Space-Time Functional Gradient Optimization for Motion Planning

Arunkumar Byravan[1], Byron Boots[1], Siddhartha S. Srinivasa[2] and Dieter Fox[1]

***Abstract*— Functional gradient algorithms (e.g. CHOMP) have recently shown great promise for producing locally optimal motion for complex many degree-of-freedom robots. A key limitation of such algorithms is the difficulty in incorporating constraints and cost functions that explicitly depend on time. We present T-CHOMP, a functional gradient algorithm that overcomes this limitation by directly optimizing in space-time. We outline a framework for joint space-time optimization, derive an efficient trajectory-wide update for maintaining time monotonicity, and demonstrate the significance of T-CHOMP over CHOMP in several scenarios. By manipulating time, T-CHOMP produces lower-cost trajectories leading to behavior that is meaningfully different from CHOMP.**

I. Introduction

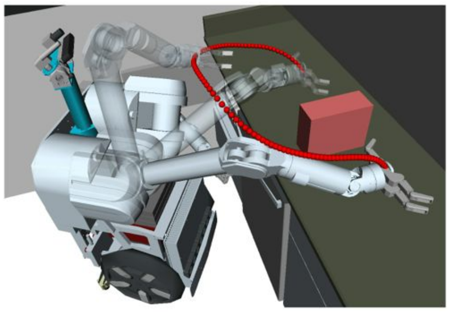
Motion planning algorithms plan a collision-free path for a given task, often subject to physical constraints. Recently, trajectory optimization algorithms [1]–[8] have demonstrated great success in a number of high-dimensional real-world planning problems. One such state-of-the-art algorithm, CHOMP [3], uses functional gradient optimization to bend an initial (often infeasible) trajectory away from obstacles while maintaining smoothness. A typical CHOMP trajectory is illustrated in Fig.1a, where the robot HERB's right arm pulls back away from the countertop to avoid the red box on its way to the goal.

Although the *theory* of CHOMP allows for arbitrary time parametrization, in *practice* CHOMP makes a deliberate choice. It parametrizes a trajectory as a set of waypoints spaced *equally* in time (detailed in Sec. III). This has a key consequence:

> *The only way to alter the timing of a trajectory is by altering the waypoints, i.e., the configurations of the robot.*

The distinction is subtle but critical. We draw an analogy: Imagine the trajectory as a set of photographs mounted on a zoetrope that spun at constant speed to produce the illusion of a movie. With CHOMP, if we wanted to speed up the middle section of our movie, we would have to take a whole new set of photographs.

Although any arbitrary trajectory timing can be generated, the process is tedious. As a result, algorithms like CHOMP are often followed by a *retiming* step, where the generated path is optimized for timing. Although motion retiming has been demonstrated to be quite successful [9]–[11], the decoupling of path planning and timing has several known disadvantages, including (1) it disallows the use of cost functionals that explicitly reason about time, like going slower near obstacles, a running example in this paper, and (2) is more prone to local minima.


(a) CHOMP trajectory

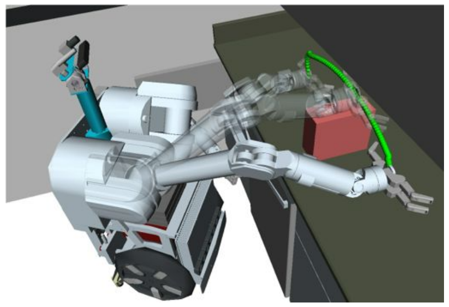

(b) T-CHOMP trajectory

Fig. 1: Space-time optimization gives T-CHOMP (green) additional freedom to manipulate time resulting in a slower, direct path above the box compared to CHOMP (red) which cannot reason about time.

Instead, we propose to add to CHOMP one additional degree of freedom: *time*. This enables us to optimize the trajectory in space-time (Sec. IV). Returning to our analogy, speeding up the middle section now merely requires us to speed up our zoetrope, *without* having to take new photographs.

As a consequence, we overcome both the above disadvantages: (1) our cost functionals *can* explicitly reason about time, and (2) we are able to escape more local minima. One such scenario is illustrated in Fig.1b. By adding a cost functional that trades off speed for obstacle avoidance, we are able to generate a trajectory that carefully and slowly goes over the box on its way to the goal, in contrast to Fig.1a where the inability to reason about time forces the trajectory to take a large detour away from the box.

Optimization in space-time poses several challenges. Key among them is *time monotonicity*, i.e., preventing time from moving backwards. To address this, we add trajectory-wide space-time constraints to enforce monotonicity. We derive the Karush-Kuhn-Tucker (KKT) first-order necessary conditions for the constraints and present an efficient algorithm for implementing them for a waypoint parametrization (Sec. V).

[1]Department of Computer Science & Engineering, University of Washington `(barun, bboots, fox)@cs.washington.edu`
[2]Robotics Institute, Carnegie Mellon University `siddh@cs.cmu.edu`.

This work was funded in part by the National Science Foundation under contract NSR-NRI 1227234.

We demonstrate our new algorithm Time CHOMP (T-CHOMP) on two scenarios: a 2D point robot (for visual clarity) and a 7DOF robot arm attached to Marvin, a segway platform and HERB, a bimanual mobile manipulator, in simulated environments.

We use a specific time-based cost functional as discussed earlier (and elaborated in Sec. VI), which slows down when it is close to an obstacle, and contrast our results with CHOMP. Our results show that reasoning directly in space-time gives us better control over path and timing, resulting in meaningfully different behavior from CHOMP. (Sec. VII).

Our paper makes the following contributions: (1) we propose a new algorithm T-CHOMP for space-time functional gradient optimization, (2) derive an efficient algorithm for the trajectory-wide space-time time monotonicity constraint, and (3) demonstrate results on a simulated robotic platform.

We also discuss several limitations of our work (Sec. VIII). Key among them is the structure of the cost functional. Adding time adds extra dimensions for optimization, but also imposes a greater burden on defining the correct cost functional. An improperly weighted cost functional can result in motion that very slowly goes through an obstacle, for no fault of the optimizer. Learning good space-time cost functionals is a key area of future work.

II. Related Work

There has been significant work on trajectory optimization for motion planning [12]. Khatib [13] and Rimon and Koditschek [14] showed the use of artificial potential functions in navigation planning. Quinlan and Khatib [1] introduced the theory of elastic bands for planning collision free paths in space, bending the band away from collision while maintaining shape. Ratliff et al. [2], [3] extended these techniques to use functional gradient optimization to plan collision free paths, leading to the algorithm CHOMP.

Kalakrishnan et al. [4] adapted the optimization to use stochastic gradient descent with the path integral framework [15]. Schulman et al. [6] used convex relaxations and sequential quadratic programming to compute paths that are out of collision. Park et al. [5] suggested an incremental replanning algorithm to generate low cost paths in dynamic environments. These methods neglect to optimize the timing of the trajectory, and are usually followed by a retiming step.

Motion retiming for pre-planned robot trajectories with constraints is a well developed field. Bobrow et al. [9] proposed a method for computing minimum-time optimal controls for a pre-planned path subject to torque limitations. Shiller et al. [10] and Canny et al. [16] extended this to include obstacles and other non-linear constraints. Hollerbach [11] proposed to dynamically rescale the velocity profile to make trajectories feasible for execution on robots. All these methods handle path planning and timing in isolation.

Optimal control techniques [17] have looked at the problem of trajectory optimization with velocity as a part of the state. Differential Dynamic Programming [18] methods used a locally quadratic approximation of the dynamics and cost to solve the optimal control problem. Similar to DDP, iLQG [7] used quadratic cost approximations and handled non-linear control costs. Toussaint et al. [19] used approximate inference to improve over iLQG. LQGMP [8] is similar and additionally handled uncertainty in state and motion. In contrast to these methods, we choose a different parametrization by adding time directly to our state. This results in a lower dimensional state representation compared to having velocities in-state while allowing the flexibility to directly manipulate time. Rawlik et al. [20] used a similar approach with approximate inference for joint optimization of movement and duration in an optimal control framework.

Witkin [21] introduced space-time constraints for generating physically realistic motion in character animation. This paradigm has been used extensively in the animation literature for transferring motion between characters [22], physics based motion transformation [23], and interactive simulation and editing of motion [24], [25]. These methods jointly optimize over space-time using physically grounded cost functions and constraints. In practice, these methods are used in conjunction with human demonstrations to guide the search and have not been applied to robot motion planning.

Witkin and Popovic [26] timewarped motions using C-splines while preserving the structure of the original motion. McCann et al. [27] used physics-based constraints to retime pre-planned motion sequences for character animation. These techniques affect timing while keeping the motion intact.

Work in Human Robot interaction has focused on using timing to enable better interaction [28], [29]. Researchers have looked at how human motion and gestures vary in timing and shape to help convey intentions better. For example, Gielniak and Thomaz studied the importance of spatiotemporal correspondences in generating human-like motion [30]. Whitaker [31] discussed the significance of timing in altering perception of people towards actions, Jeannerod [32] showed that simple reaching and grasping tasks tend to have a fast approach phase and a slow phase for grasping successfully, and Becchio et al. [33] showed that velocity profiles change based on the task being performed in isolation or in a social setting. These studies provide further motivation for extending functional gradient planners to reason about time.

III. CHOMP Preliminaries

CHOMP [3] is a functional gradient optimization algorithm (Fig.3a) that minimizes a cost functional to produce smooth collision-free trajectories. CHOMP represents a trajectory $\xi : [0, 1] \to \mathcal{Q}$ as a smooth function mapping time $\tau \in [0, 1]$ to robot configurations $\xi(\tau) \in \mathcal{Q}$ in the robot's configuration space.

Functionals are functions of functions. CHOMP considers objective functionals $\mathcal{U} : \Xi \to \mathbb{R}$ that map each path $\xi \in \Xi$ to a real number. Here, Ξ is a Hilbert space of trajectories.

The objective functional is a combination of two components: a *smoothness* term that measures the shape of the trajectory, and an *obstacle* term that measures its proximity to obstacles:

$$\mathcal{U}[\xi] = \mathcal{F}_{obs}[\xi] + \lambda \mathcal{F}_{smooth}[\xi] \tag{1}$$

For example, CHOMP uses the integral over squared path tangent norms as the smoothness cost:

$$\mathcal{F}_{smooth}[\xi] = \frac{1}{2}\int_0^1 \left\| \frac{d}{d\tau}\xi(\tau) \right\|^2 \, d\tau \tag{2}$$

The other term in the objective, $\mathcal{F}_{obs}$, encourages collision-free trajectories by penalizing points close to obstacles. Let $c : \mathbb{R}^3 \rightarrow \mathbb{R}$ be a scalar potential defined on the workspace of the robot (Fig.3a). CHOMP, for example, defines a c that increases with proximity to the the nearest obstacle. Let $\mathcal{B} \subset \mathbb{R}^3$ denote the set of points on the exterior of the robot. Let $x : \mathcal{Q} \times \mathcal{B} \rightarrow \mathbb{R}^3$ represent the forward kinematics of the robot, mapping robot configurations to 3D positions. With this, CHOMP defines the obstacle cost as:

$$\mathcal{F}_{obs}[\xi] = \int_0^1 \int_{\mathcal{B}} c(x(\xi(\tau),u)) \left\| \frac{d}{d\tau} x(\xi(\tau),u) \right\| \, du \; d\tau \tag{3}$$

This is the arc-length parametrized line integral (note the scaling by the norm of workspace velocity) of the robot's body points through the workspace cost field.

CHOMP then iteratively minimizes the objective functional $\mathcal{U}$ over the space of trajectories by minimizing a regularized first order Taylor expansion of the objective functional around the current path ξ_i:

$$\xi_{i+1} = \arg\min_{\xi} \; \mathcal{U}[\xi_i] + \bar{\nabla}\mathcal{U}[\xi_i]^T(\xi - \xi_i) + \frac{\eta}{2}\|\xi - \xi_i\|_A^2 \tag{4}$$

where the regularization is with respect to an admissible norm A in Ξ and η is a step size parameter. Taking the gradient of (4) and setting it to zero gives the update rule:

$$\xi_{i+1} = \xi_i - \frac{1}{\eta} A^{-1} \bar{\nabla}\mathcal{U}[\xi_i] \tag{5}$$

where $\bar{\nabla}\mathcal{U}[\xi_i]$ is the functional gradient of the cost functional.

IV. Optimizing In Space-Time

We outline the framework for jointly optimizing in Space-Time using functional gradient descent. Like CHOMP, we consider position trajectories $\xi : [0,1] \rightarrow \mathcal{Q}$ as smooth functions mapping $\tau \in [0,1]$ to robot configurations. Formally, we refer to τ as *pseudo* time; it mimics time, but is fixed during the optimization. We think of τ as a stepping parameter that montotonically increases as we traverse the trajectory; it is 0 at the start and 1 at the goal. With this notation, we introduce a time trajectory $t : [0,1] \rightarrow \mathbb{R}^+$ as smooth function mapping $\tau \in [0,1]$ to time $t(\tau) \in \mathbb{R}^+$. In the waypoint parametrization, t is a vector of time points, $t(\tau)$ is the time at which the robot is in configuration $\xi(\tau)$. It is important to note that ξ and t are independent of each other, and are related only through their dependence on τ. We consider objective functionals $\mathcal{U} : [\Xi, \Pi] \rightarrow \mathbb{R}$ that map a pair of path $\xi \in \Xi$ and timing $t \in \Pi$ to a real number. Here, Π is a Hilbert space of all possible time trajectories. Similar to CHOMP, the objective functional is made up of smoothness and obstacle terms.

A. Smoothness functional

The smoothness functional $\mathcal{F}_{smooth}$ measures the shape of the position trajectory ξ and time trajectory t. We use the integral over squared tangent norms of ξ and t:

$$\mathcal{F}_{smooth}[\xi, t] = \frac{1}{2}\int_0^1 \left(\left\| \frac{d}{d\tau}\xi(\tau) \right\|^2 + \gamma \left\| \frac{d}{d\tau} t(\tau) \right\|^2 \right) d\tau \tag{6}$$

where γ trades off time and position smoothness. $\mathcal{F}_{smooth}$ does not enforce coupling between the path ξ and its timing t; if the obstacle cost is zero, the resulting objective treats the trajectories independently. Minimizing $\mathcal{F}_{smooth}$ moves ξ towards a straight line and leads t to have equal spacing. The smoothness functional implicitly constrains the speed $\|x'\|/t'$, but it can be modified to directly minimize this.[1]

B. Obstacle functional

The obstacle functional $\mathcal{F}_{obs}$ measures proximity to obstacles and encodes interaction between the path ξ and its timing t. We consider a general functional $f : [\Xi, \Pi, \Xi', \Pi'] \rightarrow \mathbb{R}$ and define the obstacle functional:

$$\mathcal{F}_{obs}[\xi, t] = \int_0^1 \int_{\mathcal{B}} f(\xi, t, \xi', t') \; du \; d\tau \tag{7}$$

which is the line integral of the robot's body points through f. The CHOMP obstacle functional (3) is a special case of this general cost functional where f maps the trajectory through forward kinematics to measure a position-dependent workspace potential scaled by the workspace velocity.

C. Functional gradients

We compute the functional gradient of the objective functional by applying the Euler-Lagrange equations [34] for the path ξ and timing t. The functional gradients of the smoothness functional are straightforward:

$$\bar{\nabla}\mathcal{F}^{\xi}_{smooth}[\xi, t] = -\xi''(\tau) \; ; \; \bar{\nabla}\mathcal{F}^{t}_{smooth}[\xi, t] = -\gamma \, t''(\tau) \tag{8}$$

where $\bar{\nabla}\mathcal{U}^{\xi}$ and $\bar{\nabla}\mathcal{U}^{t}$ are the functional gradients with respect to ξ and t respectively. The obstacle functional is more general and its functional gradient is the functional gradient of f with respect to ξ and t. We have:

$$\bar{\nabla}\mathcal{F}^{\xi}_{obs}[\xi, t] = \int_{\mathcal{B}} \left[\frac{\partial f}{\partial \xi} - \frac{d}{d\tau}\frac{\partial f}{\partial \xi'} \right] du \tag{9}$$

$$\bar{\nabla}\mathcal{F}^{t}_{obs}[\xi, t] = \int_{\mathcal{B}} \left[\frac{\partial f}{\partial t} - \frac{d}{d\tau}\frac{\partial f}{\partial t'} \right] du \tag{10}$$

For the CHOMP objective, $\dfrac{\partial f}{\partial t} = 0$ and $\dfrac{\partial f}{\partial t'} = 0$; it has no time gradient. Fig.2 visualizes the gradients for a single iteration in a 2D environment with a circular obstacle. The smoothness gradients (red arrows) push ξ towards a straight line trajectory (Fig.2a) and t towards equal spacing (Fig.2b).

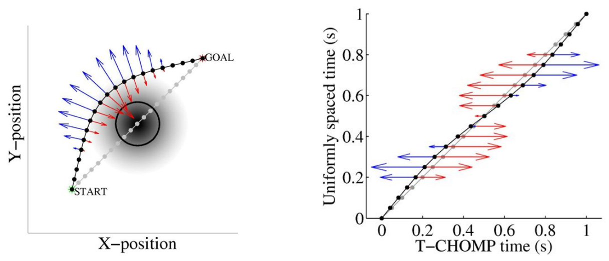
D. Gradient Descent

At every iteration of the optimization, we minimize a regularized first order taylor expansion of our functional $\mathcal{U}$ about the current trajectories (ξ_i, t_i) :

$$\begin{aligned}(\xi_{i+1}, t_{i+1}) = \arg\min_{\xi, t} \; & \mathcal{U}[\xi_i, t_i] + \bar{\nabla}\mathcal{U}^{\xi}[\xi_i, t_i]^T(\xi - \xi_i) \\ & + \bar{\nabla}\mathcal{U}^{t}[\xi_i, t_i]^T(t - t_i) + \frac{\eta}{2}\|\xi - \xi_i\|_A^2 + \frac{\eta'}{2}\|t - t_i\|_B^2 \end{aligned} \tag{11}$$

where B is an allowable metric on Π. Computing the gradient of this expression with respect to ξ and t and setting

[1]$()'$ refers to $\dfrac{d}{d\tau}$



(a) Gradients for position (ξ) (b) Gradients for time (t)

Fig. 2: Smoothness (red) and obstacle (blue) gradients for a velocity scaled obstacle functional (18). Black circle is the obstacle.

them to zero, we arrive at the update rules for the trajectories. The update for ξ is the same as (5) while for t it is:

$$t_{i+1} = t_i - \frac{1}{\eta'} B^{-1} \bar{\nabla} U^t [\xi_i, t_i] \tag{12}$$

V. Constraints

The optimization problem in (11) is unconstrained in position and time. We impose bound constraints on the path ξ to ensure that the robot stays within its joint limits. These are linear inequality constraints for which a closed form update rule can be derived in a manner similar to [35]. We impose monotonicity constraints on the time trajectory t to ensure that the optimization does not move time backwards.

To derive the update rule for the constraints, we explicitly assume a waypoint parametrization for the path ξ and its timing t. t is now a sequence of $n+1$ waypoints $t = [T_1,\ T_2,\ T_3,\ \ldots\ T_{n+1}]$ with $T_0 = 0$ and $T_{n+1} = T_{goal}$, the total duration of the motion. In this paper, we consider a fixed, pre-specified duration for the robot's motion; $T_{goal} \in \mathbb{R}^+$ is constant throughout our optimization. Similarly, the start and end points of the path ξ are assumed fixed.

A. Time monotonicity

We formulate the monotonicity constraint on t as a set of linear inequality constraints:

$$\Delta t = (T_j - T_{j-1}) \geq \epsilon;\ \forall j = 1, 2, \ldots n+1 \tag{13}$$

where $\epsilon \in \mathbb{R}^+$. While $\epsilon = 0$ is enough to ensure monotonicity, in practice this can result in infinite velocities and numerical instabilities. By choosing $\epsilon \in \mathbb{R}^+$ we implicitly constrain the maximum velocity subject to smoothness. To ensure monotonicity is satisfied throughout t, we free the end point T_{n+1}; else we have more constraints than variables. We rewrite the constraints as:

$$\delta_j^T \left(Xt + y \right) \geq \epsilon \implies -\delta_j^T \left(Xt + y \right) + \epsilon \leq 0 \tag{14}$$

where X is a $(n+1) \times (n+1)$ finite-difference matrix $[X(j,j) = 1,\ X(j,j-1) = -1\ ;\ \forall j]$, y is a $(n+1) \times 1$ vector to take care of the start point $[y(1) = -T_0,\ \ y(2, .., n+1) = 0]$ and δ_j is a $(n+1) \times 1$ indicator vector with the jth element equal to 1 and 0 otherwise.

B. Fixed time duration

Previously we assumed a fixed duration for our trajectories, but we had to free the end point T_{n+1} to ensure monotonicity. To counter this, we add a linear equality constraint on the end point:

$$T_{n+1} = T_{goal} \implies \delta_{n+1}^T t - T_{goal} = 0 \tag{15}$$

Algorithm 1 Computing W, z, v for constraints on t

```
$S = S_1 \cup S_2$ ;   $m = |S|$                         ▷ $S_1 := (n+1)$
$W \in \mathbb{R}^{m \times m}$ ; $z, v \in \mathbb{R}^{m \times 1}$
for $x = 1 : m$ do
    $j = S(x)$
    if $j \in S_1$ then                    ▷ Fixed duration (FD) constraint
        $z(x) := \beta$
        $v(x) = \eta' \left( \delta_{n+1}^T t_i - T_{goal} \right) - \delta_{n+1}^T B^{-1} \bar{\nabla} \mathcal{U}^t$
    else                                  ▷ Monotonicity (TM) constraint
        $z(x) := -\alpha_j$
        $v(x) = \eta' \left[ \delta_j^T \left( X t_i + y \right) - \epsilon \right] - \delta_j^T X B^{-1} \bar{\nabla} \mathcal{U}^t$
    for $y = 1 : m$ do
        $k = S(y)$
        if $x = 1$ and $y = 1$ then                              ▷ FD term
            $W(x,y) = \left( B^{-1} \right)_{n+1,n+1}$
        else if $x = 1$ and $y > 1$ then                    ▷ Cross terms
            $W(x,y) = \left( B^{-1} X^T \right)_{n+1,j}$
        else if $x > 1$ and $y = 1$ then                    ▷ Cross terms
            $W(x,y) = \left( X B^{-1} \right)_{j,n+1}$
        else                                                  ▷ TM terms
            $W(x,y) = \left( X B^{-1} X^T \right)_{j,k}$
```

C. Lagrangian

Taking the constraints into the optimization, we have the Lagrangian:

$$\begin{aligned} \mathcal{L}[\xi, t] &= \mathcal{U}[\xi_i, t_i] + \left(\bar{\nabla} \mathcal{U}^{\xi T} \right) (\xi - \xi_i) + \frac{\eta}{2} \| \xi - \xi_i \|_A^2 \\ &+ \left(\bar{\nabla} \mathcal{U}^{tT} \right) (t - t_i) + \frac{\eta'}{2} \| t - t_i \|_B^2 \\ &- \sum_{j=1}^{n+1} \alpha_j \left[\delta_j^T \left(Xt + y \right) - \epsilon \right] + \beta \left(\delta_{n+1}^T t - T_{goal} \right) \end{aligned} \tag{16}$$

where α_j's and β are KKT multipliers and functional gradients $\bar{\nabla} \mathcal{U}^\xi, \bar{\nabla} \mathcal{U}^t$ are evaluated at current estimates (ξ_i, t_i).

D. Time trajectory update

Taking the gradient of the Lagrangian with respect to t and setting it to zero, we get the first order optimality condition:

$$t = t_i - \frac{1}{\eta'} B^{-1} \left[\bar{\nabla} \mathcal{U}^t - \sum_{j=1}^{n+1} \alpha_j \left(X^T \delta_j \right) + \beta \delta_{n+1} \right] \tag{17}$$

The monotonicity constraints are inequality constraints and are either active or inactive, and the linear fixed duration constraint is always active. In a general case, the fixed duration constraint is active at the goal $\beta \neq 0$; $S_1 := (n+1)$ and there is some set $S_2 \subseteq [1, 2, \ldots, n+1]$ of active monotonicity constraints $\alpha_j \neq 0; \forall j \in S_2$. To solve for the unknown scalars α_j, β we can substitute the optimality condition (17) in the KKT complementarity conditions for the constraints $\beta \left(\delta_{n+1}^T t - T_{goal} \right) = 0$; $\alpha_j \left(\delta_j^T \left(Xt + y \right) - \epsilon \right) = 0; \forall j \in S_2$. This results in $|S_1 \cup S_2|$ simultaneous equations in the unknowns, which we write in the form of a matrix vector equation:

$$Wz = v \implies z = W^\dagger v$$

where z is a vector of unknown KKT multipliers. Due to space constraints, we do not detail the derivation of this structure, but Algorithm (1) gives a procedure to compute the KKT multipliers for a general case. Substituting these into (17) gives the updated time trajectory t_{i+1}.

E. What does the update do?

For a specific active constraint j the expression for v in Algorithm 1 is made up of two terms, the magnitude of the current violation at j scaled by the step size η' and a scaled version of the gradient at j. The matrix W measures correlation between all active constraints and is dependent on the metric B and the constraint matrix X. The algorithm takes the gradient and violation, scales it through the inverse correlation matrix and smoothly updates the trajectory. At j, it cancels out the gradient and corrects the violation. Away from j it scales the update based on the metric B. The matrix W is invertible for the current set of time constraints, but in a general case this may not be true. Using the pseudo-inverse amounts to projecting on a lower dimensional set of linearly independent constraints. Throughout the derivation, we did not make any explicit assumptions on the structure of X; the update is thus valid for any linear inequality constraint on t. In fact, joint limit constraints on ξ can be formulated in this way and the resulting W is an invertible submatrix of A^{-1}. To compute the constrained update, we need X, B^{-1} and W. X and B^{-1} can be pre-computed while W needs to be computed at each iteration. In the worst case, we can have active monotonicity constraints on all the $n+1$ points, leading W to have size $(n+2) \times (n+2)$. So, the added complexity of the constrained update over CHOMP is $O(n^3)$.

VI. VELOCITY SCALING

With a framework in place for space-time optimization, we proceed to choose a specific obstacle functional to evaluate the performance of our algorithm. We consider an obstacle functional of the form $f : [\Xi, \Xi', \Pi'] \to \mathbb{R}$:

$$\mathcal{F}_{obs} = \int_0^1 \int_{\mathcal{B}} c(x) \frac{\|x'\|}{t'} du \, d\tau \tag{18}$$

where the obstacle potential c from [3] is scaled by the norm of workspace velocity. We compute the functional gradient of this obstacle cost function:

$$\bar{\nabla}\mathcal{F}_{obs}^{\xi} = \int_{\mathcal{B}} J^T \frac{\|x'\|}{t'} \left[\left(I - \hat{x}'\hat{x}'^T \right) \nabla c - ck + c\frac{x'}{\|x'\|^2}\frac{t''}{t'} \right] \tag{19}$$

$$\bar{\nabla}\mathcal{F}_{obs}^{t} = \int_{\mathcal{B}} \frac{\|x'\|}{(t')^2} \left[x'^T \nabla c + c\frac{x'^T x''}{\|x'\|^2} - 2c\frac{t''}{t'} \right] du \tag{20}$$

where $k = \|x'\|^{-2} \left(I - \hat{x}'\hat{x}'^T \right) x''$ is the curvature vector. The obstacle functional scales the position-dependent potential c by the norm of the workspace velocity and it naturally encourages the trajectory to have lesser velocity and cost.

Fig.2 shows the obstacle gradients (blue) for a single obstacle environment. The gradients push ξ away from the obstacle (Fig.2a) and try to make the trajectory slower near obstacles by spreading the time points apart (Fig.2b).

A. What can T-CHOMP do differently?

For a given objective functional, CHOMP modifies the path ξ to minimize cost. T-CHOMP extends functional gradient planners to optimize over space-time, meaning it can modify both the path ξ and its timing t (or either). With this in mind, we can *broadly* group the behavior of T-CHOMP into the following three classes:

1) T-CHOMP encodes interaction between the path ξ and its timing t through the obstacle functional (7). When the algorithm successfully minimizes this functional (or in its absence), ξ and t become independent, reducing T-CHOMP to CHOMP. In the context of our current obstacle functional, if the path escapes the obstacle's influence, T-CHOMP starts to behave like CHOMP
2) When there is interaction between ξ and t, the cross terms in the functional gradients (19), (20) encourage T-CHOMP to jointly optimize ξ and t. When the path ξ reaches a local minima, T-CHOMP can reduce cost by modifying the timing t (subject to smoothness and monotonicity constraints). For the velocity scaled objective, when T-CHOMP is unable to escape obstacles, it can lower the robot's velocity profile near obstacles
3) Finally, in cases where T-CHOMP freely optimizes both ξ and t, the optimization can fall into different basins of attraction than CHOMP resulting in meaningfully different paths ξ and timing t. (Fig.1a) is a good example where the velocity scaled objective causes T-CHOMP to take a slower, more direct route to the goal.

Obviously, there could be a host of in-between cases, but to highlight our differences to CHOMP, these serve as good representatives.

VII. EXPERIMENTS

We present simulation experiments from T-CHOMP and highlight our results by comparing to CHOMP. In all experiments, we use the velocity scaled obstacle functional (18) for T-CHOMP and the arc-length parametrized functional (3) for CHOMP. We use the same workspace cost potential c and smoothness cost functional $\mathcal{F}_{smooth}$ (2) as CHOMP [3] and choose the metrics A and B to measure total acceleration in ξ and t respectively. We test the following hypotheses:

$\mathcal{H}_1$. T-CHOMP with velocity scaling manipulates time to meaningfully alter the velocity profiles of trajectories in comparison with CHOMP

$\mathcal{H}_2$. T-CHOMP with velocity scaling leads to different basins of attraction resulting in different trajectories in comparison with CHOMP

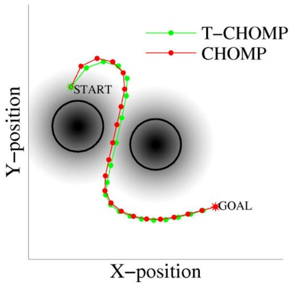
which correspond to the second and third cases discussed in Sec. VI-A. We do not discuss the case without obstacles as it is evident that T-CHOMP directly reduces to CHOMP.

We test two scenarios in simulation: (1) a 2D point robot with planar obstacles, and (2) a 7DOF BarrettWAM attached to Marvin and HERB in realistic kitchen setups. We use the following plots to aid explanation:

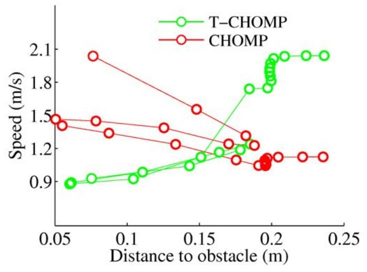
1) Rendered workspace trajectories $x(\xi)$ returned by T-CHOMP and CHOMP
2) Robot's speed $\|x'\| / t'$ at each trajectory point as a function of the distance to the closest obstacle
3) T-CHOMP timing t as a function of CHOMP's timing τ (equally spaced in 2D, parabolically retimed in 7D)

A. 2D Point Robot

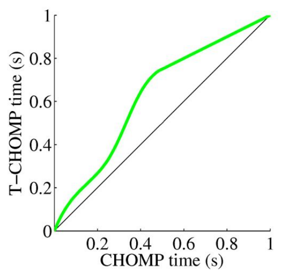
We tested the algorithms on several 2D obstacle avoidance tasks with randomly generated start and end points and



(a) Optimized paths

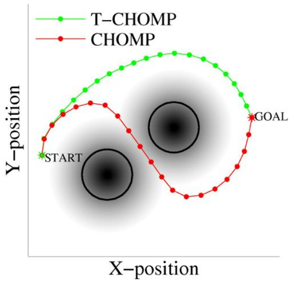


(b) Distance to obstacle vs Speed

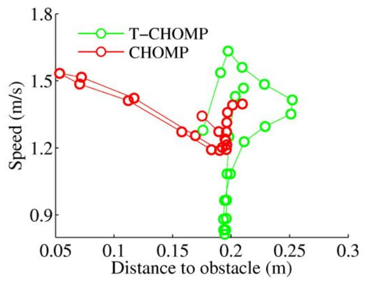


(c) Time comparison

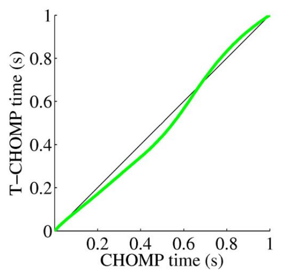
Fig. 3: 2D point robot: Although the paths are nearly identical, retiming significantly reduces cost.



(a) Optimized paths



(b) Distance to obstacle vs Speed



(c) Time comparison

Fig. 4: 2D point robot: T-CHOMP (green) escapes CHOMP's (red) local minimum and produces a different lower-cost trajectory.

circular obstacles. The duration of motion was fixed at 1 second and the algorithms were initialized with equally spaced points in time and straight line paths in space. We used the following *default* parameters: $n = 99, \epsilon = 0.005, \eta = \eta' = \frac{1}{500}, \lambda = \gamma = 1$ and ran until convergence.

Our resulting trajectories fell primarily into two classes. The first, illustrated in Fig.3, is where both paths (green for T-CHOMP and red for CHOMP in Fig.3a) are nearly identical, but T-CHOMP achieves lower cost by retiming. By manipulating timing, T-CHOMP slows down near obstacles, lowering cost. This is evident in Fig.3b where CHOMP (red) shows no interesting structure, but there is significant correlation between speed and obstacle distance for T-CHOMP. At points closer to the obstacle, the speed is less, but farther away it increases dramatically until it saturates at a distance of around 0.2 m which is exactly the obstacle tolerance for this experiment. The fact that the speed saturates means that the time difference for those points is $\approx \epsilon$. Fig.3c shows the comparison of time trajectories between the algorithms with T-CHOMP on the y-axis. Near the middle, T-CHOMP slows down significantly as it is near obstacles, then speeds up at the end when it is outside the obstacle's influence. This is exactly what we hypothesized ($\mathcal{H}_1$): even if the path is stuck, T-CHOMP can modify timing to reduce cost.

The second class, illustrated in Fig.4, is where the final paths of the algorithms are different (Fig.4a). CHOMP gets stuck in the local minima between obstacles and returns a high cost path while T-CHOMP finds a significantly lower-cost path away from the obstacles, just as hypothesized in $\mathcal{H}_2$. Once out of the obstacles' influence, the obstacle functional reaches a near global minimum and the algorithm starts to aggressively reduce the smoothness objective, pushing t towards equal spacing. This is in line with Fig.4c which shows minimal re-timing. This is exactly what we expected

Parameters	CHOMP objective	T-CHOMP objective
Default	1.7 ± 1.2	23.6 ± 1.8
$n_{obs} = 1$	-0.6 ± 0.3	3.3 ± 0.9
$n_{obs} = 5$	0.4 ± 1.3	31.2 ± 1.3
$T_{goal} = 2.0\ s$	-0.8 ± 0.4	6.1 ± 0.4
$T_{goal} = 0.75\ s$	1.1 ± 1.7	32.0 ± 2.1
$\epsilon = 0.01\ s$	1.4 ± 0.8	4.7 ± 0.8
$\epsilon = 0.001\ s$	-3.4 ± 5.6	20.7 ± 6.4

TABLE I: Cost improvement for T-CHOMP over CHOMP from 100 2D tests, measured as percentage of CHOMP trajectory cost. Positive value implies T-CHOMP has lower cost. All tests use *default* parameters (Sec. VII-A) with $n_{obs} = 3$. Error metric is Standard Error.

the algorithm to do (Sec. VI-A); when the path is able to escape obstacles, we tend more towards CHOMP. This also explains the lack of structure in Fig.4b.

Table I shows the percentage cost improvement of T-CHOMP over CHOMP on 100 random 2D scenarios with varying parameters. With the default parameters, T-CHOMP significantly outperforms CHOMP when compared under the T-CHOMP objective. This is in line with $\mathcal{H}_1$, and shows that T-CHOMP reduces cost even when stuck in local optima of the path. Predominantly, the T-CHOMP path ξ tends to be similar to CHOMP and there is variation in t. But in around 5% of our tests, we found that T-CHOMP escapes CHOMP's local optima ($\mathcal{H}_2$), leading to paths with significantly lower costs. This is reflected in the results: T-CHOMP also performs better than CHOMP under the CHOMP objective.

B. 7D Robot Manipulator

We implemented T-CHOMP in the OpenRAVE [36] simulation environment as an add-on to the CHOMP implementation from [3] and evaluated the algorithms with Marvin and HERB in a kitchen environment. In all experiments, the algorithms were initialized with straight line paths in configuration space. CHOMP was initialized with equally spaced points from 0 to 1 second and its output retimed by

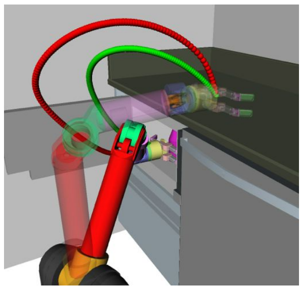


Fig. 7: End-effector paths for the task in Fig.5, initialized to a longer duration ($\approx 10s$). T-CHOMP (green) takes a completely different path from CHOMP (red).

OpenRAVE's parabolic retimer to be within velocity and acceleration limits. T-CHOMP was initialized with the duration of this retimed trajectory, with equally spaced points. Default parameters from Sec. VII-A were used.

We set up two types of reaching tasks for the robots in the kitchen environment: (1) Fig.5 where Marvin's arm starts off at a configuration above the kitchen counter and moves towards a bottle inside the microwave on its lower left and (2) Fig.6 where HERB's right arm tries to get around a box placed on the countertop. Just like in the 2D case, our results fell primarily into two classes.

The first class, illustrated in Fig.5a, is where the optimized paths are nearly identical. The distance-speed plot in Fig.5b is very similar to Fig.3b, T-CHOMP slows the robot near obstacles and speeds it up away from them. This is also reflected in Fig.5c, T-CHOMP makes the robot slower until it leaves the counter (0.3 s), then speeds it up (0.5 s) till it gets near the microwave after which it slows until the goal.

The second class, illustrated in Fig.6, is where the two trajectories are different. CHOMP follows the gradient of the obstacle potential to move away from the box resulting in a trajectory that makes a wide berth of the obstacle. T-CHOMP on the other hand is attracted to the optima above the box. By manipulating timing and reducing the velocity near the box, T-CHOMP lowers the cost of the path, making it desirable to move above the box rather than avoiding it altogether. This results in completely different end-effector paths as witnessed in Fig.6a. A look at Fig.6b reveals a similarity to the 2D example in Fig.4: there is little structure and the trajectory has lower speed throughout. Fig.6c shows the expected slowdown in timing near the middle.

Fig.7 provides further proof for our hypotheses. Given a longer duration for the motion ($\approx$ 5x previous), T-CHOMP manipulates timing to reach a different local minima that takes a direct route to the microwave. This behavior highlights an interesting property of our algorithm: We get a different path and timing for every new duration. We believe this to be an interesting feature of our algorithm that can be exploited to better optimize the objective. These scenarios show that our hypotheses are valid in higher dimensions. Joint space-time optimization allows T-CHOMP to directly exploit timing, resulting in meaningfully different behavior. The attached video elaborates these results further.

VIII. Discussion

We introduced T-CHOMP, a novel motion planner that extends functional gradient optimization to space-time. We showed that T-CHOMP with a velocity scaled objective functional results in meaningful changes to trajectories in comparison with CHOMP. While the method is general, the implementation is sensitive to parameters, especially the total duration (T_{goal}) and monotonicity constraint (ϵ).

Increasing T_{goal} affords the algorithm more flexibility in modifying timing but also reduces speed throughout the trajectory, effectively reducing cost. While this can result in different local optima (Fig.7), this can also result in risky behavior where the algorithm thinks that it has low cost even near obstacles. Table I shows results from 2D scenarios that highlight this issue: doubling the duration reduces T-CHOMP performance significantly. On the other hand, reducing the duration far too much can result in numerical instabilities. In general, the total duration has to be determined while taking into account the length of the motion, the robot's velocity limits and the number of points on the trajectory. A rule of thumb can be twice the time taken to travel the straight line trajectory at maximum velocity.

A loose monotonicity constraint (smaller ϵ) allows for more freedom in manipulating timing, but can lead to poor time smoothness, large increases in acceleration and numerical instabilities. Too tight a constraint stifles the flexibility and reduces performance, as can be seen from the results in Table I. Critical to choosing a good ϵ are the robot's velocity limits; ϵ can be larger than the time to traverse two successive waypoints of the straight line trajectory at maximum velocity.

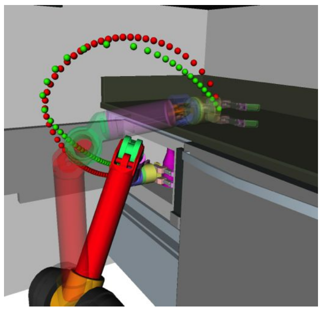
Many of our issues with local optima and parameter sensitivity are due to a fixed motion duration which restricts the solution space the optimizer can choose from. We believe that key to addressing these issues is the ability to afford the optimizer freedom to choose the total duration. T-CHOMP does not have explicit velocity constraints, rather it implicitly constrains velocity through smoothness priors and monotonicity. We believe that adding velocity constraints directly into the optimization is the correct way to restrict the search space. Combined with variable duration optimization, we see T-CHOMP as generating valid controls for the task at hand, subject to the robot's physical constraints.

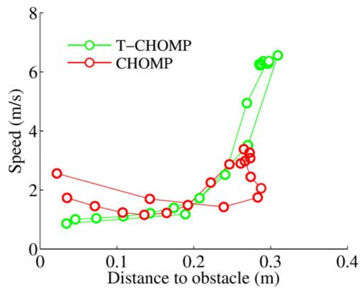
T-CHOMP can represent more general context dependent cost functionals that depend both on position and time. While this offers more freedom, it is hard for an end user to specify the correct functional to match a task. We are interested in exploring the use of Inverse Optimal Control techniques [37] to learn these cost functions from demonstrations. Time dependency in the cost functional should allow us to naturally handle dynamic obstacles in the environment and is another interesting direction for future work.

References

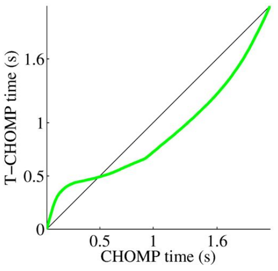
[1] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *IEEE-RA*. IEEE, 1993, pp. 802–807.

[2] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *IEEE ICRA*. IEEE, 2009, pp. 489–494.


(a) Optimized End-effector paths

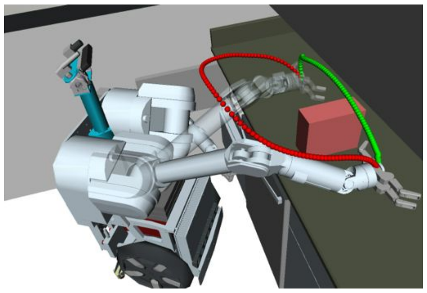


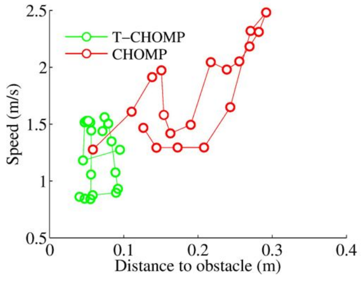
(b) Distance to obstacle vs Speed



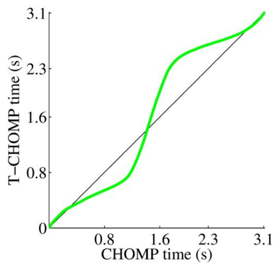
(c) Time comparison

Fig. 5: Marvin robot: Although paths are nearly identical, retiming significantly reduces cost.


(a) Optimized End-effector paths



(b) Distance to obstacle vs Speed



(c) Time comparison

Fig. 6: HERB robot: T-CHOMP (green) escapes CHOMP's (red) local minimum and produces a different lower-cost trajectory.

[3] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," 2013.

[4] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *IEEE ICRA*. IEEE, 2011, pp. 4569–4574.

[5] C. Park, J. Pan, and D. Manocha, "Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments." in *ICAPS*, 2012.

[6] J. Schulman, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," *RSS*, 2013.

[7] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *ACC*. IEEE, 2005, pp. 300–306.

[8] J. Van Den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *IJRR*, vol. 30, no. 7, pp. 895–913, 2011.

[9] J. E. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *IJRR*, vol. 4, no. 3, pp. 3–17, 1985.

[10] Z. Shiller and S. Dubowsky, "On computing the global time-optimal motions of robotic manipulators in the presence of obstacles," *IEEE-RA*, vol. 7, no. 6, pp. 785–797, 1991.

[11] J. M. Hollerbach, "Dynamic scaling of manipulator trajectories," in *ACC*. IEEE, 1983, pp. 752–756.

[12] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. of guidance,control,and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[13] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *IJRR*, vol. 5, no. 1, pp. 90–98, 1986.

[14] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE-RA*, vol. 8, no. 5, pp. 501–518, 1992.

[15] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *IEEE ICRA*. IEEE, 2010, pp. 2397–2403.

[16] J. Canny, J. Reif, B. Donald, and P. Xavier, "On the complexity of kinodynamic planning," in *Foundations of Computer Science, 1988., 29th Annual Symposium on*. IEEE, 1988, pp. 306–316.

[17] R. F. Stengel, *Optimal control and estimation*. Dover publns., 1986.

[18] D. Jacobson and D. Mayne, "Differential dynamic programming," 1970.

[19] M. Toussaint, "Robot trajectory optimization using approximate inference," in *ICML*. ACM, 2009, pp. 1049–1056.

[20] K. Rawlik, M. Toussaint, and S. Vijayakumar, "An approximate inference approach to temporal optimization in optimal control." in *NIPS*, 2010.

[21] A. Witkin and M. Kass, "Spacetime constraints," in *ACM SIGGRAPH*, vol. 22, no. 4. ACM, 1988, pp. 159–168.

[22] M. Gleicher, "Retargetting motion to new characters," in *ACM SIGGRAPH*. ACM, 1998, pp. 33–42.

[23] Z. Popović and A. Witkin, "Physically based motion transformation," in *ACM SIGGRAPH*, 1999, pp. 11–20.

[24] M. F. Cohen, "Interactive spacetime control for animation," in *ACM SIGGRAPH*, vol. 26, no. 2. ACM, 1992, pp. 293–302.

[25] J. Popović, S. M. Seitz, M. Erdmann, Z. Popović, and A. Witkin, "Interactive manipulation of rigid body simulations," in *ACM SIGGRAPH*, 2000, pp. 209–217.

[26] A. Witkin and Z. Popovic, "Motion warping," in *ACM SIGGRAPH*. ACM, 1995, pp. 105–108.

[27] J. McCann, N. S. Pollard, and S. Srinivasa, "Physics-based motion retiming," in *ACM SIGGRAPH/Eurographics SCA*, 2006, pp. 205–214.

[28] M. Huber, M. Rickert, A. Knoll, T. Brandt, and S. Glasauer, "Human-robot interaction in handing-over tasks," in *IEEE RO-MAN*. IEEE, 2008, pp. 107–112.

[29] M. Cakmak, S. S. Srinivasa, M. K. Lee, J. Forlizzi, and S. Kiesler, "Human preferences for robot-human hand-over configurations," in *IEEE/RSJ IROS*. IEEE, 2011, pp. 1986–1993.

[30] M. J. Gielniak and A. L. Thomaz, "Spatiotemporal correspondence as a metric for human-like robot motion," in *ACM/IEEE HRI*, 2011.

[31] H. Whitaker and J. Halas, *Timing for animation*. Taylor & Francis US, 2002.

[32] M. Jeannerod, "The timing of natural prehension movements." *Journal of motor behavior*, 1984.

[33] C. Becchio, L. Sartori, and U. Castiello, "Toward you the social side of actions," *Current Directions in Psychological Science*, vol. 19, no. 3, pp. 183–188, 2010.

[34] I. M. Gelfand and S. V. Fomin, *Calculus of variations*. Courier Dover Publications, 2000.

[35] A. D. Dragan, N. D. Ratliff, and S. S. Srinivasa, "Manipulation planning with goal sets using constrained trajectory optimization," in *IEEE ICRA*. IEEE, 2011, pp. 4582–4588.

[36] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, p. 79, 2008.

[37] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI*, 2008, pp. 1433–1438.