Shared Autonomy for Enhancing Trajectory Optimization

Christopher E. Mower[1], João Moura[1], Theodoros Stouraitis[1], Sethu Vijayakumar[1,2]

***Abstract*— Tasks such as non-prehensile manipulation (e.g. pushing) pose difficult computational challenges for robots. It is difficult to develop a reactive controller capable of completing these tasks within complex and unpredictable environments. In contrast, humans typically complete these tasks repeatably and with ease. The goal of this paper is to describe an approach we have been developing to incorporate human input via a shared autonomy interface to improve solver performance (e.g. convergence). We provide early empirical evidence that our approach leads to higher success rates for planned motions, and fewer iterations for the solver.**

I. INTRODUCTION

Planning presents several challenges for generating motion in realistic scenarios (e.g. involving contact). Trajectory optimization has been shown to be a promising framework since, in a single formulation, it can take into account multiple task objectives and constraints [1], [2], [3]. However, contact introduces high non-linearity and discrete changes in the cost landscape, dynamics, and constraint formulation [4], [5], [6]. This tends to increase the computational burden on the solver making it difficult to implement Model Predictive Control (MPC) for handling dynamic changes in the task and environment.

In contrast, human beings exhibit a myriad of complex, adaptable motor skills, for instance: non-prehensile manipulation and locomotion - our interest, however, in this paper is manipulation. Adapting to changes in the environment, different object geometries and topologies, and physical parameters is somehow handled naturally and almost instantaneously by the brain. Our goal is to harness this innate ability through an intuitive shared autonomy interface so that the system can take advantage of the human's natural aptitudes and improve solver performance.

The central thesis of this work is to ask: can we develop *practical methods* that allow us to transfer human-level capabilities to robots for *generalizeable* manipulation tasks? Practical methods means that the interface for the human should be intuitive, and allow the human to transfer their capabilities to the system. Generalizable refers to the fact that we want to take into account several task descriptions.

We have developed this work in the context of a robot shelving scenario. The goal for the robot is to place an object on a shelf that is out of reach for the robot - in the sense that planning a pick and place motion is insufficient. Thus, the robot must exploit contacts with the object, pushing and frictional contacts with the environment, and contact switches between contact locations. Furthermore, the system should reason about the contact locations on an arbitrary object.

Fig. 1: The Kawada Nextage humanoid robot performing a non-prehensile pushing task in a smart factory mock-up.

In this paper we explore preliminary work towards realizing this goal. Our investigation spans several approaches that map input from the human onto a solver in order to improve solver performance. We provide an analysis of the advantages and disadvantages of these approaches and provide insight for future development.

Ideas presented here were in-part inspired by the field of image processing; namely the GrabCut algorithm for foreground extraction [7]. Our insight is that prior information from a human can lead to high algorithmic performance (i.e. fast convergence). In a similar sense, we aim to develop a shared autonomy interface that improves the performance of trajectory optimization solvers, at the cost of minimal interactive effort. We present the proposed approach and several experiments in simulation highlighting the potential benefits.

II. PROBLEM FORMULATION

We can formulate a trajectory optimization problem with discrete variables as

$$x^*, u^*, z^* = \arg\min_{x,u,z} \ \phi(x,u,z) + \int_{\mathbb{T}} \psi(x,u,z)\ dt \qquad \text{(1a)}$$

subject to

$$\dot{x} = f(x,u,z), x \in \mathbb{X}, u \in \mathbb{U}, z \in \mathcal{Z} \qquad \text{(1b)}$$

[1]School of Informatics, University of Edinburgh, United Kingdom.
[2]The Alan Turing Institute, London, United Kingdom.

This research is supported by The Alan Turing Institute and has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017008, Enhancing Healthcare with Assistive Robotic Mobile Manipulation (HARMONY).

Contact: {cmower, joao.moura, theodoros.stouraitis, sethu.vijayakumar}@ed.ac.uk.

where $x \in \mathbb{R}^{n_x}$ are system states, $u \in \mathbb{R}^{n_u}$ are controls, $z \in \mathbb{Z}^{n_z}$ are integer variables (e.g. mode), $\phi(\cdot)$ is a cost term modeling the terminal state, $\psi(\cdot)$ is a summation of weighted terms integrated across time $\mathbb{T} = [0, T]$ such that T is the final time, f is the equations of motion (i.e. system dynamics) integrated over $\mathbb{T}$, and $\mathbb{X}, \mathbb{U}, \mathcal{Z}$ are the feasible regions for x, u, z respectively. The constraints (1b) are described by a set of equality and inequality constraints.

Solving (1) directly using a mixed-integer solver is typically intractable for online computation. Furthermore, highly nonlinear dynamics and cost functions can cause convergence issues (e.g. local minima). Approaches to alleviate these issues include warm-starting the optimization [8] learned mode selection [6], and landscape smoothing [9].

Let $h \in \mathbb{R}^{n_h}$ denote input from a human via some interface (e.g. joystick). The goal of our work is to harness the human's innate ability to perform complex tasks through h. Our approach aims to map h onto (1) in order to improve solver performance.

III. RELATED WORK

Data driven approaches for finding warm-start initial seeds has been shown to produce improvements in solver performance [10], [8], [11]. The methods in [10], [8] have difficulty extrapolating outside the observed database, and mixed-integer approaches, as in [11], are difficult to scale.

The concept of heuristic models were shown by Bledt and Kim to improve controller performance in locomotion [9] - improved performance in this case means the controller was robust to disturbances. Their work explored the use of heuristic models as warm-start initial seeds and regularization. The models were generated through a data driven approach from examples collected in simulation. Work presented here is inspired by their approach, whereas we explore generating heuristic methods via human interaction with a shared autonomy interface.

IV. PROPOSED APPROACH

In this section we describe our proposed approach that maps human input onto the optimization solver with the intent of improving performance. First, we describe the notion of a model that is used to introduce heuristics to the optimizer. Next, the use of these models, in order to provide warm-start initial seeds, is described. Finally, we show how the models can be used as a regularization to simplify the cost landscape.

A. Heuristic models

As opposed to traditional optimal control methods, our proposed approach utilizes heuristic models to simplify the complex cost landscape, and provide a warm-start initial seed. A similar idea was utilized in [9] who learned these models from data. A heuristic model $H(\cdot)$ is given by

$$x_H, u_H, z_H = H(h) \tag{2}$$

where x, u, z, h are as described in Section II. The human input h is mapped to the state/control/integer space.

To be effective, the GrabCut algorithm requires only a bounding box surrounding the feature of interest. In a similar fashion to the GrabCut algorithm, we do not require that the human input maps to a necessarily feasible trajectory. That is, if x_H, u_H, z_H was executed on a real system, then the resulting motion may not be optimal.

B. Initial seed warm-starting

The trajectory optimization problem described in (1) can be thought of as a mapping $TO(\cdot)$ as follows

$$X^* \leftarrow TO(X^0, \Omega) \tag{3}$$

where $X^* = (x^*, u^*, z^*)$ is the locally optimal solution, $X^0 = (x^0, u^0, z^0)$ is the initial guess, and $\Omega \in \mathbb{R}^{n_\omega}$ are problem parameters. Under this formalism, the heuristic model (2) can be substituted into (3), i.e. $TO\big(H(h), \Omega\big)$.

C. Regularization models

The problems of interest to us (e.g. placing an object on a high shelf) involves discrete variables, and nonlinear cost landscapes. This type of problem can lead to slow convergence for a solver. Including regularization terms is an established approach to ease the computational burden on the solver. It can also be used to treat problems such as over-fitting - a common issue from the Machine Learning community.

Our problem however is how to choose appropriate regularization terms. Typical choices, for example, are to minimize controls u by including $\|u\|$ as an additional term for ψ in (1). We introduce the regularization term $\rho(\cdot)$ into the cost function (1a), given by

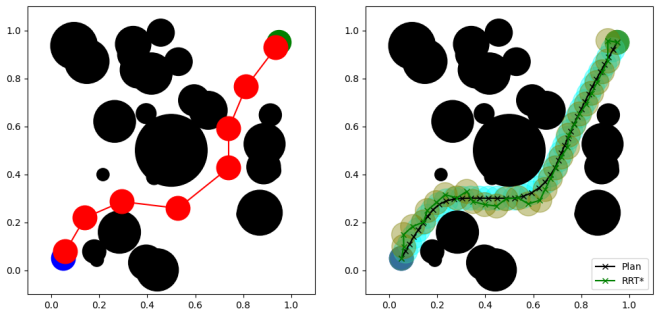
$$\rho(x, u) = \xi_x \|x - x_H\|^2_{W_x} + \xi_u \|u - u_H\|^2_{W_u} \tag{4}$$

where $0 < \xi_x, \xi_u \in \mathbb{R}$ are scaling parameters reflecting the relative importance of each of the respective terms, $\|\cdot\|$ is the weighted Euclidean vector norm with positive definite weighting matrices W_x, W_u, and x_H, u_H are given by the heuristic model (2).

D. Reduce the number of decision variables

The duration of optimization solvers is dependent on the number of decision variables that defines the problem. An optimization problem with many decision variables will take longer to find a solution than one with fewer. Previous work, for instance, has enabled MPC for shared autonomy by parameterizing the state and control trajectories [12]. By utilizing such a parameterization allows the problem to be solved in a reduced dimensional space. In [12], a parameterization was enabled from access to prior knowledge from the task - certain motion patterns were assumed to be the goal of the human operator. Such assumptions on the resulting robot motion are hard to make in our problem formulation.

In our case, however, we propose that certain variables could be traded off to the human. For example, a source of high computational burden when solving (1) is the discrete decision variables z. This requires the use of mixed-integer solvers such as branch-and-bound [13]. In the case



(a) (b)

Fig. 2: Navigation through clutter experimental setup. (a) The viewer presented to the human. Blue/green circle is the start/goal positions. Black circles are obstacles, and red circles are the input from the human. (b) The computed plan (cyan) against plan found by the RRT* solver (green).

of manipulation, contact switches, for example, require a sequence of integer variables that determine whether the robot end-effector is in contact ($z = 1$) or not ($z = 0$) at each point in time [14]. One way that the human input could be utilized in this type of problem is that the human, through the shared autonomy interface, could specify the contact sequence. Ultimately, this could be imposed on the problem by adding the additional constraint $z = z_h$ where z_h is given by (2). In this case, mixed-integer solvers are not required since the problem has been modified such that it is continuous. The modified problem can thus take advantage of efficient solvers such as IPOPT [15] and SNOPT [16].

V. EXPERIMENTS

This section describes the experiments that were performed to analyze our proposed work. All experiments were run in simulation on a PC running Ubuntu 20.04, with a 16-core Intel(R) Core(TM) i9-9900KF CPU @ 3.60GHz. All the code was implemented in the Python programming language using the CasADi framework [17]. The IPOPT solver [15] was used to solve the optimization problems.

A. Navigation through cluttered environments

We first analyze our proposed approach in the context of a navigation through clutter problem. The goal of the experiment is to analyze if using human input as the initial seed and/or as a regularization improves solver performance. We compare these against a standard straight-line initial seed.

To represent the robot we use a point-mass model and circles for obstacles in the scene. For simplicity we omit discrete variables z in this experiment. The optimization models the problem of finding a collision free path between a start and end state. The cost function is defined by $\phi = 0$ and $\psi = \|u\|$. The dynamics $f = u$ is integrated using the Euler integration method. The constraints specify the initial state, goal state, box limits on the states, and obstacle avoidance.

Under the aforementioned system, we compare three conditions: (C1) straight line initial seed, (C2) human input as the initial seed, (C3) human input as the initial seed and regularization term. Notice that the cost function in (C1/2) are the same.

Several obstacles in the environment are randomly generated. The obstacle positions and radii are uniformly sampled in the planning space. We use an RRT* planner [18] to check that at least one collision-free path exists between the start and goal positions

A trial of the experiment consists: (i) randomly generate an environment with n obstacles, (ii) solve (1) under (C1) above, (iii) collect human input using the GUI shown in Figure 2a, (iv) solve (1) under conditions (C2) and (C3). In addition, we record whether the solver fails or succeeds. Higher performance for the solver is indicated by the number of iterations, and higher success rate. An example of a solution is shown in Figure 2b. In total, 130 data points were collected.

The results for the experiment are shown in Figure 3. We plot the scatter diagrams for each condition and also plot the line-of-best-fit. The gradient m and its corresponding sum of squared residuals of the least squares fitting ρ are also reported. Lower values for the gradient m indicate higher performance, e.g. for the iteration count it suggests that even for higher number of obstacles the solver completed in a faster duration. The sum of squared residuals indicate the spread of the data, i.e. the smaller its value the more correlated the variables are considered.

It should also be noted that (C1) has a significantly smaller success rate when compared with the other two conditions. These results indicate that using the human input as an initial seed dramatically improves success rate for the solver.

The results for the iteration counts, (C3) has the shallower gradient of the three conditions. These results suggest that using the human input leads to improved convergence of the solver.

B. Planning non-prehensile pushing tasks

In this section, a similar experiment, shown in Figure 4, is described for a more complex task that includes discrete variables. The task used is a non-prehensile pushing task, i.e. the robot must push an object from a start to goal position utilizing switching contacts.

The system model used here is based on the work by Stouraitis et al., see [14] for full details. The main difference is that we have removed the gravity component, and introduced a friction component for the object representation.

In order for a human to provide input h they require an interface to the system. We use a three axis joystick for the physical interface. The human manipulates a virtual realization of the pusher and box. The pusher is in either of two modes: in-contact (trigger pressed) where the pusher P is modeled by a point-mass under velocity control, or not in-contact (trigger released) where the fixed-contact system dynamics, as in [6], is implemented.

This interface allows us to collect estimated state trajectories for the object and pusher, action timings, wrenches on the object, and the contact sequence. However, the contact forces are difficult to estimate from the collected data. This mapping constitutes our heuristic model $H(\cdot)$ that maps the

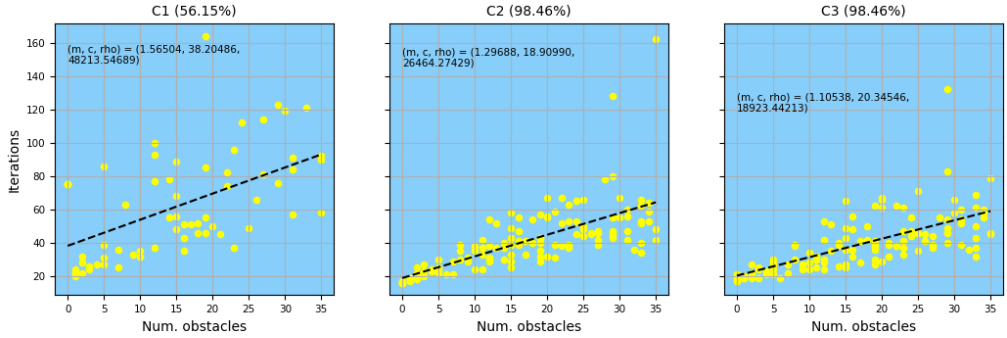


Fig. 3: Scatter plots for the navigation through clutter experiment showing number of obstacles against number of iterations. The columns are results for each condition. In addition, the title reports the success rate for each condition. For each plot, the line of best fit (m, c, ρ) is shown - m is the gradient, c is the intercept, and ρ the sum of squared residuals.

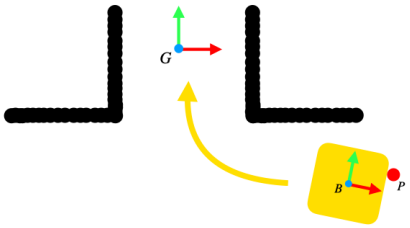


Fig. 4: Pusher P must push the object B to a goal pose G while avoiding collision.

human input signals h to a state/control trajectories x_H, u_H, and the contact sequence z_H

Similarly to the previous experiment, we compare the same three conditions. For each of the three conditions (C1-3) we have implemented our solver for a pushing task.

The number of iterations required to solve the problem for the three conditions are as follows: (C1) required 770, (C2) required 511, and (C3) required 655. Regarding resulting motion, it was noted that for (C1) and (C2) the box traveled a further distance. For (C3), the motion was similar to the demonstration, however this is expected due to the regularization term.

VI. CONCLUSION

In this paper, we have presented an approach for improving solver performance for trajectory optimization utilizing a shared autonomy interface. The tasks we have investigated and discussed are indeed solvable by existing methods, however we have illustrated the potential for our proposed concept. Our analysis provides early evidence that input from human operators can improve solver performance by being mapped through heuristic models. Future work will explore the scalability of the proposed approach to more complex tasks (e.g. placing an object on a shelf).

REFERENCES

[1] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.

[2] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[3] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[4] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[5] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[6] F. R. Hogan and A. Rodriguez, "Reactive planar non-prehensile manipulation with hybrid model predictive control," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 755–773, 2020.

[7] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": Interactive foreground extraction using iterated graph cuts," in *ACM SIGGRAPH 2004 Papers*, ser. SIGGRAPH '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 309–314. [Online]. Available: https://doi.org/10.1145/1186562.1015720

[8] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of motion for warm-starting trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2594–2601, 2020.

[9] G. Bledt and S. Kim, "Extracting legged locomotion heuristics with regularized predictive control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 406–412.

[10] W. Merkt, V. Ivan, and S. Vijayakumar, "Leveraging precomputation with problem encoding for warm-starting trajectory optimization in complex environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5877–5884.

[11] F. Eiras, M. Hawasly, S. V. Albrecht, and S. Ramamoorthy, "A two-stage optimization-based motion planner for safe urban driving," *IEEE Transactions on Robotics*, 2021.

[12] C. E. Mower, J. Moura, and S. Vijayakumar, "Skill-based Shared Control," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[13] M. Conforti, G. Cornuejols, and G. Zambelli, *Integer Programming*. Springer Publishing Company, Incorporated, 2014.

[14] T. Stouraitis, I. Chatzinikolaidis, M. Gienger, and S. Vijayakumar, "Online hybrid motion planning for dyadic collaborative manipulation via bilevel optimization," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1452–1471, 2020.

[15] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar 2006.

[16] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM Rev.*, vol. 47, no. 1, p. 99–131, jan 2005.

[17] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, Mar 2019.

[18] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, "Pythonrobotics: a python code collection of robotics algorithms," *CoRR*, vol. abs/1808.10703, 2018. [Online]. Available: http://arxiv.org/abs/1808.10703