Towards Agile Robotics – Learning and Assigning Tasks

1st Tapio Heikkilä
Intelligent Robotics
VTT Technical Research Centre of Finland
Oulu, Finland
tapio.heikkila@vtt.fi

2nd Niko Känsäkoski
Intelligent Robotics
VTT Technical Research Centre of Finland
Oulu, Finland
niko.kansakoski@vtt.fi

3rd Martin J. Kollingbaum
mjkollingbaum@hotmail.com

***Abstract*—A solution for automated task planning and resource allocation in robot systems is proposed. Deep Reinforcement Learning is used to create task policies as generic task descriptions in the object context based on product CAD models, and modified Active Inference Framework is used to estimate the quality of task execution in the robot context based on robot accuracy and stiffness models. As an application, a robotic assembly task with finding a best robot to execute the task, is introduced.**

I. Introduction

Flexible and agile production strategies, such as make-to-order and engineer-to-order [1] are means for companies to remain agile in evolving markets. Increasing autonomy (cf. Industry 4.0 or Smart Manufacturing) [2] leads the way towards this. Relying on a conventional product-specific programming and hardware poses a real obstacle to flexibility. Given multiple robots in a manufacturing system, several robots may be qualified to execute a planned and programmed task. This raises a problem of which robot to use for the task. In assembly, handling requirements like contact forces and positional accuracy determine which robot to select in the context of a particular task.

In this paper, a solution for an automated task planning and resource allocation for robotic assembly operations is proposed, based on Reinforcement Learning (RL) [3], [4] and Active Inference Framework (AIF) concepts [5], [6]. Our key novelty is the distinction made between representations in an object context and in a robot context. The planning of how a new task has to be performed is in the object context. Deep Reinforcement Learning (DRL) [3], [4] is used to create robot-agnostic task policies, based on CAD models of assemblies. Task allocation takes place in the robot context. A modified AIF [5], [6] is used to estimate the quality of execution of the learned task policies by available robots based on models of spatial accuracy and stiffness of the robots.

The rest of this paper is organized as follows. In chapter II we analyze related work, and in chapter III we introduce our conceptual approach. In chapters IV and V we apply and test our approach, and in chapters VI and VII we give discussions and short conclusions.

This research was part of the DOMINIC project, funded by the Research Council of Finland and VTT Technical Research Centre of Finland Ltd.

II. Related Work

A mostly agreed taxonomy for task representation comes from skill-based robotics, with layers for tasks, skills, and primitives [7], [8]. The classic Peg-in-Hole activity represents a typical skill in assembly where it is still possible for the peg to get stuck due to the friction forces [9]. For robot task learning, Reinforcement learning (RL) has made substantial progress in recent years. In [10], a Task-Agnostic Learning method (TAL) is proposed, learning fragmented knowledge from task-agnostic data to accomplish new tasks. RL has been also used in adding adaptation to impedance control in peg-in-hole applications [11], [12], [13]. In [14], a pretraining method for RL is presented, using geometric feature representations to describe the contact states in peg-in-hole operations.

Our work is focused on how to learn the details of assembly tasks, that is, primitive operations and skills, with aspects of task and motion planning (TAMP) as also addressed in [15], [16]. We assume that the admittance controllers of the robots can handle the compliance needs as far as the positioning errors are within acceptable limits and stiffness in admittance control can be low enough.

A. Task Allocation

Task assignment problems have been studied in the context of multi-robot systems ([17], [18]), using a variety of optimization criteria, such as energy demands, duration, precision, distances in terms of movements etc. and, correspondingly, the physical constraints and skills of a robot. In [19], deep learning methods are applied to perform such assignments. Multi-Agent Deep Reinforcement Learning for Multi-Robot Applications are discussed in [20] and [21]. These approaches are used in controlling a robotic arm in a peg-in-hole scenario [21], as well as in the coordination of multiple robots [22]. In [23], a holistic framework is proposed to solve the online multi-robot task assignment, planning and coordination problem for heterogeneous fleets of robots subject to non-cooperative tasks that are posted online. The focus is in coordinated use of the shared work space and timing of robot operations. In [24], the online allocation problem in multi-manipulator systems is addressed. In [25], a dynamic allocation of tasks based on reinforcement learning is described, optimizing the sequencing of tasks and the robot selection process.

In our case, the robot selection process should be instantaneous and automatic with the assumption that characteristic workspace-related attributes fulfill requirements, in particular, with a focus on robot accuracy and stiffness.

B. Decision making by Active Inference Framework (AIF)

Active Inference is a framework for modeling cognitive processes that drive the behavior of (human or artificial) agents, acting based on their beliefs with the intention to minimize the "expected surprise" in terms of the perceived changes in the environment [26], [27]. Partially observable Markov decision processes (POMDP) have been used to model active inference ([6], [26], [28]) and AIF applications. AIF in robotics include e.g., state-estimation and control under uncertainty [29], [30], construction of goal-driven behaviors [31], future action planning [32], and deep learning based environment modelling [33].

In our case, we evaluate a learned robot agnostic policy model against robot attributes (accuracy, stiffness) with discrete modeling approach adapted to 3D robot motions.

III. Methodology

A. Approach

A two stage approach is proposed to learn and allocate new tasks. Learning a task as a robot-agnostic policy as "nominal" paths in space from product CAD models is considered in the "object context". Robot-specific attributes are taken into account in a "robot context" to instantiate the object context robot-agnostic policy for each available robot into robot-specific policies that are compared against each other. The "best" robot is assigned then the task by the quality of movements.

First, DRL is used to learn policies describing paths for movrepresentsing assembled objects in space. E.g., in case of the peg-in-hole problem, a policy is learned to move the peg (insertion object) into its target position in the assembly, based on the assembly CAD model. Generating a policy in the object context allows a straightforward comparison of the expected performance of task executions for different robots.

Second, attributes of robots are used to form and evaluate robot-specific policies. The evaluation is based on normalized displacement uncertainties reflecting friction forces along the direction of the compliant contact motion. This allows a comparison of the performance of different robots.

B. Object Context Learning

In the object context, the learning takes place in reverse order with respect to the intended assembly operation, starting from the final state of the assembled model and leading to the start locations of the parts in their disassembled state. Given a simple peg-in-hole problem, it is assumed, that the receiving target part of the assembly is located in the robot work space, using a 3D camera to measure its precise location during task execution. This means that a learned policy has to account for possibly varying locations of the target part. This is shown in figure 1. During training, the starting location is static, whereas

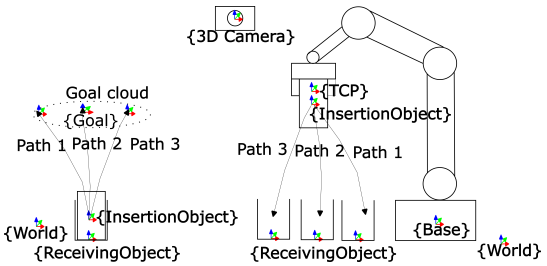

Fig. 1: Object context policy: inverse learning with a goal cloud and adapting to varying locations of the target object in the robot context

the goal state is selected as a random pose within the goal cloud, (figure 1). The insertion object is then moved using cartesian displacements as actions. The aim is to essentially perform a disassembly of the parts.

In the object context, actions are regarded as being incremental linear motions of objects and, through reinforcement learning, a policy is calculated that drives these motions of objects to achieve a goal state. Policy learning in the object context utilizes the CoppeliaSim simulator [34]. Proximal Policy Optimization (PPO) [35] is used to derive the resource-agnostic policy for each task. During the learning process, the agent attempts to maximize a total reward. The total reward is calculated from the sum of the rewards gained by the agent. The reward signal consists of a distance component (equation 1), collision component (equation 2) and goal component (equation 3),

$$r_d = -||p_g - p_o|| \tag{1}$$

where p_o refers to the position of the object and p_g to the position of the goal. The distance component considers the euclidean distance from the center of the mass of the insertion object to the goal position in the goal cloud.

$$r_c = \begin{cases} -10 & \text{Collision detected} \\ 0 & \text{Else} \end{cases} \tag{2}$$

$$r_g = \begin{cases} 1000 & \text{Goal state reached} \\ 0 & \text{Else} \end{cases} \tag{3}$$

C. Robot Context Policy and Evaluation

Different robots have different positioning accuracy and mechanical or controlled stiffness. From these, our policy evaluation method finds the robot with lowest contact forces within the object context policy. Our method is modified from the POMDP model of AIF [27], [36].

The object context policy is transformed to the robot context by extracting a sequence of cartesian displacements of the tool (figure 2) and using inverse kinematics to compose the robot arm motion path points (figure 3). Actions in the robot context are then the tool pose displacements, extended with scaled displacement errors of the robots (see equations 4 - 6). An example of a policy transformed to the robot context is illustrated in figure 4.

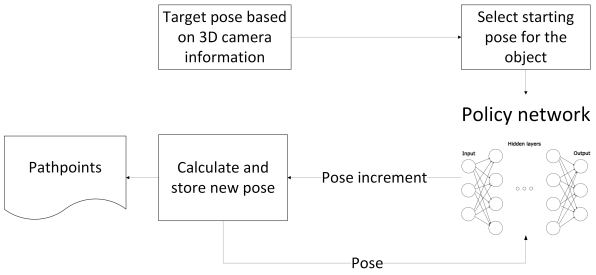

Fig. 2: Path generation from object context policy

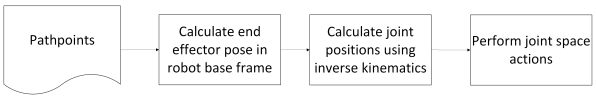

Fig. 3: Robot action generation from pathpoints

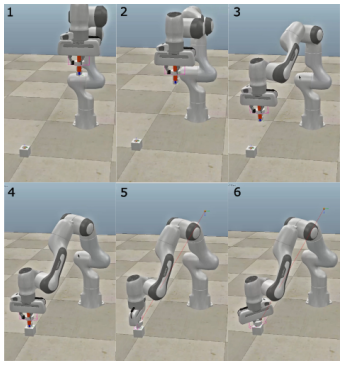

Fig. 4: Representative steps of the screwing task in the simulator environment

We consider that too high contact and friction forces may cause sticking of assembled objects. We take the displacement error and stiffness of a robot, and check how far from the nominal path the robot may at worst take the object, and relate it to the resultant contact force. Stiffness can take place as a mechanical stiffness because of joint and link elasticity, or controlled stiffness, like impedance control [37]. To compare the worst case displacements we consider force normalized displacement error in cases where penetration has occurred, using the Hooke's law:

$$F = -k\Delta x \tag{4}$$

Here, F is the contact force, Δx is the displacement over the contact surface, based on the positioning error model of a robot, and k is the stiffness of the robot arm. From there, we have k_i and x_i for all robots i. We estimate the force according to the Hooke's law. To then generate the force normalized displacement error, we define the reference stiffness as the average of all stiffness values of the robots

$$k_{\text{ref}} = \frac{1}{n}\sum_{1}^{n} k_n \tag{5}$$

The force normalized displacement error is then

$$\Delta x_i = \frac{F_i}{k_{\text{ref}}} \tag{6}$$

We use these to express uncertainty in the state transitions (matrix B below).

The policy evaluation method outlined in [27] applies to discrete models and uses the following variables:

- s_τ : state vector; estimate at timestep τ; e.g., for a 10x10 state space grid, the size of state space is 100
  - Estimates of a 3D object location (x,y,z) projected into the 2D location grids (e.g., (x,y) and (x,z) at time τ)
- o_τ : observation vector; value at time τ with lower resolution as state estimates; e.g., for a 7x7 observation space grid, the size of state space is 49
  - an observation corresponding the 2D state estimates (e.g., (x',y') and (x',z'))
- B : State transition matrix; describes an action by mapping a state to a new state; if a discretized location is a 10x10 grid, the dimension of the state space is 100
  - Given the current action, describes the state transitions, with the noise considering the positional inaccuracy of the robot
- A : Observation matrix: maps state vectors in state spaces to observations in observation spaces
  - Path point observation corresponding a state estimate in the state space; if state space is a 10x10 grid, the state space dimension is 100 and if observation space is a 7x7 grid, the observation space dimension is 49; then A is a 49x100 matrix
- C : Prior preferred observations; list of observation vectors, describing the expected observation according to the object context policy
  - Observations derived from the path points of the general object context policy
- D : Initial state; state vector at time 0
  - The starting position of the object at the beginning of the evaluation based on the robot specific policy realization
- G : expected free energy (EFE); facilitates between achieving preferred outcomes (exploitation) and gaining information to reduce uncertainty (exploration)

The state estimate in the beginning is calculated as a sigmoid function from sum of the log probabilities of the initial state estimate and the initial observation:

$$s_{\pi,\tau=1} = \sigma(\frac{1}{2}(\ln D + \ln A^T o_\tau) \tag{7}$$

The state estimates for time steps ahead are calculated in [27] from the sigmoid function of the predicted state transitions and predicted observations. We don't have predictions of observations and they are practically zero matrices, and the state estimate becomes then:

$$s_{\pi,\tau>1} = \sigma(\frac{1}{2}(\ln (B_{\pi,\tau-1} s_{\pi,\tau-1}))) \tag{8}$$

The expected free energy is then calculated from differences of desired goal observation and observations generated from the states. The former relates to the instrumental value (IV) to the exploitation, i.e., expected extrinsic reward and the latter one epistemic value (EV) to the expected information gain [38].

$$G_\pi = \sum_\tau (As_{\pi,\tau}(\ln As_{\pi,\tau} - \ln C_\tau) - diag(A^T \ln As_{\pi,\tau})) \quad (9)$$

Equation 9 considers one state factor and one observation modality. Often, and also in our case, it is beneficial to include different observation modalities - or sensor spaces. We do this within a policy, by calculating EFE's for each state factor and sensor modality using the equation 9 and calculating the sum of the EFE's per modality:

$$G_{\pi,r} = \sum_{j=1}^{n} \sum_{i=1}^{m} G_{r,i,j} \quad (10)$$

The index i represents the sensor modality, j the time period within which the expected free energy is calculated and r the robot. To reduce complexity, multiple time periods can be used.

The best policy π is finally acquired by applying the sigmoid function to the EFE-values of the policies, and selecting the one with the highest probability.

$$\pi_{best} = \max_{r=1,\ldots,k} \sigma(-G_{\pi,r}) \quad (11)$$

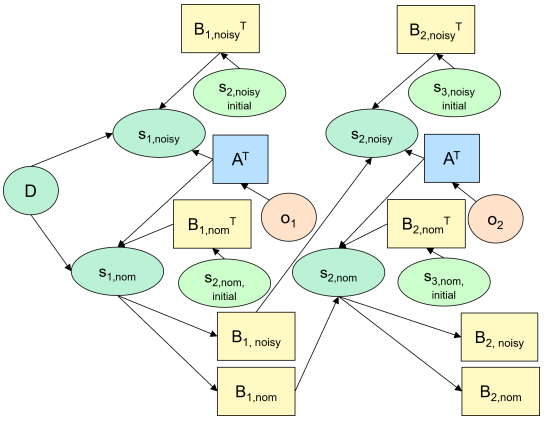

Fig. 5: State transitions with constant noise

We do not have any expected observations, only the uncertainties in the state transitions can be forecast. To comply with both accumulating and state dependent uncertainties, we keep track on nominal state transitions without uncertainties, and state transitions with uncertainties, i.e. "noisy" state transitions. Depending on the type of action in a policy, the uncertainty in the state estimate follows a constant level or accumulates within the action steps. Figure 5 indicates the state transitions between different states. The A matrices model the observations, B matrices the state transitions, s values the states and o values observations. The D matrix models the initial state. At every step there is the nominal state (the object at the timestep) and noisy state, which considers where the robot may ends up in reality. Essentially, we consider transitions to have constant noise, when observations are performed at a step, such as with joint sensor feedback of robotic arms.

IV. Application to Operations in Assembly Tasks

A. State and Observation Spaces

We have applied the object context RL and evaluation policy realizations in robot context for assembly operations. The policy evaluation method applies to discrete models and it becomes easily computationally very expensive when a discrete state space is used for expressing robot motions in 3D space. In our case, the object context learning and, therefore, also the general object context policy, as well as the robot specific polices, consider motions in 3D space. For reducing computational complexity, we discretize and map the states, observations and policy actions from 3D to corresponding 2D projection spaces, i.e., planes, correspoding separate state factors. We first divide the path data into two projections of 2D data. This is done simply by calculating the principal component analysis of the original data from the object context nominal path, and then comparing the axes against each other. The projections are then created by selecting the two axes with the highest eigenvalues as normals of the projection planes, most representing the original data. To reduce the 2D projection spaces further, we utilize a sliding discretization window along the nominal path of the object context policy. This allows us to reduce the dimensions of the state space grid. For example, if the entire path could be covered within a 100x100 grid, where the cell size represents 1 mm in the task space, instead of performing the computation for one 100x100 window, we can consider ten 10x10 windows.

During the evaluation, the position of the object throughout the nominal object path as given by the object context policy, is solved in the robot space by inverse kinematic and tool models of the robot. Conceptually, the object position in each window is considered individually as a state factor, and the expected free energy is calculated for all robot policies in all windows. This is calculated essentially from the difference between the preferred observations (nominal path) and the true perceptions, and the probability of the state estimate. Figure 6 visualizes the way the matrices of the evaluation framework are derived from the environment. The C matrix is derived from the object position in the nominal path. It is assumed to be noiseless. B matrix then is the noisy or noiseless state transition between two time steps. The observation estimate is generated by multiplying the state with the observation matrix. In the figure 6 it is represented as a set of grid cells, where the higher probability areas are colored in darker colors.

For each time step, the evaluation requires dividing the path data into two plane projections. In figure 7, a visualization of this is shown (reduced state spaces for evaluating a policy realization).

While evaluating the paths from the policy realizations against the nominal path from the general object context

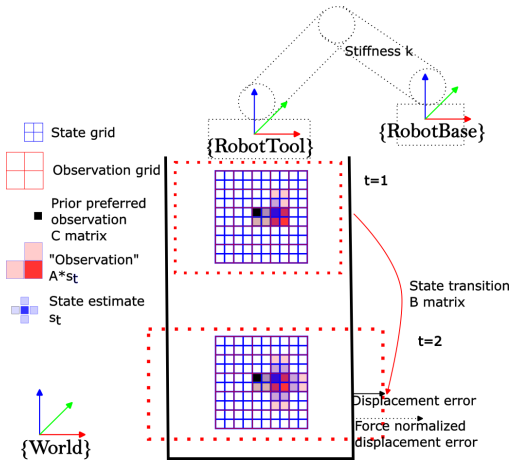

Fig. 6: Visualization of how to derive matrices used in evaluation from the environment

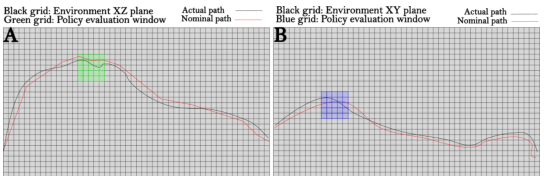

Fig. 7: Visualization of policy discretization

policy, the window is slid along the nominal path from start to end. The full grid can include large areas where no data points are present. Focusing on smaller windows reduces substantially the state space and computational costs.

B. State and Observation Space Resolutions

We considered also the possible difference in state and observation resolutions. The basic idea is that the observations that can be acquired from the environment are always less accurate than the states. As an example, we can select a state space for the evaluation as 10x10 grid, and the observation space as 8x8 grid. The spaces cover the same area of the task space, however, the different resolutions create a situation, where the observation does not give unambiguously the state of the object.

C. Simulation tests

Simulator experiments were performed for three tasks (peg-in-hole, screwing and connector insertion) using CoppeliaSim simulator with the PyRep [39] interface to control the environment. Each task was expressed in object context, resulting to the object context policy from the DRL, and with robots for evaluating the robot context policy execution. Each policy execution is simulated with three robots models: Franka Emika Panda, UR5 and Kuka LBR Iiwa 14 R820. For robot context simulations, Open Dynamics Engine (ODE) was used.

The simulator was used in two stages: 1) training the RL agent to learn the object context policy and 2) executing test runs. In between, the robot context policies were evaluated. During training, the simulator was simply used to estimate the motions of the objects based on movement actions in task space, not considering the physics of the environment. In our model, the training takes place under the assumption that moving objects are in motion because they are manipulated by a robot. This simplifies and speeds up the calculations required during the training phase. In the evaluation of the policy execution, the nominal path prescribed by the general object context policy was used as a base for the robot end effector positioning. The nominal object path was transformed into robot tool paths by adjusting the grasp and tool correction frames of each robot to align the object frame with the grasp and tool correction frames.

Inverse kinematic solutions were calculated using the library PyRep [39] and its *get_path* function. With the robot context actions, the tasks were tested in the simulator. The stiffness was taken as a computational value from the related admittance controller. The evaluation results show (8) that with Panda robot better path following results are achieved. The combination of positional accuracy and stiffness as force normalized displacement errors leads to the nominal path being followed more closely.

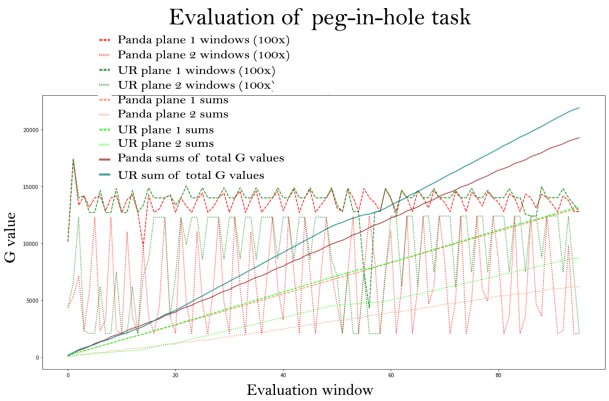

Fig. 8: G values from robot policy evaluation for peg-in-hole task

For the policy evaluation the following values were used: Franka Panda: Stiffness 1500 n/m, repeatability ± 0.1 mm, UR5: Stiffness 2000 n/m, repeatability ± 0.1 mm, Kuka LBR Iiwa 14 R820: Stiffness 2500 n/m, repeatability ± 0.15 mm Stiffness values were selected as the median value within the range of potential values.

V. Discussion

Instead of comparing robot context policies for different robots, we could compare the same robot with different controlled stiffness, or with different locations of objects. Additionally, the stiffness models and accuracy could be extended from current static values to configuration dependent ones.

In our evaluation process, the expected free energy is calculated using two state factors in several windows of the state space. These values are gathered both individually, and summed together. We can extend the validity of the policy

realization with a simple threshold value. At any step, if the expected free energy goes over this threshold value, the policy realization would be deemed unsuitable for the task.

VI. Conclusion

A solution to tackle robotic production agility by automated task planning and resource allocation was proposed. A two stage approach was taken, with Deep Reinforcement Learning to create task policies in the object context, based on product CAD models, and with modified Active Inference Framework to estimate the quality of task execution in the robot context based on robot accuracy models. The solution was tested successfully in a simulation environment. Future steps will be taken to perform tests in real robot systems.

References

[1] D. Baldwin, "Engineer-to-order: A practical guide." https://www.acumatica.com/blog/engineer-to-order, 2024. Accessed: 2026-01-09.

[2] F. El Kalach, I. Yousif, T. Wuest, A. Sheth, and R. Harik, "Cognitive manufacturing: definition and current trends," *Journal of Intelligent Manufacturing*, vol. 36, no. 6, pp. 3695–3715, 2025.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[4] B. Singh, R. Kumar, and V. P. Singh, "Reinforcement learning in robotic applications: a comprehensive survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 945–990, 2022.

[5] K. Friston, J. Kilner, and L. Harrison, "A free energy principle for the brain," *Journal of Physiology-Paris*, vol. 100, no. 1, pp. 70–87, 2006. Theoretical and Computational Neuroscience: Understanding Brain Functions.

[6] T. Parr, G. Pezzulo, and K. J. Friston, *Active Inference: The Free Energy Principle in Mind, Brain, and Behavior*. The MIT Press, Mar. 2022.

[7] M. Pantano, T. Eiband, and D. Lee, "Capability-based frameworks for industrial robot skills: a survey," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pp. 2355–2362, 2022.

[8] L. Heuss, C. Gonnermann, and R. G., "An extendable framework for intelligent and easily configurable skills-based industrial robot applications," *The International Journal of Advanced Manufacturing Technology*, vol. 120, p. 6269–6285, June 2022.

[9] M. T. Mason, "Toward robotic manipulation," *Proceedings of Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 1–28, May 2018.

[10] X. Zhang, X. Wang, X. Liu, W. Wang, X. Fan, and D. Zhao, "Task-agnostic learning to accomplish new tasks," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 18, no. 1, pp. 113–127, 2026.

[11] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, 2019.

[12] Z. Hou, Z. Li, C. Hsu, K. Zhang, and J. Xu, "Fuzzy logic-driven variable time-scale prediction-based reinforcement learning for robotic multiple peg-in-hole assembly," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 218–229, 2022.

[13] S. Zhang, Y. Wang, S. Liang, H. Han, Z. Jiang, and M. Zhang, "Research on Robotic Peg-in-Hole Assembly Method Based on Variable Admittance," *Applied Sciences*, vol. 15, no. 4, 2025.

[14] Y. Zang, P. Wang, F. Zha, W. Guo, S. Ruan, and L. Sun, "Geometric-feature representation based pre-training method for reinforcement learning of peg-in-hole tasks," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3478–3485, 2023.

[15] C. Nam, S. H. Cheong, J. Lee, D. H. Kim, and C. Kim, "Fast and resilient manipulation planning for object retrieval in cluttered and confined environments," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1539–1552, 2021.

[16] T. Ren, G. Chalvatzaki, and J. Peters, "Extended tree search for robot task and motion planning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12048–12055, 2024.

[17] H. Aziz, A. Pal, A. Pourmiri, F. Ramezani, and B. Sims, "Task Allocation Using a Team of Robots," *Current Robotics Reports*, vol. 3, no. 4, pp. 227–238, 2022.

[18] B. Fu, W. Smith, D. M. Rizzo, M. Castanier, M. Ghaffari, and K. Barton, "Robust task scheduling for heterogeneous robot teams under capability uncertainty," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1087–1105, 2023.

[19] Z. Li, N. Shi, L. Zhao, and M. Zhang, "Deep reinforcement learning path planning and task allocation for multi-robot collaboration," *Alexandria Engineering Journal*, vol. 109, pp. 408–423, 2024.

[20] J. Orr and A. Dutta, "Multi-Agent Deep Reinforcement Learning for Multi-Robot Applications: A Survey," *Sensors*, vol. 23, no. 7, 2023.

[21] G. Cao and J. Bai, "Multi-agent deep reinforcement learning-based robotic arm assembly research," *PloS one*, vol. 20, no. 2, p. e0311550, 2025.

[22] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pp. 590–595, 2019.

[23] P. Forte, A. Mannucci, H. Andreasson, and F. Pecora, "Online task assignment and coordination in multi-robot fleets," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4584–4591, 2021.

[24] X. Qin, Z. Liao, C. Liu, and Z. Xiong, "Online task allocation and scheduling in multi-manipulator system considering collision constraints and unknown tasks," *Robotics and Computer-Integrated Manufacturing*, vol. 90, p. 102808, 2024.

[25] A. Pal, A. Chauhan, and M. Baranwal, "Together we rise: Optimizing real-time multi-robot task allocation using coordinated heterogeneous plays," *CoRR*, vol. abs/2502.16079, 2025.

[26] R. Murray-Smith, J. H. Williamson, and S. Stein, "Active inference and human–computer interaction," *ACM Trans. Comput.-Hum. Interact.*, vol. 32, Dec. 2025.

[27] R. Smith, K. J. Friston, and C. J. Whyte, "A step-by-step tutorial on active inference and its application to empirical data," *Journal of Mathematical Psychology*, vol. 107, p. 102632, 2022.

[28] D. Braziunas, "POMDP solution methods: a survey," tech. rep., Department of Computer Science, University of Toronto, 2003.

[29] P. Lanillos, C. Meo, C. Pezzato, A. A. Meera, M. Baioumy, W. Ohata, A. Tschantz, B. Millidge, M. Wisse, C. L. Buckley, and J. Tani, "Active inference in robotics and artificial agents: Survey and challenges," *arXiv preprint arXiv:2112.01871v1*, 2021.

[30] M. T. Koudahl and B. de Vries, "Batman: Bayesian target modelling for active inference," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3852–3856, 2020.

[31] L. Pio-Lopez, A. Nizard, K. Friston, and G. Pezzulo, "Active inference and robot control: a case study," *Journal of The Royal Society Interface*, vol. 13, p. 20160616, 09 2016.

[32] P. Mazzaglia, T. Verbelen, and B. Dhoedt, "Contrastive active inference," in *Advances in Neural Information Processing Systems* (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), vol. 34, pp. 13870–13882, Curran Associates, Inc., 2021.

[33] A. Zelenov and V. Krylov, "Deep active inference in control tasks," in *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pp. 1–3, 2021.

[34] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. www.coppeliarobotics.com.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[36] R. Murray-Smith, J. H. Williamson, and S. Stein, "Active inference and human–computer interaction," *ACM Trans. Comput.-Hum. Interact.*, vol. 32, Dec. 2025.

[37] A. Rosales and T. Heikkilä, "Analysis and design of direct force control for robots in contact with uneven surfaces," *Applied Sciences*, vol. 13, no. 12, 2023.

[38] D. Marković, T. Goschke, and S. J. Kiebel, "Meta-control of the exploration-exploitation dilemma emerges from probabilistic inference over a hierarchy of time scales," *Cognitive, Affective, & Behavioral Neuroscience*, vol. 21, no. 3, pp. 509–533, 2021.

[39] S. James, M. Freese, and A. J. Davison, "Pyrep: Bringing v-rep to deep robot learning," *arXiv preprint arXiv:1906.11176*, 2019.