2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

# Collaborative Interaction Models for Optimized Human-Robot Teamwork

Adam Fishman[1,2] Chris Paxton[1], Wei Yang[1], Dieter Fox[1,2] Byron Boots[1,2], and Nathan Ratliff[1],

***Abstract*— Effective human-robot collaboration requires informed anticipation. The robot must anticipate the human's actions, but also react quickly and intuitively when its predictions are wrong. The robot must plan its actions to account for the human's *own* plan, with the knowledge that the human's behavior will change based on what the robot actually does. This cyclical game of predicting a human's future actions and generating a corresponding motion plan is extremely difficult to model using standard techniques. In this work, we describe a novel Model Predictive Control (MPC)-based framework for finding optimal trajectories in a collaborative, multi-agent setting, in which we simultaneously plan for the robot while predicting the actions of its external collaborators. We use human-robot handovers to demonstrate that with a strong model of the collaborator, our framework produces fluid, reactive human-robot interactions in novel, cluttered environments. Our method efficiently generates coordinated trajectories, and achieves a high success rate in handover, even in the presence of significant sensor noise.**

## I. Introduction

Human behavior is determined by a mixture of intent, world prediction, anticipation, physical limitations, and more. When planning in the presence of people, robotic decision processes often encapsulate these diverse desiderata under the lid of a black box dynamics function. When the robot and human's goals are independent [1]–[3], this model has been very successful.

However, cooperating to achieve shared goals is more difficult. Take the human-robot hand-over task shown in Fig. 1 as an example, where a robot must receive some object from a human collaborator. Humans will act based on what they imagine the robot will do [4], and, conversely, the robot should choose actions based on its best estimate of the human's intention. Predicting human intention while planning is not new, this has been explored in anticipatory planning [5], and prior work has modeled human kinematics and dynamics in order to achieve collaborative manipulation tasks [6]–[8].

However, humans often do not act according to plan. Any robot planner that tries to predict their intentions must be highly reactive. We propose a Model Predictive Control (MPC) approach, which models both human and robot as separate, fully-actuated actors in a combined trajectory optimization problem. Our MPC approach allows the robot to determine the most effective form of collaboration while still being able to react to changing circumstances and noisy sensor data. We specifically apply our approach to the problem of human-robot handover [9]–[12]. The core problem is that the human and robot must coordinate on where and how the handover is to take place [9]. In effect, we must balance between the two cost functions for human and robot, avoiding obstacles while finding the most logical location for both to reach.

[1] NVIDIA {cpaxton, wyang, nratliff}@nvidia.com
[2] University of Washington {afishman, fox, bboots}@cs.washington.edu

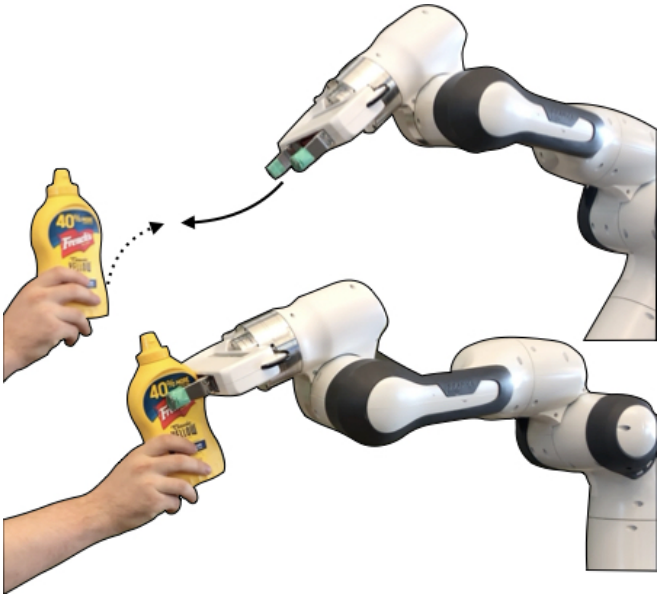


Fig. 1: Coordinating a fluid human-robot handover requires an estimate of the human's plan, so that the robot can be in position to make the handover at the correct time. Our algorithm can achieve smooth and natural human-robot collaborative motions in a variety of scenarios, even in the presence of obstacles and sensor uncertainty.

This combined human-robot system is both partially observable and under-actuated since the robot has no real control over the human and cannot directly observe factors influencing their decisions. Therefore, at a minimum, our planner must be *reactive* [13] to unforeseen human behavior. We follow a real-time Model-Predictive Control (MPC) paradigm and re-optimize with each new observation. Computation speed is also crucial. We employ a modern motion optimization strategy, which leverages fast Gauss-Newton solvers [14]–[16], and assume relevant aspects of the human and robot are fully actuated.

Additionally, to ensure spatial consistency of the resulting reactive behavior through re-optimization, we introduce a novel class of explicit sparse reward terms, *i.e.*, negative costs, around the target. Within a certain radius, the robot is explicitly rewarded for approaching the target, thus extending the target's influence beyond a terminal potential to each intermediate time step. The system is therefore able to compromise between goal accumulation and trajectory smoothness.

We evaluate our technique in both simulated ablation studies as well as real-world handovers between a human participant and a Franka robot using a real-time perception system. We show that, especially in the presence of obstacles, our technique enables the robot to anticipate the human's actions leading to well-coordinated, quick, and smooth handover behavior while timing the handover

978-1-7281-6211-9/20/$31.00 ©2020 IEEE

better than the alternatives, both quantitatively and qualitatively.

## II. Related Work

Modeling human behavior is crucial for successful human-robot collaborative manipulation and has been explored in a variety of contexts [3], [6], [11], [17]. In addition, many recent methods for human-robot handover use perception and some manner of human modeling to achieve reactivity [7], [11], [18]. However, these models are usually uni-directional, with information flowing from prediction to planner but not vice versa. For example, Ziebart et al. [1] used predicted goal-oriented pedestrian behavior to augment navigation planners to minimize interaction. Similarly, Mainprice et al. [2] modeled human reaching behaviors to reduce interaction or collision events while working along-side humans. Maeda et al. [11] use probabilistic motion primitives to model both humans and robots in a variety of collaborative tasks, including handover. Zhou et al. [17] used a recurrent neural net to model human activity for collaboration in the operating room.

In reality, however, the human will respond to qualities of the robot's motion, *e.g.*, speed and shape, trying to estimate and adapt to its motion. Humans and robots can collaborate more effectively if the robot's motions appear legible to humans [4] and allow the humans to understand the robot's goals [19]. One way to achieve legibility is to use human demonstration data to teach the robot [11]. Another approach focuses on jointly modeling the human-robot system as some sort of hybrid planning problem, [3], [6], [20], and try to structure the problem to ensure effective collaboration.

Human-robot handovers are a particularly well-studied area for human-robot collaboration, with applications both to industry [7], [21] and to in-home assistance [10]. Much prior work analyzes the formulation of the human handover and how to structure the action naturally [9], [10]. This can be particularly well represented as a hybrid planning problem [6], [20]. Toussaint et al. [6] proposed a method for offline planning based on Task and Motion Planning. This allows for longer-horizon planning across grasps as compared to our method, but is inherently less reactive. Other work used a dyadic model for collaboration between a human and a robot [20].

Our method applies more specifically to the approach phase of the handover. Related ideas include exploiting a database of human demonstrations to produce natural and fluid plans [22]. Likewise, Maeda et al. [11] use imitation learning to mimic human behavior and Medina et al. [12] use a human-inspired dynamics controller to model the entire action: approach, grasp, retract. These methods are well-suited for controlled interaction settings, but generalizing them to handle the diversity of speed and environmental variations encountered in the real world is challenging. Peternel et al. [7] also model the human during collaborative manipulation, but their goal is to minimize risk of injury, whereas our goal is to achieve fluid collaboration in the presence of obstacles.

Our work relies on motion optimization approaches that are both fast and expressive [14]–[16]. Motion planning as an optimization problem was first presented in [23], and accelerated in a quick progression of work [24]–[26]. These early optimizers addressed primarily the subproblem of smooth collision avoidance. The work of [15], [27], [28], extended the paradigm showing that generic second-order Gauss-Newton optimizers out-of-the-box could solve a more general class of constrained motion optimization problem. Soon thereafter, Mukadam et al. [14] demonstrated that standard factor graph tools could drastically simplify the modeling. We build on these ideas here, using a factor graph to model the problem and fast modern optimizers to solve the continuous optimization loop in real time.

While our setting is fundamentally partially observable, we do not address the Partially Observable Markov Decision Process (POMDP) problem directly, other than to use standard, reactive maximum a posteriori (MAP) approximation techniques [13] to motivate the importance of continuous re-optimization. Using maximum-likelihood observations and active replanning has proven useful before, even for very complex multi-stage tasks [29]. Other approaches use Monte Carlo sampling to explore possible outcomes for various actions [3], [30], [31]. Some POMDP work has even actively modeled uncertainty over human intention [3], [32], particularly in the context of autonomous vehicles [3].

## III. Theoretical framework

In games, agents try to optimize individual objectives [33], but collaborative tasks require cooperation. When collaborating, agents collectively optimize a single system objective. In this section, we formalize the collaborative system. We derive predictive models for each external, *i.e.*, uncontrolled, agent and an optimal control objective for the controlled agent, *i.e.*the robot. We then show that if the models of the external agents' behavior are sufficiently predictive, the controlled agent can achieve a stable *collaborative equilibrium* by choosing actions according to its objective.

We consider a system constituting $N+1$ total agents. Let the $0^{\text{th}}$ agent denote the controlled agent and agents $1,...,N$ be external, uncontrolled, but collaborating agents.

### A. Collaborative interaction model

Denote the $i^{\text{th}}$ agent's trajectory by $\boldsymbol{\xi}_i = (\mathbf{q}_i^0, \mathbf{q}_i^1, \cdots, \mathbf{q}_i^{T+1})$. Let $\boldsymbol{\varsigma}_i^t = (\mathbf{q}_i^{t-1}, \mathbf{q}_i^t, \mathbf{q}_i^{t+1})$ denote the trajectory's $t^{\text{th}}$ second-order clique,[1] a triple of consecutive positions used to represent position and the corresponding finite-difference approximations of velocity, and acceleration at each time step [28].

We define the joint *collaborative system trajectory* as $\boldsymbol{\xi} = (\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, \cdots, \boldsymbol{\xi}_N)$ and denote its constituent 2nd-order cliques by $\boldsymbol{\varsigma}^t = (\boldsymbol{\varsigma}_0^t, \boldsymbol{\varsigma}_1^t, ..., \boldsymbol{\varsigma}_N^t)$.

Denoting the space of all collaborative system trajectories by $\Xi$, we define the system's *collaborative interaction model* (or simply its *collaboration model*) $\mathcal{M} = \{C, G, H\}$ as

$$\min_{\boldsymbol{\xi}} C(\boldsymbol{\xi}) \quad \text{s.t.} \quad G(\boldsymbol{\xi}) \leq 0,\, H(\boldsymbol{\xi}) = 0 \tag{1}$$

where $C : \Xi \to \mathbb{R}$ is the collaborative cost, $G : \Xi \to \mathbb{R}^k$ are $k$ inequality constraint functions, and $H : \Xi \to \mathbb{R}^l$ are $l$ equality constraint functions. For compactness, we use $\Xi_{\mathcal{M}} \subset \Xi$ to denote the feasible set of trajectories that satisfy the constraints $G$ and $H$. We can then write the collaborative interaction model as $\boldsymbol{\xi}^* = \operatorname{argmin}_{\boldsymbol{\xi} \in \Xi_{\mathcal{M}}} C(\boldsymbol{\xi})$. The model for a given collaborative system can change incrementally over time as the environment, the agents' goals, or the agents themselves change. We assume that the optimizer is able to track solutions over time within a

[1] Here we use superscripts just for notational convenience of time indexing, not to be confused with the component indexing of tensor notation.

continuous optimization loop, such as Model Predictive Control (MPC). Note that both the costs / constraints and the set of available trajectories $\Xi_M$ usually change from cycle to cycle updated with the latest estimates of the world and agent states.

In our experiments, we export a $k^{\text{th}}$-order Markov structure in the system [28] enabling us to write the collaborative interaction model of Equation 1 in clique notation as

$$\min_{\xi} \sum_{t=1}^{T} c_t(\boldsymbol{\varsigma}^t) \;\text{ s.t. }\; g_t(\boldsymbol{\varsigma}^t) \leq 0,\, h_t(\boldsymbol{\varsigma}^t) = 0 \;\; \forall t. \tag{2}$$

Often, more complex task spaces are defined on these cliques by transforming them through differentiable task maps where objective terms may reside. It is common to represent the task spaces, *i.e.*, the co-domains of the task maps, with maximal coordinates, which are an explicit representation of the task space constrained to match the output of the task map. Following this paradigm, we define our optimization costs and constraints in maximal coordinates on the relevant task space. We also use soft constraints implemented as unconstrained penalties in our experiments as in [34]. In this section, though, we use the more compact notation given above for succinctness and generality.

The key intuition behind our model is that although we cannot explicitly control the $N$ external collaborating agents, we assume we can sufficiently predict their behavior and treat prediction errors as system disturbances. We make this assumption concrete below and explore it experimentally in Sec. IV-B.

When the collaborative interaction model has a unique global minimum, that minimizer acts as an equilibrium point and becomes predictable by the agents in a way we can exploit in our model (explored below in Section III-C). We, therefore, call the global minimum the system's *collaborative equilibrium* and say the system is well-defined if it has a unique global minimum. In this work, we assume both that the global minimum is well defined and that an optimizer will be able to track the global minimum over time. In practice, these assumptions amount to the agents mutually knowing the higher-level collaboration plan either in advance or by sufficiently communicating it to each other unambiguously on the fly. For complex tasks, there may be many local minima or even regions of equally good global minima, representing different equilibria. In these cases, the system would require additional estimation machinery to maintain predictive distributions across external agent behavior, which we do not address here.

We start by defining explicitly the agents' individual predictive *collaborative behavior models* implicit in the above collaborative interaction model. Let $\Xi_{\mathcal{M}}[\boldsymbol{\xi}_i]$ denote the feasible set of system trajectories where the $i^{\text{th}}$ agent's trajectory is fixed at $\boldsymbol{\xi}_i$. We define the $i^{\text{th}}$ agent's *predictive collaborative cost* to be

$$c_i(\boldsymbol{\xi}_i) = \min_{\boldsymbol{\xi}^{\setminus i} \in \Xi_{\mathcal{M}}[\boldsymbol{\xi}_i]} C(\boldsymbol{\xi}_i, \boldsymbol{\xi}^{\setminus i}), \tag{3}$$

where with a slight abuse of notation, we use $C(\boldsymbol{\xi}_i, \boldsymbol{\xi}^{\setminus i})$ to denote the collaborative cost evaluated at the joint system trajectory defined by agent $i$'s trajectory $\boldsymbol{\xi}_i$ and the remaining system trajectories $\boldsymbol{\xi}^{\setminus i}$ of all other agents $j \neq i$. This cost encodes the agent's action criteria under an assumption that all other agents are predicted as having optimal collaborative responses under the system's collaboration model. Note that these predictive models are assumed to know agent $i$'s intent (the trajectory $\boldsymbol{\xi}_i$). While this assumption is generally wrong, as the robot cannot truly know an external agent's intent, we will see below that it is valid at the system's collaborative equilibrium where equilibrium behavior becomes mutually predictable (see Section III-C).

Each agent then has its own individual *collaborative behavior model* of the form

$$\boldsymbol{\xi}_i^* = \underset{\boldsymbol{\xi}_i \in \Xi_{\text{M}}^i}{\text{argmin}} c_i(\boldsymbol{\xi}_i), \tag{4}$$

where $\Xi_{\text{M}}^i$ is the set of all trajectories $\boldsymbol{\xi}_i$ for the $i^{\text{th}}$ agent for which $\Xi_{\text{M}}[\boldsymbol{\xi}_i]$ is nonempty, *i.e.*, $\Xi_{\text{M}}^i = \{\boldsymbol{\xi}_i \mid \Xi_{\text{M}}[\boldsymbol{\xi}_i] \neq \emptyset\}$

### B. Stability of the predictive controller

We adopt definitions of stability from control theory and say that an assignment of behavior generation algorithms to the agents are collectively, or asymptotically, stable around the equilibrium if the joint system evolves stably, or stably asymptotically, around the system trajectory. Under this notion of stability, we can make following statement.

**Lemma III.1.** *Suppose we have a collaborative system and a corresponding collaboration model $\mathcal{M}$. If we can say that a MPC algorithm over $\mathcal{M}$ rejects $\epsilon$-disturbances and that each agent's collaborative behavior model is $\epsilon$-predictive of the agent's next action, including the controlled agent's execution under the environment's stochasticity, then controlling the controlled agent with the MPC algorithm will create system behavior that is stable around the collaborative equilibrium of $\mathcal{M}$.*

In other words, if our collaboration model is sufficiently predictive for the external agents and we control our controllable agent using the collaborative behavior model derived from it, the combined system behavior is stable around the collaborative equilibrium.

Note that in this stability statement, the metric $\epsilon$-predictive is undefined. This is because the statement will hold as long as the definition of $\epsilon$-predictive is consistent with the definition of $\epsilon$-disturbances, *i.e.*, the range of system deviations that can be handled by MPC. While we cannot explicitly control the external agents, if we can predict their behavior sufficiently well, then we may treat deviations as system disturbances. With this, we do not need to assume that the external agents generate behavior with the same collaborative system model, as the collaborative behavior models induced by the system are sufficiently predictive.

### C. Mutual predictability of the collaborative equilibrium

Each collaborative behavior model implicitly uses a *conditional* model to predict the behavior of external agents. Specifically, under agent $i$'s collaborative behavior model, the cost $c_i(\boldsymbol{\xi}_i)$ optimizes over each external agent $j \neq i$ *given* agent $i$'s trajectory $\boldsymbol{\xi}_i$, thus modeling the response the other agents would have if they were given knowledge of $\boldsymbol{\xi}_i$. The model assumes that all responding agents know the agent $i$'s intent, which is in general not true. However, equilibrium behavior has a mutual predictability property which enables all agents behavior to be predictable, thereby validating the conditional model specifically at the collaborative equilibrium.

Equilibrium predictions of other agent's behavior are both reflexive and transitive, creating a stationarity property of

the predictions. For instance, let $\xi_j^{\mathcal{M}}(\xi_i)$ be the implicit prediction made by agent $i$ of how agent $j$ will respond to agent $i$'s intended trajectory $\xi_i$. Let $\xi^* = \{\xi_i^*\}_{i=1}^N$ denote the collaborative equilibrium of system $\mathcal{M}$. Then for all $i$,$j$ we have $\xi_j^{\mathcal{M}}(\xi_i^*) = \xi_j^*$. Therefore, $\xi_i^{\mathcal{M}}(\xi_j^{\mathcal{M}}(\xi_i^*)) = \xi_i^*$ (reflexive) and $\xi_k^{\mathcal{M}}(\xi_j^{\mathcal{M}}(\xi_i^*)) = \xi_k^* = \xi_k^{\mathcal{M}}(\xi_i^*)$ (transitive).

In other words, each agent predicts an equilibrium response under equilibrium behavior. Even though the agent uses a conditional predictive model which assumes external agents know the agent's intent, specifically at the collaborative equilibrium, the agent's intended behavior becomes predictable as part of the equilibrium behavior validating the use of the conditional model.

## IV. Human-Robot Handover using Finite-Horizon Optimization

In this section, we formulate the handover task as an application of our general framework where the collaborative model optimizes for the human and robot successfully reaching each other to perform the handover. We detail the objective terms used in our models (Sections IV-A, IV-B, IV-C) and discuss implementing spatially consistent behavior using finite-time-horizon MPC (see Section IV-D).

In this section, we consider the robot to be the controlled agent (agent 0) and the human to be the uncontrolled external agent (agent 1), and denote their trajectories as $\xi^{\mathrm{R}} = \{\mathbf{q}_i^{\mathrm{R}}\}_{i=0}^T$ (robot) and $\xi^{\mathrm{A}} = \{\mathbf{q}_i^{\mathrm{A}}\}_{i=0}^T$ (external agent, *i.e.*human), respectively. We focus here on defining the unconstrained objective, making the common assumption that many constraints can be naturally modeled as soft constraints using fixed penalties (see, for instance, [34]). This is a reasonable approximation, especially since stochasticity makes the optimization inherently approximate.

The collaboration objective can be decomposed into three terms, a robot specific term, an external agent (human) specific term, and an interaction term.

$$C(\xi) = \lambda_{\mathrm{R}} c^{\mathrm{R}}(\xi^{\mathrm{R}}) + \lambda_{\mathrm{A}} c^{\mathrm{A}}(\xi^{\mathrm{A}}) + \lambda_{\mathrm{I}} c^{\mathrm{I}}(\xi^{\mathrm{R}}, \xi^{\mathrm{A}}) \tag{5}$$

We detail these three terms in the following sections.

### A. Modeling the robot

First, we define the cost modeling the robot trajectory,

$$c^{\mathrm{R}}(\xi^{\mathrm{R}}) = \sum_{i=0}^{T} c^{\mathrm{R}}(\mathbf{q}_i^{\mathrm{R}}, \dot{\mathbf{q}}_i^{\mathrm{R}}, \ddot{\mathbf{q}}_i^{\mathrm{R}}), \tag{6}$$

where $T$ is the total number of time steps and $\mathbf{q}^{\mathrm{R}}$, $\dot{\mathbf{q}}^{\mathrm{R}}$, $\ddot{\mathbf{q}}^{\mathrm{R}}$ are the position, velocity, and acceleration of the joints of the robot in configuration space. In what follows, we will also use $\mathbf{x}_i^{\mathrm{R}} = \phi(\mathbf{q}_i^{\mathrm{R}}) = [\mathbf{R}_i^{\mathrm{R}}, \mathbf{t}_i^{\mathrm{R}}]$ to represent the 6-DOF pose of the end effector in the world frame, after applying the forward kinematics function $\phi(\cdot)$.

Equation 6 can be split into the sum of individual cost functions, which we define in the following sections.

**Obstacle avoidance and joint constraints.** To prevent hitting the joint limits and to avoid obstacles, we include three cost functions $c_{\text{joint}}(\mathbf{q}_i^{\mathrm{R}})$, $c_{\text{joint}}(\dot{\mathbf{q}}_i^{\mathrm{R}})$, and $c_{\text{obs}}(\mathbf{q}_i^{\mathrm{R}})$.

Let $J$ denote the indices of the joints, $\theta_j^{\mathrm{R}}$ denote the angle of $j$th joint, and $(\theta_{\text{j, min}}^{\mathrm{R}}, \theta_{\text{j, max}}^{\mathrm{R}})$ denote the corresponding joint limitation,

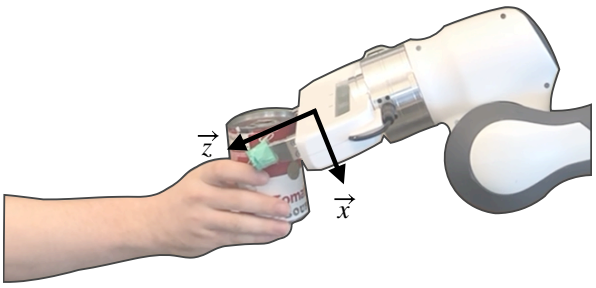



Fig. 2: The ideal orientation for the end effector. The $z$-axis points toward the human's wrist, which we are able to track with the Microsoft Azure Kinect, and the $x$-axis is as vertical as possible.

we employ a hinge-loss-based cost [35] for the joint limit:

$$c_{\text{joint}}(\mathbf{q}_i^{\mathrm{R}}) = \sum_{j \in J} \left\| c(\theta_j^{\mathrm{R}}) \right\|^2, \tag{7}$$

where $c(\theta_j^{\mathrm{R}})$ is defined as

$$c(\theta_j^{\mathrm{R}}) = \begin{cases} -\theta_j^{\mathrm{R}} + \theta_{\text{j, min}}^{\mathrm{R}} - \epsilon_j, & \text{if } \theta_j^{\mathrm{R}} < \theta_{\text{j, min}}^{\mathrm{R}} + \epsilon_j \\ \theta_j^{\mathrm{R}} - \theta_{\text{j, max}}^{\mathrm{R}} + \epsilon_j, & \text{if } \theta_j^{\mathrm{R}} > \theta_{\text{j, max}}^{\mathrm{R}} - \epsilon_j \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Here $\epsilon_j$ is the joint limit error tolerance for joint $j$.

We also impose a cost

$$c_{\text{joint}}(\dot{\mathbf{q}}_i^{\mathrm{R}}) = \sum_{j \in J} \left\| c(\dot{\theta}_j^{\mathrm{R}}) \right\|^2, \tag{9}$$

where $c(\dot{\theta}_i)$ is formulated similarly to Equation 8 using $\dot{\theta}$ in place of $\theta$.

These cost functions assume that the environment is static, *i.e.*, the camera and the obstacles are unchanging. Within the context of MPC, however, we are able to update these cost functions as the environment changes and we redefine our optimization problem. We compute a signed distance field representing a discretization of the environment. Then, as in [23], we use a sphere-based "skeleton" that covers the robot's entire volume and surface area. The spheres allow for a sparse and efficient representation of the robot's volume. Our total obstacle cost is then the sum of the cost at each sphere:

$$c_{\text{obs}}(\mathbf{q}_i^{\mathrm{R}}) = \sum_{s \in \text{spheres}} \|c(s)\|^2, \tag{10}$$

where

$$c(s) = \begin{cases} -d_s + s_{\text{radius}}, & \text{if } d_s \le s_{\text{radius}} \\ 0, & \text{if } d_s > s_{\text{radius}} \end{cases}. \tag{11}$$

Here $d_s$ is the value of the signed distance function at the sphere $s$'s center.

**Velocity and acceleration constraints.** We include independent constraints for configuration-space velocity and acceleration constraints, $c(\dot{\mathbf{q}}^{\mathrm{R}})$ and $c(\ddot{\mathbf{q}}^{\mathrm{R}})$, that each constrain these values to be zero. The velocity penalty ensures that the robot slows after after reaching its goal, while the acceleration penalty ensures the robot moves fluidly without overshooting its target.

**End effector constraints.** We also constrain the robot's end effector orientation to match an optimal value and add this to our overall cost function. In the 2D case, this optimal value is

straightforward: the robot should always be oriented towards the human. The optimal 3D orientation is more complex.

Let $G$ denote the coordinate frame of the gripper where the z-axis $\mathbf{z}_G$ points directly out from the gripper and the x-axis $\mathbf{x}_G$ points down perpendicular to the gripper. When the gripper is perfectly flat, $\mathbf{x}_G$ points straight down to the ground. We wish to align $\mathbf{z}_G$ with $\mathbf{v}$, the ray from the end effector to the human's hand. We also want the end effector to be approximately flat. We show this ideal configuration in Figure 2. Assuming the world frame has $\mathbf{z}_W$ up, we want to find $\mathbf{x}_G$ that when expressed in the world coordinates, has the lowest $\mathbf{z}$ coordinate, *i.e.*, in the world frame,

$$0 = \mathbf{z}_G \cdot \mathbf{x}_G = \frac{\mathbf{v}}{\|\mathbf{v}\|} \cdot [x_{\mathbf{x}_G}, y_{\mathbf{x}_G}, z_{\mathbf{x}_G}], \tag{12}$$

where $[x_{\mathbf{x}_G},\ y_{\mathbf{x}_G},\ z_{\mathbf{x}_G}]$ correspond to the world frame coordinates of $[1,\ 0,\ 0]_G$ in the gripper frame. We also know $x_{\mathbf{x}_G} = 1 - \sqrt{y^2_{\mathbf{x}_G}, z^2_{\mathbf{x}_G}}$.

If the gripper does not point straight up, we can first solve for $z_{\mathbf{x}_G}$, then take the derivative with respect to $y_{\mathbf{x}_G}$ and set it to zero in order to find the $\mathbf{x}_G$ that points most-down. Then, $\mathbf{y}_G = \mathbf{z}_G \times \mathbf{x}_G$ and we can use these three axes to construct our desired rotation matrix $\hat{\mathbf{R}}$.

With $\hat{\mathbf{R}}$, we use the same cost function from [36] to constrain the robot to face this direction.

$$c(\mathbf{R}^{\mathrm{R}}_i) = \log(\hat{\mathbf{R}}^{-1}\mathbf{R}^{\mathrm{R}}_i)^{\vee}$$

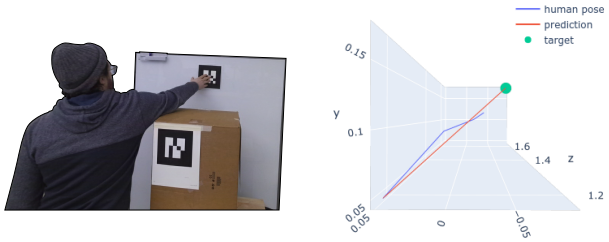
where $\log(\cdot)$ is the logarithmic map and $\vee$ is the operator that takes a skew-symmetric matrix to a vector.

As a simplifying assumption to improve planning efficiency, we only compute $\hat{\mathbf{R}}$ once at each planning cycle using current observation of both the robot and agent. Since we run many iterations closed-loop, the robot will continue to face towards the human's position.

### B. Modeling the human

We use a reduced, but similar, set of cost functions for the external agent, *i.e.*, the human, $c^{\mathrm{A}}(\boldsymbol{\xi}^{\mathrm{A}}) = \sum_{i=0}^{T} c^{\mathrm{A}}(\mathbf{q}^{\mathrm{A}}_i, \dot{\mathbf{q}}^{\mathrm{A}}_i, \ddot{\mathbf{q}}^{\mathrm{A}}_i)$. We model the human hand as a floating sphere and the parameters $\lambda_{\mathrm{A}}$ are determined through a set of 29 recorded reach-to-point tasks. In the task of handover where the robot and human are similar heights, we propose that a floating sphere is a sufficient model for the human. For other collaborative tasks, such as handover at significantly different heights, the human's morphology and the kinematic feasibility of the task would be important.

Our model for optimal human reaching is a parametrized quadratic cost function that is nearly symmetrical to the robot. We model the human as a sphere representing their hand location, so we omit the constraint of joint limits. We also assume that a cooperative human will rotate their hand to meet the robot comfortably, and so we omitted the rotational constraint as well. To evaluate our model, we recorded a set of 29 reach-to-point tasks around an obstacle using the Microsoft Azure Kinect DK and its included body tracking SDK. Between trials, we randomly changed the target position, the human's starting pose, and the camera height. See Figure 3 for an example of our setup.




(a) (b)

Fig. 3: (a) A reach-to-point task around an obstacle. We recorded 29 trials of a reach-to-point task, with varying target points, camera poses, and starting positions. (b) The predicted and measured human's trajectory for one trial. Here, the person chose to take a wider path around the obstacle than necessary.

For each trial, we calculated our prediction error with

$$\text{Loss} = \sum_{t=0}^{T} \frac{1}{T-(t+1)} \sum_{i=t+1}^{T} \|\mathbf{t}_{\text{predicted}} - \mathbf{t}_{\text{measured}}\|^2 \tag{13}$$

where $T$ is the total number of time steps it takes the human to reach the target and $x$ is the position of the hand. This loss represents the average distance between the corresponding real and predicted hand poses, which we then average over all MPC steps. We used 26 of our trials to tune our human model and performed a grid search over 6,561 parameter configurations to minimize error. We then evaluated the parameters on the remaining 3 datasets. Our average loss on the training set 7.54cm and our average loss on the evaluation set is 9.63cm and a standard deviation of 2.41cm.

### C. Modeling the robot-agent collaboration

At the end of the trajectory, the robot and the uncontrolled agent should meet. To enforce this, we encourage their end effector positions to be as close to each other as possible,

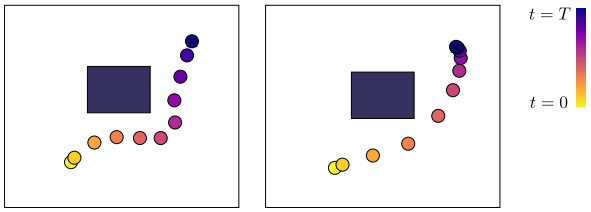
$$c^I(\boldsymbol{\xi}^{\mathrm{R}}, \boldsymbol{\xi}^{\mathrm{A}}) = \left\|\mathbf{t}^{\mathrm{R}}_T - \mathbf{t}^{\mathrm{A}}_T\right\|^2 \tag{14}$$

where $\mathbf{t}^{\mathrm{R}}_T$ denotes the position of the robot's end-effector at the final time step and $\mathbf{t}^{\mathrm{A}}_T$ denotes the position of the human's hand at that final time step (see the notation around forward kinematics in Section IV-A).

The interaction term defines interaction only at the end of the trajectory (the behavior is finished once the interaction occurs). In general, it is unclear when (time-wise) this interaction should occur, so choosing a single $T$ is challenging, even more-so when re-optimizing the system and rejecting system perturbations within an MPC loop. The next section designs a sparse reward motivated by reinforcement learning settings to eliminate this problem, enabling spatially consistent behavior using a time-parameterized trajectory model.

### D. Time Independence through Sparse Rewards

When two agents collaborate without explicit time synchronization, their interaction and behavior is often a function of combined state and not tied to a specific clock. For example, when handing over an object, both participants time their behaviors based on the observed state of the other, continually readjusting and aiming primarily to just meet in the middle. The behavior is state-dependent and not explicitly timed.




(a) Without reward term (b) With reward term

Fig. 4: Comparison motion generation for a reach-to-point task around an obstacle both with and without the proposed sparse reward term. The dots represent subsequent positions. With each MPC step, the agent starts closer to its goal. Our reward terms encourages the agent to speed up as a function of their relative distance to the goal arrive at the goal in less time than the planning horizon.

We account for deviations from the planner's output by continually re-optimizing with a fixed-time horizon at each successive time step. However, as the two agents approach each other, this fixed horizon becomes restrictive. Suppose the horizon is three seconds in the future. Placing the interaction term perpetually at the fixed time horizon means that the model will always want to interact exactly three seconds in the future, independent of where it finds itself, leading to an exponential slowdown in its behavior.

We counteract the slowdown by adding an additional distance-based reward term weighted by $\lambda_{\text{reward}}$ to the robot-agent collaboration cost defined in Eq. 14 at every point on the trajectory.

$$c(\boldsymbol{\xi}^{\text{R}},\boldsymbol{\xi}^{\text{A}}) = c(\mathbf{t}_T^{\text{R}},\mathbf{t}_T^{\text{A}}) + \lambda_{\text{reward}} \sum_{i=0}^{T} r(\mathbf{t}_i^{\text{R}},\mathbf{t}_i^{\text{A}}), \tag{15}$$

where the reward $r(\mathbf{t}_i^{\text{R}},\mathbf{t}_i^{\text{A}})$ at step $i$ is defined as

$$r(\mathbf{t}_i^{\text{R}},\mathbf{t}_i^{\text{A}}) = 1 - e^{\frac{-\left\|\mathbf{t}_i^{\text{R}}-\mathbf{t}_i^{\text{A}}\right\|^2}{2\sigma^2}}. \tag{16}$$

This reward term can also apply to a single agent moving toward a fixed target, where $\left\|\mathbf{t}_i^{\text{Agent}} - \text{p}^{\text{Target}}\right\|$ would replace $\left\|\mathbf{t}_i^{\text{R}} - \mathbf{t}_i^{\text{A}}\right\|$.

The reward is motivated by the types of sparse rewards used in reinforcement learning [37]. We are rewarding the agents for converging, and, since our goal is to minimize cost, we phrase reward as negative cost. Such a reward can be modeled as an upside down radial basis function over the distance between the robot's end-effector and the interacting agent, *i.e.*, one when the two are far apart and decreasing to zero as they draw closer.

This rewards the robot for getting within touching distance of the interacting agent (and visa-versa), but does not penalize the pair for having to be far from each other earlier in the trajectory due to competing smoothness criteria. Since we have a fixed finite-horizon, without loss of generalization, we can shift each of these reward terms up by a constant so its minimum value is zero as given in the equation. The effect can be seen in Fig. 4, which shows how this results in a more temporally-consistent trajectory. Specifically, in the absence of perturbations, the trajectory traced out by MPC's replan-execute loop is more consistent with the original trajectory initially planned at the first time step.

For safety reasons, the uncontrolled agent may stop before reaching the robot. The reward term as formulated in Eq. 16 would cause the robot to slow-down exponentially because the human policy predicts that the human will keep moving. To prevent this, we would ideally have a phase estimator that can determine when the human has stopped and switch to rewarding the robot for reaching the human's current position. As an approximation in our implementation, the human and robot are both rewarded for reaching each others' starting points, as determined at the beginning of each MPC step.

With this shift upward, our formulation of the reward (as cost) becomes identical to the Welsch robust estimator [38]. Although the reward is not a nonlinear least squares term, it can be minimized using a form of iteratively re-weighted least-squares [39] using weights given by the Radial Basis Function (RBF) $w(r) = e^{\frac{-r^2}{2\sigma^2}}$ where $r = \|\mathbf{t}_i^{\text{R}} - \mathbf{t}_i^{\text{A}}\|$ in this case. An implementation would replace these Welsch robust estimator objective terms with weighted least squares terms of the form $wr^2 = w\|\mathbf{t}_i^{\text{R}} - \mathbf{t}_i^{\text{A}}\|^2$, and re-evaluating the weight $w$ after each subproblem has converged.

These reward terms reward the system for reaching the interaction point early. As above, associated with reaching the interaction point should be a velocity penalty bringing the system to a stop. Whereas before, it sufficed to add just a single velocity penalty to the terminal potential (stop at the end) we now must also add intermediate velocity penalties preparing for the possibility of stopping early.

## V. Implementation Details

Similar to [14], we used GTSAM as a fast optimizer to minimize the cost at each MPC-step. We used the real-time body tracking SDK on the Microsoft Azure Kinect DK to obtain human pose estimates, and we are able to run our algorithm to perform a human handover in real time on a Franka Emika Panda arm.

On a workstation with an *3.4ghz* Intel© Core™ *i7* and *32GB* of RAM running Ubuntu 18.04, we obtained poses at a rate of 30hz. We use DART [40] to calibrate the robot configuration into the Azure frame, so we can obtain both human and robot starting positions at each MPC step.

We run both our optimizer and DART on the same workstation running Ubuntu 16.04 and equipped with an *3.7ghz* Intel© Core™ and *32GB* of RAM. We obtain DART's positional estimates at $10hz$, and we are able to run our optimizer with Levenberg-Marquardt between 7hz and 8hz.

When the Franka is within a minimum threshold–we used 10cm–it engages the gripper and tries to grasp the object. If it misses and closes all the way, the gripper re-opens and the planner resumes trying to engage in the handover until it succeeds. We found the robot to miss the handover when the human moves too quickly for the body tracker to maintain a stable estimate.

## VI. Experiments

We ran a set of experiments exploring: (1) How do our method's generated trajectories compare to those produced by baseline methods? (2) How robust is our algorithm to noisy sensors? and (3) Can our proposed method be used in a real world setting?

Our algorithm predicts the motion and dynamics of the uncontrolled agent and reacts accordingly. In order to evaluate each component, we benchmarked our algorithm against two different baselines:

**Robot only**: A planner that only accounts for the Euclidean position of the uncontrolled agent. At each time step, the robot

TABLE I: Algorithmic benchmarks (↑ denotes higher is better and ↓ denotes lower is better): our algorithm is best able to approximate the timing of the uncontrolled agent. The attractor-based algorithm produces trajectories with significantly greater acceleration and jerk than both the robot-only and our algorithm. The robot-only algorithm outperforms ours by a small margin in reducing acceleration and jerk, but at the cost of producing much longer trajectories.

| Metric | | Robot only | Attractor | Ours |
|---|---|---|---|---|
| Handover Time (Normalized) | ↓ | $1.33 \pm 0.27$ | $1.30 \pm 0.26$ | $\mathbf{1.20 \pm 0.26}$ |
| Trajectory Length Error | ↓ | $0.35 \pm 0.27$ | $0.37 \pm 0.29$ | $\mathbf{0.27 \pm 0.22}$ |
| Acceleration ($cm/s^2$) | ↓ | $\mathbf{4.33 \pm 1.96}$ | $7.83 \pm 3.96$ | $4.72 \pm 1.37$ |
| Jerk $\mu$ ($cm/s^3$) | ↓ | $\mathbf{6.25 \pm 2.99}$ | $10.06 \pm 5.55$ | $6.36 \pm 1.88$ |

optimizes a trajectory around any obstacles to match its end effector position with the other agent's end effector position.

**Attractor**: A planner that applies an attractor-based policy to both end effectors. This policy assumes the two arms will move toward each other at each time step. When obstacles are present, they act as repellent forces, opposing the attracting force. We implemented this method by using our same algorithm with a very short time horizon, *i.e.*, $T = 5$, which is the shortest trajectory supported by our low-level controller.

See our video submission for an example of the simulated environment. The robot and uncontrolled agent start on opposite sides of a non-convex obstacle, the position and shape of which we randomized for each trial. To evaluate the algorithms without bias, we independently planned the uncontrolled trajectory to go from a randomized location on the opposing side of the obstacle to a randomized point in the robot's reachable space, while also avoiding the obstacles. We accomplish this by minimizing our same velocity, obstacle-avoidance, and acceleration costs for the hand, while also adding a cost term with high $\lambda$ to constrain the hand to our randomly chosen start and end positions, as in [14]. We augmented the plans with noise drawn from a uniform distribution to de-bias the uncontrolled motion from the planner.

We then replayed the uncontrolled trajectory for each robot policy. At the end of the uncontrolled trajectory, the agent pauses and waits for the robot. We modeled the agent as a ball with a 10cm radius and as such, we considered a trial to be successful if the robot was able to plan a trajectory where its end effector was within 10cm of the agent in under twice the uncontrolled trajectory length.

We ran 300 trials. Qualitatively, we observed that the randomized obstacle and randomized uncontrolled trajectory led to many ill-formed handovers, such as ones where the uncontrolled trajectory goes through the obstacles. The robot-only algorithm best handled these ill-formed trajectories because it is able to ignore whether the human is in an incorrect configuration. It succeeded in $62\%$ of the trials. Our algorithm succeeded in $57\%$ of the trials and the attractor policy succeeded in $43\%$ of the trials. It is important to note that our framework assumes that uncontrolled agents are co-operative and reactive. To fairly compare the three policies, we used identical trajectories for the uncontrolled agent, but this precluded the possibility of the uncontrolled agent's policy being reactive to the robot and therefore violates our cooperative assumption. With a cooperative partner, we expect our policy to outperform the robot-only policy in success rate.

We evaluated the algorithms on the set of trials on which they mutually succeeded and adopted four different metrics, *i.e.*, *Success rate, trajectory length error, acceleration*, and *jerk*, for evaluation. We define the *trajectory length error* as $|1 - T_{\text{success}}/T_{\text{uncontrolled}}|$ where $T_{\text{success}}$ is the time it takes to finish a successful action and $T_{\text{uncontrolled}}$ is the length of the uncontrolled trajectory. Quantitative results are in Table I.

TABLE II: Robustness metrics: we evaluate our robustness to measurement noise by determining, for a given amount of measurement noise, the percentage of handovers that can be completed within twice the time of the uncontrolled trajectory

| Noise $\sigma$ (cm) | 2 | 5 | 7 | 10 | 15 |
|---|---|---|---|---|---|
| % Successful | 100 | 100 | 98 | 86 | 66 |

Both qualitatively and quantitatively, we saw that the *attractor* algorithm leads the robot to jerk heavily when the agent's path around the obstacle is non-obvious. Meanwhile, the *robot-only* planner tends to wait to move until the path around the obstacle is unobstructed, leading it to take longer to reach the agent. Our algorithm is able to smoothly predict the agents path. We also saw our algorithm produces lower *trajectory length error* than the others–meaning our algorithm is better able to match the length of the uncontrolled trajectory. See our video for a demonstration.

We also evaluated our algorithm's robustness to measurement noise. In our real-robot experiments, we observed that our planner failed when the calibration and/or body tracker were misaligned. To measure this, we planned a randomized uncontrolled trajectory in the same fashion as before. However, we also introduced Gaussian noise with increasing $\sigma$ into the robot's perception of the trajectory. We ran $50$ trials and measured how often the robot could intersect the agent at its *actual* location within twice the uncontrolled trajectory length. Results with varying $\sigma$ are in Table II.

As shown in Figure 1, we are able to run our algorithm on a real robot using the setup described in Section V. See our video for more examples.

## VII. Conclusion and Future Work

We proposed an MPC approach for multi-agent collaboration problems that simultaneously optimizes motion plans for a robot and an (uncontrolled) human in order to enable coordination on cooperative tasks, with an application to human-robot handovers in obstacle-rich environments. We presented a novel theoretical framework and demonstrated its effectiveness through both simulated and real-robot experiments. This framework assumes access to a model for the human collaborator, and future work might learn such a model from data, for example, via Inverse Optimal Control [41], which has been successfully applied to motion prediction in the past [42]. In addition, our approach cannot generate longer-term plans across manipulations, as in [6]; in the future, we will develop approaches which maintain reactivity but allow for switching between discrete modes, such as when the human is waiting for the robot at the end of a handover, possibly via the Robust Logical-Dynamical Systems formalism [43]. We also intend to apply our formal system to other coordination problems explored in the literature, such as motion in a crowd [3] and camera control [44], [45].

## References

[1] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. D. Bagnell, M. Hebert, A. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *IROS*, 2009.

[2] J. Mainprice, R. Hayne, and D. Berenson, "Goal set inverse optimal control and iterative re-planning for predicting human reaching motions in shared workspaces," *IEEE Transaction on Robotics (TRO)*, 2016.

[3] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 454–460.

[4] A. D. Dragan, K. C. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2013, pp. 301–308.

[5] H. S. Koppula, A. Jain, and A. Saxena, "Anticipatory planning for human-robot teams," in *Experimental robotics*. Springer, 2016, pp. 453–470.

[6] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4044–4051.

[7] L. Peternel, W. Kim, J. Babič, and A. Ajoudani, "Towards ergonomic control of human-robot co-manipulation and handover," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 55–60.

[8] D. Vogt, S. Stepputtis, B. Jung, and H. B. Amor, "One-shot learning of human–robot handovers with triadic interaction meshes," *Autonomous Robots*, vol. 42, no. 5, pp. 1053–1065, 2018.

[9] K. Strabala, M. K. Lee, A. D. Dragan, J. Forlizzi, S. S. Srinivasa, M. Cakmak, and V. Micelli, "Toward seamless human-robot handovers," in *HRI 2013*, 2013.

[10] C.-M. Huang, M. Cakmak, and B. Mutlu, "Adaptive coordination strategies for human-robot handovers," in *Robotics: Science and Systems*, 2015.

[11] G. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters, "Probabilistic movement primitives for coordination of multiple humanrobot collaborative tasks," *Autonomous Robots*, vol. 41, pp. 593–612, 2017.

[12] J. R. Medina, F. Duvallet, M. Karnam, and A. Billard, "A human-inspired controller for fluid human-robot handovers," *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 324–331, 2016.

[13] K. Murphy, "A survey of pomdp solution techniques," *Environment*, vol. 2, 10 2000.

[14] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *The International Journal of Robotics Research*, vol. 37, pp. 1319 – 1340, 2018.

[15] N. Ratliff, M. Toussaint, and S. Schaal, "Understanding the geometry of workspace obstacles in motion optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

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

[17] T. Zhou and J. P. Wachs, "Early prediction for physical human robot collaboration in the operating room," *Autonomous Robots*, vol. 42, no. 5, pp. 977–995, 2018.

[18] A. Kupcsik, D. Hsu, and W. S. Lee, "Learning dynamic robot-to-human object handover from human feedback," in *Robotics research*. Springer, 2018, pp. 161–176.

[19] E. C. Grigore, K. Eder, A. G. Pipe, C. Melhuish, and U. Leonards, "Joint action understanding improves robot-to-human object handover," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4622–4629.

[20] T. Stouraitis, I. Chatzinikolaidis, M. Gienger, and S. Vijayakumar, "Dyadic collaborative manipulation through hybrid trajectory optimization." in *Conference on Robot Learning (CoRL)*, 2018, pp. 869–878.

[21] V. V. Unhelkar, H. C. Siu, and J. A. Shah, "Comparative performance of human and mobile robotic assistants in collaborative fetch-and-deliver tasks," in *HRI*. IEEE, 2014, pp. 82–89.

[22] K. Yamane, M. Revfi, and T. Asfour, "Synthesizing object receiving motions of humanoid robots with human motion database," *2013 IEEE International Conference on Robotics and Automation*, pp. 1629–1636, 2013.

[23] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," 2009.

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

[25] J. P. Chonhyon Park and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

[26] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," 06 2013.

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

[28] ——, "Newton methods for k-order markov constrained motion problems," *ArXiv*, vol. abs/1407.0414, 2014.

[29] C. R. Garrett, C. Paxton, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox, "Online replanning in belief space for partially observable task and motion problems," *arXiv preprint arXiv:1911.04577*, 2019.

[30] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in neural information processing systems*, 2010, pp. 2164–2172.

[31] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Advances in neural information processing systems*, 2013, pp. 1772–1780.

[32] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 475–491.

[33] E. Tarados and V. V. Vazirani, *Algorithmic Game Theory, chapter Basic Solution Concepts and Computational Issues*. Cambridge University Press, 2007.

[34] F. Dellaert and M. Kaess, "Factor Graphs for Robot Perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[35] M. Mukadam, X. Yan, and B. Boots, "Gaussian Process Motion planning," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June. Institute of Electrical and Electronics Engineers Inc., 6 2016, pp. 9–15.

[36] J. Dong, B. Boots, and F. Dellaert, "Sparse Gaussian processes for continuous-time trajectory estimation on matrix Lie groups," *Arxiv*, vol. abs/1705.06020, 2017. [Online]. Available: http://arxiv.org/abs/1705.06020

[37] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. van de Wiele, V. Mnih, N. Heess, and J. T. Springenberg, "Learning by playing solving sparse reward tasks from scratch," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmssan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 4344–4353. [Online]. Available: http://proceedings.mlr.press/v80/riedmiller18a.html

[38] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics - Theory and Methods*, vol. 6, no. 9, pp. 813–827, 1977. [Online]. Available: https://doi.org/10.1080/03610927708827533

[39] Z. Zhang, "Parameter estimation techniques: a tutorial with application to conic fitting," *Image Vision Comput.*, vol. 15, pp. 59–76, 1997.

[40] T. Schmidt, R. A. Newcombe, and D. Fox, "DART: Dense articulated real-time tracking," in *Robotics: Science and Systems*, 2014.

[41] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Human behavior modeling with maximum entropy inverse optimal control," in *AAAI Spring Symposium on Human Behavior Modeling*, 2009.

[42] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *European Conference on Computer Vision*. Springer, 2012, pp. 201–214.

[43] C. Paxton, N. Ratliff, C. Eppner, and D. Fox, "Representing robot task plans as robust logical-dynamical systems," *IEEE Conference on Intelligent Robots and Systems (IROS)*, 2019.

[44] D. Rakita, B. Mutlu, and M. Gleicher, "An autonomous dynamic camera method for effective remote teleoperation," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2018, pp. 325–333.

[45] R. Bonatti, C. Ho, W. Wang, S. Choudhury, and S. A. Scherer, "Towards a robust aerial cinematography platform: Localizing and tracking moving targets in unstructured environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.