User-Guided Offline Synthesis of Robot Arm Motion from 6-DoF Paths

Pragathi Praveena[1], Daniel Rakita[1], Bilge Mutlu[1] and Michael Gleicher[1]

***Abstract*— We present an offline method to generate smooth, feasible motion for robot arms such that end-effector pose goals of a 6-DoF path are matched within acceptable limits specified by the user. Our approach aims to accurately match the position and orientation goals of the given path, and allows deviation from these goals if there is danger of self-collisions, joint-space discontinuities or kinematic singularities. Our method generates multiple candidate trajectories, and selects the best by incorporating sparse user input that specifies what kinds of deviations are acceptable. We apply our method to a range of challenging paths and show that our method generates solutions that achieve smooth, feasible motions while closely approximating the given pose goals and adhering to user specifications.**

I. Introduction

Computing robot joint configurations such that the motions executed by its end-effector match a given path is useful in scenarios such as having a robot reproduce a measured human demonstration [1], [2], transferring a motion from one robot to another [3], [4], or having a robot follow a designed path [5]. While accurately matching the robot end-effector to the 6-degree of freedom (6-DoF) position and orientation goals along the path is important, it is also crucial that the resulting robot motion is *feasible* – that is, there should be no self-collisions, joint-space discontinuities or singular configurations. Furthermore, it is possible that there is no feasible sequence of robot configurations that result in exactly following the given path. In practice, exact matches are typically not necessary. There are often many approximate solutions, with different tradeoffs. Providing for user control of importance criteria can lead to more effective selection of acceptable solutions. For instance, if the robot is carrying a loaded tray, the user may prefer errors in position to errors that cause the tray to tilt, or specify that positions during movement are less important to be precise than when the tray is being placed.

In this paper, we introduce an approach that creates robot trajectories in configuration space that closely follow a 6-DoF Cartesian workspace path within user-specified tolerances while accommodating kinematic constraints. The novelty of this method rests on two ideas: (1) the capacity to deviate from the input path to meet feasibility criteria, and (2) provision of control over this deviation for the user. We realize our approach using point-to-point methods: applying nonlinear optimization at every time step to minimize Cartesian-space errors while enforcing motion goals like avoidance of self-collisions and singularities, maintaining smoothness by regularization with outputs from previous time steps, and relaxing certain goals in order to meet other goals deemed more important by the user. To improve robustness and find more accurate solutions, our approach tries many alternative starting points to create a sampling of candidate trajectories. Our methods provide control over tolerances, critical frames, and the relative importance of different axes. This allows a user to guide the algorithm to select trajectories that either meet accuracy requirements or have the least-problematic errors.

This paper builds on prior work [6] that provides a real-time point-to-point inverse kinematics solver called *RelaxedIK* that allows pose goals to be relaxed in order to achieve feasibility goals. This paper extends the *RelaxedIK* solver to more fully address the offline path following problem. Our approach uses *RelaxedIK* to generate multiple candidate trajectories which are evaluated to select the best. This provides more precise solutions and improved robustness (i.e., the method is more likely to find a solution that meets specified tolerances). Our approach also provides additional controls that allow a user to specify what kinds of errors are most acceptable in cases where perfect solutions are not achieved. We evaluate our approach by showing results of creating suitable trajectories for two robot platforms, the Universal Robots UR5 (6-DoF) and Rethink Robotics Sawyer (7-DoF), for a range of challenging paths.

II. Related Work

Our user-guided method for synthesis of feasible motion draws from prior work in robotics and animation, primarily from motion planning, trajectory optimization and constraint-based motion synthesis methods.

Hauser [7] considers the problem of creating continuous mappings from workspace paths to robot trajectories. He notes that many robots do not have global continuity, even when they have redundancy; meaning that for some desired paths, there may be no feasible (i.e., continuous) solutions. This motivates methods, such as this paper, that seek feasible paths at the expense of exact path following. Whereas Hauser attempts to find one-to-one mappings from end-effector space to robot configurations, we instead consider inexact mappings that necessarily map from goals to regions of configuration space, and use the flexibility to choose within these regions to obtain feasible trajectories.

One approach for path following is sampling-based algorithms (e.g., [8], [9], [10]) that search for a solution

[1]Pragathi Praveena, Daniel Rakita, Bilge Mutlu and Michael Gleicher are with the Department of Computer Sciences, University of Wisconsin–Madison, Madison 53706, USA [pragathi|rakita|bilge|gleicher]@cs.wisc.edu

This research was supported in part by NSF award 1830242 and a UW2020 award from the University of Wisconsin–Madison Office of the Vice Chancellor for Research and Graduate Education.

trajectory by selectively sampling and exploring the space of collision-free and feasible motions. While these methods are successful in producing feasible solutions, they may be computationally inefficient [11], [12] or produce jerky trajectories that need to be post-processed [13]. Other recent work [14], [15] addresses similar problems with graph search algorithms that seek to provide optimal solutions, but do not provide user control over the trajectories generated.

Another approach employed for path following [16], [17] is to choose sparse sample points (i.e., keyfames) and rely on configuration space interpolation to create piecewise continuous trajectories. Such an approach is problematic because it provides little guarantee on the movements between the keyframes: the interpolated paths must still be checked to confirm that they are feasible (e.g., no collisions) [18], and they may not approximate a desired path closely.

A third approach is to consider the whole trajectory holistically [19], [20], [21], treating the entire mapping problem as a single optimization problem over the entire trajectory. Such an approach is valuable because choices on any particular frame must be made in a coordinated manner. A poor choice on an early frame may lead to a motion that cannot achieve a later goal. Unfortunately, the need to consider the entire motion as one large problem leads to scalability concerns. In theory, such global approaches can provide feasible and accurate solutions when they exist. In practice, such approaches are often inefficient and require good starting points. While our approach cannot provide the guarantees of sophisticated global methods, it can complement these approaches by providing quick solutions for common cases, reserving the complex solvers for when they are needed. It can also provide a set of good starting points for a more global method to work with.

Luo and Hauser [22] consider a similar problem to match workspace paths to robot trajectories. Their method represents trajectories using B-Spline interpolation to insure smoothness. Their approach performs a single, global optimization over the entire motion to best configure the parameters to approximate the input motion. However, their approach does not consider orientation goals or self-collisions; these features would be difficult to add to the large and complex non-linear optimizations. Their method also does not provide any user control over accuracy.

Many of the methods from animation discussed above incorporate user-defined constraints in the definition of the trajectory optimization problem. Some work in robotics (e.g., [23], [24], [25]) considers user-guidance in the configuration space. Our method includes control over the importance of certain features, and we show the merit of this addition with examples in Section V.

III. Problem Definition and Approach Overview

Our problem is to transform an input Cartesian workspace path sampled as $\{\mathbf{x}_1, \mathbf{x}_2 ... \mathbf{x}_m\}$, to an output configuration space trajectory, $\{\mathbf{q}_1, \mathbf{q}_2 ... \mathbf{q}_n\}$. At any given time t, the path has a position and orientation, $\mathbf{x}_t \in SE(3)$. The output configuration space is a vector of values for each of the P parameters (e.g., joint angles) of the robot, $\mathbf{q}_t \in \mathbb{R}^P$.

We assume the input is a Cartesian workspace path represented as a sequence of frames with six degrees of freedom. We assume that the input path is sampled densely enough (in time) that it well-approximates a smooth path, i.e., neighboring frames are close together so any simple interpolation between configurations is sufficient.

In creating the output sequences, there are three goals. First, at each frame, a designated coordinate frame (e.g., the end-effector) on the robot should match the corresponding input frame within tolerances prescribed by the user. Second, the configuration at each frame should be feasible, i.e., free of self-collisions and kinematic singularities, and away from joint limits. Third, the output sequence should be smooth, in a discrete sense.

We additionally require consecutive configurations to be close in configuration space, which addresses two concerns. First, even though two consecutive output configurations are individually feasible, the interpolation between these feasible states may pass through problematic ones. Therefore, we require configurations to be sufficiently far from problems (e.g., self-collisions and singularities) so that the interpolation between them will also be feasible. Second, while it is possible for close end-effector positions to be achieved with configuration-space distant configurations, it is unlikely that interpolating these distant states will keep the end-effector close; even if it does, these large arm motions are usually undesirable. Therefore, our approach places bounds on either the configuration-space distance between consecutive frames, or the magnitude of the finite-difference approximations to the higher derivatives.

A naïve approach to the mapping problem would apply an inverse kinematics solver to compute a configuration that achieves the target position and orientation for each input frame. Such an approach ignores the need for feasibility and continuity. A recent approach, *RelaxedIK* [6], provides feasible solutions by extending the IK solver to consider kinematic feasibility and to create smoothness by regularizing with previous frames. By *relaxing* the requirement of exactly matching the end-effector target pose, the method can select from within a range of solutions that come close and result in feasible and smooth trajectories. Applying *RelaxedIK* successively to each frame provides an effective solution to the path mapping problem. The empirical tests in Rakita et al. [6] show the method is able to create feasible motions (e.g., sufficiently smooth as well as avoiding singularities and self-collisions) with better accuracy than other state-of-the-art motion synthesis approaches. The sequential application of *RelaxedIK* is online (it does not use information about the future) and fast. However, it often fails to provide accurate solutions in challenging cases because it cannot look ahead or take time to explore alternatives. Therefore, it creates unnecessary deviations from path goals in order to preserve feasibility.

In this paper, we provide an application of *RelaxedIK* to better address the offline path mapping problem. Our

approach exploits the offline nature to achieve higher precision and greater robustness. First, we use the ability to look ahead to future frames to better sample the solution space. Our approach effectively applies sequential *RelaxedIK* in different ways to provide many candidate trajectories in order to find one that best meets the goals. Second, we provide for user control over the tradeoffs in the solutions by specifying what kinds of errors should be avoided. We first review the *RelaxedIK* solver in Section IV and detail our specific adaptations in Section V.

IV. Relaxed IK

The core of our approach is the *RelaxedIK* solver of Rakita et al. [6]. It is an optimization-based inverse kinematics solver that determines robot configurations (joint angles) given a 6-DoF Cartesian end-effector target pose and awareness of feasibility constraints.

As formulated in Equation 1, *RelaxedIK* minimizes a weighted sum of objectives that encodes various sub-goals such as end-effector position and orientation matching, and avoiding discontinuities by minimizing backward differences up to the third order between consecutive frames. There are also constraints to maintain distance from self-collision, stay within the joint limits and avoid singularities.

$$\begin{aligned} \Theta = \arg\min_{\Theta} \quad & \sum_{s \in S} w_s \ * f_s(\Theta, \Omega_s) \\ \text{s.t. } & \mathbf{c}_i(\Theta) \ \geq \ \mathbf{0}, \quad \mathbf{c}_e(\Theta) \ = \ \mathbf{0} \\ & l_j \leq \Theta_j \leq u_j, \forall j \end{aligned} \tag{1}$$

Here, Θ is the current robot configuration, $f_s(\Theta, \Omega_s)$ is an objective term that encodes a single sub-goal (s) with Ω_s being the model parameters, and w_s is a static weight value that determines the relative importance of the sub-goal. $\mathbf{c}_i(\Theta)$ is a set of inequality constraints, $\mathbf{c}_e(\Theta)$ is a set of equality constraints, and l_j and u_j values define the upper and lower bounds for the robot's joints.

V. Improved Solutions for Offline Problems

In this section, we describe our methods for offline generation of trajectories from given paths. Our approach extends the online approach (i.e., sequential application of *RelaxedIK* as described above) by improving the choice of starting points for the iterative optimization, considering starting frames other than the beginning, and allowing for control of the error tradeoffs.

A. Starting Configurations

The iterative optimization in *RelaxedIK* is sensitive to the starting point, which is a hyper-parameter to the solver and provided in the form of a *starting configuration*, $\mathbf{q}_{start}$. A different starting configuration may result in a very different solution trajectory. Therefore for the first frame of the sequence, our approach tries a collection of starting configurations and selects the one that achieves the best solution. A preprocessing step, performed once for any particular robot, determines a list of configurations that is used for all subsequent applications of our method. To generate this list, the preprocessor samples a large number of random targets within the workspace and creates a list of solutions (one per target) that have low position and orientation error (0.01 mm and 0.01°). It reduces this list to 30 representative configurations by applying a maximal different subset algorithm [26].

For subsequent frames, our approach uses the solution for the previous frame as a starting point, and includes regularization terms to match prior frames in order to create smooth paths. After finding this initial solution, our method re-runs the IK solver with the previous solution as a starting point and smoothness regularization disabled. In practice this often leads to more accurate solutions that have acceptable smoothness, see Figure 1 for an example.

Our approach checks each solution to ensure it meets the continuity criteria of being close enough to the prior frame. Failure is rare for the regularized solutions, but happens occasionally for the iterated solutions. If the size of the discontinuity is small (but above the tolerance), our approach has an option to insert interpolated goals between the previous and current time steps. The interpolated goals are created by linear interpolation of the positions and spherical linear interpolation [27] of the orientations. An example is shown in Figure 2. This feature creates timing differences between the input and output paths, and may be disabled. If an unregularized solution is too far from the previous frame, the initial regularized solution is used. If the initial regularized solution is too far from the previous frame's solution, our approach uses the previous frame's solution enforcing temporal feasibility at the expense of matching. This tradeoff will be controlled in later steps.

B. Initial Frame

Our approach treats the first frame of the sequence specially: using the starting-configuration sampling method described in Section V-A to find accurate solutions, and then proceeding from this frame. This approach is greedy: it provides for high accuracy on the first frame, but the selection of the first frame may lead to poor performance later in the sequence. If the greedy process from the beginning of the motion fails to provide an acceptable solution, our approach applies two other methods to find improved solutions: use of different initial frames and the testing of multiple paths.

While the first frame of the motion ($\mathbf{x}_1$) is the most natural place to start the sequential solution, we can, in fact choose any frame to start and work sequentially forwards and backwards from that frame, which we term as the *initial frame*. Our approach is to try multiple initial frames in order to find one that yields an acceptable solution. For candidates, we select frames that the user designates as requiring high accuracy (see user control in Section V-C), as well as frames with large errors in prior solutions (often caused by the re-use of previous frames to maintain smoothness as described in Section V-A). We restrict candidate frames to be at least 50 time steps apart. Figure 3 shows an example where

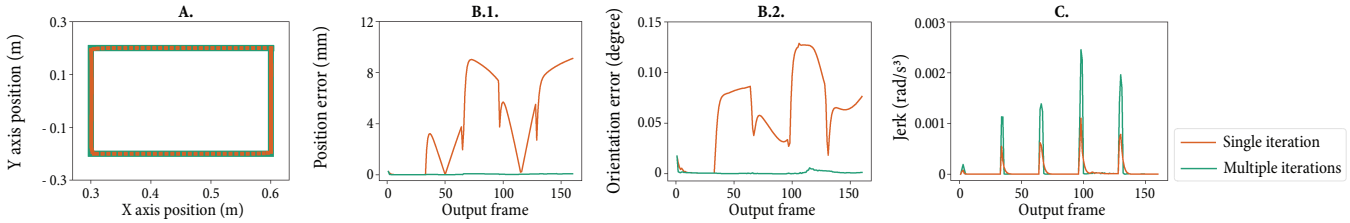

Fig. 1. **Rectangle tracing:** Comparison between trajectories generated using single (orange) and multiple (green) iterations of the IK solver. **A. Top view:** Overall, both solutions follow the path closely. **B. Errors:** The single iteration strategy yields solutions that have similar errors to prior work [6] (orange). The multiple iteration strategy by disabling smoothness regularization can reduce errors to less than 0.1 mm and 0.05°, which is less than the repeatability of the robots (green). **C. Motion quality:** As expected, multiple iterations result in a higher jerk trajectory, but this is within the defined acceptable limit of 0.03 rad/s^3.

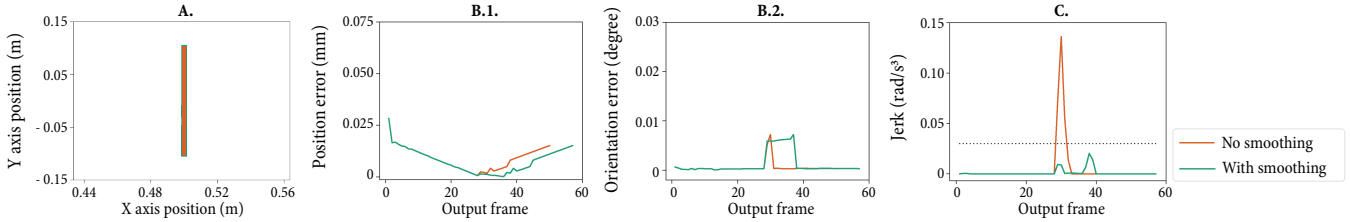

Fig. 2. **Line tracing:** Comparison between a trajectory generated that is not smooth (orange) and another that has smoothing by interpolation (green). **A. Top view:** Both solutions follow the path closely. **B. Errors:** Both trajectories have errors less than 0.1 mm and 0.05°, which is less than the repeatability of the robots. **C. Motion quality:** Smoothing reduces the overall jerk in the system to the defined acceptable limit of 0.03 rad/s^3 (indicated by the dotted black line). Note that the interpolation for smoothing results in a delayed output motion.

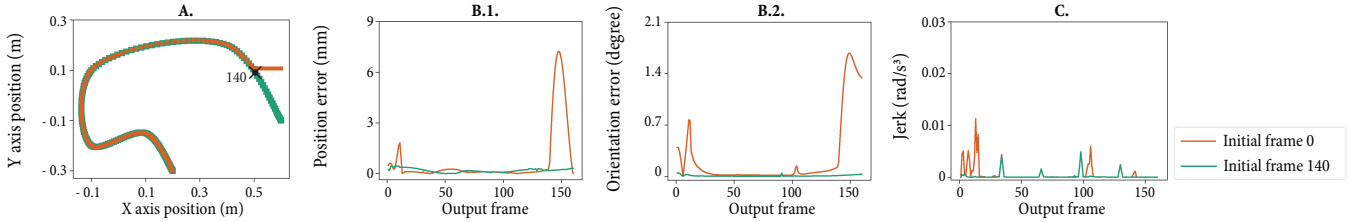

Fig. 3. **Spline tracing:** Comparison between trajectories generated from frame 0 (orange) vs 140 (green). **A. Top view:** The trajectory obtained by sequentially applying the solver from frame 0 yields a solution that follows the path closely until the last few frames (orange). Starting with frame 140 results in a better solution throughout (green). **B. Errors:** The solution obtained from starting from frame 140 and working backwards and forwards from that frame yields lower position and orientation error overall. **C. Motion quality:** Both solutions have jerk below the defined limit.

computing the trajectory from $\mathbf{x}_{140}$ resulted in an acceptable solution.

For the initial frame, our approach chooses the best solution (in terms of error) given the set of starting configurations. However, this greedy choice may be optimal for the initial frame and not the entire motion. If the resulting trajectory is not acceptable, our method will try using other solutions for the initial frame that have higher errors, but are still within the specified tolerance. To avoid redundant computation, we cluster the solutions for the initial frame using *DBSCAN* resulting in 2-15 options for each initial configuration.

C. User Guidance

Our approach attempts to find high accuracy solutions. However, if such solutions cannot be found, we provide for user control to determine where errors may be tolerable. Specifically, we allow for users to specify frames where accuracy is most critical (used as candidate initial frames above), and to specify the relative importance of different matching goals.

We allow a user to specify the relative importance of each axis of position and orientation matching. For example, in writing, the height of the pen tip and orientation of the pen must be precise for the path to succeed, while other position errors result in sloppier writing (see Figure 5). The specification of importance is restricted to two levels depending on whether matching the goals along a certain axis is emphasized or not. To implement these controls, we adapt *RelaxedIK* to have separate weights for each axis of its matching objective. If no acceptable solution is found for a frame using the default weights (where all axes are equally important), the solver is re-run with reduced weights for axes specified as less important, and the threshold for acceptability is raised for that axis. An example trajectory computed using dynamic weights is shown in Figure 4.

D. Algorithm Summary

To summarize the methods described in this section: our approach first applies *RelaxedIK* sequentially, starting from

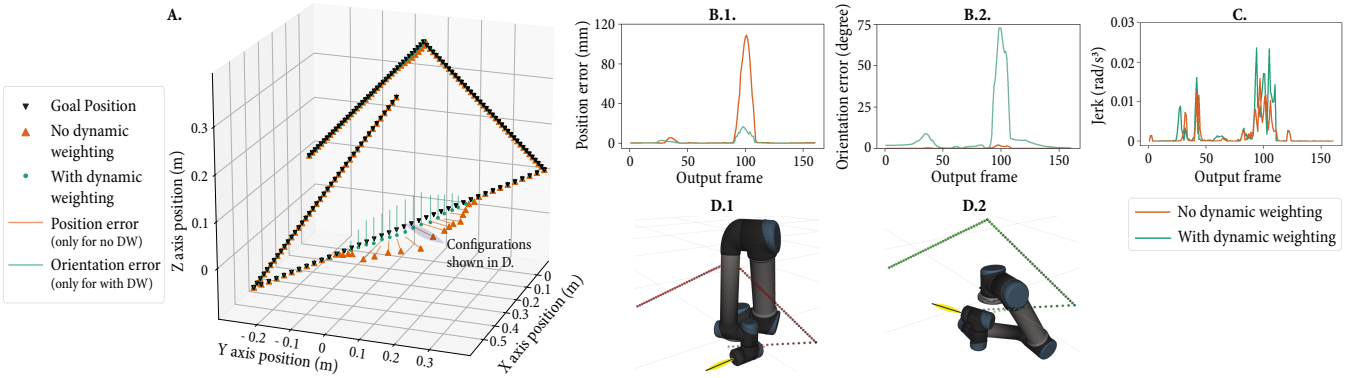

Fig. 4. **Polygonal chain tracing:** Comparison between trajectories generated with and without dynamic weighting functions. **A. 3D view:** Without dynamic weighting, the algorithm tries to match all degrees of freedom and provides a solution that does not match any well. In the improved solution, better position matching was obtained by dynamically relaxing orientation. Configurations corresponding to the shaded time step are shown in D. **B. Errors:** Position error in the initial solution was traded for orientation error in the improved solution with dynamic weighting that favored position matching over orientation matching. **C. Motion quality:** Both solutions have jerk below the defined limit and are acceptable solutions. **D. Configurations:** 1. Without any relaxation, the end-effector would collide with the base of the robot, thus resulting in a high position error. 2. The relaxation on orientation (orientation can be inferred from the direction of the yellow line) results in a configuration that leads to better position matching.

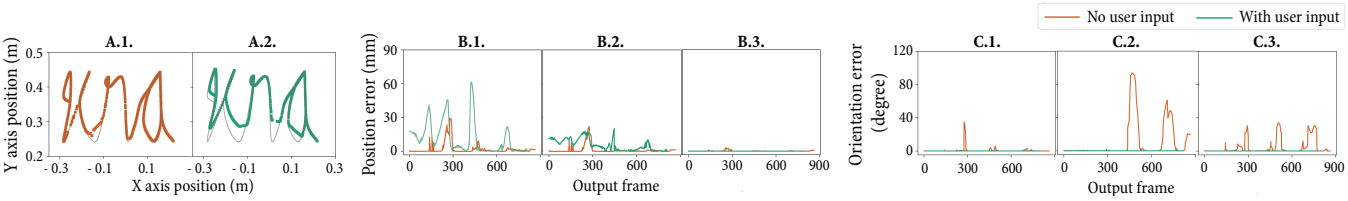

Fig. 5. **Writing task:** Comparison between trajectories generated with two different objectives. For this writing task on the $X-Y$ plane, the orientation is very important because the writing implement has to be held correctly. Additionally, there should be no position error along the z axis because it will either result in no writing when the writing implement is off the plane or damage it. Thus, the appropriate user input will be to emphasize on matching orientation and z position. **A. Top view:** The input path (thin gray line) is unachievable by the robot. In the default solution, better position matching was obtained by dynamically relaxing orientation. By incorporating user inputs, position matching is found to be worse, but still good enough for the text to be read as *icra*. **B. Errors:** The baseline trajectory shows undesirable errors along the degrees of freedom specified as important by the user. The performance is improved at the cost of some position accuracy, so that a trajectory that can actually be executed by the robot is obtained.

the beginning of the motion. The initial configuration is chosen by trying a sampling of starting configurations from a pre-created list. At each time step, the solver is re-run without smoothness regularization to see if this provides an improved solution. If the resulting trajectory is not acceptable, the sequential application process is repeated for different initial frames. For each of these alternate initial frames, multiple starting configurations are used, to create a number of alternative trajectories. In the process of creating these alternative trajectories, on frames where an accurate solution cannot be found, the solver is re-run in a manner that de-emphasizes matching axes deemed unimportant by the user. The trajectory that best meets the specified tolerances is returned.

VI. Evaluation

Prior work [6] showed that *RelaxedIK* performed favorably compared to state-of-the-art motion synthesis methods, Trac-IK [28], and the RRT Connect and PRM planners in the open-source library OMPL [29]. Thus, we use the performance of the native *RelaxedIK* solver as our baseline. We present our evaluations for the adapted solver alongside the baseline in Figure 6.

Our algorithm was implemented in Python and integrated with ROS. The testing was performed on an Intel Core i7-6700HQ 2.60 GHz CPU with 32 GB RAM in simulation for two robot platforms, the Universal Robots UR5 (6-DoF) and Rethink Robotics Sawyer (7-DoF). We generated different types of paths, including chains of Catmull-Rom spline segments, polygonal chains of line segments and standard shapes like line, rectangle and circle. Orientations were changed $1-3$ times for every path. These 6-DoF paths ranged from $150-1800$ frames, which correspond to $5-60$ s of an input sampled at 30 Hz. This resulted in a total of 15 paths with 10,000 frames. The paths were shifted to two different parts of the workspace, one where the robot was always able to reach all targets without danger of self-collisions (*Set P* in Figure 6), and another closer to the static base of the robot designed to induce relaxation of position or orientation goals to meet feasibility (*Set Q* in Figure 6). For the case where goals were relaxed, different user input combinations were tested, where (A) position was more important than orientation, and (B) orientation and z position were more important than x and y positions. In addition to these paths, we also evaluate the algorithm on 15 paths obtained from human demonstrations (*Demos* in Figure 6) of a simple pick-and-place task, collected at 120 Hz using

Paths	Mode	Max Position Error (mm)				Matched	Max Orientation Error (degree)				Matched	Max Joint	Time per
		X	Y	Z	Total	frames (%)	X	Y	Z	Total	frames (%)	Jerk (rad/s³)	frame (ms)
Set P	Baseline	3.53 ± 2.17	3.94 ± 2.31	3.81 ± 2.66	7.75 ± 2.47	0	2.11 ± 1.03	2.48 ± 1.31	2.12 ± 1.44	4.19 ± 1.38	0	0.014 ± 0.004	28 ± 8
	Our method	0.07 ± 0.03	0.06 ± 0.03	0.06 ± 0.04	0.13 ± 0.03	94	0.04 ± 0.02	0.04 ± 0.02	0.03 ± 0.02	0.07 ± 0.02	86	0.026 ± 0.003	242 ± 92
Set Q	Baseline	18.4 ± 21.5	19.8 ± 14.4	17.2 ± 15.1	38.9 ± 15.9	0	14.2 ± 12.1	13.7 ± 9.30	11.2 ± 4.50	25.0 ± 10.2	0	0.023 ± 0.005	34 ± 12
	Our method (A)	3.71 ± 3.12	4.84 ± 4.18	2.37 ± 2.10	7.81 ± 3.96	92	15.7 ± 20.9	21.0 ± 27.9	14.9 ± 16.1	41.5 ± 20.6	NA	0.028 ± 0.006	782 ± 451
	Our method (B)	29.7 ± 29.9	20.1 ± 27.6	2.07 ± 1.20	45.3 ± 23.8	91	0.82 ± 0.47	1.63 ± 0.56	1.42 ± 0.39	2.27 ± 0.43	80	0.029 ± 0.007	757 ± 389
Demos	Baseline	3.42 ± 2.35	4.06 ± 2.03	3.96 ± 2.57	7.29 ± 2.26	0	2.47 ± 1.63	2.90 ± 1.72	2.02 ± 1.62	5.33 ± 1.56	0	0.016 ± 0.006	28 ± 7
	Our method	0.07 ± 0.03	0.05 ± 0.03	0.06 ± 0.03	0.12 ± 0.03	97	0.04 ± 0.02	0.04 ± 0.03	0.03 ± 0.02	0.07 ± 0.03	78	0.027 ± 0.003	246 ± 63

Fig. 6. Summary of aggregated results of our method against the baseline *RelaxedIK* solver. Each table entry represents an aggregate over both robots for all paths in the set. *Set P* and *Set Q* have 15 paths that are generated in the workspace of the robot. *Set B* has more challenging paths that are close to the static base of the robot. *Demos* consist of 15 paths obtained from human demonstrations using motion capture technology. The mean and standard deviation (across all paths) of maximum (over each path) joint jerks, computation time per-frame and errors along every axis are reported. The percentage of the total frames that are within the error limits are also shown as *matched frames*.

a motion capture system that tracked reflective markers.

In our evaluation, the tolerance for each axis is set at 0.1 mm for position matching and 0.05° for orientation matching. These values correspond to the repeatability of the robots used in this evaluation. We use 0.03 $\mathrm{rad}/\mathrm{s}^3$ as an acceptable amount of joint jerk in our evaluation, assuming an input sampling rate of 30 Hz.

Our results are summarized in Figure 6. We report mean and standard deviation (across all paths) of maximum errors over each path. The overall performance of the method is reflected by the *matched frames* metric. Frames are considered "matched" if they are within the error tolerance. The baseline never achieves this level of accuracy, while our method achieves it on the majority of frames. On the easier test cases (*Set P* and *Demos*), the maximum errors are barely above the tolerances, suggesting that the missed frames are all "near-misses" (approximately 0.1 mm total error). For the more challenging examples (*Set Q*), some frames have higher errors (indicated by the higher max error rates), which we expect because the robot must avoid self-collisions which are problematic in these motions with goals close to its body. The results also show how the change of objective function leads to different errors: when the algorithm prioritizes positional accuracy (Objective A), these errors are much lower; emphasizing orientation and z-axis positional error (Objective B) lead to lower errors in these axes, at the expense of greater x and y-axis errors. The results also show the drawbacks to our approach relative to the baseline. First, the results exhibit higher maximum jerk, although this is still within the defined acceptable range. Second, while the baseline reliably produced paths at rate of approximately 30ms per frame (averaged over the whole motion), our method was much slower, sometimes taking over a second per frame in challenging cases.

VII. Conclusion

In this paper, we present an offline method for user-guided synthesis of feasible and smooth trajectories that closely follow a Cartesian workspace path. We evaluate its performance on many challenging paths for robots with different degrees of freedom and show that the output trajectories honor kinematic and path constraints within the limits prescribed by the user.

Limitations — As discussed in Section III, our approach provides weak feasibility guarantees on the interpolation between computed states. Our methods rely on keeping samples sufficiently close to each other and sufficiently far from self-collisions and singularities. Similarly, we also do not consider dynamics. However, our kinematically feasible trajectories could be used as an input to a time-scaling approach (e.g., [30], [31], [32]). At present, our notion of feasibility includes self-collisions, but not a more general view of collisions. While it is possible to include other static obstacles in our objectives, it is unclear how well distance functions for complex obstacles can be efficiently approximated for use in the objective functions.

Our approach provides no notion of global optimality. While the paths are "sufficiently good" there may be other paths that are better in some way (e.g., shorter, smoother, farther from obstacles). Therefore, our approach is inappropriate for situations where such optimality is required. Optimality guarantees are challenging given the non-linear nature of the problem and the feasibility requirements. Similarly, if our methods fail to find a trajectory that meet tolerance criteria, there is no guarantee that one does not exist. However, if our methods fail to find a solution within tolerance, they provide feasible solutions that are close.

Future Work — Our method is a sequential approach and relies on an IK solver [6] that does not include future frames of an input sequence in its objectives. There is potential for utilizing this information to reduce the number of passes required to yield a satisfactory output trajectory.

Our current implementation is slow; it can take as much as ten times the duration of the motion to provide an acceptable trajectory. Additional work is required to determine computationally demanding parts of the algorithm and ways to speed them up.

The inputs discussed in Section V are provided to our method programmatically. Adding a graphical interface is a natural extension of our implementation towards creating an effective robot motion authoring system.

To date, we have applied our approach to a variety of paths in simulation experiments and used it on the robots in our laboratory. The ability of our method to reliably create feasible and accurate trajectories given input paths, and to allow for user control over errors will make the methods valuable for a variety of uses.

REFERENCES

[1] S. Chernova and A. L. Thomaz, "Robot learning from human teachers," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 3, pp. 1–121, 2014.

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, pp. 1371–1394, Springer, 2008.

[3] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[4] M. N. Nicolescu and M. J. Mataric, "Experience-based representation construction: learning from human and robot teachers," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, pp. 740–745, IEEE, 2001.

[5] J.-C. Latombe, *Robot motion planning*, vol. 124, Springer Science & Business Media, 2012.

[6] D. Rakita, B. Mutlu, and M. Gleicher, "RelaxedIK: Real-time synthesis of accurate and feasible robot arm motion," in *Robotics: Science and Systems (RSS)*, 2018.

[7] K. Hauser and S. Emmons, "Global redundancy resolution via continuous pseudoinversion of the forward kinematic map," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 932–944, 2018.

[8] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5002–5008, IEEE, 2010.

[9] L. Liu, K. Yin, M. van de Panne, T. Shao, and W. Xu, "Sampling-based contact-rich motion control," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 128, 2010.

[10] L. Liu, K. Yin, and B. Guo, "Improving sampling-based motion control," in *Computer Graphics Forum*, vol. 34, no. 2, pp. 415–423, Wiley Online Library, 2015.

[11] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[12] J. Luo and K. Hauser, "An empirical study of optimal motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1761–1768, IEEE, 2014.

[13] K. Hauser and V. Ng-Thow-Hing, "Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2493–2498, IEEE, 2010.

[14] D. Rakita, B. Mutlu, and M. Gleicher, "Stampede: A discrete-optimization method for solving pathwise-inverse kinematics," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[15] R. Holladay, O. Salzman, and S. Srinivasa, "Minimizing task-space Frechet error via efficient incremental graph search," *IEEE Robotics and Automation Letters*, 2019.

[16] C. B. Phillips, J. Zhao, and N. I. Badler, "Interactive real-time articulated figure manipulation using multiple kinematic constraints," in *Proceedings of the 1990 Symposium on Interactive 3D Graphics*, ser. I3D 1990, pp. 245–250, ACM, 1990.

[17] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.

[18] J.-C. Nebel, "Keyframe interpolation with self-collision avoidance," in *Computer Animation and Simulation*, pp. 77–86, Springer, 1999.

[19] M. Gleicher, "Retargetting motion to new characters," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH 1998, pp. 33–42, ACM, 1998.

[20] J. Chai and J. K. Hodgins, "Constraint-based motion optimization using a statistical dynamic model," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, p. 8, 2007.

[21] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4569–4574, IEEE, 2011.

[22] J. Luo and K. Hauser, "Interactive generation of dynamically feasible robot trajectories from sketches using temporal mimicking," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3665–3670, IEEE, 2012.

[23] J. Denny, J. Colbert, H. Qin, and N. M. Amato, "On the theory of user-guided planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4794–4801, IEEE, 2016.

[24] F. Islam, O. Salzman, and M. Likhachev, "Online, interactive user guidance for high-dimensional, constrained motion planning," in *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4921–4928, International Joint Conferences on Artificial Intelligence Organization, 2018.

[25] O. B. Bayazit, G. Song, and N. M. Amato, "Enhancing randomized motion planners: Exploring with haptic hints," *Autonomous Robots*, vol. 10, no. 2, pp. 163–174, 2001.

[26] G. Palubeckis, A. Ostreika, and D. Rubliauskas, "Maximally diverse grouping: an iterated tabu search approach," *Journal of the Operational Research Society*, vol. 66, no. 4, pp. 579–592, 2015.

[27] E. B. Dam, M. Koch, and M. Lillholm, *Quaternions, interpolation and animation*, vol. 2, Citeseer, 1998.

[28] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 928–935, IEEE, 2015.

[29] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[30] J. Luo and K. Hauser, "Robust trajectory optimization under frictional contact with iterative learning," in *Robotics: Science and Systems (RSS)*, 2015.

[31] K. Hauser, "Fast dynamic optimization of robot paths under actuator limits and frictional contact," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2990–2996, IEEE, 2014.

[32] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1533–1540, 2014.