nature machine intelligence

Article

https://doi.org/10.1038/s42256-025-01045-3

A process-centric manipulation taxonomy for the organization, classification and synthesis of tactile robot skills

Received: 16 June 2023

Accepted: 30 April 2025

Published online: 23 June 2025

Check for updates

Lars Johannsmeier[1], Samuel Schneider[1], Yanan Li[2], Etienne Burdet[3] & Sami Haddadin[1,4]

Despite decades of research in robotic manipulation, only a few autonomous manipulation skills are currently used. Traditional and machine-learning-based end-to-end solutions have shown substantial progress but still struggle to generate reliable manipulation skills for difficult processes like insertion or bending material. To facilitate the deployment and learning of tactile robot manipulation skills, we introduce here a taxonomy based on formal process specifications provided by experts, which assigns a suitable skill to a given process. We validated the inherent scalability of the taxonomy on 28 different skills from industrial application domains. The experimental results had success rates close to 100%, even under goal pose disturbances, with high performance attained by the skill models in terms of execution times and contact moments in partially known environments. The basic elements of the models are reusable and facilitate skill-learning to optimize control performance. Like established curricula for human trainees, this framework could provide a comprehensive platform that enables robots to acquire relevant manipulation skills and act as a catalyst to propel automation beyond its current capabilities.

Advances in fields such as robot hardware development[1,2], motion and interaction control[3], interaction policy design[4], vision, motion and task planning, learning[5] and human–robot interaction[6], have led to systematic approaches for implementing skilful and versatile physical robotic manipulation capabilities called 'manipulation skills'. Considering that many workplaces may be automated in the near future and that important steps have been made to increase robotic manipulation performance[7,8], manipulation skills have gained increasing interest from various application domains[9–12] involving manual work. Human workers can rely on systematic curricula that have been specifically developed for their profession. For example, in Germany, there are various well-established standard works on professional industrial training[13], such as those described in refs. 14–16. However, no such curriculum exists for robots when learning manipulation skills.

If robots are to work in such applications, they require sophisticated tactile capabilities to ensure their safe, reliable and efficient integration into human-centred processes. Such capabilities, which we refer to as robot skills, have been the subject of extensive research efforts, such as object–action complexes[17,18], combinations of hierarchical task structures with low-level motion and interaction controllers[19–21], a cognitive architecture that recognizes and imitates actions[22], or goal-directed action sequences that consist of motor primitives[23]. Other approaches are based on Riemannian manifolds[24] or geometric fabrics[25] or are learned end-to-end, such as refs. 26,27. Recently, vision–language–action models have gained increasing popularity as more relevant robot

[1]Chair of Robotics and Systems Intelligence, Munich Institute of Robotics and Machine Intelligence, Technical University Munich, Munich, Germany. [2]School of Engineering and Informatics, University of Sussex, Brighton, UK. [3]Department of Bioengineering, Imperial College of Science, Technology and Medicine, London, UK. [4]Department of Robotics, Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE. e-mail: lars.johannsmeier@tum.de; Sami.Haddadin@mbzuai.ac.ae

data have become available[4,28]. In this work, we focus on three important issues: the missing scalability of the solution skills, the integration of learning and task constraints such as safety and robustness, and the need for a clear mapping from process specification to skill implementation.

To address these problems, we propose a new taxonomy-based approach for manipulation skills that integrates control and learning on a basic level. Several taxonomies have been proposed for anthropomorphic robotic hands where different grasps are classified into a hierarchical structure[29,30]. However, taxonomies that classify robot skills remain rare. Ref. 31 introduced an assembly taxonomy that decomposes complex assembly tasks into simple skills that can be reused to reduce the programming time and overhead, whereas ref. 32 combined planning methods and compliant manipulation schemes to classify typical household tasks.

We consider our taxonomy as a first step towards developing a systematic curriculum for robotic manipulation, which would allow us to constructively scale to arbitrary skills with unmatched versatility. Additionally, we introduce a formalism for tactile skills that could seamlessly integrate well-understood manipulation processes and their constraints and connect to learning capabilities.

The proposed integration of process specifications, a taxonomy and a tactile skill framework can also be viewed as an end-to-end approach informed by requirements. In robotics, end-to-end frameworks typically map available sensory inputs to motor torque outputs using, for example, large neural networks to represent the policy[33,34]. Instead, we seek a mapping from a process description of physical manipulation to a parametric representation of a tactile skill. Both are constructed with a compatible formalism and, thus, can be framed as a joint learning problem. Compared to common sensor-to-torque approaches, the solution space is much smaller, such that it can be more efficiently sampled and learned. Although the approach introduced in this manuscript is not limited to a particular subset of robotic manipulation skills, we focus on well-established manufacturing processes as a highly relevant application domain with notable spillover potential. Also note that we performed our studies with a widely used state-of-the-art manipulator with seven degrees of freedom (DoF) and equipped with a standard rigid end effector. While highly specialized machines are more suited to individual skills and can execute them with higher performance than this manipulator, but our results should be compared with other general-purpose manipulators. Furthermore, all objects are grasped with form closure.

Results

Tactile skill

We first introduce the concept of a tactile skill as a computational policy–controller–learner complex that, together with a tactile platform, constitutes the system class of tactile robots. Figure 1 depicts the overall architecture of a tactile skill, which is composed of three fundamental components as described in its definition. A nomenclature explaining all symbols used in the following is presented in appendix 1 in the Supplementary Information.

Definition 1. (Tactile skill) A tactile skill integrates (1) a tactile policy, which encodes coordinated desired wrench and twist commands, and drives (2) a tactile controller that simultaneously regulates and tracks compliance and contact force by generating inputs to (3) a tactile platform that represents the low-level joint dynamics integrating the actual physical system and delivering the percept vector $\mathbf{\Omega}$ to measure the performance $\mathcal{Q}$ that is used to (4) learn the policy and control parameters $\boldsymbol{\theta}_\pi$ and $\boldsymbol{\theta}_c$, respectively.

Definition 2. (Tactile policy) A tactile policy $\boldsymbol{\pi}_d$ encodes and generates coordinated twist and wrench commands $\boldsymbol{\pi}_d = [\dot{\mathbf{x}}_d^T, \mathbf{f}_d^T]^T$ to drive the tactile controller based on the percept vector $\mathbf{\Omega}$ and policy parameters $\boldsymbol{\theta}_\pi$.

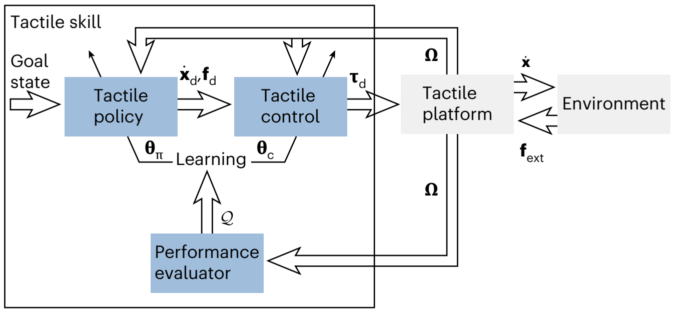

Fig. 1 | Tactile skill architecture. $\dot{\mathbf{x}}_d$ is the desired twist, $\mathbf{f}_d$ the desired wrench, $\boldsymbol{\tau}_d$ the desired joint torque, $\dot{\mathbf{x}}$ is the actual twist of the robot, $\mathbf{f}_{ext}$ the external wrench, $\mathbf{\Omega}$ the percept vector, $\mathcal{Q}$ the performance of the skill, $\boldsymbol{\theta}_\pi$ the policy parameters and $\boldsymbol{\theta}_c$ the controller parameters.

A particularly useful instance of a tactile policy $\boldsymbol{\pi}_d$ can be constructed using dynamic movement primitives[35]. Any suitable dynamical system with appropriate geometric properties is a potential candidate. A possible combined system with dynamic movement primitives for both motion and wrench is defined by

$$\begin{aligned}
\ddot{\mathbf{p}}_d &= \Theta_{\pi,p_1}\left[\Theta_{\pi,p_2}(\mathbf{p}_g - \mathbf{p}_d) - \dot{\mathbf{p}}_d\right] + \left(\boldsymbol{\gamma}^T(\alpha(t))I_{N\times 1}\right)^{-1}\Theta_{\pi,p_3}\boldsymbol{\gamma}(\alpha(t))\alpha(t),\\
\dot{\boldsymbol{\eta}}_d &= \Theta_{\pi,r_1}\left[\Theta_{\pi,r_2}\, 2\log^r(\mathbf{r}_g \otimes \bar{\mathbf{r}}_d) - \boldsymbol{\eta}_d\right]\\
&\quad + \left(\boldsymbol{\gamma}^T(\alpha(t))I_{N\times 1}\right)^{-1}\Theta_{\pi,r_3}\boldsymbol{\gamma}(\alpha(t))\alpha(t),\\
\dot{\mathbf{r}}_d &= \frac{1}{2}\boldsymbol{\eta}_d \otimes \mathbf{r}_d,\\
\ddot{\mathbf{f}}_d &= \Theta_{\pi,f_1}\left[\Theta_{\pi,f_2}(\mathbf{f}_g - \mathbf{f}_d) - \dot{\mathbf{f}}_d\right] + \left(\boldsymbol{\gamma}^T(\alpha(t))I_{N\times 1}\right)^{-1}\Theta_{\pi,f_3}\boldsymbol{\gamma}(\alpha(t))\alpha(t),\\
\dot{\alpha} &= -\boldsymbol{\theta}_{\pi,\alpha}\alpha,
\end{aligned} \tag{1}$$

where $\otimes$ denotes the quaternion product, $\mathbf{p}_g$ and $\mathbf{p}_d$ the goal and desired positions, $\mathbf{r}_g$ and $\mathbf{r}_d$ the goal and desired orientations in quaternion representation, $\boldsymbol{\eta}_d$ the desired angular velocity and $\bar{\mathbf{r}}$ the conjugated orientation quaternion. $\boldsymbol{\gamma}(\alpha(t))$ is the Gaussian basis function vector with N elements driven by the synchronizing joint phase variable α. The matrices $\Theta_{[]} = \mathrm{diag}\{\boldsymbol{\theta}_{[]}\}$ parameterize the dynamical system and the Gaussians. See appendix 2 in the Supplementary Information for more details.

Several more example policies are presented in appendix 7 in the Supplementary Information. These are described in the time domain for clarity, although in the actual system they would be encoded into a form like equation (1). Note that in our implementation the measured external wrench $\mathbf{f}_{ext}$ is used at the policy level to inform the policy-switching conditions.

Definition 3. (Tactile controller) A tactile controller is driven by the tactile policy $\boldsymbol{\pi}_d$ (the desired twist and wrench). It is parameterized by $\boldsymbol{\theta}_c$ and informed by the percept vector $\mathbf{\Omega}$. It generates effort-level commands to the physical system (the tactile platform) through a generalized interaction control framework that simultaneously regulates or tracks motion and force.

For simplicity, we use an impedance control architecture[36] combined with force application and regulation using tactile measurements $\mathbf{\Omega}_t \in \mathbf{\Omega}$. Overall, the closed-loop system can be written as

$$\mathbf{f}_{ext} = M_x(\mathbf{q})\ddot{\tilde{\mathbf{x}}} + D_d(M_x(\mathbf{q}), \boldsymbol{\theta}_{c,k}, \boldsymbol{\theta}_{c,d})\dot{\tilde{\mathbf{x}}} + K_d(\boldsymbol{\theta}_{c,k})\tilde{\mathbf{x}} + \mathbf{u}_f(\mathbf{\Omega}_t), \tag{2}$$

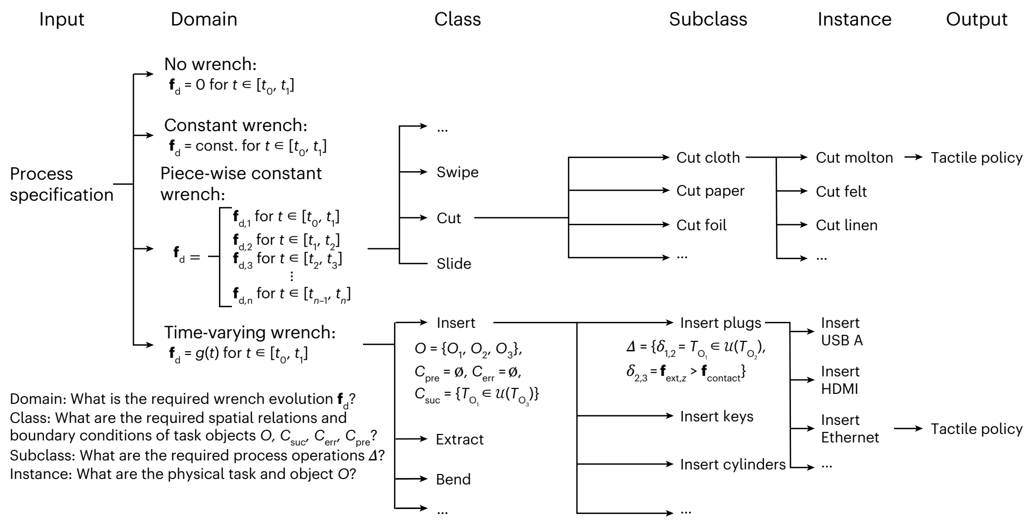
Input
Domain
Class
Subclass
Instance
Output
Process specification
No wrench: $\mathbf{f}_d = 0$ for $t \in [t_0, t_1]$
Constant wrench: $\mathbf{f}_d$ = const. for $t \in [t_0, t_1]$
Piece-wise constant wrench:
$\mathbf{f}_d = \mathbf{f}_{d,1}$ for $t \in [t_0, t_1]$, $\mathbf{f}_{d,2}$ for $t \in [t_1, t_2]$, $\mathbf{f}_{d,3}$ for $t \in [t_2, t_3]$, ⋮, $\mathbf{f}_{d,n}$ for $t \in [t_{n-1}, t_n]$
Time-varying wrench: $\mathbf{f}_d = g(t)$ for $t \in [t_0, t_1]$
...
Swipe
Cut
Slide
Insert
$O = \{O_1, O_2, O_3\}$, $C_{pre} = \emptyset$, $C_{err} = \emptyset$, $C_{suc} = \{T_{O_1} \in \mathcal{U}(T_{O_3})\}$
Extract
Bend
...
Cut cloth
Cut paper
Cut foil
...
Insert plugs
$\Delta = \{\delta_{1,2} = T_{O_1} \in \mathcal{U}(T_{O_2}), \delta_{2,3} = \mathbf{f}_{ext,z} > \mathbf{f}_{contact}\}$
Insert keys
Insert cylinders
...
Cut molton
Cut felt
Cut linen
...
Insert USB A
Insert HDMI
Insert Ethernet
...
Tactile policy
Tactile policy
Domain: What is the required wrench evolution $\mathbf{f}_d$?
Class: What are the required spatial relations and boundary conditions of task objects O, C_{suc}, C_{err}, C_{pre}?
Subclass: What are the required process operations Δ?
Instance: What are the physical task and object O?

Fig. 2 | Taxonomy of manipulation skills. This visualization shows the ranks of the taxonomy and how a given process specification may propagate through it to output a tactile skill.

where $\mathbf{f}_{ext}$ is the external wrench. $\tilde{\mathbf{x}} = \mathbf{x} - \mathfrak{f}_{q2a}(\mathbf{x}_d)$ denotes the motion error where $\mathfrak{f}_{q2a}$ is a transformation from quaternion to axis-angle representation. $M_x(\mathbf{q})$ denotes the Cartesian mass matrix[37], where $\mathbf{q}$ are the joint angles. K_d is the desired positive definite and diagonal stiffness matrix, which is parameterized by $\boldsymbol{\theta}_{c,k}$, where k indicates the parameters for the stiffness matrix. D_d is the desired positive definite damping matrix based on an appropriate damping design[38], where $\boldsymbol{\theta}_{c,d}$ are the damping factors. $\mathbf{u}_f(\boldsymbol{\Omega}_t)$ is a feedback term that yields an output based on a measured tactile quantity $\boldsymbol{\Omega}_t$. The specific richness of the tactile information $\boldsymbol{\Omega}_t$ depends on the specificity of the tactile platform. In the simplest case, $\mathbf{u}_f$ is a time-dependent feedforward wrench trajectory. A more complex option is a force controller[39], where $\boldsymbol{\Omega}_t = \mathbf{f}_{ext}$. With further advances in force and tactile sensing, future tactile platforms will leverage the measurement of pressure and shear stress distribution $\boldsymbol{\Omega}_t = {}^i\mathbf{v}_j$. For details, refer to sections 'Contact event chain' and 'Tactile-feedback level' in appendix 3 in the Supplementary Information, which describe the propagation of contacts into the internal robot structure and how the richness of $\boldsymbol{\Omega}_t$ determines the tactile feedback level of the given tactile platform.

Tactile platform. A tactile platform is a real-world physical realization that is close to the ideal robot dynamics and model. It incorporates the structure of tactile perception of external contacts through the contact event chain. It connects the tactile skill framework with the environment. A formal definition and technical specification details are provided in appendix 3 in the Supplementary Information.

Learning. Learning is an integral part of a tactile skill. Each parameter can evolve during learning. Specifically, the set of all parameters $\boldsymbol{\theta}_i = [\boldsymbol{\theta}_{c,i}^T, \boldsymbol{\theta}_{\pi,i}^T]^T$ at episode i is evaluated using a performance evaluator. The computed quality metric $\mathcal{Q}$ guides the selection of parameters for the next episode $\boldsymbol{\theta}_{i+1}$, that is, $\boldsymbol{\theta}_{i+1} = \mathcal{L}(\mathcal{Q}_i, \boldsymbol{\theta}_i)$, where $\mathcal{L}$ represents the learning method at hand and the performance evaluator measures the quality metric $\mathcal{Q}$ based on the convex combination:

$$\mathcal{Q}(\boldsymbol{\Omega}) = \sum_{l=1}^{n} \mathcal{Q}_l(\boldsymbol{\Omega})\,\omega_l, \quad \omega_l \leq 0, \quad \sum_{l=1}^{n} \omega_l = 1, \tag{3}$$

of performance measures $\{\mathcal{Q}_l(\boldsymbol{\Omega})\}$. The choice of weights ω_i is part of the cost function design and depends on the desired overall performance objectives. Details of the learning method used are provided in Methods.

Tactile skill representation in the state space. All the above elements of a tactile skill can be integrated into its unified state-space representation:

$$\begin{aligned}
\mathbf{y}(t) &:= \left[\mathbf{x}_d(t)^T, \dot{\mathbf{x}}_d(t)^T, \mathbf{f}_d(t)^T, \dot{\mathbf{f}}_d(t)^T, \mathbf{x}(t)^T, \dot{\mathbf{x}}(t)^T\right]^T, \\
\dot{\mathbf{y}}(t) &= \mathcal{A}(t, \mathbf{y}(t), \boldsymbol{\theta}_\pi, \boldsymbol{\theta}_c) + \mathcal{B}(t, \boldsymbol{\theta}_c)\mathbf{u}(t), \\
\mathbf{z}(t) &= \mathcal{C}(t, \boldsymbol{\theta}_c)\mathbf{y}(t) + \mathcal{D}(t, \boldsymbol{\theta}_c)\mathbf{u}(t),
\end{aligned} \tag{4}$$

where $\mathbf{y}(t)$ is the state vector, $\mathcal{A}$ the system matrix, $\mathcal{B}$ the input matrix, $\mathcal{C}$ the output matrix, $\mathcal{D}$ the feedthrough matrix, $\mathbf{u}(t)$ the control vector and $\mathbf{z}(t)$ the output vector. $\boldsymbol{\theta}_\pi$ and $\boldsymbol{\theta}_c$ are the parameter vectors of the tactile policy and the tactile controller, respectively, and $\mathbf{u}(t)$ is the total control vector:

$$\mathbf{u}(t) = \begin{bmatrix}
\left(\boldsymbol{\gamma}^T(\alpha(t)) I_{N\times 1}\right)^{-1} \Theta_{\pi,p_3} \boldsymbol{\gamma}(\alpha(t))\alpha(t) \\
\left(\boldsymbol{\gamma}^T(\alpha(t)) I_{N\times 1}\right)^{-1} \Theta_{\pi,r_3} \boldsymbol{\gamma}(\alpha(t))\alpha(t) \\
\left(\boldsymbol{\gamma}^T(\alpha(t)) I_{N\times 1}\right)^{-1} \Theta_{\pi,f_3} \boldsymbol{\gamma}(\alpha(t))\alpha(t) \\
\mathfrak{f}_{q2a}(\mathbf{x}_d(t)) \\
\mathfrak{f}_{q2a}(\dot{\mathbf{x}}_d(t)) \\
\mathbf{u}_f(\boldsymbol{\Omega}_t(t)) \\
\mathbf{f}_{ext}(t)
\end{bmatrix}. \tag{5}$$

Details of the system components are presented in appendix 2 in the Supplementary Information.

For clarity, we treat estimated and measured sensory quantities alike. We assumed the bandwidth to be consistently high enough and relevant errors to be comparably negligible, meaning that they are handled by lower-level loops so that the policy generator does not need to treat them explicitly. To our knowledge, current off-the-shelf technology complies with these assumptions. The stability and performance of the specific measurement and controller set-up depend on various factors, which go beyond the scope of this work.

Taxonomy

A tactile skill is the output of the taxonomy of manipulation skills (TMS), which encodes the skill selection process. The input to the taxonomy is the process specification. It is the interface used by process

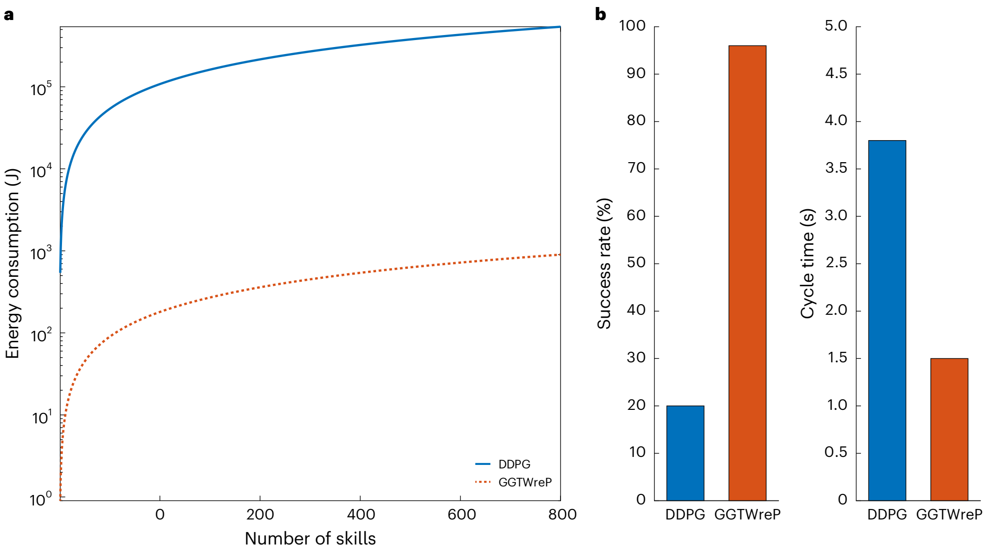

Fig. 3 | Energy demand comparison. a, Comparison of the energy required to learn a great number of skills. We compare the DDPG algorithm with the GGTWreP framework. **b**, Comparison of the final performance of successful episodes for both approaches averaged over several experiments.

experts, such as technicians, robot operators and shop-floor workers, to frame their process knowledge. The developed TMS connects the two domains of process-centric and robot-centric representations. Specifically, it allows a user to map a given process specification to a unique tactile skill using its underlying classification scheme. Figure 2 visually explains this mapping process. Each rank answers a specific question, as stated in the bottom left. This is done for the example of Ethernet plug insertion. The details are given below.

Process specification

A process p is specified by the process operations that are needed to achieve the process objective and the process requirements. There are four process states: initial state s_0, error state s_e, final state s_1 and policy state s_π. The policy state s_π contains a number of process operations. The process requirements are formalized by the transitions $\delta \in \Delta$ that connect the process operations and three boundary conditions ($\mathcal{C}_{\rm pre}$, $\mathcal{C}_{\rm err}$ and $\mathcal{C}_{\rm suc}$) that determine the switching between the top-level states:

- The precondition $\mathcal{C}_{\rm pre}(\mathcal{O}) = c_{1,\rm pre}(\mathcal{O}) \wedge \cdots \wedge c_{n,\rm pre}(\mathcal{O})$ checks whether the process is ready to start and switches from s_0 to s_π.
- The error condition $\mathcal{C}_{\rm err}(\mathcal{O}) = c_{1,\rm err}(\mathcal{O}) \vee \cdots \vee c_{n,\rm err}(\mathcal{O})$ is triggered by an irreversible failure. It immediately terminates the process and enters the error state s_e from s_0 or s_π.
- The success condition $\mathcal{C}_{\rm suc}(\mathcal{O}) = c_{1,\rm suc}(\mathcal{O}) \wedge \cdots \wedge c_{n,\rm suc}(\mathcal{O})$ indicates the successful execution of the process. Its activation triggers a switch from s_0 or s_π to s_1.

The boundary conditions depend on a set of objects $\mathcal{O}$ that constitute the process environment. An object $o \in \mathcal{O}$ is characterized by its Cartesian pose $\mathbf{T}_o$ and possibly also physical properties such as mass, centre of mass or inertia. All objects have a unique identifier (object) and a handle (o_1).

The TMS input process specifications are supplied by process experts using established standards like the German curricula for trainees in metalworking[40], electronics[16] and mechatronics[15]. These standards and associated norms form the backbone of various industrial processes by delineating boundary conditions, procedural steps, requisites and goals. Process experts leverage these resources to configure automation tasks and streamline their optimization.

By removing the need for explicit robotics expertise, our framework reduces the reliance on integrators, thereby minimizing planning complexities and financial burdens. As a first step, the TMS encompasses processes such as machine-tending (operating levers and pressing buttons), assembly (insertion) and material-processing (bending and cutting).

Skill synthesis

We devised a synthesis procedure to formally close the gap between process specification and skill implementation. The selection of a robot manipulation policy $\boldsymbol{\pi}_{\rm d}(\boldsymbol{\Omega}, \boldsymbol{\theta}_\pi)$ from a desired manipulation process p is expressed by

$$\mathcal{T} = \mathcal{T}_1 \circ \mathcal{T}_2 \circ \mathcal{T}_3 \circ \mathcal{T}_4 : p \rightarrow \pi_{\rm d}, \tag{6}$$

where $\mathcal{T}$ is the taxonomic algorithm that maps p to a unique $\boldsymbol{\pi}_{\rm d}$. Then $\boldsymbol{\pi}_{\rm d}$ is jointly learned with the controller using a suitable algorithm (the policy and controller parameters $\boldsymbol{\theta}_\pi$ and $\boldsymbol{\theta}_{\rm c}$ are learned) such that $\pi_{\rm d}^* = \pi_{\rm d}(\boldsymbol{\Omega}, \boldsymbol{\theta}_\pi^*)$ and $\boldsymbol{\tau}_{\rm d}^* = \boldsymbol{\tau}_{\rm d}(\boldsymbol{\theta}_{\rm c}^*)$ are the optimal policy and controller solving p. The algorithm steps $\mathcal{T}_1$ to $\mathcal{T}_4$ correspond to the ranks in the taxonomy. Π_0 is the initial set of all available policies. $\mathcal{T}$ iteratively narrows Π_0 down to a single $\boldsymbol{\pi}_{\rm d}$ by executing the following steps, which are currently still done manually but can be automated:

(1) The domain rank $\mathcal{T}_1$ selects policies based on their desired wrench $\mathbf{f}_{\rm d}$: $\Pi_1 = \{\boldsymbol{\pi}_{\rm d} \mid \mathbf{f}_{\rm d}(t) = \mathbf{0} \oplus \mathbf{f}_{\rm d}(t) = \text{const.} \oplus \mathbf{f}_{\rm d}(t) \in \{\mathbf{f}_{{\rm d},1}, \ldots, \mathbf{f}_{{\rm d},n}\} \oplus \mathbf{f}_{\rm d} = \mathbf{g}(t)\}$, where $\mathbf{g}(t)$ is an arbitrary function that drives the evolution of the wrench and $\oplus$ is an exclusive or.

(2) The class rank $\mathcal{T}_2$ selects policies that reach s_1, that is the ones that can, in principle, adhere to the boundary conditions of p: $\Pi_2 = \{\boldsymbol{\pi}_{\rm d} \mid s(t) = s_1 \text{ for } t \rightarrow \infty\}$.

(3) The subclass rank $\mathcal{T}_3$ selects policies that follow the process operations defined by Δ: $\Pi_3 = \{\boldsymbol{\pi}_{\rm d} | \delta \rightarrow \text{true} \;\forall\, \delta \in \Delta\}$.

(4) The instance rank $\mathcal{T}_4$ selects the policy with the fewest parameters: $\Pi_4 = \pi_{\rm d} = c(\{\boldsymbol{\pi}_{\rm d} | \min_{|\boldsymbol{\theta}_\pi|} \pi_{\rm d} \in \Pi_{\rm o,p}\})$. c represents a choice if more than one policy is left. Furthermore, the parameter domain $\mathbb{D} = f_{\mathbb{D}}(\boldsymbol{\theta}_\pi, \boldsymbol{\theta}_{\rm c}, \mathbb{C})$ is determined by system and process constraints $\mathbb{C}$. This (so far manual) step is represented by $f_{\mathbb{D}}$.

The rationale behind $\mathcal{T}_4$ selecting the policy with the fewest parameters is to facilitate the subsequent learning problem. Furthermore, we empirically found in refs. 41,42 that process-driven and tailored policies generally outperform more complex and initially unspecific

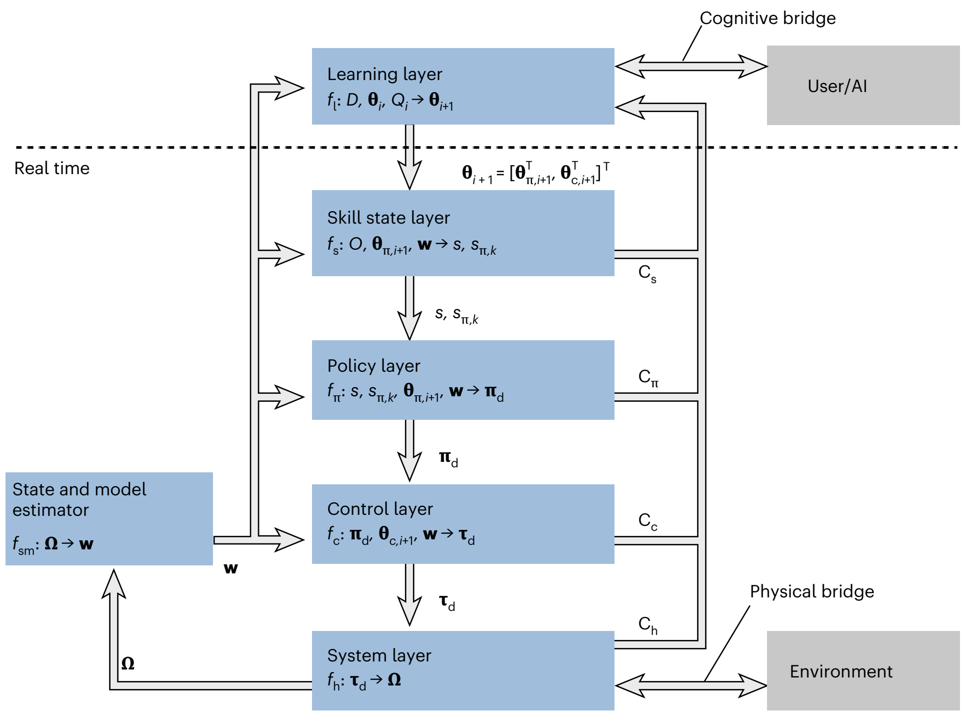
Cognitive bridge
User/AI
Learning layer
f_l: D, $\boldsymbol{\theta}_i$, $Q_i \rightarrow \boldsymbol{\theta}_{i+1}$
Real time
$\boldsymbol{\theta}_{i+1} = [\boldsymbol{\theta}^T_{\pi,i+1}, \boldsymbol{\theta}^T_{c,i+1}]^T$
Skill state layer
f_s: O, $\boldsymbol{\theta}_{\pi,i+1}$, $\mathbf{w} \rightarrow s$, $s_{\pi,k}$
C_s
s, $s_{\pi,k}$
Policy layer
f_π: s, $s_{\pi,k}$, $\boldsymbol{\theta}_{\pi,i+1}$, $\mathbf{w} \rightarrow \boldsymbol{\pi}_d$
C_π
$\boldsymbol{\pi}_d$
State and model estimator
f_{sm}: $\boldsymbol{\Omega} \rightarrow \mathbf{w}$
Control layer
f_c: $\boldsymbol{\pi}_d$, $\boldsymbol{\theta}_{c,i+1}$, $\mathbf{w} \rightarrow \boldsymbol{\tau}_d$
C_c
$\mathbf{w}$
$\boldsymbol{\tau}_d$
Physical bridge
C_h
$\boldsymbol{\Omega}$
System layer
f_h: $\boldsymbol{\tau}_d \rightarrow \boldsymbol{\Omega}$
Environment

Fig. 4 | An architectural overview of the GGTWreP framework. The learning layer generates parameters for downstream layers. The skill state layer selects the appropriate policy state, which is executed by the policy layer. Generated commands are sent to the control layer, and the system layer then forwards torque commands to the robot hardware while receiving the latest robot state. This state is processed by the state and model estimator to produce an updated world state, which is made available to all layers.

ones (for example, modern vision–language–action models), if a strict specification is provided, that is a narrow problem is to be solved in principle.

The synthesis procedure for our two examples, Ethernet plug insertion and cutting cloth, is described below. More details are provided in appendix 6 in the Supplementary Information. In these examples, we use f_g and $f_{g,d}$ to denote the 1-DoF grasp force measured by the gripper and the desired grasp force.

Synthesis 1. Inserting an Ethernet plug

Process specification:

$$\mathcal{O} = \{o_1, o_2, o_3\},\ \mathcal{C}_{\text{pre}} = \{f_g \geq f_{g,d}\},\ \mathcal{C}_{\text{err}} = \{f_g < f_{g,d}\},$$
$$\mathcal{C}_{\text{suc}} = \{T_{o_1} \in \mathcal{U}(o_3)\},$$
$$\Delta = \{\delta_{1,2} := T_{o_1} \in \mathcal{U}(o_2), \delta_{2,3} := f_{\text{ext},z} > f_{\text{contact}}\}$$

f_{contact} is a contact threshold and $f_{\text{ext},z}$ is the z-axis component of the external wrench.

(1) The process involves a search behaviour with complex interaction forces. $\mathcal{T}_1$ yields

$$\Pi_1 = \{\pi_{d,13}, \pi_{d,14}, \pi_{d,15}, \pi_{d,16}, \pi_{d,17}, \pi_{d,18}, \pi_{d,19}, \pi_{d,20}, \pi_{d,27}, \pi_{d,28}, \pi_{d,32}\}$$

(2) Policies that can reach the final state s_1 are selected by $\mathcal{T}_2$ to be

$$\Pi_2 = \{\pi_{d,17}, \pi_{d,20}, \pi_{d,27}, \pi_{d,28}, \pi_{d,32}\}.$$

(3) Policies not guaranteed to reach the process substates are removed, so that $\mathcal{T}_3$ yields

$$\Pi_3 = \{\pi_{d,27}, \pi_{d,28}, \pi_{d,32}\}.$$

(4) From Π_3, the least complex policy is selected by $\mathcal{T}_4$ to be $\boldsymbol{\pi}_{d,27}$.

Synthesis 2. Cutting cloth

Process specification:

$$\mathcal{O} = \{o_1, o_2, o_3, o_4, o_5\},\ \mathcal{C}_{\text{pre}} = \{f_g \geq f_{g,d}\},\ \mathcal{C}_{\text{err}} = \{f_g < f_{g,d}, \mathbf{f}_{\text{ext}} < \mathbf{f}_{\text{cut}}\},$$
$$\mathcal{C}_{\text{suc}} = \{T_{o_1} \in \mathcal{U}(o_5)\},$$
$$\Delta = \{\delta_{1,2} := T_{o_1} \in \mathcal{U}(o_2), \delta_{2,3} := \mathbf{f}_{\text{ext}} > \mathbf{f}_{\text{cut}}, \delta_{3,4} := T_{o_1} \in \mathcal{U}(o_4)\}$$

$\mathbf{f}_{\text{cut}}$ is the desired cutting force.

(1) The process requires a constant cutting force, but it also has phases without any contact. Thus, $\mathcal{T}_1$ yields

$$\Pi_1 = \{\pi_{d,22}, \pi_{d,24}, \pi_{d,26}, \pi_{d,30}, \pi_{d,31}\}.$$

(2) Policies that can reach the final state s_1 are selected by $\mathcal{T}_2$ to be

$$\Pi_2 = \{\pi_{d,26}, \pi_{d,31}\}.$$

(3) Policies not guaranteed to reach the process substates are removed, so that $\mathcal{T}_3$ yields

$$\Pi_3 = \{\pi_{d,26}\}.$$

(4) From Π_3, the least complex policy is selected by $\mathcal{T}_4$ to be $\boldsymbol{\pi}_{d,26}$.

Experimental study

We implemented solution skills based on our Graph-Guided Twist-Wrench Policy (GGTWreP) framework[41] (Fig. 4; see Methods for details). The framework encodes expert knowledge about robotics through process-tailored policies with low-level control parameter spaces. It is connected to the process specification through the TMS. This approach allows for sample-efficient learning and tuning by seamlessly combining knowledge from various domains and state-of-the-art machine learning. To illustrate the power of this

https://doi.org/10.1038/s42256-025-01045-3

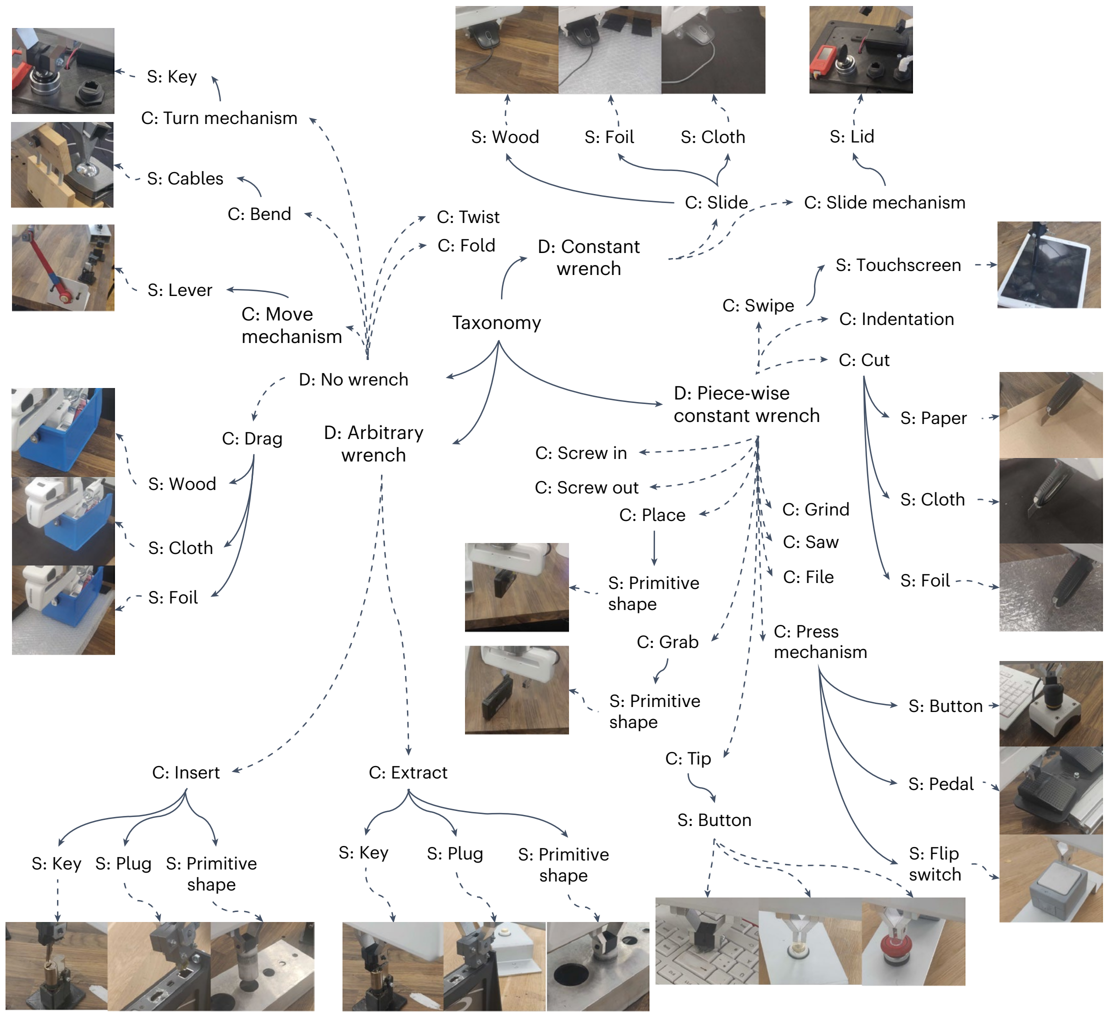

Fig. 5 | Experimental set-ups. Experimentally validated skills are shown with the set-up used in the context of the taxonomy. For clarity, we indicate the taxonomy ranks as domain (D), class (C) and subclass (S). Instances are omitted because they are represented by the pictures.

approach, we implemented 28 real-world manipulation skills using the GGTWreP framework (Fig. 5 and Supplementary Table 1). Overall, high performance levels could be achieved with minimum execution time and contact moments. We injected artificial errors $\mathbf{e}$ at the kinesthetically taught poses of the skills to test their robustness to disturbances from exteroception (which is explained in more detail in Methods). The injected errors $\mathbf{e} \sim \mathcal{U}_{[\mathbf{e}_l,\mathbf{e}_u]}$ follow a uniform distribution with lower bound $\mathbf{e}_l$ and upper bound $\mathbf{e}_u$.

Table 1 summarizes the achieved performance, the robustness against goal pose randomization, and the average and standard deviation of these metrics. The skills were either autonomously learned or tuned by a domain expert using the simple procedure detailed in Methods. Surprisingly, the seeming disadvantage of having to design a large number of different skills is mitigated because the vast majority of policies can be transferred without modification within the same skill class. Thus, we used one policy for each skill class and needed only to adapt (or relearn) the parameters $\boldsymbol{\theta} = \left[\boldsymbol{\theta}_\pi^\mathrm{T}, \boldsymbol{\theta}_\mathrm{c}^\mathrm{T}\right]^\mathrm{T}$ to find the new optimum. Some policies are even directly transferable between different classes. When looking at the building blocks of our policies, we may say that many manipulation processes can be solved by using a small toolset of building blocks, which confirms that the proposed approach is versatile enough to be relevant for realistic scenarios.

As a final verification case, we solved the assembly of an industrial bottle-gripping mechanism, which is widely used in large numbers in bottle-filling plants. The mechanism is a part of a rapidly rotating filling machine, and it grabs and holds bottles during the filling process. This assembly involves eight successive steps using various skills, including insertion, screwing, placing and grasping, and it features a range of physical properties for different materials (aluminium or plastic) and high-precision tolerances <0.1 mm. More details, such as formal skill descriptions and the step-by-step assembly process, can be found in appendix 9 in the Supplementary Information. We verified the approach by running the application $n = 50$ times. The final execution time required for the assembly was 100 s, and the success rate was 100 %. Note that we cannot disclose any information about our collaboration partner using this device due to confidentiality reasons and that this work has neither been financed nor influenced by this collaboration.

Discussion

The TMS introduced in this paper allows us to encode robot controls and policies by connecting formal process definitions with compatible models of tactile skills. By leveraging established process definitions and the experimentally validated GGTWreP framework, the TMS provides a systematic approach for developing a comprehensive manipulation curriculum for robots.

A key question in effectively using the TMS is how to identify proven solutions for industrial manufacturing processes directly from the taxonomy, which would enable robots to use them as foundational process data for their manipulation capabilities. Human vocational training curricula seem well suited for this purpose. In Germany, these curricula cover over 130 state-certified technical apprenticeships

Table 1 | Experimental results

Skill	Task	Execution time		Contact torque	
		Robustness (%)	Value (s)	Robustness (%)	Value (Nm)
Insert	Cylinder	94	1.76 ± 0.36	92	3.21±0.35
	Key	100	1.1±0.18	98	2.97±0.244
	Ethernet plug	100	1.04±0.19	100	2±0.55
Extract	Cylinder	100	0.35±0.05	100	6.03±0.123
	Key	100	0.31±0.02	100	5.46±1.277
	Ethernet plug	100	0.23±0.001	100	2.29±0.023
Press mechanism	Pedal	100	NA	100	4.12±0.223
	Flip switch	100	NA	100	1.6±0.083
	User stop	100	NA	98	3.39±0.17
Tip	Enter key	100	0.82±0.005	100	0.59±0.013
	Red button	100	0.69±0.034	100	1.41±0.137
	White button	100	0.69±0.008	100	1.32±0.023
Grab	HDMI switch	100	2.87±0.005	NA	NA
Place	HDMI switch	100	2.42±0.064	NA	NA
Slide object	Wood	100	1.21±0.008	100	8.34±0.125
	Cloth	100	1.21±0.005	100	8.69±0.098
	Foil	100	1.21±0.005	100	9.56±0.077
Drag	Wood	100	1.02±0.06	100	6.42±0.04
	Cloth	100	1.09±0.06	100	6.5±0.016
	Foil	100	1.01±0.008	100	11.86±0.078
Cut	Carton	100	1.69±0.009	100	7.16±0.502
	Cloth	100	1.75±0.089	100	6.08±0.105
	Foil	100	1.72±0.011	100	5.92±0.145
Turn mechanism	Key	95	0.71±0.19	100	0.94±0.345
Swipe	Tablet	100	1.4±0.17	66	3.1±0.155
Move mechanism	Red lever	100	1.33±0.036	86	4±0.079
Bend	Cables	100	1.99±0.006	100	4.37±0.195
Slide off	Battery case	98	1.13±0.66	96	7.22±0.623

NA, not applicable.

(out of over 320 in total), such as industrial mechanic, mechatronics technician, electronics technician and tool mechanic, and roughly 34,000 DIN norms. They contain definitions for standard processes that are taught to 0.5 million technical apprentices each year based on specialized standard literature[15,40]. In Germany, Switzerland and Austria, these curricula are standardized at the national level. This standardization has evolved over a century, beginning with the establishment of the Chamber of Commerce and Industry in 1842, followed by the founding of Deutsches Institut für Normung (DIN) in 1917, Schweizerische Normen-Vereinigung in 1919 and Österreichisches Normungsinstitut in 1920.

If robots are to automate today's industry and become useful assembly assistants, following such established industrial curricula may provide a foundation of prior knowledge for formalizing and then learning and extending these skills. Although these curricula are intended for human use, such as in vocational schools or as an everyday reference for professionals, considering recent advances in large language models (LLMs), it seems reasonable to (at least partially) automate the transformation of these curricula into formal tactile robot skills. This could lead to the creation of a taxonomy of skills and would essentially form a solid starting point for a robot curriculum.

The clear descriptions with transferable parameters and the built-in learning capabilities of the GGTWreP framework yield a high degree of versatility. These features may enable process experts without specific robotics knowledge to deploy robots in the field with little configuration time. As a first use case, we implemented 28 tactile skills from manufacturing industry. The implementation exhibited robust behaviour and high performance in various automation tasks that have been subjected to significant process disturbances. Importantly, the simple transfer of policies and the efficient learning through the parameter vector $\boldsymbol{\theta}$ demonstrate the versatility enabled by the taxonomy.

Energy consumption

As this approach enables the learning of a wide range of skills in realistic settings, the issue of energy consumption in real-world 24/7 skill acquisition settings is important. Therefore, we compare the computational energy required for our approach with that of an exemplary state-of-the-art deep learning system, as illustrated in Fig. 3 (details can be found in appendix 4 in the Supplementary Information). The GGTWreP model[41] (see Fig. 4) used in this work is compared with the deep deterministic policy gradient method (DDPG)[43]. The results indicate that using current state-of-the-art data-based methods to learn many skills may have substantial resource demands, as was anticipated, for example, in ref. 44. However, using the GGTWreP framework requires an order of magnitude less energy than DDPG. We compare these approaches to highlight potential limitations and expenses of data-driven methods in contrast to structured methodologies. We anticipate that these insights will hold substantial significance for forthcoming industrial applications, particularly those intended for extensive scaling. Additionally, on the right of Fig. 3 we show bar plots that compare the achieved success rate and performance of the two methods on the cylinder insertion problem from the taxonomy (see Fig. 5).

Limitations and outlook

Limitations of the current implementation of GGTWreP include that manual design is required to reuse its process-tailored models for different classes. However, recent works, such as ref. 45, indicate that LLMs could create such policies in a scalable way. Furthermore, the current scope of our taxonomy was limited to a moderate set of manipulation skills and serial manipulators with linear two-fingered grippers; we did not consider interactions with soft materials, which could be addressed by integrating suitable controllers. As we do not make assumptions about the robot controller, it would be possible to extend the presented approach to processes that involve deformable objects[46].

A related aspect is the flexibility required of the grasping device. Current industrial processes are typically rigid. They disregard unexpected variations and treat deviations as faults rather than opportunities for adaptation. Future robots must overcome these limitations if they are to be applied in domains requiring complex processes like textile manufacturing. Although our approach does not yet handle gripper slippage or object shifts during interactions, promising directions for future developments include: (1) automated additive manufacturing of task-specific optimized gripper fingers[47] and (2) compliant or force-controlled grippers equipped with tactile sensors to enhance robustness and to adapt nominal tactile policies to varying conditions[48].

We plan to extend the experimental work to even more manipulation processes and skills, which would also require us to extend the taxonomy to other domains and robot systems. Specific examples are bimanual tasks in the household service domain, including skills such as handing items to humans or supporting them physically. In fact, further efforts conceptually building upon the framework described in this manuscript have already been carried out[49]. Furthermore, the integration of the GGTWreP framework with state-of-the-art deep learning techniques may open up possibilities to make the skill models more generic.

Finally, the formal architecture of the TMS, with its well-defined semantics, could provide an automated library for high-level robot programming languages that have become widespread in recent years (see, for example, ref. 2). In this context, we plan to use LLMs and vision–language models to generate formal process definitions from natural language input. By using its synthesis procedure to generate new skills on the fly from user input and its ability to learn them, it could become a programming tool for non-experts. However, a caveat of this approach is that the amount of data required to build proper, specialized LLMs or vision–language models is typically very large and implies extensive community effort. Our initial thoughts on this topic are provided in appendix 10 in the Supplementary Information.

Methods

GGTWreP framework

To implement process-compatible tactile skills, we rooted our efforts in the GGTWreP framework[41], which has several hierarchical layers, with each layer modelling a different aspect of tactile manipulation. This multilayered structure descends from a learning layer down to the hardware system layer that is directly connected to the physical robot platform, which is coupled to the real world (Fig. 4). $\mathbf{w} \in \mathcal{W}$ denotes an element of the world state space $\mathcal{W}$, containing, for example, the robot poses, external forces or object positions. $\mathbf{\Omega}$ denotes the percept vector, which contains information received by internal or external sensors. Appendix 1 in the Supplementary Information provides a nomenclature for all symbols used in the following.

Layers. The framework layers are described in detail in the sections below. Each layer receives inputs and extra parameters from the layer above and provides outputs to the layer below. The layers also provide constraints $\mathbb{C}$ in the context of the task and the limits of the system. These constraints model the limits of a valid input to the respective layer (for example, the maximum admissible velocity). The state and model estimator updates and provides the world state $\mathbf{w}$ with the other components based on the percept vector $\mathbf{\Omega}$ and internal models. Figure 4 provides an overview of the GGTWreP framework with its different layers.

- The learning layer proposes parameters for the next episode in a learning process based on the parameters and quality metric of the previous episode.
- The skill state layer controls a state machine that governs the discrete behaviour of the system.
- The policy layer holds a set of (in general) ordinary differential equations embedded into a graph structure, which produce coordinated twist and wrench commands.
- The control layer implements a unified force and impedance controller that is fed by the policy layer commands and provides desired motor commands for the system layer. This layer also contains safety mechanisms to meet the system and process constraints. It also contains safety mechanisms that ensure that the system and process constraints are fulfilled.
- The system layer is the lowest layer. It sends motor commands from the control layer to the robot hardware. It provides the current robot state to the other layers.

Objects. A skill is instantiated through objects $\mathcal{O}$ that define the environment relevant to the skill, which is like the definition of manipulation processes introduced above. Note that all skills also contain an end effector as a default object. It has the handle EE.

Learning layer. The learning layer executes a learning algorithm that proposes a parameter candidate $\mathbf{\theta}_{i+1} \in \mathbb{D}$ for episode $i+1$ based on the parameters $\mathbf{\theta}_i$ and quality metric $\mathcal{Q}_i$ of the previous episode i and passes the candidate to the skill state layer. $\mathbb{D}$ is the parameter domain and is informed by the constraints $\mathbb{C}$. The learning layer is represented by the functional mapping:

$$f_l : \mathbb{D}, \mathbf{\theta}_i, \mathcal{Q}_i \rightarrow \mathbf{\theta}_{i+1}. \tag{7}$$

Skill state layer. The skill state layer contains a discrete two-layered state machine that consists of four skill states: initial state s_0, policy state s_π, error state s_e and final state s_1. s_0 denotes the beginning, s_1 is active at the end, s_e represents the end state if an error occurs, and s_π activates the policy layer. Three transitions govern the switching behaviour at the top level of the state machine. They directly implement the boundary conditions from the process specification introduced above. Additionally, some of the default conditions come from the physical realities of the robot system:

- The default precondition $\mathcal{C}_{\mathrm{pre},0} = \{T_{\mathrm{EE}} \in \mathrm{ROI}\}$ states that the robot has to be within a suitable region of interest (ROI) depending on the task at hand.
- The three default error conditions $\mathcal{C}_{\mathrm{err},0} = \{|\mathbf{f}_{\mathrm{ext}}| > \mathbf{f}_{\mathrm{ext,max}}, T_{\mathrm{EE}} \notin \mathrm{ROI}, t > t_{\max}\}$ state that the robot may not leave the ROI, exceed the maximum external forces or exhaust the maximum time for skill execution. $\mathbf{f}_{\mathrm{ext,max}}$ is a positive vector.

The policy state s_π contains a state machine layer known as the manipulation graph. It implements the policy state from the process specification. In this graph $G(\Pi_g, \Delta)$, Π_g denotes the set of policies (nodes) and Δ the set of transitions (edges). The transitions are conditions that, if true, switch the current policy according to the graph structure. The skill state layer is represented by the functional mapping:

$$f_s : \mathcal{O}, \boldsymbol{\theta}_\pi, \mathbf{w} \rightarrow s, s_{\pi,k}, \quad (8)$$

where s is the current skill state and $s_{\pi,k}$ the kth substate in the policy state.

Policy layer. The policy layer contains a set of ordinary differential equations Π_g. Each system represents one policy $\boldsymbol{\pi}_d$ and implements one process state while maintaining the stated conditions. The currently active $\boldsymbol{\pi}_d$ is determined by the skill state layer. The policy layer functional mapping is expressed as:

$$f_\pi : s, s_{\pi,k}, \boldsymbol{\theta}_\pi, \mathbf{w} \rightarrow \boldsymbol{\pi}_d. \quad (9)$$

For $s \neq s_\pi$, a default policy

$$\boldsymbol{\pi}_d = \begin{bmatrix} \dot{\mathbf{x}}_d \\ \mathbf{f}_d \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad f_{g,d} = f_g,$$

is activated, where f_g denotes the current grasp force. Note that $f_{g,d}$ is the desired grasp force for the end effector of the robot and is passed directly to the robot. For clarity, it was omitted from Fig. 4.

Control layer. The control layer receives commands $\boldsymbol{\pi}_d$ from the policy layer and calculates the desired motor commands $\boldsymbol{\tau}_d$. We chose a basic form of unified force and impedance control:

$$\boldsymbol{\tau}_d = J_x(\mathbf{q})^T \left[\mathbf{f}_d - K_d(\boldsymbol{\theta}_{c,k})\tilde{\mathbf{x}} - D_d(M_x(\mathbf{q}), \boldsymbol{\theta}_{c,k}, \boldsymbol{\theta}_{c,d})\dot{\tilde{\mathbf{x}}}\right]. \quad (10)$$

$\tilde{\mathbf{x}} = \mathbf{x} - \mathfrak{f}_{q2a}(\mathbf{x}_d)$ denotes the motion error and $\mathfrak{f}_{q2a}$ is a transformation from quaternion to axis-angle representation. K_d is the desired positive definite stiffness matrix, D_d is the desired positive definite damping, and $\boldsymbol{\theta}_{c,d}$ and $\boldsymbol{\theta}_{c,k}$ are the damping factors and stiffness gains. $M_x(\mathbf{q})$ denotes the Cartesian mass matrix[37].

Architecturally, the control layer is encoded by the functional mapping

$$f_c : \boldsymbol{\pi}_d, \boldsymbol{\theta}_c, w \rightarrow \boldsymbol{\tau}_d. \quad (11)$$

Furthermore, the control layer hosts safety mechanisms such as value and rate limitations, collision detection, reflexes and virtual walls.

System layer. The system layer is expressed by the functional mapping

$$f_h : \boldsymbol{\tau}_d \rightarrow \boldsymbol{\Omega}. \quad (12)$$

It defines the control/sensing interface for the hardware system and other devices in the robot and encapsulates any subsequent hardware-specific control loops.

State and model estimator. The state and model estimator holds all the models for internal and external processes. Examples of internal models are the estimated mass matrix $\hat{M}(\mathbf{q})$, Coriolis forces $\hat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})$ and gravity vector $\hat{\mathbf{g}}(\mathbf{q})$. External models describe the state of environmental elements, such as the physical objects handled by the robot. For example, if the robot were to place an object at a new location, a model of the object would be updated with the new pose. The estimator continuously updates the models using $\boldsymbol{\Omega}$. Its functional mapping is

$$f_i : \boldsymbol{\Omega} \rightarrow w. \quad (13)$$

Task frame. The task frame T defines a coordinate frame ${}^{O}T_T$ relative to the origin frame of the robot O. $\boldsymbol{\pi}_d$ is then calculated in the task frame and transformed through ${}^{O}T_T$ into the frame of the origin.

Implementation example

In this section, the steps from a process to a skill implementation is outlined for the two process examples inserting an Ethernet plug and cutting a piece of cloth. The details of the policy selection through $\mathcal{T}$ can be found in appendix 6 in the Supplementary Information together with a visualization in Supplementary Fig. 1.

Inserting an Ethernet plug. In general, an insertion process involves fitting one object into another by aligning their geometries to achieve a form fit. In an industrial context, this process is essential for tasks such as part-mating. Process experts may use specialized literature, such as ref. 50 and norms[51], which is a source of process constraints and requirements, such as maximum forces, velocities and so on. In the GGTWreP framework, these constraints can be directly represented as $\mathbb{C}_s$, $\mathbb{C}_\pi$, $\mathbb{C}_c$ and $\mathbb{C}_h$. These constraints set the limits of the parameter domain for the skills $\mathbb{D}$. To underline the performance of our approach (also for learning) and the difficulty of the addressed insertion problems, we compare related work in appendix 8 in the Supplementary Information. In the following, we outline details of the skill implementation based on the GGTWreP framework.

Process specification. The process specification states that the insertable o_1 has to be moved towards an approach pose o_3. From there, contact is established in the direction of the container o_2. Finally, the insertable has to be inserted into the container:

$$\mathcal{O} = \{o_1, o_2, o_3\},\ \mathcal{C}_{pre} = \{f_g \geq f_{g,d}\},\ \mathcal{C}_{err} = \{f_g < f_{g,d}\},\ \mathcal{C}_{suc} = \{T_{o_1} \in \mathcal{U}(o_3)\},$$
$$\Delta = \{\delta_{1,2} := T_{o_1} \in \mathcal{U}(o_2), \delta_{2,3} := f_{ext,z} > f_{contact}\}.$$

Conditions. There is a default precondition that the robot has to be within the user-defined ROI and an implementation-specific precondition that the robot must have grasped the insertable o_1. The default error conditions are that the external forces and torques must not exceed a predefined threshold, the ROI must not be left and the maximum execution time must not be exceeded. Additionally, the robot must not lose the insertable o_1 at any time. Note that, for clarity, we do not explicitly show the default conditions in Supplementary Fig. 1. The process specification states that, to be successful, o_1 has to be matched with o_2. In the implementation, this is expressed by a predefined maximum distance $\mathcal{U}(o_2)$.

Policies. The insertion skill model consists of three distinct phases: (1) approach, (2) contact and (3) insert. The approach phase uses a simple point-to-point motion generator to drive the robot through free space to o_3. The contact phase drives the robot into the direction of o_2 until contact has been established, that is, when external forces that exceed a defined contact threshold $f_{contact}$ have been perceived. The insertion phase attempts to move o_1 to o_2 by pushing downwards with a constant wrench, while employing a Lissajous figure to overcome friction and material dynamics. Additionally, a simple motion generator controls the orientation of the end effector and its lateral motion towards the goal pose. A grasp force $f_{g,d}$ is applied simultaneously to all three phases to hold o_1 in the gripper.

Cutting a piece of cloth. A cutting process is characterized by dividing an object into two parts using a cutting tool such as a knife. Again, process experts may use specialized literature such as ref. 52 to define a process specification and set up its optimization. In the following section, we outline the details of the skill implementation using the GGTWreP framework.

Process specification. The process specification states that the knife o_1 has to be moved towards an approach pose o_3. From there, contact is established in the direction of the surface o_2. Then, o_1 is moved towards a goal pose o_4 while maintaining contact with the surface. Finally, o_1 is moved to a final retract pose o_5. f_{cut} is the desired cutting force:

$$\mathcal{O} = \{o_1, o_2, o_3, o_4, o_5\},\ \mathcal{C}_{\text{pre}} = \{f_{\text{g}} \geq f_{\text{g,d}}\},\ \mathcal{C}_{\text{err}} = \{f_{\text{g}} < f_{\text{g,d}}, \mathbf{f}_{\text{ext}} < \mathbf{f}_{\text{cut}}\},$$

$$\mathcal{C}_{\text{suc}} = \{T_{o_1} \in \mathcal{U}(o_5)\},$$

$$\Delta = \{\delta_{1,2} := T_{o_1} \in \mathcal{U}(o_2), \delta_{2,3} := f_{\text{ext},z} > f_{\text{cut}}, \delta_{3,4} := T_{o_1} \in \mathcal{U}(o_4)\}$$

Conditions. There is a default precondition that the robot has to be within the user-defined ROI and an implementation-specific precondition that the robot must have grasped the knife o_1. The default error conditions are that the external forces and torques must not exceed a predefined threshold, the ROI must not be left and the maximum execution time must not be exceeded. Additionally, the robot must not lose the knife o_1 at any time, and $f_{\text{ext},z} < f_{\text{contact}}$ must be maintained when moving from o_3 to o_4 in $\boldsymbol{\pi}_3$. The process specification states that, to be successful, o_1 has to be moved towards o_5.

Policies. The cutting skill model consists of four distinct phases: (1) approach, (2) contact, (3) cut and (4) retract. The approach phase uses a simple point-to-point motion generator to drive the robot through free space towards o_3. The contact phase drives the robot into the direction of o_2 until contact has been established, that is, when external forces that exceed a defined contact threshold f_{contact} have been perceived. The cut phase moves o_1 to o_4 using a point-to-point motion generator combined with a constant downward-pushing wrench. The retract phase moves o_1 to o_5 using a point-to-point motion generator. A grasp force $f_{\text{g,d}}$ is simultaneously applied to all four phases to hold o_1 in the gripper.

Experimental set-up

All experiments use the following off-the-shelf hardware:

- A Franka Emika robot arm[2,53]: A 7-DoF manipulator with link-side joint torque sensors and a 1-kHz torque-level real-time interface, which allowed us to directly connect the GGTWreP framework to the system hardware.
- A Franka Emika robot hand: A standard two-fingered gripper that was sufficient for the processes considered.
- Intel NUC: A small PC with an Intel i7 CPU, 16 GB RAM and a solid-state drive. Note that our learning approaches do not require GPU acceleration or distributed computing clusters.

Software: The GGTWreP framework was implemented using a software stack developed at the Munich Institute of Robotics and Machine Intelligence. The code can be downloaded from ref. 54.

For the validation experiment, we executed each skill model 50 times on the same set-up. A single trial involved executing a particular skill model until it terminated. When appropriate, we used artificial errors $\mathbf{e}$ to offset the manually taught goal poses of the skill in the validation experiment to simulate a more realistic process environment with major disturbances. For example, in typical industrial environments, the moving parts of heavy machines cause process disturbances that impact the precision of the robot. The process-specific experiment set-ups are depicted in Fig. 5. Supplementary Table 1 provides a short description of the skills and lists the selected policy and the injected pose error when the latter is available. For the validation experiment and the optimization experiments (both autonomous learning and manual tuning), roughly 6,000 episodes were run in total. Taking into account the optimization times and set-up times (physically adjusting the environment around the robot for the next experiment), the experimental work took about one net month to complete.

Learning and tuning skills

The parameters for tactile skills $\boldsymbol{\theta} = [\boldsymbol{\theta}_{\text{c}}^{\text{T}}, \boldsymbol{\theta}_{\pi}^{\text{T}}]^{\text{T}}$ were partially learned and partially manually tuned. The parameter learning procedure is based on our previous work, such as refs. 41,42,55. We used the physical experimental set-ups and goal poses described in 'Results'.

Algorithm for partitioning the parameter space. We used the parameter space partition algorithm that was introduced in ref. 42. The algorithm runs for k generations with n_{e} episodes per generation. For each episode i, parameters $\boldsymbol{\theta}_i$ were sampled ~ $q(a)$ in a hypercube sample space with $q(a)$ as the sampling policy. These were translated into a solution space and applied to the optimization problem. The resulting reward r_i was stored together with the parameters $\boldsymbol{\theta}_i$. When an episode was unsuccessful, the reward r_i was set to a negative value, $r_i = -1$. This was done to ensure that there was a negative classification in the update step. At the end of each generation, the sampling policy $q(a)$ was updated. The sampling policy $q(a)$ consists of two elements: a proposal policy $p(a)$ and a filtering policy $f(p(a))$. $p(a)$ generated parameter candidates until one was accepted by the filtering policy $f(p(a))$. Specifically, $p(a)$ proposed parameters $\boldsymbol{\theta}_i$, which were then evaluated by the filtering policy $f(\boldsymbol{\theta}_i)$. The filtering policy was implemented as a nonlinear support vector machine.

Proposal policy. At the beginning, the proposal policy was a Latin hypercube sampler[56], as there was then not enough data to generate meaningful parameter proposals. Instead, the available solution space was evenly sampled. After the first generation, a uniform random sampler was used. In later generations, assuming sufficient data were available, a Gaussian mixture model was used as the policy.

Filtering policy. The filtering policy is a nonlinear support vector machine with radial-basis-function kernels. It was used only if enough successful (in the sense of a successful skill execution) samples were available to ensure a robust estimation.

Optimization procedure. Each optimization procedure was run for $n_{\text{e}} = 200$ episodes. Optimization minimized the execution time and contact moments in two separate experiments. Each episode had the following steps:

- The learning algorithm proposed policy and controller parameters $\boldsymbol{\theta}_i = [\boldsymbol{\theta}_{\pi,i}^{\text{T}}, \boldsymbol{\theta}_{\text{c},i}^{\text{T}}]^{\text{T}}$.
- A skill was executed with $\boldsymbol{\theta}_i$, and the measured quality metric $\mathcal{Q}_i$ was fed back to the algorithm.
- A predefined reset procedure moved the robot on a path back to its initial state.

Thereafter, all the skills converged to an optimal parameter set $\boldsymbol{\theta}^*$, which was used in the experiments presented. Detailed examples for this skill-learning approach can be found in refs. 41,42. The procedure for manual parameter tuning is like autonomous learning, except that the role of the learning algorithm is taken by an expert programmer.

Data availability

All models used are described in the Supplementary Materials. Optimized parameters for the skill models as well as the performance results of the main experiment can be found in ref. 57.

Code availability

The software needed to run the robot during the experiments is available from ref. 54.

References

1. Hirzinger, G. et al. DLR's torque-controlled light weight robot. III. Are we reaching the technological limits now? In *Proc. International Conference on Robotics and Automation (ICRA)* 1710–1716 (IEEE, 2002).
2. Haddadin, S. et al. The Franka Emika robot: a reference platform for robotics research and education. *IEEE Robot. Autom. Mag.* **29**, 46–64 (2022).
3. Albu-Schäffer, A., Ott, C. & Hirzinger, G. A unified passivity-based control framework for position, torque and impedance control of flexible joint robots. *Int. J. Robot. Res.* **26**, 23–39 (2007).
4. Black, K. et al. π_0: A vision-language-action flow model for general robot control. Preprint at arxiv.org/abs/2410.24164 (2024).
5. Kroemer, O., Niekum, S. & Konidaris, G. A review of robot learning for manipulation: challenges, representations, and algorithms. *J. Mach. Learn. Res.* **22**, 1–82 (2021).
6. Tsarouchi, P., Makris, S. & Chryssolouris, G. Human–robot interaction review and challenges on task planning and programming. *Int. J. Comput. Integr. Manuf.* **29**, 916–931 (2016).
7. Pedersen, M. R. et al. Robot skills for manufacturing: from concept to industrial deployment. *Robot. Comput.-Integr. Manuf.* **37**, 282–291 (2016).
8. Indri, M., Grau, A. & Ruderman, M. Guest editorial special section on recent trends and developments in industry 4.0 motivated robotic solutions. *IEEE Trans. Ind. Inform.* **14**, 1677–1680 (2018).
9. Grischke, J., Johannsmeier, L., Eich, L. & Haddadin, S. Dentronics: review, first concepts and pilot study of a new application domain for collaborative robots in dental assistance. In *Proc. International Conference on Robotics and Automation (ICRA)* 6525–6532 (IEEE, 2019).
10. Seidita, V., Lanza, F., Pipitone, A. & Chella, A. Robots as intelligent assistants to face Covid-19 pandemic. *Brief. Bioinform.* **22**, 823–831 (2021).
11. Martinez-Martin, E. & del Pobil, A. P. in *Personal Assistants: Emerging Computational Technologies* (eds Costa, A. et al.) 77–91 (Springer, 2018).
12. Tröbinger, M. et al. Introducing GARMI – a service robotics platform to support the elderly at home: design philosophy, system overview and first results. *IEEE Robot. Autom. Lett.* **6**, 5857–5864 (2021).
13. Zinggeler, M. V. The educational duty of the German Chamber of Commerce. *Glob. Bus. Lang.* **7**, 9 (2010).
14. Burmester, J. et al. *Fachkunde Metall* (Verlag Europa-Lehrmittel, 2020).
15. Hebel, H. et al. *Fachkunde Mechatronik* (Verlag Europa-Lehrmittel, 2020).
16. Bumiller, H. et al. *Fachkunde Elektrotechnik* (Verlag Europa-Lehrmittel, 2021).
17. Geib, C. et al. Object action complexes as an interface for planning and robot control. In *Proc. International Conference on Humanoid Robots (Humanoids)* (IEEE-RAS, 2006).
18. Krüger, N. et al. *A Formal Definition of Object-action Complexes and Examples at Different Levels of the Processing Hierarchy*. PACO-PLUS Technical Report (2009).
19. Nicolescu, M. N. & Matarić, M. J. A hierarchical architecture for behavior-based robots. In *Proc. International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)* 227–233 (2002).
20. Zoliner, R., Pardowitz, M., Knoop, S. & Dillmann, R. Towards cognitive robots: building hierarchical task representations of manipulations from human demonstration. In *Proc. International Conference on Robotics and Automation (ICRA)* 1535–1540 (IEEE, 2005).
21. Cohen, B. J., Chitta, S. & Likhachev, M. Search-based planning for manipulation with motion primitives. In *Proc. International Conference on Robotics and Automation (ICRA)* 2902–2908 (IEEE, 2010).
22. Demiris, Y. & Johnson, M. Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning. *Connect. Sci.* **15**, 231–243 (2003).
23. Erlhagen, W. et al. Goal-directed imitation for robots: a bio-inspired approach to action understanding and skill learning. *Robot. Auton. Syst.* **54**, 353–360 (2006).
24. Saveriano, M., Abu-Dakka, F. J. & Kyrki, V. Learning stable robotic skills on Riemannian manifolds. *Robot. Auton. Syst.* **169**, 104510 (2023).
25. Xie, M. et al. Neural geometric fabrics: efficiently learning high-dimensional policies from demonstration. In *Proc. Conference on Robot Learning (CoRL)* 1355–1367 (PMLR, 2023).
26. Levine, S., Wagener, N. & Abbeel, P. Learning contact-rich manipulation skills with guided policy search. In *Proc. International Conference on Robotics and Automation (ICRA)* 156–163 (IEEE, 2015).
27. Gu, S., Holly, E., Lillicrap, T. & Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proc. International Conference on Robotics and Automation (ICRA)* 3389–3396 (IEEE, 2017).
28. Kim, M. J. et al. Openvla: an open-source vision-language-action model. In *8th Annual Conference on Robot Learning* (CoRL, 2024).
29. Cutkosky, M. R. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Trans. Robot. Autom.* **5**, 269–279 (1989).
30. Bullock, I. M., Ma, R. R. & Dollar, A. M. A hand-centric classification of human and robot dexterous manipulation. *IEEE Trans. Haptics* **6**, 129–144 (2012).
31. Huckaby, J. O. & Christensen, H. I. A taxonomic framework for task modeling and knowledge transfer in manufacturing robotics. In *Proc. Workshops at the 26th Conference on Artificial Intelligence (AAAI)* (2012).
32. Leidner, D., Borst, C., Dietrich, A., Beetz, M. & Albu-Schäffer, A. Classifying compliant manipulation tasks for automated planning in robotics. In *Proc. International Conference on Intelligent Robots and Systems (IROS)* 1769–1776 (IEEE/RSJ, 2015).
33. Levine, S., Finn, C., Darrell, T. & Abbeel, P. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.* **17**, 1334–1373 (2016).
34. Nguyen, H. & La, H. Review of deep reinforcement learning for robot manipulation. In *Proc. International Conference on Robotic Computing (IRC)* 590–595 (IEEE, 2019).
35. Saveriano, M., Abu-Dakka, F. J., Kramberger, A. & Peternel, L. Dynamic movement primitives in robotics: a tutorial survey. *Int. J. Robot. Res.* **42**, 1133–1184 (2023).
36. Karacan, K., Grover, D., Sadeghian, H., Wu, F. & Haddadin, S. Tactile exploration using unified force-impedance control. *IFAC-PapersOnLine* **56**, 5015–5020 (2023).
37. Khatib, O. Inertial properties in robotic manipulation: an object-level framework. *Int. J. Robot. Res.* **14**, 19–36 (1995).
38. Albu-Schäffer, A., Ott, C., Frese, U. & Hirzinger, G. Cartesian impedance control of redundant robots: recent results with the DLR-light-weight-arms. In *Proc. International Conference on Robotics and Automation (ICRA)* 3704–3709 (IEEE, 2003).
39. Haddadin, S. & Shahriari, E. Unified force-impedance control. *Int. J. Robot. Res.* **43**, 2112–2141 (2024).
40. Fischer, U. et al. *Mechanical and Metal Trades Handbook* (Europa Lehrmittel, 2010).
41. Johannsmeier, L., Gerchow, M. & Haddadin, S. A framework for robot manipulation: skill formalism, meta learning and adaptive control. In *Proc. International Conference on Robotics and Automation (ICRA)* 5844–5850 (IEEE, 2019).

42. Voigt, F., Johannsmeier, L. & Haddadin, S. Multi-level structure vs. end-to-end-learning in high-performance tactile robotic manipulation. In *Proc. Conference on Robot Learning (CoRL)* 2306–2316 (2020).
43. Lillicrap, T. P. et al. Continuous control with deep reinforcement learning. Preprint at arxiv.org/abs/1509.02971 (2015).
44. Thompson, N. C., Greenewald, K., Lee, K. & Manso, G. F. The computational limits of deep learning. In *Ninth Computing within Limits* (LIMITS, 2023).
45. Bharadhwaj, H. et al. RoboAgent: generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2024).
46. Li, Y. et al. Force, impedance, and trajectory learning for contact tooling and haptic identification. *IEEE Trans. Robot.* **34**, 1170–1182 (2018).
47. Ringwald, J. et al. Towards task-specific modular gripper fingers: automatic production of fingertip mechanics. *IEEE Robot. Autom. Lett.* **8**, 1866–1873 (2023).
48. Yamaguchi, A. & Atkeson, C. G. Recent progress in tactile sensing and sensors for robotic manipulation: can we turn tactile sensing into vision? *Adv. Robot.* **33**, 661–673 (2019).
49. Karacan, K., Sadeghian, H., Kirschner, R. & Haddadin, S. Passivity-based skill motion learning in stiffness-adaptive unified force-impedance control. in *Proc. International Conference on Intelligent Robots and Systems (IROS)* 9604–9611 (IEEE/RSJ, 2022).
50. Feldmann, K. (ed.) *Handbuch Fügen, Handhaben, Montieren* (Hanser, 2014).
51. Deutsches Institut für Normung. *Fertigungsverfahren Fügen – Teil 1: Zusammensetzen; Einordnung, Unterteilung, Begriffe* (DIN, 2003).
52. Dietrich, J. in *Praxis der Umformtechnik: Umform-und Zerteilverfahren, Werkzeuge, Maschinen* 266–290 (2018).
53. Haddadin, S. The Franka Emika robot: a standard platform in robotics research. *IEEE Robot. Autom. Mag.* **31**, 136–148 (2024).
54. Johannsmeier, L. et al. Machine intelligence operating system (mios). *GitHub* https://github.com/SchneiderROS/mios, *Zenodo* https://doi.org/10.5281/zenodo.15126974 (2020).
55. Johannsmeier, L. & Haddadin, S. Can we reach human expert programming performance? A tactile manipulation case study in learning time and task performance. In *Proc. International Conference on Intelligent Robots and Systems (IROS)* 12081–12088 (IEEE/RSJ, 2022).
56. Loh, W.-L. On Latin hypercube sampling. *Ann. Stat.* **24**, 2058–2080 (1996).
57. Schneider, R. O. S. Experimental data. *GitHub* https://github.com/SchneiderROS/TactileSkillTaxonomy_ExperimentalData, *Zenodo* https://doi.org/10.5281/zenodo.15127107 (2025).

Acknowledgements

We thank F. Voigt for his support with the application experiment. This work was funded by the German Research Foundation (Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy (EXC 2050/1, Project ID 390696704, Cluster of Excellence 'Centre for Tactile Internet with Human-in-the-Loop' of Technische Universität Dresden (S.H.)). We gratefully acknowledge generous support from Vodafone (S.H.). We gratefully acknowledge funding for this work from the German Research Foundation through the Gottfried Wilhelm Leibniz Programme (Grant No. HA7372/3-1 to S.H.). We acknowledge financial support from the Bavarian State Ministry for Economic Affairs, Regional Development and Energy for the Lighthouse Initiative KI.FABRIK (Phase 1: Infrastructure as well as the research and development programme under Grant No. DIK0249 to S.H.).

Author contributions

The taxonomy concepts were developed by L.J. and S.H. and discussed by all authors. S.H. and L.J. developed the foundations for the tactile skills, tactile platform, contact event chain and tactile feedback level. L.J. and S.H. conceptualized and planned the experiments. L.J. implemented, conducted and processed the results. S.S. assisted in conducting the experiments and processing the results. L.J. and S.H. interpreted the results and wrote the manuscript, which was edited by the co-authors. All authors have approved the content of this paper.

Funding

Open access funding provided by Technische Universität München.

Competing interests

L.J. has a potential conflict of interest as a former employee of Franka Robotics GmbH. S.H. has a potential conflict of interest as a founder and minority shareholder of Franka Emika GmbH. The other authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-025-01045-3.

Correspondence and requests for materials should be addressed to Lars Johannsmeier or Sami Haddadin.

Peer review information *Nature Machine Intelligence* thanks Fangyi Zhang and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.