

# Interactive Generation of Dynamically Feasible Robot Trajectories from Sketches Using Temporal Mimicking

Jingru Luo Kris Hauser

School of Informatics and Computing, Indiana University Bloomington

{luojing, hauserk}@indiana.edu

**Abstract**—This paper presents a method for generating dynamically-feasible, natural-looking robot motion from freehand sketches. Using trajectory optimization, it handles sketches that are too fast, jerky, or pass out of reach by enforcing the constraints of the robot’s dynamic limitations while minimizing the relative temporal differences between the robot’s trajectory and the sketch. To make the optimization fast enough for interactive use, a variety of enhancements are employed including decoupling the geometric and temporal optimizations and methods to select good initial trajectories. The technique is also applicable to transferring human motions onto robots with non-human appearance and dynamics, and we use our method to demonstrate a simulated humanoid imitating a golf swing as well as an industrial robot performing the motion of writing a cursive “hello” word.

## I. INTRODUCTION

This paper considers the use of *sketches* — freehand gestures or other coarse input trajectories — to generate expressive robot motion. While sketch-based trajectory design has allowed artists and novices to animate virtual characters in intuitive, life-like ways [14], [10], robots differ from virtual characters in important respects. Specifically, virtual characters are permitted to violate physical constraints as long as the motion appears lifelike, but robots are subject to hard constraints such as joint limits, actuator limits, and collision avoidance. It is therefore a challenge for robots to mimic sketches that move too fast, possess high-frequency components, or pass out of reach.

This paper considers the problem of converting sketches into strictly feasible motion without sacrificing the temporal qualities that are essential for gestures and performances to convey meaning. We present a trajectory optimization system that blends together geometric and temporal objectives. The geometric objective encourages a designated point on the robot to follow the sketch closely in space, while the temporal objective attempts to mimic its timing. The system incorporates several technical contributions to make the nonlinear optimization fast, which is important to provide designers with feedback in a reasonable amount of time (e.g., less than a minute). We use a decoupled two-stage scheme that performs a geometric optimization first without considering dynamics, and then performs a temporal optimization using a time-scaling algorithm (Chapter 11 of [4]). The decoupled optimizations individually are smaller and have smoother optimization “landscapes” than a coupled optimization, leading to fast convergence. Numerical experi-

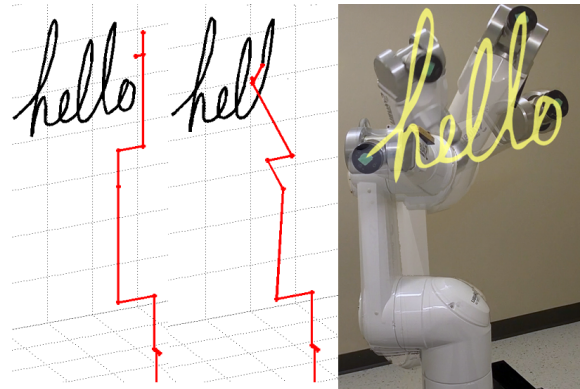


Fig. 1. A cursive “hello” word sketch mimicked by the Staubli TX90L industrial robot. Left: the desired “hello” word sketch. Middle: hand motion mimicking in simulation. Right: writing motion performed by the robot.

ments suggest this scheme is orders of magnitude faster than a simultaneous optimization of both geometric and temporal objectives, and produces paths of comparable quality. We also introduce an efficient method for generating good initial trajectories that help avoid local minima problems.

The system is demonstrated to faithfully transfer 2D freehand sketches that combine slow and fast motions at multiple spatial scales onto simulated articulated robots. All results are produced in less than a minute, whereas traditional trajectory optimization techniques can run for hours or days [1], [7]. Our technique is also appropriate for transferring human motions onto robots with non-human kinematic and actuation characteristics. We demonstrate its ability to transfer the human motion of a golf swing from the CMU Graphics Lab Motion Capture Database onto a simulated humanoid, and the writing motion of a cursive “hello” word onto a real Staubli TX90L industrial robot (Fig. 1).

## II. RELATED WORK

Sketch-based interfaces have been widely considered for animation of virtual characters [14], [10] and CAD prototyping [8], but they have not been widely considered for robot control. Similar problems of dynamic and kinematic infeasibility arise when adapting human motion to robots [16], [9], [15]. Yamane et. al. (2010) present a method for generating motions for non-human-like characters by learning a statistical mapping function which is followed

by an optimization process to improve its realism [16]. Pollard et. al. (2002) adapted the human motion data to the humanoid robot by limiting and scaling the human’s motion [9]. Ude et. al. used a B-spline wavelets to represent the joint trajectory for a humanoid robots and optimize the B-spline to transform optically tracked human motion data to the robot’s movements [15]. One difference between our work and the previous work is we formulate temporal matching as part of the optimization objective which leads to greater flexibility when dealing with infeasible input motions.

The temporal optimization stage of our system makes use of prior work from trajectory optimization along specified paths. Bobrow et. al. (1985) and Sin and McKay (1985) solved the minimal time control problem for robot along specified paths by computing switching curves in the velocity-displacement phase plane so that the bang-bang control can be applied to achieve minimal following time [2]. Various enhancements to the algorithmic efficiency and generality of these techniques have also been proposed [12], [6], [5]. Using the concept of switching curves, Bobrow presented an optimization technique to find an optimal path and velocities between endpoint states [1]. Shiller and Dubowsky also used path and velocity optimization as a second local optimization step to improve the quality of paths from a global grid search [11]. Other work has considered different objective functions, such as minimum jerk or minimum torque rate, rather than just execution time [3], [7].

In the current paper we are interested not only in execution time but also timing throughout the path. While minimizing time may be a priority for industrial manipulators, temporal mimicking is essential to attain human-like performances.

### III. PROBLEM FORMULATION

This work considers generating a trajectory  $q(t) : [0, t_f] \mapsto \mathbb{R}^n$  for an  $n$ -DOF robot that mimics an input *sketch*  $x(t) : [0, T_f] \mapsto \mathbb{R}^3$ . The sketch indicates the desired motion of a single designated point  $R$  on the robot, and is assumed to be given in the Cartesian work space of the robot. We envision a convenient paradigm for motion design in a CAD interface using a mouse or tablet input device in which the designer chooses  $R$  and the sketch using a single “click and drag” gesture. For the problem of human motion transfer, the association between points on the human and robot are typically known (e.g., hand motion maps to end effector motion), but some scaling and transformation is typically required to place the sketch in the robot-centric reference frame. In this paper such a transformation is assumed given. For simplicity we present our technique using a single point on the robot. Coordinating the motions of multiple points is a straightforward generalization.

We address the problem of finding a robot trajectory  $q(t)$  and ending time  $t_f$  that minimizes an objective functional  $f(q, t_f, x)$  while respecting the constraints: (a) joint limits  $q_{min} \leq q \leq q_{max}$ ; and velocity, acceleration, and torque bounds: (b)  $\dot{q}_{min} \leq \dot{q} \leq \dot{q}_{max}$ ; (c)  $\ddot{q}_{min} \leq \ddot{q} \leq \ddot{q}_{max}$ ; (d)  $\tau_{min}(q, \dot{q}) \leq \tau \leq \tau_{max}(q, \dot{q})$ .

Torques are related to accelerations via the standard dynamic equations:

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = \tau \quad (1)$$

where  $M$  is the mass matrix,  $C$  is the coriolis forces, and  $G$  is the generalized gravity vector.

We define the objective functional  $f(q, t_f, x)$  as a weighted sum of geometric deviation, temporal deviation, and total time objectives:

$$f(q, t_f, x) = f_g(q, x) + \beta f_t(q, x) + \gamma t_f. \quad (2)$$

Later we will show how  $\beta$  and  $\gamma$  can be chosen by the designer to reflect the relative importance of geometric fidelity, temporal fidelity, and efficiency.

**Geometric Objective.** Define  $t_x(s) : [0, 1] \mapsto [0, T_f]$  as the arc-length parameterization of  $x$  that maps the unit interval to the domain of the sketch. In other words,  $x(t_x(s))$  travels at uniform speed. Also define a monotonic function  $t_q(s) : [0, 1] \mapsto [0, t_f]$  as the *time scaling* of the robot’s trajectory. The domain of the parameter  $s$  is used to define correspondences between  $x$  and  $q$ . Let  $FK(q)$  denote the Cartesian position of point  $R$  at configuration  $q$  as determined through forward kinematics. We define the geometric objective as

$$f_g(q, x) = \int_0^1 \|FK(q(t_q(s))) - x(t_x(s))\|^2 ds \quad (3)$$

**Temporal Objective.** Temporal mimicking can be defined in several ways. *Absolute mimicking* can be defined by the sum-of-squares objective  $f_t(q, x) = \int_0^1 (t_x(s) - t_q(s))^2 ds$ . The problem with the absolute formulation is that if the robot falls behind the pace of the sketch due to actuation limits, it will later speed up in order to recover the timing. This is often an unnatural response. Instead we use the *relative mimicking* objective:

$$f_t(q, x) = \int_0^1 (t_q(s) - t_x(s) \frac{t_f}{T_f})^2 ds \quad (4)$$

which permits the robot to slow down or speed up the motion overall in order to mimic the relative timing of events. In other words, this objective allows for a global change in tempo.

To approximate the integrals in (3) and (4) as sums and enforce the constraints (a–d) pointwise we use the standard practice of choosing collocation points  $\{s_i | i = 0, 1, \dots, m\}$  uniformly in the domain of the parameter  $s \in [0, 1]$ .

### IV. DECOUPLED OPTIMIZATION

In order to solve this nonlinear optimization efficiently we introduce an approximate decoupled approach inspired by the techniques of Bobrow (1998) [1] but specifically tailored to the sketch following problem. The system progresses in two stages. First, it optimizes a geometric path  $p(s)$  by modifying the control points of cubic B-spline in order to match the sketch. At this point the path is subject to kinematic constraints only. In the second stage, it keeps the path fixed and optimizes a time scaling  $s(t)$  to match the

timing of the sketch and reduce total time, subject to the dynamic constraints. The robot's trajectory is then defined as  $q(t) = p(s(t))$ .

The decoupling runs the risk of producing a bad path for the temporal optimization stage, but we take steps to mitigate this risk. Novel contributions include an informed method for generating good initial paths and a formulation of the time scaling optimization with favorable numerical properties. We also introduce an additional curvature objective term that regularizes the smoothness of the path and improves path execution time.

### A. Geometric Path Optimization

To model the path we use a non-uniform cubic B-spline. B-splines are advantageous in that they are twice differentiable, so acceleration and torque are well-defined, and they support local control, which makes shape easy to adjust [13]. The spline is defined by the knot vector  $K = \{k_1, k_2, \dots, k_{n_K}\}$  and  $n_{cp} = n_K - 4$  control points  $p_1, \dots, p_{n_{cp}}$ . To enforce that the start and ending control points specify the exact start and end of the path, we used a clamped B-spline whose knot vector repeats the starting knot and ending knot 3 times. The resulting path is then  $p(s) = \sum_i p_i \beta_i(s, K)$  where  $\beta_i$  are the standard B-spline basis functions. Let  $P$  denote an  $n \times n_{cp}$  matrix that contains the control points in its columns. When necessary, we will denote the explicit dependence of  $p(s)$  on  $P$  using the notation  $p(s; P)$ . Keeping the knots constant we will optimize the control points  $P$  so that  $p(s; P)$  minimizes the objective (3). Here  $p(s; P)$  is constrained to the joint limits (a). We also introduce a smoothing term to the objective function, weighted by the parameter  $\alpha$ , to smooth the motion.

Specifically, the optimization problem is:

$$\min_P f_g(p(\cdot, P), x) + \alpha f_c(p(\cdot, P)) \text{ such that} \quad (5)$$

$$q_{min} \leq p(s; P) \leq q_{max} \text{ for all } s \in [0, 1]$$

The second term  $f_c$  in the objective is a smoothness term that penalizes high curvature motions that cannot be tracked quickly.  $f_c$  integrates the curvature for  $s \in [0, 1]$ . The curvature is computed as the 2nd derivative of the B-spline which is a linear function of  $P$  and can be computed analytically using the B-spline recursion formulas.  $\alpha$  is a weight for this term. Experiments suggest major improvements in path quality are achieved at very small values of  $\alpha$  without introducing significant deviations from the sketch. The joint limit constraint in (5) can be implemented efficiently by box constraints  $q_{min} \leq p_i \leq q_{max}$  for  $i = 1, \dots, n_{cp}$  because B-splines have a convexity property that states that the path lies entirely in the convex hull of the control points.

To solve (5) we use the SQP implementation in the Matlab optimization toolbox and provide the analytical gradients for the objective functions.

Using a 4-link robot (Fig. 2, left) in a 2D simulation we compare how the smoothing weight  $\alpha$  affects the shape of the resulting path and optimal execution time (as determined by the method in the following section). As  $\alpha$  increases, the average deviation of the planned path increases but execution

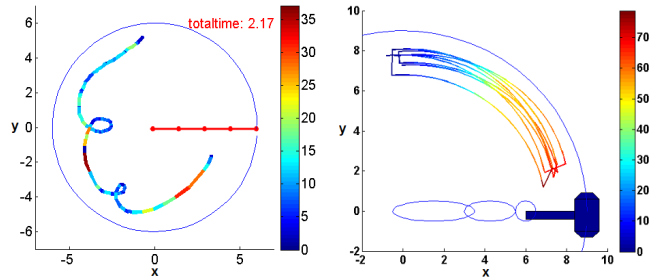


Fig. 2. Left: A 4-link robot follows the sketch whose color indicates the velocity. Right: Hammer example for testing the effects of the timing weight parameter  $\beta$ . The desired sketch descends quickly and raises up slowly. The path color indicates the velocity of the desired motion.

time decreases (see Fig. 3) and Table I. Some smoothing in fact is appropriate for eliminating small high-frequency jerks from noisy sketches.

### B. Curvature Based Path Initialization

To start the optimization process, we need to provide a good starting value for knots  $K$  and control points  $P$ . One possibility for choosing  $K$  is simply a uniform discretization in the range  $[0, 1]$ , but this may place too many knots in straight regions of the sketch while placing too few in corners and sharp curves. Rather than simply adding more knots and control points, which slows down optimization, we present a technique that uses the curvature of the sketch to allocate knots and control points more effectively.

We define a goodness metric  $g(s)$  that we use to distribute knot points more heavily in areas where the curvature is large.  $g(s)$  contains a geometric curvature term that measures the second derivative of  $x(s)$  as well as an arc length term:

$$g(s) = (1 - \epsilon) \frac{x(s+h) + x(s-h) - 2x(s)}{h^2} + \epsilon \quad (6)$$

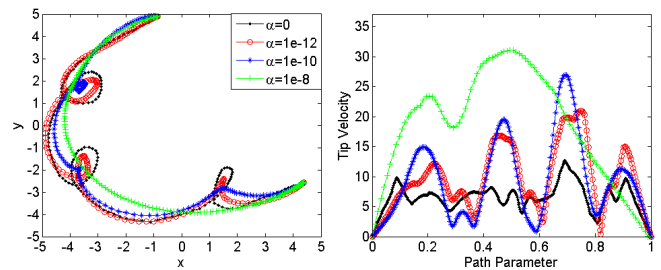


Fig. 3. Effects of smoothing weight  $\alpha$  on the shape and speed of the path. Left: the planned path deviates further from the desired one as  $\alpha$  increases. Right: the tip velocity increases as  $\alpha$  increases.

$\alpha$	Geometric Deviation	Exec. Time
0	0.042	4.69
1e-12	0.226	2.69
1e-10	0.459	2.11
1e-8	0.960	0.92

TABLE I  
EFFECTS OF THE SMOOTHING WEIGHT  $\alpha$ .

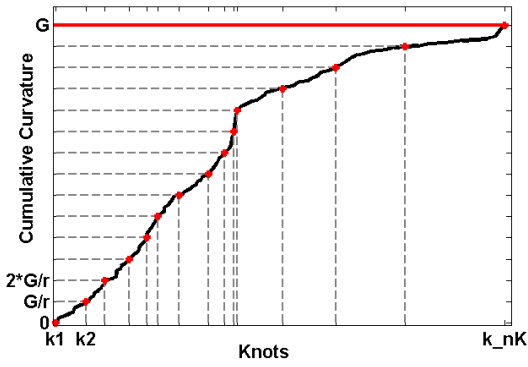


Fig. 4. Selection of knots. The y-axis is the cumulated curvature computed as the sum of eq. (6). The maximum cumulated curvature  $G$  is divided into  $r$  equal intervals and the set of parameter  $\{s\}$  whose corresponding curvature is  $\frac{G}{r}, \frac{2G}{r}, \dots$  are selected as the knots  $\{k_1, k_2, \dots, k_{nK}\}$ .

The value  $h$  is a step size that is set to approximately twice that of the typical noise contained in the sketch. Here  $\epsilon$  is a weight determining the relative importance of distance-based allocation compared to curvature-based allocation.  $\epsilon = 0$  indicates that the allocation should be entirely curvature-based, while  $\epsilon = 1$  indicates that the allocation should be uniform. Knots are then allocated non-uniformly in the  $s$  domain to achieve a uniform division of the cumulative sum of  $g$  along the path (see Fig. 4).

Once the knots are determined, the control points  $p_1, \dots, p_{n_{cp}}$  are computed to match the sketch relatively closely. The following scheme works well in practice to reduce the risk of running into local minima. We make use of a numerical Newton-Raphson inverse kinematics solver  $IK(q, x)$ , which takes as input a seed configuration  $q$  and a goal position  $x$  and produces a configuration at which the desired point  $R$  is closer to  $x$  (attaining it, if possible) and for which joint limits are respected. First, we set  $p_1 \leftarrow IK(q^0, x(0))$  seeded from an arbitrary initial configuration  $q^0$ . Then to choose  $p_2$ , we find the value of  $s$  that maximizes the basis function  $\beta_2(s)$ . Let it be  $s_2^{max}$ . We then set  $p_2 \leftarrow IK(p_1, x(s_2^{max}))$ . We continue in this manner, setting  $p_i \leftarrow IK(p_{i-1}, x(s_i^{max}))$  until all control points are found. If it is found that these solutions from IK interact poorly with joint limits, it is useful to re-seed this process several times and keep the initial path that is closest to the sketch.

### C. Time Scaling

Once a path  $q(s) : [0, 1] \rightarrow \mathbb{R}^n$  is fixed the temporal optimization stage finds a time scaling  $s(t) : [0, t_f] \rightarrow [0, 1]$  that respects the dynamic constraints (b–d) and matches the timing of the desired trajectory and achieve minimum motion time. We also enforce the constraints that the robot starts and ends at rest. We formulate a path velocity parameterization that produces a relatively small, numerically stable optimization problem.

Following [2] we can rewrite constraints (b–d) and (1) in terms of  $s, \dot{s}$  and  $\ddot{s}$  through the relations  $\dot{q}(s, \dot{s}) = \frac{dq}{ds} \dot{s}$  and  $\ddot{q}(s, \dot{s}) = \frac{d^2q}{ds^2} \dot{s}^2 + \frac{dq}{ds} \ddot{s}$ . Then let  $a(s, \dot{s}) = M(q(s)) \frac{dq}{ds}$ ,  $b(s, \dot{s}) = M(q(s)) \frac{d^2q}{ds^2} + C(q(s), \frac{dq}{ds} \dot{s})$  and  $c(s, \dot{s}) = G(q(s))$ ,

we can get:

$$\tau = a(s)\ddot{s} + b(s)\dot{s}^2 + c(s) \quad (7)$$

In this way, the constraints are formulated as a function of the time scaling parameter  $s$  rather than the  $n$  joint variables  $q$ . Now let us define the velocity of the path as a piecewise linear interpolation between velocity variables  $V = \{\dot{s}_0, \dot{s}_1, \dots, \dot{s}_k\}$ . Thus  $s(t)$  is a piecewise quadratic curve consisting of  $k$  segments  $s_i(t)$ ,  $i = 1, \dots, k$  with constant acceleration in each segments.

By constraining the domain of each curve segment to uniform intervals  $\Delta s$ , the expended time  $\Delta t_i = t_i - t_{i-1}$  can be derived as  $\Delta t_i = \frac{2\Delta s}{\dot{s}_i + \dot{s}_{i-1}}$ .

The absolute time of each segment is given by  $t_i(V) = \sum_{j=1}^i \Delta t_j = \sum_{j=1}^i \frac{2\Delta s}{\dot{s}_j + \dot{s}_{j-1}}$ . Using this expression we can compute the total time  $t_f(V) = t_k(V)$  as well as the temporal objectives  $f_t(V)$  given by (4). Specifically, we evaluate  $t_q(s)$  by 1) finding the segment  $s_i$  such that  $s \in [(i-1)\Delta s, i\Delta s)$ , 2) solving the quadratic equation for  $t$  such that  $s = s_i(t) = \Delta s + \dot{s}_{i-1}(t - t_{i-1}) + \frac{1}{2}\ddot{s}_i(t - t_{i-1})^2$ .

With these definitions the optimization is specified over the independent variable  $V$  as follows:

$$\begin{aligned} \min_V \quad & \beta f_t(V) + \gamma t_f(V), \quad \text{s.t.} \\ & \dot{s}_0 = 0 \\ & \dot{s}_k = 0 \\ & \dot{q}_{min} \leq \dot{q}(s, \dot{s}) \leq \dot{q}_{max} \\ & \ddot{q}_{min} \leq \ddot{q}(s, \dot{s}) \leq \ddot{q}_{max} \\ & \tau_{min} \leq a(s)\ddot{s} + b(s)\dot{s}^2 + c(s) \leq \tau_{max} \end{aligned} \quad (8)$$

A collocation method provides an efficient way of evaluating the above constraints. The  $s$  domain  $[0, 1]$  is discretized uniformly at  $s_1, \dots, s_m$ , and the velocity independent coefficients  $M(q(s))$ ,  $C(q(s), \frac{dq}{ds})$ , and  $G(q(s))$  are precomputed at each  $s_i$  to make evaluation faster. Hence each constraint is no more than quadratic in the elements of  $V$ .

Again we use a SQP method to optimize (8) using analytically computed gradients. As a starting point for the optimization we pick each  $\dot{s}_i$  as the reciprocal of the velocity of the desired motion  $1/t'_x(i\Delta s)$ . This leads to fast convergence if the robot can closely match the timing of the desired motion.

## V. COMPARING DECOUPLED AND COUPLED OPTIMIZATIONS

As mentioned earlier, the main risk faced by decoupled optimization is that the geometrically optimized path may not admit a high-quality time scaling. This section evaluates our approach against optimizations in which the geometric and temporal variables are optimized simultaneously. Our experiments show that the decoupled optimization is far faster but does not sacrifice much in terms of path quality.

The *coupled* approach optimizes simultaneously over the control points  $P$  and time scaling velocities  $V$  with the objective:

$$\min_{P, V} f_g(P) + \beta f_t(P, V) + \gamma t_f(V) \quad (9)$$

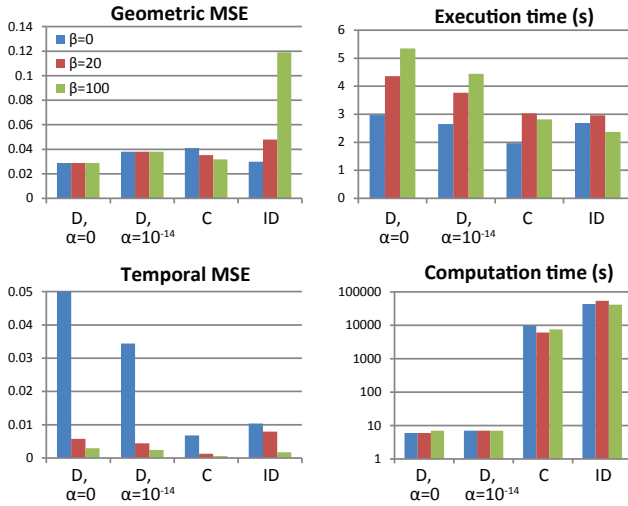


Fig. 5. Comparing the decoupled (D), coupled (C), and iterative decoupled (ID) approaches in terms of geometric objective, temporal objective, path execution time, and computation time for three different values of  $\beta$ . Computation time is plotted on a log scale.

while maintaining constraints (a – d) and (1);

We also considered the *iterative decoupled* approach of [1] in which the time scaling optimization is placed into the objective function as an inner loop:

$$\min_P (f_g(P) + \min_V [\beta f_t(P, V) + \gamma t_f(V)]) \quad (10)$$

We compared the three approaches under the same problem settings for the example motion showed in the left plot of Fig. 2. We used a 4-link robot with a reach of 6. Keeping the parameter  $\gamma$  as a constant 0.01, we tested different values of the timing weight  $\beta$ , and for the decoupled approach we varied the smoothing weight  $\alpha$  as well. Initial trajectories were the same across all three methods and 16 control points were used to control the shape of each trajectory. Fig. 5 plots their results in terms of each objective function component and computation time. As a reference point, the method of Pollard et al (2002) [9] which enforces exact time matching but allows deviating from the path to satisfy velocity limits obtains a geometric MSE on this example of 0.3, which is approximately 10 times worse than the decoupled optimization.

The coupled and iterative decoupled approaches are able to obtain better quality paths than the decoupled, but are orders of magnitude slower. Decoupled optimization takes seconds to converge completely, while the coupled optimizations take hours to obtain similar quality results Fig. 5. One fruitful approach might be to use the decoupled optimization to generate initial points for further coupled optimizations, which could speed their convergence to an optimal path.

## VI. RESULTS

Our simulation experiments test how well our method is able to track dynamically infeasible freehand sketches, and also test the sensitivity of our method to optimization parameters. Each generated trajectory is feasible with respect

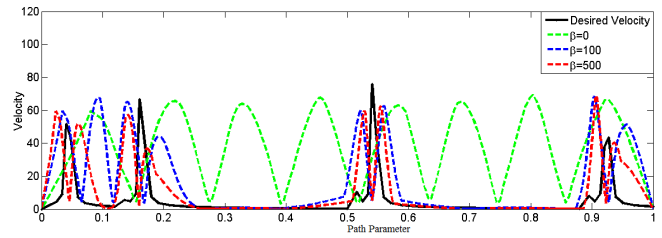


Fig. 6. Temporal matching results for the hammer example. The tip’s velocity is plotted against the normalized path parameter. The black curve is the desired velocity profile, and the blue, green, red ones show the optimized velocity profiles for different  $\beta$ . As  $\beta$  increases the matching improves.

to gravity, inertia, and velocity/acceleration/torque limits. We also demonstrate its use in transferring human motion capture data onto robots. We use data from the CMU Motion Capture Database to guide a 3D humanoid robot model to perform a golf swing and guide a real industrial robot (Staubli TX90L) to imitate a human motion of writing a cursive “hello” word. (See the supplemental video or visit the website at <http://www.indiana.edu/~motion/sketchmimicking/>.)

### A. Effects of Timing Weight Parameter

Our system provides the timing weight  $\beta$  as a tuning parameter to control how important motion timing is considered relative to path following. To illustrate its effects we consider the example of a repeated hammer strike (Fig. 2, right). The sketch has the endpoint moving faster while it falls, then the velocity instantaneously switches due to “impact”, and then it rises more slowly. This velocity profile is infeasible to follow exactly because the hammer switches instantaneously from upward to downward velocity upon striking the object, but there is no actual object in the simulator model and instead this rebound must be simulated by the robot’s actuators. Fig. 6 shows that when  $\beta$  is 0, the robot completes the motion as fast as possible but without regard to the speed of each phase. The result appears more like a waving motion than hammering. As we increase the timing weight, the sketch velocities are matched more closely over time and gives a better appearance of an impact occurring.

### B. Golf Swing on a Simulated Humanoid

We applied our method to replicate human motion using a simulated 29 DOF model of a humanoid upper torso (Fig. 7). Joint limit, velocity, acceleration, and torque constraints were set to relatively large human-like values.

Taking a golf swing from the CMU Graphics Lab Motion Capture Database, we used the motion of four points on the human – hands, head, and pelvis – as sketches for the corresponding points on the humanoid. The objective functions for each point were combined into a single objective function for an overall optimization. We do not constrain the relative orientation of the hands (as though a club were being held), but such a constraint could be easily formulated in the geometric optimization phase of our method.

The plot in Fig. 7 and the supplemental video demonstrate the necessity of temporal mimicking. With the timing weight

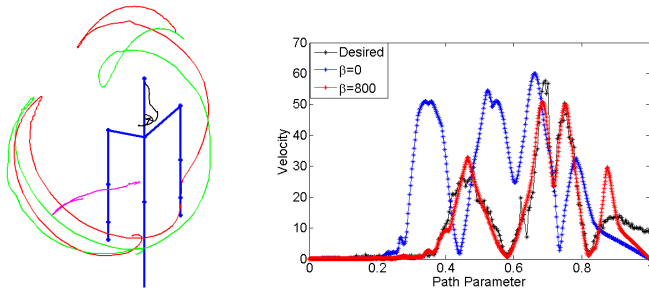


Fig. 7. Left: Humanoid torso following a human golf swing motion. The black, red, green and magenta curves are the desired paths for the head, left and right hands and the pelvis. Right: Left hand’s velocity profile as a function of the normalized path parameter, for two values of  $\beta$ . With higher  $\beta$  the timing of the swing is more natural.

$\beta$  set to zero, the motion can be executed faster, completing in 4.5s. But, backward pivot is executed just as quickly as the rest of the swing, which does not appear natural. Increasing the weight to  $\beta = 800$  generates a solution which slows down the motion to 7.8s but matches the appearance of a human golf swing much better. The velocity profile of the left hand is plotted in Fig. 7, demonstrating significant improvement with temporal mimicking. (Similar plots are obtained for the other mimicked points).

### C. Word Writing on Staubli TX90L Industrial Robot

We also applied a writing motion to the 6DOF Staubli TX90L industrial robot in our lab. To emphasize the end-effector trajectory, we tried the sketch of a cursive “hello” word. The sketch was generated by mouse stroke with a total length of 3.68 m and total time of 12.13 s. The optimized motion has the geometric MSE of 0.02, relative temporal MSE of 0.13 and execution time of 18 s. Fig. 1 and the results shown in the supplemental video indicate that trajectory executed by the robot’s tip is qualitatively similar to that of the input sketch.

## VII. CONCLUSION

We presented a system for enabling a robot to simultaneously match the geometry and timing of input “sketches” while respecting its physical limitations. The method is based on a trajectory optimization that decouples the geometric and temporal optimization phases. The technique is able to produce natural-looking results for complex sketches within seconds, and is demonstrated to successfully transfer human motions onto non-human robots both in simulation and on a real industrial robot. Such techniques will be useful for programming expressive entertainment robots, as well as for social robots and robot coworkers that use gesture to communicate with human teammates.

We intend to investigate motion design and/or human-robot interaction using our system to control a robot using

real-time human input. Further speedups may be needed, because fluid human-robot interaction may require responses in fractions of a second. One approach might use our system to populate a database of dynamically-feasible gestures and to stitch them together on-line. Another issue is that the relative importance of achieving good timing, pose, and low physical effort depends on whether the user is indicating a desired gesture, posture, or an action upon the environment. Future work might consider the *information content* of the sketch to help the robot select an appropriate objective.

## REFERENCES

- [1] J. E. Bobrow. Optimal robot path planning using the minimum-time criterion. *IEEE J. of Robotics and Automation*, 4:443–450, 1988.
- [2] J. E. Bobrow, S. Dubowsky, and J. Gibson. Time-optimal control of robotic manipulators along specified paths. *Int. J. of Robotics Research*, 4(3):3–17, 1985.
- [3] J. E. Bobrow, B. Martin, G. Sohl, E. C. Wang, F. C. Park, and J. Kim. Optimal robot motions for physical criteria. *J. of Robotic Systems*, 18(12):785–795, 2001.
- [4] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT press, 2005.
- [5] D. Constantinescu and E. A. Croft. Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *J. of Robotic Systems*, 17:223–249, 2000.
- [6] O. Dahl. Path constrained robot control with limited torques – experimental evaluation. In *IEEE Int. Conf. on Robotics and Automation*, pages 493–498 vol.2, May 1993.
- [7] K. Harada, K. Hauser, T. Bretl, and J.-C. Latombe. Natural motion generation for humanoid robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.
- [8] S. Murugappan and K. Ramani. Feasy: A sketch-based interface integrating structural analysis in early design. In *ASME Int. Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, August 2009.
- [9] N. Pollard, J. Hodgins, M. Riley, and C. Atkeson. Adapting human motion for the control of a humanoid robot. In *IEEE Int. Conf. on Robotics and Automation*, pages 1390–1397, 2002.
- [10] J. Popović, S. M. Seitz, and M. Erdmann. Motion sketching for control of rigid-body simulations. *ACM Trans. Graphics*, 22:1034–1054, October 2003.
- [11] Z. Shiller and S. Dubowsky. On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Trans. on Robotics and Automation*, 7(6):785–797, Dec 1991.
- [12] J.-J. Slotine and H. Yang. Improving the efficiency of time-optimal path-following algorithms. *IEEE Trans. on Robotics and Automation*, 5(1):118–124, Feb 1989.
- [13] S. E. Thompson and R. V. Patel. Formulation of joint trajectories for industrial robots using b splines. *IEEE Trans. on Industrial Electronics*, IE-34(2):192–199, may 1987.
- [14] M. Thorne, D. Burke, and M. van de Panne. Motion doodles: an interface for sketching character motion. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH ’07, New York, NY, USA, 2007. ACM.
- [15] A. Ude, C. G. Atkeson, and M. Riley. Planning of joint trajectories for humanoid robots using b-spline wavelets. In *IEEE Int. Conf. on Robotics and Automation*, pages 2223–2228. Press, 2000.
- [16] K. Yamane, Y. Ariki, and J. Hodgins. Animating non-humanoid characters with human motion data. In *Proc. 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA ’10, pages 169–178, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.