Co-Optimizing Robot, Environment, and Tool Design via Joint Manipulation Planning

Marc Toussaint[1,2], Jung-Su Ha[2,1], Ozgur S. Oguz[3,2]

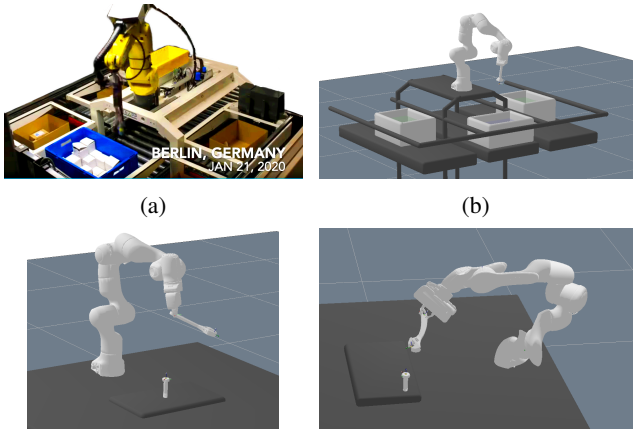
***Abstract*— Existing work on sequential manipulation planning and trajectory optimization typically assumes the robot, environment and tools to be given. However, in particular in industrial applications, it is highly interesting to ask, what would be an optimal robot design, tool shape, or robot station geometry for a particular ensemble of manipulation tasks. To tackle this problem we propose a formulation to jointly optimize over static design parameters and the sequential manipulation trajectory. We can include optimization objectives such as penalizing velocities (path length) and joint torques. Our evaluations show that design optimization can significantly improve on such metrics. For instance, in a wrench tool demonstration scenario we show that the shape of the wrench tool as well as design of the robot can be optimized to allow for exerting a necessary external torque with minimal effort.**

I. Introduction

Trajectory optimization, force-based manipulation planning, and combined task and motion planning are typically addressed for a fixed robotic hardware. However, the design and kinematics of the robotic system clearly has a strong influence on the quality or feasibility of optimized trajectories and manipulation plans. The importance of the hardware design has been discussed for a long time, including the discussion of funnels [1], [2] that can be created by clever hardware design, morphological computation [3]–[5] that is realized implicitly by compliant and soft hardware when interpreted as integral part of the control strategy, and embodiment [6]. So far, however, these considerations remained largely separate from the methods developed for trajectory optimization.

Further, tool use planning has previously been considered as an instance of TAMP and force-based manipulation planning [7]. But tool creation or tool design that is co-optimized with the task has received only very little attention [8]–[10].

In this paper we consider the problem of co-optimizing robot, environment, or tool design parameters *jointly* with sequential manipulation trajectories, for objectives such as control costs or torque loads. Previous work [11], [12] considered similar formulations but assumed given endeffector waypoints or trajectories, while we propose a general constrained optimization formulation to include sequential manipulation and environment torque objectives. By including a representative ensemble of task instances in the optimization we can co-optimize the design parameters with the solutions to sequential manipulation problems.



(a) (b) (c) (d)

Fig. 1: (a) A bin picking and re-sorting robot station presented by Knapp/Covariant.AI for logistics applications[1]. (b) Our analogous re-sorting demonstration scenario (in non-deformed design). (c) Our wrench tool scenario (non-deformed) where the robot needs to exert a constant torque on the bolt during interaction. In both scenarios the pick and place location of the object and bolt location, respectively, are randomized to yield the ensemble of K scenarios. (d) An example deformed-design for the wrench tool scenario.

Industrial applications are a core motivation for this work: Robots in production lines, bin picking, or logistics re-sorting applications typically run on the same task ensemble for their whole lifetime. The design of the overall robot station (including the mounting and tilt of the robot, or of bins, relative to the production line) as well as the design of the robot itself (its link transformations, and thereby its kinematics and joint axes) could be tailored to a particular task ensemble to yield higher speed and efficiency. However, default industrial robotics hardware is dominated by stereotypical designs with orthogonal consecutive joint axes, see Fig. 2. These designs have been long proven, yield a simple kinematic description and a good *general-purpose* workspace. But they are far from optimal w.r.t. typical trajectory ensembles.

Our core contribution is a formulation to jointly optimize over sequential manipulation trajectories and static design parameters of the robot, the environment, and tools, for objectives such as path velocities, accelerations and joint torques. To bring this to application we further propose a spe-

This research has been supported by the German Research Foundation (DFG) under Germany's Excellence Strategy – EXC 2120/1 – 390831618, and via the Max Planck Fellowship (MPI for Intelligent Systems).

[1]Learning & Intelligent Systems Lab, TU Berlin, Germany
[2]Max Planck Institute for Intelligent Systems, Germany
[3]Machine Learning & Robotics Lab, University of Stuttgart, Germany

Fig. 2: Typical robot designs are dominated by orthogonal consecutive joint axes, non-optimized to particular tasks.

cific parameterization of link deformations, model pick-and-place and wrench tool constraints in a differentiable manner, and introduce wrench decision variables and constraints to allow us to evaluate and/or penalize forces and torques in the structure. We demonstrate the approach first on a bin picking and re-sorting industrial application, analogous to the one demonstrated by Knapp/Covariant.AI, see Fig. 1(a,b). We then consider a wrench tool scenario. Our evaluations show that design optimization can significantly improve the performance metrics.

II. Related Work

a) Co-optimizing Design Parameters with Trajectories: Most closely related to our approach is the work [12], which equally co-optimizes design parameters with joint trajectories, but assumes endeffector trajectories given and takes the alternating optimization approach, as well as the work [11], which leverages IK methods to co-optimize manipulator designs (in particular link lengths) for given target waypoints in endeffector space. Our approach leverages a general constraint optimization formulation of sequential manipulation that can also represent pick-and-place problems as well as include objectives on force-exchanges with the environment. Further, our approach optimizes design parameters w.r.t. an ensemble representation of the task variety.

b) Tool Use, Design, and Creation: A highly interesting related line of research concerns tool use and design. [13] proposes methods to co-optimize the shape of a throwing tool with its motion to maximize effectiveness, [10] presents an extensive treatment of tool use planning, [14] considers tool selection and substitution, and [15] tool creation based on combining parts. Further [8] considers learning and reasoning for tool creation on the symbolic level. Our work instead integrates the problem of tool design in the general problem of sequential manipulation planning on the motion and force level.

c) Combined Task and Motion Planning: By planning high-level action sequences along with feasible motion paths to realize those actions, TAMP formulations provide effective methods to solve sequential manipulation problems. The class of problems tackled by TAMP methods include *pick-and-place planning* [16], [17], *manipulation planning* [18], *tool-use* [19], *navigation among obstacles* [20], and *rearrangement planning* [21]. Classical AI search algorithms for task planning are combined with either sampling-based [17] or optimization-based methods [19] for motion planning to tackle those broad set of problems. In this paper we include also the co-optimization of environment and robot design parameters along with the robot motion plans for executing an ensemble of sequential manipulation tasks.

[1] https://www.youtube.com/watch?v=ApiQwHlxdmA

III. Co-Optimizing Design, Motion, and Manipulation Plans

A. General Optimization Formulation

We formulate the problem as a nonlinear mathematical program (NLP) over a path in configuration space that spans the full ensemble of K sample tasks. Specifically, we consider the path $x : [0, KT] \to \mathcal{X}$, where each time interval $[kT, (k+1)T]$ of duration T performs the kth sample task. The configuration space $\mathcal{X} \subset \mathbb{R}^n \times \mathbb{R}^D \times SE(3)^m \times \mathbb{R}^{6 \cdot n_\mathrm{F}}$ includes the following degrees-of-freedom (dofs):

- the n robot joint angles $q \in \mathbb{R}^n$ subject to control,
- D design parameters $\delta \in \mathbb{R}^D$ explained in more detail below,
- the (relative) poses of m rigid objects ($SE(3)^m$), where m may change from time slice to time slice depending on how objects are manipulated,
- the 6D wrenches $f \in \mathbb{R}^{6 \cdot n_\mathrm{F}}$ between n_F pairs of objects or links.

As our solver discretizes time we directly formulate the NLP using a discrete time notation. Let T be an integer (the number of time slices per task phase), and the path $x : \{1, .., KT\} \to \mathcal{X}$ is discretized in time slices $x_t \in \mathcal{X}$ for $t \in \{1, .., KT\}$. We solve an NLP of the generic form

$$\min_x \quad \sum_{t=1}^{KT} f_t(\bar{x}_t) \tag{1}$$

$$\text{s.t.} \quad \forall_t : \; h_t(\bar{x}_t) = 0, \; g_t(\bar{x}_t) \le 0, \tag{2}$$

where $\bar{x}_t$ is a tuple of configurations on which the cost, equality and inequality objectives depend. The solver can handle arbitrary tuples of configurations, but in our applications we assume $\bar{x}_t = (x_{t-2}, x_{t-1}, x_t)$, which means that objectives may depend on up to three consecutive time slices. Specifically, objectives can be formulated in terms of any features ϕ that depend on one time slice $\phi_t \equiv \phi(x_t)$, or finite-difference velocities of features $\dot{\phi}_t \equiv (\phi_t - \phi_{t\text{-}1})/\tau_t$, or finite-difference accelerations of features $\ddot{\phi}_t \equiv (\dot{\phi}_t - \dot{\phi}_{t\text{-}1})/\tau_t$, where τ_t is the time interval between two time slice t-1 and t. Our framework generally requires that the *path Jacobian* $\frac{\partial \phi_t}{\partial x}$ of all features (w.r.t. the *full* path) are sparse. This is ensured in our case as they depend only on consecutive tuples $\bar{x}_t$ which, however, include a global shared design parameter δ which makes Jacobians non-banded but still sparse.

The following section first details the design parameters. Next, we will individually detail all cost, equality and inequality objectives we use to formulate pick-and-place constraints, collision constraints, control costs, force constraints and torque costs.

B. Deformations and Design Parameters δ

1) Shared decision variables and sparse path Jacobians: The design parameters δ are the core interest of this paper. As

they represent static hardware parameters, they cannot vary over time therefore need to be constant throughout the path. We nevertheless treat them analogously to robot joint angles q to enable their co-optimization with all other decision variables. There are several options to ensure constant δ throughout the path when formulating the mathematical program: We could impose local equality constraints on δ at two consecutive time slices. This would ensure that all objectives of the mathematical program are in fact temporally local, the resulting Hessian of all objectives would be band diagonal, and Newton steps could be computed in time linear in KT – see [22] for a detailed discussion of exploiting temporally local ("k-order") structure of path optimization problems.

However, we decided for a formulation of the NLP that breaks this temporal locality: We follow a true parameter sharing approach where the design parameters δ in each time slice are identical to the very first time slice, leading to only a single decision variable in the mathematical program. Consequently, the objectives (e.g., a robot endeffector position or joint torque) of the NLP now depend on both, temporally local variables such as the robot poses q and forces f, as well as the design parameter δ in the first time slice. To cope with this non-local structure of the NLP we use sparse representations of all path Jacobians, also leading to a sparse Hessian. Exploiting efficient sparse solvers (we use eigen's Simplicial LDLT) we can almost retain the linear complexity of computing Newton steps.

2) Link Deformations: Our basic approach to parameterize design is to introduce additional dofs for pose transformations relative to the original design. We first describe this in the context of deformations of robot links. Pose alterations of other aspects of the environment can be handled analogously.

Consider a link transformation $Q_i \in SE(3)$ in the robot's kinematic tree. Such a transformation Q_i reflects the rigid hardware link and transforms from the coordinate frame where the link is mounted (the output coordinate frame of the previous motor) to the coordinate frame where the next motor is mounted. We introduce a *link deformation* $\delta_i \in SE(3)$ such that after the deformation the link transformation is

$$Q_i' = Q_i \circ \delta_i \ ,$$

i.e., the deformation is introduced between the old link and the next motor. It can thereby elongate, shorten or sheer the old link shape, as well as introduce a rotation of the next motor axes by twisting the link shape.

We generally parameterize deformations δ_i using 7D position-quaternion coordinates, while adding also a quaternion normalization equality constraint to the mathematical program. We always impose limits on the deformations, e.g., for panda link deformations we impose upper limits $[.05, .05, .05, 1.05, .3, .3, .3]$ and lower limits $[-.05, -.05, -.05, .8, -.3, -.3, -.3]$ on the 7D position-quaternion deformation coordinates. This means $\pm$5cm translations, while the amount of rotation is limited by the $\pm.3$ limits on the xyz-coordinates of the respective quaternion, where .3 corresponds to $\sim 35^\circ$ around each axis.

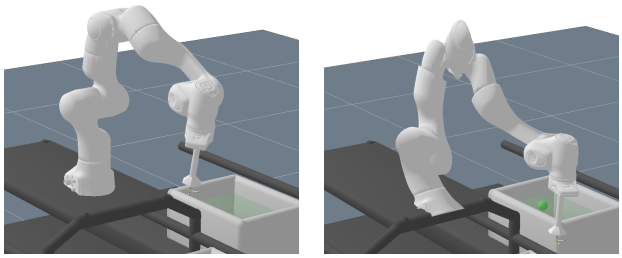


Fig. 3: Illustration of the original Panda design, and a deformed design. Deformation transformations are introduced before each joint. The transformations are illustrated via a simple linear warping of the mesh models of the preceding link. The resulting deformations often lead to highly non-orthogonal joint axes.

We make the simplifying assumption that these deformations do not alter the total mass or inertia for the link.

Fig. 3 illustrates a deformation of the Panda arm links optimized for minimal torques. For the purpose of this illustration, the mesh models have been linearly warped to reflect the deformation. Of course, in practice the specific shape designs for the deformed links could be chosen differently depending on what is cheapest and easiest to manufacture. The naively warped mesh models are for illustration purpose only.

For completeness we describe this linear warping of a mesh model: For every link we identify a central start point a (center of the mount coordinate frame) and central end point b (center of the output coordinate frame). It holds $b = Q^{-1}a$. After the deformation we want $a' = a$ while $b' = (Q \circ \delta)^{-1}a = \delta^{-1}b$. For every vertex v of the mesh model we identify the linear interpolation coefficient $\alpha = \frac{(v-a)^\top (b-a)}{(b-a)^2}$, which is 0 for $v = a$ and 1 for $v = b$. We then transform each mesh vertex according to $v' = (\alpha\dot{\delta}^{-1})v$, where $\alpha\dot{\delta}^{-1}$ is shorthand for the interpolated transformation, more rigorously defined via the exponential map of the respective Lie group.

We can apply the same scheme also for parameterizing the mounting of objects in the environment, or for the deformation of a tool.

C. Manipulation Constraints

1) Pick-and-Place Constraints: For consecutive pick-and-place tasks, we assume that the travel from pick-to-place takes the same number of time slices as from place-to-pick. On a pick time slice we impose the constraints

$$0 = \phi_{\text{pos}}^{\text{endeff}} - \phi_{\text{pos}}^{\text{pickPoint}} \tag{3}$$

$$0 = \phi_{\text{z}}^{\text{endeff}} - \phi_{\text{z}}^{\text{object}} \ , \tag{4}$$

where $\phi_{\text{pos}} \in \mathbb{R}^3$ is the 3D position of the endeffector, or the given desired object pick point in world coordinates, and $\phi_{\text{z}} \in \mathbb{R}^3$ is the z-axis of the endeffector or object in world coordinates. The latter equality means that the endeffector tool aligns normally with the object.

We also impose zero endeffector velocity at the pick time

$$0 = \dot{\phi}^{\text{endeff}}_{\text{pos}} \ , \tag{5}$$

as well as a constant deceleration/acceleration during a time interval before and after the pick, i.e., *down* and *up* motions,

$$0 = \ddot{\phi}^{\text{endeff}}_{\text{pos}} - a\phi^{\text{object}}_{\text{z}} \ . \tag{6}$$

This means that the endeffector position has a constant deceleration/acceleration exactly in the direction of the object normal, scaled by a positive number a (which we chose $a = 0.1\text{m/sec}^2$ in the experiments). We impose this during a time interval ranging 10% of the pick-to-place interval before and after the pick time slice.

The place constraints are modeled analogously as

$$0 = \phi^{\text{endeff}}_{\text{pos}} - \phi^{\text{dropPoint}}_{\text{pos}} \tag{7}$$

$$0 = \phi^{\text{endeff}}_{\text{z}} - \phi^{\text{world}}_{\text{z}} \ , \tag{8}$$

where the drop point is given by the task, and the endeffector is aligned vertically. As before, we impose zero endeffector velocity at the place time slice, and constant vertical deceleration/acceleration around it.

Note that the pick is under-constrained: The orientation of the object fixation is a decision variable. If we had less constraints on the pick point (e.g., constrain it only to be within a region), also the relative translation between endeffector and object is a free decision variable of the optimization problem. We handle this as in previous work [7], [19], [23]: We introduce respective dofs in those time slices where the object is in hand, to represent the relative pose of the object. Kinematically, the object is attached to the endeffector in those time slices. Assuming a stable grip, we constrain this relative pose to be equal throughout the handling by imposing a zero *relative* pose velocity (in position/quaternion space). In addition we need to add boundary constraints, ensuring that this relative pose is consistent with the initial pick pose, as well as the final drop pose, now by imposing zero *absolute* pose velocity at the kinematic switches.

2) Collision Constraints: To avoid collisions during manipulation we impose inequality constraints on the signed negative distance between pairs (o_1, o_2) of convex shapes,

$$\phi^{o_1 o_2}_{\text{negDist}} \leq 0 \ . \tag{9}$$

3) Control Costs: We can generally penalize sum-of-squares of joint accelerations $\ddot{q}$ or any quantity linear in $\ddot{q}$, which includes $M^{-1}\ddot{q} - F$ (where we assume M and F locally constant, neglecting their Jacobians w.r.t. q). In our experiments we chose a constant diagonal M^{-1} penalizing joint accelerations. As for $\ddot{\phi}$, we define $\ddot{q}$ via finite differencing $(q_t + q_{t\text{-}2} - 2q_{t\text{-}1})/\tau_t^2$ in terms of the discrete time slices. Further, to explicitly aim for shorter paths, we also penalize the sum-of-squares of joint velocities $\dot{q}$.

D. Torque/Force Constraints and Costs

There are various approaches to evaluate forces and torques for a given motion path. First, without introducing further decision variables, we could use the standard robot inverse dynamics equation

$$\theta = M(q, \delta)\ddot{q} + C(\dot{q}, q, \delta) + F(q, \delta) \tag{10}$$

to compute torques for each joint. In our framework we need these torques to be fully differentiable w.r.t. all parameters, which in this case means that we would have to differentiate the inertia matrix M, Coriolis terms C, and gravity term F also w.r.t. the design parameters δ.

In manipulation settings we often have force loops, meaning that the robot endeffector exchanges forces with the environment. In this case it is natural to introduce additional decision variables to the mathematical program to represent this force exchange with the environment [7], and impose constraints to ensure consistency of the exchanged forces with object accelerations (the Newton-Euler equations). We do the same to represent the wrench exchanges between two consecutive robot links. In that way, the solver will optimize for the correct joint torques rather than using the analytical inverse dynamics equation to compute them. This approach is of course less efficient, but it saves us from implementing specialized differentiation of the M, C, F w.r.t. δ.

Specifically, we introduce wrench exchange variables $f \in \mathbb{R}^{6\cdot F}$ in each time slice, between F pairs of links or objects. For each rigid object or link not mounted to the ground we impose the Newton-Euler constraint

$$\begin{pmatrix} \dot{v} \\ \dot{w} \end{pmatrix} + g - M^{-1}F = 0 \ , \tag{11}$$

with the object's linear and angular velocity (v, w), acceleration $(\dot{v}, \dot{w})$, the gravity vector $g \in \mathbb{R}^6$, and the object's inertia matrix $M \in \mathbb{R}^{6\times 6}$.

So far, this only computes the wrenches without imposing costs and, in fact, without influencing the optimization of the path or design parameters δ. To penalize torques around a joint axes we introduce sum-of-squares costs

$$\left\langle f_j, \phi^{\text{joint } j}_{\text{screw}} \right\rangle^2 \ , \tag{12}$$

where f_j is the wrench at joint j, and ϕ_{screw} is the joint axis (actually its 6D screw vector, handling also translational joints), which we can easily differentiate.

IV. Experiments

We evaluate the approach on the two scenarios illustrated in Fig. 1(b,c). In the re-sorting scenarios the robot with suction cup picks objects placed randomly in the center box and places them at random destinations in either the left or right side box. In the wrench tool scenario, the robot aligns the wrench fork with the bolt head then exerts a constant torque while turning it, after which it does the same for the next randomly placed bolt.

The source code for the described methods and to reproduce the following evaluations is available here[2]. The accompanying video[3] illustrates the optimized sequential manipulation trajectories with and without design optimization.

[2] https://github.com/MarcToussaint/21-ICRA-DesignOpt

[3] https://youtu.be/9ihdtMJvcQk

A. Baselines & Metrics

We compare three methods to generate sequential manipulation solutions in these scenarios:

- BASELINE: We use our solver to find a full sequential manipulation path fulfilling the task, but (1) do not allow for deformations of the designs, and (2) only optimize for control costs (path accelerations). Forces and joint torques are not objectives. The respective mathematical program includes wrench decision variables and respective constraints, but only to compute and report them, not to penalize them.
- DESIGNOPT: As the BASELINE, but allowing for deformations of the designs.
- DESIGNOPTTORQUE: Our full approach, which includes also sum-of-square penalization of joint torques in the co-optimization of motion and design deformations.

Our evaluations focus on three metrics:

- *Torques:* The sum-of-squares of joint torques throughout the whole manipulation sequences.
- *Vel:* The sum-of-squares joint velocities throughout the whole manipulation sequences, which relates to path length and maximal execution speed when hardware imposes velocity limits on the joints.
- *Acc:* The sum-of-squares joint accelerations throughout the whole manipulation sequences.

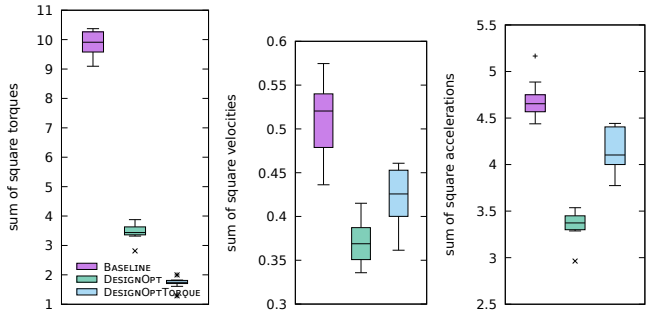
Each single run computes optimal sequential manipulations for a K ensemble of randomized tasks. The $K = 15$ ensembles ensure that the designs do not collapse to be optimal for only a single positioning of objects, but rather for a whole variety of tasks. We repeat each run 15 times with varying random seeds (i.e., for 15 different random ensembles) to allow us to report mean metrics as well as standard deviations.

Each single run optimizes over up to about 50 000 decision variables (dimensionality of x) and takes 10 to 30 minutes on a single CPU, except for BASELINE, which requires 1 to 5 minutes.

B. Re-sorting Scenario

Fig. 1(a) illustrates a robot application in logistics for re-sorting items to boxes, which we mimic with out test scenario Fig. 1(b), which displays the non-deformed original design.

a) Qualitative Discussion: Fig. 3(b) illustrates a deformation due to the DESIGNOPTTORQUE method. One qualitative observation we made was that joint axes were consistently reoriented to become non-orthogonal. They seemed to be optimized so that they more equally contribute to each phase of the manipulation, whereas in the original orthogonal design different joints have very different loads during each phase of the manipulation. An example for this is the first joint, which rotates about the vertical axis in the original design and mostly contributes only to lateral movements. In the optimized deformations, the first joint was significantly tilted as well as non-orthogonal to the following axis.



(a) Torques (b) Velocities (c) Accelerations

Fig. 4: Performance for the re-sorting scenario. Optimizing for design significantly reduces all three metrics. Optimizing additionally for torques yields designs with drastically smaller torques (a), while there is a trade-off with trajectory length and smoothness (b,c).

A second qualitative observation is that when strongly penalizing joint torques (as in Fig. 3(b)), the optimized designs often arrange three axes around the elbow rather close to each other in a non-orthogonal way. The designs seemed to "lean back", where the elbow region seems to counterweight endeffector load, so as to reduce torques on the basis axes.

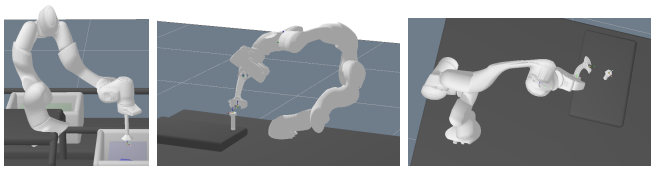
While these are only qualitative observations, they exemplify the range of effects we can achieve with design optimization.

b) Quantitative Discussion: Fig. 4 compares the three methods over 15 randomized runs. We find that design optimization significantly reduces all three metrics. However, including joint torques additionally as an objective makes a large difference as well: The torque metric itself of course reduces significantly for DESIGNOPTTORQUE vs. DESIGNOPT. But this comes with the trade-off of longer and less smooth trajectories. Qualitatively we found that this is due to the system avoiding poses with high static (gravitational) load on the joints.

C. Wrench Tool Scenario

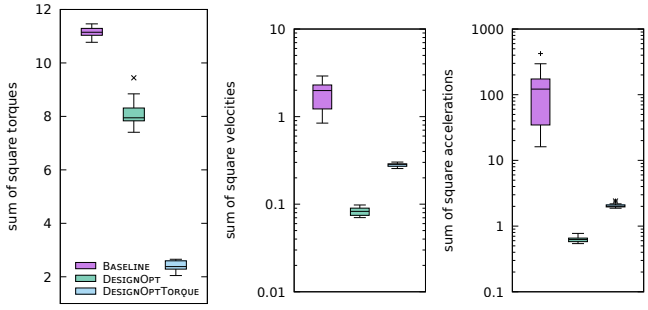
The second scenario is meant to illustrate the optimization of tools and other aspects of the environment. We designed a wrench tool scenario as illustrated in Fig. 1(c), where the robot has to first slide the wrench tool over the tip of a bolt, then turn and exert a constant torque, then retract again and move on to the next randomly placed bolt. The *approach* and *retract* are modeled exactly as the *down* and *up* approaches of pick-and-place, with imposing constant accelerations of the bolt tip in wrench tip coordinates. The design parameter δ included also a deformation of the wrench tip as well as a translation of the pose of the table on which the bolts are fixed. The latter illustrates the possibility to include station design parameters in the co-optimization.

a) Qualitative Discussion: In this scenario, DESIGNOPT and DESIGNOPTTORQUE had large effects on the robot design. Fig. 5(b) displays an extreme case, where the design was optimized for one particular bolt placement. The



(a) (b) (c)

Fig. 5: Example designs for both scenarios.



(a) Torques (b) Velocities (c) Accelerations

Fig. 6: Performance for the wrench tool scenario.

design highly specialized on applying the required wrench by a pull maneuver; in the leaning back design gravitational loads cancel with the pulling torques to minimize overall torque penalties in the joints. The wrench tool and links were elongated and bent to simplify exerting the torque. In a second example, Fig. 5(c), we also allowed the board positioning to be deformed and optimized for 10 random placements on the board. The board was moved further away, the lower arm and wrench tool were elongated significantly, and the torque motion mostly generated by lower axes movements; the twisted wrench tool translates a pull to torque.

b) Quantitative Discussion: Fig. 6 compares the three methods over 15 randomized runs. The results match our findings for the re-sorting scenario, including the trade-off between torque optimization and longer and less smooth trajectories, but are more emphasized. Note the log scale for velocities and accelerations. Qualitatively we found this to be due to the particular movements necessary in the wrench tool scenario, which seem near the edge of feasibility for some bolt placements for the original design.

V. Conclusions

We proposed a method for co-optimization of robot, environment and tool designs with the resulting sequential manipulation trajectories. Our evaluations show that design optimization can significantly improve on metrics such as penalizing velocities (path length) and joint torques. The wrench tool scenario demonstrated that the solver can find interesting strategies to exert the necessary external forces with minimal effort. The re-sorting scenario aimed to demonstrate the high relevance of tailored robot station design for industrial applications. While the current hardware industry cannot yet deliver systems with custom link shapes, a modular actuator design combined with 3D-printed link structures of optimized shapes is highly promising and would bring our robot design methods directly to application, enabling tailored robot designs for each specific industrial task ensemble.

A highly interesting endeavor for future research is to also consider discrete design decisions, such as the number of joints, for co-optimization. This could be tackled by hybrid optimization approaches, such as logic-geometric programming [23] or other branch-and-bound approaches. We thank the insightful reviewers to hint at this.

References

[1] M. Mason, "The mechanics of manipulation," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference On*, vol. 2. IEEE, 1985, pp. 544–548.

[2] C. Eppner, G. Bartels, and O. Brock, "A compliance-centric view of grasping," DOI 10.14279/depositonce-5050, Feb. 2012.

[3] C. Paul, "Morphological computation: A basis for the analysis of morphology and control requirements," *Robotics and Autonomous Systems*, vol. 54, no. 8, pp. 619–630, 2006.

[4] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass, "Towards a theoretical foundation for morphological computation with compliant bodies," *Biological cybernetics*, vol. 105, no. 5-6, pp. 355–370, 2011.

[5] A. Verl, A. Albu-Schäffer, O. Brock, and A. Raatz, *Soft Robotics*. Springer, 2015.

[6] R. Pfeifer, M. Lungarella, O. Sporns, and Y. Kuniyoshi, "On the information theoretic implications of embodiment–principles and methods," in *50 Years of Artificial Intelligence*. Springer, 2007, pp. 76–86.

[7] M. Toussaint, J.-S. Ha, and D. Driess, "Describing physics for physical reasoning: Force-based sequential manipulation planning," *IEEE Robotics and Automation Letters*, 2020.

[8] H. Wicaksono and C. Sammut, "A learning framework for tool creation by a robot," in *Proceedings of ACRA*, 2015.

[9] O. Taylor and A. Rodriguez, "Optimal shape and motion planning for dynamic planar manipulation," *Autonomous Robots*, vol. 43, no. 2, pp. 327–344, 2019.

[10] R. M. Holladay, "Force-and-motion constrained planning for tool use," PhD Thesis, Massachusetts Institute of Technology, 2019.

[11] J. Whitman and H. Choset, "Task-specific manipulator design and trajectory synthesis," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 301–308, 2018.

[12] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Computational co-optimization of design parameters and motion trajectories for robotic systems," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1521–1536, 2018.

[13] O. Taylor and A. Rodriguez, "Optimal shape and motion planning for dynamic planar manipulation," *Autonomous Robots*, vol. 43, no. 2, pp. 327–344, 2019.

[14] L. Nair, N. Shrivatsav, and S. Chernova, "Tool macgyvering: A novel framework for combining tool substitution and construction," *arXiv preprint arXiv:2008.10638*, 2020.

[15] L. Nair, N. S. Srikanth, Z. M. Erickson, and S. Chernova, "Autonomous Tool Construction Using Part Shape and Attachment Prediction." in *Robotics: Science and Systems*, 2019.

[16] D. Driess, O. Oguz, and M. Toussaint, "Hierarchical task and motion planning using logic-geometric programming (hlgp)," RSS Workshop on Robust Task and Motion Planning, 2019.

[17] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *The Int. Journal of Robotics Research*, vol. 37, no. 1, pp. 104–136, 2018.

[18] T. Simon, J.-P. Laumond, J. Corts, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The Int. Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, 2004.

[19] M. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Proc. of Robotics: Science and Systems (R:SS 2018)*, 2018.

[20] M. Stilman and J. Kuffner, "Navigation among movable obstacles: real-time reasoning in complex environments," in *4th IEEE/RAS Int. Conf. on Humanoid Robots, 2004.*, vol. 1, 2004, pp. 322–341 Vol. 1.

[21] J. E. King, M. Cognetti, and S. S. Srinivasa, "Rearrangement planning using object-centric and robot-centric action spaces," in *2016 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 3940–3947.

[22] M. Toussaint, "A tutorial on Newton methods for constrained trajectory optimization and relations to SLAM, Gaussian Process smoothing, optimal control, and probabilistic inference," in *Geometric and Numerical Foundations of Movements*, J.-P. Laumond, Ed. Springer, 2017.

[23] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA 2017)*, 2017.