Optimizing performance in automation through modular robots

Stefan B. Liu and Matthias Althoff
Department of Informatics, Technical University of Munich, Germany, Email: [stefan.liu | althoff]@tum.de

Abstract—Flexible manufacturing and automation require robots that can adapt to changing tasks. We propose to use modular robots that are customized from given modules for a specific task. This work presents an algorithm for proposing a module composition that is optimal with respect to performance metrics such as cycle time and energy efficiency, while considering kinematic, dynamic, and obstacle constraints. Tasks are defined as trajectories in Cartesian space, as a list of poses for the robot to reach as fast as possible, or as dexterity in a desired workspace. In a simulative comparison with commercially available industrial robots, we demonstrate the superiority of our approach in randomly generated tasks with respect to the chosen performance metrics. We use our modular robot proModular.1 for the comparison.

I. INTRODUCTION

Increasing demand for mass-customization requires robots that can adapt to frequently changing tasks. However, because standard industrial robots cannot change their kinematics, it may occur that new positions are unreachable, or new trajectories can only be followed in a suboptimal manner. Modular robots – robots which are composed from a certain number of reconfigurable hardware modules – are a potential solution [1]. Conceptual advantages of modular robots are imminent: flexible reconfiguration allows the robot to adapt to changing tasks; broken robots can be easily repaired by exchanging affected modules; and, by mass-producing standard modules, the costs of robots can be reduced.

Nowadays, a robot is typically selected for a specific application according to the required performance [2], e.g., weight-carrying capacity, working envelope, repeatability, and speed. Due to increasing efforts of reducing carbon footprints, the energy consumption of robots is also becoming a relevant factor [3]. Further performance metrics of manipulators have been surveyed in [4]. In this work, we propose a computationally-feasible automatic module selection process that optimizes these performance metrics. We compare the performance for several manufacturing applications categorized by their predominant path type (see Tab. I). 1) fully specified trajectories, e.g., dispensing a fluid on a fixed part at a steady speed; 2) a list of poses for the robot to reach as fast as possible, e.g., handling a part; 3) optimizing the designated working area of the robot using workspace performance metrics (see Tab. II) if the path is not known. In all cases, the composed modular robot shall be capable of reaching the desired poses, holding its own weight, exerting the required external forces, and avoiding collisions with itself and surrounding objects. We do not regard repeatability in this work, because it depends largely on the mechanical structure and the controller, which are not part our evaluation.

To evaluate performance gains of modular robots, we have designed proModular.1, a set of 52 reconfigurable modules for industrial use (see Fig. I). These modules are capable of automatically generating the composed kinematic, dynamic, and geometric model given the information stored inside each module. The joint modules can exert torques at a range from 25 to 226 Nm. We evaluate performance gains by these modules for industrial tasks by comparing them to three commercially available industrial robots for 2700 randomly generated tasks, for which we each find an optimal module composition.

Previous works on task-based optimization of module compositions can be found in [1], [5]–[13]. These have dealt with large search-spaces in different manners: 1) using incidence matrices for efficient enumeration [6], [7], 2) using Genetic Algorithms for finding the optimal module sequence [10]–[13], and 3) choosing a hierarchical elimination approach [1], [9], [10]. The authors of [5], [13], [14] also addressed optimizing modular robots for fault tolerance. Most of previous authors only consider optimizing the kinematics, or module masses [8], [10].

In our work we choose a practical approach, which
only enumerates compositions resembling common industrial robot kinematics [15], combined with a hierarchical elimination to exclude infeasible module sequences. In contrast to previous work, we consider the full dynamics of the modular robots. In addition to [1], we also consider collision avoiding trajectories using Rapidly-exploring random trees (RRT) [16] connected with time-optimal trapezoids [17] to plan around complex obstacles and prevent self-collision. We have been able to decrease the computation time for optimizing a single task from multiple hours [14] to around (fast transition between poses which are given as a list of pairs integrated over the time horizon in Table II) connected with time-optimal trapezoids [17] to plan around complex obstacles and prevent self-collision. Therefore, for the first time, we are able to demonstrate the performance gains of modular robots versus commercially available industrial arms in a comprehensive comparison.

In Sec. II we formulate the optimization problems for the three previously-mentioned cases in manufacturing. The module selection process is then explained in Sec. III. We evaluate the performance gains in Sec. IV and conclude the paper in Sec. V.

II. PROBLEM FORMULATION

We aim to find the optimal composition of modules given task specifications in Cartesian space and an obstacle map. Although we chose to only consider energy, cycle time, and dexterity, most of the performance metrics of Tab. II and [4] can be included without much additional computational burden, because our module selection process already requires simulating the Jacobian, as well as the kinematics and dynamics of the composed robot. Subsequently, we denote the set of possible compositions as $C = \{C_1, C_2, ..., C_N\}$, where $C_i$ is the $i^{\text{th}}$ composition and $N$ is the number of considered compositions. We perform the evaluation for the following three cases.

Case 1: Desired trajectory

In applications such as in welding, dispensing, and processing, a desired trajectory of the end-effector is typically specified by the time-dependent functions $p(t), \dot{p}(t), \ddot{p}(t), R(t), \omega(t), \dot{\omega}(t), F(t)$, which are the desired three-dimensional position, velocity, acceleration, and the three-dimensional rotational velocity, acceleration, and the six-dimensional Cartesian force profile, respectively. These functions are used to compute the motor torque vector $u_i(t)$ and the joint velocity vector $\dot{\theta}_i(t)$, which are different for each composition $C_i$. We minimize the energy consumption $E_i$:

$$E_i = \int_0^{t_f} |u_i(t)^T|\dot{\theta}_i(t)|dt$$

subject to $\forall t \in [0, t_f]$:

$$q_i(t) \in \left[q_{i,b}, q_{i,e}\right]$$ (solution within joint limits) (3)

$$u_i(t) \in \left[u_{i,b}, u_{i,e}\right]$$ (torque limits) (4)

$$A_i(q_i(t)) \cap O = \emptyset$$ (no collisions) (5)

where $A_i(q_i(t))$ returns the occupancy of the robot (no self-collision allowed) in Cartesian space and $O$ is the set of obstacles.

Case 2: List of desired poses

For applications such as handling, machine tooling, and assembly, the exact trajectory is not important, rather than a fast transition between poses which are given as a list of pairs $(p, R)$, and forces $F$ (e.g., a payload to transport) to minimize cycle time. We therefore search for the fastest collision-free trajectory between each pose for every composition $C_i$. The optimization problem now depends on a weighted sum of the time $t_{f,i}$ and energy consumption $E_i$ to reach the final pose, using weighting factors $w_i, w_e$:

$$\min_{C_i \in C} w_i t_{f,i} + w_e E_i,$$ (6)

subject to the same constraints of (3)-(5).

Case 3: Desired workspace

If no path has been specified, then workspace optimization can optimize the robot performance within a designated area, instead. As an example, we maximize the mean dexterity index $\omega$ integrated over $M$ uniformly sampled points of a three-dimensional desired workspace:

$$\max_{C_i \in C} \frac{\sum_{j=1}^M D_{i,j}}{M},$$ (8)

$$D_{i,j} = \frac{1}{3} \left( \frac{\Delta \alpha_{i,j}}{2\pi} + \frac{\Delta \beta_{i,j}}{\pi} + \frac{\Delta \gamma_{i,j}}{2\pi} \right)$$ (9)

where $\Delta \alpha_{i,j}, \Delta \beta_{i,j}, \Delta \gamma_{i,j}$ are the range of possible yaw, pitch and roll angles about each of the sampled points, evaluated via the existence of an inverse kinematics solution.

III. OPTIMAL MODULE COMPOSITION

Evaluating the costs introduced in the previous section for all possible module compositions is computationally expensive. Since the number of possible compositions increases exponentially with the length of the module sequence, calculation may become infeasible. In the following, we introduce our contributions towards a computationally feasible optimization: enumeration of common industrial kinematics, their hierarchical elimination, and the generation of collision models of modular robots.

<table>
<thead>
<tr>
<th>Requires</th>
<th>Performance metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobian</td>
<td>Manipulability, minimum singular value, condition number, isotropy, velocity ratio</td>
</tr>
<tr>
<td>Kinematics</td>
<td>Reachability, Dexterity, operating volume, orientation angle workspace</td>
</tr>
<tr>
<td>Dynamics</td>
<td>Dynamic manipulability, dynamic isotropy</td>
</tr>
</tbody>
</table>
The presented approach is similar to our previous work [9], but additionally considers full robot dynamics and collisions with complex objects.

The hierarchy order is determined by the amount of computation needed and consists of five steps. To quickly eliminate robots with insufficient size and dynamics we initially perform the following feasibility checks for select poses of the tasks (here: via-points): step 1) the existence of an inverse kinematics solution [3], then step 2) whether the gravity torques computed via the Newton-Euler (NE) Algorithm [18] can be held by the motors (4), and then step 3) whether collisions (5) are present by evaluating (5). In step 4), the previous checks are performed again, but for the complete path by using:

- Case 1: the inverse kinematics trajectory [18].
- Case 2: the interpolated path between all desired poses and the gravity torques via the NE-Algorithm, and
- Case 3: the inverse kinematics solution of the entire workspace.

If, for case 2, there is a collision, then the shortest collision-free path is computed. In this work, we use RRT-connect [16] to generate paths in joint space, while the collision checks are done in Cartesian space. A time-optimal trapezoidal trajectory [17] is then generated following the previously computed path. In step 5) the trajectory is then used to compute the robot dynamics and the NE-Algorithm to check (6) again, while considering external forces. At last, the cost functions of Sec. II can be evaluated using the quantities computed in this section. For case 3, we refer to [19] for efficiently evaluating the dexterity index. The elimination algorithm is also shown in Fig. 4. In the next subsection we detail the generation of models, which are needed for our approach.

C. Automatic robot modelling

Kinematic, dynamic, and geometric modeling of modular robots is required to evaluate the cost functions of Sec. II. For kinematic and dynamic modeling we refer to previous work in [20], while in this subsection, we describe the novel geometric modeling for collision checking. We make use of the extended Denavit-Hartenberg (DH) formulation [20], which defines a frame of reference $DH_{ext,i}$ for each link $i$ of the composed robot.

The geometry of a robot link $i$ consists of the the distal part of a joint module $J_i$ (axis $i$), a link module $L_i$, and the proximal part of joint module $J_{i+1}$ (axis $i + 1$). The kinematics of each module can be defined using the following $4 \times 4$ transformation matrices (see Fig. 5):

- Joint modules: $T_{J_{i,1}}$ is the transformation between the input frame $in$ and the auxiliary frame $a$ before the joint (see [20]), and $T_{J_{i,2}}$ between the second auxiliary frame $b$ after the joint and the output frame $out$.
- Link modules: $T_L$ is the transformation between $in$ and $out$.

Subsequently, we use superscripts to indicate the reference frame. To define the non-convex geometry of a module, we
B. Compute full trajectory $q(t)$, $\dot{q}(t)$, $\ddot{q}(t)$ or workspace IK

A. Compute inverse kinematics (IK) solutions at select poses

1. Check existence of IK solution and joint limits (see (3))
2. Check whether gravity forces are within torque limits (see (4))
3. Check for collisions at these poses (see (5))

Case 1:

1. Check existence of IK solution and joint limits (see (3))
2. Check whether gravity forces are within torque limits (see (4))
3. Check for collisions at these poses (see (5))

Case 2:

4. Check (3), (4) only for gravity forces, and (5) for path/workspace

Case 3:

Interpolate path (RRT-connect) [16]
Compute time-optimal trajectory [17]
Compute IK-solutions for entire workspace

5. Compute torques $u(t)$ of the full dynamics and check (4)

Compute cost function (1)/(6)/(8)

Considered composition $i$

**A. Compute inverse kinematics (IK) solutions at select poses**

1. Check existence of IK solution and joint limits (see (3))
2. Check whether gravity forces are within torque limits (see (4))
3. Check for collisions at these poses (see (5))

**B. Compute full trajectory $q(t)$, $\dot{q}(t)$, $\ddot{q}(t)$ or workspace IK**

**Case 1:**
- Compute IK-trajectory
- Interpolate path in joint space
- Compute IK-solutions for entire workspace
- If collision, compute collision-free path (RRT-connect) [16]
- Compute time-optimal trajectory [17]
- 5. Compute torques $u(t)$ of the full dynamics and check (4)

**Case 2:**
- Compute IK-trajectory
- Interpolate path in joint space
- Compute IK-solutions for entire workspace
- If collision, compute collision-free path (RRT-connect) [16]
- Compute time-optimal trajectory [17]
- 5. Compute torques $u(t)$ of the full dynamics and check (4)

**Case 3:**
- Compute IK-trajectory
- Interpolate path in joint space
- Compute IK-solutions for entire workspace
- If collision, compute collision-free path (RRT-connect) [16]
- Compute time-optimal trajectory [17]
- 5. Compute torques $u(t)$ of the full dynamics and check (4)

**Compute cost function (1)/(6)/(8)**

Fig. 4. Hierarchical elimination algorithm with five feasibility checks. The computation for one composition is aborted, if any constraint is not satisfied.

**Fig. 5. Illustration of kinematics for synthesizing the geometry of robot link $i$.** Two modules are connected via attaching the output of a module (dark gray) to the input of the next module (light gray), such that the output frame and input frame overlap, e.g., at frame $z_0$.

Fig. 3. The proModular.1 modules are categorized into joints (J), links (L), wrists (W) and end effectors (E) according to Fig. 2B. The number of modules is shown in brackets. Not displayed are module variants with different interface sizes and/or varying motor sizes for wrist modules.

use meshes. A mesh consists of a set of vertices $V \in \mathbb{R}^{3 \times m}$ and a set of faces, i.e., triangles defined by the vertices:

- Joint modules: $V_{J,1}$ are vertices relative to auxiliary frame $a$ for the proximal part, and $V_{J,2}$ relative to the output frame for the distal part,
- Link modules: $V_L$ are vertices relative to the output frame.

The geometry of robot link $i$ relative to frame $DH_{ext,i}$ is therefore represented by the set of vertices $V = \{V_{J,2}^*, V_{L,1}^*, V_{J,i+1}^*\}$, where

$$[V_{J,i} 1]^T = R(\phi_i)T_{J,i+1}^a T_{L,1}^{out} [V_{J,i} 1]^T$$

$$[V_L^* 1]^T = R(\phi_i)T_{J,i+1}^a [V_L 1]^T$$

$$[V_{J,i+1}^* 1]^T = R(\phi_i) [V_{J,i+1} 1]^T$$

where for computing angle $\phi_i$ we refer to [20]. Standard forward kinematics algorithms can be then used to transform the vertices of each robot link into Cartesian space, where collision checks are performed. Simpler geometries, such as boxes or spheres, can also be used with this approach, but lead to more conservative collision checks.

**IV. Evaluation Study**

In a simulated evaluation study, we compare the performances of optimized modular robots to commercially available industrial manipulators for randomly generated tasks. Data and videos showing the randomly generated tasks for this study have been made available to the public.

Collision-free paths and collision checks are implemented using the Open Motion Planning (OMPL) and the Fast Collision Checking Library (FCL).

For our comparison, we use a Schunk LWA-4P 6-DOF lightweight robot (Industrial robot 1), a KUKA LWR 4+ 7-DOF lightweight robot (Industrial robot 2), and a Staubli TX90 robot (Industrial robot 3). These robots have been particularly chosen for the availability of their models [22]–[24]. The base coordinate system of all robots is set to the origin $(0,0,0)$ without loss of generality, except for ETL-class robots, which are hanging from above, for which we set the base at $(0,0,1)$. For optimization, we consider the kinetic and the potential of the robots. For case 1 and 2, a random payload between 0-50N represents the external force. In the
following, we describe the experiments and their results for each case.

A. Case 1: Desired trajectory

For case 1 we generate trapezoidal trajectories at random constant speeds in Cartesian space connecting, two, three, and four poses for each industrial robot, which results in nine different settings, each with 100 experiments resulting in 900 experiments in total. To make sure these poses can be reached by the respective industrial robot, the poses are randomly generated in their joint spaces instead of the Cartesian space. As a simplification, but without loss of generality, the trajectories do not alter the orientation of the end-effector, and we do not consider external obstacles. The optimal module composition is chosen according to Sec. III. An example of the experiments is seen in Fig. 8 for comparing industrial robot 1 with the optimal modular robot for a trajectory with 4 poses, where the improvement is also displayed.

In Tab. IV we show the mean and variance of the improvement in percent of the energy for each setting. We observe that the energy savings become less when the trajectory is longer. We can see in the comparison with industrial robot 1 that due to clever optimization, energy savings of proModular.1 are possible despite having a higher robot mass to a large extent due to the customized kinematics. We also see that the energy savings are bigger compared to industrial robot 3, because the average modular robot in these cases have much lower masses.

B. Case 2: Desired poses

Similar to case 1, we automatically sample two, three and four poses in Cartesian space for each industrial robot. Additionally, we consider these scenarios: 1) a scenario without obstacles in the environment, and 2) a pick and place task, where the robot gripper goes in and out of a milling machine. The optimal module composition selects the fastest and energy efficient robots for this task with \( w_1, w_2 = [1, 0.2] \), \( \bar{q}_{\max} = 1 \frac{\text{rad}}{s} \), and \( \bar{q}_{\max} = 1 \frac{\text{rad}}{s} \). The results are shown in Tab. IV and an example is shown in Fig. 9.

Similar to case 1, time and energy savings are less, when the trajectory length increases. A further decrease in cost savings is visible when considering obstacles. This might be an indication that a higher number of task constraints lead to less room for better performance in a direct comparison. However, what has been neglected in these experiments is that through optimization and adaptation, a modular robot is limitless when it comes to performing tasks, that would be infeasible for fixed-shape industrial robots.

Another interesting outcome is shown in Fig. 6, which presents the frequency a kinematics has been chosen in all experiments of case 1 and 2. The DLR kinematics has been chosen by far the most for case 1, which supports the fact that kinematic redundancy leaves more room for trajectory optimization in inverse kinematics control. For case 2, the PUMA kinematics is chosen the most. Elbow robots have almost never been selected.

C. Case 3: Desired workspace

In this experiment we showcase the results of optimizing the dexterity index of workspaces. The three investigated workspaces are shown in Fig. 7. We consider poses that lead to self-collisions as not reachable. The results for the optimal modular robot, as well as the three industrial robots are shown in Tab. V. It displays clearly that a dexterity > 0.7 can always be achieved (maximum is 1), while such is not the case for industrial robots.

V. CONCLUSIONS

In this work we show a computationally feasible approach for the task-based optimization of modular robots based on the enumeration of common industrial robot kinematics and on hierarchical elimination, that involves feasibility checks of the kinematics, dynamics, and collision checking of the composed robot. In an evaluation we show that our
TABLE V
CASE 2: ENERGY AND TIME SAVINGS OF MODULAR ROBOTS – WITH AND WITHOUT OBSTACLES (100 EXPERIMENTS PER SITUATION)

<table>
<thead>
<tr>
<th>number of poses</th>
<th>Time savings in %</th>
<th>Energy savings in %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time savings in %</td>
<td>Energy savings in %</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>no obstacles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Industrial robot 1</td>
<td>33.1</td>
<td>22.2</td>
</tr>
<tr>
<td>Var.</td>
<td>4.13</td>
<td>3.04</td>
</tr>
<tr>
<td>Industrial robot 2</td>
<td>37.0</td>
<td>28.3</td>
</tr>
<tr>
<td>Var.</td>
<td>3.90</td>
<td>2.59</td>
</tr>
<tr>
<td>Industrial robot 3</td>
<td>43.4</td>
<td>30.1</td>
</tr>
<tr>
<td>Var.</td>
<td>4.80</td>
<td>1.89</td>
</tr>
</tbody>
</table>

proModular.1 robot often achieves better performance for standard tasks in manufacturing than three commercially available industrial robots. This work provides a path towards a broader deployment of modular robots in industrial automation. Future works shall introduce translational joints for considering more kinematics (e.g., SCARA) and applying optimization methods to further reduce computational load. Furthermore, we aim at methods to evaluate the repeatability of modular robots.

ACKNOWLEDGMENT

The authors gratefully acknowledge financial support by the Central Innovation Programme of the German Federal Government under grant ZF4086004LP7. We especially thank Paul Maroldt, Hanna Krasowski, Florian Skiba, and Daniel Gheorgita for their contributions in designing proModular.1 and Constantin Dresel for implementing the collision avoidance planning.
REFERENCES


