Journal Title
XX(X):1–15
©The Author(s) 2018
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

I-Planner: Intention-Aware Motion Planning Using Learning Based Human Motion Prediction

Jae Sung Park[1], Chonhyon Park[2] and Dinesh Manocha[3]

http://gamma.cs.unc.edu/SafeMP (video included)

Abstract

We present a motion planning algorithm to compute collision-free and smooth trajectories for high-DOF robots interacting with humans in a shared workspace. Our approach uses offline learning of human actions along with temporal coherence to predict the human actions. Our intention-aware online planning algorithm uses the learned database to compute a reliable trajectory based on the predicted actions. We represent the predicted human motion using a Gaussian distribution and compute tight upper bounds on collision probabilities for safe motion planning. We also describe novel techniques to account for noise in human motion prediction. We highlight the performance of our planning algorithm in complex simulated scenarios and real-world benchmarks with 7-DOF robot arms operating in a workspace with a human performing complex tasks. We demonstrate the benefits of our intention-aware planner in terms of computing safe trajectories in such uncertain environments.

Keywords

Robot motion planning, human motion prediction

Introduction

Motion planning algorithms are used to compute collision-free paths for robots among obstacles. As robots are increasingly used in workspace with moving or unknown obstacles, it is important to develop reliable planning algorithms that can handle environmental uncertainty and the dynamic motions. In particular, we address the problem of planning safe and reliable motions for a robot that is working in environments with humans. As the humans move, it is important for the robots to predict the human actions and motions from sensor data and to compute appropriate trajectories.

In order to compute reliable motion trajectories in such shared environments, it is important to gather the state of the humans as well as predict their motions. There is considerable work on online tracking and prediction of human motion in computer vision and related areas Shotton et al. (2013). However, the current state of the art in gathering motion data results in many challenges. First of all, there are errors in the data due to the sensors (e.g., point cloud sensors) or poor sampling Choo et al. (2014). Secondly, human motion can be sudden or abrupt and this can result in various uncertainties in terms of accurate representation of the environment. One way to overcome some of these problems is to use predictive or estimation techniques for human motion or actions, such as using various filters like Kalman filters or particle filters Vasquez et al. (2009). Most of these prediction algorithms use a motion model that can predict future motion based on the prior positions of human body parts or joints, and corrects the error between the estimates and actual measurements. In practice, these approaches only work well when there is sufficient information about prior motion that can be accurately modeled by the underlying motion model. In some scenarios, it is possible to infer high-level human intent using additional information, and thereby perform a better prediction of future human motions Bandyopadhyay et al. (2013); Bera et al. (2016). These techniques are used to predict the pedestrian trajectories based on environmental information in 2D domains.

Main Results: We present a novel high-DOF motion planning approach to compute collision-free trajectories for robots operating in a workspace with human-obstacles or human-robot cooperating scenarios (I-Planner). Our approach is general, and doesn't make much assumptions about the environment or the human actions. We track the positions of the human using depth cameras and present a new method for human action prediction using a combination of classification (to predict the type of human motion) and regression (to predict the actual future human motion) methods. Given the sensor noises and prediction errors, our online motion planner uses probabilistic collision checking to compute a high dimensional robot trajectory that tends to compute safe motion in the presence of uncertain human motion. In contract to prior methods, the main benefits of our approach include:

[1]Department of Computer Science, UNC Chapel Hill, NC, USA
[2]Zoox Inc., Menlo Park, CA, USA
[3]Department of Computer Science and Electrical & Computer Engineering, University of Maryland at College Park, MD, USA

Corresponding author:
Jae Sung Park, Department of Computer Science, UNC Chapel Hill, NC, USA.
Email: jaesungp@cs.unc.edu

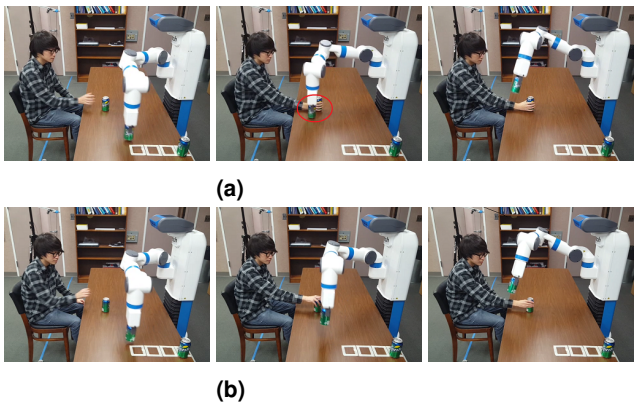


Figure 1. A 7-DOF Fetch robot is moving its arm near a human, avoiding collisions. (a) While the robot is moving, the human tries to move his arm to block the robot's path. The robot arm trajectory is planned without human motion prediction, which may result in collisions and a jerky trajectory, as shown with the red circle. This is because the robot cannot respond to the human motion to avoid collisions. (b) The trajectory is computed using our human motion prediction algorithm; it avoids collisions and results in smoother trajectories. The robot trajectory computation uses collision probabilities to anticipate the motion and compute safe trajectories.

1. A novel data-driven algorithm for intention and motion prediction, given noisy point cloud data. Compared to prior methods, our formulation can account for big noise in skeleton tracking in terms of human motion prediction.
2. An online high-DOF robot motion planner for efficient completion of collaborative human-robot tasks that uses upper bounds on collision probabilities to compute safe trajectories in challenging 3D workspaces. Furthermore, our trajectory optimization based on probabilistic collision checking results in smoother paths.

We highlight the performance of our algorithms in a simulator with a 7-DOF KUKA arm operating and in a real world setting with a 7-DOF Fetch robot arm in a workspace with a moving human performing cooperative tasks. We have evaluated its performance in some challenging or cluttered 3D environments where the human is close to the robot and moving at varying speeds. We demonstrate the benefits of our intention-aware planner in terms of computing safe trajectories in these scenarios. A preliminary version of this paper was published Park et al. (2017c). As compared to Park et al. (2017c), we improve the human motion prediction algorithm using depth sensor data. We present a mathematical analysis of the robustness of our prediction algorithm and highlight its benefits and improved accuracy for challenging scenarios. We also analyze the performance of our algorithm with varying human motion speeds.

Related work

In this section, we give a brief overview of prior work on human motion prediction, task planning for human-robot collaborations, and motion planning in environments shared with humans.

Intention-aware motion planning and prediction

Intention-Aware Motion Planning (IAMP) denotes a motion planning framework where the uncertainty of human intention is taken into account Bandyopadhyay et al. (2013). The goal position and the trajectory of moving pedestrians can be considered as human intention and used so that a moving robot can avoid pedestrians Unhelkar et al. (2015).

In terms of robot navigation among obstacles and pedestrians, accurate predictions of humans or other robot positions are possible based on crowd motion models Fulgenzi et al. (2007); van den Berg et al. (2008) or integration of motion models with online learning techniques Kim et al. (2014) for 2D scenarios and they are orthogonal to our approach.

Predicting the human actions or the high-DOF human motions has several challenges. Estimated human poses from recorded videos or realtime sensor data tend to be inaccurate or imperfect due to occlusions or limited sensor ranges Choo et al. (2014). Furthermore, the whole-body motions and their complex dynamics with many high-DOF makes it difficult to represent them with accurate motion models Hofmann and Gavrila (2012). There has been a considerable literature on recognizing human actions Turaga et al. (2008). Machine learning-based algorithms using Gaussian Process Latent Variable Models (GP-LVM) Ek et al. (2007); Urtasun et al. (2006) or Recurrent neural network (RNN) Fragkiadaki et al. (2015) have been proposed to compute accurate human dynamics models. Recent approaches use the learning of human intentions along with additional information, such as temporal relations between the actions Nikolaidis et al. (2013); Hawkins et al. (2013) or object affordances Koppula and Saxena (2016) to improve the accuracy. Inverse Reinforcement Learning (IRL) has been used to predict 2D motions Ziebart et al. (2008); Kuderer et al. (2012) or 3D human motions Dragan and Srinivasa (2013).

Robot task planning for human-robot collaboration

In human-robot collaborative scenarios, robot task planning algorithms have been developed for the efficient distribution of subtasks. One of their main goal is reducing the completion time of the overall task by interleaving subtasks of robot with subtasks of humans with well designed task plans. In order to compute the best action policy for a robot, Markov Decision Processes (MDP) have been widely used Busoniu et al. (2008). Nikolaidis et al. (2013) use MDP models based on mental model convergence of human and robots. Koppula et al. (2016) use Q-learning to train MDP models where the graph model has the transitions corresponding to the human action and robot action pairs. Pérez-D'Arpino and Shah (2015) used a Bayesian learning algorithm on hand motion prediction and tested the algorithm in a human-robot collaboration tasks. Our MDP models extend these approaches, but also take into account the issue of avoiding collisions between the human and the robot.

Motion planning in environments shared with humans

Prior work on motion planning in the context of human-robot interaction has focused on computing robot motions that satisfy cognitive constraints such as social acceptability Sisbot et al. (2007) or being legible to humans Dragan et al. (2015).

In human-robot collaboration scenarios where both the human and the robot perform manipulation tasks in a shared environment, it is important to compute robot motions that avoid collisions with the humans for safety reasons. Dynamic window approach Fox et al. (1997) (which searches the optimal velocity in a short time interval) and online replanning Petti and Fraichard (2005); Park et al. (2012, 2013) (which interleaves planning with execution) are widely used approaches for planning in such dynamic environments. As there are uncertainties in the prediction model and in the sensors for human motion, the future pose is typically represented as the *Belief* state, which corresponds to the probability distribution over all possible human states. Mainprice and Berenson (2013) explicitly construct an occupied workspace voxel map from the predicted Belief states of humans in the shared environment and avoid collisions.

Partially Observable Markov Decision Process (POMDP) techniques are widely used for motion planning with uncertainty in the robot state and in the environment. These approaches estimate the robot environment states, represent them in a probabilistic manner, and tend to compute the best action or the best robot trajectory considering likely possibilities. Because the search space of the exact POMDP formulation is too large, many practical and approximate POMDP solutions have been proposed Kurniawati et al. (2011); Van Den Berg et al. (2012) to reduce the running time and obtain almost realtime performance. Bai et al. (2015) use an approximate and realtime POMDP motion planner on autonomous driving carts. Our algorithm solves the problem in two steps: first, the future human motion trajectory is predicted; second, our planning algorithm generates a collision-free trajectory by considering the predicted trajectory. Our current formulation does not fully account for the uncertainty in the robot state and can be combined with POMDP approaches to handle this issue.

Notation and assumptions

In this section, we first introduce the notation and terminology used in the paper and give an overview of our motion planning algorithms.

In the context of this paper, we use the term human 'intention' as a predetermined simple action to interact with the robotic environment. This is in the user's mind before the action is taken. The intention is a combination of what to do (e.g. reaching his/her hand to grab an object) and how to achieve it (e.g. moving his/her arm from an idle pose to the cup while avoiding the robot). More formally in the context of learning algorithms, the former is the action class in a discrete domain, the latter is a set of possible motions in a continuous domain, and 'intention' is a combination of both components.

I-Planner (Intention-Aware Motion Planner) is our robot motion planning algorithm that predicts human intentions and plans robot motions with the information of the predicted human intentions, given a whole task to be completed by the human and the robot working cooperatively. A task consists of multiple subtasks that the human or the robot can accomplish. The goal of I-Planner is to complete the given task efficiently, while guaranteeing safety for the human by avoiding any collisions. We specifically focus on three parts: predicting human intention in a short time window, determining a subtask for the robot to be achieved next, and planning a robot's motion trajectory that avoids the human for his/her safety. The predicted human action class (the first component of 'intention') is used in determining a subtask for the robot. The predicted human motion trajectory (the second component of 'intention') is used in planning the robot trajectory.

As we need to learn about human actions and short-term motions, a large training dataset of human motions is needed. We collect N demonstrations of how human joints typically move while performing some tasks and in which order subtasks are performed. Each demonstration is represented using $T^{(i)}$ time frames of human joint motion, where the superscript (i) represents the demonstration index. The motion training dataset is represented as following:

- ξ is a matrix of tracked human joint motions. $\xi^{(i)}$ has $T^{(i)}$ columns, where a column vector represents the different human joint positions during each time frame.
- F is a feature vector matrix. $F^{(i)}$ has $T^{(i)}$ columns and is computed from $\xi^{(i)}$.
- $\mathbf{a}^h$ is a human action (or subtask) sequence vector that represents the action labels over different time frames. For each time frame, the action is categorized into one of the m^h discrete action labels, where the action label set is $A^h = \{a^h_1, \cdots, a^h_{m^h}\}$.
- $L = \{(\xi^{(i)}, F^{(i)}, \mathbf{a}^{h,(i)})\}_{i=1}^{N}$ is the motion database used for training. It consists of human joint motions, feature descriptors and the action labels at each time frame.

During runtime, MDP-based human action inference is used in our task planner. The MDP model is defined as a tuple (P, A^r, T):

- $A^r = \{a^r_1, \cdots, a^r_{m^r}\}$ is a robot action (or subtask) set of m^r discrete action labels.
- $P = \mathcal{P}(A^r \cup A^h)$, a power set of union of A^r and A^h, is the set of states in MDP. We refer to the state $\mathbf{p} \in P$ as a *progress state* because each state represents which human and robot actions have been performed so far. We assume that the sequence of future actions for completing the entire task depends on the list of actions completed. $\mathbf{p}$ has $m^h + m^r$ binary elements, which represent corresponding human or robot actions have been completed ($\mathbf{p}_j = 1$) or not ($\mathbf{p}_j = 0$). For cases where same actions can be done more than once, the binary values can be replaced with integers, to count the number of actions performed.
- $T : P \times A^r \to \Pi(P)$ is a transition function. When a robot performs an action a^r in a state $\mathbf{p}$, $T(\mathbf{p}, a^r)$ is a

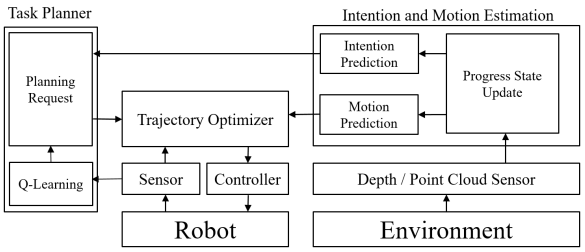
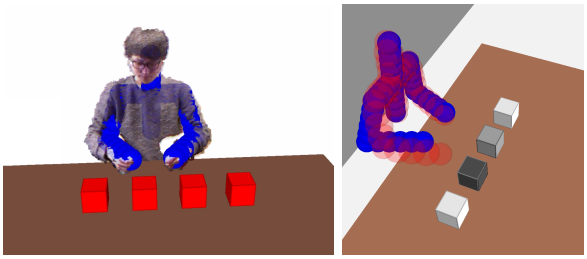


Figure 2. Overview of our Intention-Aware Planner: Our approach consists of three main components: task planner, trajectory optimization, and intention and motion estimation.



(a) **(b)**

Figure 3. Motion uncertainty and prediction: (a) A point cloud and the tracked human (blue spheres). The joint positions are used as feature vectors. (b) Prediction of next human action and future human motion, where 4 locations are colored according to their probability of next human action from white (0%) to black (100%). Prediction of future motion after 1 second (red spheres) from current motion (blue spheres) is shown as performing the action: *move right hand to the second position* which has the highest probability associated with it.

probability distribution over the progress state set P. The probability of being state $\mathbf{p}'$ after taking an action a^r from state $\mathbf{p}$ is denoted as $T(\mathbf{p}, a^r, \mathbf{p}')$.

- $\pi : P \rightarrow A^r$ is the action policy of a robot. $\pi(\mathbf{p})$ denotes the best robot action that can be taken at state $\mathbf{p}$, which results in maximal performance.

We use Q-learning Sutton and Barto (1998) to determine the best action policy during a given state, which rewards the values that are induced from the result of the execution.

For the robot representation, we denote a single configuration of the robot as a vector $\mathbf{q}$ that consists of joint-angles. The n-dimensional space of configuration $\mathbf{q}$ is the configuration space $\mathcal{C}$. We represent each link of the robot as R_i. The finite set of bounding spheres for link R_i is $\{B_{i1}, B_{i2}, \cdots\}$, and is used as a bounding volume of the link, i.e., $R_i \subset \cup_j B_{ij}$. The links and bounding spheres at a configuration $\mathbf{q}$ are denoted as $R_i(\mathbf{q})$ and $B_{ij}(\mathbf{q})$, respectively. In our benchmarks, where the robot arms are used, these bounding spheres are automatically generated using the medial axis of robot links. We also generate the bounding spheres $\{C_1, C_2, \cdots\}$ for humans and other obstacles.

For a planning task with start and goal configurations $\mathbf{q}_s$ and $\mathbf{q}_g$, the robot's trajectory is represented by a matrix $\mathbf{Q}$,

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q}_s & \mathbf{q}_1 & \cdots & \mathbf{q}_{n-1} & \mathbf{q}_g \\ t_0 & t_1 & \cdots & t_{n-1} & t_n \end{bmatrix},$$

where robot trajectory passes through the $n+1$ waypoints. We denote the i-th waypoint of $\mathbf{Q}$ as $\mathbf{x}_i = \begin{bmatrix} \mathbf{q}_i^T & t_i \end{bmatrix}$.

Human action prediction

In this section, we describe our human action prediction algorithm, which consists of offline learning and online inference of actions.

Learning of human actions and temporal coherence

We collect N demonstrations to form a motion database L. The 3D joint positions are tracked using OpenNI library Pri (2010), and their coordinates are concatenated to form a column vector $\xi^{(i)}$. For full-body motion prediction, we used 21 joints, each of which has 3D coordinates tracked by OpenNI. So, $\xi^{(i)}$ is a 63-dimensional vector. For upper-body motion prediction, there are 10 joints and thus $\xi^{(i)}$ is length 30. Then, feature vector $F^{(i)}$ is derived from $\xi^{(i)}$. It has joint velocities and joint accelerations, as well as joint positions.

To learn the temporal coherence between the actions, we deal with only the human action sequences $\{\mathbf{a}^{h,(i)}\}_{i=1}^{N}$. Based on the progress state representation, for any time frame s, the prefix sequence of $\mathbf{a}^{h,(i)}$ of length s yields a progress state $\mathbf{p}_s^{(i)}$ and the current action $c_s^{h,(i)} = \mathbf{a}_s^{h,(i)}$. The next action label $n_s^{h,(i)}$ performed after frame s can also be computed, at which the action label differs at the first time while searching in the increasing order from frame $s+1$. Then, for all possible pairs of demonstrations and frame index (i, s), the tuples $(\mathbf{p}_s^{(i)}, c_s^{h,(i)}, n_s^{h,(i)})$ are collected to compute histograms $h(n^h; \mathbf{p}, c^h)$, which counts the next action labels at each pair $(\mathbf{p}, c^h)$ that have appeared at least once. We use the normalized histograms to estimate the next future action for the given $\mathbf{p}$ and c^h. i.e.,

$$p(n^h = a_j^h | \mathbf{p}, c^h) = \frac{h(a_j^h; \mathbf{p}, c^h)}{\sum_{k=1}^{m} h(a_k^h; \mathbf{p}, c^h)}. \quad (1)$$

In the worst case, there are at most $O(2^m)$ progress states since there are m binary values per action. However, in practice, only $O(N \cdot m)$ progress states are generated. This is because the number of unique progress states is less than m, and the subtask order dependency may allow only a few possible topological orders.

In order to train the human motion, the motion sequence $\xi^{(i)}$ and the feature sequence $F^{(i)}$ are learned, as well as the action sequences $\mathbf{a}^{(i)}$. Because we are interested in short-term human motion prediction for collision avoidance, we train the learning module from multiple short periods of motion. Let n_p be the number of previous consecutive frames and n_f be the number of future consecutive frames to look up. n_f and n_p are decided so that the length of motion is short-term motion (about 1 second) that will be used for short-term future collision detection in the robot motion planner. At the time frame s, where $n_p \leq s \leq T^{(i)} - n_f$, the columns of feature matrix $F^{(i)}$ from column index $s - n_p + 1$ to s are denoted as $F_{prev,s}^{(i)}$. Similarly, the columns of motion matrix from index $s+1$ to $s+n_f$ are denoted as $\xi_{next,s}^{(i)}$.

Tuples $(F^{(i)}_{prev,s}, \mathbf{p}^{(i)}_s, c^{h(i)}_s, \xi^{(i)}_{next,s})$ for all possible pairs of (i,s) are collected as the training input. They are partitioned into groups having the same progress state $\mathbf{p}$. For each progress state $\mathbf{p}$ and current action c^h, the set of short-term motions are regressed using SPGP with the DTW kernel function Müller (2007), considering $\{F_{prev}\}$ as input and $\{\xi_{next}\}$ as multiple channeled outputs. We use SPGPs, a variant of Gaussian Processes, because it significantly reduces the running time for training and inference by choosing M pseudo-inputs from a large number of an original human motion inputs. The final learned probability distribution is

$$p(\xi_{next}|F_{prev}, \mathbf{p}, c^h) = \prod_{c:\text{channels}} p(\xi_{next,c}|F_{prev}, \mathbf{p}, c^h),$$
$$p(\xi_{next,c}|F_{prev}, \mathbf{p}, c^h) \sim \mathcal{GP}(m_c, K_c), \quad (2)$$

where $\mathcal{GP}(\cdot,\cdot)$ represents trained SPGPs, c is an output channel (i.e., an element of matrix ξ_{next}), and m_c and K_c are the learned mean and covariance functions of the output channel c, respectively.

The current action label $c^{h,(i)}_s$ should be learned to estimate the current action. We train $c^{h,(i)}_s$ using Tuples $(F^{(i)}_{prev,s}, \mathbf{p}^{(i)}_s, c^{h(i)}_s)$. For each state $\mathbf{p}$, we use Import Vector Machine (IVM) classifiers to compute the probability distribution:

$$p(c^h = a^h_j|F_{prev}, \mathbf{p}) = \frac{\exp(f_j(F_{prev}))}{\sum_{a^h_k}\exp(f_k(F_{prev}))}, \quad (3)$$

where $f_j(\cdot)$ is the learned predictive function Zhu and Hastie (2012) of IVM.

Runtime human intention and motion inference

Based on the learned human actions, at runtime we infer the next most likely short-term human motion and human subtask for the purpose of collision avoidance and task planning, respectively. The short-term future motion prediction is used for the collision avoidance during the motion planning. The probability of future motion is given as:

$$p(\xi_{next}|F, \mathbf{p}) = \sum_{c^h \in A^h} p(\xi_{next}, c^h|F, \mathbf{p}).$$

By applying the Bayes theorem, we get

$$p(\xi_{next}|F, \mathbf{p}) = \sum_{c^h \in A^h} p(c^h|F, \mathbf{p}) p(\xi_{next}|F, \mathbf{p}, c^h). \quad (4)$$

The first term $p(c^h|F, \mathbf{p})$ is inferred through the IVM classifier in (3). To infer the second term, we use the probability distribution in (2) for each output channel.

We use Q-learning for training the best robot action policy π in our MDP-based task planner. We first define the function $Q : P \times A^r \to \mathbb{R}$, which is iteratively trained with the motion planning executions. Q is updated as

$$Q_{t+1}(\mathbf{p}_t, a^r_t) = (1-\alpha_t)Q_t(\mathbf{p}_t, a^r_t) + \alpha_t(R_{t+1} + \gamma \max_{a^r} Q_t(\mathbf{p}_{t+1}, a^r)),$$

where the subscripts t and $t+1$ are the iteration indexes, R_{t+1} is the reward function after taking action a^r_t at state $\mathbf{p}_t$, and α_t is the learning rate, where we set $\alpha_t = 0.1$ in our experiments. A reward value R_{t+1} is determined by several factors:

- Preparation for next human action: the reward gets higher when the robot runs an action before a human action which can be benefited by the robot's action. We define this reward as $R_{prep}(\mathbf{p}_t, a^r_t)$. Because the next human subtask depends on the uncertain human decision, we predict the likelihood of the next subtask from the learned actions in (1) and use it for the reward computation. The reward value is given as

$$R_{prep}(\mathbf{p}_t, a^r_t) = \sum_{a^h \in A^h} p(n^h = a^h|\mathbf{p}_t) H(a^h, a^r_t), \quad (5)$$

  where $H(a^h, a^r)$ is a prior knowledge of reward, representing the amount of how much the robot helped the human by performing the robot action a^r before the human action a^h. If the robot action a^r has no relationship with a^h, the H value is zero. If the robot helped, the H value is positive, otherwise negative.
- Execution delay: There may be a delay in the robot motion's execution due to the collision avoidance with the human. To avoid collisions, the robot may deviate around the human and make it without delay. In this case the reward function is not affected, i.e. $R_{delay,t} = 0$. However, there are cases that the robot must wait until the human moves to another pose because the human can block the robot's path, which causes delay d. We penalize the amount of delay to the reward function, i.e. $R_{delay,t} = -d$. Note that the delay can vary during each iteration due to the human motion uncertainty.

The total reward value is a weighted sum of both factors:

$$R_{t+1} = w_{prep}R_{prep}(\mathbf{p}_t, a^r_t) + w_{delay}R_{delay,t}(\mathbf{p}_t, a^r_t),$$

where w_{prep} and w_{delay} are weights for scaling the two factors. The preparation reward value is predefined for each action pairs. The delay reward is measured during runtime.

Human motion prediction with noisy input

In most scenarios, our human motion prediction algorithm deals with the noisy data. As a result, it is important to analyze the performance of our approach in relation to these limitations. To analyze the robustness of our human motion prediction algorithm, we account for input noise in our Gaussian Process model.

Equation 2 is the Gaussian Process Regression for human motion prediction where the input variable is F_{prev} and the output variables are represented as$\xi_{next,c}$. Following the standard notation of the Gaussian Process, we use the symbol $\mathbf{x}$ as a D-dimensional input vector instead of F_{prev}, y as an output variable instead of $\xi_{next,c}$, and $y = f(\mathbf{x}) + \epsilon_y$ instead of $p(\xi_{next,c}|F_{prev})$ $\mathcal{GP}(m_c, K_c)$. We add an input noise term ϵ_x to the standard GP model,

$$y = \tilde{y} + \epsilon_y, \epsilon_y \sim \mathcal{N}(0, \sigma^2_y),$$
$$\mathbf{x} = \tilde{\mathbf{x}} + \epsilon_{\mathbf{x}}, \epsilon_x \sim \mathcal{N}(\mathbf{0}, \Sigma_x),$$

where we assume that the input noise is Gaussian and that the D-dimensional input vector is independently noised, i.e. Σ_x is diagonal. With the input noise term, the output becomes

$$y = f(\tilde{\mathbf{x}} + \epsilon_x) + \epsilon_y,$$

and the first term Taylor expansion on the function f yields

$$y = f(\tilde{\mathbf{x}}) + \epsilon_x^T \partial_f(\mathbf{x}) + \epsilon_y,$$

where $\partial_f(\mathbf{x})$ is the D-dimensional derivative of f with respect to $\mathbf{x}$. We have N training data items, represented as $(\mathbf{x}_i, y_i)_{i=1}^N$. $\mathbf{y}$ is an N-dimensional vector $\{y_1, \cdots, y_N\}^T$. If we follow the derivation of the Gaussian Process (GP) with the additional error term presented in McHutchon and Rasmussen (2011), GP with noisy input becomes

$$m_c(\mathbf{x}_*) = \mathbf{k}_* \left(K + \sigma_y^2 I + \text{diag}\left(\Delta_f \Sigma_x \Delta_f^T\right)\right)^{-1} \mathbf{y},$$
$$K_c(\mathbf{x}_*) = k_{**} - \mathbf{k}_* \left(K + \sigma_y^2 I + \text{diag}\left(\Delta_f \Sigma_x \Delta_f\right)\right)^{-1} \mathbf{k}_*,$$

where $\mathbf{x}_*$ is the input of the mean function m_c and of the variance function K_c; k_{**} is the kernel function value on $\mathbf{x}_*$, i.e. $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$; K is a matrix of kernel function values on all pairs of input points, i.e. $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$; and Δ_f is a matrix whose i-th row is the derivative of f at $\mathbf{x}_i$. $\text{diag}(\cdot)$ results in a diagonal matrix, leaving the diagonal elements and reducing the non-diagonals to zero. Compared to the standard GP, the additional term is the diagonal matrix $\text{diag}\left(\Delta_f \Sigma_x \Delta_f\right)$, which acts as the output noise term $\sigma_y I$.

In our human motion prediction algorithm, we use the human joint positions as input and output, so the input and the output have the same amount of noise. In other words, $\Sigma_x = \sigma_x I$ and $\sigma_x = \sigma_y$. Instead of differentiating between input and output noise, we use $\sigma = \sigma_x = \sigma_y$. Also, the derivative ∂_f is proportional to the joint velocities because f is proportional to the joint position values. Therefore, the elements of the diagonal matrix can be expressed as

$$\left(\text{diag}\left(\Delta_f \Sigma_x \Delta_f\right)\right)_{ii} = \sigma_x ||\partial_f(\mathbf{x}_i)||^2 = \sigma ||\mathbf{v}_i||^2,$$

where $\mathbf{v}_i$ is the joint velocity. Because the joint velocity is a variable for every training input data, instead of taking joint velocities, we set the joint velocity limits $\mathbf{v}$, satisfying $||\mathbf{v}_i||^2 \leq ||\mathbf{v}||^2$. As a result, the Gaussian Process regression for human motion prediction can be given as:

$$m_c(\mathbf{x}_*) = \mathbf{k}_* \left(K + \sigma^2 \left(1 + ||\mathbf{v}||^2\right) I\right)^{-1} \mathbf{y},$$
$$K_c(\mathbf{x}_*) = k_{**} - \mathbf{k}_* \left(K + \sigma_y^2 \left(1 + ||\mathbf{v}||^2\right) I\right)^{-1} \mathbf{k}_*.$$

If we keep the joint velocity small, the Gaussian Process with input noise behaves robustly, like how it behaves without input noise. The mean function is over-smoothed and the variance function becomes higher if the joint velocity limit is set high.

I-Planner: Intention-aware motion planning

The main goals of our motion planner are: (1) planning high-level tasks for a robot by anticipating the most likely next human action and (2) computing a robot trajectory that reduces the probability of collision between the robot and the human or other obstacles by using motion prediction.

At the high-level task planning step we use MDP, which is used to compute the best action policies for each state. The state of an MDP graph denotes the progress of the whole task. The best action policies are determined through reinforcement learning with Q-learning. Then, the best action policies are updated within the same state. The probability of choosing an action increases or decreases according to the reward function. Our reward computation function is affected by the prediction of intent and the delay caused by predictive collision avoidance.

We also estimate the short-term future motion from learned information to avoid future collisions. From the joint position information, motion features are extracted based on human poses and surrounding objects related to human-robot interaction tasks, such as human joint positions, positions of a hand relative to other objects, etc. The motions are classified over the human action set A^h. To classify the motions, we use an Import Vector Machine (IVM) Zhu and Hastie (2012) for classification and a Dynamic Time Warping (DTW) Müller (2007) kernel function for incorporating the temporal information. Given the human motions database of each action type, we train future motions using Sparse Pseudo-input Gaussian Process (SPGP) Snelson and Ghahramani (2005). Combining these two prediction results, the final future motion is computed as the weighted sum over different action types weighed by the probability of each action type that could be performed next. For example, if the action classification results in probability are 0.9 for action *Move forward* and 0.1 for action *Move backward*, the future motion prediction (the results of SPGP) for *Move forward* dominates. If the action classification results in a probability of 0.5 for both actions, the predicted future motions for each action class will be used in avoiding collisions, but with weights of 0.5. However, in this case, the current motion does not have specific features to distinguish the action, meaning that the future motion in the short term will be similar and there will be an overlapped region in the 3D space that works as the future motion of weight 1.

After deciding which robot task will be performed, a robot motion trajectory that tends to avoid collisions with humans is then computed. An optimization-based motion planner Park et al. (2012) is used to compute a locally optimal solution that minimizes the objective function subject to many types of constraints such as robot related constraints (e.g., kinematic constraint), human motion related constraints (e.g., collision free constraint), etc. Because future human motion is uncertain, we can only estimate the probability distribution of the possible future motions. Therefore, we perform probabilistic collision checking to reduce the collision probability in future motions. We also continuously track the human pose and update the predicted future motion to re-plan safe robot motions. Our approach uses the notion of online probabilistic collision detection Pan et al. (2011); Park et al. (2016, 2017b) between the robot and the point-cloud data corresponding to human obstacles to compute reactive costs and integrate them with our optimization-based planner.

Out motion planner is based on an optimization formulation, where $n+1$ waypoints in the space-time domain $\mathbf{Q}$ define a robot motion trajectory to be optimized. Specifically, we use an optimization-based

planner, ITOMP Park et al. (2012), that repeatedly refines the trajectory while interleaving the execution and motion planning for dynamic scenes. We handle three types of constraints: smoothness constraint, static obstacle collision-avoidance, and dynamic obstacle collision avoidance. To deal with the uncertainty of future human motion, we use probabilistic collision detection between the robot and the predicted future human pose.

Let s be the current waypoint index, meaning that the motion trajectory is executed in the time interval $[t_0, t_s]$, and let m be the replanning time step. A cost function for collisions between the human and the robot can be given as:

$$\sum_{i=s+m}^{s+2m} p\left(\bigcup_{j,k} B_{jk}(\mathbf{q}_i) \cap C_{dyn}(t_i) \neq \emptyset\right) \tag{6}$$

where $C_{dyn}(t)$ are the workspace volumes occupied by dynamic human obstacles at time t. The trajectory being optimized during the time interval $[t_s, t_{s+m}]$ is executed during the next time interval $[t_{s+m}, t_{s+2m}]$. Therefore, the future human poses are considered only in the time interval $[t_{s+m}, t_{s+2m}]$.

The collision probability between the robot and the dynamic obstacle at time frame i in (6) can be computed as a maximum between bounding spheres:

$$\max_{j,k,l} p\left(B_{jk}(\mathbf{q_i}) \cap C_l(t_i) \neq \emptyset\right), \tag{7}$$

where $C_l(t_i)$ denotes bounding spheres for a human body at time t_i whose centers are located at line segments between human joints. The future human poses ξ_{future} are predicted in (4) and the bounding sphere locations $C_l(t_i)$ are derived from it. Note that the probabilistic distribution of each element in ξ_{future} is a linear combination of current action proposal $p(c^h|F, \mathbf{p})$ and Gaussians $p(\xi_{future}|F, \mathbf{p}, c^h)$ over all c^h, i.e., (7) can be reformulated as

$$\max_{j,k,l} \sum_{c^h} p(c^h|F, \mathbf{p}) p\left(B_{jk}(\mathbf{q_i}) \cap C_l(t_i) \neq \emptyset\right).$$

Let $\mathbf{z}_l^1$ and $\mathbf{z}_l^2$ be the probability distribution functions of two adjacent human joints obtained from $\xi_{future}(t_i)$, where the center of $C_l(t_i)$ is located between them by a linear interpolation $C_l(t_i) = (1-u)\mathbf{z}_l^1 + u\mathbf{z}_l^2$ where $0 \leq u \leq 1$. The joint positions follows Gaussian probability distributions:

$$\begin{aligned} \mathbf{z}_l^i &\sim \mathcal{N}(\mu_l^i, \Sigma_l^i) \\ \mathbf{c}_l(t_i) &\sim \mathcal{N}((1-u)\mu_l^1 + u\mu_l^2, (1-u)^2\Sigma_l^1 + u^2\Sigma_l^2) \quad (8) \\ &= \mathcal{N}(\mu_l, \Sigma_l), \quad (9) \end{aligned}$$

where $\mathbf{c}_l(t_i)$ is the center of $C_l(t_i)$. Thus, the collision probability between two bounding spheres is bounded by

$$\int_{\mathbb{R}^3} I(||\mathbf{x} - \mathbf{b}_{jk}(\mathbf{q}_i)||^2 \leq (r_1 + r_2)^2) f(\mathbf{x}) d\mathbf{x}, \tag{10}$$

where $\mathbf{b}_{jk}(\mathbf{q}_i)$ is the center of bounding sphere $B_{jk}(\mathbf{q}_i)$, r_1 and r_2 are the radius of $B_{jk}(\mathbf{q}_i)$ and $C_l(t_i)$, respectively, $I(\cdot)$ is an indicator function, and $f(\mathbf{x})$ is the probability distribution function. The indicator function restricts the integral domain to a solid sphere, and $f(\mathbf{x})$ is the probability density function of $\mathbf{c}_l(t_i)$, in (9). There is no closed form solution for (10), therefore we use the maximum possible value to approximate the probability. We compute $\mathbf{x}_{max}$ at which $f(\mathbf{x})$ is maximized in the sphere domain and multiply it by the volume of sphere, i.e.

$$p\left(B_{jk}(\mathbf{q_i}) \cap C_l(t_i) \neq \emptyset\right) \leq \frac{4}{3}\pi(r_1 + r_2)^2 f(\mathbf{x}_{max}). \tag{11}$$

Since even $\mathbf{x}_{max}$ does not have a closed form solution, we use the bisection method to find λ with

$$\mathbf{x}_{max} = (\Sigma^{-1} + \lambda I)^{-1}(\Sigma^{-1}\mathbf{p}_{lm} + \lambda \mathbf{o}_{jk}(\mathbf{q}_i)),$$

which is on the surface of sphere, explained in Generalized Tikhonov regularization Groetsch (1984) in detail.

The collision probability, computed in (10), is always positive due to the uncertainty of the future human position, and we compute a trajectory that is guaranteed to be nearly collision-free with sufficiently low collision probability. For a user-specified confidence level δ_{CL}, we compute a trajectory that its probability of collision is upper-bounded by $(1 - \delta_{CL})$. If it is unable to compute a collision-free trajectory, a new waypoint $\mathbf{q}_{new}$ is appended next to the last column of $\mathbf{Q}$ to make the robot wait at the last collision-free pose until it finds a collision-free trajectory. This approach computes a guaranteed collision-free trajectory, but leads to delay, which is fed to the Q-learning algorithm for the MDP task planner. The higher the delay that the collision-free trajectory of a task has, the less likely the task planner selects the task again.

Upper bound of collision probability

Using the predicted distribution and user-specified threshold δ_{CL}, we can compute an upper bound using the following lemma.

Lemma 1. *The collision probability is less than* $(1 - \delta_{CL})$ *if* $\frac{4}{3}\pi(r_1 + r_2)^2 f(\mathbf{x}_{max}) < 1 - \delta_{CL}$.

Proof. This bound follows Equations (10) and (11).

$$\begin{aligned} p\left(B_{jk}(\mathbf{q_i}) \cap C_l(t_i) \neq \emptyset\right) &\leq \frac{4}{3}\pi(r_1 + r_2)^2 f(\mathbf{x}_{max}) \\ &< 1 - \delta_{CL}. \end{aligned}$$

We use this bound in our optimization algorithm for collision avoidance.

Safe trajectory optimization

Our goal is to compute a robot trajectory that will either not collide with a human or that will reduce the probability of collision below a certain threshold. Sometimes, there is no feasible collision-free trajectory for the robot. However, if there is any trajectory for which the collision probability is less than a threshold, we want to be able to compute it. Our optimization-based planner also generates multiple initial trajectories and finds the best solution in parallel. In this manner, it expands the search space and reduces the probability of the robot being stuck in a local minima configuration. This can be expressed as the following theorem:

Theorem 2. *An optimization-based planner with n parallel threads will compute a global solution trajectory with a collision probability less than $(1-\delta_{CL})$, with the probability $1-(1-\frac{|A(\delta_{CL})|}{|S|})^n$, if it exists, where S is the entire search space. $A(\delta_{CL})$ corresponds to the neighborhood around the optimal solutions with a collision probability being less than $(1-\delta_{CL})$, where the local optimization converges to one of the global optima. $|\cdot|$ is the measure of the search or configuration space.*

We give a brief overview of the proof. In this case, $\frac{|A(\delta_{CL})|}{|S|}$ measures the probability that a random sample lies in the neighborhood of the global optima. In general, $|A(\delta_{CL})|$ will become smaller as the environment becomes more complex and has more local minima. Overall, this theorem provides a lower bound on the probability for our intention-aware planner with n threads. If the limit for n increases, the planner will compute the optimal solution, if it exists. This can be stated as:

Corollary 2.1. Probabilistic Completeness. *Our intention-aware motion planning algorithm with n trajectories becomes probabilistically complete as n increases.*

Proof. When the number of threads increases, we have a higher chance of computing the global optimal trajectory and, in the same manner, the increasing number of threads improves the probability that the planner computes an acceptable solution, according to the following:

$$\lim_{n\to\infty} 1-\left(1-\frac{|A(\delta_{CL})|}{|S|}\right)^n = 1.$$

Theorem 2 and Corollary both implicate that the use of more threads with different initial random function seed values increases the probability of finding a valid solution, where validity is a trajectory with the upper bound of collision probability at less than a certain safety value, as shown in Lemma 1. Since I-Planner uses an optimization-based formulation and all optimization threads run independently, the parallel algorithm significantly increases the probability of finding a reliable trajectory with a small collision probability while not taking additional running time. From the implications of Lemma 1, Theorem 2, and Corollary , our I-Planner algorithm with multiple optimization threads can efficiently find a valid and safe robot trajectory with a collision probability less than a certain safety value.

Implementation and performance

We highlight the performance of our algorithm in a situation where the robot is performing a collaborative task with a human and computing safe trajectories. We use a 7-DOF KUKA-IIWA robot arm. The human motion is captured by a Kinect sensor operating with a 15Hz frame rate, and only upper body joints are tracked for collision checking. We use the ROS software Quigley et al. (2009) for robot control and sensor data communication. The motion planner has a 0.5s re-planning timestep, The number of pseudo-inputs M of SPGPs is set to 100 so that the prediction computation is performed online.

Performance of human motion prediction

We have tested our human motion prediction model on labeled motion datasets corresponding to a human's reaching action. Our human motion prediction model allows human joint position errors, as discussed in the previous section. To demonstrate the robustness of our model against input noise errors, we measure the classification accuracy and regression accuracy, varying the sensor error parameter and the maximum human joint velocity limits.

In the human reaching action motion datasets, the human is initially in a static pose in front of a table. Then, his or her left or right arm moves and reaches towards one of the target locations on the table. Then the arm returns to the initial pose. The dataset contains 8 different target positions on the table and 30 reaching motions for each target position. Because the result of our human motion prediction model depends on human joint velocities, we synthesize fast and slow motions by changing the speed of captured motions.

We measure the correctness of human motion classification and human motion regression in the following ways:

- Motion classification precision: Because human motion is continuous and the transition between motions can be difficult to measure, we only count the motion frames in which the multi-class classifier yields the highest probability greater than 50%.
- Motion regression precision: At a certain time, human motion trajectory for the next T seconds is predicted. We compute the regression precision as the integral of distance between the predicted and ground-truth motions over the T-second window as:

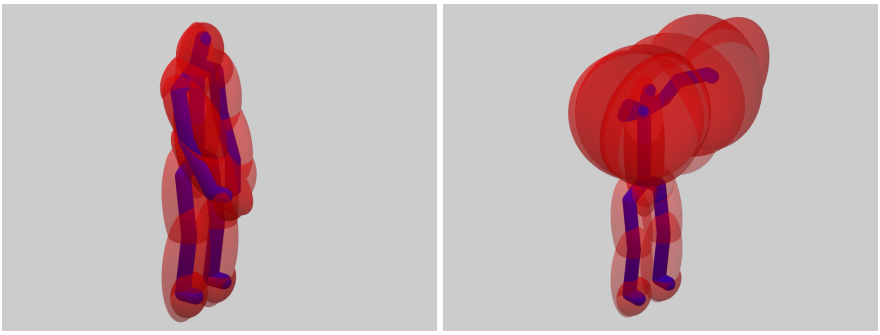
$$\int_0^T \sum_{i:joint} ||\mathbf{p}_{pred,i}(t) - \mathbf{p}_{true,i}(t)||^2 \, dt,$$

  where $\mathbf{p}_{pred}(t)$ is the collection of resulting mean values of the Gaussian Process with noisy input for joint i.
- Motion regression accuracy: As with the precision, accuracy is the integral of the volume of ellipsoids generated by the Gaussian distributions:

$$\int_0^T \sum_{i:joint} \det\left(\mathbf{K}_{i,pred}(t)\right) \, dt,$$

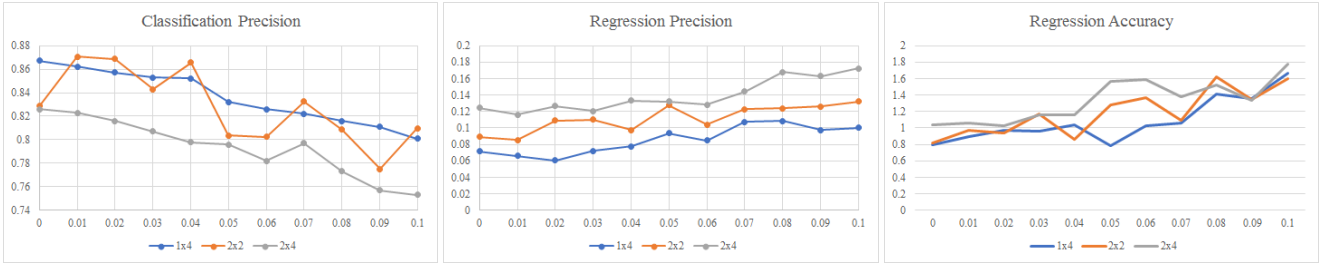
  where $\mathbf{K}_{i,pred}(t)$ is the collection of resulting variance of the Gaussian Process with noisy input for joint i. In this case, a lower value results in a better prediction result.

Figure 4 shows the human motion prediction results. The Gaussian Process regression algorithm corresponds to the use of Gaussian distribution ellipsoids around the predicted mean values of the joint positions. In (a), when the human is in an idle position, the motion classifier results in near uniform distribution among the action classes and the motion regression algorithm generates human motion trajectories that progress slightly towards the target positions. In (b), when the human arm gets close to the target position, the motion classifier predicts the motion class with a dominant probability and the motion regression algorithm infers a correct future motion trajectory that is more accurate than



(a) (b)

Figure 4. Human motion prediction results: The result of human motion prediction is represented by Gaussian distributions for each skeleton joint. The ellipsoid boundaries within which the integral of Gaussian distribution probability is 95% are drawn in red and the human skeleton is shown in blue. Bounding ellipsoids have transparency values that are proportional to the action classifier probability. (a) Undistinguished human action class: This occurs when the classifier fails to distinguish the human action class, generating nearly uniform probability distribution among the action classes. (b) Prediction results when untrained human motion is given: These cases result in larger boundary spheres around the human skeleton. This is because, in the Gaussian Process, the output has a uniform mean and a high variance when the input point is outside the range of the training input data.



(a) (b) (c)

Figure 5. Performance of human motion prediction: The precision/accuracy of classification and regression algorithms are shown in the graphs, with varying input noise. (a) Classification precision versus input noise: We only take the input points where the probability of an action classification is dominant, i.e. probability $> 50\%$. (b) Regression precision versus input noise: Measured by the integral of distance between the predicted human motion and the ground-truth human motion trajectory. (c) Regression precision versus input noise: The integral of the volume of Gaussian distribution ellipsoids.

(a). In (c), when an untrained human motion is given, the motion classifier and the motion regression algorithm result in uniform values and high variances, respectively, generating a conservative collision bound around the human. This is a normal behavior in terms of classification and regression because the given input point is outside the range of training data input points.

Figure 5 shows the classification precision, regression precision, and accuracy with varying input noises. When the input noise is high, the accuracy of human motion classification and regression can decrease.

Comparison with prior work

We compared our method against Anticipatory Temporal Conditional Random Field (ATCRF) Koppula and Saxena (2016), a prior method for human activity classification and future motion regression. The ATCRF algorithm creates a learning model based on Conditional Random Field, learning human intentions from a dataset with annotated human action classes, human motion trajectories, and object affordances (the functionality of the object). For this comparison, we used the CAD-120 dataset Koppula et al. (2013), which contains 120 RGB-D videos of 4 different subjects and 10 activity classes. We also train their algorithm on our dataset, which contains 100 RGB-D videos with 5 activity classes performed by one subject.

By using additional information about properties of objects close to humans, ATCRF can predict the action classes and generate future human motion trajectories. However, compared to our human motion prediction method, ATCRF cannot handle noisy input or predict the variance around the predicted motion trajectories. Thus, we measure the motion classification and regression precision to compare the performance of our method with that of ATCRF. For a fair comparison, the input noise parameter in our human motion prediction is set to zero.

Table 1 shows the results of ATCRF and our human motion prediction method. The classification precision and recall of our algorithm is lower than that of the ATCRF algorithm. Note that ATCRF uses object affordance annotations as additional information in the learning phase, whereas our algorithm does not. There is a big difference in motion regression result. With our definition of regression precision, our algorithm predicts human motion trajectory 2 times closer to the ground-truth human motion trajectory than that of ATCRF. Also, we could measure regression accuracy with our algorithm that could be used in probabilistic collision detection, whereas ATCRF could not.

Algorithm	CAD-120 Dataset			
	Action Classification		Motion Regression	
	Precision	Recall	Precision	Accuracy
ATCRF	**74.8**	**66.2**	0.305	N/A
Our model	68.3	60.3	**0.153**	**1.25**

Table 1. Performances of action classification and motion regression of ATCRF Koppula and Saxena (2016) and our human motion prediction algorithm. As ATCRF algorithm does not predict the variance of motion around the trajectory, the motion regression accuracy cannot be measured on the algorithm. ATCRF uses object affordance annotations as additional information in the learning phase, and we got higher precision and recall with ATCRF than with our algorithm. In terms of motion regression, our algorithm outperformed ATCRF in regression precision.

Robot motion planning with human motion prediction

In the simulated benchmark scenario, the human is sitting in front of a desk. In this case, the robot arm helps the human by delivering objects from one position that is far away from the human to target position closer to the human. The human waits till the robot delivers the object. As different tasks are performed in terms of picking the objects and their delivery to the goal position, the temporal coherence is used to predict the actions. The action set for a human is $A^h = \{Take_0, Take_1, \cdots\}$, where $Take_i$ represents an action of taking object i from its current position to the new position. The action set for the robot arm is defined as $A^r = \{Fetch_0, Fetch_1, \cdots\}$.

We quantitatively measure the following values:

- Modified Hausdorff Distance (MHD) Dubuisson and Jain (1994): The distance between the ground-truth human trajectory and the predicted mean trajectory is used. In our experiments, MHD is measured for an actively moving hand joint over 1 second. This evaluates the quality of the anticipated trajectory of the human motion.
- Smoothness: We also measure the smoothness of the robot's trajectory with and without human motion prediction results. The smoothness is computed as

$$\frac{1}{T}\int_0^T \sum_{i=1}^{n} \ddot{\mathbf{q}}_i(t)^2 \, dt, \tag{12}$$

  where the two dots indicate acceleration of joint angles.
- Jerkiness: Jerkiness is defined as

$$\max_{0\le t\le T} \sum_{i=1}^{n} \dddot{\mathbf{q}}_i(t)^2. \tag{13}$$

- Distance: The closest distance between the robot and the human during task collaboration.
- Efficiency: The ratio of the task completion time when the robot and the human collaborate to complete all the subtasks to the task completion time when the human performs all the tasks without any help from the robot. This is used to evaluate the capability of the resulting human-robot system.

To compare the performance of our I-Planner, we use the following algorithm:

- **ITOMP**. This model is the same as the real-time motion planner for dynamic environments, ITOMP Park et al. (2012), without human motion prediction.
- **I-Planner, no noisy input (I-Planner, no NI)**. This is our motion planning algorithm with human motion prediction, but the motion prediction does not assume noisy input. The details of this algorithm are given in the preliminary paper Park et al. (2017c).
- **I-Planner, noisy input (I-Planner, NI)**. This is the modified algorithm presented in the previous section, which also considers noise in the human motion prediction data.

Table 2 highlights the performance of our algorithm in three different variations of this scenario: *arrangements of blocks*, *task order* and *confidence level*. The numbers in the "Task Order" column indicates the identifiers of human actions. Parentheses mean that the human actions in the parentheses can be performed in any order. Arrows mean that the right actions can be performed only if the left actions are done. For example, $(0, 1) \rightarrow (2, 3)$ means that the possible action orders are $0 \rightarrow 1 \rightarrow 2 \rightarrow 3$, $0 \rightarrow 1 \rightarrow 3 \rightarrow 2$, $1 \rightarrow 0 \rightarrow 2 \rightarrow 3$ and $1 \rightarrow 0 \rightarrow 3 \rightarrow 2$. Table 3 shows the performance of our algorithm in a simulation compared to algorithms without human motion prediction or input noise. The 3 different algorithms were tested in 9 different scenarios and we found that our complete algorithm outperformed other versions of the algorithm in various measurements in most scenarios. Table 4 highlights the performance of our algorithm with a real robot. Our algorithm has been implemented on a PC with 8-core i7-4790 CPU. We used OpenMP to parallelize the computation of future human motion prediction and probabilistic collision checking.

Robot motion responses to human motion speed

Figure 6 shows the robot's responses to three different speeds of human movements in a human-robot scenario. In this scenario, the human arm serves as a block or an obstacle to the robot's motion from left to right. The human moves at a slow speed in (a), a medium speed in (b), and a fast speed in (c). Because the human motion prediction and the robot motion planning process operate in parallel at the same time, the motion planner needs to account for the current results of the motion predictor. If the human moves slowly, the robot motion planner is given the future predicted human motion and the planner has enough planning time to adjust the robot's trajectory. This results in smooth and collision-free trajectories for the robot. However, if the robot moves quickly, the prediction is not very accurate due to the limited processing time. At the next planning timestep, the

Scenarios	Arrangement	Task Order	Confidence Level	Average Prediction Time
Different Arrangements	1×4	$(0,1) \rightarrow (2,3)$	0.95	52.0 ms
	2×2	$(1,5) \rightarrow (2,6)$	0.95	72.4 ms
	2×4	$(0,4) \rightarrow (1,5) \rightarrow (2,6) \rightarrow (3,7)$	0.95	169 ms
Temporal Coherence	1×4	$0 \rightarrow 1 \rightarrow 2 \rightarrow 3$	0.95	52.1 ms
	1×4	Random	0.95	105 ms
	1×4	$(0,2) \rightarrow (1,3)$	0.95	51.7 ms
Confidence Level	1×4	$0 \rightarrow (1,2) \rightarrow 3$	0.90	47.2 ms
	1×4	$0 \rightarrow (1,2) \rightarrow 3$	0.95	50.7 ms
	1×4	$0 \rightarrow (1,2) \rightarrow 3$	0.99	155 ms

Table 2. Three different simulation scenarios: *Different Arrangements*, *Temporal Coherence*, and *Confidence Level*. We consider different arrangements of blocks as well as the confidence levels used for probabilistic collision checking. These confidence levels are used for motion prediction.

Scenarios	Model	Prediction Time	MHD	Smoothness	Jerkiness	Distance	Efficiency
Different Arrangements (1)	ITOMP	N/A	N/A	2.96	5.23	3.2 cm	1.2
	I-Planner, no NI	52.0 ms	6.7 cm	1.08	1.52	6.7 cm	1.6
	I-Planner, NI	58.5 ms	7.2 cm	0.92	1.03	8.1 cm	1.7
Different Arrangements (2)	ITOMP	N/A	N/A	5.78	7.19	2.3 cm	1.1
	I-Planner, no NI	72.4 ms	6.2 cm	1.04	1.60	8.2 cm	1.6
	I-Planner, NI	70.8 ms	7.0 cm	0.84	1.32	10.7 cm	1.6
Different Arrangements (3)	ITOMP	N/A	N/A	4.82	6.82	1.6 cm	1.2
	I-Planner, no NI	169 ms	10.4 cm	1.15	1.30	6.2 cm	1.6
	I-Planner, NI	150 ms	12.6 cm	1.02	1.20	8.9 cm	1.6
Temporal Coherence (1)	ITOMP	N/A	N/A	1.79	3.22	6.0 cm	1.3
	I-Planner, no NI	52.1 ms	4.3 cm	0.65	1.56	9.3 cm	1.5
	I-Planner, NI	48.9 ms	5.2 cm	0.62	1.53	9.7 cm	1.5
Temporal Coherence (2)	ITOMP	N/A	N/A	5.49	7.30	2.0 cm	1.0
	I-Planner, no NI	105 ms	8.2 cm	1.21	1.28	8.8 cm	1.6
	I-Planner, NI	110 ms	9.9 cm	1.08	1.10	9.9 cm	1.6
Temporal Coherence (3)	ITOMP	N/A	N/A	3.12	3.18	8.0 cm	1.3
	I-Planner, no NI	51.7 ms	6.8 cm	1.00	1.20	12.1 cm	1.5
	I-Planner, NI	60.0 ms	8.2 cm	0.79	0.93	13.2 cm	1.5
Confidence Level (1)	ITOMP	N/A	N/A	2.90	3.40	7.2 cm	1.2
	I-Planner, no NI	47.2 ms	7.9 cm	1.17	1.58	9.5 cm	1.6
	I-Planner, NI	52.1 ms	9.1 cm	1.08	1.32	10.2 cm	1.7
Confidence Level (2)	ITOMP	N/A	N/A	3.12	3.80	10.3 cm	1.2
	I-Planner, no NI	50.7 ms	7.9 cm	1.28	1.71	13.3 cm	1.6
	I-Planner, NI	48.2 ms	9.1 cm	1.20	1.66	14.1 cm	1.8
Confidence Level (3)	ITOMP	N/A	N/A	3.76	4.33	13.0 cm	1.2
	I-Planner, no NI	155 ms	7.9 cm	1.40	1.90	16.2 cm	1.7
	I-Planner, NI	129 ms	9.1 cm	1.35	1.73	17.7 cm	1.8

Table 3. Performance of our planning algorithm in terms of robot motion simulation. The use of motion prediction results in a smoother trajectory and we observe up to 4X improvement in our smoothness metric, as defined in Equation (12). Our resulting planning algorithm (I-Planner) runs in real-time.

Scenarios	Model	Prediction Time	MHD	Smoothness	Jerkiness	Distance
Waving Arms	ITOMP	N/A	N/a	4.88	6.23	2.1 cm
	I-Planner, no NI	20.9 ms	5.0 cm	0.91	1.33	9.3 cm
	I-Planner, NI	23.5 ms	6.1 cm	0.83	1.25	10.5 cm
Moving Cans	ITOMP	N/A	N/A	5.13	7.83	3.9 cm
	I-Planner, no NI	51.7 ms	7.3 cm	1.04	1.82	8.7 cm
	I-Planner, NI	50.0 ms	8.8 cm	0.93	1.32	13.5 cm

Table 4. Performance of our planning algorithm on a real robot running on a 7-DOF Fetch robot next to dynamic human obstacles. Our online motion planner computes safe trajectories for challenging benchmarks like "moving cans." We observe almost 5X improvement in the smoothness of the trajectory due to our prediction algorithm.

robot's trajectory may abruptly change to avoid the current human pose, generating a jerky or non-smooth motion. This highlights how the performance of the prediction algorithm affects the smoothness of a robot's trajectory.

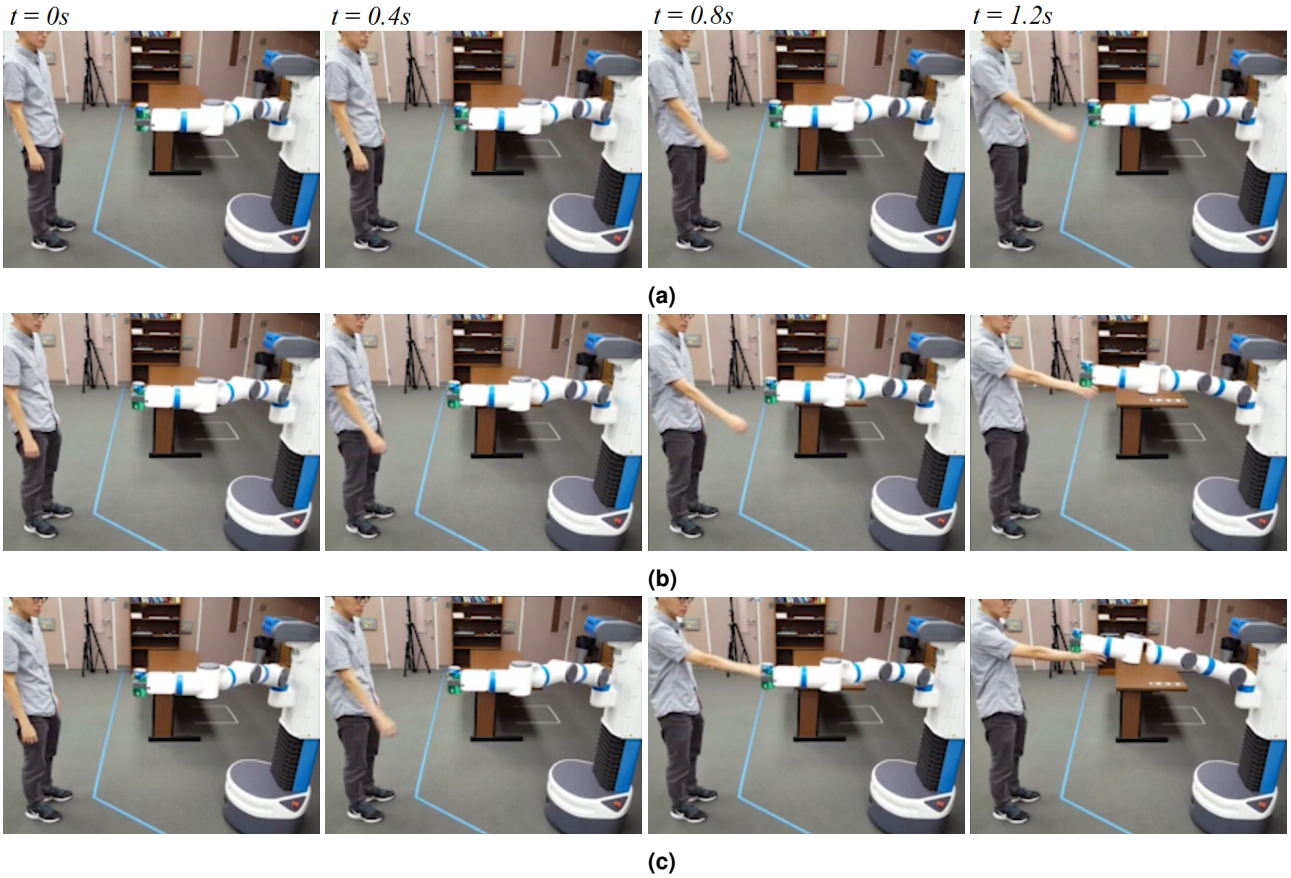


Figure 6. Responses to three different human arm movement speeds: While the robot arm moves from left to right, the human moves his arm to block the robot's trajectory at different speeds. (a) The human arm moves slowly. The robot has enough time to predict the human arm motion, generating the smoothest and the least jerky robot trajectory. (b) As the human moves at a medium speed, the robot predicts the human's future motion, recognizes that it will block the robot's path, and therefore changes the trajectory upwards (at $t = 0.8s$) to avoid the obstacle and generate a smooth trajectory. (c) When the human arm moves faster, the robot trajectory abruptly changes to move upwards (at $t = 0.8s$), generating a less smooth trajectory while still avoiding the human.

Benefits of our prediction algorithm

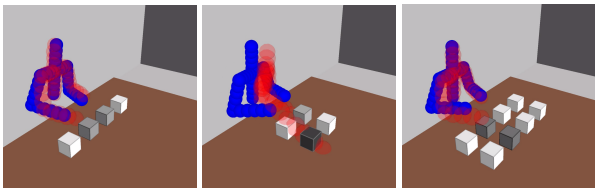


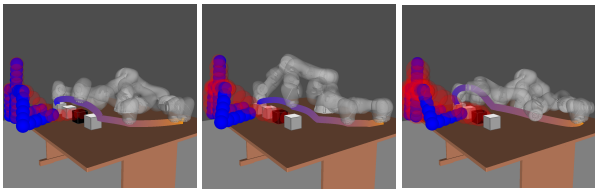
Figure 7. Different block arrangements: Different arrangements in terms of the positions of the blocks, results in different human motions and actions. Our planner computes their intent for safe trajectory planning. The different arrangements are: (a) 1×4. (b) 2×2. (c) 2×4.

In the *Different Arrangements* scenarios, the position and layout of the blocks changes. Fig. 7 shows three different arrangements of the blocks: 1×4, 2×2 and 2×4. In the two cases 2×2 and 2×4, where positions are arranged in two rows unlike the 1×4 scenario, the human arm blocks a movement from a front position to the back position. As a result, the robot needs to compute its trajectory accordingly.

Depending on the temporal coherence present in the human tasks, the human intention prediction may or may not improve the performance of our the task planner. It is shown in the *Temporal Coherence* scenarios. In the sequential order coherence, the human intention is predicted accurately with our approach with 100% certainty. In the random order, however, the human intention prediction step is not accurate until the human hand reaches the specific position. The personal order varies for each human, and reduces the possibility of predicting the next human action. When the right arm moves forward a little, $Fetch_0$ is predicted as the human intention with a high probability whereas $Fetch_1$ is predicted with low probability, even though position 1 is closer than position 0.

In the *Confidence Level* scenarios, we analyze the effect of confidence level δ_{CD} on the trajectory computed by the planner, the average task completion time, and the average motion planning time. As the confidence level becomes higher, the robot may not take the smoothest and shortest path so as to compute a collision-free path that is consistent with the confidence level.

In all cases, we observe the prediction results in smoother trajectory, using the smoothness metric defined as Equation (12). This is because the robot changes its path in advance before the human obstacle actually blocks the robot's shortest path if human motion prediction is used.



(a) (b) (c)

Figure 8. Probabilistic collision checking with different confidence levels: A collision probability less $(1 - \delta_C D)$ implies a safe trajectory. The current pose (i.e., blue spheres) and the predicted future pose (i.e. red spheres) are shown. The robot's trajectory avoids these collisions before the human performs its action. The higher the confidence level is, the longer the distance between the human arm and the robot trajectory. (a) $\delta_{CD} = 0.90$. (b) $\delta_{CD} = 0.95$. (c) $\delta_{CD} = 0.99$.

Evaluation using 7-DOF Fetch robot

We integrated our planner with the 7-DOF Fetch robot arm and evaluted in complex 3D workspaces. The robot delivers four soda cans from start locations to target locations on a desk. At the same time, the human sitting in front of the desk picks up and takes away the soda cans delivered to the target positions by the robot, which can cause collisions with the robot arm. In order to evaluate the collision avoidance capability of our approach, the human intentionally starts moving his arm to a soda can at a target location, blocking the robot's initially planned trajectory, when the robot is delivering another can moving fast. Our intention aware planner avoids collisions with the human arm and results in a smooth trajectory compared to motion planner results without human motion prediction.

Figure 1 shows two sequences of robot's trajectories. In the first row, the robot arm trajectory is generated an ITOMP Park et al. (2012) re-planning algorithm without human motion prediction. As the human and the robot arm move too fast to re-plan collision-free trajectory. As a result, the robot collides (the second figure) or results in a jerky trajectory (the third figure). In the second row, our human motion prediction approach is incorporated as described in Section . The robot re-plans the arm trajectory before the human actually blocks its way, resulting in collision-free path.

Conclusions, limitations, and future work

We present a novel intention-aware planning algorithm to compute safe robot trajectories in dynamic environments with humans performing different actions. Our approach uses offline learning of human motions and can account for large noise in terms of depth cameras. At runtime, our approach uses the learned human actions to predict and estimate the future motions. We use upper bounds on collision guarantees to compute safe trajectories. We highlight the performance of our planning algorithm in complex benchmarks for human-robot cooperation in both simulated and real world scenarios with 7-DOF robots. Compared to Park et al. (2017c), our improved human motion prediction model can better handle input noise and can generate smoother robot trajectories.

Our approach has some limitations. As the number of human action types increases, the number of states of MDP can increase significantly. In this case, it may be useful to use POMDP for robot motion planning under uncertainty Kurniawati et al. (2012). Our classification can be improved with more information of the robotic environment such as object affordances. So we would like to improve our algorithm by using additional annotations as future work. Our probabilistic collision checking formulation is limited to environment uncertainty and does not take into account robot control errors. The performance of motion prediction algorithm depends on the variety and size of the learned data. Currently, we use supervised learning with labeled action types, but it would be useful to explore unsupervised learning based on appropriate action clustering algorithms. Furthermore, our analysis of human motion prediction noise assumes the use of a Gaussian Process model and it would be useful to extend it other noise models. Moreover, we would to measure the impact of robot actions on human motion, and thereby establish a two-way coupling between robot and human actions. Our realtime planner can also be combined with natural language processing to generate robot movements Park et al. (2017a).

Acknowledgements

This research is supported in part by ARO grant W911NF16-1-0085 and Intel.

References

(2010) *Prime Sensor NITE 1.3 Algorithms notes*. PrimeSense Inc. URL http://www.primesense.com. Last viewed 19-01-2011 15:34.

Bai H, Cai S, Ye N, Hsu D and Lee WS (2015) Intention-aware online pomdp planning for autonomous driving in a crowd. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, pp. 454–460.

Bandyopadhyay T, Jie CZ, Hsu D, Ang Jr MH, Rus D and Frazzoli E (2013) Intention-aware pedestrian avoidance. In: *Experimental Robotics*. Springer, pp. 963–977.

Bera A, Kim S, Randhavane T, Pratapa S and Manocha D (2016) Glmp-realtime pedestrian path prediction using global and local movement patterns. *ICRA* .

Busoniu L, Babuska R and De Schutter B (2008) A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 38(2): 156–172.

Choo B, Landau M, DeVore M and Beling PA (2014) Statistical analysis-based error models for the microsoft kinecttm depth sensor. *Sensors* 14(9): 17430–17450.

Dragan AD, Bauman S, Forlizzi J and Srinivasa SS (2015) Effects of robot motion on human-robot collaboration. In: *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*. ACM, pp. 51–58.

Dragan AD and Srinivasa SS (2013) A policy-blending formalism for shared control. *The International Journal of Robotics Research* 32(7): 790–805.

Dubuisson MP and Jain AK (1994) A modified hausdorff distance for object matching. In: *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing.,*

Proceedings of the 12th IAPR International Conference on, volume 1. IEEE, pp. 566–568.

Ek CH, Torr PH and Lawrence ND (2007) Gaussian process latent variable models for human pose estimation. In: *Machine learning for multimodal interaction*. Springer, pp. 132–143.

Fox D, Burgard W and Thrun S (1997) The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* 4(1): 23–33.

Fragkiadaki K, Levine S, Felsen P and Malik J (2015) Recurrent network models for human dynamics. In: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 4346–4354.

Fulgenzi C, Spalanzani A and Laugier C (2007) Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In: *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, pp. 1610–1616.

Groetsch CW (1984) The theory of tikhonov regularization for fredholm equations of the first kind .

Hawkins KP, Vo N, Bansal S and Bobick AF (2013) Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration. In: *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*. IEEE, pp. 499–506.

Hofmann M and Gavrila DM (2012) Multi-view 3d human pose estimation in complex environment. *International journal of computer vision* 96(1): 103–124.

Kim S, Guy SJ, Liu W, Wilkie D, Lau RW, Lin MC and Manocha D (2014) Brvo: Predicting pedestrian trajectories using velocity-space reasoning. *The International Journal of Robotics Research* .

Koppula HS, Gupta R and Saxena A (2013) Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research* 32(8): 951–970.

Koppula HS, Jain A and Saxena A (2016) Anticipatory planning for human-robot teams. In: *Experimental Robotics*. Springer, pp. 453–470.

Koppula HS and Saxena A (2016) Anticipating human activities using object affordances for reactive robotic response. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 38(1): 14–29.

Kuderer M, Kretzschmar H, Sprunk C and Burgard W (2012) Feature-based prediction of trajectories for socially compliant navigation. In: *Robotics: science and systems*. Citeseer.

Kurniawati H, Bandyopadhyay T and Patrikalakis NM (2012) Global motion planning under uncertain motion, sensing, and environment map. *Autonomous Robots* 33(3): 255–272.

Kurniawati H, Du Y, Hsu D and Lee WS (2011) Motion planning under uncertainty for robotic tasks with long time horizons. *The International Journal of Robotics Research* 30(3): 308–323.

Mainprice J and Berenson D (2013) Human-robot collaborative manipulation planning using early prediction of human motion. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, pp. 299–306.

McHutchon A and Rasmussen CE (2011) Gaussian process training with input noise. In: *Advances in Neural Information Processing Systems*. pp. 1341–1349.

Müller M (2007) Dynamic time warping. *Information retrieval for music and motion* : 69–84.

Nikolaidis S, Lasota P, Rossano G, Martinez C, Fuhlbrigge T and Shah J (2013) Human-robot collaboration in manufacturing: Quantitative evaluation of predictable, convergent joint action. In: *Robotics (ISR), 2013 44th International Symposium on*. IEEE, pp. 1–6.

Pan J, Chitta S and Manocha D (2011) Probabilistic collision detection between noisy point clouds using robust classification. In: *International Symposium on Robotics Research (ISRR)*.

Park C, Pan J and Manocha D (2012) ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments. In: *Proceedings of International Conference on Automated Planning and Scheduling*.

Park C, Pan J and Manocha D (2013) Real-time optimization-based planning in dynamic environments using GPUs. In: *Proceedings of IEEE International Conference on Robotics and Automation*.

Park C, Park JS and Manocha D (2016) Fast and bounded probabilistic collision detection for high-dof robots in dynamic environments. In: *Workshop on Algorithmic Foundations of Robotics*.

Park JS, Jia B, Bansal M and Manocha D (2017a) Generating realtime motion plans from complex natural language commands using dynamic grounding graphs. *CoRR* abs/1707.02387. URL http://arxiv.org/abs/1707.02387.

Park JS, Park C and Manocha D (2017b) Efficient probabilistic collision detection for non-convex shapes. In: *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE.

Park JS, Park C and Manocha D (2017c) Intention-aware motion planning using learning based human motion prediction. *Proceedings of Robotics: Science and Systems* .

Pérez-D'Arpino C and Shah JA (2015) Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 6175–6182.

Petti S and Fraichard T (2005) Safe motion planning in dynamic environments. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 2210–2215.

Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R and Ng AY (2009) Ros: an open-source robot operating system. In: *ICRA workshop on open source software*, volume 3. Kobe, Japan, p. 5.

Shotton J, Sharp T, Kipman A, Fitzgibbon A, Finocchio M, Blake A, Cook M and Moore R (2013) Real-time human pose recognition in parts from single depth images. *Communications of the ACM* 56(1): 116–124.

Sisbot EA, Marin-Urias LF, Alami R and Simeon T (2007) A human aware mobile robot motion planner. *Robotics, IEEE Transactions on* 23(5): 874–883.

Snelson E and Ghahramani Z (2005) Sparse gaussian processes using pseudo-inputs. In: *Advances in neural information processing systems*. pp. 1257–1264.

Sutton RS and Barto AG (1998) *Reinforcement learning: An introduction*. MIT press.

Turaga P, Chellappa R, Subrahmanian VS and Udrea O (2008) Machine recognition of human activities: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on* 18(11): 1473–1488.

Unhelkar VV, Pérez-DArpino C, Stirling L and Shah JA (2015) Human-robot co-navigation using anticipatory indicators of human walking motion. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, pp. 6183–6190.

Urtasun R, Fleet DJ and Fua P (2006) 3d people tracking with gaussian process dynamical models. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1. IEEE, pp. 238–245.

Van Den Berg J, Patil S and Alterovitz R (2012) Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research* 31(11): 1263–1278.

van den Berg J, Patil S, Sewall J, Manocha D and Lin M (2008) Interactive navigation of multiple agents in crowded environments. In: *Proceedings of the 2008 symposium on Interactive 3D graphics and games*. ACM, pp. 139–147.

Vasquez D, Fraichard T and Laugier C (2009) Growing hidden markov models: An incremental tool for learning and predicting human and vehicle motion. *The International Journal of Robotics Research* .

Zhu J and Hastie T (2012) Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics* .

Ziebart BD, Maas AL, Bagnell JA and Dey AK (2008) Maximum entropy inverse reinforcement learning. In: *AAAI*. pp. 1433–1438.