2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Run-Time Optimization of Overall Energy Consumption in Lightweight Collaborative Arms for Repetitive Tasks

Ahmadreza Zarei[1], Sajad Shahsavari[1], Juha Plosila[1], and Hashem Haghbayan[1]

Abstract— **Lightweight industrial robots are increasingly deployed alongside humans to perform diverse and intelligent industrial tasks. A major concern with these robots is energy efficiency, driven by rising operational costs and environmental impacts. A growing contributor to energy use is the heavy computational workload of their electronic components. Although motion configurations and computational load are often interdependent, current state-of-the-art energy optimization methods tend to address them separately, focusing on individual consumption. In this work, we demonstrate that computational energy is comparable to mechanical energy and show how their dependency affects overall consumption in a *Franka Emika Panda* robot equipped with a multi-core processing system and two depth cameras. Building on this understanding, we propose a Bayesian approach for the joint optimization of mechanical motion and computational frequency in a robotic arm. Experiments show that the proposed method enables the *Franka* arm to reduce energy use by 3.7% in pick-and-place tasks and 6.2% in sorting tasks, compared to methods that optimize locomotion and computation separately.**

I. INTRODUCTION

Optimizing energy efficiency is an important research area for industrial robots [1], [2] due to their growing large-scale deployment in modern manufacturing systems and the need for energy sustainability [3]. As a result, significant efforts have been made to reduce the energy consumption of robots through more efficient hardware design [4] and energy-aware software optimization [5].

In large, heavy-duty industrial robots, most energy consumption comes from the mechanical system, while ancillary operations such as computation, communication, and sensing contribute only a small fraction. However, with the rise of lightweight industrial robots, the energy consumed by these ancillary operations has become comparable to that of the mechanical system [1]. This trend is further amplified by the increasing complexity of onboard operations, including advanced computation [6], high-resolution sensing [7], and networked communication [8], highlighting the need to incorporate these factors into energy-efficiency optimization.

State-of-the-art approaches often model ancillary energy consumption as a constant offset, relying on separate local energy management techniques [9]. Consequently, energy optimization efforts typically focus on minimizing mechanical energy, while neglecting its interaction with other subsystems such as computation and sensing. For instance, some works manipulate waiting times between robot operations [10], or find the optimal motion configuration [11], [5], or optimize joint trajectories [12], under the assumption that ancillary energy consumption is static. This assumption ignores the fact that robot motion configurations influence the energy demand of other subsystems. For example, if a sensor is mounted on the end-effector, its velocity and acceleration directly influence the quality and amount of data captured, and consequently the computational load required for processing.

In this paper, we provide a motivational example that highlights the dependency between motion-related decision-making and computational workload in a lightweight Franka Emika arm robot. The robot performs different tasks such as pick-and-place, with data processing executed on onboard computing units running applications such as obstacle avoidance, object detection, and grasp pose estimation. For each task, a minimum accuracy level is required, represented by a target frame rate for processing camera images. This requirement can be met by slowing the end-effector motion and/or increasing CPU frequency, both of which affect the robot's total energy consumption.

Our experiments reveal that, for the selected arm robot and applications workload, computational energy consumption is comparable to mechanical energy. This creates a trade-off between CPU frequency (computational performance) and motion parameters. Based on these observations, we propose a Bayesian optimization approach to jointly optimize the computational and mechanical components of the robot at runtime. Our method simultaneously tunes the arm's velocity and acceleration while dynamically adjusting CPU frequency to minimize total energy consumption.

Experimental results demonstrate that the proposed joint optimization strategy significantly reduces overall energy consumption compared to state-of-the-art approaches that optimize mechanical energy in isolation, without considering its impact on computational components. The main contributions of this paper are listed as follows:

1) We provide a motivational example demonstrating that computational energy can be comparable to mechanical energy in lightweight industrial robots, motivating the need for joint optimization.
2) We propose a Bayesian-based optimization framework that simultaneously tunes motion configurations and computational performance to minimize total energy.
3) We implement and evaluate the proposed approach on a lightweight arm robot, comparing it with state-of-the-art methods that treat mechanical and computational

[1]Ahmadreza Zarei, Sajad Shahsavari, Juha Plosila, and Hashem Haghbayan are with the Autonomous Systems Laboratory (ASL), Department of Computing, University of Turku, 20014 Turku, Finland (e-mail: arzare@utu.fi, sajsha@utu.fi, juplos@utu.fi, mohhag@utu.fi).

components separately.

II. Related Work

The related work in this research can be divided into three categories: energy-aware control of industrial robots by optimizing mechanical energy, energy-aware computational resource management, and co-management of both parts.

Prior works have explored various methods to reduce the mechanical energy consumption of industrial and lightweight industrial arms, which can be divided into hardware and software-based approaches [13], [2]. Hardware methods involve structural or component changes, such as structural design [4], [14] and adding energy-recovery units [15]. Software methods, on the other hand, reduce energy consumption without hardware modifications by tuning parameters such as the velocity and acceleration [11], [16], [5], trajectory planning [12], [17], and operation scheduling [18]. Most of these works focus on optimizing mechanical energy consumption as the main source of energy, oblivious to ancillary operations such as computational units and sensors in the optimization method. In fact, in these methods the optimization of electrical energy is considered to be done separately based on the application executing on the processing units and types of sensors used in the robot.

Considering computing energy management, several research efforts have focused solely on computation, depending on the application and available control of computing actuators. Dynamic voltage and frequency scaling (DVFS) is one of the most widely used techniques for controlling computation [19]. DVFS adjusts processor voltage and frequency according to workload, making it effective in robotics applications where large volumes of sensor data must be processed continuously [20]. Other studies have explored energy optimization in heterogeneous multiprocessor systems through workload-based power control [21] and efficient CPU–GPU task allocation [22]. Only a limited number of works have addressed computational energy in robotic arms. For instance, [1] analyzed four manipulators (UR5e, UR10e, Franka Emika FR3, Kinova Gen3) and showed that 60–90% of the total energy is consumed by the electronic components of the control box. Similarly, [9] proposed an energy consumption model for lightweight industrial robots that includes electronic components such as the arm controller, but treats them as constant. All of these approaches optimize mechanical and computational parts in isolation, overlooking the fact that motion configurations affect the energy demand of other subsystems. Mechanical parameters and computational workload should therefore be jointly optimized for meaningful energy savings.

Although there is no direct research on co-managing mechanical and computational energy consumption for industrial robots, a few studies have addressed this issue in mobile robots. For example, in [23], [24] the authors investigated the co-dependence of mechanical and computational units and proposed a runtime optimization strategy that tunes the velocity of a mobile robot equipped with an event-based camera and Jetson TX2 processing unit by adjusting

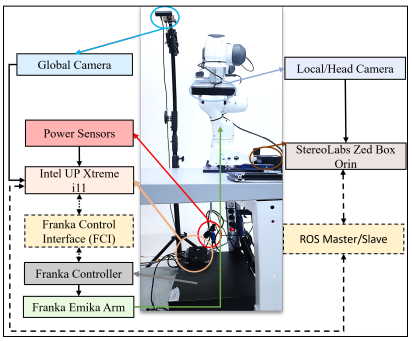


Fig. 1: Experimental hardware setup.

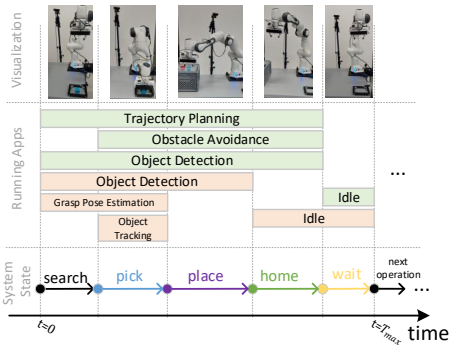


Fig. 2: Overview of the working scenario. Application colors denote the host processing board.

CPU frequency to preserve battery life. While such co-management of CPU frequency and velocity significantly reduces energy consumption in mobile robots, the structure and operation of industrial robots differ, including the types of sensors and their stationary operation during diverse tasks. These differences demand new investigations into the co-dependence of energy consumption between mechanical and computational parts, which is the focus of this paper.

III. Working Scenario and Motivation

A. Working Scenario

The reference robotic arm considered in this study is depicted in Fig. 1 and can be divided into three main components: (1) the mechanical arm manipulator along with its low-level controller, (2) the high-level computational unit, and (3) the intelligence software stack. Each of them is described below.

Mechanical Arm Manipulator: This part consists of the physical structure of the arm and its several electric motors enabling spatial motion as well as an end-effector for object grasping. In practice, this is a Franka Emika Panda arm manipulator that is equipped with 7 torque-controlled brushless DC electric motors on its joints. The arm supports a payload of up to 3 kg and offers a reach of 855 mm, and achieves sub-millimeter accuracy, making it suitable for delicate and repetitive tasks. The low-level controller, i.e., the Franka Control Interface (FCI) drives the arm motors according to the externally commanded trajectory. This last part is excluded from the energy-optimization analysis in our considered scenario because its consumption is inherently static and not subject to control. However, for comprehensiveness,

we report the energy consumption and improvements for both cases, including and excluding this part in Section V.

Computing Unit: The computing unit includes two embedded computing boards that are connected to two stereo RGB cameras and provide processing capacity for perception, planning, and control. A ZED2 camera mounted on the arm's end-effector as local vision sensor is connected to a ZED Box Orin[1]. An OAK-D camera is used as a global vision sensor and is connected to an UP Xtreme i11[2] board. The two boards exchange vision data through a Robot Operating System (ROS) channel to enable coordinated perception and planning. Several power sensors are also installed and interfaced with the UP Xtreme i11 to measure the instantaneous power consumption of the arm and computing boards. All sensors are powered directly by the boards and do not require additional power supply.

Intelligence Software Stack: The software stack for the autonomous and intelligent operation consists of several running applications for processing the sensory data. Specifically, the applications are i) object detection [25], ii) object tracking [26], iii) obstacle avoidance [27], iv) trajectory planning [28], and v) grasp pose estimation [29]. In the considered scenario, the software stack manages the execution of time-constrained repetitive pick-and-place operations. The overview of this operation is depicted in Fig. 2 and is performed in 5 consecutive *states*, namely, search, pick, place, home, and wait; in the search state, the system recognizes the object, estimates the optimal grasp pose, and plans the trajectory to a pre-grasp configuration; in the pick state, the arm moves to the estimated grasp pose and grasps the object; in the place state, it moves to the specified placement pose and releases the object; then, in the home state the arm moves back to its home position and finally waits there for the next operation in the wait state. The running applications in the software stack vary in different states (in either halt-and-restart fashion or go-to-idle) and result in a varying computation power consumption. Moreover, as elaborated in [30], the computing power is also affected by the motion of the robot by variable Frames per Second (FPS) settings on the vision sensors. Indeed, when operations are executed at higher speed, a higher FPS is required to maintain the tracking accuracy and guarantee safety [30]. On the contrary, when the operation is slow, the FPS can be set to a lower value, reducing the computational demands. This is implemented in our setup by means of a software middleware between the sensors and running applications that filters out a portion of the incoming frames based on the actual speed of the arm's end-effector.

We consider two control knobs to manage the energy consumption of the mechanical and computational parts of the system. For the mechanical part, we utilize the commonly used motion velocity and acceleration of the arm motors and configure an upper bound as their maximum values. For the computing part, the Dynamic Voltage/Frequency Scaling (DVFS) level of the two boards are managed by a runtime controller that utilizes the OS-level commands for dynamic computing resource allocation. Indeed, the two knobs are among the main contributors to the energy consumption of the two mechanical and computational parts [5], [23].

[1]https://www.stereolabs.com/en-fi/store/products/zed-box-orin

[2]https://www.aaeon.com/en/product/list/aaeon-up-developer-boards

B. Motivational Discussion

Recent methods for energy-optimal control in the arm robots have been mainly focused on minimizing the locomotion energy, i.e., the energy spent by the electric motors to control the motion of the arm [9], [17], [16], [11], [12]. In this subsection, by presenting a motivational example, we argue that the cost of computation should also be taken into account as it considerably affects the overall energy consumption and the optimal configuration point. First, we analyze the share of energy between the two parts by performing design-time experiments. In Fig. 3(a), we performed operation-wise configuration, meaning that the velocity and acceleration are set to the same value and are kept fixed for the whole operation. The figure shows the energy consumption of the mechanical and computational parts of the arm for various velocity and acceleration values, averaged over five operations (each lasting $T_{max} = 25$ seconds) to mitigate the effect of measurement and actuation noise. In all of these motion configurations, the DVFS configuration is set to the minimum value that satisfies the FPS requirements. The velocity and acceleration were varied from 10% to 100% of the arm's nominal maximum for each operation. Note that the arm starts from a stationary position, and the instantaneous speed is bounded by the chosen motion configuration and also varies across the arm joints. The power consumption of the mechanical arm and computing units was measured and integrated over time to obtain the energy consumption. It can be seen in Fig. 3(a) that computational energy is comparable to the energy consumed in the mechanical part in all velocity values. More importantly, the optimal point for the velocity and acceleration changes from 20% (for when considering the mechanical energy only) to 50% (for when the computational energy is taken into account). We hypothesize that this change is due to the intricate interplay of the computing power and operation duration. Indeed, at higher velocities, the time interval during which higher DVFS level is required, is shorter, and despite higher computing power, this leads to a decrease in overall energy.

We further analyzed the effect of motion configuration within a single state of the pick-and-place operation due to its more stable power profile. The home state was selected because it exhibits stable power behavior and minimal environmental influence. We fixed the velocity and acceleration in all other states to the design-time optimal value found from Fig. 3(a), i.e., 50%, and varied the velocity only in the home state. Fig. 3(b) shows the results of this state-wise operation, revealing the same trend as in the operation-wise configuration. However, this time the optimal point for the velocity within the home state is significantly different from those global operation-wise configurations, in both cases when considering mechanical only, or the overall energy.

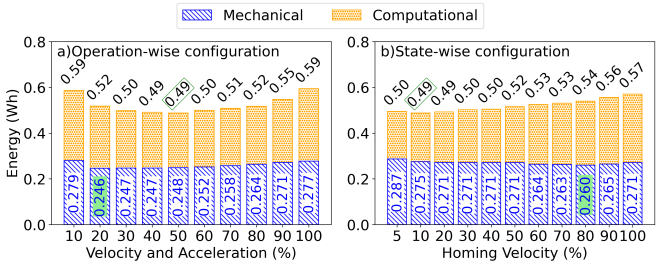


Fig. 3: Mechanical and computational energy across motion configurations: (a) operation-wise settings; (b) state-wise (homing velocity) settings.

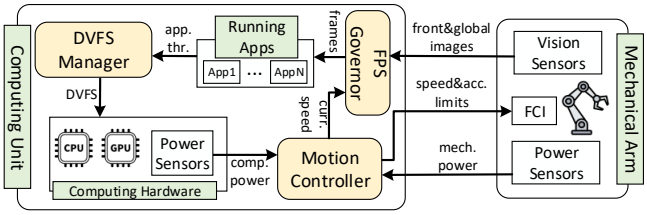


Fig. 4: Overview of the proposed controllers for the mechanical motion and computing DVFS management.

This shows the necessity of considering the overall energy in state-level motion configuration.

In conclusion, we demonstrated that the computational energy not only contributes significantly to the overall energy consumption of the system, but can change the inherent dynamics of the system and cause noticeable impact on the optimal motion configurations. An energy-aware operational controller should take into account this contribution and dynamic changes at runtime, as it varies over system operation states. Moreover, the situation is exacerbated when considering other a priori-unknown environmental factors such as object position, destination position, and computational workload that also contribute to determining the overall energy.

IV. METHODOLOGY

We propose a runtime energy-aware control strategy to coordinately manage the configuration knobs in the two parts of the robotic system, i.e., motion of the mechanical arm and the computing resources of the processing units. For the mechanical part, the arm motion velocity and acceleration are optimized by an online Bayesian optimization method that identifies the optimal configurations across various system states. This method learns a surrogate model over time to estimate the overall energy given the arm motion configuration, scenario-specific context information and historical data. The computing resource manager, on the other hand, continuously adjusts the DVFS configuration of the processing board based on the current performance (throughput) level of the running applications. The objective of this manager is to consume a minimum amount of computing resources while maintaining the required throughput for the running applications, i.e., the number of processed frames per second. The two controllers run in tandem with each other through the variable FPS of the vision sensory data, as motivated in the discussion presented in Section III-B. The motion controller needs to account for both mechanical and computational energy when optimizing the mechanical motion, and the computing part acts according to the motion of the robot and adjusts its performance and energy consumption. The overview of the system components and their interface with the proposed controller setup is presented in Fig. 4. The motion controller of the robotic arm receives the mechanical and computing power measurements and outputs the Bayesian optimization results, i.e., the joint velocity and acceleration configurations, which are executed by the arm driver (FCI). The DVFS manager for the computing part receives the running applications' throughput data and adjusts the frequency levels of the processing board accordingly. The frontal and global images from the stereo cameras are consumed by the running vision applications through the FPS governor, which adjusts the input rate according to the speed of the arm end-effector. In the following, we describe our control methods for the two parts.

A. Motion Controller

We employ a modified version of the meta-algorithm proposed in [5] and couple it with a constrained context-aware Bayesian optimization method to perform energy-aware motion actuation on the mechanical arm. The adaptive probabilistic nature of this method, along with its ability for uncertainty quantification and its proven sampling efficiency compared to data-driven methods, renders this choice suitable for arm manipulation operations with varying and a priori unknown task types and environments.

Concretely, this optimization outputs the two variables for adjusting the velocity and acceleration of the arm joints relative to their individual nominal values. This output can be denoted as $\mathbf{x} = \{(v_s, a_s) | v_s, a_s \in [0,1], s \in \mathcal{S}\}$, where v_s and a_s denote the proportional velocity and acceleration limits for the state s, and $\mathcal{S}$ is the set of operational states. The actual value of the maximum allowable velocity and acceleration for the i th joint in state s is then computed as $\mathbf{c}_s^i = \left[\begin{smallmatrix} v_s \\ a_s \end{smallmatrix}\right] \cdot \left[\begin{smallmatrix} v_{max}^i \\ a_{max}^i \end{smallmatrix}\right]$, where v_{max}^i and a_{max}^i are the nominal maximum speed and acceleration for the i th joint motor, respectively. We formalize the optimization problem as:

$$\mathbf{x}^* = \underset{\mathbf{x} \in P}{\operatorname{argmin}}\ E(\mathbf{x}, \mathbf{c}) \quad \text{s.t. } T(\mathbf{x}, \mathbf{c}) \leq T_{\max},\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{2\times|\mathcal{S}|}$ is the proportional velocity and acceleration configurations across operation states, $\mathbf{c} \in \mathbb{R}^m$ is the context vector containing information such as initial position of the detected object, its target position and weight, $E(\mathbf{x}, \mathbf{c})$ is the overall energy consumption when the given configuration and context information are applied, T_{max} is the operation time constraint, P is the feasible set for the configurations, and $\mathbf{l}$ and $\mathbf{u}$ are the lower and upper bounds for the configuration, respectively.

We integrate the time constraint into the objective cost function by an exponentially weighted penalty and, in the Bayesian optimization setup, assume this cost as noisy observations of a function $f : \mathbb{R}^{2\times|\mathcal{S}|} \times \mathbb{R}^m \to \mathbb{R}$, as:

$$y = E(\mathbf{x}, \mathbf{c}) + \exp(\eta) \cdot t_{over} \\ = f(\mathbf{x}, \mathbf{c}) + \varepsilon \quad (2)$$

where η is the weight for overtime penalty, $t_{over} = max\{0, t - T_{max}\}$ with t being the operation time, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian measurement noise. This generic function is approximated iteratively by a Gaussian Process (GP) surrogate model parameterized by its mean function $\mu(\mathbf{x})$ and covariance kernel $k(\mathbf{x}, \mathbf{x}')$. We consider each pick-and-place operation as one optimization iteration. To do so, we translate the power sensory data for an operation to iteration-level overall energy consumption e_i and collect it along with the operation time t_i, motion configuration $\mathbf{x}_i$ and context vector $\mathbf{c}_i$ to form the accumulating training dataset.

$$\mathcal{D}_{1:n} = \{(\mathbf{x}_i, \mathbf{c}_i, y_i)\}_{i=1}^{n} \quad (3)$$

where y_i is the weighted cost function in Eq. 2. The posterior parameters of the GP surrogate model are then identified as:

$$\mu_n(\mathbf{x}) = \mu_0(\mathbf{x}) + \mathbf{k}(\mathbf{x})^\top (\mathbf{K} + \sigma^2 I)^{-1} \mathbf{z} \quad (4)$$

$$\sigma_n^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\top (\mathbf{K} + \sigma^2 I)^{-1} \mathbf{k}(\mathbf{x}) \quad (5)$$

where μ_0 is the prior mean function, k is the squared exponential covariance kernel [31], $\mathbf{k}(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_t)]$, $\mathbf{K}$ is the Gram matrix: $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{z} = [z_1, \ldots, z_n], z_i = y_i - \mu_0(\mathbf{x}_i)$. The next evaluation configuration $\mathbf{x}_{n+1}$ is then selected to maximize the Expected Improvement (EI) acquisition function, which at the same time yields the minimum objective cost, as:

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x} \in \mathcal{P}} \mathbb{E}\left[\max\{0, y^* - y\}) | y \sim \mathcal{N}(\mu_n(\mathbf{x}), \sigma_n^2(\mathbf{x})\right] \quad (6)$$

where y^* is the best known cost so far. The motion optimization procedure is summarized in Algorithm 1. Inputs include the GP prior mean and covariance functions μ_0 and k, time constraint T_{max}, over time penalty weight η and parameter bounds $\mathbf{l}$ and $\mathbf{u}$. Each optimization iteration (Lines 3-12) starts with identification of search state (Line 3). Then, current context information is estimated by the object detection application (Line 4) and Bayesian optimization is performed by the previously collected data points based on Eqs. 4, 5 (Line 5). Then, optimal motion configuration is selected based on Eq. 6 (Line 6). The configuration is applied in Line 7, and the overall energy consumption is observed. The objective cost is computed in Line 8, while the cumulative dataset $\mathcal{D}$ and the best-so-far cost y^* are updated in Lines 9 and 11, respectively. Finally, the controller waits for the next operation to start.

Algorithm 1 Motion Controller Optimization Loop

Inputs:
- μ_0 ▷ GP prior mean
- k ▷ GP covariance function
- $T_{\max}$ ▷ Maximum allowable time duration for the task
- η ▷ Exponential weight for overtime
- $\mathbf{l}, \mathbf{u}$ ▷ Configuration lower and upper bounds

1: $\mathcal{D} \leftarrow \emptyset$ ▷ Training dataset initialization
2: **while** `true` **do**
3: WaitForSearch()
4: $ctx \leftarrow$ ReadContext()
5: $\mu_n, \sigma_n^2 =$ GetGaussianPosteriors$(\mu_0, k, \mathcal{D})$ ▷ Eqs. 4,5
6: $v_{curr}, a_{curr} =$ GetArgmaxEI$(\mu_n, \sigma_n^2, k, ctx, y^*, \mathbf{l}, \mathbf{u})$ ▷ Eq. 6
7: $e, t \leftarrow$ RunIteration(v_{curr}, a_{curr}) ▷ Pick&place operation
8: $y \leftarrow e + \exp(\eta) \cdot \max(0, t - T_{max})$
9: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(v_{curr}, a_{curr}, y)\}$
10: **if** $y < y^*$ **then**
11: $y^* \leftarrow y$
12: WaitForNextOperation()

B. DVFS Manager

We adopt the resource management policy proposed in [19] to perform runtime DVFS adjustments in order to minimize computing energy while maintaining the required throughput for the running applications, i.e., the number of processed FPS. We utilize various computing clusters within one processing board, e.g., Arm Cortex CPU cluster and Ampere GPU in ZED Box Orin. We map the running applications onto predefined clusters depending on their characteristics. The DVFS control knob can be adjusted in the cluster-level to trade-off computing power with execution performance. We aim to minimize the computing power, that results in minimized computing energy along the pick-and-place operation, by identifying the minimum level of DVFS of each cluster that still satisfies the throughput requirements of the applications running on that cluster. As discussed in [19], [32], the application throughput has a linear relationship with the CPU/GPU frequency level. Similar to [23], to determine the frequency level of each cluster, we consider its most critical application, defined as the application with the minimum throughput level. Given the current frequency level f_{curr}, the current throughput of the most critical application th_{curr}, and its required throughput th_{new}, we determine the new frequency level for each computing cluster as follows:

$$th_{new} = \frac{f_{new}}{f_{curr}} \cdot th_{curr} \quad (7)$$

V. EXPERIMENTS

A. Setup

Our evaluation setup is the one described in Section III-A. We have implemented our proposed method in ROS architecture, utilized the MoveIt motion planning framework [33], and RRTConnect trajectory planner [28]. The applied velocity and acceleration limits for the mechanical motion are in the continuous space of $[0.01, 1]$, whereas the DVFS configuration of the computational part is in the discrete space ranging from 100MHz to 2.0GHz for ZED BOX ORIN, and 400MHz to 4.4GHz for UP Xtreme i11. The value of $T_{\max}$ is different for each task and operation, and η is set to 1.1. For the vision applications, the processing rate is set to be at least 1 FPS when the end-effector was stationary, increasing linearly up to 30 FPS at maximum end-effector speed. The static power consumption of the FCI controller, with the arm disconnected as in [1], was measured at 69.8 W, which we refer to as the *static power*. Results are reported both excluding and including this static power.

Tasks: We perform our energy-aware control method in two distinct tasks: 1) pick-and-place in which one single object is picked from and placed to a certain position, and 2) sorting

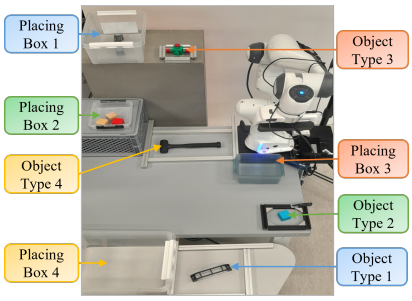


Fig. 5: Experimental setup for the robotic sorting task.

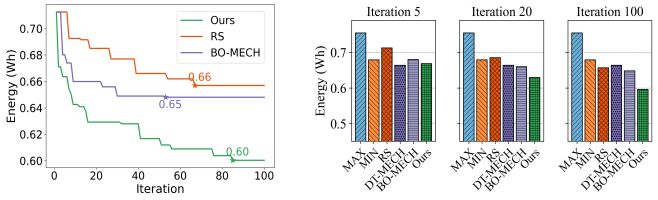
where several objects with different shapes, weights and picking and placing positions are sorted. Fig. 2 represents the pick-and-place operation and Fig. 5 illustrates the setup for the sorting task.

Baselines: To evaluate the performance of our proposed approach, we compared it against five baselines: (i) **MAX**, where velocity and acceleration are set to the manufacturer-specified maximum values; (ii) **MIN**, where velocity and acceleration are set to the minimum values that still ensure task completion within the maximum allowable duration; (iii) **RS** (Random Search) [5], [34], where a random search method is embedded in the meta-algorithm proposed in [5] where motion configurations are randomly selected to explore the allowable speed values; (iv) **DT-MECH** (Design-time Mechanical Optimization) [11], where the configuration is selected based on whether the arm carries a payload or not, and optimal velocity/acceleration are obtained by modeling the energy consumption as a function of motion duration; and (v) **BO-MECH** (Bayesian Optimization – Mechanical only) [5], where Bayesian Optimization is applied considering only the mechanical energy. The MIN, MAX, and DT-MECH baselines, inspired by [23] are static methods, whereas the rest runtime optimizers, which we refer to as optimization-based baselines.

B. Results

Pick-and-place: As a first step, we evaluate the performance of methods on a representative pick-and-place task, in which a single object of approximately 200 g was manipulated, with the task duration constrained by $T_{\max} = 30$ s. For the proposed method, as well as the optimization-based baselines RS and BO-MECH, the algorithms were executed for 100 iterations, all starting from the same random initial configuration to be comparable. Fig. 6a presents the cumulative minimum energy for these baselines versus our method, demonstrating the convergence of the algorithms to optimal configurations. As shown, RS and BO-MECH achieve their best solutions at iterations 67 and 53, respectively, and do not improve further up to iteration 100. In contrast, the proposed algorithm reaches its best solution at iteration 92, while it has already outperformed the best of RS and BO-MECH by iteration 12.

Fig. 6b compares the overall energy consumption of the baselines versus our method at optimization iterations 5, 20, and 100. For each iteration, the reported energy corresponds to the best motion configuration identified up to that point,



(a) (b)

Fig. 6: Energy optimization results in pick-and-place: (a) cumulative minimum energy across iterations, and (b) comparison with all baselines.

averaged over ten repeated executions to reduce the impact of measurement noise. The results show that the proposed method outperforms MAX, MIN, RS, and BO-MECH after only 5 iterations and surpasses all baselines after 20 iterations. This rapid improvement highlights the data efficiency of the proposed approach, especially when contrasted with other data-efficient optimization methods such as [35], which typically require substantially more samples to converge to energy-optimal solutions.

The velocity and acceleration for each method, together with the average end-effector speed and the average CPU frequency of the computing units during each motion state, are summarized in Table I. The CPU frequency was computed in two steps: first, the frequency of all CPU cores was averaged for each computing unit over the duration of a given motion state; second, the averages from the two units were combined to obtain the final reported value. For the searching and waiting states, the average CPU frequency was about 0.35 GHz and 0.25 GHz, respectively.

To further validate robustness, we additionally tested the proposed algorithm and all baselines on 20 different pick-and-place tasks with object weights ranging from 100 g to the robot's maximum payload (3 kg), as well as varying picking, placing, and waiting positions. The improvements achieved by our method across all baselines are reported in Table II.

TABLE I: Velocity (v), acceleration (a), average end-effector speed (s) in m/s, and average frequency (f) in GHz across motion states for the proposed method and baselines.

Method	Picking				Placing				Homing			
	v	a	s	f	v	a	s	f	v	a	s	f
MAX	1.00	1.00	0.43	1.7	1.00	1.00	0.48	1.5	1.00	1.00	0.52	1.3
MIN	0.12	0.12	0.12	0.5	0.12	0.12	0.14	0.4	0.12	0.12	0.13	0.3
RS	0.65	0.42	0.26	0.9	0.83	0.61	0.38	1.2	0.39	0.47	0.32	1.3
DT-MECH	0.23	0.23	0.16	0.6	0.18	0.18	0.14	0.4	0.23	0.23	0.17	0.4
BO-MECH	0.91	0.79	0.26	0.9	0.19	0.27	0.18	0.5	0.26	0.12	0.15	0.4
Ours	0.89	0.52	0.23	0.8	0.93	0.39	0.38	1.1	0.07	0.04	0.09	0.3

TABLE II: Energy savings of the proposed method relative to baselines for the pick and place task.

Baseline	**Excluding static power**	**Including static power**
MAX	20.9%	10.6%
MIN	12.2%	5.6%
RS	9.6%	4.2%
DT-MECH	10.2%	4.9%
BO-MECH	8.0%	3.7%

Sorting: In the next experiment, we evaluate the proposed

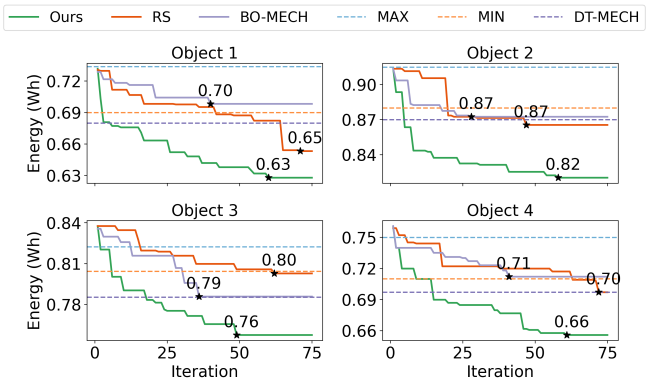


Fig. 7: Cumulative minimum energy for the sorting task. Solid lines show optimization-based methods, while dashed lines indicate non-optimization baselines.

methodology on a sorting task, with the setup shown in Fig. 5. Multiple objects of different shapes and weights are used, and both picking and placing locations are varied to emulate realistic scenarios. Each operation has a fixed duration $T_{\max}$: 30 s for Objects 1 and 4, 40 s for Object 2, and 35 s for Object 3.

Fig. 7 compares our approach with several baselines, implemented identically to the pick-and-place task. Since object types and pick/place locations vary, the BO-Mech baseline must optimize six control parameters per operation, yielding 24 parameters across four operations. These are estimated once and then used sequentially from Operation 1 to Operation 4. For the other baselines, the operation order is randomized, with each executed using the respective method's estimated optimal controls. As shown in Fig. 7, our method outperforms all baselines across the four objects. On average, energy is reduced relative to MAX, MIN, DT-MECH, RS, and BO-MECH by 11.2%, 7.8%, 5.8%, 6.2%, and 7.2%, respectively. We further repeated the experiment for 20 sorting tasks with varying objects, positions, and $T_{\max}$; average energy savings relative to each baseline, both excluding and including static power, are reported in Table III.

Unlike the pick-and-place task, the sorting task requires BO-Mech to optimize 24 control parameters. In most sorting tests, BO-Mech does not outperform the other baselines. As the number of parameters increases, the efficiency of black-box optimization methods such as Bayesian optimization degrades; it is most effective with fewer than about 20 control parameters [36]. To illustrate, Fig. 8 compares our algorithm with standard Bayesian optimization on a sorting operation, aiming to minimize overall energy(rather than only mechanical energy as in BO-Mech). As shown, our context-aware Bayesian approach achieves substantially better performance.

In Fig. 9, we examine how the context-aware Bayesian approach explores the parameter space and identifies optimal settings for Object 1 in the sorting task. Energy is plotted as a function of picking-state acceleration (A_1) and velocity (V_1). In Fig. 9(a), A_1 varies from 5% to 100% while other parameters are fixed; in Fig. 9(b), V_1 varies under the same conditions. The optimal A_1 is about 10% for mechanical

TABLE III: Energy savings of the proposed method relative to baselines for the sorting task.

Baseline	Excluding static power	Including static power
MAX	13.8%	11.3%
MIN	9.0%	7.3%
RS	6.4%	5.1%
DT-MECH	6.0%	4.8%
BO-MECH	7.7%	6.2%

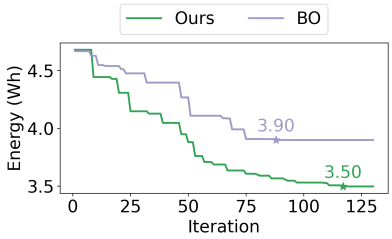


Fig. 8: Cumulative minimum energy across iterations between our context-aware and standard Bayesian.

energy and 40% for overall energy, while the optimal V_1 is about 20% and 60%, respectively. Fig. 10 shows how the algorithm converges to these optima: it first samples widely across the space, then progressively concentrates near the optimal values—around 40% for A_1 and 60% for V_1.

VI. CONCLUSIONS

This paper presents a runtime optimization framework that jointly optimizes mechanical motion and computational workloads to reduce overall energy consumption in a collaborative lightweight robotic arm. Unlike most prior energy optimization approaches for robotic arms that overlook computational energy consumption and focus solely on energy-aware locomotion control, our method captures the interaction between motion parameters and computational demand. It employs a context-aware Bayesian approach for the joint manipulation of arm locomotion and processor frequency. Experiments on a *Franka Emika Panda* robot performing pick-and-place and sorting tasks demonstrated energy savings of 3.7% and 6.2%, respectively, compared to methods that focus solely on minimizing mechanical energy. This work highlights a significant gap in the current state of the art in the field of energy-aware optimization of industrial robots, opening several challenges for more complex scenarios such as collaborative multi-arm systems with distributed computing architectures.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101177564 — HAIF.

REFERENCES

[1] J. Heredia, R. J. Kirschner, C. Schlette, S. Abdolshah, S. Haddadin, and M. B. Kjægaard, "Ecdp: Energy consumption disaggregation pipeline for energy optimization in lightweight robots," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6107–6114, 2023.

[2] G. Carabin, E. Wehrle, and R. Vidoni, "A review on energy-saving optimization methods for robotic and automatic systems," *Robotics*, vol. 6, no. 4, p. 39, 2017.

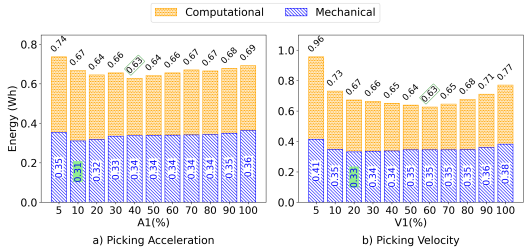


Fig. 9: Motion analysis in the picking state with other parameters fixed: (a) varying acceleration A_1; (b) varying velocity V_1.

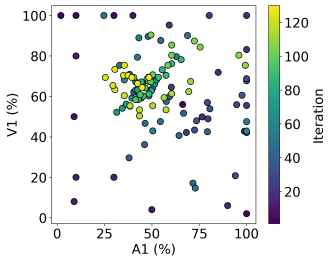


Fig. 10: Velocity V_1 versus acceleration A_1 for picking state across iterations. Point color encodes the iteration index.

[3] A. Vergnano, C. Thorstensson, B. Lennartson, P. Falkman, M. Pellicciari, C. Yuan, S. Biller, and F. Leali, "Embedding detailed robot energy optimization into high-level scheduling," in *2010 IEEE international conference on automation science and engineering*, pp. 386–392, IEEE, 2010.

[4] Y.-J. Kim, "Anthropomorphic low-inertia high-stiffness manipulator for high-speed safe interaction," *IEEE Transactions on robotics*, vol. 33, no. 6, pp. 1358–1374, 2017.

[5] A. Torayev, G. Martinez-Arellano, J. C. Chaplin, D. Sanderson, and S. Ratchev, "Online and modular energy consumption optimization of industrial robots," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 2, pp. 1198–1207, 2023.

[6] H.-Y. Lin, S.-C. Liang, and Y.-K. Chen, "Robotic grasping with multi-view image acquisition and model-based pose estimation," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 11870–11878, 2020.

[7] K. Shimonomura, H. Nakashima, and K. Nozu, "Robotic grasp control with high-resolution combined tactile and proximity sensing," in *2016 IEEE International Conference on Robotics and automation (ICRA)*, pp. 138–143, IEEE, 2016.

[8] A. Marino, "Distributed adaptive control of networked cooperative mobile manipulators," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 5, pp. 1646–1660, 2017.

[9] J. Heredia, C. Schlette, and M. B. Kjærgaard, "Data-driven energy estimation of individual instructions in user-defined robot programs for collaborative robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6836–6843, 2021.

[10] M. Pellicciari, G. Berselli, F. Leali, and A. Vergnano, "A method for reducing the energy consumption of pick-and-place industrial robots," *Mechatronics*, vol. 23, no. 3, pp. 326–334, 2013.

[11] J. Heredia, C. Schlette, and M. B. Kjærgaard, "Going green with lightweight robots: Energy optimal programming of lightweight robots," in *2023 21st International Conference on Advanced Robotics (ICAR)*, pp. 212–219, IEEE, 2023.

[12] A. Fenucci, M. Indri, and F. Romanelli, "An off-line robot motion planning approach for the reduction of the energy consumption," in *2016 IEEE 21st International conference on emerging technologies and factory automation (ETFA)*, pp. 1–8, IEEE, 2016.

[13] J. Muru and A. Rassõlkin, "A scoping review of energy consumption in industrial robotics," *Machines*, vol. 13, no. 7, p. 542, 2025.

[14] J. Jia and X. Sun, "Structural optimization design of a six-degrees-of-freedom serial robot with integrated topology and dimensional parameters," *Sensors*, vol. 23, no. 16, p. 7183, 2023.

[15] D. Meike and L. Ribickis, "Recuperated energy savings potential and approaches in industrial robotics," in *2011 IEEE international conference on automation science and engineering*, pp. 299–303, IEEE, 2011.

[16] M. Raessa, J. C. Y. Chen, W. Wan, and K. Harada, "Human-in-the-loop robotic manipulation planning for collaborative assembly," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1800–1813, 2020.

[17] R. Kantabussabong, C. Nantabut, W. Pongaen, N. Rothong, and B. Butsanlee, "Energy-efficient motion planning for dual-arm collaborative robots using a digital twin," in *2025 11th International Conference on Engineering, Applied Sciences, and Technology (ICEAST)*, pp. 239–242, IEEE, 2025.

[18] L. Bukata, P. Šůcha, Z. Hanzálek, and P. Burget, "Energy optimization of robotic cells," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 92–102, 2016.

[19] D. Angioletti, F. Bertani, C. Bolchini, F. Cerizzi, and A. Miele, "A runtime resource management policy for opencl workloads on heterogeneous multicores," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1385–1390, IEEE, 2019.

[20] J. Zidar, T. Matić, I. Aleksi, and Ž. Hocenski, "Dynamic voltage and frequency scaling as a method for reducing energy consumption in ultra-low-power embedded systems," *Electronics*, vol. 13, no. 5, p. 826, 2024.

[21] E. Del Sozzo, G. C. Durelli, E. Trainiti, A. Miele, M. D. Santambrogio, and C. Bolchini, "Workload-aware power optimization strategy for asymmetric multiprocessors," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 531–534, IEEE, 2016.

[22] A. K. Singh, A. Prakash, K. R. Basireddy, G. V. Merrett, and B. M. Al-Hashimi, "Energy-efficient run-time mapping and thread partitioning of concurrent opencl applications on cpu-gpu mpsocs," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–22, 2017.

[23] S. Shahsavari, H. Haghbayan, A. Miele, E. Immonen, and J. Plosila, "A coordinated approach to control mechanical and computing resources in mobile robots," *IEEE Transactions on Robotics*, 2024.

[24] S. A. Mohamed, M.-H. Haghbayan, A. Miele, O. Mutlu, and J. Plosila, "Energy-efficient mobile robot control via run-time monitoring of environmental complexity and computing workload," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7587–7593, IEEE, 2021.

[25] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, Y. Kwon, K. Michael, J. Fang, Z. Yifu, C. Wong, D. Montes, *et al.*, "ultralytics/yolov5: v7. 0-yolov5 sota realtime instance segmentation," *Zenodo*, 2022.

[26] A. Lukezic, T. Vojir, L. ˇCehovin Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6309–6318, 2017.

[27] T. Brito, J. Lima, P. Costa, and L. Piardi, "Dynamic collision avoidance system for a manipulator based on rgb-d data," in *Iberian Robotics conference*, pp. 643–654, Springer, 2017.

[28] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 2, pp. 995–1001, IEEE, 2000.

[29] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *arXiv preprint arXiv:1804.05172*, 2018.

[30] J. Leng, J. Peng, J. Liu, Y. Zhang, J. Ji, and Y. Zhang, "Profiling power consumption in low-speed autonomous guided vehicles," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6027–6034, 2024.

[31] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[32] A. Pathania, Q. Jiao, A. Prakash, and T. Mitra, "Integrated cpu-gpu power management for 3d mobile games," in *Proceedings of the 51st Annual Design Automation Conference*, pp. 1–6, 2014.

[33] MoveIt, "Moveit motion planning framework." `https://moveit.ai/`. Accessed: 2025-09-16.

[34] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The journal of machine learning research*, vol. 13, no. 1, pp. 281–305, 2012.

[35] J. Yan and M. Zhang, "A transfer-learning based energy consumption modeling method for industrial robots," *Journal of Cleaner Production*, vol. 325, p. 129299, 2021.

[36] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.