn	Two fingers	4	4
3	Cube	Lift	UR5 arm + Shadow C6 hand	30	26
4	Cube	Push horizontally	UR5 arm + Shadow C6 hand	30	26
5	Cube	Turn horizontally	Shadow C6 hand	24	20
6	Cube	Flip backwards	Shadow C6 hand	24	20
7	Cube	Flip left	Shadow C6 hand	24	20
8	Cylinder	Turn horizontally	Shadow C6 hand, fixed	24	20

Table 2.1.: Overview of manipulation tasks investigated in this chapter.

Each task is defined by a loss function L_T (negative reward). This task loss integrates motions from start state to termination state as a 6D vector. For the lifting and pushing tasks, the task loss is defined as the distance between object pose during the termination state and a goal pose. For the turning tasks, the squared difference between average angular velocity and an angular goal velocity is computed. In tasks 1 to 7, the objects can move freely. During task 8, the cylinder is mounted on a vertical revolute joint and the task loss additionally minimizes forces and torques acting against the mounting joint of the object. Object positions are randomized along the horizontal X and Y axes. The initial object positions follow a Gaussian distribution. Training episodes run for a finite number of steps. The robots are controlled in joint space. Proximal and distal joints on the fingers of the humanoid hand are coupled.

2.4. Differentiable physics engine

As the basis for investigations in the following sections, a differentiable physics engine is developed. The main design goals are easy modification and experimentation, and the ability to directly interface force computations with multimodal learning. The baseline contact model therefore uses a force-based approach and is subsequently adapted and replaced. Normal forces are computed from penetration depth according to material stiffness via Hooke's law. Tangential forces are computed by projecting relative velocities at contact points and normal forces into friction cones. Forces are integrated according to Newtonian dynamics via semi-implicit Euler integration. The shapes of the objects and robot parts are represented as sets of primitives and convex meshes. Collisions are detected using GJK [64] and EPA [17]. Robot, object and world models are loaded from URDF files.

2.5. Learning with smooth contacts

A multilayer perceptron with one hidden layer and 256 hidden neurons receives object and joint positions as input and generates joint velocity commands as output. The policy network is trained through gradient-based policy optimization with either stochastic gradient descent or sequential least squares and conjugate gradients (see section 2.9). With the baseline contact model, in all 8 tasks described above, the object simply lays on the table and no learning progress can be observed, since without touching the object, the policy gradient is zero.

The normal force computation is modified with a smooth non-zero falloff. In the baseline contact model, the normal force F_C depends linearly on positive penetration depth, which is equal to a ramp or "ReLU" [57] function over signed distance d_S. This "ReLU" [57] function is replaced with a smooth and continuously increasing non-zero "SoftPlus" [45] function, scaled by material stiffness D_M and a smoothness parameter C_S. For small C_S, the modified normal force function approaches the behavior of the baseline model.

$$F_C = D_M \, C_S \, ln(1 + e^{-d_S \frac{1}{C_S}}) \tag{2.1}$$

Learning the manipulation tasks defined in section 2.3 is attempted again with the modified contact model. For tasks 1 and 2, learning converges within a few minutes and the fingers are able to turn the object (see figure 2.2). For tasks 3 to 8, the learning fails. Instead of approaching and manipulating the object, the hand slowly moves away. Since the hand starts above the object (see figure 2.1), even with the smooth contact model, the fingers are unable to push or pull the object into the correct direction. The smoothness term pushes the object slightly into the table, resulting in a gradient away from the object.

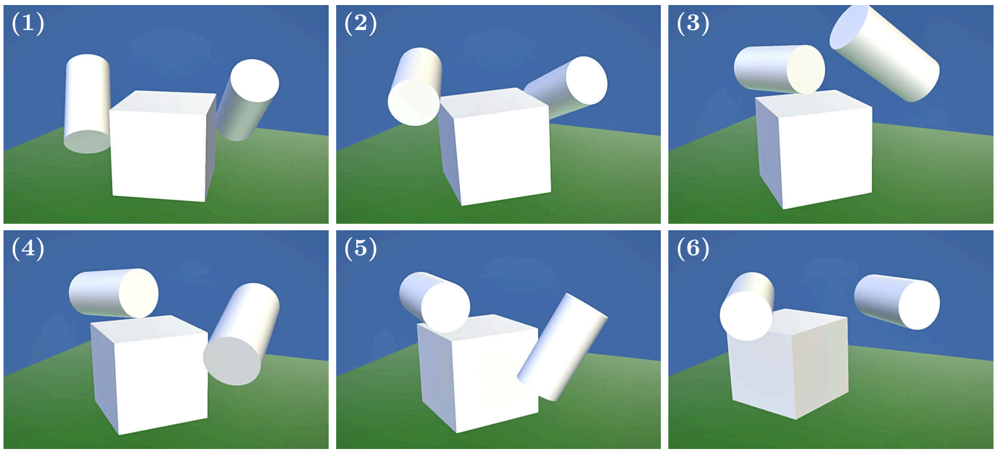


Figure 2.2.: A neural policy network is trained with differentiable physics and a smooth contact model to turn a cube using two cylindrical fingers. Different states from the same training episode are shown with increasing time from left to right and top to bottom (1 to 6).

2.6. Contact-based policy optimization

Contact force computation is replaced with neural predictions. The policy network N_P maps neural network weights W_P and an observation vector O_t for each time step t to an output vector consisting of joint commands M_{Jt} for n joints as well as one contact force vector F_{Ct} and two contact point candidates P_{CAt}, P_{CBt} for each of m pairs of collision shapes.

$$N_P : O_t \mapsto M_{Jt1} \ldots M_{Jtn} \, , \, [P_{CAt1}, P_{CBt1}, F_{Ct1}] \, \ldots \, [P_{CAtm}, P_{CBtm}, F_{Ctm}] \tag{2.2}$$

Each pair of contact point candidates P_{CAti}, P_{CBti} is averaged to obtain a single contact response point P_{Rti}.

$$P_{Rti} = \frac{P_{CAti} + P_{CBti}}{2} \tag{2.3}$$

The dynamics model M_P maps simulation states S_{Pt}, robot commands M_{Jt}, contact points P_{Rt} and contact forces F_{Ct} to object and robot velocities dS_{Pt}.

$$M_P : \; S_{Pt} \, , \, M_{Jt1} \ldots M_{Jtn} \, , \, [P_{Rt1}, F_{Ct1}] \ldots [P_{Rtm}, F_{Ctm}] \; \mapsto \; dS_{Pt} \tag{2.4}$$

Simulation states S_{Pt} are mapped to observations O_t and task loss L_{Tt}.

$$S_{Pt} \; \mapsto \; O_t \qquad \qquad S_{Pt} \; \mapsto \; L_{Tt} \tag{2.5}$$

Contact predictions are evaluated with respect to the current simulation state $S_{P\,t}$ by a physical consistency loss $L_{P\,t}$ (section 2.7).

$$S_{P\,t}\,,\,[P_{CA\,t\,1}, P_{CB\,t\,1}, F_{C\,t\,1}]\dots[P_{CA\,t\,m}, P_{CB\,t\,m}, F_{C\,t\,m}] \mapsto L_{P\,t} \tag{2.6}$$

The training process optimizes neural network weights to minimize both task loss $L_{T\,t}$ (section 2.3) and physical consistency $L_{P\,t}$ (section 2.7) over time t and over initializations $S_{A\,0}\dots S_{A\,k}$ with probability densities $P_{A\,0}\dots P_{A\,k}$ of k state variables.

$$\min_{W_P} \quad \int\cdots\int \quad \int \quad L_{T\,t} + L_{P\,t}\; dt\; dP_{A\,0}(S_{A\,0})\dots dP_{A\,k}(S_{A\,k}) \tag{2.7}$$

The learning problem is solved through stochastic gradient-based optimization (section 2.9), drawing a random batch of initial states S_A for each optimization step. Time is integrated at a uniform step size via semi-implicit Euler integration.

2.7. Physical consistency loss

Physical consistency loss $L_{P\,t}$ evaluates contact force $F_{C\,t\,i}$ and contact point $P_{C[A,B]\,t\,i}$ predictions generated by the neural policy network for n_{SP} pairs of object and robot shapes. $L_{P\,t}$ consists of a sum of squared errors.

$$L_{P\,t} = L_{SP\,t} + L_{DP\,t} + L_{FP\,t} + L_{LP\,t} + L_{CP\,t} \tag{2.8}$$

Each contact candidate point $P_{C[A,B]\,t\,i}$ should lie within the respective collision shape. For this purpose, the shape is approximated by $n_{P[A,B]\,i}$ boundary planes defined by normal vectors $N_{[A,B]\,i\,l}$ and offsets $d_{[A,B]\,i\,l}$ with plane index l. The loss is only applied outside the shape using the ramp function R.

$$L_{SP\,t} = \sum_{i=1}^{n_{SP}} \left(\sum_{l=1}^{n_{PA\,i}} R(N_{A\,i\,l} \cdot P_{CA\,t\,i} + d_{A\,i\,l})^2 + \sum_{l=1}^{n_{PB\,i}} R(N_{B\,i\,l} \cdot P_{CB\,t\,i} + d_{B\,i\,l})^2 \right) \tag{2.9}$$

High contact forces $F_{C\,t\,i}$ contradict large distances and large distances contradict high contact forces.

$$L_{DP\,t} = \sum_{i=1}^{n_{SP}} \| F_{C\,t\,i} \otimes (P_{CA\,t\,i} - P_{CB\,t\,i}) \|^2 \tag{2.10}$$

Each contact force vector $F_{C\,t\,i}$ is further constrained by a polygonal friction cone, consisting of n_F planes with normals $N_{F\,i\,q}$ and offsets $d_{F\,i\,q}$. The direction of the friction cone, i.e. contact normal $n_{C\,i\,q\,t}$, is approximated via the separating vector from GJK/EPA [64, 17]. Slippage is regularized according to object velocities $V_{C[A,B]}$ at $P_{C[A,B]}$.

$$L_{FP\,t} = \sum_{i=1}^{n_{SP}} \left(\sum_{q=1}^{n_F} R(N_{F\,i\,q} \cdot F_{C\,t\,i} + d_{F\,i\,q})^2 \right) \tag{2.11}$$

$$L_{LP\,t} = \sum_{i=1}^{n_{SP}} \| F_{C\,t\,i} \otimes ((V_{CA} - V_{CB}) - n_{C\,i\,q\,t}\,(n_{C\,i\,q\,t} \cdot (V_{CA} - V_{CB}))) \|^2 \tag{2.12}$$

Objects are prevented from intersecting by minimizing intersection depths $d_{AB\,i\,t}$ along the separating axes.

$$L_{CP\,t} = \sum_{i=1}^{n_{SP}} R(d_{AB\,i\,t})^2 \tag{2.13}$$

2.8. Manipulation experiments

The contact-based formulation is used to train neural network policies for all eight tasks defined in section 2.3. For each task, the learning converges to a reasonable solution (see figures 2.4 and 2.5), despite using only local gradient-based optimization.

In case of the two-fingered rotation tasks, the result is similar to that of the baseline method. For the lifting task, after a few iterations, the policy begins to learn placing contacts with non-zero forces on the surface of the object and to move the hand towards the object. The contact force vectors point upwards and inwards at an approximately 45° angle (see figure 2.3). During the course of the optimization, the network learns to form a grasp around the object, move the hand upwards, and to adapt the horizontal position of the hand according to object position. For the cylindrical object, the policy learns periodic turning motions with perpendicular forces (see figures 2.4 and 2.3).

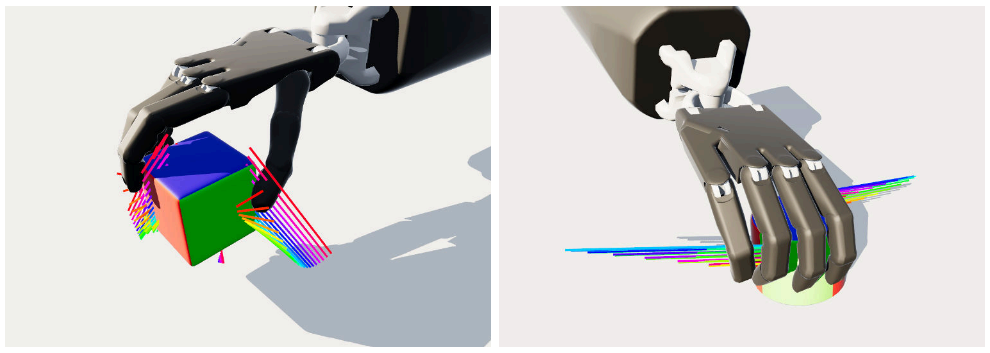


Figure 2.3.: Contact forces while learning to grasp a cube (left) and turning a cylinder (right) with a Shadow C6 hand through contact-based policy optimization. Image [289] © 2021 IEEE

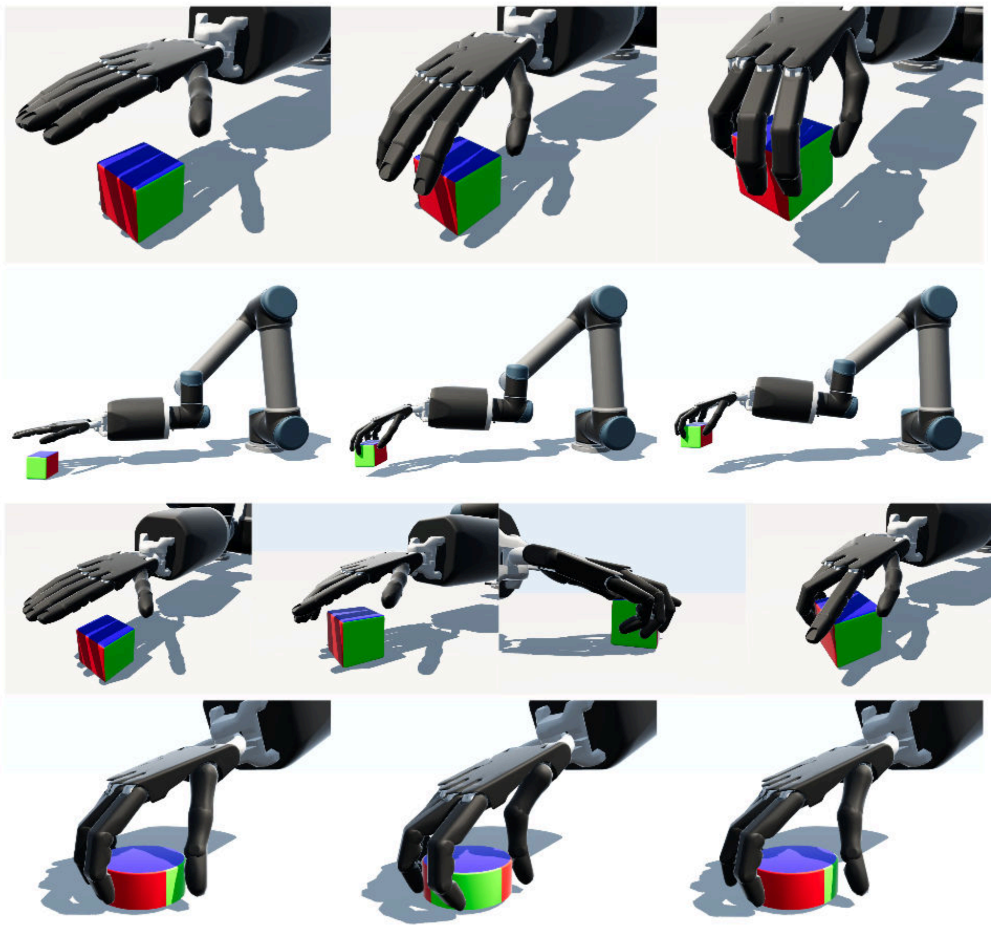


Figure 2.4.: Learning to grasp a cube (1st and 2nd row), push a cube (3rd row) and turn a cylindrical wheel (bottom) through contact-based direct policy optimization. Image [289] © 2021 IEEE

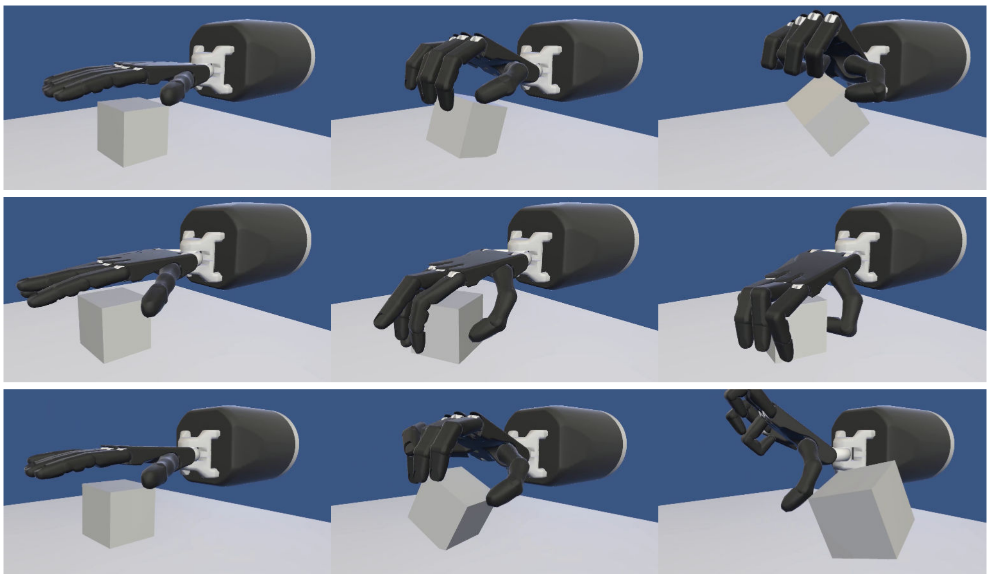


Figure 2.5.: Learning to turn a cube around the X (top), Z (middle) and Y (bottom) axis with a Shadow C6 hand through contact-based neural policy optimization.

2.9. Training methods

The method introduced above solves deep learning and contact resolution together in a single optimization. Artificial neural networks are commonly trained via steepest gradient descent, implemented through backpropagation, and variations thereof [107, 102]. Constraint solvers in state-of-the-art physics engines rely on different methods, handling joint and contact resolution as linear-complementary problems [212], linear-quadratic programs [184] or last-squares problems [132, 16]. These optimization methods can be particularly efficient for sparse systems, which includes typical contact resolution problems in physics engines. For deep learning problems, gradient matrices would typically be large and densely populated, which would result in a quadratic increase of memory use and computation time with increasing numbers of variables or network weights.

The contact-based learning problems in this chapter are therefore proposed to be solved via a matrix-free sequential least-squares method. At each outer iteration, the model is linearized around the current solution. In order to avoid exploding memory use and computation time from explicitly computed gradient matrices, the linearization is instead created through symbolic transformation as a linearized program. A small amount of diagonal regularization is added for stability. An inner loop then solves each linearization approximately via conjugate gradient descent.

During the manipulation experiments reported in section 2.8, the policies are trained using the proposed optimization method. As a baseline, traditional steepest gradient descent with backpropagation is tried as well across multiple different learning rates. Some but slower progress can be observed for tasks 1 and 2. For the other tasks, backpropagation fails completely to converge within similar orders of magnitude in training time. Figure 2.6 shows training loss over time for the proposed training method and backpropagation with multiple different learning rates for task 3. After some initial progress, the learning appears to be stuck, in some cases even failing to touch the object. Experiments using an SGD variant with adaptive learning rates [102] fail as well. Small initial learning rates and slow adaptation parameters result in similar convergence to baseline SGD [107] while higher initial learning rates and/or faster adaptation lead to instabilities and floating-point errors, causing the optimization process to fail.

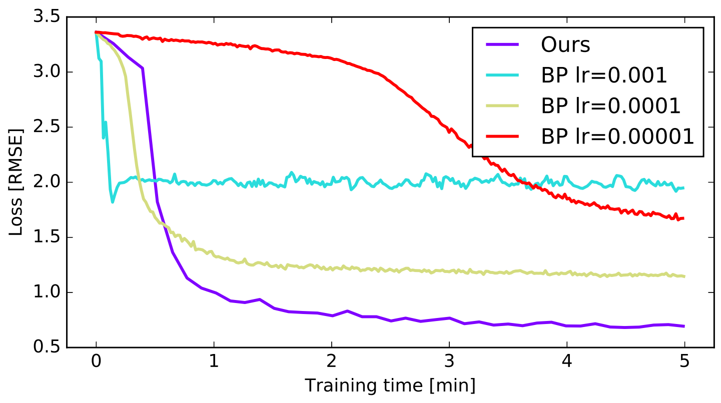


Figure 2.6.: Loss curves recorded while training a neural network for task 3 (sec. 2.3) using the matrix-free sequential least-squares solver and using traditional gradient descent with different learning rates. Image [289] © 2021 IEEE

3. Automatic linearization for robot learning and control

The learning methods presented in chapter 2 combine geometric robot models with artificial neural networks directly in a single optimization. The problem is solved via a matrix-free sequential least-squares method. An inner loop repeatedly propagates gradients back and forth through the same linearization. State-of-the-art deep learning frameworks with automatic differentiation miss opportunities to simplify equations in such cases. This chapter introduces an automatic differentiation and optimization framework that separates gradient propagation from non-linear evaluation. For each non-linear operator, a linearization function can pre-compute linear coefficients. These can be used to perform forward and reverse gradient propagation without having to re-evaluate non-linearities. Additionally, data types can be automatically replaced during gradient propagation. Absolute orientations can be represented as quaternions or matrices while their gradients can be represented as rotation vectors.

3.1. Related work

Many problems in robotics and artificial intelligence can be handled via gradient-based optimization. Deep learning typically derives gradient terms automatically through backpropagation [107, 215, 146] and solves the learning problems via steepest gradient descent [107] or variations thereof [102]. While automatic differentiation has proven successful for training artificial neural networks, this is currently not always the case for model-based approaches.

Physical, spatial and robot control problems can often be solved more efficiently by exploiting local linearizations [62, 168, 165, 61, 110, 140, 212, 184, 23, 15, 170, 98, 54, 167, 160, 216]. In several state-of-the-art physics engines [212, 184, 217], gradient terms are written explicitly by human engineers. When manually constructing gradient matrices involving spatial rotations, it is common to consider different angle representations for values and gradients [51, 38, 23]. State-of-the-art automatic differentiation frameworks use the same data types for values and gradients [7, 146, 215, 213]. For some control problems, it is possible to exploit sparse algebraic structure despite dense hessians [218, 51, 26, 171]. Artificial neural networks typically exhibit high-dimensional dense gradients. Some works propose to use approximate Gauss-Newton or Newton methods for training artificial neural networks [20, 33, 75]. Inner linear equations from applying the Gauss-Newton method to non-linear optimization problems can also be approximated via iterative methods. While some iterative linear solvers still require a gradient matrix [168], others can be implemented using only matrix-vector products or an equivalent linear function [28, 69, 82, 138].

Some later works begin to also address certain aspects of the above issues. A PyTorch [146] extension [175] adds support for orientation quaternions with 3D gradient vectors. A recent differentiable shading language separates linearization from gradient propagation [13].

3.2. Methods

The proposed framework first records a symbolic representation of the non-linear function using operator overloading. The problem can be specified in C++ or Python. The recorded function is transformed into programs for linearization, forward gradient propagation, reverse gradient propagation and gradient accumulation. The recorded and generated programs are then simplified.

3.2.1. Linearization programs

For each non-linear operator, a linearizer can be defined. The linearization program is constructed by finding the corresponding linearization function, if available, for each non-linear operator, and emitting a call to the linearization function with the arguments and return values of the non-linear function as arguments to the linearizer. Each linearization operator can return a linearization data structure, which is passed to the forward and reverse gradient programs. The forward and reverse gradient programs consist of only multiply and add instructions.

3.2.2. Forward gradient programs

Each operator is mapped to a corresponding forward gradient operator. If a corresponding linearizeration operator is provided, the forward gradient operator is called with input gradients and the linearization data structure as inputs. If no linearizeration operator is provided, the forward gradient operator is called with input gradients and arguments as well as return values from the non-linear program as arguments. Data types for values and gradients can be different.

3.2.3. Reverse gradient programs

The non-linear program is reversed and each operator is replaced with the corresponding reverse gradient operator, adding gradient arguments, substituting gradient types, and replacing value arguments with linearizers if available. If the same variable is read multiple times, a binary tree is generated to merge reverse gradients.

3.2.4. Accumulation programs

During gradient-based optimization, gradients have to be accumulated and added to the candidate solution, which would amount to a trivial vector addition if gradient and value types were the same. However, if value types can be replaced with separate gradient types, an additional accumulation operator has to be provided. The addition operator is overloaded for value and gradient types. For each argument of the original non-linear function, a call to the overloaded addition operator is emitted into an accumulation program. The accumulation program receives return values from both the non-linear program and the reverse gradient program as arguments and returns updated non-linear values.

3.2.5. Program simplification

The generated programs are finally simplified by pre-computing constant expressions, identifying and skipping redundant move instructions, detecting and eliminating duplicate constants with equal values, compressing constants with the value zero through special zero operators, and removing unused constants and instructions. Memory is allocated statically and defragmented.

3.2.6. Operators and linearizations

A library of operators and linearizers is defined within the framework presented above. Special attention is given to efficient gradient propagation across kinodynamic robot models in three-dimensions space. Three exemplary operators are detailed in the following.

Rotation chaining

Robot simulation and control problems frequently involve the concatenation of multiple rotations. This is typically handled via quaternion or 3-by-3 matrix products. The quaternion representation is more compact, requiring only 4 scalars, while the matrix representation requires 9 scalars, potentially leading to increased register spill and decreased cache locality. Furthermore, quaternion multiplication requires only 16 multiplications and 12 additions, while a 3-by-3 matrix multiplication requires 27 multiplications and 18 additions. For reference, a state-of-the-art symbolic automatic differentation and computer algebra system [7] is used to derive and simplify forward and reverse gradient terms. This results in 99 and 90 instructions for forward and reverse matrix gradients as well as 60 and 56 instructions for forward and reverse quaternion gradients.

The proposed automatic differentiation framework can represent rotational gradients as 3D vectors while using quaternions Q_R for absolute orientations. The linearizer $f_{l,q,q}$ simply stores the first quaternion Q_{R1}.

$$f_{l,q,q}(Q_{R1}, Q_{R2}) = Q_{R1} \tag{3.1}$$

Forward gradient propagation $f_{g,q,q}$ transforms the second rotation vector V_{R2} by the first rotation quaternion Q_{R1} and adds the first rotation vector V_{R1}.

$$f_{g,q,q}(V_{R1}, V_{R2}, Q_{R1}) = V_{R1} + Q_{R1}\, V_{R2}\, Q_{R1}^{\star} \tag{3.2}$$

The chain product between a quaternion, a vector and the conjugate of the same quaternion can be computed efficiently via 15 multiplications and 15 additions [219]. Adding the first vector gradient results in 15 multiplications and 18 additions, a total of 33 instructions.

Reverse gradient propagation $f_{r,q,q}$ transforms the incoming gradient V_{R3} by the conjugate of the first rotation quaternion Q_{R1}.

$$f_{r,q,q}(V_{R3}, Q_{R1}) = (V_{R3}\,,\, Q_{R1}^{\star}\, V_{R2}\, Q_{R1}) \tag{3.3}$$

The conjugate quaternion-vector-quaternion chain product can be computed using the same method as before with reversed signs, resulting in only 15 multiplications and 15 additions, or 30 arithmetic instructions, for backpropagation.

The proposed formulation reduces instruction counts for forward and reverse gradient propagation by almost one half compared to quaternion gradients and to exactly one third compared to matrix gradients.

The linearizer simply references one quaternion and operates on 3D vectors, requiring up to 10 load and 3 store operations. Automatic differentiation through quaternion or matrix products depends on 8 or 18 scalars for the values, 8 or 18 scalars for the inputs and 4 or 9 scalars for the outputs, resulting in 20 memory operations for quaternions or 45 memory operations for rotation matrices. The proposed method reduces memory bandwidth by 35 % or 71 % compared to quaternion or matrix gradients.

Angle-axis rotations

Three-dimensional rotations can depend on scalar angles, for example when modeling revolute robot joints. The rotation angle turns a dependent frame around a rotation axis. Performing all calculations in axis-angle representation can be inefficient and inconvenient. The angle-axis representation is therefore typically converted into a quaternion or rotation matrix. Doing so requires computationally expensive trigonometric functions. Applying the chain rule directly for automatic differentiation results in gradient terms that still contain trigonometric functions. However, when repeatedly propagating gradients through the same linearizations (e.g. see section 2.9), the relationship between the gradients is linear and only additions and multiplications should be required. For simplicity, the following assumes a constant rotation axis. The axis can be rotated by applying the rotation operator from the previous subsection to the output.

The forward differentiation operator $f_{g,\alpha,v}$ simply multiplies a rotation angle gradient α' by the rotation axis V_X.

$$f_{g,\alpha,v}(\alpha', V_X) = \alpha' V_X \tag{3.4}$$

The reverse operator $f_{r,\alpha,v}$ computes the dot product between rotation vector gradient V'_R and rotation axis V_X.

$$f_{r,\alpha,v}(V'_R, V_X) = V'_R \cdot V_X \tag{3.5}$$

When propagating gradients through programs that use angle-axis rotations, the proposed framework can automatically replace computationally expensive trigonometric functions with simple vector-scalar multiplications and vector dot products.

Trigonometry

Some applications may also evaluate trigonometric functions explicitly. For the sine function, gradients are proportional to the cosine. Instead of always re-evaluating the cosine during gradient propagation, it is computed once by the linearization function. During forward or reverse gradient propagation, the derivative is simply multiplied by the precomputed factor.

3.3. Validation

Three different test suites are developed to validate correctness. The gradient operators are sampled for random inputs and the results are compared to different baselines.

- A first finite difference test estimates derivatives for random values and gradients numerically using the non-linear operator, then calls the linearization, forward and reverse operators for the same inputs, and compares the results.
- A second self-test uses basic scalar operators from the proposed framework itself to derive symbolic gradient terms for the non-linear operators. Both the automatically derived terms from scalar automatic differentiation and the linearization, forward and reverse block operators are sampled with random inputs and the results are compared.
- A third spatial transformation test propagates gradients across chains of five random coordinate frames and compares the results with traditional scalar automatic differentation.

The tests appear successful for reasonable error bounds and input domains. For some operators and some input ranges, such as real-valued square roots of negative values, outputs may be undefined. For this purpose, input domains for the first two tests are split into different ranges and results are divided by input domain. The third test can be executed on arbitrary random numbers, and for double-precision floating point numbers, the results agree with a mean squared error of $1 \cdot 10^{-15}$.

4. Tactile sensor skin

Capturing human interaction forces requires sensors to be placed either around human hands or on objects and tools. Physically modifying all tools and objects or workpieces would not always be practical. Tactile sensors that are placed on human hands may have to curve around the hands and fingers to acquire all relevant contact information. The sensors should be stretchable to adapt to hand motions and deformation of soft tissue and to at least partially preserve human tactile sensation. To re-use prior human experience during teaching, for compatibility with existing tools and objects designed for human hands, and for easy cleaning, a smooth and closed surface may be advantageous. Skin-like tactile sensors can also be the preferred choice for robots to distinguish multiple simultaneous contacts, simplify human-to-robot transfer and to maximize internal space available for mechanics and actuation. For safety reasons, materials used in the constructions of tactile sensors should be non-toxic to humans as well as non-corrosive to robots. Fully solid-state construction should be preferred for durability and to simplify handling, use and storage.

4.1. Related work

Principles for measuring tactile information include piezo-resistive, capacitive and optical sensing. Piezo-resistive materials and surfaces change their electrical resistivity depending on contact pressure. Such materials can be placed between conductive materials to construct pressure sensors [220, 180, 25, 221, 50, 100, 46]. Common limitations are susceptibility to environmental factors such as moisture, relatively high hysteresis and aging [50]. Capacitive pressure sensors place a non-conductive compressible layer between two conductive layers and measure capacitive coupling between the conductive layers across the dielectric layer [108, 222]. Capacitive sensors can also be created using liquid droplets [136]. Optical tactile sensors measure deformation of an elastic membrane using cameras or other light sensors [201, 127, 104, 109, 178, 189]. On robots, the camera can be placed inside the finger. However, this is typically not practical for measuring interaction forces on human hands. Current tactile sensor gloves suffer from low elasticity, low spatial resolution, mechanical obstruction and poor coverage around curved surfaces such as fingertips [220, 180, 222, 25, 193]. Forces on fingertips can also be measured using rigid transducers and force-torque sensors. However, this requires relatively large amounts of space and the shells impede human sensing [14]. Fingertip sensors for robots can be constructed with conductive liquid chambers inside elastic shells [53, 133, 162].

Elastic components can be produced from non-toxic silicone rubber [53, 126, 8, 65]. Softness can be increased with liquid fillers, producing gels, but these can suffer from poor durability [126, 73, 209, 203, 130]. Silicone rubber and other elastomers can be imparted with electrical conductivity via conductive fillers [172, 192, 42, 10, 21]. Achieving good conductivity with regular carbon black filler can be challenging since high filler content increases viscosity and due to cure inhibition from contaminants [126]. Other newer conductive fillers are subject to health concerns [163, 10]. Conductors in stretchable electrics can also be created using liquid metals [108, 183, 196, 63] or conductive grease [72]. Current liquid metals can offer good conductivity but lead to fire and explosion (NaK) [125], health (Hg) or corrosion (Ga) hazards. Various types of sensors can also be constructed using conductive fabrics and yarns [222, 25, 180, 223, 224]. Distributions can be measured across surfaces by either connecting a set of sensor cells to readout electronics individually [127], by inferring distributions from boundary measurements [105, 34, 205, 203] or by arranging sensor cells as a sensor matrix [220, 100, 108].

4.2. Preliminary experiments

4.2.1. Fabric-based glove and object instrumentation

First multimodal human demonstration data for robot learning experiments (chapter 7) is recorded via a hybrid approach with a fabric-based glove and instrumented objects (see figure 4.1). One half of the tactile sensor is placed on the glove, the other half on the object. The glove is equipped with pads of electrically conductive fabric[1]. Object surfaces are covered in an inner layer of metal foil and an outer layer of a piezo-resistive film[2]. The electrodes are connected to a microcontroller[3], which is programmed to apply excitation voltages to one set of electrodes and measure voltages between sensors and a set of reference resistors at the other set of electrodes. In addition to force estimates and approximate contact locations, this approach can also be used to directly measure pair-wise correspondences between fingers and object surfaces, without an immediate need for complex reconstruction algorithms. Furthermore, by placing only one additional layer of fabric on the glove, the glove remains fairly flexible, even when using simple off-the-shelf fabrics. One drawback of this approach is that current can flow between different electrode pads on the glove if the fingers touch each other, leading to ambiguities when attributing contacts to electrode pads. Such cases could potentially be detected via pair-wise resistance measurements between all electrodes. However, if the fabric on the glove is fully conductive, the measurements would still be corrupted. As a second drawback, this approach requires both the glove and the objects to be modified. Nevertheless, it enables successful data recording for first teaching experiments.

[1]Conductive Fabric - 12"×13" Ripstop, Zhiwei Robotics Corp., Shanghai, China
[2]Pressure-Sensitive Conductive Sheet (Velostat/Linqstat), Adafruit Industries LLC, NYC, USA
[3]ESP32, Espressif Systems, Shanghai, China

4.2.2. Fabric-based tactile sensors

Additional fabric-based tactile sensor prototypes are created that integrate both electrode layers and an intermediate piezo-resistive layer on the glove (see figure 4.1, middle, right). The parts are attached with small amounts of glue to minimize fraying. A first inner layer of conductive fabric[1] is attached to a thin fabric glove. A layer of pressure-sensitive conductive foil[2] is placed on top, followed by a third layer of conductive fabric. Small extensions from the electrode pads reach to the back of the fingertip and are connected to readout wires. The sensors are read by applying a readout voltage and measuring the conductivity against reference resistors.

While in principle functional, the sensors suffer from cross-talk between pressure and stretch, with measurements being noticeably affected by both finger motion and contact force. Additionally, mounting all four layers on top of each other leads to high mechanical stiffness, increased thickness and reduced flexibility. In practical systems, at least one further layer may be required for insulation. Despite using glue for mounting the parts, fraying still appears to be an issue. During use of the glove, fibers occasionally detach from the fabric and could potentially lead to short circuits in larger systems with many sensor cells. Applying additional glue mainly seems to decrease flexibility. Finally, with the additional layers mounted on top, seams in the glove itself become increasingly noticeable and due to varying thickness, it would be hard to teach precise contact locations.

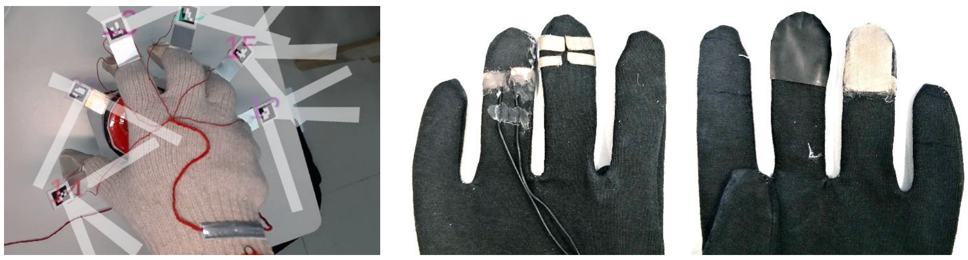


Figure 4.1.: Early fabric-based sensor glove with conductive finger-tips, Aruco [156] tracking markers and overlayed marker detections (white) and instrumented bottle with a conductive pressure-sensitive rim (left), multi-layered fabric-based sensor prototypes on a glove (middle, right).

4.2.3. Robot fingertips

A fingertip-shaped plastic core with elevated ridges is designed in Blender [225] and manufactured using an FFF printer. The top of each longitudinal ridge is covered with a strip of conductive metal tape. The tip of each lateral ridge is then first covered with insulating electrical tape and then by a slightly thinner strip of metal tape. A piece of pressure-sensitive film[2] and then a piece of rubber foam are repeatedly pulled over a second smooth fingertip print to mold the films into the same shape, then placed onto the print with the conductive traces, and finally fixed at the edges with double-sided adhesive tape (see figure 4.2). The electrodes are connected to a flat cable on the back of the sensor using conductive ink.

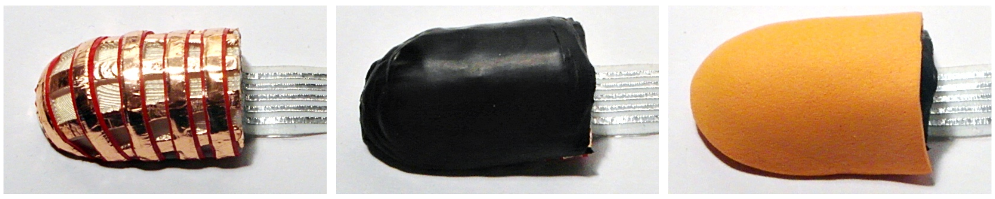


Figure 4.2.: 3D-printed core with metal electrodes (left), pressure-sensitive film (middle) and complete fingertip sensor with foam cover (right).

The sensor can approximately detect contact forces and locations (see figure 4.3) while wrapping around a curved fingertip shape. However, it shows a non-linear response and saturates easily under high forces, suggesting that it may be advantageous to place electrodes on both sides of the sensing elements despite complicating mechanical strain relief.

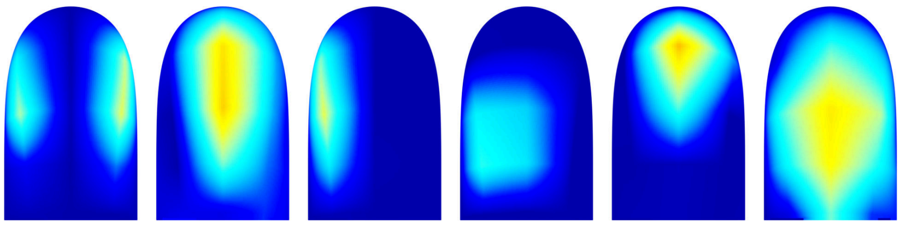


Figure 4.3.: Example responses to mechanical contacts of a first double-curved robot fingertip prototype.

4.3. Foil-based tactile matrix sensors

Resistive tactile matrix sensors are created from aluminum tape and piezo-resistive polymer films. The sensors consist of a first layer with parallel strips of metal tape, a second pressure-sensitive polymer layer, a third layer of parallel strips of metal tape perpendicular to the first, and finally a non-conductive surface layer. The sensor matrix is read by applying voltages to electrodes on one layer and measuring currents flowing through the pressure-sensitive polymer film to the perpendicular electrodes. For the pressure-sensitive layer, a piezo-resistive ESD protection foil is used[2]. The electrodes are produced from an aluminum adhesive tape[4]. Electrode shapes are designed in OpenSCAD [226] and Inkscape [227]. The metal foil is cut using a cutting plotter[5] [228]. Electrodes extend out of the sensors as flexible cables (see figure 4.4).

[4]AT502, Advance Tapes International Ltd., Westmoreland Avenue, Thurmaston, UK
[5]Portrait 2, Silhouette America, Inc., USA

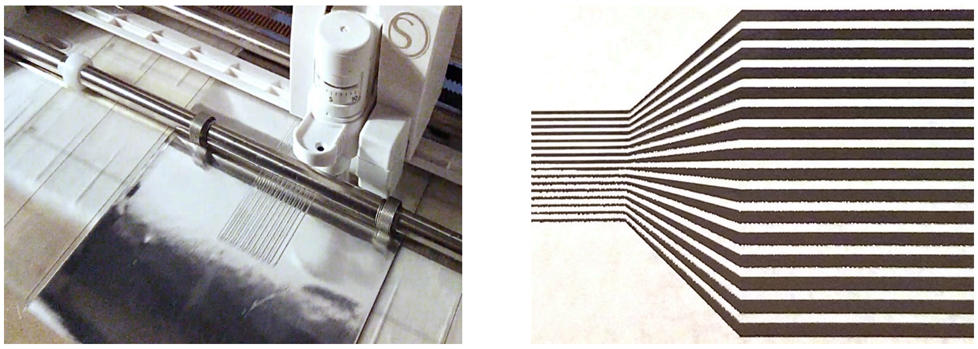


Figure 4.4.: Cutting electrode shapes using a cutting plotter (left), sensor electrodes fanning out from narrower cable conductors (right).

A flat prototype with 16 rows and 16 columns is assembled. Each metal tape layer is glued to a non-conductive plastic foil. The assembly is placed on a thin rubber foam sheet. The sensor responds to human touch and can detect multiple simultaneous contacts, as shown in figure 4.5.

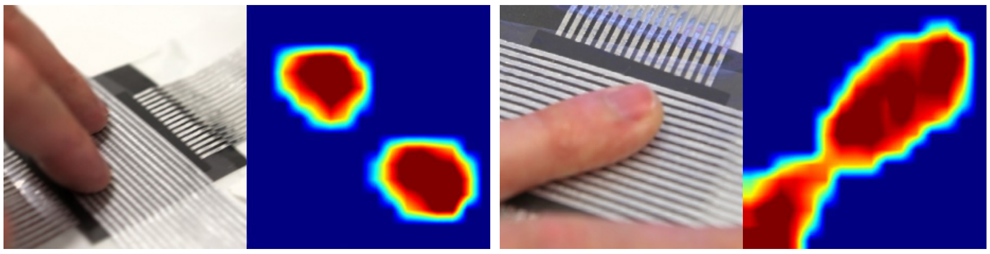


Figure 4.5.: 16-by-16 tactile sensor matrix and contact response.

The cable extensions from the electrodes are clamped onto the edges of circuit boards with rows of contact pads. The excitation circuit uses a 16-channel port extender IC[6] and the reception circuit a 16-channel analog multiplexer[7]. Sequencing, analog-to-digital conversion and host communication are implemented using a microcontroller[3]. The sensor is exposed to different objects and stimuli. It can successfully detect edges and corners, multiple contacts from the coils of a telephone cord, the holes in washers, and the bubbles in a piece of bubble wrap (see figure 4.6).

[6]MCP23017SO, Microchip Technology Inc., AZ, USA
[7]CD74HC4067, Texas Instruments Incorporated, Dallas, Texas, USA

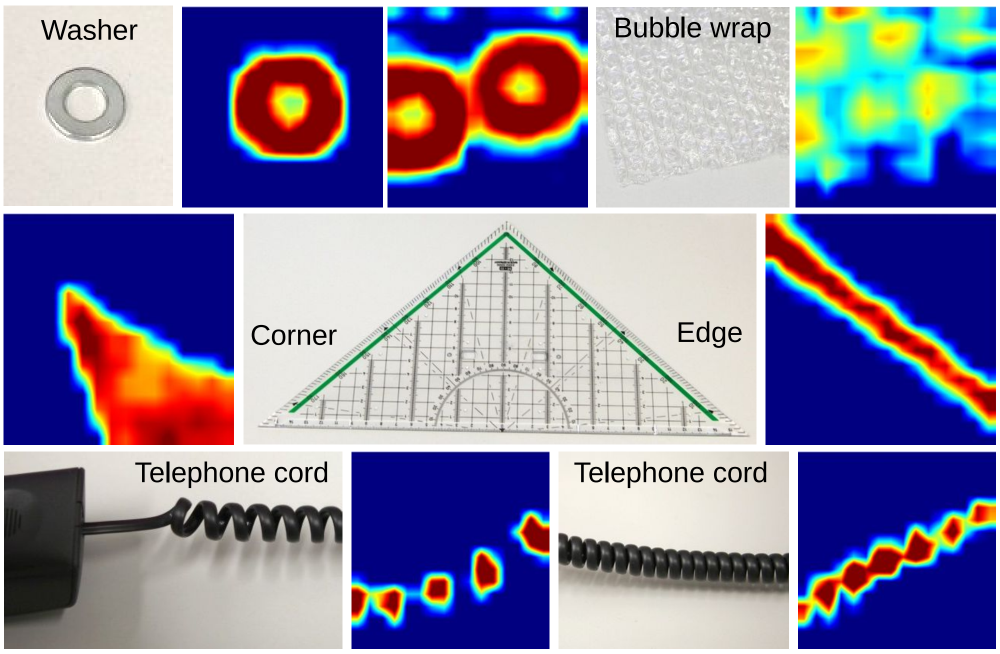


Figure 4.6.: Response of a resistive tactile sensor matrix to different stimuli (first published in [290] under CC BY 4.0).

4.3.1. Crosstalk compensation

The sensor matrices can be read by connecting each row via a reference resistor to ground or a another fixed reference voltage, successively applying a voltage to each column and measuring the voltage at each row. Since this approach allows variable voltages at the reception lines, small amounts of current can also flow through other sensor cells, leading to crosstalk.

The effect can be avoided by reading the sensors via transimpedance amplifiers [229, 230]. The reference resistor is placed between the inverting output of an operational amplifier and the tactile reception line. This ensures that the reception lines always maintain constant voltage levels (see figure 4.7, bottom-left), suppressing crosstalk. However, active suppression requires additional electronics.

Crosstalk compensation can alternatively be performed by computational means. Each sensor cell is modeled as a variable resistor. The sensor model computes voltages at the reception lines given a resistance distribution and excitation voltages. The reconstruction method then continuously solves for resistance distributions that explain the current measurements from the real sensor. Solutions to the reconstruction problem can be efficiently approximated by an iterative method [24]. Results are successfully verified by applying the sensor model again and comparing outputs to original measurements. Computational crosstalk compensation allows the sensors to be read using off-the-shelf microcontrollers (and for some large sensors multiplexers), without requiring active compensation circuits. The method is used for the experiments in figure 4.5, figure 4.6 and section 4.4.

4.3.2. Characterization

Sensor response is characterized by pressing a cylindrical tip onto a sensor cell and plotting sensor readings over force. The measurements are performed using a custom motorized calibration device (figure 4.7, top-left). The sensors show a mostly linear response until reaching saturation (figure 4.7, right). Sensitivity can be adjusted by the choice of feedback resistors. One sensor cell can detect forces below 1N and as high as 25N.

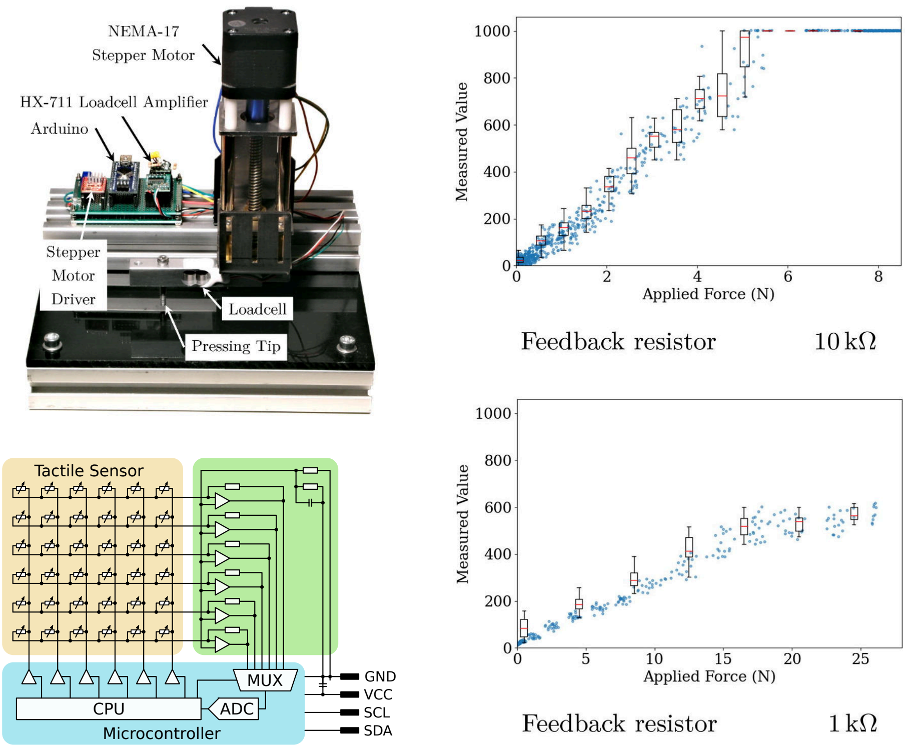


Figure 4.7.: Calibration device (top-left), readout circuit (bottom-left, sensor response for different feedback resistors (right) [290] CC BY 4.0.

4.3.3. Integration

The sensors are integrated on an instrumented can and on the palm of a Shadow C6 humanoid robot hand (see figure 4.8). The instrumented can features 512 sensor cells distributed across two 16-by-16 matrices, demonstrating scalability and applicability to curved surfaces. The Shadow hand was previously only equipped with tactile sensors on the fingertips. The palm sensor consists of multiple segments to match the shape of the robot hand. One segment curves around the ball of the thumb. The cables from the palm sensor are connected to the readout board using Z-axis tape[8]. See [290, 288] for further details on robot integration and applications.

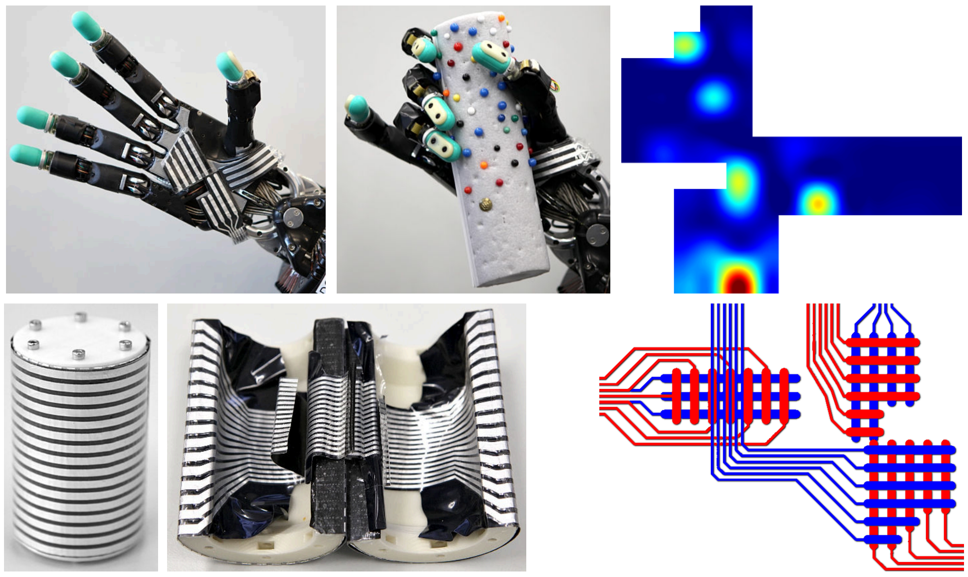


Figure 4.8.: Tactile palm sensor on a Shadow C6 humanoid robot hand with sensor response (top [290] CC BY 4.0). Instrumented can (bottom-left [290] CC BY 4.0), flexible cables inside instrumented can (bottom-center), electrode shapes for robot palm sensor (bottom-right).

4.4. Double-curved tactile sensor matrices

In order to produce double- and multi-curved tactile matrix sensors, the parts are assembled on three-dimensional fixtures, which are subsequently removed. A first fingertip prototype is designed in OpenSCAD [226] via CSG (constructive solid geometry). The assembly fixture is equipped with small protrusions for part alignment. Conductive strips are designed to wrap around the fixture and extend through a flat cable along the back of the finger. Slightly wider non-conductive films serve as insulation. The non-conductive film merges into a single wide rectangular part along the cable. The electrodes are separated along the entire length. Strips of pressure-sensitive film[2] sit between inner and outer electrodes. A small piece of insulating film is positioned above the finger nail to prevent short circuits between driving and receiving traces. The fixture is split lengthwise into four segments for extraction after assembly.

[8]3M™ Electrically Conductive Adhesive Transfer Tape 9703

See figure 4.9 for renderings of the parts and fixture. The parts are cut using a cutting plotter[5] and the assembly jig is printed on an FFF 3D printer. Insulating parts are cut from adhesive Vinyl tape. To assemble the sensor, the four pieces of the fixture are clamped together. The components for the sensor are then attached one after another to the assembly fixture and pressed together. Finally, the clamp holding the fixture together is removed and the four segments are successively extracted.

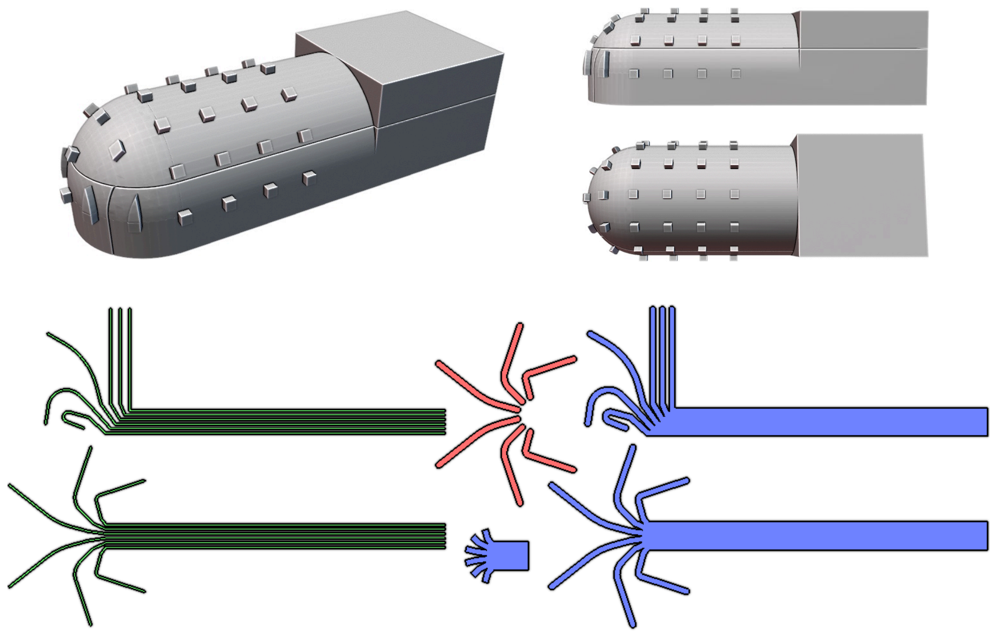


Figure 4.9.: CAD models of an assembly jig (top) as well as part shapes (bottom) for electrodes (green), pressure-sensitive film (red) and insulating layers (blue) for constructing a double-curved tactile fingertip sensor.

The sensor can be worn on a human finger while recording tactile data and it can resolve multiple simultaneous contacts (see figure 4.10). However, since the device is not stretchable, it severely limits human tactile sensation. Furthermore, after continued use, the metal electrodes begin to break. This typically happens first at the fingertip, but in some cases eventually also along the cable.

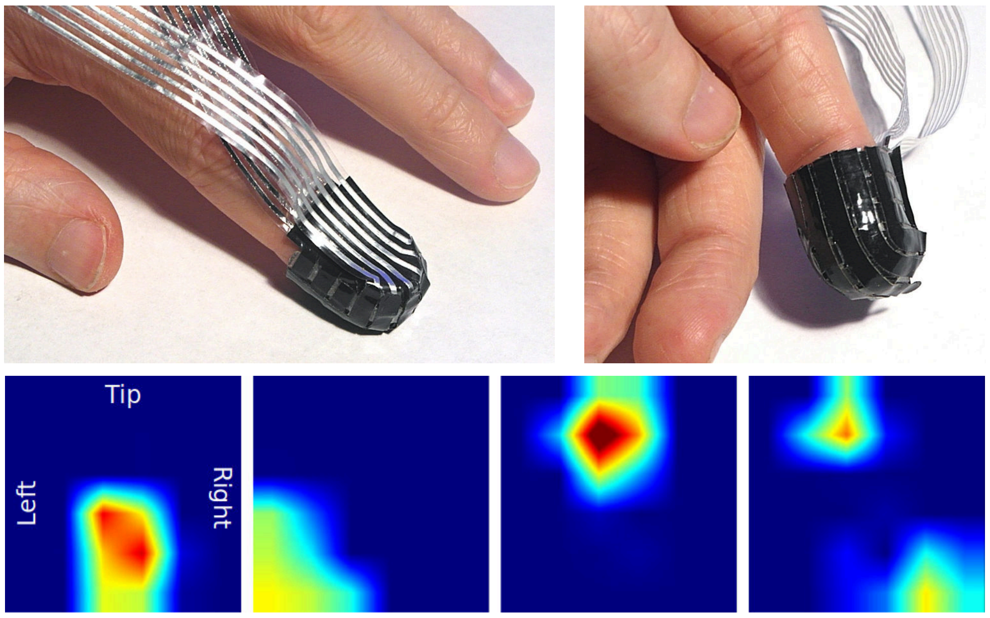


Figure 4.10.: Double-curved tactile matrix sensor on a human fingertip (top) and sensor outputs for different contact locations (bottom).

4.5. Elastic tactile sensor skin

4.5.1. Feasibility study

In order to assess the general feasibility of creating tactile matrix sensors completely from stretchable materials, a first prototype (figure 4.11, top) is constructed from commercially available TPU (thermoplastic polyurethane). A layer of electrically conductive TPU[9] is dispensed using an FFF printer. The dielectric layers are created from a textured TPU foil[10]. The sensor consists of a first dielectric layer, a second layer of horizontal conductive strips, followed by two layers of the dielectric TPU, vertical conductive strips and a final non-conductive layer. The components are placed on top of each other and joined using a laminator[11]. With the right pressure and temperature, the layers adhere to each other while the textured TPU surfaces form small compressible cavities. If pressure is applied to the sensor, the distance between the conductive TPU strips decreases and the capacitance increases. The conductive strips extend out of the sensors on two sides, on one side covered by non-conductive foils. The ends of the conductive traces are connected to metal electrodes with wires using Z-axis tape[8]. The sensor mainly responds to pressure (figure 4.11, bottom-left) and is

[9]Ninjaflex Eel, Fenner Precision Polymers, PA, USA
[10]TPU Film Thermoplatic Polyurethane, Fabimonti, Fabio Montenegro, Norderstedt, Germany
[11]Olympia A 230 Plus, Pam's Naturprodukte Shop, Mellrichstadt, Germany

less affected by bending (figure 4.11, bottom-right). The prototype can be used as a tactile sensor, but suffers from low sensitivity, low elasticity and plastic deformation under high stretch.

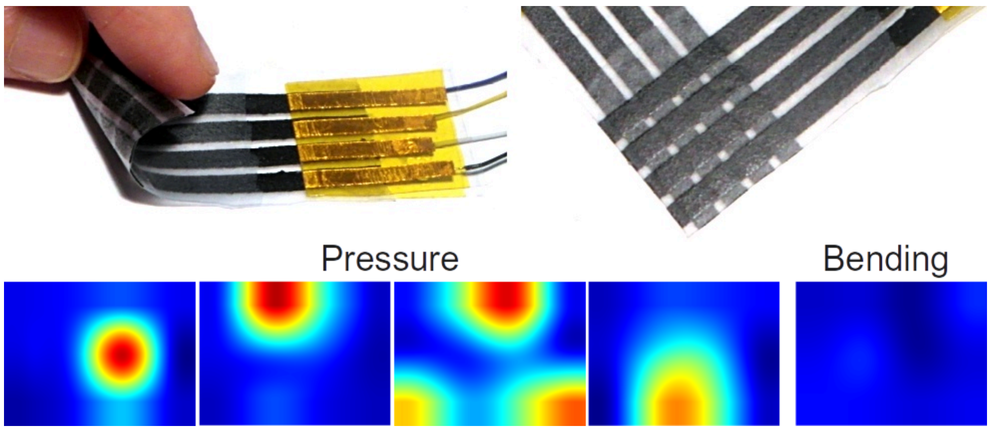


Figure 4.11.: TPU-based capacitive sensor prototype (top) and sensor response under contact forces (bottom-left) and bending (bottom-right).

4.5.2. Silicone materials

Material properties are further adapted by developing new silicone-based formulations. Electrically conductive parts are constructed with carbon black as a conductive filler. Dielectric layers are created from non-conductive silicone.

During first experiments, various commercially available liquid silicone rubbers are mixed with different kinds of carbon black. In most trials, the silicone fails to cure. Two-part addition-cure silicone is known to be susceptible to cure inhibition [126]. A one-part low-viscosity acetoxy silicone[12] successfully cures with regular carbon black. However, since the viscosity increases with filler content and the curing of acetoxy silicone is initiated by air moisture, sufficient mixing can be difficult. Additionally, using acetoxy silicone for conductive parts can complicate further sensor production steps, since acetoxy silicone can inhibit addition-cure silicone. Further experiments are conducted with addition-cure silicones, elevated cure temperatures, and a specialty-grade carbon black originally intended for food-contact applications with low amounts of contaminants[13]. First mixing three parts part A and three parts part B of a two-component addition-cure silicone[14] each with one part of the food-contact carbon black, then combining both mixtures, pressing the material between two sheets of PTFE foil and heating for two hours at 120° C results in a successfully cured electrically conductive sheet of silicone rubber. The material can be stretched by 100 % and has a

[12]AS1525, CHT Germany GmbH, Tübingen, Germany
[13]Food Contact Royale Black PP802, CAS 1333-86-4, Profiltra BV, Netherlands
[14]Silikon Kautschuk Typ 2 Abformsilikon mittelhart, Troll Factory Rainer Habekost e.K., Germany

volume resistivity of 0.3 Ωm, already showing better conductivity than the conductive TPU[9]. If the conductive filler content is increased further, mixing and processing becomes more difficult due to higher viscosity and the material fails to cure.

To further increase both stretchability and conductive filler content, fully custom formulations are developed. 74.3 g vinyl-terminated polydimethylsiloxane[15] with a viscosity of 20 000 cSt is mixed with 24.8 g carbon black[13], 0.87 g of a methylhydrosiloxane-dimethylsiloxane [16] as crosslinker and a Platinum catalyst[17]. The resulting conductive silicone rubber can be stretched by 400 % without damage. However, before curing, the mixture still has a high viscosity and can be hard to process. The viscosity can be slightly decreased using a lower-molecular-weight siloxane and a chain extender. For an additional decrease in viscosity while at the same time increasing filler content, solvents are added. Hydrocarbons can be used, but the amount should be limited for successful curing. 48.5 g hydrocarbon solvent[18], 31 g vinyl-terminated polydimethylsiloxane[15] with a viscosity of 2 000 cSt, 19.4 g carbon black, 0.912 g hydride-terminated polydimethylsiloxane[19] as a chain extender, 0.228 g of the crosslinker[16] and 10 mg of the catalyst[17] are mixed and spread on a PET sheet with a blade. After one hour, the sheet is heated to 100°C for multiple hours on a hotplate to remove the solvent. The material shows a volume resistivity of 0.025 Ω m and an elongation at break above 550 %, outperforming the first conductive silicone formation, the conductive TPU, and materials from related work using carbon nanotubes and graphene [192, 42]. However, the formulation requires high processing time, heating and ventilation. Easier processing, higher solvent content and additional slight improvements in material properties are achieved using a volatile silicone oil, Hexamethyldisiloxane[20] (HMDS), as solvent. Higher solvent content enables not only higher filler content but also the use of a high-molecular-weight silicone gum for improved elasticity. 400 g hexamethyldisiloxane[20] are mixed with 53 g vinyl-functional siloxane gum[21], 40 g carbon black[13], 4 g hydroxy-termianted PDMS[22] as a processing aid and 632 mg crosslinker[16]. After a small amount of catalyst[17] is added, the mixture is spread using a blade and cures at room temperature into a conductive silicone rubber film. A bare piece of conductive silicone film can be stretched by approximately 600 %. If glued to a non-conductive silicone film, it can be stretched by 700 % before either the non-conductive film tears or the films delaminate.

Non-conductive parts can be produced from standard molding silicone[23]. Recently developed high-elongation silicones [8, 65] can be more stretchable but require increased cure times of approximately one day or elevated cure temperatures. When casting layers of artificial skin, solvent-based processing can also be used for the non-conductive parts, enabling high elasticity as well as short cure times. 500 g HMDS[20]

[15]CAS 68083-19-2, Nedform BV, Netherlands
[16]CAS 69013-23-6, Crosslinker 200, Evonik Industries AG, Germany
[17]CAS 68478-92-2, Karstedt's catalyst solution, Nedform BV, Netherlands
[18]CAS 649-327-00-6, hydrocarbons C10 - C13
[19]CAS 70900-21-9, hydride-terminated PDMS 5 cSt, Nedform BV, Netherlands
[20]CAS 107-46-0, hexamethyldisiloxane, WACKER AK 0,65, Wacker Chemie AG, Germany
[21]CAS 67762-94-1, Vinyl Gum 0.074, Nedform BV, Netherlands
[22]CAS 70131-67-8, PDMS, hydroxy terminated, M.W. 4200, VWR International GmbH, Germany
[23]Silikon Kautschuk Typ 1 Abformsilikon mittelhart, Troll Factory Rainer Habekost e.K., Germany

are mixed with 80 g vinyl-functional siloxane gum, 673 mg crosslinker[16], 16 g hydrophobic fumed silica as a reinforcing filler[24] and 800 mg zinc oxide as an opacifier for easier processing. When adding a hydrosilylation catalyst[17] and spreading the mixture to allow the solvent to evaporate, a soft silicone rubber film with a maximum elongation at break of approximately 800 % is formed within 30 min to 1 h.

Internal pressure-sensitive dielectric structures are cast in closed molds, requiring solvent-free processing. Commercial molding silicones can contain liquid fillers or plasticizers, potentially leading to changes in mechanical properties over time [126]. These potential issues are avoided by producing custom fast-curing (approximately 20min) liquid molding silicones from vinyl-terminated PDMS[15] with a viscosity of 2 000 cSt, hydride-terminated PDMS[19] chain extender, hydrophobic fumed silica as a reinforcing filler, crosslinker[16], zinc oxide as an opacifier and zinc stearate as a release agent. In some prototypes, dielectric structures are also cast from a commercial high-elongation silicone [8, 65] without liquid fillers, which requires increased temperature or cure time.

4.5.3. Dielectric structures

The sensor cells form variable capacitors with compressible cavities between two layers of electrodes. Experiments are performed with different structures and production methods.

The dielectric structure can consist of closed air pockets or of open-celled column arrays. In a closed-celled structure (figure 4.13), compression depends mainly on air pressure, potentially minimizing the effects of material hysteresis. However, when stretching the skin, the surface area increases, leading to higher atmospheric force, resulting in additional crosstalk between stretch and pressure. Open-celled dielectric structures (figure 4.14) put higher demands on material properties, since the response depends solely on the elastic properties of the materials and sensor geometry. However, with open-celled dielectric structures, internal and external air pressure can equalize when stretched, potentially reducing crosstalk between stretch and pressure.

The dielectric structures can be cast in-place on 3D-printed skin molds. However, this limits the structure size to the resolution of the printer and relatively large but thin three-dimensional parts have to be carefully transferred from one mold to another. If the dielectric patterns are molded as individual small pieces and subsequently transferred to the skin mold, finer patterns can be created using regular 3-axis CNC mills (figure 4.12). However, this requires additional work for piece-wise assembly.

[24]CAS 68909-20-6, Aerosil R 812 S, Evonik Industries AG, Germany

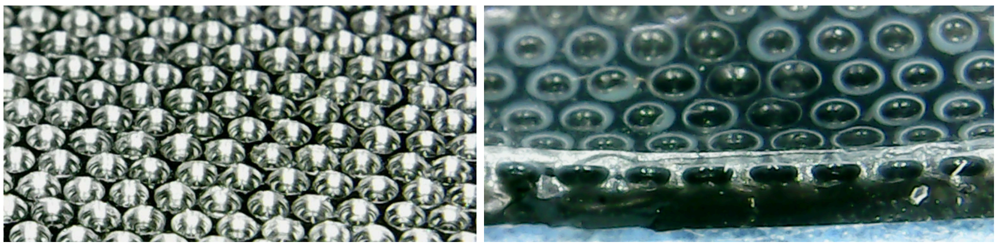


Figure 4.12.: CNC-milled dielectric mold (left), closed sensor skin cells (right).

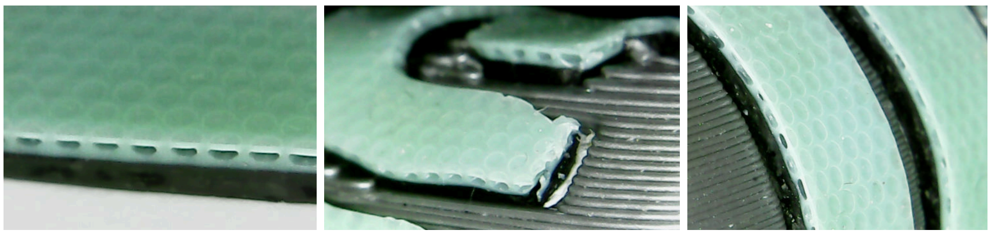


Figure 4.13.: Closed-celled dielectric structures after casting (left [292]) and on fixtures for sensor assembly (middle [292], right).

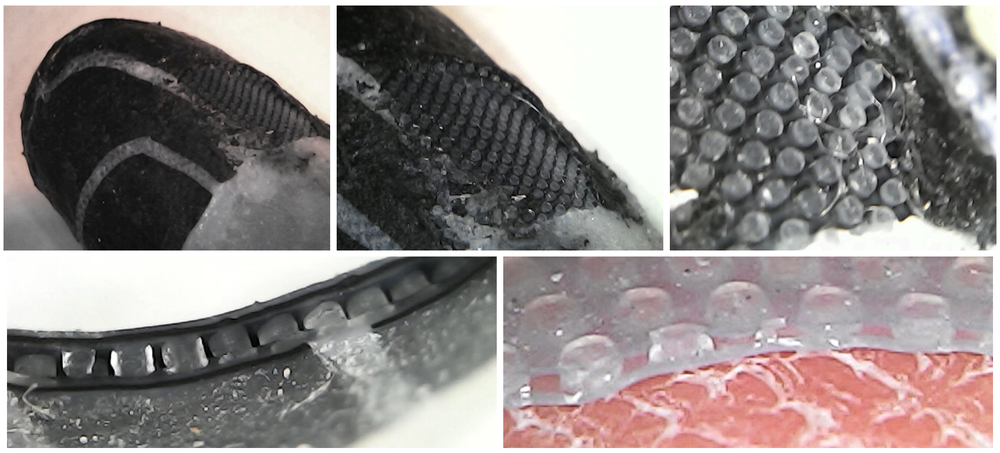


Figure 4.14.: Open-celled dielectric, partially dissected on finger sensor (top), between electrodes (bottom-left), on human skin (bottom-right). [296]

4.5.4. Computer-aided design and toolpath generation

The production of multi-curved sensor skins is first planned via computer-aided design. The shape is specified as a subdivision surface [29, 137] mesh. Sensor layout is defined via an additional polygon mesh, which is projected onto the base mesh. Molds with and without features for part alignment are derived from the base mesh and the sensor layout by projecting the part shapes onto the base mesh and extruding affected or unaffected parts. In case of in-situ molding of dielectric structures, additional molds are created via procedural displacement mapping. The meshes are modeled in Blender [225]. Part shape and mold generation are implemented using Geometry Nodes [231].

The parts shapes are subsequently unwrapped from the 3D model, projected onto a plane, arranged for efficient production and converted into tool paths for a cutting plotter. Common 3D modeling packages such as Blender [225] implement angle-preserving UV unwrapping [111] for texture mapping. However, for sensor skin production, stretch should be minimized. A custom stretch-minimizing unwrapper is developed to reduce excess mechanical strain. The problem is solved through sequential least squares. Evaluating only edge length would allow the parts to be flipped. Therefore, during each iteration, polygon alignments are computed via the Kabsch algorithm [95, 188]. To overcome local minima, gradually decaying Gaussian noise is added. See figure 4.15 for examples.

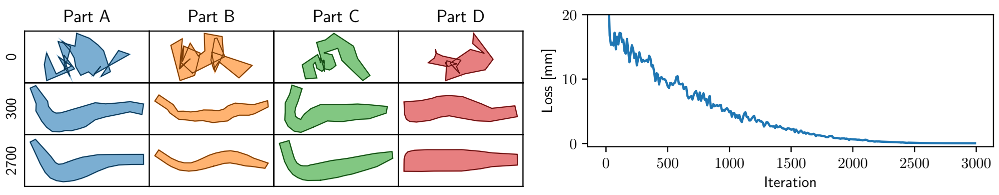


Figure 4.15.: Shapes (left) and loss (right) while unwrapping parts for a double-curved tactile fingertip sensor. [292]

After unwrapping, the part shapes are packed to minimize waste during manufacture. Popular open-source bin packing solvers are available for rectangular [89, 232] and convex shapes [233]. However, shapes for sense skin production can be highly non-convex. The parts are arranged via simulated annealing, minimizing the area of the combined bounding rectangle. Pair-wise distances between the parts are constrained to be above a fixed margin. An example for the packing process is shown in figure 4.16. The tool paths are written to SVG files and executed on a cutting plotter via an open-source Inkscape extension [228].

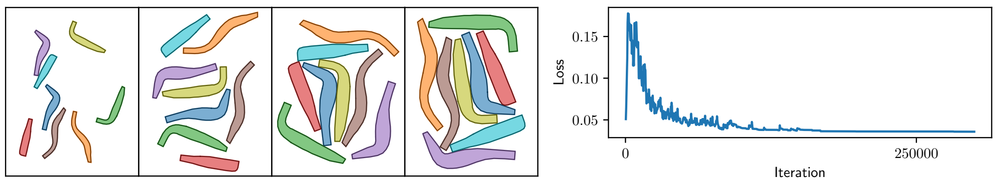


Figure 4.16.: Part shapes (left) and loss (right) for increasing iteration counts while arranging parts for manufacturing a tactile fingertip sensor. [292]

4.5.5. Sensor readout

The tactile sensor skins are read by applying excitation signals to one set of electrodes and measuring currents that are coupled across the sensor cells into another set of electrodes (see figure 4.17, right). To achieve acceptable signal-to-noise ratios from high-impedance capactive sensor cells and across cables made from electrically conductive silicone rubber, all sensor cells are sampled simultaneously and sine waves are used as excitation signals. However, the readout electronics should also be compact enough for wearable devices. Signal generation, digital-to-analog conversion, analog-to-digital conversion and filtering are integrated together on an FPGA. Delta-sigma conversion [234] is used to interface between analog and digital domains.

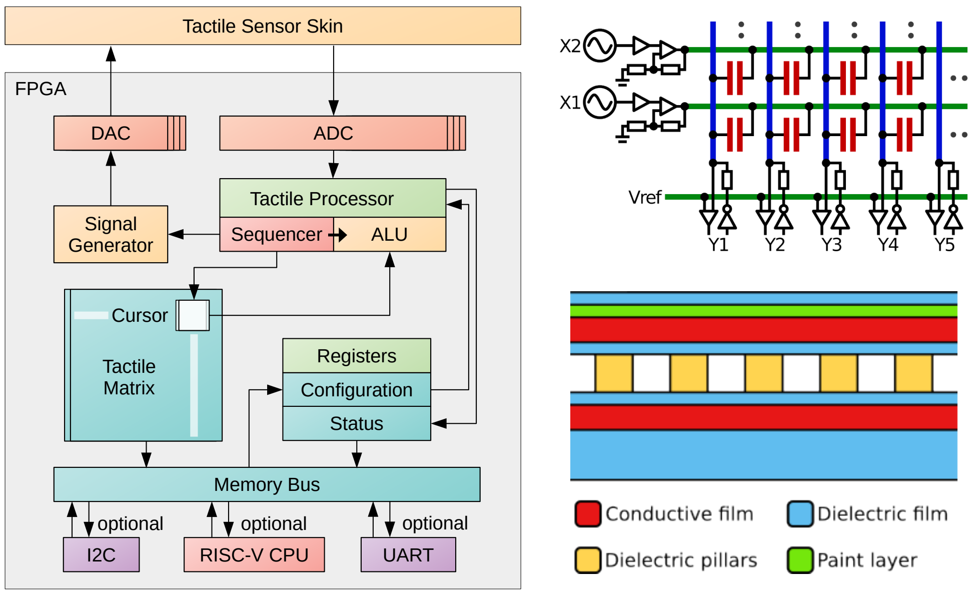


Figure 4.17.: FPGA architecture for sensor readout (left) [292], equivalent circuit of a capacitive sensor matrix with excitation drivers and receivers (top-right) [296], layer structure of the sensor (bottom-right) [296].

A DDS signal generator creates sine waves with different frequencies. These are converted to delta-sigma bit streams. In some prototypes, external RC lowpass filters and amplifiers are used to increase excitation voltage. The generated excitation signals are transmitted into the sensor matrix. Currents coupled into the excitation lines are measured by current-mode delta-sigma ADCs while keeping the reception lines at a constant voltage level to avoid crosstalk. A tactile processor (figure 4.17, left) iterates over a virtual tactile sensor matrix and correlates measurements with sines and cosines of the excitation frequencies, reconstructing complex impedances. Parameters including excitation frequencies and sampling rates can be configured via a set of registers. The configuration registers and the virtual sensor matrix can be accessed via a memory bus. Depending on the application, the memory bus interfaces with a RISC-V softcore [235], an I2C bus [236] or a UART connection. The FGPA architecture is developed and synthesized using open-source toolchains [237, 238, 239, 169, 240, 152, 241].

4.5.6. Experiments

A piece of tactile sensor skin with a cell size of 10mm × 10mm and an open-celled dielectric is placed ontop of a load cell. Different amounts of force are applied. Outputs from the tactile sensor and from the load cell are recorded. The response can be approximated by a simple rational model. The model is fit by minimizing mean squared errors [61]. See figure 4.18 for results.

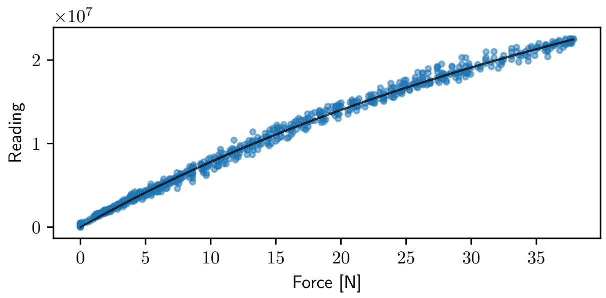
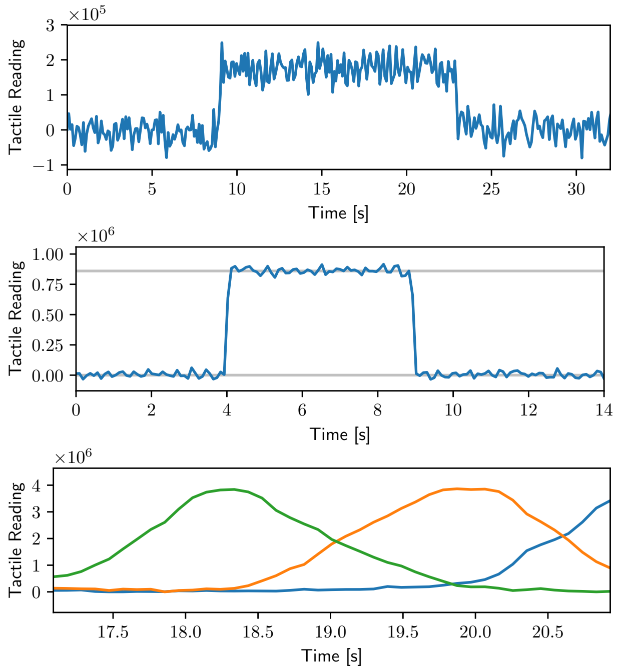


Figure 4.18.: Measurements (blue) and fitted model (black) for sensor response (Y axis) to forces (X axis). [292]

A piece of tactile sensor skin (figure 4.19, right) with a cell size of 5mm × 5mm and an open-celled dielectric is exposed to various stimuli. The sensor is connected to the readout electronics via an 8 cm long stretchable silicone cable with excitation lines on one side, reception lines on the other side and one ground trace in-between for shielding. Each conductor has a width of 0.5 mm. Different weights are placed onto the sensor and removed. Sensor response is recorded for a 1 g weight and for a 50 g weight (figure 4.19, top-left, middle-left). Both weights can be detected by the sensor. The noise is significantly lower than the response for the 1 g weight. The sensor can also be used to track the location of moving contacts. When a 2€ coin is rolled over the piece of sensor skin, different cells show slowly increasing and decreasing overlapping activations (figure 4.19, bottom-left). Figure 4.20 summarizes the response to different weights (b-e), touching the matrix (f) or cable (g) with a human finger, stretching the cable (i) or sensor matrix (j-k), and touching the cable while interrupting the ground connection to the shield trace (h). The sensor can detect and distinguish forces across multiple orders of magnitude. A small amount of crosstalk between pressure and stretch can be observed. However, for practical applications, it should be below typical contact forces and could be further compensated through multimodal integration with motion tracking. The ground trace between the transmission and reception bundles reduces interference from mutual capacitance.



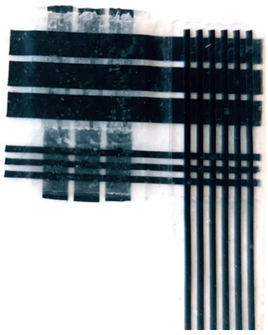


Figure 4.19.: Response to a 1 g weight (top-left), a 50 g weight (middle-left) and a rolling contact (bottom-left) from a piece of sensor skin (right). [292]

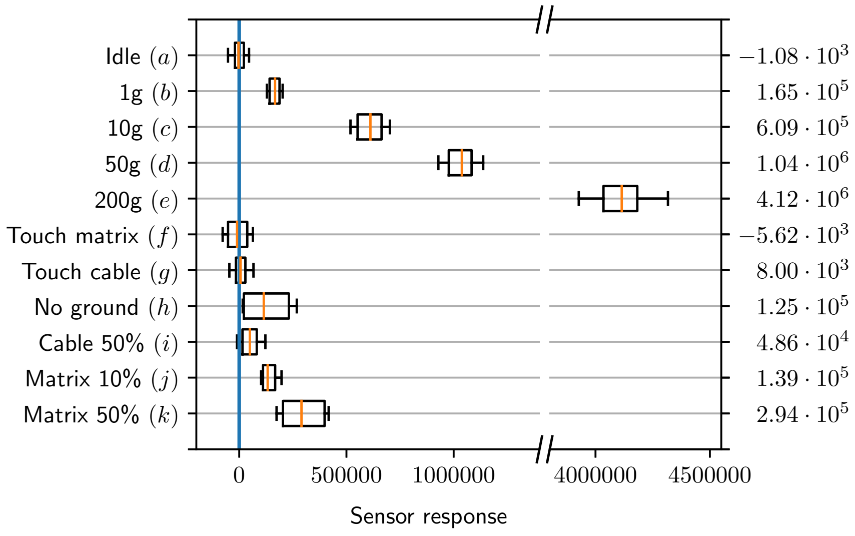


Figure 4.20.: Averages (right), median (orange), quartiles (boxes) and 5 % quantiles for the sensor response to various stimuli (left). [292]

4.6. Wearable multimodal fingertip

A first fingertip prototype features a double-curved 4-by-4 sensor matrix with 16 tactile cells. The readout electronics is integrated locally above the finger nail, minimizing potential signal interference, attenuation and crosstalk. This also provides opportunities for integration with other modalities. The prototype readout board is equipped with an IMU and an LED, which can be used to support motion tracking and vibration measurement.

A model of the tactile sensor matrix is created using Blender [225]. The fingertip and electrode shapes are specified via subdivision surfaces [29, 137]. The part shapes are projected onto the fingertip shape. Assembly fixtures are generated by extruding and subtracting the part shapes from the fingertip model. The part shapes are processed as described in section 4.5.4. In the first foil-based sensor prototypes described in section 4.4, the rows and columns of the sensor matrix are aligned in lateral and longitudinal directions, allowing for easy mathematical description in OpenSCAD, but leading to less even coverage at the tip. For the sensor skin prototype introduced in this section, the rows and columns wrap around the tip at a 45° angle, with one corner of the matrix at the tip, one at the base, and two corners at the left and right of the fingertip. See figure 4.21 for renderings of the assembly fixture and the electrode layout.

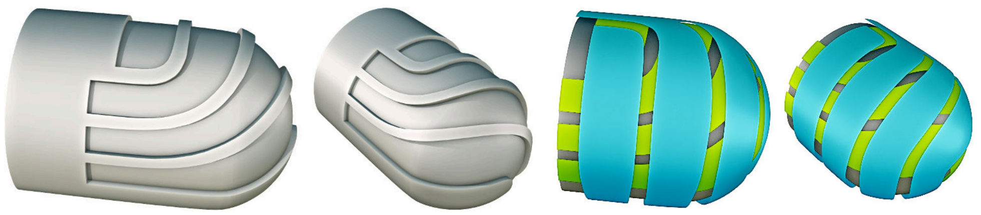


Figure 4.21.: Assembly fixture (left) and electrode layout (right) for a double-curved fingertip sensor [292].

A conductive silicone film is cast and cut into two halves. One half is covered with a closed-celled dielectric pattern. A hexagonal grid pattern is milled into an aluminum plate. A negative casting with a hexagonal pin pattern is created using polyurethane resin. The polyurethane casting is used to cast a silicone honeycomb structure onto one of the conductive silicone films. To close the honeycomb structure, a film of non-conductive silicone is dispensed onto a flat plate using a blade and the honeycomb structure is pressed against the partially cured silicone film. The silicone films and structures are cut according to the unwrapped part shapes using a cutting plotter. The inner electrodes are cut from the bare conductive film and the outer electrodes are cut from the conductive film with the dielectric honeycomb structure.

The silicone electrodes are connected to the readout board via solder tabs. One side of a copper sheet is coated with a galvanic tin layer and the sheet is cut into small rectangular pieces. The bare copper surfaces of the solder tabs are glued to the silicone electrodes using small amounts of additional conductive silicone and a primer[25]. The silicone electrodes are connected to the readout board by soldering the metal tabs onto corresponding solder pads on the readout board (figure 4.22, left). The electrodes are glued together with non-conductive silicone and onto a hollow silicone fingertip (figure 4.22, right). The assembly fixture is printed using an FFF printer.

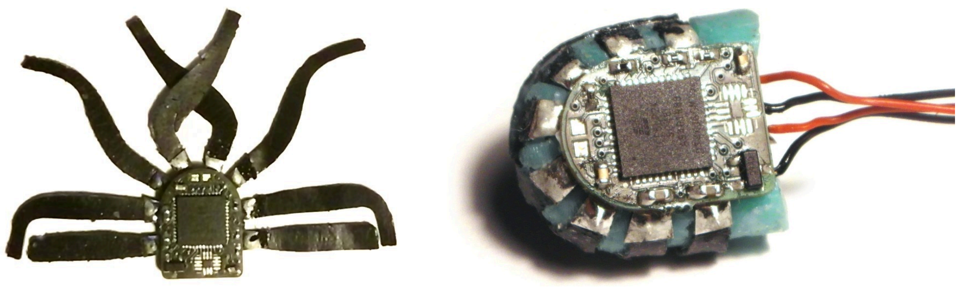


Figure 4.22.: Silicone electrodes are soldered via metal tabs to a finger-nail readout board (left) and assembled into a double-curved wearable fingertip sensor (right) [292].

The readout board above the finger nail carries a small low-power FPGA[26]. The FPGA reads the sensor matrix directly via delta-sigma modulation as described in section 4.5.5. In addition to the FPGA itself, only one reference resistor is required for each reception line. Additional lowpass filters, which would typically by integrated in delta-sigma ADCs and DACs are not required. The digital DAC bitstream can be directly fed to the excitation lines. ADC comparators and feedback paths can be directly connected to the reception lines. Impedance properties of the conductive silicone provide sufficient filtering. Four wires connect the fingertip electronics to a host microcontroller[3]. Two of the wires provide power and two additional wires are used for data and programming. During startup, the two signal wires act as clock and data connections for uploading the FPGA bitstream in SPI mode. At runtime, the wires form an I2C bus to exchange configuration and sensor data, providing access to the tactile sensor, IMU and tracking LED. The fingertip sensor can be worn on a human hand. The tactile sensor records contact forces and locations while the IMU records motion and vibration data (see figure 4.23).

[25]Wacker Primer G790, Wacker Chemie AG, Germany
[26]ICE5LP1K, Lattice Semiconductor Corporation, Hillsboro, OR, USA

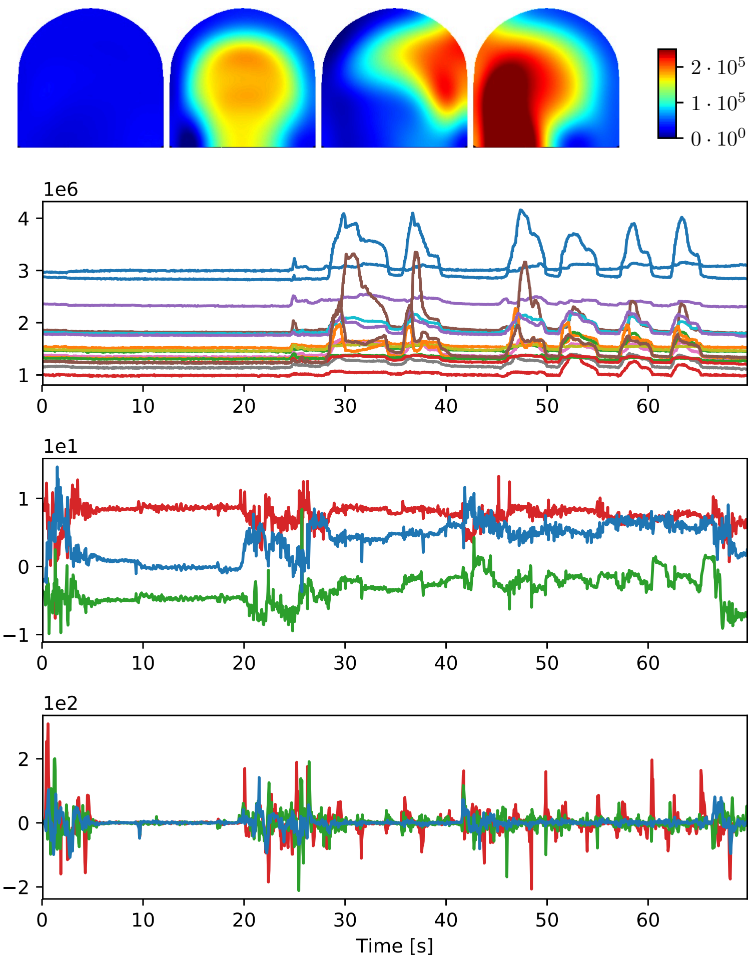


Figure 4.23.: Tactile and IMU data recorded from a human finger using a wearable fingertip sensor during object manipulation [292].

4.7. Wearable finger sleeve

The fingertip sensor design is adapted to also cover the middle and proximal phalanges. The finger sleeve is constructed without rigid parts. Not only the sensor electrodes but also the cables connecting the transducers to the readout electronics are made from electrically conductive silicone rubber. Parallel strips of conductive silicone are embedded into the sleeve along the back of the finger and extend past the proximal joint as a cable. The cable connects to a readout board, which can be placed on the wrist. The wearable finger sensor is assembled inside-out with an in-situ molded open-celled dielectric structure.

Fixtures and part shapes are designed using the same methods as the fingertip sensor. An additional mold for in-place dielectric casting is generated through variable displacement of mesh vertices along the surface normals. Two smooth skin layers are cast by pouring liquid silicone rubber over finger molds and letting the silicone cure. Conductive silicone films are cast on adhesive carrier foils and cut to form the sensor electrodes. The carrier film with the conductive silicone is cut to form the sensor electrodes and the parts are attached to the assembly fixtures via the adhesive carrier films. One of the previously prepared skin layers is glued over the assembly fixture and onto the electrodes with liquid silicone rubber. After curing, the skin is removed from the fixture and the adhesive plastic foils are removed from the electrodes. The same process is repeated to create a second layer of skin with perpendicular electrodes. The dielectric mold is covered with liquid silicone rubber, one of the sensor skin layers is pulled over the mold, and the silicone is cured, forming the dielectric pattern. The outer layer is finally glued onto the inner layer, with the dielectric pattern forming small pressure-sensitive cavities. A silicone flat cable is prepared by cutting electrically conductive silicone into parallel strips and filling the space between the conductive traces with non-conductive silicone. The cable is then glued to the back of the finger sensor. The cable extension is reinforced and insulated by an additional layer of non-conductive silicone. Electrical connections between parts on different layers (cable traces and sensor electrodes) are created by punching holes with a hollow needle and filling the holes with electrically conductive silicone. A small piece of inelastic plastic foil is glued onto the end of the silicone cable to ensure correct contact alignment when attaching the cable onto a rigid circuit board. Exposed conductive silicone is insulated with non-conductive silicone. The silicone is post-cured in an oven for multiple hours. The end of the cable is attached to a readout board using a 3D-printed mechanical clamp.

Attempts to read the finger sensor across the silicone cable using similar circuitry as for the sensor fingertip fail, producing mostly noise. An obvious explanation could be limited ADC accuracy. However, adding additional transimpedance amplifiers fails to improve signal-to-noise ratio. Increasing the excitation frequency results in increased signal attenuation and decreasing the excitation frequency also reduces the response from the capacitive sensor cells. Therefore, the excitation amplitude is increased by amplifying the excitation signals to approximately 18 V. Linear analog amplification of the excitation signals would first require low-pass filtering of the delta-sigma bitstream. To save board space, the delta-sigma bitstreams are amplified directly using digital drivers[27].

The finger sensor can be used to measure contacts on the distal, middle and proximal phalanges of a human finger (fig. 4.24) or on a humanoid robot finger (fig. 4.25). Both the sensing elements and the silicone cable can be stretched by multiple times without damage. Impedance measurements from the sensor cells reveal that the open-celled dielectric improves sensitivity and reduces crosstalk between pressure and stretch compared to the closed-celled dielectric in the previous fingertip sensor. However, the in-situ-molded dielectric suffers from higher variation in sensitivity between cells.

[27]MCP1415, Tiny 1.5A, High-Speed Power MOSFET Driver, Microchip Technology Inc., AZ, USA

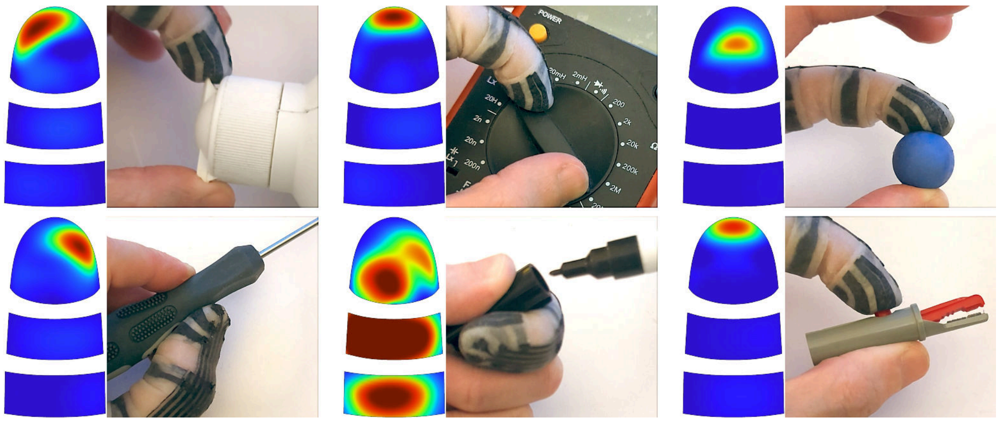


Figure 4.24.: Recording contact information from a human finger during object manipulation using elastic tactile sensor skin [292].

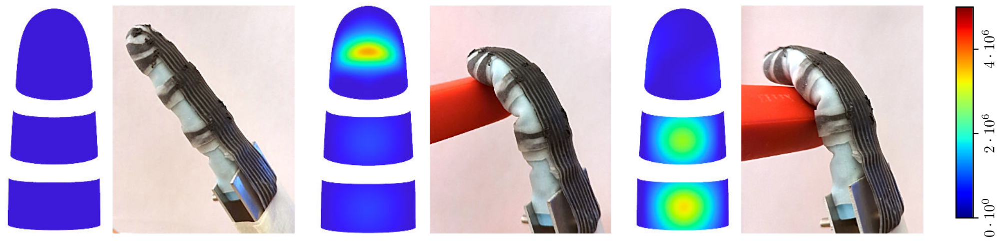


Figure 4.25.: Elastic tactile sensor skin applied to a tendon-driven humanoid robot finger [292].

4.8. Elastic tactile teaching glove

In order to simultaneously measure contact information on all five fingers and the palm during teaching by human demonstration, a full sensor glove with tactile sensor skin is developed. The glove, including sensor electrodes and electrical connections, is completely made from stretchable silicone rubber. The sensor cells are produced with an open-celled separately-molded dielectric to minimize thickness and pressure-stretch crosstalk while maximizing sensitivity. The base layer of the glove, electrodes, conductors and insulating layers are cast using the final solvent-based formulations (section 4.5.2). Readout electronics are placed on the wrist to avoid obstruction, interfacing with the sensor array via an elastic flat cable made from the same materials as the glove.

4.8.1. Design

The glove is first designed as a virtual CAD model (figure 4.26). The underlying hand shape is generated using MakeHuman [242], a parameterizable subdivision-surface human model. The finger sensor models developed in the previous section are projected onto the hand model. The palm of the hand model is retopologized accordingly and connected to the base of each finger. A multi-curved 6-by-6 palm sensor with a 2-by-3 cutout between thumb and first finger is designed. The sensor matrices are modeled as closed quad meshes and the electrodes are separated programmatically to ensure even spacing. Flat cables, consisting of parallel conductors, are modeled as quad meshes as well, with alternating material assignments for gaps and conductors. Cables to the fingers and palm are joined at the back of the hand, merging into a single flat cable along the upper side of the wrist. Smooth subdivision surfaces [29, 137] are generated for electrodes, cables and sensor dielectrics, and projected onto the hand mesh. Alignment markers are generated following the outlines of the part shapes and subtracted from the base mesh in order to generate a mold for casting and assembly. Electrode and dielectric shapes are exported as OBJ files. The shapes are unwrapped and compacted as described in section 4.5.4.

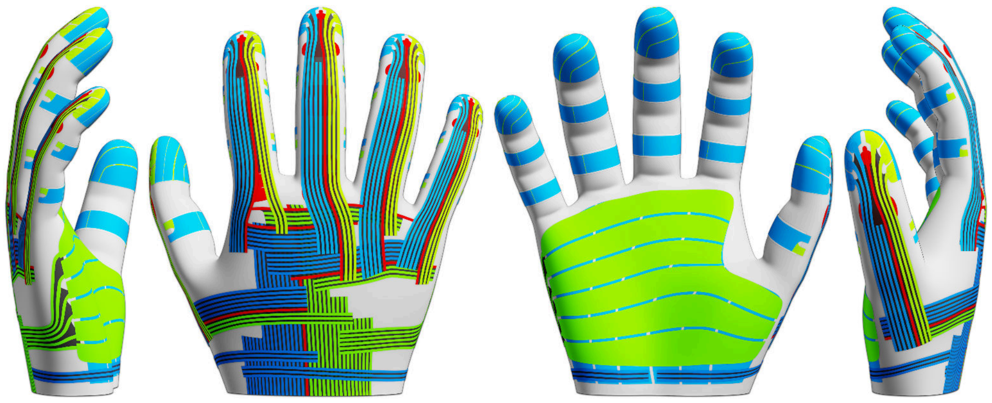


Figure 4.26.: CAD model of a tactile sensor glove [296]. Electrodes for sensor rows and columns are shown in green and blue, spacing constraints in black and ground electrodes in red.

4.8.2. Assembly

The glove is manufactured by casting a continuous base layer, casting and cutting electrode and dielectric parts, gluing the parts to the glove, creating vertical connections between layers, and finally applying layers of paint and varnish. See figure 4.27 for an overview of the production process.

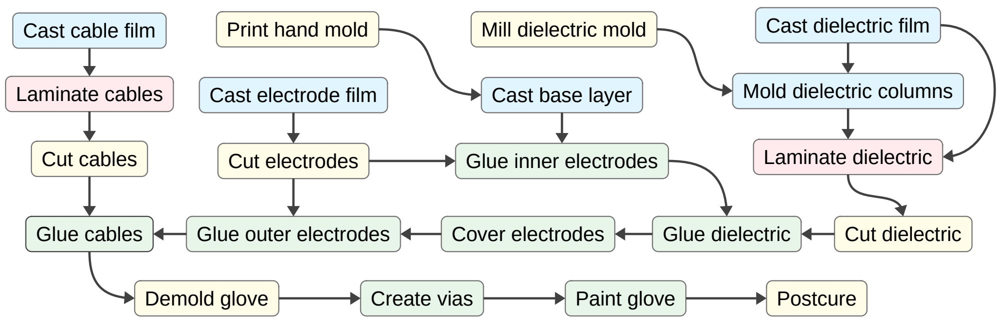


Figure 4.27.: Productions steps for manufacturing a tactile sensor glove [296].

The hand mold is printed using an SLA printer. Alignment markers are created by printing the mold in a black resin and subsequently filling printed grooves with a white putty[28]. The mold is covered in a plastic primer[29] and a clear varnish[30] to ensure a smooth inner surface and to avoid cure inhibition.

The base layer of the glove is produced by preparing a sufficient amount of the solvent-based dielectric silicone, pouring it over the mold, and letting it dry and cure for approximately one hour. The parts are attached to the glove (fig. 4.28) by applying small amounts of solvent-based silicone with a brush as glue, letting the solvent evaporate, pressing the parts onto the glove, and letting the silicone cure.

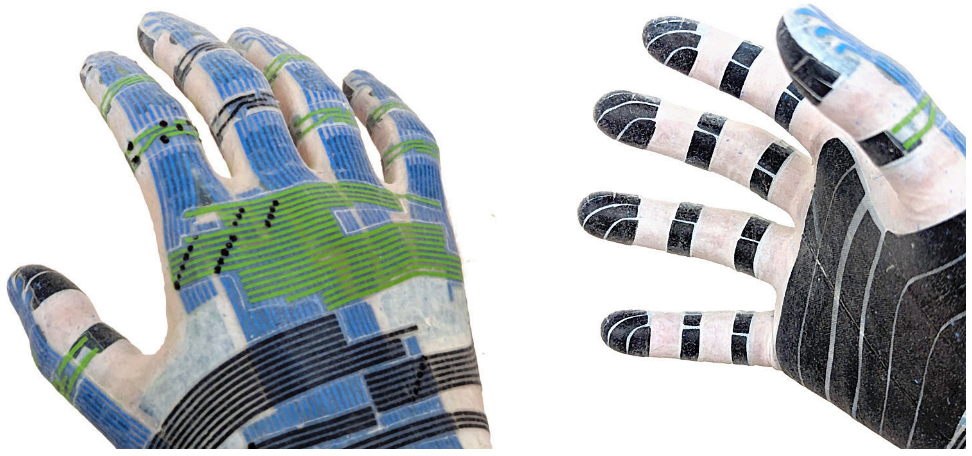


Figure 4.28.: Partially finished sensor glove on a human hand, with sensor elements and conductor bundles transfered to the glove surface. A subset of conductive traces is colored in green and blue for easier identification [296].

[28]Molto Holzkitt, Bauhaus AG, Switzerland
[29]OBI Kunststoff-Grundierung Spray Transparent matt, OBI Home and Garden GmbH, Germany
[30]DUPLI-COLOR 385858 CAR'S, European Aerosols GmbH, Germany

Electrodes and cable conductors are first cast as flat sheets by spreading solvent-based conductive silicone over a flat surface with a blade and letting the silicone dry and cure. Small amounts of the silicone paint formulations can be applied to the conductive sheets for easier identification during assembly. The parts are then cut with a cutting plotter[5] according to the tool paths generated from the CAD model. Vertical connections between conductive traces and sensor electrodes on different layers are created by first removing the glove from the mold, filling it with paper, dipping a needle in solvent-based conductive silicone and repeatedly sticking the needle through the sensor skin.

The dielectric structures are prepared by first casting a thin layer of dielectric silicone on a flat plastic sheet, then casting the dielectric columns onto the silicone film using a CNC-milled aluminum mold, and finally covering the dielectric grid with a second closed dielectric film on the opposite side. The dielectric is cut using the cutting plotter[5] and glued to the glove. After all parts are assembled and checked for connectivity using a multimeter or an attached tactile readout board (see figure 4.29), multiple layers of silicone paint and clear conductive silicone as varnish are applied. The end of the cable extension is glued to a small strip of plastic foil to ensure correct alignment when attaching the readout electronics. The glove is finally baked for post-curing. The end of the cable is clamped to a readout board, which is housed inside a wrist-worn readout module. See figure 4.30 for a photo of the fully assembled sensor glove.

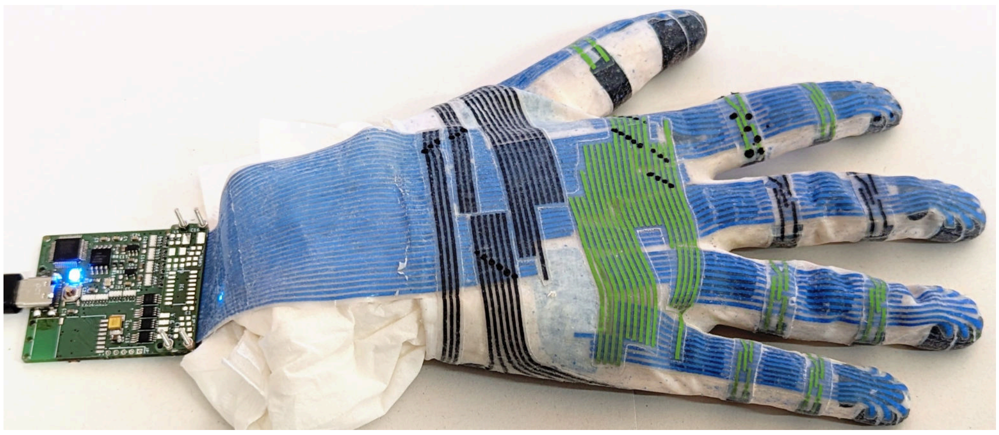


Figure 4.29.: Tactile sensor glove during assembly with attached readout board for diagnostics [296].

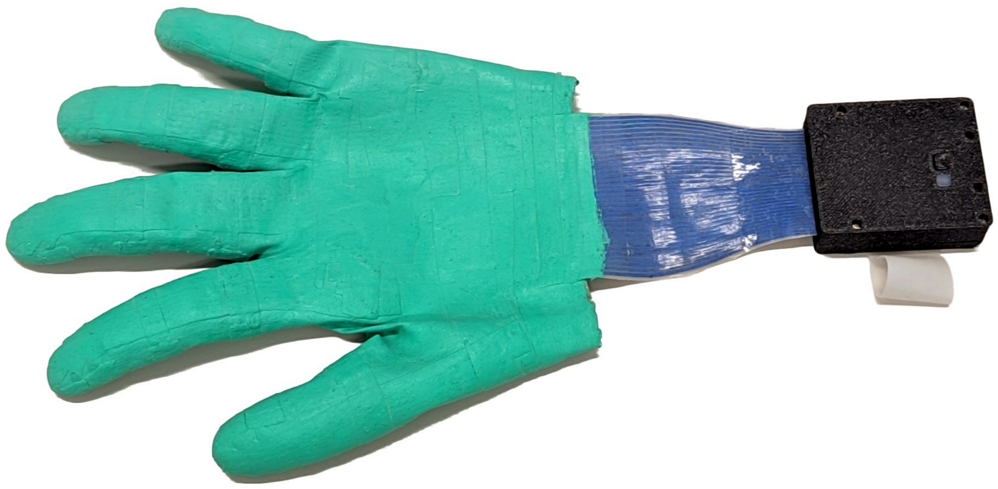


Figure 4.30.: Fully assembled tactile sensor glove [296].

4.8.3. Readout module

The sensor glove connects to a readout board inside a wrist-worn readout module (see figure 4.31). The module is powered by an integrated battery. The board carries an FPGA[31], analog parts for tactile sensor readout, a radio transceiver[32], an IMU, and supporting components. The readout module can be worn by a silicone wristband.

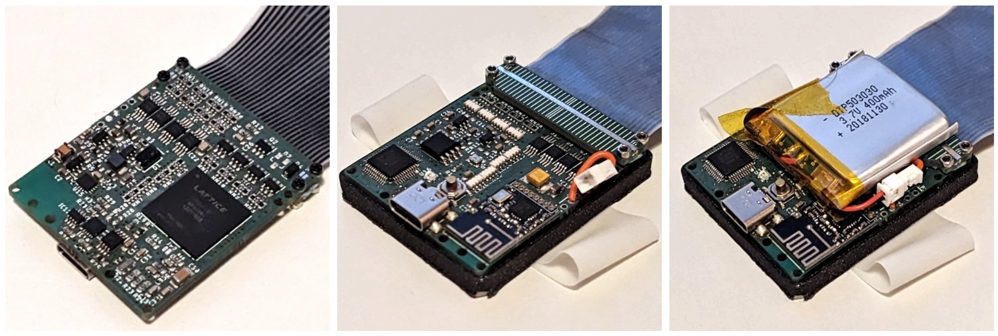


Figure 4.31.: Readout module for the tactile sensor glove [296].

The stretchable flat cable is connected to a row of electrodes on the readout board using a mechanical clamp. The Z-conductive tape used in some of the foil-based sensor prototypes does not sufficiently adhere to the silicone. The module supports 16 excitation and 16 reception lines for reading tactile matrix sensors. The digital amplification

[31]LFE5U-12F-6BG256C, Lattice Semiconductor Corporation, Hillsboro, OR, USA
[32]nRF24L01, Nordic Semiconductor, Trondheim, Norway

scheme in section 4.7 results in excessive power consumption. Thus, for the battery-powered glove readout module, discrete analog low-pass filters and analog amplifiers[33] are added to the excitation paths. Excitation amplifiers are powered by a 30 V boost converter and the FPGA is powered by a 1.2 V buck converter. Both the excitation and reception paths include series resistors for current limiting. The reception paths feature additional diode arrays to protect against short circuits between excitation and reception lines. The battery is managed by an on-board battery controller and can be charged via a USB-C port. The USB port can also be used for programming of an on-board SPI flash and for JTAG debugging. A common USB bridge IC[34] is used for compatibility with existing open-source programmers and debuggers [243, 152]. Sensor data can be transmitted wirelessly via a radio transceiver[32].

4.8.4. Experiments

The sensor glove is worn on a human hand while handling several objects and tools. Tactile sensor readings are recorded together with corresponding images from an RGB camera. The tactile readings are projected onto the original CAD model with bicubic interpolation [101, 165] and rendered using a colormap (as in figure 4.25). See figure 4.32 for results. During this experiment, the glove is functionally complete but not yet painted, partially revealing the internal circuitry.

Grasping the handles of a set of pliers, a knife, manual screw drivers and an electric screw driver leads to responses at the contacts. When lightly touching the trigger of the electric screw driver with the first finger, the corresponding sensor cells show a weak response. When pressing the trigger further to activate the screw driver, a stronger response can be observed. At first perhaps surprisingly, a similarly strong response can be observed at the ball of the thumb (figure 4.32, bottom-left). The force exerted by the first finger while pressing the trigger has to result in an equal counteracting force [135].

[33]TLV9362, Texas Instruments, Dallas, TX, USA
[34]FT232HL, Future Technology Devices International Ltd., UK

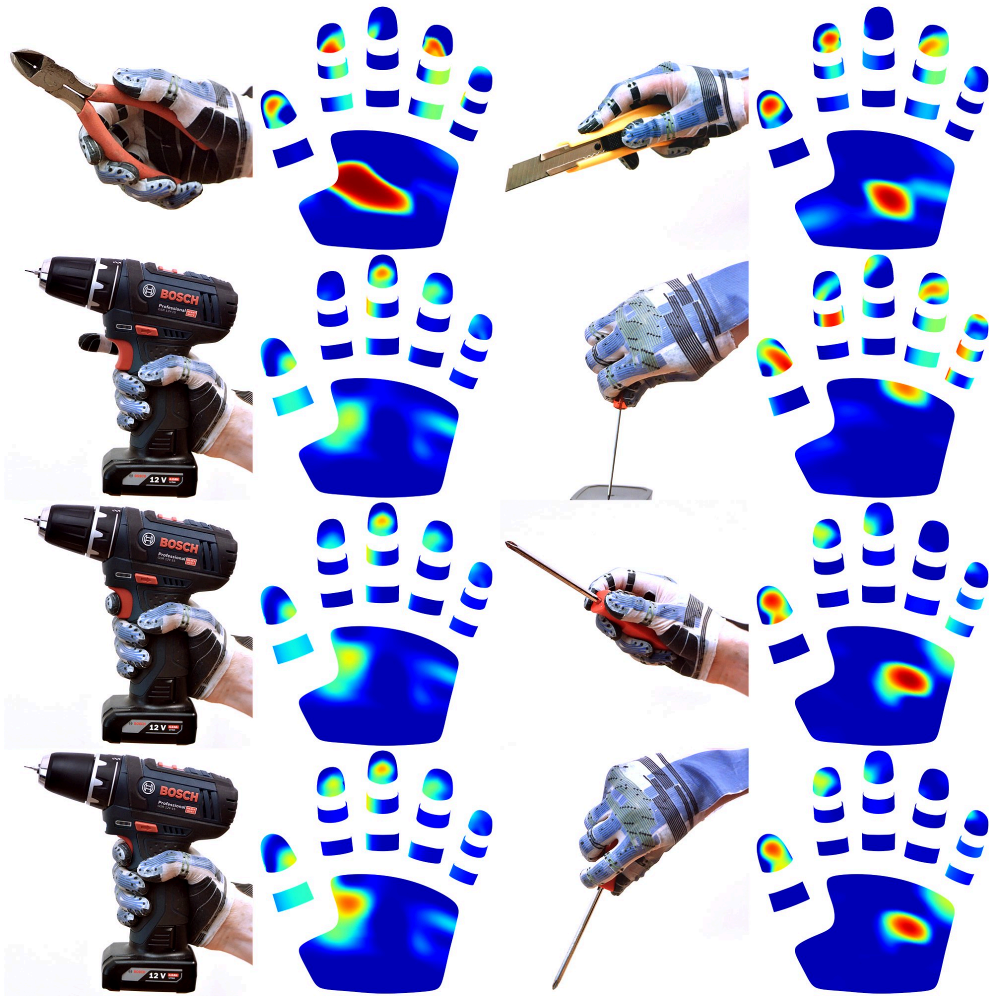


Figure 4.32.: Tactile measurements (2nd and 4th column) captured from a human hand with corresponding photos (1st and 3rd column) while handling a set of pliers, a knife, manual screw drivers and an electric screw driver. Images in row 1 and 2 first published in [296].

In order to qualitatively validate material properties, multiple fingers are each stretched to multiple times their original length as shown in figure 4.33. The trials are performed with a fully assembled and painted glove. When released, the fingers return to their original shape and the sensor cells remain functional. The following experiments are performed after the stretchability tests.

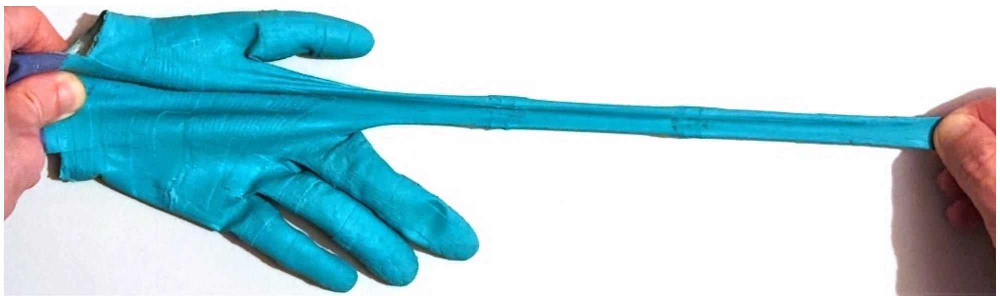


Figure 4.33.: The teaching glove, including conductors and sensors, is completely made from highly stretchable silicone rubber [296].

Tactile sensor output is compared to force measurements from a commercial force-torque sensor[35]. While wearing the sensor glove, a human repeatedly presses the first finger against the force-torque sensor and forces are recorded together with tactile data. See figure 4.34 for tactile recordings from three adjacent cells and forces measurements over time. The tactile readings track the measured forces with little drift or hysteresis. Measurements from cell 142, which is closest to the contact and shows the strongest response, agree with force measurements with a correlation coefficient [181] of 0.99.

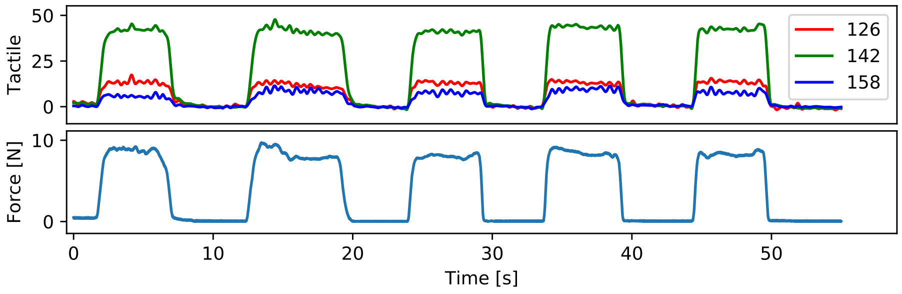


Figure 4.34.: Readings from three different sensor cells on a tactile sensor glove with corresponding force measurements while repeatedly pressing against a force-torque sensor with the first finger [296].

The experiment is repeated with lower contact forces of approximately 0.25 N. For both sensors, the filter bandwidth is configured to 5 Hz. Figure 4.35 shows measurements from a single tactile cell at the contact and from the force-torque sensor over time. Compared to the commercial force-torque sensor, the tactile sensor shows a similar or lower amount of noise and drift.

[35]ATI Mini45, ATI Industrial Automation, USA

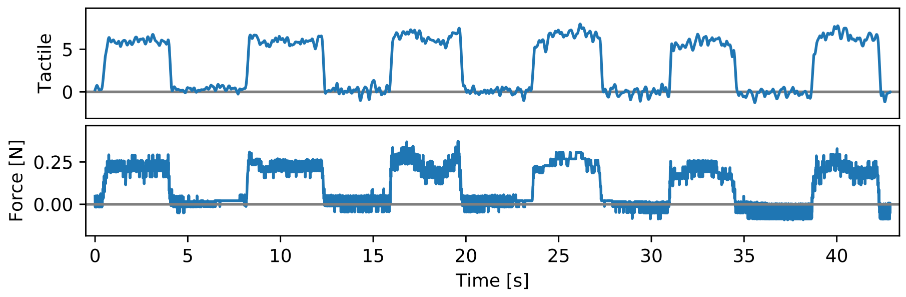


Figure 4.35.: Tactile readings and corresponding force measurements while lightly touching a force-torque sensor with the first finger [296].

Slowly increasing and decreasing pressure is applied with the first finger against the force-torque sensor while wearing the sensor glove. Figure 4.36 shows tactile readings for 5 different sensor cells plotted against force measurements. The response can be approximately modeled as a linear relationship (figure 4.36, red). The tactile cell closest to the contact point (cell 142) shows a relative non-linearity [47] of 0.039.

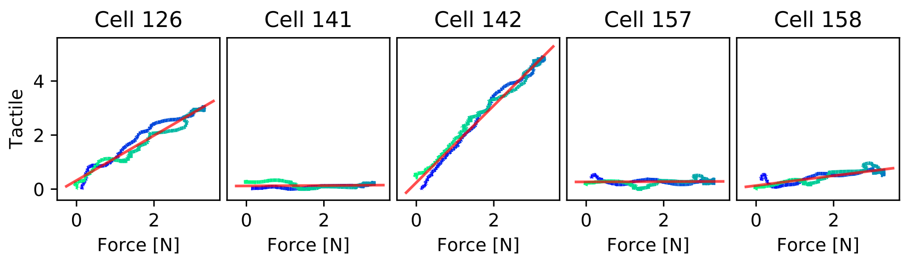


Figure 4.36.: Response (blue, green) of five different tactile cells for slowly increasing and decreasing forces with linear approximation (red). [296]

5. Multimodal demonstration capture

During teaching by human demonstration, visual observations and tactile data are captured and pre-processed to obtain spatio-temporal reconstructions for efficient learning, interpretability and human-to-robot transfer. The measurements can be corrupted by noise, outliers and occlusions, which should be suppressed during the reconstruction process. However, the tasks should still be learned from demonstration data without forcing the user to explicitly build geometric object models.

5.1. Related work

Motion capture has become a frequently-used tool in many fields from entertainment to robotics. Motions of humans and objects can be captured using active LED markers and line cameras [244] or webcams [161]. Hand motions can also be recorded using colored spherical markers and webcams [154] or other wearable devices [164, 245, 155]. Known two-dimensional patterns can be used to track motions in three-dimensional space using only a single camera and to improve unique identification [156, 200, 141, 190]. An early approach towards reducing obstruction by discrete markers uses a multi-colored glove and machine learning with a nearest-neighbor method [191]. Deep learning can detect keypoints on humans without markers or gloves [120, 118, 173, 194, 9, 134, 131]. These can be used to regress articulated hand poses [160, 143, 177, 207, 147, 23, 15]. Visual feature detectors and descriptors can also be designed or trained to recognize stable keypoints on objects [119, 159, 44, 43, 246]. Several existing datasets are available with visual recordings of human object manipulation [121, 247, 139]. Methods can be extended beyond pure motion capture to also include other modalities. Performance capture records body movements together with voice and facial expressions for animation and game development [248, 249]. Some works record tactile or force data using hand-held gadgets [199, 200, 193, 66]. Other combinations include vision and IMUs [121] or RGB cameras with IMUs, a smart watch and a fitness tracker [247].

5.2. Low-cost visuo-tactile data recording

As a first low-cost prototype for a visuo-tactile teaching setup, the fabric-based glove prototype from section 4.2.1 is combined with a webcam, object instrumentation and fiducial markers. Both the object and the glove are equipped with metal fins and Aruco tags for motion tracking. One marker is placed at the tip of each finger and can be used to track position and orientation. The camera is calibrated and localized using an Aruco board [156]. See figure 5.1 for a visuo-tactile recording of a bottle-opening task. A human approaches the object with a hand, performs repeated turning motions to open the lid, grasps it, and places it next to the bottle. The data can be used to successfully train a robot (see section 7.6.7). However, the object has to be modified with tracking markers and electrodes, the single-camera setup is susceptible to occlusions, and the workspace is further restricted by the resolution of the webcam and the resolution required to identify the fiducial markers. With a 720p webcam, the usable workspace is approximately 40 cm wide. Additionally, the method suffers from noise and outliers. Figure 5.1 shows noticeable outliers along the Z-axis. Limited frame rate and motion blur, and the reliance on spatial patterns for marker identification, also limits velocities, requiring two minutes for demonstrating the bottle opening task (figure 5.1). Faster motions result in failures to detect and identify the fiducial markers due to motion blur.

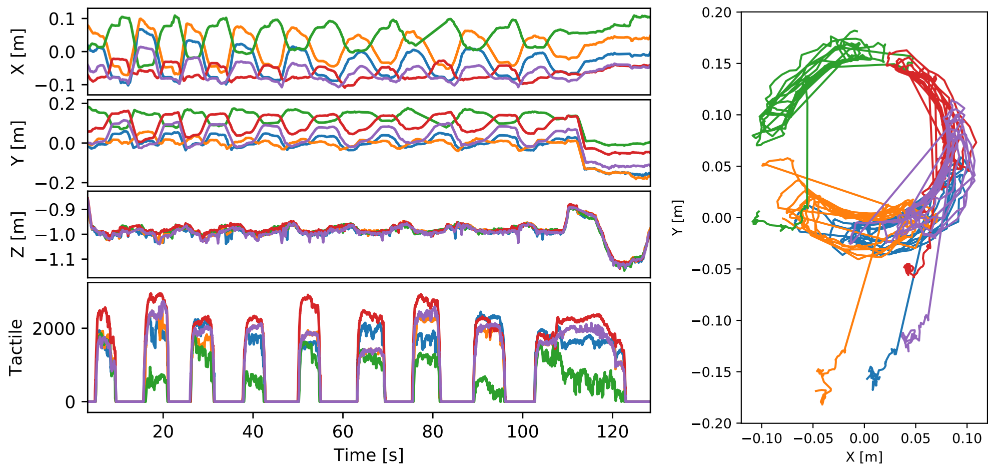


Figure 5.1.: Human fingertip positions (top-left, right) and tactile data (bottom-left) recorded using a glove with passive markers and fabric-based contact sensors for a bottle opening task.

5.3. Robust marker-based trajectory reconstruction

In order to increase temporal and spatial resolution as well as the workspace, hand motions are captured with a Phasespace Impulse X2 motion tracking system [244]. A glove is equipped with eight LEDs as tracking markers, one at each fingertip, one at the base of the hand close to the wrist, one at the base of the first finger, and one at the base of the little finger. The markers are observed by 10 pairs of perpendicular high-resolution line cameras. The line cameras are arranged around a table with a size of approximately 1.2 m × 1.2 m. The motion tracking system can output raw 1D observations from the line sensors and 3D reconstructions. While less noisy than the data obtained using the webcam and fiducial markers in section 5.2, the 3D reconstructions still suffer from gaps and outliers (see figure 5.2). The problem remains even after re-calibrating the system multiple times. Motion reconstruction is therefore performed directly from the 1D line camera detections via trajectory optimization.

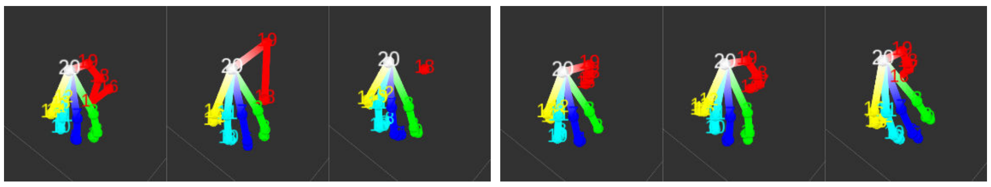


Figure 5.2.: Hand pose reconstructions using only the original software (left) and using the proposed trajectory optimization method (right).

For each marker i and time step t, a position vector P_{mit} is optimized. Each line camera j is modeled by a position vector P_{cj}, an orientation Q_{Cj} and polynomial distortion coefficients d_{jk}. Camera orientations are represented as rotation matrices and rotational derivatives as 3D rotation vectors. Observations O_{tij} consist of positions on the line sensors. Errors e_{Rtij} between observations O_{tij} and projections J_{tij} of reconstructed positions P_{it} are minimized via a robust [87] loss function L_{Rtij} with smoothness δ.

$$J_{tij} = \frac{\left(Q_{Cj}^{-1}\left(P_{mit} - P_{cj}\right)\right)_0}{\left(Q_{Cj}^{-1}\left(P_{mit} - P_{cj}\right)\right)_2} \tag{5.1}$$

$$e_{Rtij} = J_{tij} + \sum_k d_{jk} O_{tij}^{2k} - O_{tij} \tag{5.2}$$

$$L_{Rtij} = \begin{cases} \frac{1}{2} e_{Rtij}^2 & \textbf{if } \|e_{Rtij}\| < \delta \\ \delta\left(\|e_{Rtij}\| - \frac{1}{2}\delta^2\right) & otherwise \end{cases} \tag{5.3}$$

In order to fill gaps and reject outliers, additional kinodynamic terms are added. Regularizers L_{Vit} and L_{Ait} promote continuity and smoothness. A shape loss L_{Silt} is applied between adjacent joints with indices i and l.

$$L_{Vit} = \left(\frac{dP_{mit}}{dt}\right)^2 \qquad L_{Ait} = \left(\frac{d^2 P_{mit}}{dt^2}\right)^2 \qquad L_{Silt} = \left(\frac{dP_{mit}}{dt} - \frac{dP_{mlt}}{dt}\right)^2 \tag{5.4}$$

The reconstruction process minimizes a time integral over a weighted sum of the above loss terms with weights w_L. For initial calibration, not only the marker positions P_m but also the camera parameters P_c, Q_C, d are optimized.

$$\min_{P_m[,P_c,Q_C,d]} = \sum_i \int_t \left(w_{L1} L_V + w_{L2} L_A + \sum_j w_{L0} L_{Rtij} + \sum_l w_{L3} L_{Silt}\right) \tag{5.5}$$

The trajectory optimization problem is solved via sequential least squares. While computationally more expensive and currently not realtime capable, the trajectory-based reconstruction method can successfully suppress outliers and fill short gaps from finger occlusion in demonstration trajectories. See figure 5.2 for examples of artifacts in the original triangulations and solutions from trajectory optimization.

The trajectory optimization method can be used to obtain demonstration trajectories without gaps or outliers for dexterous manipulation tasks. See figure 5.3 for a recording of a similar bottle opening task as in section 5.2.

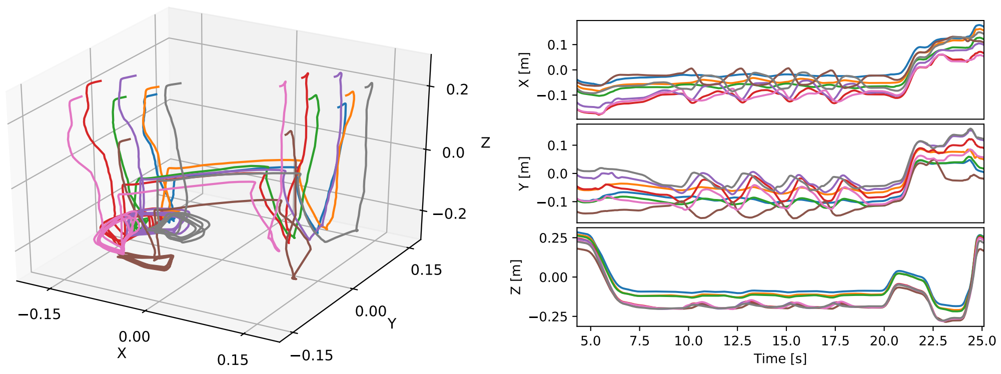


Figure 5.3.: Hand motions reconstructed via trajectory optimization for a multi-fingered bottle opening task. Trajectories are plotted in 3D (left) and as coordinates over time (right).

5.4. Multimodal sensor system

Hand motions should be recorded together with visual observations of the objects and measurements from the sensor glove. All modalities should be accurately synchronized. An FPGA-based multimodal data acquisition system is developed, consisting of multiple camera modules, a central hub module and a radio module for communication with the glove. To simplify setup for robot teaching, each peripheral module is connected to the sensor hub via a single cable carrying power, data and synchronization signals. Region of interest and resolution of the camera readout can be changed at runtime while maintaining accurate synchronization, allowing the field of view to dynamically follow the hands of a human teacher along with the sensor glove.

The modules are connected using regular RJ-45 cables. Data is transferred via two bi-directional LVDS (low-voltage differential signaling) lanes. The remaining wires are used for power supply. Since bandwidth usage can be highly asymmetric, such as for downstream camera images and upstream configuration and synchronization, data transfer is performed in half-duplex mode. Each module is equipped with an FPGA[1] and SPI flash memory for configuration. A JTAG port is provided via a 5-pin JST

[1]LFE5U-12F-6BG256C, Lattice Semiconductor Corporation, USA

SH connector[2] for programming and diagnostics. The VLSI design is developed in Verilog and synthesized using the open-source tools Yosys [237], Project Trellis [238] and NextPNR [169].

The hub module (figure 5.4) features eight RJ-45 connectors for sensor communication. Power to the sensor modules can be disabled for efficient standby and device reset during development. The sensor hub connects to a host computer via USB-3 SuperSpeed. The USB connection is implemented using an FT601[3] IC. A custom driver with ROS [250] integration is developed in C++ using libusb [251]. The driver is partially based on existing documentation and open-source driver development efforts [252]. Each RJ-45 and USB port is equipped with high-speed TVS (transient voltage suppressor) arrays[4] for protection and with addressable status LEDs[5]. Synchronization signals are generated directly on the sensor hub by the FPGA using an on-board reference oscillator[6].

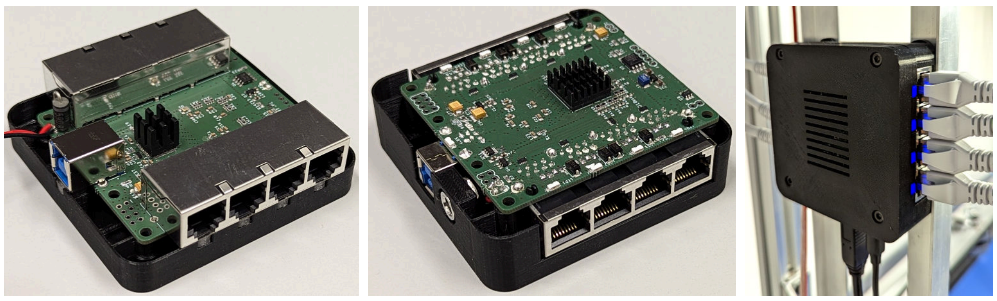


Figure 5.4.: Sensor hub with exposed circuit board (left, middle) and installed for data recording (right).

Each camera module (figure 5.5) is equipped with a 5MP MT9P001 image sensor [253]. The sensor is selected for good sensitivity to minimize motion blur, resolution and framerate, publicly available documentation, and to avoid licensing issues regarding proprietary protocols used by other sensors [253, 254, 255, 256, 257, 258, 259, 260]. A RAM IC[7] is added as a framebuffer for experimentation with onboard processing and for efficient bandwidth usage. Image data can be transferred not only during sensor readout but also during exposure. In order to still allow for each frame to be triggered individually with minimal protocol overhead over a single half-duplex connection, the synchronization packets from the sensor hub to the cameras are scheduled to be sent between subsequent frame transmissions while the previous frame is still being captured. Each synchronization packet contains fields for resolution and region of interest, allowing the field of view to be changed dynamically while maintaining ac-

[2]J.S.T. Deutschland GmbH, Germany
[3]FT601, Future Technology Devices International Limited, UK
[4]PESD4USB5U-TBS, ESD protection for high-speed interfaces, Nexperia BV, Netherlands
[5]SK6805 SIDE-G, Shenzhen Normand Electronic Co.,LTD, China
[6]XO53 Series Oscillators, 40.000 MHz, Euroquartz Ltd., UK
[7]APS6408L-OBM-BA, AP Memory, Zhubei, Taiwan

curate synchronization. The radio module is equipped with a wireless transceiver[32] for communication with wearable devices such as the sensor glove. Complex impedance measurements from the tactile sensor glove are compressed via a shared-exponent encoding (similar to [261]).

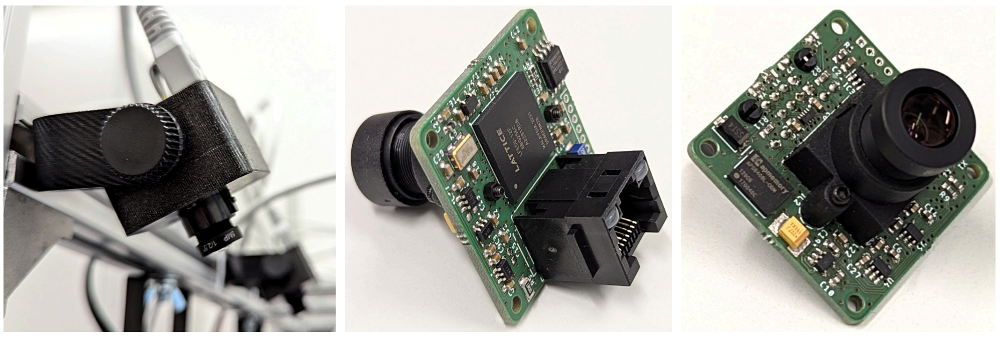


Figure 5.5.: Camera modules installed in a multimodal recording setup (left) and internal circuit board (middle, right).

The wired modules of the multimodal sensor system are mounted on a cube-shaped metal frame. Seven cameras and one radio transceiver are attached to the top and the sensor hub is mounted at the side. The upper-front metal bar is shifted backwards by approximately one third of the side length for easier access. The lower front and rear beams are recessed downwards to align with the table surface and to prevent sliding. A first prototype is assembled with a size of one cubic meter. For later experiments, the size is reduced to a side length of 80 cm, allowing the recording setup to fit onto common office desks.

5.5. Multicamera calibration

The cameras are calibrated using an Aruco board [156], a checkerboard pattern with additional fiducial markers for identification. Monocular initial guesses are estimated using an existing camera calibration routine [262, 206]. The solution is further refined via bundle adjustment [186] across all cameras and observations. To facilitate integration with subsequent processing steps, camera model and bundle adjustment are implemented using the optimization framework introduced in chapter 2.

5.6. Markerless vision

Hand motions are reconstructed through a combination of neural keypoint detection and trajectory optimization. An existing image recognition network detects 21 keypoints on each hand [120, 118]. The same reconstruction method is used for glove and hand tracking. Since the network is trained mostly on bare hands, a hue filter is first applied when using the sensor glove. The previous reconstruction method from section 5.3 is adapted for 2D image sensors and extended with an articulated hand model. The same model [242] is used that also serves as a reference for constructing the sensor glove (section 4.8).

During a first initialization phase, a linear equation is solved for each hand keypoint with at least two simultaneous observations to find a 3D vector that minimizes squared distances to corresponding observation rays. All finger joints are initially fixed and a single least-squares estimate for the wrist pose is found using the Kabsch algorithm [95, 188]. Hand pose and joint angles are then optimized together for each hand and time step to obtain sequences of articulated kinematic hand pose estimates. During occlusions, wrist poses and joint angles are linearly interpolated. Kinodynamic trajectory optimization is performed using the kinematic estimates as an initial guess to avoid local minima. Squared reprojection errors [61, 186] are minimized together with joint limit penalties and regularization terms. A smoothness regularizer minimizes Cartesian jerk at each joint via a quadratic loss for bridging occlusions and to reduce noise. Additional diagonal regularization is added for stability.

Objects are represented as 3D point features. Relationships between the point features are learned at a later stage together with the manipulation task from the demonstration data. 2D point features are detected in RGB images from the camera system, matched and triangulated. First experiments are performed with a MobileNet-based [263] architecture and transfer learning. A three-layer decoder maps embeddings to keypoint heatmaps. Input layer and hidden layer in the decoder contain 32 neurons each and use TanH activation. The output is resampled to the original image resolution through bicubic interpolation [101] and local maxima are interpreted as keypoint detections. Keypoints are marked in a small subset of images and a new linear decoder layer is trained to detect the keypoints (see section 7.6.6). In later experiments, the network is replaced with a different architecture based on U-Net [157] using skip connections for accurate localization and further data augmentation. The model consists of 12 convolutional layers with a receptive field of 3-by-3, increasing and decreasing channel counts from 32 over 256 to 32, ReLU activation [57] and max pooling [198].

5.7. Tactile projection

Tactile measurements from the sensor glove (section 4.8) are projected onto the hand motion reconstructions using the glove model (section 4.8.1). Not directly connected tactile elements are padded to avoid interpolation artifacts. The hand model is brought into the same pose as the glove model and coordinates of the sensor elements in the readout matrix are projected onto the hand model. These tactile coordinates are saved as additional vertex attributes. During reconstruction, the position of each vertex is estimated [122] using the bone indices and blend weights from the underlying MakeHuman model [242]. Corresponding tactile data is interpolated by sampling the tactile sensor matrix using the tactile vertex coordinates with bicubic interpolation [101].

5.8. Experiments

The multimodal sensor system (section 5.4) is used to record human demonstrations of object manipulation tasks. The data is processed as described in sections 5.5, 5.6 and 5.7 to create 3D reconstructions.

During a one-handed bottle opening task, the object is held via a power grasp while the lid is turned using the thumb. As the thumb moves from the left to the right, a rolling contact moves around the tip of the thumb from the right to the left. The other fingers maintain stable grasping forces. Tactile noise and drift from the sensor glove are considerably lower than the interaction forces.

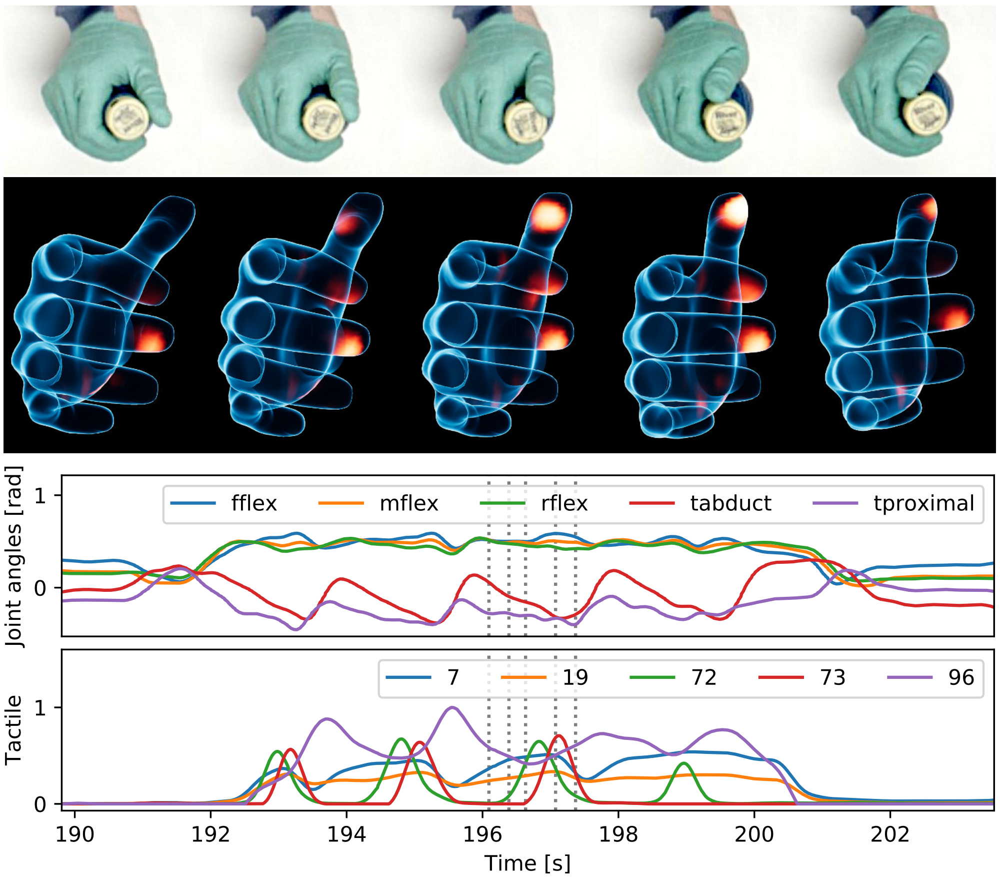


Figure 5.6.: RGB images are recorded together with tactile measurements from a sensor glove (top). Reconstructions are rendered using the hand model (middle) with tactile activations (red, yellow, white) and plotted over time (bottom). Correspondences between images (top, middle) are indicated in the plot (bottom) as dotted gray lines. Figure first published in [296].

Bimanual demonstrations are recorded for an assembly task. A small camera module[8] with a flexible cable is installed on a circuit board. An image sensor and lens assembly with a size of approximately 6 mm × 6 mm is located at one end of the cable and a board-to-board connector is located at the other end of the cable. The circuit board features a second matching board-to-board connector. The camera module is picked up via a precision grasp with one hand at the opposite end of the connector and placed above the circuit board to align both connectors. The connectors are pressed together with the first finger of the other hand. If the connector is inserted correctly, the circuit board can be lifted by the installed camera module.

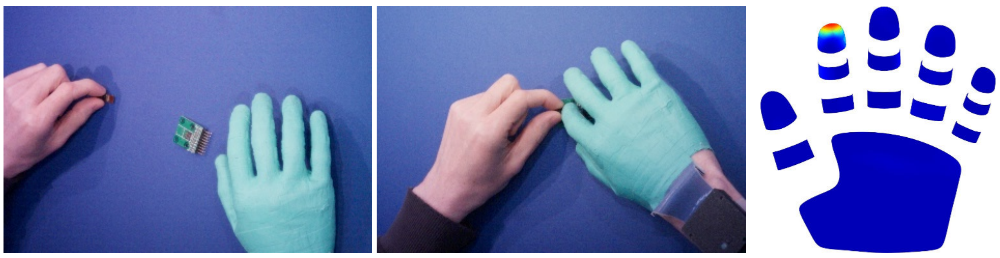


Figure 5.7.: Camera images (left, middle) and tactile data (right) for an assembly task.

The task is demonstrated with one glove and one bare hand as shown in figure 5.7. Hands are switched to capture tactile data for both grasping the camera module and for pressing on the connector. See figure 5.8 for renderings of the 3D reconstructions and figure 5.9 for plots of the reconstructed trajectories and tactile recordings. The average deviation between reconstructed object feature positions and observation rays is 0.2 mm and the average reprojection error is 1.8 pixels.

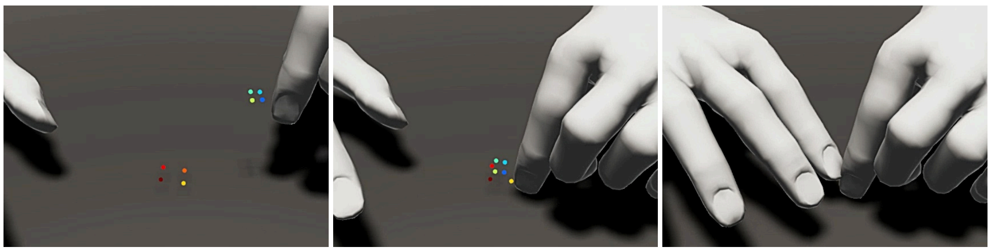


Figure 5.8.: Human hand pose reconstructions and object keypoint positions for a bimanual assembly task.

The reconstructed trajectories start with parallel fingertip motions during an approach phase while the objects remain stationary. The system is able to capture small opposing finger motions during the grasping phase. While fixing the connector, object features are temporarily occluded. The tip of the first finger of the other hand

[8]OV7660 1/6 lens, ModuLink UG, Germany

is detected close to the last detected positions of the occluded object features. The tactile sensor glove can successfully capture the grasping and insertion forces despite the small object size.

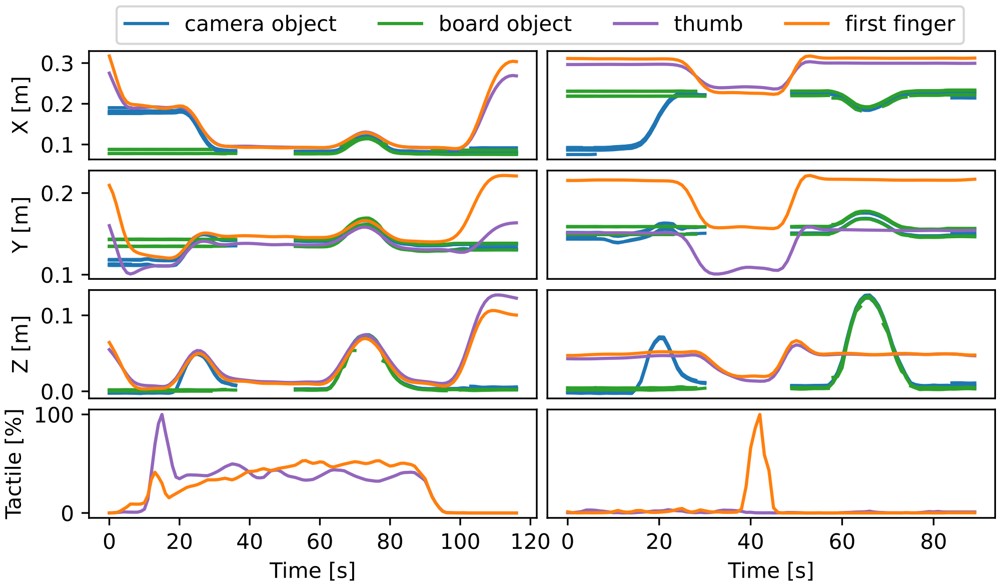


Figure 5.9.: Motions and tactile data are captured for a bi-manual assembly task and plotted over time. A camera module (blue) is first grasped with the right hand and fixed onto a circuit board (green) with the left hand (left column). Hands are switched during a second demonstration (right column).

6. Teachable robot workcell

Facilities for multimodal teaching by human demonstration and for robotic task execution are integrated into a single self-contained workcell (figure 6.1). The box-shaped frame houses a robot with two end-effectors, control circuitry, a radio transceiver for wireless teaching devices, lights and a multi-camera system. Translation is implemented using linear rails. These are complemented by additional rotary axes at the end-effectors. All axes support position, force and impedance control. The hardware is designed to balance speed, precision, low cost and ease of setup.

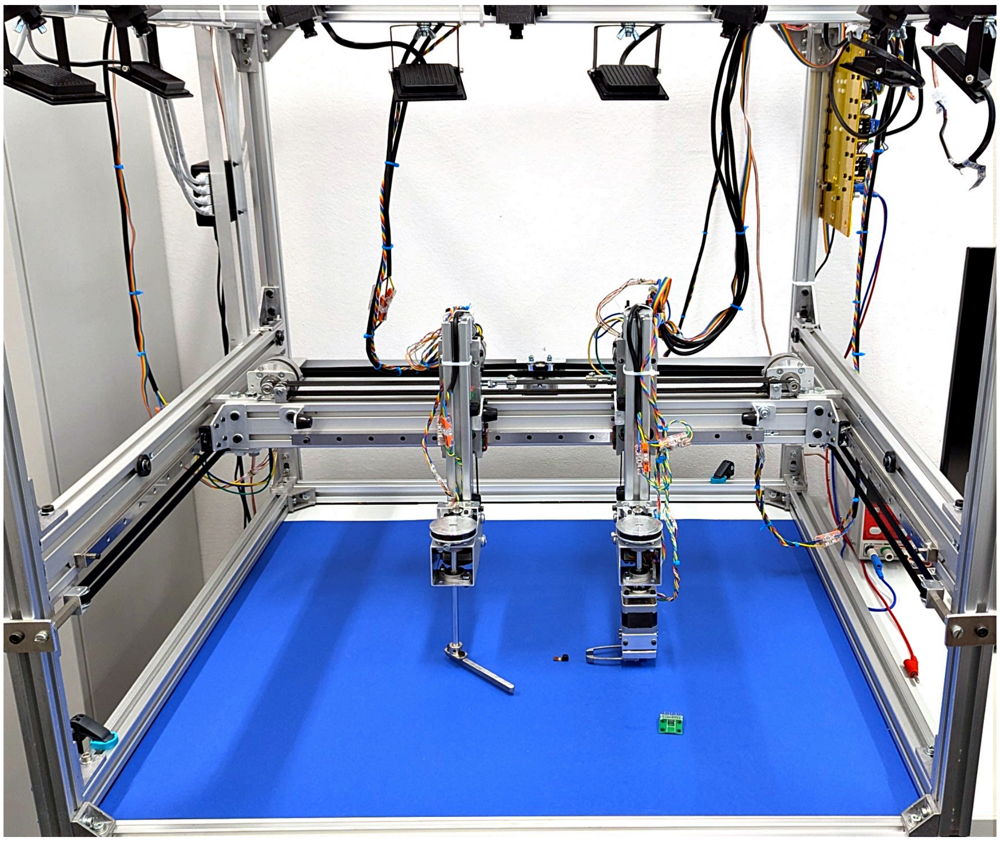


Figure 6.1.: Integrated multimodal workcell for teaching by human demonstration and robotic execution.

6.1. Related work

Machines such as CNC mills, printers and 3D printers are frequently offered as fully integrated desktop devices, in some cases featuring box-shaped enclosures [97, 264, 265, 266]. The same form factor is used successfully in various environments from individual home use over small businesses to large-scale industrial 3D-printing farms [267]. Products currently offered as industrial robots are typically individual robot arms with low-level control units [268, 269, 166]. Some offerings include cabinets, tables or part holders [270, 271]. Advanced uses such as dexterous object manipulation require the integration with additional sensors and end-effectors [272, 142, 273]. Mobile robots, robot dogs and humanoids can include cameras or LiDAR, but the designs focus on inside-out sensing [5, 123, 274]. Humanoid robots typically feature two arms [123, 274]. Some robots with single end-effectors are advertised as desktop robots [275, 276], but do not include advanced sensors and end-effectors, nor built-in support for teaching by human demonstration.

Fully articulated robot arms with only revolute joints can require high actuator torques and frequently use strain wave gears for high reduction ratios and low backlash [4, 166, 268, 269]. 3D printers typically use stepper motors and toothed belts for the horizontal axes [264, 265], offering accurate and fast position control at low cost. Some industrial robot designs integrate toothed belts as intermediate gear stages [277]. Recent robot actuators for dynamic interaction through contacts employ BLDC motors with low reduction ratio planetary gears, offering fast motion and torque control via motor currents, but can suffer from backlash [99]. BLDC motors can be controlled smoothly and efficiently using feedback loops and combinations of sine waves [145, 174, 19].

Robot gripper designs typically feature two or more forward-facing fingers [81, 36]. Linear profile rails can offer a cost-effective solution for mounting robot fingers with low backlash [36]. Antagonistic grasping motions can be generated via a spiral cam mechanism [278]. For humanoid systems, it can be challenging to integrate separate motors for all joints inside the hands. Tendon drives, however, can suffer from friction, slackening and wear. This can be countered with antagonistic systems at the expense of additional motors [70]. Other designs use underactuation with less motors than joints [112]. During manipulation, robot fingers can partially or fully occlude objects. While small occlusions can have little to no effect, large occlusions can significantly degrade performance [59, 187].

6.2. Robot design

The robot is housed in a box-shaped aluminum frame with a footprint of 80 cm × 80 cm and a height of approximately 70 cm (figure 6.2, center). The multimodal sensor system developed in section 5.4 is mounted on the beams at the top. The robot arms can move along linear profile rails[1][2]. Rails on the left and right of the frame let the robot move

[1]Linearführung MGN15H 600mm, anzado GmbH, Germany
[2]MGN15, DOLD Mechatronik GmbH, Germany

back and forth. A shared perpendicular rail with two independent carriages let the arms move left and right. An additional vertical linear axis on each arm permits lifting and lowering the end-effectors. The end of each arm features a revolute wrist joint. The robot is constructed mostly from aluminum and steel.

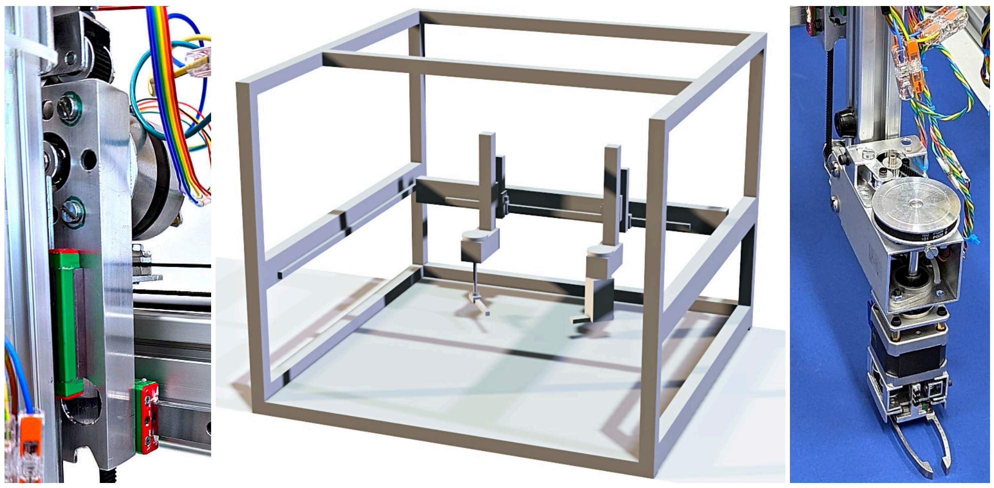


Figure 6.2.: Mechanisms for X and Z axes (left), CAD model for construction, visualization and control (center), arm with gripper (right).

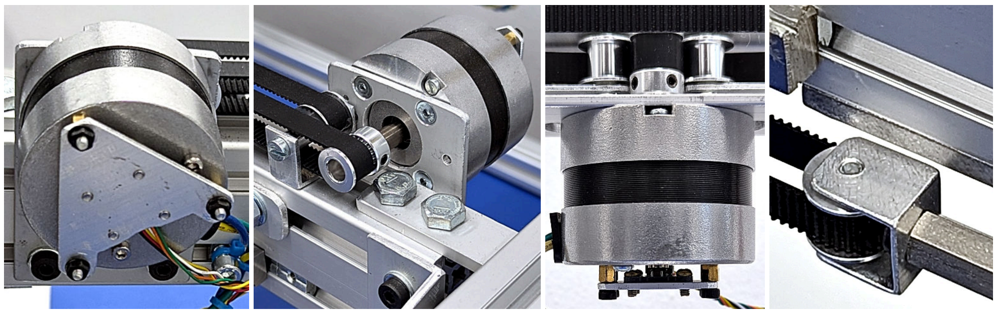


Figure 6.3.: X-axis motor with encoder (left), X-axis motor with pulley, idler and drive belts (2nd from left), Y-axis drive with encoder (2nd from right), Y-axis tensioner (right).

All axes are driven via BLDC motors and toothed belts [60], with larger motors[3] and 10 mm wide belts for the linear axes (figure 6.2, left), and smaller motors[4] with 6 mm wide belts for the gripper and rotary axes. The X axes of the two arms are

[3]Brushless DC Motor 57BL, ACT Motor GmbH, Germany
[4]Brushless DC Motor 42BLF, ACT Motor GmbH, Germany

driven via two parallel belt loops (figure 6.3). The drive belts for the Z axes lead around upper and lower return pulleys for an additional 2-to-1 reduction (figure 6.2, left). The rotary axes are driven via a 5-to-1 reduction with 16 teeth on the motor shaft and 80 teeth on the output shaft (figure 6.2, right).

The original digital hall sensors pre-installed on the motors for trapezoidal commutation are removed. Instead, the motors are retrofitted with diametrical encoder magnets and custom encoder boards with 15-bit absolute encoders[5] for sinusoidal commutation and accurate position measurement (see figure 6.3).

The gripper consists of a brushless motor, a mechanical transmission and exchangeable fingers. Each finger is made from a single piece of steel and features a widened tip at one end, a flat mounting plate at the base and a thin connecting rod from the base to the tip in order to minimize visual occlusion. Each finger is mounted on a sliding carriage with a linear profile rail. A circumferential timing belt drives the fingers along opposite directions from a central motor shaft. The drive belt runs around fixed idlers on one side and through a tensioner with two additional idlers on the other side around the motor pinion. Vertical beams transmit force from the belt to the fingers. See figure 6.4 for images of the fingers and drive mechanics. The gripper is mounted on one of the arms. For the current prototype, the other arm is equipped with a single rotatable finger.

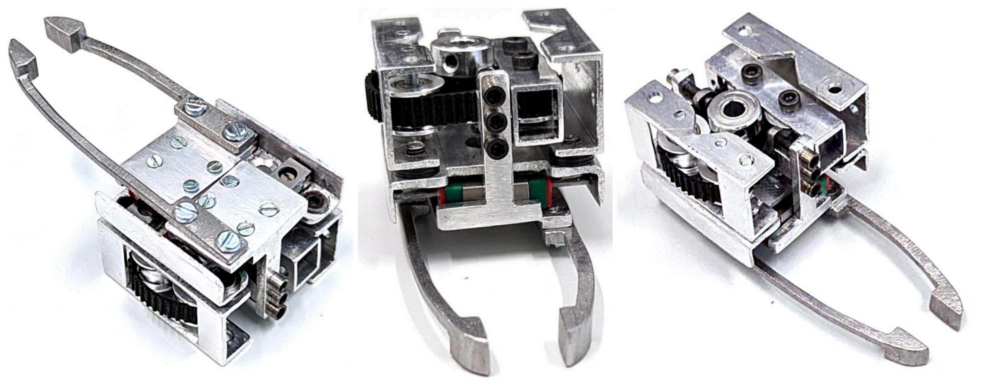


Figure 6.4.: Gripper head before installing the motor and tensioning the drive belt.

A central control board integrates an FPGA[6], a USB interface[7] and digital isolators[8]. The USB interface is used to exchange commands and sensor data with a host computer and provides access to an additional JTAG port for programming and diagnostics. The motors are connected to the FPGA via driver boards with H-bridges as well as amplifiers and ADCs for current sensing. External pullups are added to activate motor

[5] A1333, Precision, High Speed, Hall-Effect Angle Sensor IC with Integrated Diagnostics for Safety-Critical Applications, Allegro MicroSystems, USA
[6] LFE5U-12F-6BG256C, Lattice Semiconductor Corporation, USA
[7] FT2232, Future Technology Devices International Limited, UK
[8] Quad-Channel Digital Isolators, Analog Devices, Inc., USA

braking in case of connection failure. The FPGA implements encoder and current readout, low-level filtering, sinusoidal commutation, symmetric PWM modulation and a watchdog timer. Driver software on the host computer integrates the robot with ROS [250] and ros_control [37].

6.3. Experiments

The gripper is first tested in isolation. A position controller is loaded and the fingers are commanded to open and close repeatedly. The gripper is monitored via joint position measurements, a camera and a microphone. The gripper can fully open and close 10 times per second. A folded piece of tissue paper is placed between the finger tips and the gripper is commanded to hold a fully closed position. The gripper can be lifted off the table by the object. The gripper is switched to effort control and programmed to apply small alternating efforts with a frequency of 0.2 Hz. The gripper fully opens and closes. If a blueberry is placed between the tips, the fingers stop and the fruit can be lifted without piercing the skin as shown in figure 6.5. The robot fingers can be stopped or backdriven by human hands.

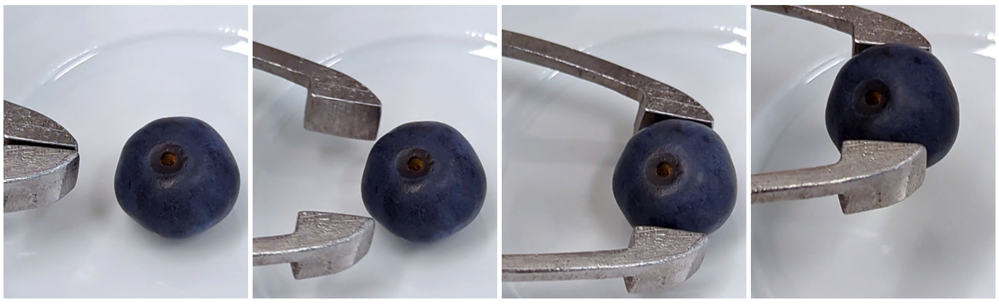


Figure 6.5.: Grasping a blueberry with compliant control.

The gripper is mounted on one of the arms. The watchdog timer is configured for a timeout of 10 ms. If the USB cable is unplugged or the control node is terminated, the robot stops without noticeable delay. Motor braking is activated and the Z-axes gently descend, from an elevated position over the course of multiple seconds. A PID controller with a weak proportional term and a narrowly clamped integral term is loaded. All joints can be passively backdriven by a human. If released, the encoder measurements return to the commanded positions within 0.1°.

Cartesian precision is evaluated by programming the robot to repeatedly drive against a dial gauge (see figure 6.6). After each measurement, the robot breaks contact with the measuring tip. Robot axes are first moved individually by 60 ° for revolute and 1 cm for prismatic joints. For the horizontal Y-axis, the measurements are taken at the wrist (i.e. top of the gripper motor) and for vertical, gripper and revolute joints, the measurements are taken at one of the finger tips. Additional tests are performed with larger random distances and simultaneous motions along the X, Y and Z axes. During each trial, the robot is moved into two different randomly selected poses. Horizontal positions are chosen from in interval of 10 cm and vertical positions are chosen from an interval of 8 cm. The experiments on individual axes are each repeated 12 times and the randomized experiment is repeated 10 times. It has been suggested to quantify robot precision via standard deviations or three-times standard deviations [90]. See table 6.1 for results.

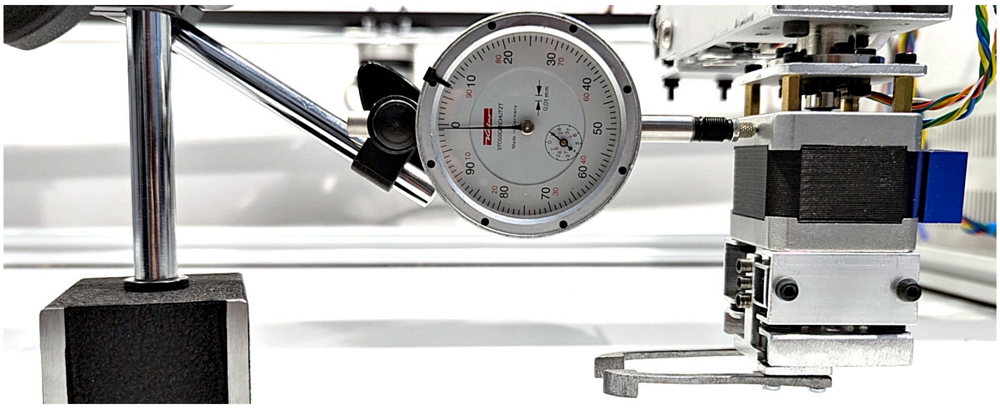


Figure 6.6.: Various parts of the robot are repeatedly driven against a dial gauge to evaluate precision.

Joints	End-effector	$3\,\sigma$	$1\,\sigma$ (std. dev.)
Y	wrist	0.03 mm	0.009 mm
Z	finger tip	0.05 mm	0.015 mm
Wrist	finger tip	0.014 mm	0.0047 mm
Gripper	finger tip	0.02 mm	0.0065 mm
X, Y, Z	wrist	0.07 mm	0.023 mm

Table 6.1.: Repeatability for different joint combinations and end-effectors.

7. Teaching dexterous manipulation by human demonstration

This chapter introduces a machine learning framework for teaching dexterous manipulation tasks by demonstration with human hands in the real world. Contact information is captured using instrumented gloves. The learned behavior can be rolled out recursively as neural simulations for evaluation prior to execution. During execution, limited-horizon trajectories are generated from the neural network and transferred to robot models through online trajectory optimization while enforcing kinodynamic limits and collision avoidance. Experiments are performed using the robot workcell (chapter 6) and on humanoid hand-arm systems.

7.1. Related Work

Some collaborative robots support teach modes in which the robots can be physically pushed by humans into different poses [269]. The poses can be saved and used in robot programs as waypoints. These programs are typically created by writing instructions in a scripting language [279] or by composing a list or tree of pre-defined operations via a graphical user interface [269].

It is also possible to record dynamic motions from humans through motion capture or from robots using teach mode or teleoperation. These recordings can be split into segments, labeled and annotated as rhythmic or discrete. The annotated segments can be used to create motion primitives and adapted by higher-level programming [88, 144, 30, 48, 67, 38]. Robots can be teleoperated using special input devices [22, 66, 280, 56, 245, 80, 18, 155] or via cameras and human motions [115, 114, 113, 76, 9, 92, 106, 94]. Artificial neural networks can be trained via backpropagation with motion and force data from instrumented tools for manipulation tasks such as peg-in-hole insertion [41, 103]. Some works use separate unimodal networks [41]. Studies suggest that sensor fusion can be improved through early integration during learning [41]. Robot hands can be trained by teleoperation in simulation [91, 151]. Learning from demonstration can also be formulated as identifying goals or cost functions [1].

Recent works model imitation learning as denoising tasks [176, 83] to handle one-to-many mappings [35]. Diffusion policies can be conditioned on camera images, motion trajectories or point clouds [35, 56, 36, 202]. Noisy actions are provided as additional inputs and the networks are trained to reduce noise in the action space [35, 176, 83]. The demonstrations are performed on the robot via teleoperation [56, 36]. Human motions can also be used to guide learning in simulation [148, 151]. Random resets to poses from motion capture can accelerate convergence [148]. Some learning, analysis and control methods for walking use simplified point-based models [27, 195]. Motions can be adapted according to robot kinematics, dynamic constraints or for collision avoidance through numerical optimization [167, 208, 74, 96, 179, 170, 98].

7.2. Learning methods and architectures

Manipulation tasks are learned from multimodal observations of human demonstrations. An artificial neural network N_{MT} maps an input subset $O_{t-dt,I}$ of observations O_{t-dt} from one time step to an output subset $O_{t,O}$ of observations O_t from a following time step.

$$O_{t-dt,I} \in O_{t-dt} \qquad O_{t,O} \in O_t \qquad N_{MT} : O_{t-dt,I} \mapsto O_{t,O} \tag{7.1}$$

During training, the inputs are corrupted by artificial noise R and the neural network N_{MT} is trained to predict observations with reduced noise.

$$O_{t,O} \overset{!}{=} N_{MT}(O_{t-dt,I} + R) \tag{7.2}$$

The observations include a set of 3D keypoints. During learning, each keypoint represents either a human hand keypoint or a point on an object. During execution, each keypoint represents either a point on an object or is mapped to a point on a robot model. For robot execution, a receding-horizon controller optimizes robot commands to match outputs from the neural network. The behavior can also be executed as neural simulations through recursive feedback of predictions as inputs. In both cases, at each time step t, a set $C_{O\,t}$ of consistency constraints $c_{O\,t}$ is applied.

$$\forall c_{O\,t} \in C_{O\,t} \;:\; c_{O\,t} \overset{!}{=} 0 \tag{7.3}$$

In overconstrained cases, a least-squares solution is used.

$$\min_{O_t} \sum_{c_O \in C_O} \|c_{O\,t}\|^2 \tag{7.4}$$

The solution should match outputs from the network. Keypoint positions $P_{t\,i}$ should be consistent with keypoint velocities $V_{t\,i}$.

$$\frac{P_{t\,i} - P_{t-dt,i}}{dt} - V_{t\,i} \in C_{O\,t} \tag{7.5}$$

7.2.1. Feed-forward architecture

The feed-forward network receives a list of positions and velocities for object and hand keypoints as input. The input data is flattened into an input vector. This input vector is then fed through a stack of five dense layers with ReLU activation [57]. The first layer consists of 2048 units and the three inner layers are made up of 512 units each. The final layer has three neurons for each hand keypoint to generate 3D velocities.

7.2.2. Denoising

The learned behavior should adapt to changed object positions and correct for drift. Additionally, control errors might accumulate over time as increasing state errors. The network should thus be able to compensate for such errors. Random noise is added to the inputs and the network is trained to remove part of the noise. The noise is added to the input positions P_I but, scaled by a denoising factor, subtracted from the target output velocities V_O. Since velocities are produced as outputs, which are already fed back to future input positions, the same output channel can also be used for denoising.

The learned behavior should ideally be robust to large perturbations but also be capable of precise control. Gaussian noise with a large amplitude might cause the network to overfit to denoising and neglect fine control. A small noise amplitude might fail to generalize to larger perturbations. The denoising is therefore controlled by an exponential scaling factor. A random exponent $R_{U,t}$ is drawn once for each training sample at time t and a constant base is raised to the random exponent. Added Gaussian noise R_Z is scaled by the resulting factor and a constant gain f_n. If only velocity outputs are used for motion generation, these should be perturbed accordingly.

$$P_{I,t,i,d}{}' = P_{I,t,i,d} + R_{Z,t,i,d} \cdot b_n^{R_{U,t}} \tag{7.6}$$

$$V_{O,t,i,d}{}' = V_{O,t,i,d} - R_{Z,t,i,d} \cdot b_n^{R_{U,t}} \cdot f_n \tag{7.7}$$

7.2.3. Position representation

Position invariance can be introduced through normalization. For each time step, the arithmetic mean of all hand point positions can be computed and used as a 3D reference point. Position vectors can be fed to the neural networks relative to the reference point, subtracting the reference point from each input vector. However, in some cases, such as bimanual manipulation, it may be unclear how to choose a single meaningful reference point. Position information can instead be encoded as pairwise relative translation vectors P_{tjk}, normalized by squared distance with a small bias ϵ for stability.

$$\frac{P_{tj} - P_{tk} - P_{tjk}}{\|P_{tjk}\|^2 + \epsilon} \in C_{Ot} \tag{7.8}$$

7.2.4. Augmentation

During execution, the robot may be located in a different direction from the objects than the human was during demonstration. The training data can be augmented with random rotations. Input position and velocity vectors as well as target vectors for output velocities are multiplied with rotation matrices. For feed-forward networks, one random angle is drawn for each time step and for a recurrent policy (section 7.2.6), one angle is drawn for each rollout. The angles are chosen from 0° to 360°. It is assumed that gravity can be significant for many manipulation tasks. The random rotations are thus only performed around the Z-axis i.e. gravity vector.

7.2.5. Masking

Some features or modalities may not always be observable. Visual features can be occluded. Tactile is only available if a tactile glove is used during demonstration. Masks are generated with ones for observed features and zeros for missing features. Inputs are multiplied by the masks after normalization. During training, the residuals that are used for computing the training loss are multiplied by the mask as well. In order to maintain support for neural simulations and receding horizon rollouts, masks are also predicted as outputs together with absolute values, and the corresponding consistency constraints are masked.

7.2.6. Recurrent architecture

Manipulation can change object states in ways that might not always be immediately observable. The feed-forward architecture would always map the same observations to the same robot commands. Recurrent connections are added to allow remembering previous actions. At the same time, the robot should mainly react to sensor input, e.g. adapting its motions to observed object positions. The recurrent connections are therefore introduced through a separate recurrent branch with dropout and reduced layer sizes (neural bottleneck). The recurrent branch has three dense layers with 32 input and 16 hidden units. Keypoint positions and velocities are combined with recurrent feedback through a concatenation layer. The same outputs from the concatenation layer are fed into the feed-forward control branch and into the recurrent branch.

7.2.7. Recurrent training

The recurrent policy is trained on whole demonstration trajectories. Output velocities are integrated over time and fed back to the network as input positions. In order to accelerate convergence during long-running tasks, the positions are reset to measured positions from the demonstration data at randomly selected time frames. Feedback in the recurrent architecture is only reset at the start of a demonstration.

7.2.8. Recurrent denoising

Large recurrent networks can be expensive to train due to vanishing or exploding gradients and long critical execution paths limiting parallelization. Additionally, long time integrals, whether from a recurrent neural network or from a feed-forward network executed in a control loop, can be sensitive to small changes in initial conditions.

Recurrent memory can instead be modeled as an additional sensor modality. The neural network is trained to predict and denoise recurrent activations together with original sensor data. One output is trained to produce predictions H_{Pt} for current values H_t without noise and another output is trained to predict the original noise-free rate of change $dH_{P\,t}$.

$$H_t - H_{P\,t} \in C_{O\,t} \tag{7.9}$$

$$H_t - H_{t-dt} - dH_{P\,t}\,dt \in C_{O\,t} \tag{7.10}$$

The recurrent activations can be pre-computed. In the current prototype, a single-layer recurrent network with randomly initialized weights is used. The structure is inspired by reservoir computing and Euler State Networks [58, 49]. However, it uses linear activation for simplicity. A sequence of recurrent activations H_t is computed using a scaled random orthogonal matrix W_R and a normally distributed input matrix W_I from keypoint velocities V_t with occlusion masks $m_{V\,t}$. The calculation is performed both forwards $H_{t\,,+1}$ and backwards $H_{t\,,-1}$ in time.

$$\forall dt \in \{-1, +1\} : \; H_{t+dt} = W_R H_t + W_I m_{V\,t} V_t \tag{7.11}$$

$$H_t = H_{t\,,+1} + H_{t\,,-1} \tag{7.12}$$

7.2.9. Model-based learning

The robot should also learn to automatically adapt contact forces from human demonstration data. In a purely feed-forward supervised manner with only a policy network, forces as inputs and motion commands as outputs, this would require explicit training examples with incorrect forces being corrected. This could necessitate additional data collection and the samples with wrong forces that should not be imitated might may have to be labeled as such, requiring additional human effort during the teaching process.

Learning to control contact forces can be handled through a model-based approach. A tactile model receives the same other input modalities as the policy and predicts contact forces. The model can be trained on the same data recordings as the policy, but can be sampled for incorrect positions and forces when training the policy. If the human teacher makes and breaks contacts with an object during demonstration, the data set has to contain examples for force gradients close to the contact points. Thus, the tactile model can learn to predict increasing contact forces if the robot or policy network pushes against an object.

A similar approach can also be used with instrumented objects. Sensors can be attached to objects during teaching in order to directly measure otherwise hard to observe state information. The data is used to train an object state model. During execution on the robot, object instrumentation is not required and the object variables can be predicted.

Each model network consists of three dense layers with TanH activation. Layer sizes are set to 64 for the tactile model and to 32 for the object state model. The different activation function is chosen to provide smooth model gradients during policy learning.

7.2.10. Tactile control

Tactile control can also be achieved using only a single neural network by predicting tactile data as an additional output. During robot execution, predicted tactile readings are added to the control efforts. Tactile recordings may not be available for all hands and demonstrations. Thus, if unavailable, the tactile loss is masked during training.

7.2.11. Object dynamics

During object manipulation by humans or robots, the objects are controlled only indirectly through contacts. If accurate object models and complete state information would be available, the dynamics could be computed via differentiable physics (see chapter 2), for example by linearizing the dynamics around a current operating point and calculating a dynamics Jacobian. However, when learning manipulation tasks from human demonstrations, only partial observations are available. The network is trained to predict a dynamics matrix $J_{D\,t}$ as one of the outputs, which maps hand keypoint velocities to object keypoint velocities. Input hand velocities $V_{t\,H\,I}$ are multiplied by the dynamics matrix to predict object velocities $V_{t\,O\,P}$.

$$V_{t\,O\,P} = J_{D\,t} V_{t\,H\,I} \qquad V_{t\,H\,I} \in O_{t\,I} \qquad V_{t\,O\,P}, J_{D\,t} \in O_{t\,O} \tag{7.13}$$

Dynamics predictions $J_{D\,t}$ are trained using motion and tactile via a dynamics loss $L_{P\,t}$. The predicted velocities should match observations $V_{t\,O\,O}$. However, contact interaction is only possible if a non-zero contact force T_t is present. Physically implausible interactions without contact force are minimized together with velocity errors. A small constant ϵ is added for stability [153].

$$L_{P\,t} = \|J_{D\,t} V_{t\,H\,I} - V_{t\,O\,O}\|^2 + \|J_{D\,t} (T_t + \epsilon)^{\circ -1}\|^2 \tag{7.14}$$

During inference, the learned dynamics is included in the constraint set.

$$[J_{D\,t} V_{t\,H\,I} - V_{t\,O\,O}] \in C_{O\,t} \tag{7.15}$$

7.3. Robot setups

First experiments are performed on humanoid robot setups, each combining an industrial robot arm with a humanoid robot hand. During later experiments, the robot introduced in chapter 6 is used.

7.3.1. Hand-arm systems

As robot arms, a 7-DoF KUKA LBR [166], a 6-DoF UR10 [6] and a 6-DoF UR10e [269] are used. The arm is equipped with either a Shadow C5 hand or a Shadow C6 hand [272]. The Shadow C5 hand is actuated by pairs of pneumatic muscles, while the Shadow C6 hand is driven by electric gear motors. In both case, the actuators are mounted at the forearm and drive the joints through tendons. The distal and middle joints of the fingers are mechanically coupled. The finger-tips are equipped with single-channel pressure sensors.

7.3.2. Pneumatic hand control

Due to incompatibilities between the original control software and current ROS versions, a custom hardware interface is developed for the C5 hand. The hand connects to a microcontroller board[1] via a CAN bus and to the host computer via USB [251]. The original software and firmware setup controls joint positions directly through valve commands via a PID controller and can lead to increasing pressure buildup when attempting to hold a contact. The new software uses a two-stage PID controller and is able to hold stable contact forces. A lower-level regulator controls the air pressure inside each muscle according to desired pressure values. A higher-level regulator controls the joint positions via pressure differentials between antagonistic muscles according to a proportional stiffness parameter. The pressure differential is either added to or subtracted from a fixed bias and forwarded to the lower-level regulators.

7.4. Trajectory control

The learned behavior is mapped to robots through online trajectory optimization. During each trajectory generation cycle, the neural network is recursively rolled out over a limited horizon $0 \dots t_h$, producing a set of Cartesian goal trajectories. A joint-space robot trajectory with positions $P_{J\,t\,j}$ for m joints and keyframes $t \in \{1 \dots t_h\}$ and joint indices j is optimized to match the Cartesian goal trajectories while enforcing safety and feasibility constraints. The start of the trajectory $1 \dots t_f$ is set to be constant and equal to the currently executing trajectory to maintain continuity. Joint positions for the remaining keyframes $t_f + 1 \dots t_h$ are optimized as free variables.

Motions are remapped by minimizing a sum of squared distances L_Q between goal positions $P_{C\,t\,i}$ and the positions of corresponding points on the robot hand (finger joints, wrist) according to robot kinematics K over keypoint indices $i \in \{1 \dots n\}$.

$$L_Q = \sum_{t=t_f+1}^{t_h} \sum_{i=1}^{n} \| P_{C\,t\,i} - K_i(P_{J\,t}) \|^2 \tag{7.16}$$

Joint positions $P_{J\,t\,j}$ are constrained to remain within upper $U_{J\,t\,j}$ and lower $L_{J\,t\,j}$ joint limits and within a fixed trust region δ with respect to joint positions $P_{J\,t\,j\,1}$ during the previous solver iteration.

$$L_{J\,t\,j} < P_{J\,t\,j} < U_{J\,t\,j} \tag{7.17}$$

$$P_{J\,t\,j\,1} - \delta < P_{J\,t\,j} < P_{J\,t\,j\,1} + \delta \tag{7.18}$$

Joint velocities $V_{J\,t\,j}$ and accelerations $A_{J\,t\,j}$ are constrained by velocity limits $V_{J\,t\,j\,U}$ and acceleration limits $A_{J\,t\,j\,U}$.

$$-V_{J\,t\,j\,U} < V_{J\,t\,j} < V_{J\,t\,j\,U}, \quad -A_{J\,t\,j\,U} < A_{J\,t\,j} < A_{J\,t\,j\,U} \tag{7.19}$$

[1]Arduino Due, Arduino S.r.l., Italy

The robot poses are further subject to collision avoidance constraints. For the humanoid robot hands, links are modeled as capsules, i.e. collision spheres at the joints with connecting cylinders between the joints. The workspace is defined by a set of planes with normals N_k and positions along the normals P_k. In order to prevent collisions between the robot and the workspace boundaries, the center of each collision sphere P_l is constrained by the radius r_l of the sphere and by each workspace boundary plane N_k, P_k.

$$N_k \cdot P_l < P_k - r_l \tag{7.20}$$

During later experiments with the teachable robot workcell (chapter 6), an improved version of the trajectory optimizer is used with full collision modeling. Collision shapes are defined via primitives, convex polyhedra and convex decomposition [124]. The shapes are extracted from a URDF file. Closest points and separating axes are initialized using GJK [64] and EPA [17]. Collision constraints are formulated as separating planes N_q, P_q between sets of points P_r, P_s.

$$N_q P_r - P_q < N_q P_s - P_q \tag{7.21}$$

When controlling humanoid robot hands, a data-driven regularizer L_H is added to prefer natural human-like hand poses. The regularization term uses averages μ_{Jj} and standard deviations σ_{Jj} from an existing hand pose dataset [18, 80].

$$L_H = \sum_{j=1}^{m} \left(\frac{P_{Jtj} - \mu_{Jj}}{\sigma_{Jj}} \right)^2 \tag{7.22}$$

The trajectory optimization problem is solved through sequential linear-quadratic programming with a primal-dual interior-point [52, 54, 170, 98] method for the quadratic programs and LU decomposition [117, 281] for the inner linear equations.

7.5. Implementation

Data collection, processing, learning and robot control are implemented as a set of packages for the robot operating system ROS [250]. The robot actuators are controlled via roscontrol [37]. The neural networks are implemented in Python [158] using PyTorch [146], TensorFlow [215] and Keras [282], and are trained using Adam [102]. MoveIt [38] is used to load and inspect the geometric robot models and for validating collision avoidance.

7.6. Experiments

Human demonstrations are collected for multiple different manipulation tasks. The proposed neural network architectures are trained with the recorded data and the learned behavior is executed on robots. Tasks are chosen to test different features of the proposed methods.

7.6.1. Pick-place task

10 demonstrations are recorded for a pick-place task. A human teacher grasps a box-shaped object and places it on a rectangular plate-like target. Before each demonstration, object and target are moved to new positions and orientations on the table. The recorded data is used to train a feed-forward policy. In this experiment, both hand and object motions are tracked using a motion tracking system as described in section 5.3.

The policy network is first executed as a neural simulation. Observed object poses and an initial robot state are fed into the policy network as a start state. Outputs from the neural network are integrated and recursively fed back as inputs. The simulation mode can be run interactively. As a human moves the objects in front of the robot, the simulated trajectories track the observed object poses.

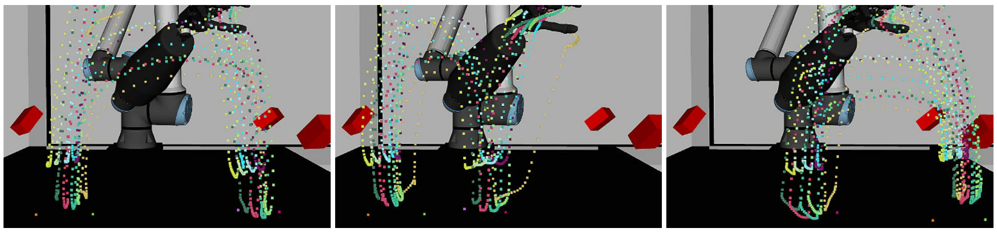


Figure 7.1.: Neural simulations for a pick-place task. The learned behavior automatically adapts to new object positions.

The learned behavior is successfully executed on a UR10e arm with a Shadow C5 hand (see figure 7.2). Both objects can be placed in new poses that are not included in the training set and the motions adapt to the observed poses.

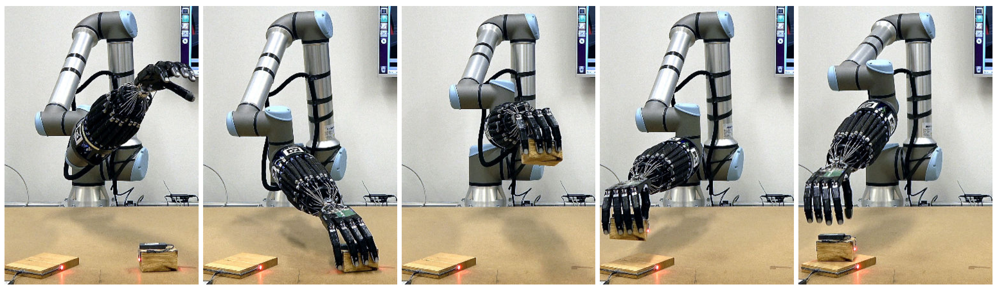


Figure 7.2.: Executing a learned pick-place task on a robotic hand-arm system.

7.6.2. Wiping task

The feed forward policy is trained with 5 demonstrations of a wiping task. A brush is grasped, picked up and pressed against a target object while performing circular wiping motions. The learned policy is executed on a UR10e arm with a Shadow C5 hand (see figure 7.3). The behavior can adapt to new initial brush and target positions. Part of the bristles make contact with the target and wipe across the surface.

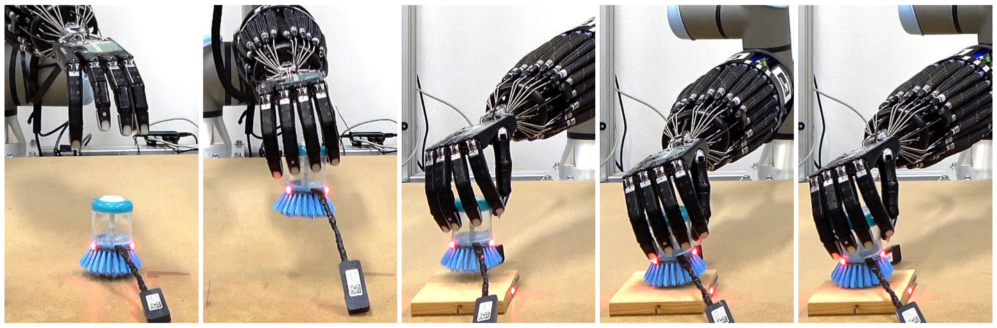


Figure 7.3.: A robot has been taught to pick up a brush and wipe it across a target.

7.6.3. Bottle task

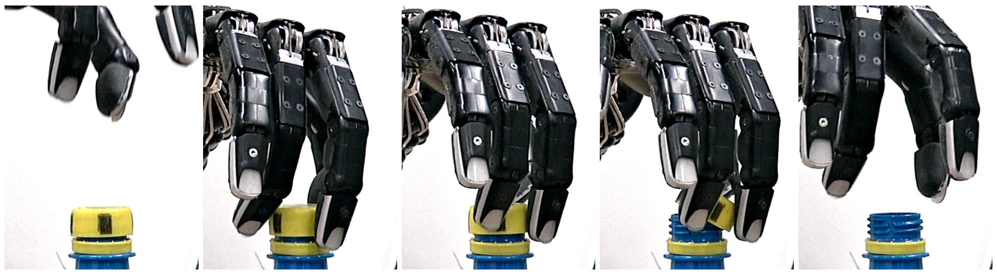


Figure 7.4.: A robot has been taught to open a bottle.

A single demonstration is recorded for opening a drinking bottle and used to train a feed-forward policy. As before, the behavior is executed on a UR10e arm with a Shadow C5 hand. If the control horizon for the online trajectory optimizer is set to less than half a second, the robot fails to turn the lid. If the control horizon is set to 1 s, the task is executed successfully. The robot hand approaches the lid of the bottle and performs repeated turning motions, rotating and removing the lid (see figure 7.4). The approach motion can automatically adapt to new initial robot and object positions.

7.6.4. Lab bottle task

One human demonstration is captured using the motion tracking system for opening a wide-lid chemical bottle. A human teacher approaches the object with a hand, performs repeated turning motions until the lid is unscrewed, grasps the lid, lifts it up, and places it next to the bottle (see figure 5.3). The recording is used to train the feed-forward architecture and the behavior is executed on a KUKA LBR arm with a Shadow C5 hand. During a first trial, object perception is simulated by loading marker positions from the training set and adjusting the pose manually. The robot hand approaches the object and performs repeated turning motions, successfully rotating the lid (see fig. 7.5). Since the feed-forward policy does not have memory, the robot continues to perform turning motions indefinitely, even after the lid has been unscrewed and could be lifted. The initial robot pose and object position can be changed and the robot can still successfully approach the object as long as the hand is positioned mostly above the object with the inside of the palm facing downwards.

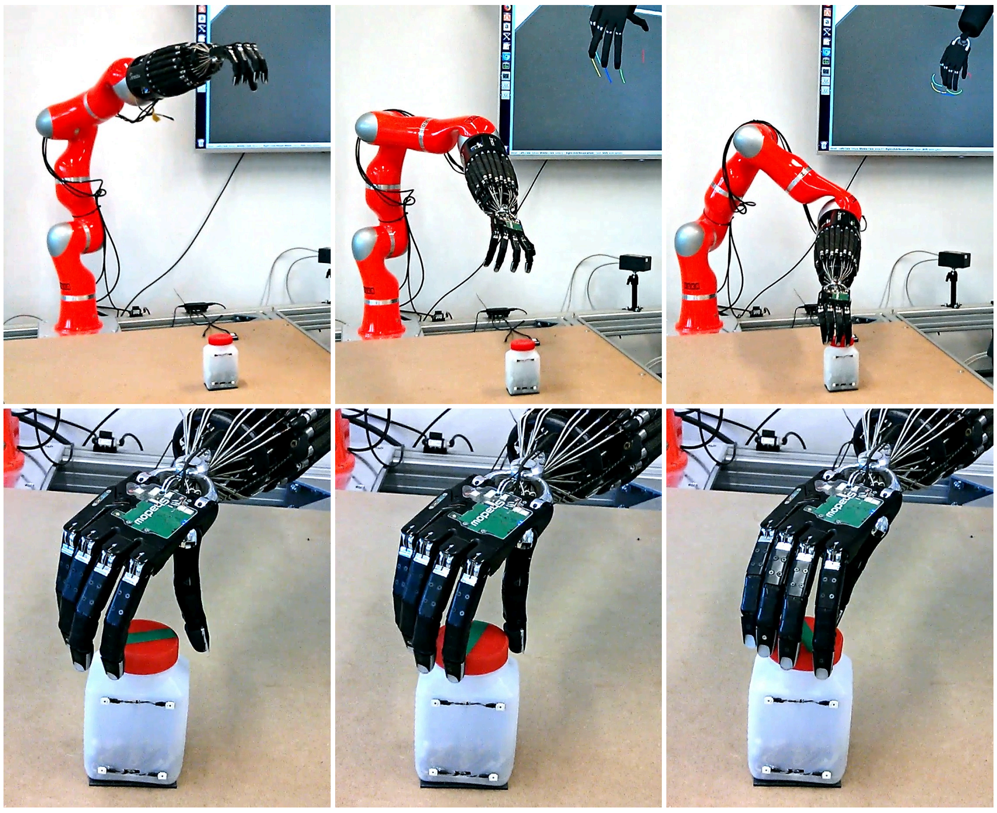


Figure 7.5.: A bottle opening task is learned from human demonstration and executed on a KUKA LBR arm with a Shadow C5 hand.

7.6.5. Recurrent policy

The same demonstration data as before is used to train the recurrent policy for the bottle opening task. The recurrent policy can successfully lift the lid after loosening it and place it next to the bottle. The output neurons of the recurrent branch show slow oscillations during execution (see figure 7.6). The oscillations are slower than the turning motions and appear to serve as short-term memory, allowing the network to implicitly learn when the turning phase has been completed and the pick-place phase can be initiated, without requiring explicit higher-level programming or sub-task annotations.

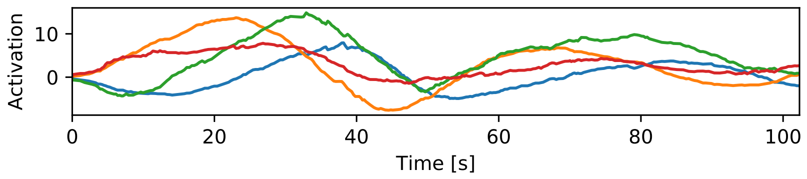


Figure 7.6.: Recurrent neural activations during a bottle opening task.

7.6.6. Camera vision

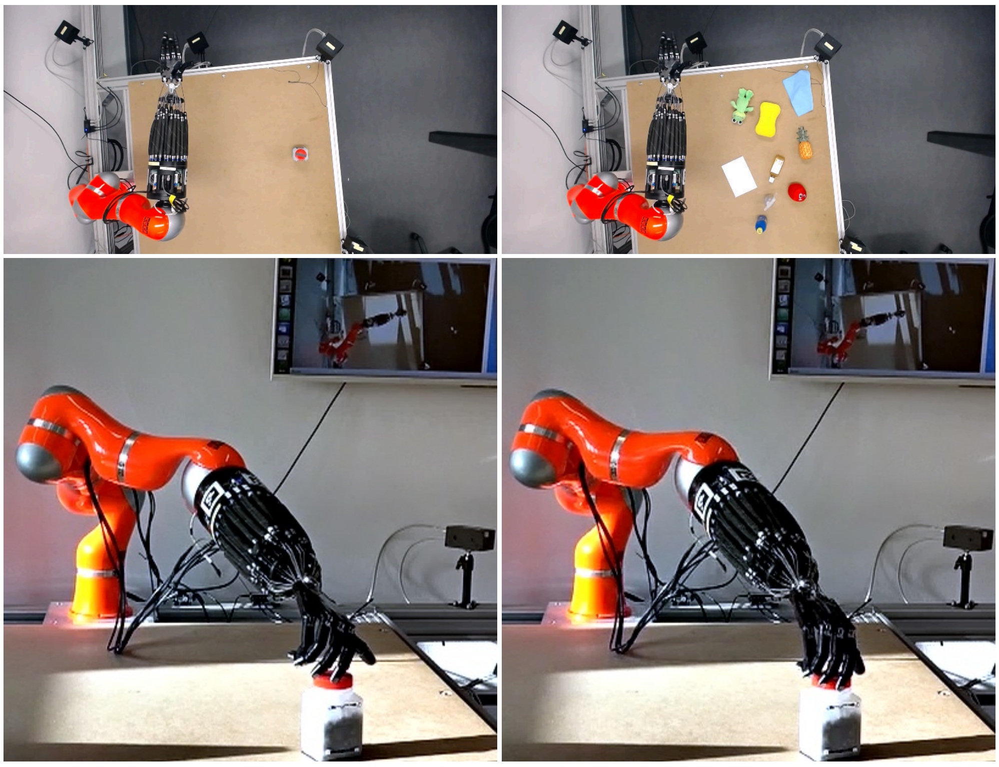


Figure 7.7.: A lab bottle is localized with an overhead camera (top) and opened with a robotic hand-arm system (bottom).

To enable fully autonomous operation, the visual transfer learning network (section 5.6) is trained with 18 images of the robot workspace. 14 of the training images show annotated observations of the target object (fig. 7.7, top-left) and 4 images include other objects as negative examples (fig. 7.7, top-right). The task can still be executed successfully when using the vision network for object localization. Though the training images are captured under even artificial lighting, the perception can generalize to changed illumination with sunlight and shadows (fig. 7.7, bottom).

7.6.7. Model-based learning

A new multimodal dataset is recorded for the bottle-opening task using a glove with fabric-based contact sensors and passive markers (see section 5.2 and figure 5.1), capturing finger tip positions, tactile data and lid pose. The data is used to train the model-based architecture. The object state model is trained for lid orientation. At runtime, the fingertip pressure sensors on the Shadow hand are used for tactile sensing. For increased stability, the network receives a mixture of equal proportions between predicted tactile outputs from the model network and tactile measurements from the robot hand. The learned behavior is successfully executed on a UR10 arm with a Shadow C6 hand. The robot approaches the object, turns the lid multiple times, lifts it up, and places it next to the bottle, as shown in figure 7.8. The multimodal architecture can adapt the finger motions to incorrect object positions by approximately 2 cm. Whereas the unimodal architectures occasionally fail to make sufficient contact with the object, the multimodal model-based architecture performs the task successfully over a sequence of 10 trials. However, this is not the case if the tactile sensors on the hand are disabled and only predicted tactile readings are used, or if tactile prediction is disabled and only tactile measurements from the pressure sensors on the robot hand are fed to the policy as tactile inputs. While, on an NVIDIA GTX 1080, the recurrent architecture requires 3 h of training time, the model-based architecture can be trained within 30 min.

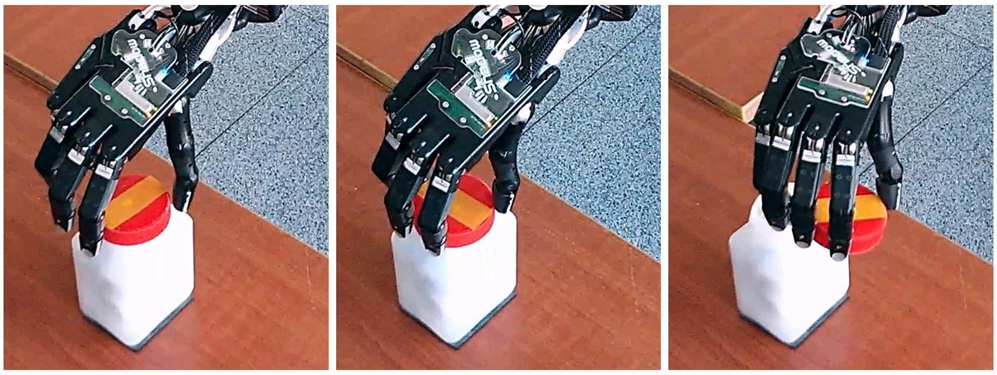


Figure 7.8.: Opening a lab bottle using the model-based learning architecture.

7.6.8. Bimanual precision assembly

The robot developed in chapter 6 is taught by human demonstration to install a miniature camera module on a circuit board. The task is demonstrated with two hands, using one hand to grasp and place the camera module and the other hand to fix the connector (see section 5.8). The board is subsequently lifted by the camera module to verify correct insertion. 15 demonstrations are recorded. Hands are swapped to capture tactile data for both grasping the object and fixing the connector using only one glove. Initial object positions are varied between demonstrations. See figure 7.9 for the initial sample distribution and figure 5.9 for time plots from two of the demonstrations. The data set is split into a training set consisting of three demonstrations and a validation set consisting of 12 demonstrations. The training set contains two tactile demonstrations for grasping the camera module and one tactile demonstration for pressing the connector. Since the robot gripper is mounted on the left arm (chapter 6), left-handed trajectories are sampled with a probability of 60% and right-handed trajectories are sampled with a probability of 40% during training. The training set is augmented by copying and mirroring each corresponding demonstration.

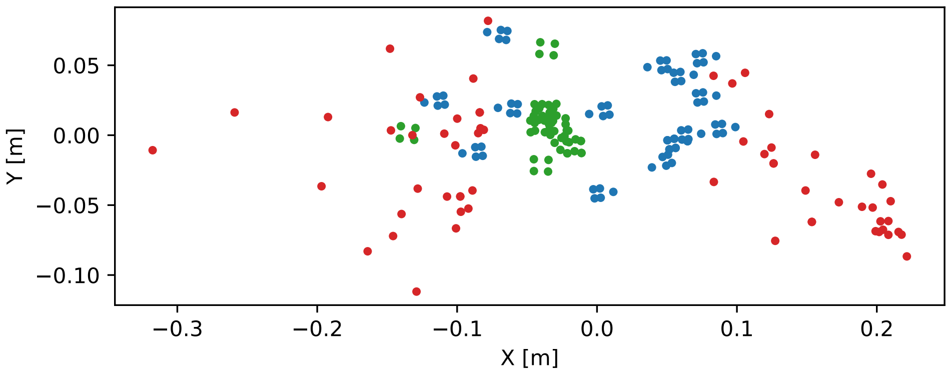


Figure 7.9.: Initial fingertip (red), circuit board (green) and camera (blue) keypoint positions in the assembly demonstration data set.

The task is learned from the training set using the methods described in section 7.2 with object dynamics (section 7.2.11), recurrent denoising (section 7.2.8), tactile prediction and pair-wise translation outputs. The behavior is first evaluated in simulated experiments by recursively feeding outputs back as inputs. Results for the proposed multimodal method (1) are compared to several baselines (2-10). Each simulation is initialized with the start state of a demonstration trajectory from the validation set. The simulated trajectory is evaluated against the demonstrated validation trajectory by computing the average Euclidean distance. For each keypoint and time step from one trajectory, the closest position for the same keypoint is found in the other trajectory, the distance is computed, and the distances are averaged over the entire trajectory. See table 7.1 for results.

Learning Method	Tactile	Memory	Observed masks Hand	Observed masks Object	Both masks Hand	Both masks Object
(1) Multimodal	Yes	Yes	0.010	0.0021	0.010	0.0021
(2) No tactile	No	Yes	0.010	0.0024	0.010	0.0024
(3) Single state	No	No	0.025	0.0044	0.025	0.0044
(4) Time window	No	No	0.015	0.0029	0.015	0.0030
(5) DMP	No	No	0.021	0.012	0.021	0.012
(6) Conditional ProMP	No	No	0.017	0.0082	0.021	0.0082
(7) Simple ProMP	No	No	0.026	0.014	0.026	0.014
(8) Diffusion U-Net	No	No	0.022	0.0036	0.022	0.0039
(9) Diffusion Transformer	No	No	0.017	0.0034	0.019	0.0030
(10) None	No	No	0.034	0.010	0.034	0.010

Table 7.1.: Distances between simulated trajectories and the validation set using the proposed learning method (1) and several baselines.

As a first baseline, the tactile modality is removed (2). Recurrent feedback is subsequently disabled as well (3). As an alternative to recurrent denoising, inputs are extended from a single time step to short time windows with a length of 16 samples (4). Earlier samples are averaged over lengths of powers of two (8, 4, 2, 1, 1 samples). An additional naive baseline is created by simply loading and returning one of the demonstration trajectories from the training set (10).

For comparison with state-of-the-art methods, the data is also used to train dynamic motion primitives (5,6,7) [88, 144, 48] and diffusion policies (8,9) [35].

The diffusion policies are based on either a U-Net 1D model (8) or a diffusion transformer (9). The observation space consists of 3D keypoint positions and the action space of 3D keypoint velocities. Hyperparameters are set to the default values from the original implementation [283, 284].

While the bottle opening task can not be learned by a single DMP alone due to the transition from repeating motions to a discrete pick-place action, the assembly task in this section could in principle be approximated by a DMP. A DMP [88] is trained with one of the training demonstrations (5). A probabilistic motion primitive (ProMP) [144] is trained using the entire training set. The ProMP is tested with (6) and without (7) conditioning on the start state. Occluded features are interpolated [55]. Results are computed for hand and object keypoints separately as well as with and without masking by predicted object keypoint occlusion.

If tactile and memory are both disabled (3), motions are stuck close to the start state. Using a fixed time window (4) instead of recurrent denoising (section 7.2.8) produces overall similar trajectories to the proposed method (1) but leads to reduced accuracy when adapting to new positions. The DMP (5) varies the initial approach trajectory but afterwards fails to adapt trajectories further. The conditional ProMP (6) generates different trajectories depending on the initial state but suffers from high validation errors. If the number of DMP/ProMP weights is increased, simulations are stuck after the initial pick-place phase. The naive baseline (8) and the non-conditioned

ProMP (7) do not adapt trajectories to new object positions. The diffusion transformer (9) performs slightly better than the U-Net model (8) and better than DMPs and ProMPs (5-7). However, it performs worse than the proposed method (1) and two of the baseline variants (2,4). Even in the above simulated experiment, removing tactile from the proposed method (1) slightly decreases accuracy (2) for object keypoints.

The three best-performing methods (1,2,4) are executed on the robot introduced in chapter 6. Object features are tracked using the vision network (section 5.6) and the camera system (section 5.4, seven cameras). The vision network is retrained to also detect three additional keypoints for the robot end-effectors (see figure 7.11). Using the baseline architectures (2,4), task execution fails, being unable to grasp the camera module, dropping the object when approaching the circuit board, or failing to correctly align the connector for successful insertion. The proposed learning method (1) successfully completes the task (see figure 7.10). The robot grasps the camera module, picking it up, placing it above the board, aligning the connectors, fixing the connector with the opposite arm, and lifting the assembly, demonstrating successful insertion. See figure 7.12 for recordings of sensor modalities, tactile predictions and a subset of recurrent activations during successful robot execution.

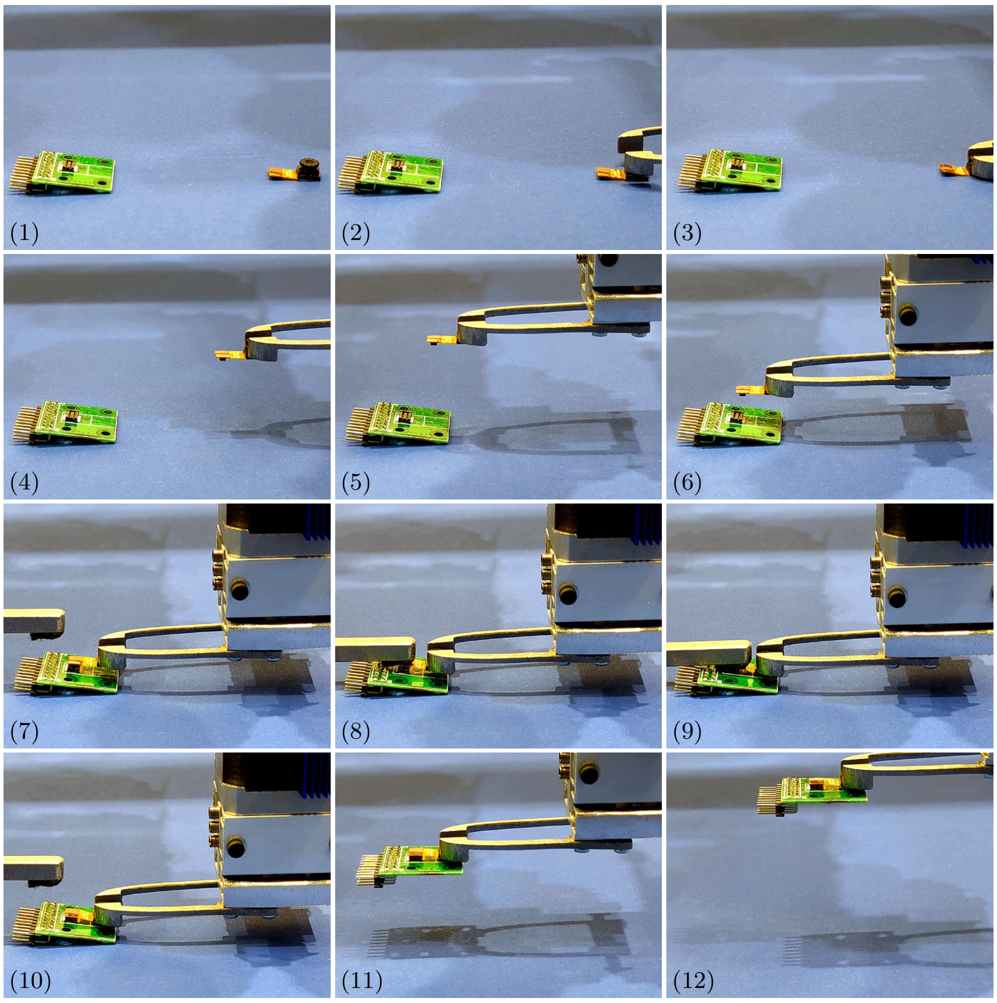


Figure 7.10.: A robot has been taught by demonstration with human hands to install a camera module on a circuit board.

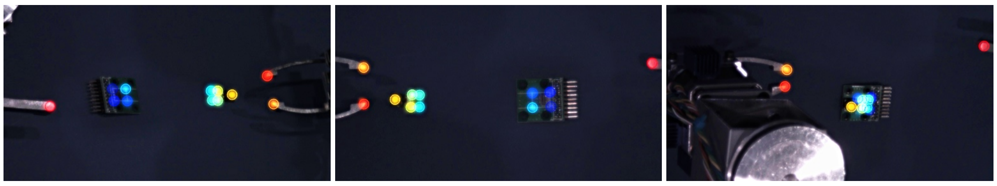


Figure 7.11.: Visual keypoint detection during robot execution of an assembly task.

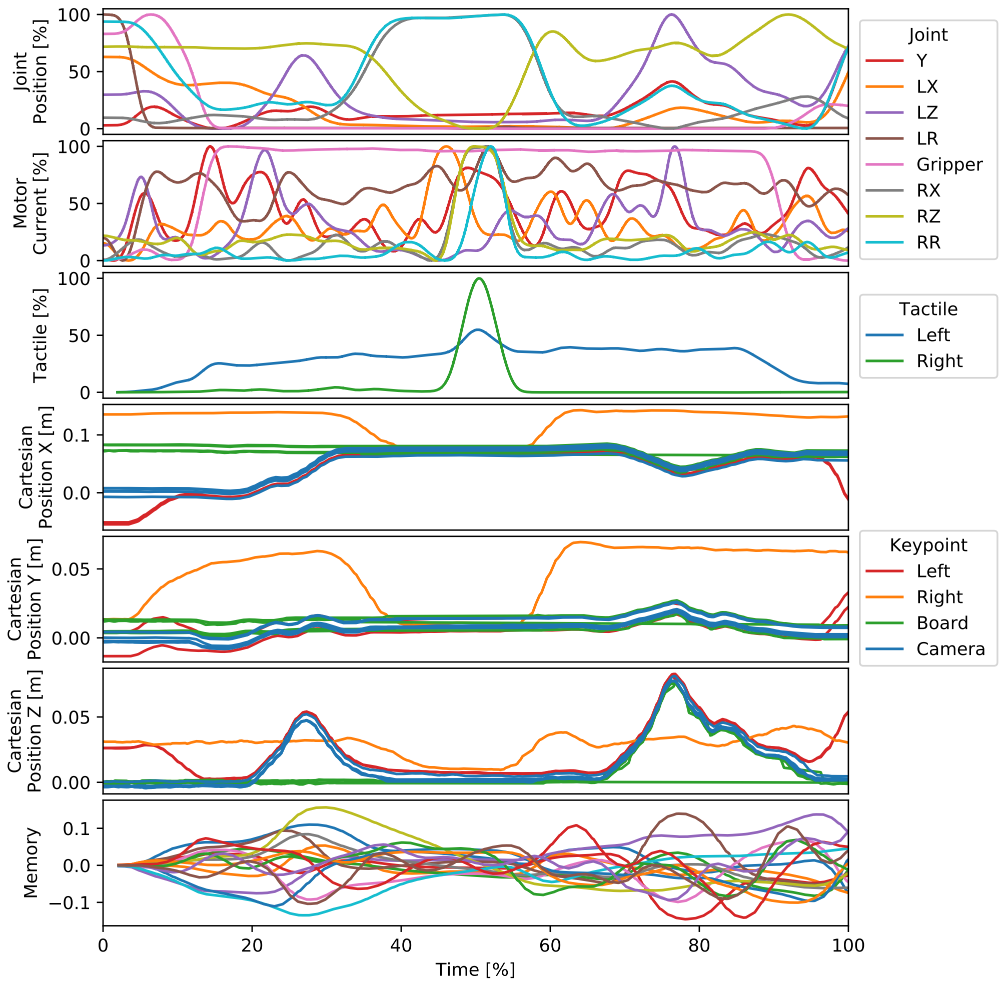


Figure 7.12.: Joint position measurements, motor currents, tactile predictions, keypoint positions and a subset of recurrent activations during successful robot execution of a bi-manual assembly task.

8. Conclusions and outlook

Machine learning of object manipulation tasks can benefit from multimodality. In virtual environments, learning and simulation can be combined into a single gradient-based optimization, simultaneously minimizing task loss and physical inconsistencies. This can enable an artificial neural network to directly tap into the simulator and modify contact variables to overcome local minima.

It is also possible to teach robots by simply demonstrating manipulation tasks with human hands in the real world, without having to program task-specific simulation environments. Learned denoising does not only offer a solution for one-to-many mappings but can also generate corrective actions from previously unseen states. Prior knowledge about physically plausible perturbations is not necessary. By deferring consistency constraints to inference, it is possible to combine denoising and object dynamics into a single supervised learning problem. Memory can be modeled as an additional sensor modality without requiring backpropagation through time or into large context windows. Contact information recorded from human hands can improve robot performance, even if it is only available for parts of the demonstration data. It is possible to produce tactile matrix sensors completely from stretchable rubber-like materials and in three-dimensional multi-curved shapes. These can be processed into sensor gloves, forming artificial second skins around human hands to capture contact information during teaching. Behaviors learned from human demonstration can be transferred to humanoid robot hands and to non-humanoid hardware. Crossmodal architectures with corresponding outputs for all input modalities also enable recursive data-driven simulated rollouts for evaluation prior to robot execution. The methods are applicable to dexterous manipulation beyond pure grasping.

The optimization framework from chapter 2 and a simplified version of the differentiable contact model are further evaluated on human motion reconstruction tasks in [293] and shown to outperform state-of-the-art methods. An early prototype of the online trajectory optimizer from chapter 7 is re-used in human interaction studies [291, 294] and successfully adapted for avoiding collisions with humans.

Future work towards practical application of the learning and teaching methods might benefit from a cross-platform re-implementation and the development of a unified graphical user interface. Advanced customization and hardware configuration could be facilitated through optional integration with graphical scripting (see [231, 79, 285, 286]). Visualization of data-driven simulations could possibly be enhanced through keypoint-conditioned image generation [204]. Estimating manufacturing cost for the sensor glove and the feasibility of automated production might require further experiments. An alternative route could be the development of pre-trained visuo-tactile foundation models for visual-only fine-tuning.

9. References

Print resources

[1] Pieter Abbeel and Andrew Ng. “Apprenticeship Learning via Inverse Reinforcement Learning”. In: *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004* (Sept. 2004). DOI: 10.1007/978-0-387-30164-8_417.

[2] *ACES Primer - Understanding and Integrating the Academy Color Encoding System.* Academy of Motion Picture Arts and Sciences (A.M.P.A.S.), 2018.

[3] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter. “Differentiable Convex Optimization Layers”. In: *Advances in Neural Information Processing Systems.* 2019.

[4] Alin Albu-Schäeffer, Sami Haddadin, Christian Ott, Andreas Stemmer, Thomas Wimböck, and Gerd Hirzinger. “The DLR Lightweight Robot – Design and Control Concepts for Robots in Human Environments”. In: *Industrial Robot: An International Journal* 34 (Aug. 2007), pp. 376–385. DOI: 10.1108/01439910710774386.

[5] Robin Amsters and Peter Slaets. “Turtlebot 3 as a Robotics Education Platform”. In: *Robotics in Education.* Springer International Publishing, Jan. 2020, pp. 170–181. ISBN: 978-3-030-26944-9. DOI: 10.1007/978-3-030-26945-6_16.

[6] Thomas Timm Andersen. *Optimizing the Universal Robots ROS driver.* English. Technical University of Denmark, Department of Electrical Engineering, 2015.

[7] Joel Andersson, Johan Åkesson, and Moritz Diehl. “CasADi: A Symbolic Package for Automatic Differentiation and Optimal Control”. In: *Recent Advances in Algorithmic Differentiation* (2012), pp. 297–307.

[8] Barry Arkles, Jonathan Goff, Santy Sulaiman, and Alison Phillips. “Ultra-High Elongation Silicone Elastomers”. In: *Rubber World* 254 (June 2016), pp. 29–34.

[9] Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. “Dexterous Imitation Made Easy: A Learning-Based Framework for Efficient Dexterous Manipulation”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA).* 2023, pp. 5954–5961. DOI: 10.1109/ICRA48891.2023.10160275.

[10] Rickard Arvidsson, Max Boholm, Mikael Johansson, and Monica Montoya. “"Just Carbon": Ideas About Graphene Risks by Graphene Researchers and Innovation Advisors”. In: *NanoEthics* 12 (Dec. 2018). DOI: 10.1007/s11569-018-0324-y.

[11] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. “End-to-End Differentiable Physics for Learning and Control”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018.

[12] Steve Bako, Thijs Vogels, Brian Mcwilliams, Mark Meyer, Jan NováK, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. “Kernel-predicting convolutional networks for denoising Monte Carlo renderings”. In: *ACM Trans. Graph.* 36.4 (July 2017). ISSN: 0730-0301. DOI: 10.1145/3072959.3073708.

[13] Sai Bangaru, Lifan Wu, Tzu-Mao Li, Jacob Munkberg, Gilbert Bernstein, Jonathan Ragan-Kelley, Fredo Durand, Aaron Lefohn, and Yong He. “SLANG.D: Fast, Modular and Differentiable Shader Programming”. In: *ACM Transactions on Graphics (SIGGRAPH Asia)* 42.6 (Dec. 2023), pp. 1–28. DOI: 10.1145/3618353.

[14] Edoardo Battaglia, Matteo Bianchi, Alessandro Altobelli, Giorgio Grioli, Manuel G. Catalano, Alessandro Serio, Marco Santello, and Antonio Bicchi. “ThimbleSense: A Fingertip-Wearable Tactile Sensor for Grasp Analysis”. In: *IEEE Transactions on Haptics* 9.1 (2016), pp. 121–133. DOI: 10.1109/TOH.2015.2482478.

[15] Patrick Beeson and Barrett Ames. “TRAC-IK: An Open-Source Library for Improved Solving of Generic Inverse Kinematics”. In: *Proc. IEEE RAS Humanoids Conference*. Seoul, Korea, Nov. 2015.

[16] Jan Bender, Matthias Müller, and Miles Macklin. “Position-Based Simulation Methods in Computer Graphics”. In: *EG 2015 - Tutorials*. The Eurographics Association, May 2015. DOI: 10.2312/egt.20151045.

[17] Gino Bergen. “Proximity queries and penetration depth computation on 3d game objects”. In: *In Game developers conference, volume 170* (Jan. 2001).

[18] Alexandre Bernardino, Marco Henriques, Norman Hendrich, and Jianwei Zhang. “Precision grasp synergies for dexterous robotic hands”. In: *Proc. IEEE International Conference on Robotics and Biomimetics*. Dec. 2013, pp. 62–67.

[19] Felix Blaschke. “Das Verfahren der Feldorientierung zur Regelung der Drehfeldmaschine”. Dissertation. PhD thesis. Technische Universitat Braunschweig, Dec. 1973.

[20] Aleksandar Botev, Hippolyt Ritter, and David Barber. “Practical Gauss-Newton Optimisation for Deep Learning”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 557–565.

[21] Clementine Boutry, Yukitoshi Kaizawa, Bob Schroeder, Alex Chortos, Anais Legrand, Zhen Wang, James Chang, Paige Fox, and Zhenan Bao. “A stretchable and biodegradable strain and pressure sensor for orthopaedic application”. In: *Nature Electronics* 1 (May 2018). DOI: 10.1038/s41928-018-0071-7.

[22] Bernhard Brunner. “Programming robots via learning by showing in a virtual environment”. In: *Proc. Virtual REality World*. 1995.

[23] Herman Bruyninckx, Peter Soetens, and Bob Koninckx. “The Real-Time Motion Control Core of the Orocos Project”. In: *IEEE International Conference on Robotics and Automation*. 2003, pp. 2766–2771.

[24] H. C. Burger and P. H. van Cittert. “Wahre und scheinbare Intensitätsverteilung in Spektrallinien”. In: *Zeitschrift für Physik* 79.11 (Nov. 1932), pp. 722–730. ISSN: 0044-3328. DOI: 10.1007/BF01340490.

[25] Gereon Büscher, Risto Kõiva, Carsten Schürmann, Robert Haschke, and Helge J. Ritter. “Tactile dataglove with fabric-based sensors”. In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. 2012, pp. 204–209. DOI: 10.1109/HUMANOIDS.2012.6651521.

[26] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. “The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives”. In: *SII 2019 - International Symposium on System Integrations*. Paris, France, Jan. 2019.

[27] Guillermo A. Castillo, Bowen Weng, Shunpeng Yang, Wei Zhang, and Ayonga Hereid. “Template Model Inspired Task Space Learning for Robust Bipedal Locomotion”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 8582–8589. DOI: 10.1109/IROS55552.2023.10341263.

[28] Emil Cătinaş. “Inexact Perturbed Newton Methods and Applications to a Class of Krylov Solvers”. In: *Journal of Optimization Theory and Applications* 108 (Mar. 2001), pp. 543–570. DOI: 10.1023/A:1017583307974.

[29] E. Catmull and J. Clark. “Recursively generated B-spline surfaces on arbitrary topological meshes”. In: *Computer-Aided Design* 10.6 (1978), pp. 350–355. ISSN: 0010-4485. DOI: 10.1016/0010-4485(78)90110-0.

[30] Chunyang Chang, Kevin Haninger, Yunlei Shi, Chengjie Yuan, Zhaopeng Chen, and Jianwei Zhang. “Impedance Adaptation by Reinforcement Learning with Contact Dynamic Movement Primitives”. In: *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)* (2022), pp. 1185–1191.

[31] Ricky T. Q. Chen, Brandon Amos, and Maximilian Nickel. “Learning Neural Event Functions for Ordinary Differential Equations”. In: *International Conference on Learning Representations* (2021).

[32] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. “Neural Ordinary Differential Equations”. In: *Advances in Neural Information Processing Systems* (2018).

[33] Sheng-Wei Chen, Chun-Nan Chou, and Edward Y. Chang. “EA-CG: An Approximate Second-Order Method for Training Fully-Connected Neural Networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019.

[34] Ying Chen, Leizhi Jin, Peng Wang, and Haibin Liu. “Selection of Optimal Hyperparameter for Detecting Multiple Contacts from Large-Area Tactile Sensors Based on Electrical Impedance Tomography”. In: *Engineering Research Express* 5 (Mar. 2023). DOI: 10.1088/2631-8695/acc515.

[35] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. “Diffusion Policy: Visuomotor Policy Learning via Action Diffusion”. In: *Proceedings of Robotics: Science and Systems (RSS)*. 2023.

[36] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. “Universal Manipulation Interface: In-The-Wild Robot Teaching Without In-The-Wild Robots”. In: *Proceedings of Robotics: Science and Systems (RSS)*. 2024.

[37] Sachin Chitta et al. “ros_control: A generic and simple control framework for ROS”. In: *The Journal of Open Source Software* (Dec. 2017). DOI: 10.21105/joss.00456.

[38] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. “Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study”. In: *Journal of Software Engineering for Robotics* (Apr. 2014).

[39] Lin Cong, Michael Görner, Philipp Ruppel, Hongzhuo Liang, Norman Hendrich, and Jianwei Zhang. “Self-Adapting Recurrent Models for Object Pushing from Learning in Simulation”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5304–5310. DOI: 10.1109/IROS45743.2020.9341076.

[40] R. L. Cook and K. E. Torrance. “A Reflectance Model for Computer Graphics”. In: *ACM Trans. Graph.* 1.1 (Jan. 1982), pp. 7–24. ISSN: 0730-0301. DOI: 10.1145/357290.357293.

[41] R. Cortesao, R. Koeppe, U. Nunes, and G. Hirzinger. “Data fusion for robotic assembly tasks based on human skills”. In: *IEEE Transactions on Robotics* 20.6 (2004), pp. 941–952. DOI: 10.1109/TRO.2004.832789.

[42] N. S. Danial, Muhammad Mahyiddin Ramli, Dewi Suriyani Che Halin, Honghong Hong, S. Salwa M. Isa, Mohd Mustafa Al Bakri Abdullah, N. Asyikin M. Anhar, L. F. A. Talip, and N. S. Mazlan. “Incorporation of polydimethylsiloxane with reduced graphene oxide and zinc oxide for tensile and electrical properties”. In: *AIP Conference Proceedings*. 2017.

[43] L. Dreschler and H.-H. Nagel. “Volumetric model and 3D trajectory of a moving car derived from monocular TV frame sequences of a street scene”. In: *Computer Graphics and Image Processing* 20.3 (1982), pp. 199–228. ISSN: 0146-664X. DOI: 10.1016/0146-664X(82)90081-8.

[44] Leonie Dreschler. “Zur Reproduzierbarkeit von markanten Bildpunkten bei der Auswertung von Realwelt-Bildfolgen”. In: *Modelle und Strukturen*. Ed. by Bernd Radig. Berlin, Heidelberg: Springer Berlin Heidelberg, 1981, pp. 76–82. ISBN: 978-3-642-68138-7.

[45] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. “Incorporating second-order functional knowledge for better option pricing”. In: *Proceedings of the 13th International Conference on Neural Information Processing Systems*. NIPS’00. Denver, CO: MIT Press, 2000, pp. 451–457.

[46] Andrius Dzedzickis, Ernestas Sutinys, Vytautas Bucinskas, Urte Samukaite-Bubniene, Baltramiejus Jakstys, Arunas Ramanavicius, and Inga Morkvenaite-Vilkonciene. “Polyethylene-Carbon Composite (Velostat®) Based Tactile Sensor”. In: *Polymers* 12.12 (2020). DOI: `10.3390/polym12122905`.

[47] Kenneth Emancipator and Martin H. Kroll. “A quantitative measure of nonlinearity.” In: *Clinical chemistry* 39 5 (1993), pp. 766–72.

[48] Alexander Fabisch. “movement_primitives: Imitation Learning of Cartesian Motion with Movement Primitives”. In: *Journal of Open Source Software* 9.97 (2024), p. 6695. DOI: `10.21105/joss.06695`.

[49] Kayson Fakhar, Fatemeh Hadäghi, and Claus-Christian Hilgetag. “Causal Influences Decouple From Their Underlying Network Structure In Echo State Networks”. English. In: *2022 International Joint Conference on Neural Networks (IJCNN)*. 1st ed. Vol. 2022. IEEE, 2022, pp. 1–8. ISBN: 978-1-66549-526-4. DOI: `10.1109/IJCNN55064.2022.9892782`.

[50] Anis Fatema, Ivin Kuriakose, Deeksha Devendra, and Aftab Hussain. “Investigation of the Mechanical Reliability of a Velostat-Based Flexible Pressure Sensor”. In: *2022 IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS)*. June 2022. DOI: `10.1109/FLEPS53764.2022.9781575`.

[51] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2007. ISBN: 0387743146.

[52] Anthony V. Fiacco and Garth P. McCormick. *Nonlinear programming: Sequential unconstrained minimization techniques*. Society for Industrial and Applied Mathematics, Jan. 1968.

[53] Jeremy A. Fishel, Veronica J. Santos, and Gerald E. Loeb. “A robust micro-vibration sensor for biomimetic fingertips”. In: *2008 2nd IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics*. 2008, pp. 659–663. DOI: `10.1109/BIOROB.2008.4762917`.

[54] Ragnar Frisch. “The Multiplex Method for Linear Programming”. In: *The Indian Journal of Statistics* (Sept. 1957), pp. 329–362.

[55] F. N. Fritsch and J. Butland. “A Method for Constructing Local Monotone Piecewise Cubic Interpolants”. In: *SIAM Journal on Scientific and Statistical Computing* 5.2 (1984), pp. 300–304. DOI: `10.1137/0905021`.

[56] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. “Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation”. In: *Conference on Robot Learning (CoRL)*. 2024.

[57] Kunihiko Fukushima. “Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements”. In: *IEEE Transactions on Systems Science and Cybernetics* 5.4 (1969), pp. 322–333. DOI: `10.1109/TSSC.1969.300225`.

[58] Claudio Gallicchio. "Euler State Networks: Non-dissipative Reservoir Computing". In: *Neurocomputing* 579 (2024), p. 127411. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2024.127411.

[59] Ge Gao, Mikko Lauri, Yulong Wang, Xiaolin Hu, Jianwei Zhang, and Simone Frintrop. "6D Object Pose Regression via Supervised Learning on Point Clouds". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 3643–3649. DOI: 10.1109/ICRA40945.2020.9197461.

[60] *Gates PowerGrip GT2 Drive Design Manual*. The Gates Rubber Company.

[61] Carl Friedrich Gauss. *Abhandlungen zur Methode der kleinsten Quadrate*. Berlin: Stankiewicz, 1887.

[62] Carl Friedrich Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Hamburg, Germany: Perthes et Besser, 1809.

[63] Prosenjit Kumar Ghosh and Prabha Sundaravadivel. "Stretchable Sensors for Soft Robotic Grippers in Edge-Intelligent IoT Applications". In: *Sensors* 23.8 (2023). ISSN: 1424-8220. DOI: 10.3390/s23084039.

[64] 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* 4.2 (1988), pp. 193–203. DOI: 10.1109/56.2083.

[65] Jonathan Goff, Santy Sulaiman, Barry Arkles, and James P. Lewicki. "Soft materials with recoverable shape factors from extreme distortion states". In: *Advanced Materials* 28.12 (Jan. 2016). DOI: 10.1002/adma.201503320.

[66] Claudia González, J. Solanes, Adolfo Muñoz, Luis Gracia, Vicent Girbés Juan, and Josep Tornero. "Advanced teleoperation and control system for industrial robots based on augmented virtuality and haptic feedback". In: *Journal of Manufacturing Systems* 59 (Apr. 2021), pp. 283–298. DOI: 10.1016/j.jmsy.2021.02.013.

[67] Michael Görner*, Robert Haschke*, Helge Ritter, and Jianwei Zhang. "MoveIt! Task Constructor for Task-Level Motion Planning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019.

[68] H. Gouraud. "Continuous Shading of Curved Surfaces". In: *IEEE Trans. Comput.* 20.6 (June 1971), pp. 623–629. ISSN: 0018-9340. DOI: 10.1109/T-C.1971.223313.

[69] Serge Gratton, Amos Lawless, and Nancy Nichols. "Approximate Gauss-Newton Methods for Nonlinear Least Squares Problems". In: *SIAM Journal on Optimization* 18 (Jan. 2007), pp. 106–132. DOI: 10.1137/050624935.

[70] Markus Grebenstein, Maxime Chalon, Gerd Hirzinger, and Roland Siegwart. "Antagonistically driven finger design for the anthropomorphic DLR Hand Arm System". In: *2010 10th IEEE-RAS International Conference on Humanoid Robots*. 2010, pp. 609–616. DOI: 10.1109/ICHR.2010.5686342.

[71] Larry Gritz and Eugene d'Eon. "The Importance of Being Linear". In: *GPU Gems 3*. 2007. ISBN: 9780321515261.

[72] Sonja Gross, Diego Hidalgo-Carvajal, Silija Breimann, Nicolai Stein, A. Ganguly, Djallil Naceri, and Sami Haddadin. "Soft Sensing Skins for Arbitrary Objects: An Automatic Framework". In: May 2023, pp. 12507–12513. DOI: `10.1109/ICRA48891.2023.10161344`.

[73] Guoying Gu, Haipeng Xu, S. Peng, L. Li, Sujie Chen, Tongqing Lu, and Xiaojun Guo. "Integrated Soft Ionotronic Skin with Stretchable and Transparent Hydrogel-Elastomer Ionic Sensors for Hand-Motion Monitoring." In: *Soft robotics* 6 3 (2019), pp. 368–376.

[74] Arthur Haffemayer, Armand Jordana, Médéric Fourmy, Krzysztof Wojciechowski, Guilhem Saurel, Vladimír Petrík, Florent Lamiraux, and Nicolas Mansard. "Model predictive control under hard collision avoidance constraints for a robotic arm". In: *Ubiquitous Robots 2024*. Korea Robotics Society. New York (USA), United States, June 2024. DOI: `10.1109/UR61395.2024.10597485`.

[75] M.S Al-Haik, H Garmestani, and I.M Navon. "Truncated-Newton training algorithm for neurocomputational viscoplastic model". In: *Computer Methods in Applied Mechanics and Engineering* 192.19 (2003), pp. 2249–2267. ISSN: 0045-7825. DOI: `10.1016/S0045-7825(03)00261-5`.

[76] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. *DexPilot: Vision-Based Teleoperation of Dexterous Robotic Hand-Arm System.* 2020. DOI: `10.1109/ICRA40945.2020.9197124`.

[77] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. "NeuralSim: Augmenting Differentiable Simulators with Neural Networks". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2021.

[78] Eric Heitz. "Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs". In: *Journal of Computer Graphics Techniques (JCGT)* 3.2 (June 2014), pp. 48–107. ISSN: 2331-7418.

[79] N. Hendrich. "A Java-Based Framework for Simulation and Teaching: HADES — the Hamburg Design System". In: *Microelectronics Education.* Ed. by B. Courtois, N. Guillemot, G. Kamarinos, and G. Stéhelin. Dordrecht: Springer Netherlands, 2000, pp. 285–288. ISBN: 978-94-015-9506-3.

[80] Norman Hendrich and Alexandre Bernardino. "Affordance-Based Grasp Planning for Anthropomorphic Hands from Human Demonstration". In: *Proc. ROBOT2013: First Iberian Robotics Conference.* 2014, pp. 687–701.

[81] Jaime Hernandez, Md. Samiul Sunny, Javier Sanjuan, Ivan Rulik, Ishrak Islam, Sheikh Ahamed, Helal Ahmed, and Mohammad Rahman. "Current Designs of Robotic Arm Grippers: A Comprehensive Systematic Review". In: *Robotics* 12 (Jan. 2023), p. 5. DOI: `10.3390/robotics12010005`.

[82] Magnus R. Hestenes and Eduard Stiefel. “Methods of conjugate gradients for solving linear systems”. In: *Journal of research of the National Bureau of Standards* 49 (1952), pp. 409–435.

[83] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising Diffusion Probabilistic Models”. In: *NIPS’20: Proceedings of the 34th International Conference on Neural Information Processing Systems* (2020).

[84] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. “DiffTaichi: Differentiable Programming for Physical Simulation”. In: *ACM Transactions on Graphics (TOG)* (2019).

[85] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. “Taichi: a language for high-performance computation on spatially sparse data structures”. In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), p. 201.

[86] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. “ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics”. In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)* (2019).

[87] Peter J. Huber. “Robust estimation of a location parameter”. In: *Annals of Mathematical Statistics* 35.1 (Mar. 1964), pp. 73–101. ISSN: 0003-4851. DOI: 10.1214/aoms/1177703732.

[88] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. “Learning Attractor Landscapes for Learning Motor Primitives”. In: *Proc. Advances in Neural Information Processing Systems.* Jan. 2002, pp. 1523–1530.

[89] Manuel Iori, Vinicius Loti de Lima, Silvano Martello, and Michele Monaci. “2DPackLib: a two-dimensional cutting and packing library”. In: *Optimization Letters* 16 (Mar. 2022), pp. 1–10. DOI: 10.1007/s11590-021-01808-y.

[90] *IS 14533: Manipulating industrial robots - Performance criteria related test methods.* India: Bureau of Indian Standards, 2005.

[91] Divye Jain, Andrew Li, Shivam Singhal, Aravind Rajeswaran, Vikash Kumar, and Emanuel Todorov. “Learning Deep Visuomotor Policies for Dexterous Hand Manipulation”. In: *2019 International Conference on Robotics and Automation (ICRA).* 2019, pp. 3636–3643. DOI: 10.1109/ICRA.2019.8794033.

[92] Haiyang Jin, Liwei Zhang, Sebastian Rockel, Jun Zhang, Ying Hu, and Jianwei Zhang. “A novel optical tracking based tele-control system for tabletop object manipulation tasks”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2015, pp. 636–642. DOI: 10.1109/IROS.2015.7353439.

[93] Lill Maria Gjerde Johannessen, Mathias Hauan Arbo, and Jan Tommy Gravdahl. “Robot Dynamics with URDF & CasADi”. In: *2019 7th International Conference on Control, Mechatronics and Automation (ICCMA).* IEEE. 2019.

[94] Yannick Jonetzko, Judith Hartfill, Niklas Fiedler, Fangwei Zhong, Frank Steinicke, and Jianwei Zhang. “Evaluating Visual and Auditory Substitution of Tactile Feedback during Mixed Reality Teleoperation”. In: *1st International Conference on Cognitive Computation and Systems (ICCCS 2022)*. 2022.

[95] W. Kabsch. “A solution for the best rotation to relate two sets of vectors”. In: *Acta Crystallographica Section A* 32.5 (1976), pp. 922–923. DOI: 10.1107/S0567739476001873.

[96] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. “STOMP: Stochastic trajectory optimization for motion planning”. In: *Proc. IEEE International Conference on Robotics and Automation*. 2011, pp. 4569–4574. DOI: 10.1109/ICRA.2011.5980280.

[97] Antreas Kantaros, Florian Ion Tiberiu Petrescu, Konstantinos Brachos, Theodore Ganetsos, and Nicolae Petrescu. “Evaluating Benchtop Additive Manufacturing Processes Considering Latest Enhancements in Operational Factors”. In: *Processes* 12.11 (2024), p. 2334.

[98] N. Karmarkar. “A New Polynomial-Time Algorithm for Linear Programming”. In: *Combinatorica* 4.4 (Dec. 1984), pp. 373–395. ISSN: 0209-9683. DOI: 10.1007/BF02579150.

[99] Benjamin Katz. “A low cost modular actuator for dynamic robots”. In: *Graduate Theses*. Massachusetts Institute of Technology, 2018.

[100] Marek Kciuk, Zygmunt Kowalik, Grazia Lo Sciuto, Sebastian Sławski, and Stefano Mastrostefano. “Intelligent Medical Velostat Pressure Sensor Mat Based on Artificial Neural Network and Arduino Embedded System”. In: *Applied System Innovation* 6.5 (2023). ISSN: 2571-5577. DOI: 10.3390/asi6050084.

[101] R. Keys. “Cubic convolution interpolation for digital image processing”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.6 (1981), pp. 1153–1160. DOI: 10.1109/TASSP.1981.1163711.

[102] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *Proc. International Conference for Learning Representations*. Dec. 2014.

[103] R. Koeppe, A. Breidenbach, and G. Hirzinger. “Skill representation and acquisition of compliant motions using a teach device”. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS ’96*. Vol. 2. 1996, 897–904 vol.2. DOI: 10.1109/IROS.1996.571071.

[104] Mike Lambeta et al. “DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor with Application to In-Hand Manipulation”. In: *IEEE Robotics and Automation Letters* PP (Feb. 2020), pp. 1–1. DOI: 10.1109/LRA.2020.2977257.

[105] R. E. Langer. “On the determination of earth conductivity from observed surface potentials”. In: *Bulletin of the American Mathematical Society* 42.10 (1936), pp. 747–754.

[106] Hemin Latif, Nasser Sherkat, and Ahmad Lotfi. “Teleoperation through Eye Gaze (Tele Gaze): A Multimodal Approach”. In: *2009 IEEE International Conference on Robotics and Biomimetics, ROBIO 2009.* Jan. 2010, pp. 711–716. DOI: 10.1109/ROBIO.2009.5420585.

[107] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[108] Hyung-Kew Lee, Sun-Il Chang, and Euisik Yoon. “A Flexible Polymer Tactile Sensor: Fabrication and Modular Expandability for Large Area Deployment”. In: *Journal of Microelectromechanical Systems* 15.6 (2006), pp. 1681–1686. DOI: 10.1109/JMEMS.2006.886021.

[109] Nathan Lepora. “Soft Biomimetic Optical Tactile Sensing with the TacTip: A Review”. In: *IEEE Sensors Journal* 21 (Oct. 2021), pp. 21131–21143. DOI: 10.1109/JSEN.2021.3100645.

[110] Kenneth Levenberg. “A Method for the Solution of Certain Non-Linear Problems in Least Squares”. In: *Quarterly of Applied Mathematics* 2 (1944), pp. 164–168.

[111] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. “Least squares conformal maps for automatic texture atlas generation”. In: *ACM Trans. Graph.* 21.3 (July 2002), pp. 362–371. ISSN: 0730-0301. DOI: 10.1145/566654.566590.

[112] Haoran Li, Christopher J. Ford, Matteo Bianchi, Manuel G. Catalano, Efi Psomopoulou, and Nathan F. Lepora. “BRL/Pisa/IIT SoftHand: A Low-Cost, 3D-Printed, Underactuated, Tendon-Driven Hand With Soft and Adaptive Synergies”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 8745–8751. DOI: 10.1109/LRA.2022.3187876.

[113] Shuang Li, N. Hendrich, Hongzhuo Liang, Philipp Ruppel, Changshui Zhang, and Jianwei Zhang. “A Dexterous Hand-Arm Teleoperation System Based on Hand Pose Estimation and Active Vision”. In: *IEEE Transactions on Cybernetics* PP (Sept. 2022), pp. 1–12. DOI: 10.1109/TCYB.2022.3207290.

[114] Shuang Li, Jiaxi Jiang, Philipp Ruppel, Hongzhuo Liang, Xiaojian Ma, Norman Hendrich, Fuchun Sun, and Jianwei Zhang. *A Mobile Robot Hand-Arm Teleoperation System by Vision and IMU.* Las Vegas, NV, USA, 2020. DOI: 10.1109/IROS45743.2020.9340738.

[115] Shuang Li, Xiaojian Ma, Hongzhuo Liang, Michael Görner, Philipp Ruppel, Bin Fang, Fuchun Sun, and Jianwei Zhang. “Vision-based Teleoperation of Shadow Dexterous Hand using End-to-End Deep Neural Network”. In: *2019 International Conference on Robotics and Automation (ICRA)* (2018), pp. 416–422.

[116] Tingguang Li et al. “Learning to Solve a Rubik’s Cube with a Dexterous Hand”. In: *Proc. IEEE International Conference on Robotics and Biomimetics.* Dec. 2019.

[117] Xiaoye Sherry Li. “An overview of SuperLU: Algorithms, implementation, and user interface”. In: *ACM Trans. Math. Softw.* 31 (2003), pp. 302–325.

[118] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. “SSD: Single Shot MultiBox Detector”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Cham: Springer International Publishing, 2016, pp. 21–37.

[119] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60 (2004), pp. 91–110.

[120] Camillo Lugaresi et al. “MediaPipe: A Framework for Perceiving and Processing Reality”. In: *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019*. 2019.

[121] Lingni Ma et al. “Nymeria: A Massive Collection of Multimodal Egocentric Daily Motion in the Wild”. In: *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XXIV*. Milan, Italy: Springer-Verlag, 2024, pp. 445–465. ISBN: 978-3-031-72690-3. DOI: 10.1007/978-3-031-72691-0_25.

[122] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. “Joint-dependent local deformations for hand animation and object grasping”. In: *Proceedings on Graphics Interface ’88*. Edmonton, Alberta, Canada: Canadian Information Processing Society, 1989, pp. 26–33.

[123] Rabbia Mahum, Faisal Shafique Butt, Kashif Ayyub, Seema Islam, Marriam Nawaz, and Daud Abdullah. “A review on humanoid robots”. In: *International Journal of Advanced and Applied Sciences* 4 (Feb. 2017), pp. 83–90. DOI: 10.21833/ijaas.2017.02.015.

[124] Khaled Mamou and Faouzi Ghorbel. “A simple and efficient approach for 3D mesh approximate convex decomposition”. In: *2009 16th IEEE International Conference on Image Processing (ICIP)* (2009), pp. 3501–3504.

[125] Philip Mason, Frank Uhlig, Václav Vaněk, Tillmann Buttersack, Sigurd Bauerecker, and Pavel Jungwirth. “Coulomb explosion during the early stages of the reaction of alkali metals with water”. In: *Nature chemistry* 7 (Mar. 2015), pp. 250–4. DOI: 10.1038/nchem.2161.

[126] Piotr Mazurek, Sindhu Vudayagiri, and Anne Ladegaard Skov. “How to tailor flexible silicone elastomers with mechanical integrity: a tutorial review.” In: *Chemical Society reviews* 48 6 (2019), pp. 1448–1464.

[127] Philipp Mittendorfer and Gordon Cheng. “Uniform Cellular Design of Artificial Robotic Skin”. In: *ROBOTIK 2012; 7th German Conference on Robotics*. 2012, pp. 1–5.

[128] Igor Mordatch, Zoran Popović, and Emanuel Todorov. “Contact-invariant optimization for hand manipulation”. In: *Proc. Eurographics conference on Computer Animation*. July 2012, pp. 137–144.

[129] Igor Mordatch, Emanuel Todorov, and Zoran Popović. “Discovery of Complex Behaviors through Contact-Invariant Optimization”. In: *ACM Transactions on Graphics* 31 (July 2012). DOI: 10.1145/2185520.2185539.

[130] Robert Moučka, Michal Sedlačík, Josef Osička, and Vladimir Pata. "Mechanical properties of bulk Sylgard 184 and its extension with silicone oil". In: *Scientific Reports* 11.1 (Sept. 2021), p. 19090. ISSN: 2045-2322. DOI: `10.1038/s41598-021-98694-2`.

[131] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. "GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 49–59. DOI: `10.1109/CVPR.2018.00013`.

[132] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. "Position Based Dynamics". In: *Journal of Visual Communication and Image Representation* 18 (Apr. 2007), pp. 109–118. DOI: `10.1016/j.jvcir.2007.01.005`.

[133] Yashraj S. Narang, Balakumar Sundaralingam, Karl Van Wyk, Arsalan Mousavian, and Dieter Fox. "Interpreting and Predicting Tactile Signals for the SynTouch BioTac". In: *The International Journal of Robotics Research* (2021).

[134] Alejandro Newell, Kaiyu Yang, and Jia Deng. "Stacked Hourglass Networks for Human Pose Estimation". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Cham: Springer International Publishing, 2016, pp. 483–499. ISBN: 978-3-319-46484-8.

[135] I. Newton. *Philosophiae naturalis principia mathematica*. Jussu Societatis Regiae ac Typis Josephi Streater. Prostat Venales apud Sam. Smith ad insigna Principis Walliae in Coemiterio D. Pauli, aliosq, nonnullos Bibliopolas, 1687.

[136] Baoqing Nie, Ruya Li, James Brandt, and Tingrui Pan. "Iontronic microdroplet array for flexible ultrasensitive tactile sensing". In: *Lab on a chip* 14 (Jan. 2014). DOI: `10.1039/c3lc50994j`.

[137] Matthias Nießner, Charles T. Loop, and Günther Greiner. "Efficient Evaluation of Semi-Smooth Creases in Catmull-Clark Subdivision Surfaces". In: *Eurographics*. 2012.

[138] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN: 9780387400655.

[139] Takehiko Ohkawa, Kun He, Fadime Sener, Tomas Hodan, Luan Tran, and Cem Keskin. "AssemblyHands: Towards Egocentric Activity Understanding via 3D Hand Pose Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 12999–13008.

[140] Heinrich Wilhelm Olbers. *Göttingische Anzeigen von gelehrten Sachen*. Göttingen, Germany: Königliche Gesellschaft der Wissenschaften, Jan. 1802.

[141] Edwin Olson. "AprilTag: A robust and flexible visual fiducial system". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[142] OpenAI et al. "Learning Dexterous In-Hand Manipulation". In: *The International Journal of Robotics Research* (Aug. 2018).

[143] Paschalis Panteleris, Iasonas Oikonomidis, and Antonis A. Argyros. "Using a Single RGB Frame for Real Time 3D Hand Pose Estimation in the Wild". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017), pp. 436–445.

[144] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. "Probabilistic Movement Primitives". In: *Proc. Advances in Neural Information Processing Systems.* Jan. 2013.

[145] Joon Park, Jin-Hong Kim, Byong Hyon, Jun-Kyuk Choi, and In-Song Jung. "Development of Sinusoidal BLDC Drive with Hall Sensors". In: *International Conference on Industrial Application Engineering.* Jan. 2016, pp. 490–493. DOI: `10.12792/iciae2016.089`.

[146] Adam Paszke et al. "PyTorch: an imperative style, high-performance deep learning library". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems.* Red Hook, NY, USA: Curran Associates Inc., 2019.

[147] Esteban Peña-Pitarch, Neus Falguera, and Jingzhou Yang. "Virtual human hand: model and kinematics". In: *Computer methods in biomechanics and biomedical engineering* 17 (Aug. 2012). DOI: `10.1080/10255842.2012.702864`.

[148] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. "DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills". In: *ACM Trans. Graph.* 37.4 (July 2018), 143:1–143:14. ISSN: 0730-0301. DOI: `10.1145/3197517.3201311`.

[149] Bui Tuong Phong. "Illumination for computer generated pictures". In: *Seminal Graphics: Pioneering Efforts That Shaped the Field, Volume 1.* New York, NY, USA: Association for Computing Machinery, 1998, pp. 95–101. ISBN: 158113052X. DOI: `10.1145/280811.280980`.

[150] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C. Lin. "Scalable differentiable physics for learning and control". In: *Proceedings of the 37th International Conference on Machine Learning.* ICML'20. 2020.

[151] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations". In: *Proceedings of Robotics: Science and Systems (RSS).* 2018.

[152] Dominic Rath et al. *Open On-Chip Debugger.* Diploma Thesis, University of Applied Sciences Augsburg, Department of Computer Science.

[153] Robert Reams. "Hadamard inverses, square roots and products of almost semidefinite matrices". In: *Linear Algebra and its Applications* 288 (1999), pp. 35–43. ISSN: 0024-3795. DOI: `10.1016/S0024-3795(98)10162-3`.

[154] Eugen Richter. “Hand Pose Reconstruction using a Three-Camera Stereo Vision System”. MA thesis. University of Hamburg, Aug. 2011.

[155] Robert Rohling. “Optimized fingertip mapping and hand modeling for telemanipulation”. MA thesis. McGill University, 1993.

[156] Francisco Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. “Speeded Up Detection of Squared Fiducial Markers”. In: *Image and Vision Computing* 76 (June 2018). DOI: 10.1016/j.imavis.2018.05.004.

[157] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015.* Ed. by Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4.

[158] G. van Rossum. *Python tutorial.* Tech. rep. CS-R9526. Amsterdam: Centrum voor Wiskunde en Informatica (CWI), May 1995.

[159] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International Conference on Computer Vision.* 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[160] Philipp Ruppel, N. Hendrich, Sebastian Starke, and Jianwei Zhang. “Cost Functions to Specify Full-Body Motion and Multi-Goal Manipulation Tasks”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA).* May 2018. DOI: 10.1109/ICRA.2018.8460799.

[161] Philipp Ruppel, Norman Hendrich, and Jianwei Zhang. “Low-cost multi-view pose tracking using active markers”. In: *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS).* 2019, pp. 261–268. DOI: 10.1109/ICPHYS.2019.8780297.

[162] Philipp Ruppel, Yannick Jonetzko, Michael Görner, Norman Hendrich, and Jianwei Zhang. “Simulation of the SynTouch BioTac Sensor”. In: *Intelligent Autonomous Systems 15.* Ed. by Marcus Strand, Rüdiger Dillmann, Emanuele Menegatti, and Stefano Ghidoni. Cham: Springer International Publishing, 2019, pp. 374–387. ISBN: 978-3-030-01370-7.

[163] Katharine Sanderson. “Carbon nanotubes: the new asbestos?” In: *Nature* (May 2008). DOI: 10.1038/news.2008.845.

[164] Mithileysh Sathiyanarayanan and Sharanya Rajan. “MYO Armband for physiotherapy healthcare: A case study using gesture recognition application”. In: *2016 8th International Conference on Communication Systems and Networks (COMSNETS).* 2016, pp. 1–6. DOI: 10.1109/COMSNETS.2016.7439933.

[165] Robert Schaback. “Die Beiträge von Carl-Friedrich Gauß zur Numerischen Mathematik”. In: *Mitteilungen (Gauss-Gesellschaft e. V., Göttingen)* (2014).

[166] Günter Schreiber, Andreas Stemmer, and Rainer Bischoff. “The Fast Research Interface for the KUKA Lightweight Robot”. In: *IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications (ICRA 2010).* May 2010.

[167] John Schulman et al. “Motion planning with sequential convex optimization and convex collision checking”. In: *The International Journal of Robotics Research* 33 (Aug. 2014), pp. 1251–1270. DOI: 10.1177/0278364914528132.

[168] Philipp Ludwig Seidel. *Über ein Verfahren, die Gleichungen, auf welche die Methode der kleinsten Quadrate führt, sowie lineäre Gleichungen überhaupt, durch successive Annäherung aufzulösen.* München: Verlag der k. Akademie, 1873.

[169] David Shah, Eddie Hung, C. Wolf, Serge Bazanski, Dan Gisselquist, and Miodrag Milanovic. “Yosys+nextpnr: An Open Source Framework from Verilog to Bitstream for Commercial FPGAs”. In: *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM).* Apr. 2019, pp. 1–4. DOI: 10.1109/FCCM.2019.00010.

[170] David F. Shanno. “Who Invented the Interior-Point Method?” In: *Documenta Mathematica, Extra Volume: Optimization Stories* (2012).

[171] Michael Sherman, Ajay Seth, and Scott Delp. “Simbody: Multibody dynamics for biomedical research”. In: *Procedia IUTAM* 2 (Dec. 2011). IUTAM Symposium on Human Body Dynamics, pp. 241–261. DOI: 10.1016/j.piutam.2011.04.023.

[172] Jun Shintake, Egor Piskarev, Seung Hee Jeong, and Dario Floreano. “Ultrastretchable Strain Sensors Using Carbon Black-Filled Elastomer Composites and Comparison of Capacitive Versus Resistive Sensors”. In: *Advanced Materials Technologies* 3.3 (2018), p. 1700284. DOI: 10.1002/admt.201700284.

[173] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. “Hand Keypoint Detection in Single Images using Multiview Bootstrapping”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2017.

[174] Antun Skuric, Hasan Sinan Bank, Richard Unger, Owen Williams, and David González-Reyes. “SimpleFOC: A Field Oriented Control (FOC) Library for Controlling Brushless Direct Current (BLDC) and Stepper Motors”. In: *Journal of Open Source Software* 7.74 (2022), p. 4232. DOI: 10.21105/joss.04232.

[175] Bart Smets, Jim Portegies, Erik Bekkers, and Remco Duits. “PDE-based Group Equivariant Convolutional Neural Networks”. In: *Journal of Mathematical Imaging and Vision* (2022). DOI: 10.1007/s10851-022-01114-x.

[176] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37.* ICML’15. Lille, France, 2015, pp. 2256–2265.

[177] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. “Interactive Markerless Articulated Hand Motion Tracking using RGB and Depth Data”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* Dec. 2013.

[178] Huanbo Sun, Katherine J. Kuchenbecker, and Georg Martius. "A soft thumb-sized vision-based sensor with accurate all-round force perception". In: *Nature Machine Intelligence* 4 (Feb. 2022).

[179] Balakumar Sundaralingam et al. "CuRobo: Parallelized Collision-Free Robot Motion Generation". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 8112–8119. DOI: 10.1109/ICRA48891.2023.10160765.

[180] Subramanian Sundaram, Petr Kellnhofer, Yunzhu Li, Jun-Yan Zhu, A. Torralba, and W. Matusik. "Learning the signatures of the human grasp using a scalable tactile glove". In: *Nature* 569 (2019), pp. 698–702.

[181] J.R. Taylor. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. Series of books in physics. University Science Books, 1982. ISBN: 9780935702101.

[182] Manu Mathew Thomas, Gabor Liktor, Christoph Peters, Sungye Kim, Karthik Vaidyanathan, and Angus G. Forbes. "Temporally Stable Real-Time Joint Neural Denoising and Supersampling". In: *Proc. ACM Comput. Graph. Interact. Tech.* 5.3 (July 2022). DOI: 10.1145/3543870.

[183] Carl Thrasher, Zachary Farrell, Nicholas Morris, C.L. Willey, and Christopher Tabor. "Mechanoresponsive Polymerized Liquid Metal Networks". In: *Advanced Materials* 31 (Aug. 2019), p. 1903864. DOI: 10.1002/adma.201903864.

[184] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[185] Marc Toussaint, Jung-Su Ha, and Danny Driess. "Describing Physics For Physical Reasoning: Force-Based Sequential Manipulation Planning". In: *IEEE Robotics and Automation Letters* PP (July 2020), pp. 1–1. DOI: 10.1109/LRA.2020.3010462.

[186] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. "Bundle Adjustment - A Modern Synthesis". In: *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*. ICCV '99. Berlin, Heidelberg: Springer-Verlag, 1999, pp. 298–372. ISBN: 3540679731.

[187] Yuyang Tu, Junnan Jiang, Shuang Li, Norman Hendrich, Miao Li, and Jianwei Zhang. "PoseFusion: Robust Object-in-Hand Pose Estimation with SelectLSTM". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 6839–6846. DOI: 10.1109/IROS55552.2023.10341688.

[188] S. Umeyama. "Least-squares estimation of transformation parameters between two point patterns". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.4 (1991), pp. 376–380. DOI: 10.1109/34.88573.

[189] Feng Wang, Huaping Liu, Fuchun Sun, and Haihong Pan. “Fabric recognition using zero-shot learning”. In: *Tsinghua Science and Technology* 24 (Dec. 2019), pp. 645–653. DOI: 10.26599/TST.2018.9010095.

[190] John Wang and Edwin Olson. “AprilTag 2: Efficient and robust fiducial detection”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2016.

[191] Robert Y. Wang and Jovan Popović. “Real-time hand-tracking with a color glove”. In: *ACM Trans. Graph.* 28.3 (July 2009). ISSN: 0730-0301. DOI: 10.1145/1531326.1531369.

[192] Weichen Wang et al. “Strain-insensitive intrinsically stretchable transistors and circuits”. In: *Nature Electronics* 4 (Feb. 2021), pp. 1–8. DOI: 10.1038/s41928-020-00525-1.

[193] Dehao Wei and Huazhe Xu. “A wearable robotic hand for hand-over-hand imitation learning”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2024.

[194] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. “Convolutional pose machines”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[195] Patrick M. Wensing and Shai Revzen. “Control of motion and compliance: Template models for control”. In: *Bioinspired Legged Locomotion*. Nov. 2017, pp. 240–266. ISBN: 9780128037669. DOI: 10.1016/B978-0-12-803766-9.00006-3.

[196] Stephanie J. Woodman, Dylan S. Shah, Melanie Landesberg, Anjali Agrawala, and Rebecca Kramer-Bottiglio. “Stretchable Arduinos embedded in soft robots”. In: *Science Robotics* 9.94 (2024), eadn6844. DOI: 10.1126/scirobotics.adn6844.

[197] Zhou Xian, Bo Zhu, Zhenjia Xu, Hsiao-Yu Fish Tung, Antonio Torralba, Katerina Fragkiadaki, and Chuang Gan. “FluidLab: A Differentiable Environment for Benchmarking Complex Fluid Manipulation”. In: *International Conference on Learning Representations*. 2023.

[198] Kouichi Yamaguchi, Kenji Sakamoto, Toshio Akabane, and Yoshiji Fujimoto. “A neural network for speaker-independent isolated word recognition”. In: *ICSLP*. 1990.

[199] Fengyu Yang, Chenyang Ma, Jiacheng Zhang, Jing Zhu, Wenzhen Yuan, and Andrew Owens. “Touch and Go: Learning from Human-Collected Vision and Touch”. In: *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2022.

[200] Kelin Yu, Yunhai Han, Qixian Wang, Vaibhav Saxena, Danfei Xu, and Ye Zhao. “MimicTouch: Leveraging Multi-modal Human Tactile Demonstrations for Contact-rich Manipulation”. In: *8th Annual Conference on Robot Learning*. 2024.

[201] Wenzhen Yuan, Siyuan Dong, and Edward Adelson. "GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force". In: *Sensors* 17 (Nov. 2017), p. 2762. DOI: 10.3390/s17122762.

[202] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. "3D Diffusion Policy: Generalizable Visuomotor Policy Learning via Simple 3D Representations". In: *Proceedings of Robotics: Science and Systems (RSS).* 2024.

[203] Huiyang Zhang, Anubha Kalra, Andrew Lowe, Yang Yu, and Gautam Anand. "A Hydrogel-Based Electronic Skin for Touch Detection Using Electrical Impedance Tomography". In: *Sensors* 23.3 (2023). ISSN: 1424-8220. DOI: 10.3390/s23031571.

[204] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. "Adding Conditional Control to Text-to-Image Diffusion Models". In: *IEEE International Conference on Computer Vision (ICCV).* 2023.

[205] Yang Zhang and Chris Harrison. "Tomo: Wearable, Low-Cost Electrical Impedance Tomography for Hand Gesture Recognition". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (2015).

[206] Z. Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334. DOI: 10.1109/34.888718.

[207] Christian Zimmermann and Thomas Brox. "Learning to Estimate 3D Hand Pose from Single RGB Images". In: *IEEE International Conference on Computer Vision (ICCV).* 2017.

[208] Matt Zucker et al. "CHOMP: Covariant Hamiltonian optimization for motion planning". In: *The International Journal of Robotics Research* 32 (Aug. 2013), pp. 1164–1193. DOI: 10.1177/0278364913488805.

[209] Runze Zuo, Zhanfeng Zhou, Binbin Ying, and Xinyu Liu. "A Soft Robotic Gripper with Anti-Freezing Ionic Hydrogel-Based Sensors for Learning-Based Object Recognition". In: *ICRA 2021, accepted.* 2021.

Online resources

[210] Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. *Continuous control with deep reinforcement learning.* Accessed 2024-12. URL: https://arxiv.org/abs/1509.02971.

[211] OpenAI et al. *Solving Rubik's Cube with a Robot Hand.* Accessed 2024-12. Oct. 2019. URL: https://arxiv.org/abs/1910.07113.

[212] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning.* Accessed 2024-12. 2016–2023. URL: http://pybullet.org.

[213] *CppAD.* Accessed 2024-08. URL: https://github.com/coin-or/CppAD.

[214] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. *Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation.* Version 0.10.5. Accessed 2024-12. 2021. URL: http://github.com/google/brax.

[215] Martin Abadi, Ashish Agarwal, Paul Barham, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* Accessed 2024-12. 2015. URL: http://tensorflow.org/.

[216] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. *Ceres Solver.* Version 2.2. Accessed 2024-12. Oct. 2023. URL: https://github.com/ceres-solver/ceres-solver.

[217] *Open Dynamics Engine.* Accessed 2024-11. URL: http://www.ode.org/.

[218] Akshay Srinivasan1 and Emanuel Todorov. *Graphical Newton.* Accessed 2024-12. URL: https://homes.cs.washington.edu/~todorov/papers/SrinivasanICRA17.pdf.

[219] *A faster quaternion-vector multiplication.* Accessed 2024-09. URL: https://blog.molecular-matters.com/2013/05/24/a-faster-quaternion-vector-multiplication/.

[220] *Tekscan.* Accessed 4-April-2022. URL: https://www.tekscan.com/.

[221] Matthias Zenker and Risto Kõiva and Robert Haschke. *Shadow Hand Tactile Fingertip.* Accessed 4-April-2022. URL: https://ni.www.techfak.uni-bielefeld.de/tactile/fingertip.

[222] PPS Medical Tactile Inc. *TactileGlove - Hand Pressure And Force Measurement.* Accessed 05-April-2022. URL: https://pressureprofile.com/body-pressure-mapping/tactile-glove.

[223] *How to Work With Conductive Fabric.* Accessed 2024-12. URL: https://www.instructables.com/How-to-Work-With-Conductive-Fabric/.

[224] *DIY Wearable Technology Documentation.* Accessed 2024-12. URL: https://www.kobakant.at/DIY/.

[225] Blender Online Community. *Blender - a 3D modelling and rendering package.* Accessed 2024-12. Blender Foundation. 2024. URL: http://www.blender.org.

[226] *OpenSCAD.* Accessed 2024-12. URL: https://openscad.org/.

[227] *Inkscape.* Accessed 2024-12. URL: https://inkscape.org/.

[228] *inkscape-silhouette.* Accessed 2024-12. URL: https://github.com/fablabnbg/inkscape-silhouette.

[229] Bonnie Baker. *How to Design Stable Transimpedance Amplifiers for Automotive and Medical Systems.* Accessed 2024-12. 2017. URL: https://www.digikey.com/en/articles/how-to-design-stable-transimpedance-amplifiers-automotive-medical-systems.

[230] Paul Semig. *Transimpedance Amplifier Circuit.* Accessed 2024-12. 2018. URL: https://www.ti.com/lit/an/sboa268b/sboa268b.pdf.

[231] *Geometry Nodes.* Accessed 2024-12. URL: https://docs.blender.org/manual/en/latest/modeling/geometry%5C_nodes/index.html.

[232] Thomas Skjølberg and others. *3d-bin-container-packing.* Accessed 2024-12. URL: https://github.com/skjolber/3d-bin-container-packing.

[233] Tamas Meszaros et al. *Libnest2D, a library and framework for the 2D bin packaging problem.* Accessed 2024-12. URL: https://github.com/tamasmeszaros/libnest2d.

[234] Lattice Semiconductor Corporation. *Simple Sigma-Delta ADC - Reference Design.* Accessed 2024-09. URL: https://www.latticesemi.com/view_document?document_id=35762.

[235] *PicoRV32.* Accessed 2024-12. URL: https://github.com/YosysHQ/picorv32.

[236] *UM10204 - I2C-bus specification and user manual.* Accessed 2024-12. 2021. URL: https://web.archive.org/web/20241205234533/https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[237] *Yosys Open SYnthesis Suite.* Accessed 2024-12. URL: https://yosyshq.net/yosys/.

[238] *Project Trellis.* Accessed 2024-12. URL: https://github.com/YosysHQ/prjtrellis.

[239] *Project IceStorm, Open source Verilog-toBitstream flow for iCE40 FPGAs.* Accessed 2024-12. URL: https://github.com/YosysHQ/icestorm.

[240] Stephen Williams. *The ICARUS Verilog Compilation System.* Accessed 2024-12. URL: https://github.com/steveicarus/iverilog.

[241] Wilson Snyder et al. *Verilator.* Accessed 2024-12. URL: https://www.veripool.org/verilator/.

[242] *MakeHuman.* Accessed 2024-12. URL: https://static.makehumancommunity.org/.

[243] Greg Davill et al. *ecpprog.* Accessed 2024-12. URL: https://github.com/gregdavill/ecpprog.

[244] *PhaseSpace Motion Capture - Infinite Possibilities.* Accessed 2024-11. URL: https://www.phasespace.com/.

[245] *CyberGlove Systems LLC.* Accessed 11-2024. URL: https://www.cyberglovesystems.com/.

[246] Zhicheng Wang and Genzhi Ye. *MediaPipe KNIFT: Template-based feature matching.* Accessed 2024-12. 2020. URL: https://developers.googleblog.com/en/mediapipe-knift-template-based-feature-matching/.

[247] *Quality of Life Grand Challenge | Kitchen Capture.* Accessed 2024-11. URL: http://kitchen.cs.cmu.edu/.

[248] *Performance Capture Software | What Is Performance Capture? | Autodesk.* Accessed 2024-11. URL: https://www.autodesk.com/solutions/performance-capture.

[249] *Virtual production: performance capture for everyone - Unreal Engine.* Accessed 2024-11. URL: https://www.unrealengine.com/en-US/blog/virtual-production-performance-capture-for-everyone.

[250] *Robot Operating System.* Accessed 2024-12. URL: https://www.ros.org/.

[251] *libusb.* Accessed 2024-08. URL: https://libusb.info/.

[252] *ft60x-rs.* Accessed 2024-11. URL: https://github.com/apertus-open-source-cinema/ft60x-rs.

[253] *1/2.5-Inch 5 Mp CMOS Digital Image Sensor.* Accessed 11-2024. URL: https://www.onsemi.com/download/data-sheet/pdf/mt9p001-d.pdf.

[254] *Waveshare Raspberry Pi Camera (B), OV5647 Adjustable-focus 5 Megapixel FFC.* Accessed 11-2024. URL: https://eckstein-shop.de/WaveshareRaspberryPiCameraB2COV5647Adjustable-focus5MegapixelFFC.

[255] *OdSeven Camera HD OV5647 5Mpx - für Raspberry Pi.* Accessed 11-2024. URL: https://botland.de/kameras-fur-raspberry-pi/5619-odseven-camera-hd-ov5647-5mpx-fur-raspberry-pi-5904422333850.html.

[256] *OEM H.264 YUV MJPEG IMX577 IMX477 1/2,3 Zoll 12MP CMOS-Sensor Autofokus Fixed focus RGB Mini-USB-UVC-Kamera modul 4K.* Accessed 11-2024. URL: https://german.alibaba.com/product-detail/OEM-H-264-YUV-MJPEG-IMX577-1601178243127.html.

[257] *12MP Sony IMX577 Sensor 4K 30FPS 1080P 120FPS USB Camera Small Case with M12 25mm lens.* Accessed 11-2024. URL: https://www.elpcctv.com/12mp-sony-imx577-sensor-4k-30fps-1080p-120fps-usb-camera-small-case-with-m12-25mm-lens-p-512.html.

[258] *RASP CAM HQ Raspberry Pi - Kamera, 12MP, 75°, C-/CS-Fassung.* Accessed 11-2024. URL: https://www.reichelt.de/de/de/raspberry-pi-kamera-12mp-75-c-cs-fassung-rasp-cam-hq-p276919.html.

[259] *low light/ IR: Comparing OV5647 and Aptina MI5100.* Accessed 12-2022. URL: https://web.archive.org/web/20221210090947/https://forums.raspberrypi.com/viewtopic.php?t=152476#p999454.

[260] *Protocols, PHYs and the MIPI Alliance IPR Terms.* Accessed 11-2024. URL: https://www.mipi.org/blog/protocols-phys-and-the-mipi-alliance-ipr-terms.

[261] NVIDIA Corporation Mark J. Kilgard. *EXT_texture_shared_exponent - Khronos Registry*. Accessed 2024-12. URL: https://registry.khronos.org/OpenGL/extensions/EXT/EXT_texture_shared_exponent.txt.

[262] *Camera Calibration and 3D Reconstruction*. Accessed 2024-12. URL: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html.

[263] Andrew Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. Accessed 2024-12. Apr. 2017. URL: https://arxiv.org/abs/1704.04861.

[264] *Prusa CORE One*. Accessed 2024-11. URL: https://www.prusa3d.com/de/produkt/prusa-core-one/.

[265] *UltiMaker S3*. Accessed 2024-11. URL: https://ultimaker.com/de/3d-printers/s-series/ultimaker-s3/.

[266] *Small CNC machines (desktop) | isel Germany AG*. Accessed 2024-11. URL: https://www.isel.com/en/products/systems/cnc-machines/cnc-milling-machines/cnc-desktop-machines.html.

[267] *Three hundred 3D printers in one room: A quick look to our printing farm*. Accessed 2024-11. URL: https://blog.prusa3d.com/a-quick-look-to-our-printing-farm_7474/.

[268] *UR5 robot description*. Accessed 2024-12. URL: http://wiki.ros.org/ur5%5C_description.

[269] *Universal Robots e-Series User Manual*. Accessed 2024-12. 2009-2020. URL: https://www.universal-robots.com/download/manuals-e-seriesur20ur30/user/ur10e/59/user-manual-ur10e-e-series-sw-59-english-international-en/.

[270] *Unchained Robotics MalocherBot Base Unit*. Accessed 2024-11. URL: https://unchainedrobotics.de/en/products/turnkey-solutions/unchained-robotics-malocherbot-base.

[271] *EasyRobotics Produkte - kollaborative Roboterzellen*. Accessed 2024-11. URL: https://easyrobotics.biz/de/products/.

[272] *Dexterous Hand Series - Shadow Robot*. Accessed 2024-11. URL: https://www.shadowrobot.com/dexterous-hand-series/.

[273] *Intel® RealSense™ Technology*. Accessed 2024-11. URL: https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html.

[274] *Optimus - Gen 2 | Tesla*. Accessed 2024-11. URL: https://www.youtube.com/watch?v=cpraXaw7dyc.

[275] *Desktop Collaborative Robot - MTI-MG400*. Accessed 2024-11. URL: https://www.mtixtl.com/MTIMG400.aspx.

[276] *JR3303 DeskTop xyz - DispenseRite*. Accessed 2024-11. URL: https://www.dispenserite.ca/janome-desktop-xyz.html.

[277] *Inside Axis 4, 5 & 6 of KUKA KR5 Robot.* Accessed 2024-11. URL: https://www.youtube.com/watch?v=iRKDfknqtbc.

[278] *Hybrid Stepper Motor With Electric Gripper.* Accessed 2024-11. URL: https://nmbtc.com/product-category/hybrid-gripper/.

[279] *The URScript Programming Language for e-Series.* Accessed 2024-12. URL: https://www.universal-robots.com/download/manuals-e-seriesur20ur30/script/script-manual-e-series-sw-511/.

[280] *Intuitive | Maker of Da Vinci & Ion Robotic Systems.* Accessed 11-2024. URL: https://www.intuitive.com/en-us.

[281] Gaël Guennebaud et al. *Eigen v3.* Accessed 2024-12. 2010. URL: http://eigen.tuxfamily.org.

[282] François Chollet et al. *Keras.* Accessed 2024-11. 2015. URL: https://keras.io.

[283] *Diffusion Policy.* Accessed 2024-12. URL: https://github.com/real-stanford/diffusion_policy.

[284] Patrick von Platen et al. *Diffusers: State-of-the-art diffusion models.* Accessed 2024-12. 2022. URL: https://github.com/huggingface/diffusers.

[285] *Introduction to Blueprints Visual Scripting in Unreal Engine | Unreal Engine 5.5 Documentation | Epic Developer Community.* Accessed 2024-12. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/introduction-to-blueprints-visual-scripting-in-unreal-engine.

[286] *ComfyUI.* Accessed 2024-12. URL: https://github.com/comfyanonymous/ComfyUI.

A. List of figures

B. List of tables

C. Notation

Symbols

P	Position
V	Velocity
A	Acceleration
T	Tactile measurement
R	Random number
b_n	Denoising base
f_n	Denoising factor
K	Robot forward kinematics
i	Index
j	Index
k	Index
l	Index
t	Time
dt	Time step
U	Upper limit
L	Lower limit
δ	Trust region
N	Normal vector
r	Radius
L_T	Task loss
d_S	Signed distance
F_C	Contact force
D_M	Material stuffness
C_S	Contact smoothness
N_P	Policy network
O	Observations
M_J	Joint commands
n	Count
m	Count
d	Distortion coefficient
t_h	Control horizon
t_f	Fixed keyframes
N_{MT}	Neural network
m_V	Occlusion mask
P_m	Marker position

P_c	Camera position
Q_C	Camera orientation
J_D	Dynamics Jacobian
J	Projection
e_R	Reprojection error
δ	Huber loss delta
$P_{C[A,B]}$	Contact point candidate
P_{CA}	Contact point candidate A
P_{CB}	Contact point candidate B
$V_{C[A,B]}$	Contact point velocity
V_{CA}	Contact point velocity A
V_{CB}	Contact point velocity B
P_R	Contact response point
M_P	Physics simulation model
S_P	Physics simulation state
W_P	Policy network weights
L_P	Physical consistency loss
L_{SP}	Shape loss
L_{DP}	Distance loss
L_{FP}	Friction loss
L_{LP}	Slippage loss
L_{CP}	Intersection loss
L_R	Reprojection loss
L_V	Velocity loss
L_A	Acceleration loss
L_S	Shape loss
C_O	Consistency constraint set
c_O	Consistency constraint element
L_H	Hand pose regularizer
L_Q	Position loss
σ	Standard deviation
μ	Average
w_L	Loss weights
W_R	Recurrent weight matrix
W_I	Input weight matrix
P_A	Probability density function of initial states
S_A	Initial simulation state
$N_{[A,B]}$	Boundary plane normal
N_A	Boundary plane normal A
N_B	Boundary plane normal B
$d_{[A,B]}$	Boundary plane distance
d_A	Boundary plane distance A
d_B	Boundary plane distance B
R	Ramp function (ReLU)
n_F	Friction cone plane count

d_{AB}	Collision depth
$n_{P[A,B]}$	Boundary plane counts
n_{PA}	Boundary plane count of shape A
n_{PB}	Boundary plane count of shape B
n_{SP}	Number of shape pairs
N_F	Friction cone boundary plane normal
d_F	Friction cone boundary plane distance
n_C	Contact normal
H	Recurrent activations
f_l	Linearization function
f_g	Forward gradient function
f_r	Reverse gradient function
Q_R	Quaternion
V_R	Rotation vector
V_X	Rotation axis
q	Index

Subscripts

t	Time
i	Index
j	Index
k	Index
l	Index
q	Index
r	Index
s	Index
d	Spatial dimension
I	Input
O	Output
P	Prediction
Z	Normal distribution
U	Uniform distribution
J	Joint
C	Cartesian
U	Upper limit
n	Count
m	Count
H	Hand
O	Object
O	Observations

D. Abbreviations

3D	Three-dimensional
6D	Six-dimensional
ADC	Analog-to-digital converter
AI	Artificial intelligence
BLDC	Brushless direct current
CAD	Computer-aided design
CAN	Controller Area Network
CAS	Chemical Abstracts Service
CAS	Computer algebra system
CNC	Computer numerical control
CPU	Central processing unit
CSG	Constructive solid geometry
DDS	Direct digital synthesis
DMP	Dynamic Motion Primitive
DMS	Dimethylsiloxane
EPA	Expanding Polytope Algorithm
ESD	Electro-static discharge
FFF	Fused filament fabrication
FPGA	Field programmable gate array
GJK	Gilbert–Johnson–Keerthi distance algorithm
GPU	Graphics Processing Unit
HD	High Definition
IC	Integrated circuit
IMU	Inertial measurement unit
LED	Light-emitting diode
LVDS	Low-voltage differential signaling
OBJ	Wavefront object file
PDMS	Polydimethylsiloxane
PE	Polyethylene
PID	Proportional, integral, differental
PLA	Polylactic acid
POM	Polyoxymethylene
ProMP	Probabilistic Movement Primitives
PTFE	Polytetrafluoroethylene
RAM	Random-access memory
ReLU	Rectified linear unit
RGB	Red, green, blue

ROS	Robot Operating System
RC	Resistor–capacitor
RGB	Red, green and blue
RISC	Reduced Instruction Set Computer
ROS	Robot Operating System
SIMD	Single instruction, multiple data
SLA	Stereolithography
SPI	Serial Peripheral Interface
SVG	Scalable Vector Graphics
TPU	Thermoplastic polyurethane
TVS	Transient voltage suppressor
UART	Universal Asynchronous Receiver Transmitter
URDF	Unified Robot Description Format
USB	Universal Serial Bus
VLSI	Very large scale integration

E. Publications

[287] Philipp Ruppel and Jianwei Zhang. "Learning Object Manipulation with Dexterous Hand-Arm Systems from Human Demonstration". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5417–5424. DOI: 10.1109/IROS45743.2020.9340966.

[288] Niklas Fiedler, Philipp Ruppel, Yannick Jonetzko, Norman Hendrich, and Jianwei Zhang. "A Low-Cost Modular System of Customizable, Versatile, and Flexible Tactile Sensor Arrays". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 1771–1777. DOI: 10.1109/IROS51168.2021.9635851.

[289] Philipp Ruppel, Norman Hendrich, and Jianwei Zhang. "Direct Policy Optimization with Differentiable Physical Consistency for Dexterous Manipulation". In: *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2021, pp. 650–655. DOI: 10.1109/ROBIO54168.2021.9739435.

[290] Niklas Fiedler*, Philipp Ruppel*, Yannick Jonetzko, Norman Hendrich, and Jianwei Zhang. "Low-cost fabrication of flexible tactile sensor arrays". In: *HardwareX* 12 (2022), e00372. ISSN: 2468-0672. DOI: 10.1016/j.ohx.2022.e00372.

[291] Jianzhi Lyu, Alexander Maÿe, Michael Görner, Philipp Ruppel, Andreas K. Engel, and Jianwei Zhang. "Coordinating human-robot collaboration by EEG-based human intention prediction and vigilance control". In: *Frontiers in Neurorobotics* 16 (2022). ISSN: 1662-5218. DOI: 10.3389/fnbot.2022.1068274.

[292] Philipp Ruppel, Norman Hendrich, and Jianwei Zhang. "Elastic Tactile Sensor Skin on Double-Curved Surfaces for Robots and Wearables". In: *IEEE Access* 10 (2022), pp. 91103–91118. DOI: 10.1109/ACCESS.2022.3201824.

[293] Lin Cong*, Philipp Ruppel*, Yizhou Wang, Xiang Pan, Norman Hendrich, and Jianwei Zhang. "Efficient Human Motion Reconstruction from Monocular Videos with Physical Consistency Loss". In: *SIGGRAPH Asia 2023 Conference Papers*. SA '23. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9798400703157. DOI: 10.1145/3610548.3618169.

[294] Jianzhi Lyu, Philipp Ruppel, Norman Hendrich, Shuang Li, Michael Görner, and Jianwei Zhang. "Efficient and Collision-Free Human–Robot Collaboration Based on Intention and Trajectory Prediction". In: *IEEE Transactions on Cognitive and Developmental Systems* 15.4 (2023), pp. 1853–1863. DOI: 10.1109/TCDS.2022.3215093.

[295] Philipp Ruppel and Jianwei Zhang. “Efficent Gradient Propagation for Robot Control and Learning”. In: *Cognitive Computation and Systems*. Ed. by Fuchun Sun, Jianmin Li, Huaping Liu, and Zhongyi Chu. Singapore: Springer Nature Singapore, 2023, pp. 237–246.

[296] Philipp Ruppel and Jianwei Zhang. “Elastic Tactile Sensor Glove for Dexterous Teaching by Demonstration”. In: *Sensors* 24.6 (2024). ISSN: 1424-8220. DOI: `10.3390/s24061912`.

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Sofern im Zuge der Erstellung der vorliegenden Dissertationsschrift generative Künstliche Intelligenz (gKI) basierte elektronische Hilfsmittel verwendet wurden, versichere ich, dass meine eigene Leistung im Vordergrund stand und dass eine vollständige Dokumentation aller verwendeten Hilfsmittel gemäß der Guten wissenschaftlichen Praxis vorliegt. Ich trage die Verantwortung für eventuell durch die gKI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate

Ort, Datum

Unterschrift