ASME Journal of Manufacturing Science and Engineering
Online journal at:
https://asmedigitalcollection.asme.org/manufacturingscience

A Spatio-Temporal Prediction and Planning Framework for Proactive Human–Robot Collaboration

Jared Flowers[1]
Department of Mechanical and Aerospace Engineering,
University of Florida,
1064 Center Dr., Rm. 181,
Gainesville, FL 32653
e-mail: jared.flowers@ufl.edu

Gloria Wiens
Department of Mechanical and Aerospace Engineering,
University of Florida,
1064 Center Dr., Rm. 181,
Gainesville, FL 32653
e-mail: gwiens@ufl.edu

A significant challenge in human–robot collaboration (HRC) is coordinating robot and human motions. Discoordination can lead to production delays and human discomfort. Prior works seek coordination by planning robot paths that consider humans or their anticipated occupancy as static obstacles, making them nearsighted and prone to entrapment by human motion. This work presents the spatio-temporal avoidance of predictions-prediction and planning framework (STAP-PPF) to improve robot–human coordination in HRC. STAP-PPF predicts multi-step human motion sequences based on the locations of objects the human manipulates. STAP-PPF then proactively determines time-optimal robot paths considering predicted human motion and robot speed restrictions anticipated according to the ISO15066 speed and separation monitoring (SSM) mode. When executing robot paths, STAP-PPF continuously updates human motion predictions. In real-time, STAP-PPF warps the robot's path to account for continuously updated human motion predictions and updated SSM effects to mitigate delays and human discomfort. Results show the STAP-PPF generates robot trajectories of shorter duration. STAP-PPF robot trajectories also adapted better to real-time human motion deviation. STAP-PPF robot trajectories also maintain greater robot/human separation throughout tasks requiring close human–robot interaction. Tests with an assembly sequence demonstrate STAP-PPF's ability to predict multi-step human tasks and plan robot motions for the sequence. STAP-PPF also most accurately estimates robot trajectory durations, within 30% of actual, which can be used to adapt the robot sequencing to minimize disruption. [DOI: 10.1115/1.4063502]

Keywords: human–robot collaboration, human–robot interaction, robot motion planning, control and automation, production systems optimization, robotics and flexible tooling

1 Introduction

A significant challenge in human–robot collaboration (HRC) in manufacturing is coordinating robot and human motions for synergistic operation. In an HRC workcell, humans and robots work together in shared workspaces, requiring close proximity, to complete tasks. The unique skills of the robot and of the human complement each other to accomplish tasks collaboratively [1]. The Industry 4.0 (I4.0) revolution aims for smart factories of the future to permit mass customization of products, requiring advancements in HRC workcells, such as enhanced sensing and data processing, to allow flexibility in manufacturing [2,3]. Prior HRC workcells in industry still require mostly rigid design of tasks to avoid unintended human contact. Industry 5.0 (I5.0) requires further advancement to HRC by improving human–robot interactions, requiring advancements in humans' perception of safety and human-machine work balance [4]. HRC workcells in the smart factory envisioned by Society 5.0 (S5.0) must also improve the quality of life for workers via improved human–robot relationships [5]. The work, herein, contributes to the features of the HRC workcell in the smart factory of the future described by I4.0, I5.0, and S5.0 by developing the spatio-temporal avoidance of predictions-prediction and planning framework (STAP-PPF). STAP-PPF combines human motion prediction with proactive robot path planning. The goal of STAP-PPF is to generate robot paths that better coordinate with humans to improve robot efficiency (permitting flexibility for I4.0) and improve human safety and comfort (improving human-machine relationships for I5.0 and S5.0) in close human–robot collaboration.

The evolution of robot's role in manufacturing can be described in three stages [2]. The first stage, Robotics 1.0, occurred between the 1960s and 1980s. It is characterized by industrial robots being programmed to perform repetitive tasks due to minimal sensing capability and no tolerance for sequence deviations. The second stage of robot evolution, Robotics 2.0, occurred between the 1990s and 2000s. It is characterized by increased sensing capability, data processing capability, and capability for communication with other factory processes to enable more intelligent robots. During

[1]Corresponding author.

Manuscript received May 31, 2023; final manuscript received September 12, 2023; published online October 17, 2023. Assoc. Editor: Pai Zheng.

robotics 2.0, the first collaborative robot was introduced to permit robots and humans to share workspaces. The third stage of robotic evolution, Robotics 3.0, began in the 2010s and is ongoing. It is enabled by data sharing between machines, increased computing power, new artificial intelligence (AI) for processing data such as images and other signals from sensor suites. Robotics 4.0 is anticipated to start in this decade due to further increased computing power, data communication capabilities, and more powerful AI models.

Robotics 3.0 also coincides with I4.0. I4.0 is the fourth industrial revolution which seeks on-demand manufacturing and greater product customization (small-batch, high-mix products), production flexibility, and greater use of collaborative robots to achieve those goals [3]. I4.0 is also marked by technological developments such as Internet of Things to improve data sharing and processing among machines, including cloud computing [2]. The next industrial revolution is I5.0. It is marked by improvements to human-machine interactions and load balancing between humans and machines [4]. I5.0 also aims to make manufacturing more sustainable and supply chains more resilient. S5.0 is a societal revolution related to I5.0 which seeks to improve the quality of life through fusion of the cyberspace and physical space [5]. Both I5.0 and S5.0 are enabled by AI advancements and seek to improve the relationship between technology and humans.

STAP-PPF is motivated by the challenges of coordinating human and robot motions in HRC workcells to permit goals of I4.0, I5.0, and S5.0. If robot and human motions are not optimally coordinated, then the robot may have to reduce speed or stop to avoid collisions according to HRC safety standards [6]. Such reactive behavior leads to production delays [7]. Another consequence of uncoordinated robot and human motions is human discomfort. Uncoordinated motions can lead to robot paths that take the robot very close to the human and make the human feel less safe [8]. Therefore, a solution that coordinates robot and human motions can improve productivity and humans' sense of safety. Achieving coordination in an HRC workcell presents a challenge because, traditionally, it requires nearly perfect workcell layout and robot and human sequencing to maintain efficiency and human safety [9]. Perfect workcell design can be very time consuming as robot motions must be carefully designed to avoid robot/human interference while still permitting robot motions. This increases changeover time when new processes are introduced to a workcell and diminishes the degree of transformable production [9]. The lack of flexibility reduces manufactures' capabilities for transformable production with mass customization of parts [10]. To ease the burden of precise HRC workcell design while seeking the goals of flexibility and better robot–human interaction of Industry 4.0/5.0, this work investigates proactive robot path planning. To plan proactively, human motion must be predicted. Then a robot path planner must account for the predicted human motions when determining trajectories for the robot. Human motions include time dependency, requiring the robot path planner to perform additional computations to account for timing. Additionally, robot motion should be updated in real-time to accommodate human motion that deviates from the predictions. This is a challenge since computation must be minimized to allow fast updates.

The STAP-PPF framework, herein, provides a solution to the mentioned challenges of coordinating robot and human motion in HRC. STAP-PPF extends the human trajectory prediction algorithm in Ref. [11]. STAP-PPF uses that prediction algorithm to generate multi-step and multi-arm predictions. STAP-PPF also extends the STAP robot path planning method from Ref. [12]. STAP-PPF encodes input human motion predictions in a spatio-temporal avoidance model that indicates time intervals over which robot passage is blocked by predicted human occupancy of 3D points. STAP-PPF determines optimal robot paths and estimates robot delay using a time-based cost function. Sources of robot delay are the time intervals of occupancy and speed reduction enforced by a safety controller. The safety controller employs the speed and separation monitoring (SSM) robot control mode defined in ISO15066 to ensure human safety in HRC [6,13]. To improve performance, STAP-PPF approximates the spatio-temporal avoidance model with a neural network (NN), pre-samples robot configurations for path planning, and utilizes batch sampling to take advantage of computer hardware acceleration. STAP-PPF also adapts the robot's path in real-time based on continuously updated human predictions and estimated SSM effect. Hence, STAP-PPF accommodates deviations in real-time human motion.

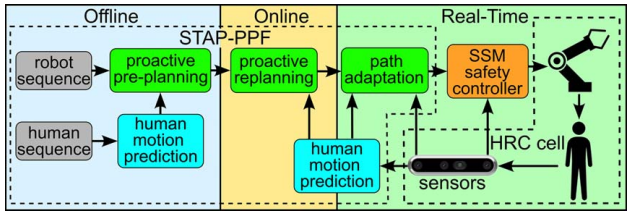

Fig. 1 HRC workcell control scheme

The novelties of STAP-PPF are a framework that incorporates multi-step human motion prediction, the planning method that selects optimal robot paths based on human motion predictions and anticipated SSM effect, and method to adapt the robot's path in real-time based on updated predictions. A goal of STAP-PPF is to generate robot paths that allow the robot to maintain efficiency amidst human proximity, contributing to the I4.0 goal of workcell flexibility. Additionally, STAP-PPF seeks to generate robot paths that improve human safety and comfort in an HRC workcell, contributing to the I5.0 goal of improving human-machine interaction and the S5.0 goal of improved human-technology relationships. Figure 1 shows how STAP-PPF is incorporated into the control loop for an HRC workcell. In offline planning, STAP-PPF can pre-plan robot motions considering the robot and human sequences. Online planning can consider updated human predictions to re-plan the robot's nominal motion for an upcoming robot motion segment. The real-time block includes computation that can occur faster than 30 Hz. It includes the component of STAP-PPF that warps the robot's nominal path to account for continuously updated human predictions. It also includes the SSM enabled safety controller that ensures human safety. The rest of this work is organized as follows. Section 2: literature review, Sec. 3: methods, Sec. 4: experiments, Sec. 5: results and discussion, and Sec. 6: conclusions.

2 Literature Review

This section will first review methods for human motion prediction, which could serve as input to a proactive robot path planning method. The review explores existing methods that predict human arm motion in an HRC workcell and methods for predicting general human motion not limited to manufacturing settings. Additionally, the review of human motion prediction is organized into review of prediction methods that model the human with a dynamic model first. Next, prediction methods that use sets of probabilistic models to generate prediction of human motion are reviewed. The final type of prediction methods reviewed estimate probability of humans occupying workcell volumes at any point in a task.

Next, this section reviews methods for planning robot paths considering human proximity to the robot in an HRC setting, referred to as human-aware robot path planners. The review of path planning methods considers two types of path planners for an HRC workcell. The first type of planners are those that proactively consider predicted human motion to generate robot paths. The second type of planners are those that react to real-time human poses. This type either pre-plans a set of robot paths to switch between in real-time as the human moves or plans robot motion considering the SSM effect the current human pose would have on robot speed throughout a path.

2.1 Human Motion Prediction. Some prior works model human motion using dynamic models. Liu and Wang predict a human's trajectory to reach a goal by estimating parameters of a linear human dynamic model [14]. Wang et al. fit an autoregressive integrated moving average model to recorded human arm motions to predict a future sequence of human joint angles [15]. Liu and Liu predict human motion using a recurrent neural network (RNN), with a modified Kalman filter to adapt RNN weights as motion is observed [16]. Martinez et al. utilized a sequence-to-sequence RNN with gated recurrent units to predict a sequence of human motion based on recently observed motion [17]. Li et al. utilized an encoder/decoder network to predict human pose sequences considering long- and short-term history of recent motion as input [18]. Mao et al. modeled human motion with the discrete Cosine transform (DCT) and predicted future DCT coefficients with a graph convolutional neural network, considering recent sequences of DCT coefficients as input [19]. These methods iteratively predict motion at a next time-step based on the prediction for the current time-step. Therefore, error builds exponentially as the prediction iterates farther into the future. This limited many prior works to an error reporting horizon of up to 1 s, due to the exponential growth of error [17–19].

Other prior works developed sets of human motion models. Mainprice and Berenson used Gaussian mixture models (GMMs) to predict human occupancy in the HRC workspace [20]. Kanazawa et al. used GMMs to estimate the pose of humans in a workcell [21]. Then they used Gaussian mixture regression (GMR) to predict human trajectories. Li et al. recorded human trajectories and applied multi-step Gaussian process regression to the previous observations to predict human reaching motions [22]. Callens et al. applied probabilistic principal component analysis to observe human motion to develop a database of human motion models and detect motion onset and speed [23]. Each of these methods generates a set of models based on previously observed human motions. When predicting human motion, the model best suited to the human's current task, based on a sequence of recent motion, must be determined from among the models in the set of models. A disadvantage of these predictors is that many time-steps of observed motion may be required before the most appropriate model can be determined from the set of models.

Another strategy in prior works is to estimate the probability of occupying points in the robot's workcell. Pellegrinelli et al. modeled the probability of human occupancy based on time spent in workcell voxels throughout tasks and inferred likely human occupancy volumes (HOVs) [24]. Hayne et al. also determined a workspace voxel occupancy probability based on the number of time-steps the voxel is occupied [25]. These models provide a system with occupancy data so a robot can avoid the general area where a human will work, but do not encode the timing of occupancy. This can lead them to be too conservative in separating robot and human activity and less effective in tight human–robot collaboration.

Flowers and Wiens developed a generative neural network (GNN) to predict a time-sequence of human poses based on the human's current pose and reaching the target [11]. This GNN is the human motion predictor used in STAP-PPF. The inputs to the GNN are the wrist target for completing a reaching motion combined with the human's current pose. The GNN predicts a multi-step sequence of poses the human will take in going from the current pose to a final pose at which the reaching wrist is at the target.

2.2 Human-Aware Robot Path Planners. Recent works in robot motion planning have developed path planner variations to consider human proximity to the robot and find robot paths that balance human safety and comfort with efficiency. Some of these works consider human motion predictions in planning. Phillips et al. considered anticipated intervals of robot passage among humans, with application to a robotic vehicle considering humans as point obstacles [26]. Mainprice and Berenson used the human prediction GMMs and the STOMP path optimizer to adapt the robot's goal and corresponding trajectory to avoid anticipated human occupancy [20,27]. Zanchettin et al. used linear programming to determine robot paths that minimize robot speed reduction due to human proximity [28]. Hayne et al. determined robot and human lane penetration costs based on probability of workspace voxel occupancy and signed distance field (SDF) and used a trajectory optimizer to minimize both robot and human costs [25]. Pellegrinelli et al. applied RRT with HOVs in Ref. [24]. Liu and Tomizuka estimated a safety index for a predicted human trajectory and determined optimal, safe robot control sets [29]. Wang et al. generated robot trajectories that minimized time to reach a goal such that a minimum distance between the robot and human arm predictions is maintained [15]. Zhao and Pan developed a cost map that also considered prior human occupancy, SDF, and robot's penetration into anticipated human occupied regions [30]. Kanazawa et al. generated robot trajectories that optimized reaching the robot goal, avoiding collision with the predicted human trajectories generated by GMR, and maintaining robot kinematic limits [21].

Another strategy of recent works is to react to a human's current pose. Sisbot and Alami determined robot/human object transfer points and robot paths to optimize a balance between human safety and comfort and minimize the duration of robot–human handovers [31]. Holmes et al. developed the autonomous reachability-based manipulator trajectory design which used trajectory optimization to switch between pre-generated trajectories at run-time according to real-time obstacle positions [32]. Faroni et al. developed the human-aware motion planner (HAMP) to generate robot paths that minimize the SSM effect of human proximity on robot speed reduction [7]. The SSM robot speed control mode of ISO15066 dictates reduction in robot speed limit proportional to the distance between robot and human [6]. Tonola et al. developed a framework to use parallel processing to generate multiple paths via AnytimeRRT and switch paths during execution to accommodate human activity in HRC [33]. Since these works react to the human's current pose without anticipating future human poses, they can be nearsighted. They may react to a current pose only to have their paths blocked again once the human moves, possibly allowing the human to entrap the robot. STAP-PPF considers human motion predictions, so it does not suffer from this nearsighted behavior.

2.3 Human–Robot Collaboration Framework. Nicora et al. developed a framework that allows planning of robot activity for an HRC workcell at a motion segment level [34]. They defined a robot motion segment as either restrictive or non-restrictive. A restrictive segment only allows robot timing deviations, but not deviation in robot pose from the planned path. This restriction is desired when the robot is performing a short motion to place an object in its gripper or release an object into a fixture. A non-restrictive segment allows modifications in both robot pose and timing relative to the nominal path. This is desired for relatively long motions in which the robot is traversing workcell areas, either carrying a part or going to get a part. Non-restrictive segments are defined with a behavior dictating how the motion can be pre-planned and then modified online. STAP-PPF provides a new, proactive behavior type for non-restrictive segments in the framework started by Nicora et al. STAP-PPF can be used to pre-plan segments offline, re-plan segments online, or warp segments in real-time.

2.4 Novelties of STAP-PPF. One novelty of STAP-PPF is the extension of the GNN-based human motion prediction method in Ref. [11] into a framework permitting proactive robot path planning. The STAP-PPF human prediction GNN predicts all steps of motion in one forward pass of the network so it does not suffer from exponential increase in error as time horizon increases and has no limit on the time horizon for prediction as prior works do.

The STAP-PPF human prediction GNN can also update predictions as fast as 500 Hz, permitting real-time updates to human predictions, which is not possible for prior works using sets of models. STAP-PPF also contains a novel, time-optimal, sampling-based robot path planner. It estimates robot arrival time based on estimated time intervals over which the predicted human blocks robot passage and estimated SSM effect of predicted human proximity on robot speed. Some prior works considered SSM effect of current human poses, but not for human predictions as in STAP-PPF. Prior path planners that considered human predictions did not consider HRC safety standards such as the ISO15066 SSM robot control mode. The STAP-PPF planner is also novel in its adaptation of RRT* which considers estimated robot connection durations and propagates them along robot paths. This is in contrast to prior human prediction-based path planners that used a robot path optimizer instead of sampling-based planner. STAP-PPF also presamples robot configurations along the human prediction and uses batch-sampling to improve planner convergence. Finally, STAP-PPF develops a method for real-time adaptation of the robot's path according to real-time updates to the human motion prediction. Many prior methods do not include real-time updates to human predictions or the robot's path. STAP-PPF also considers a more complex human prediction and higher degrees-of-freedom robot than some other works.

3 Methods

The methods of STAP-PPF are divided into three parts. First, STAP-PPF predicts nominal human motions with the multi-step human motion prediction part of the framework. Then the proactive robot path planner in STAP-PPF determines time-optimal, nominal robot paths given spatio-temporal constraints imposed by the human motion prediction. Next, STAP-PPF includes an algorithm to warp the nominal robot path according to continuously updated human motion predictions.

3.1 Human Motion Prediction. STAP-PPF utilizes the human motion prediction method developed in Ref. [11]. It predicts a sequence of human poses that a human will take in reaching toward a Cartesian target. The input to that prediction method is a feature vector consisting of the Cartesian reaching target and the quaternion representation of the human's current pose. This subsection will first describe the human pose represented as pelvis location and human link quaternions. Then a summary of the prediction method from Ref. [11] is provided. Finally, the inclusion of the prediction method into STAP-PPF to provide multi-step sequences of human trajectories is presented in this subsection.

3.1.1 Human Pose Representation. Herein, human pose is represented as a pelvis location and the set of quaternions relating each human link to the world z-axis. The human pose is given by

$$\mathbf{h} = \left[P_p, q_t, q_n, q_s, q_{uL}, q_{fL}, q_{uR}, q_{fR}\right]^T \tag{1}$$

The P_p is the pelvis position (x, y, z) in the world coordinate frame in meters. The $q_t \cdots q_{fR}$ are the human link quaternions for the torso, neck, shoulders, left upper arm, left forearm, right upper arm, and right forearm. The quaternions in the human pose define a rotation angle and vector that would align the world frame z-axis with the vector along the human link. The quaternions are defined by unitless w, x, y, z parameters, where the w element determines the rotation angle and the x, y, and z elements determine the rotation vector. The quaternion elements are bounded between ± 1 and the quaternions must be unitized for representing rotations. Figure 2(*a*) shows the pelvis location and human links of the upper body. The human pose representation given by $\mathbf{h}$ combined with the human link lengths provides the line segments for human links. Human link radii are required to convert the link line segments to cylinders for each link, which provide volumes for collision avoidance.

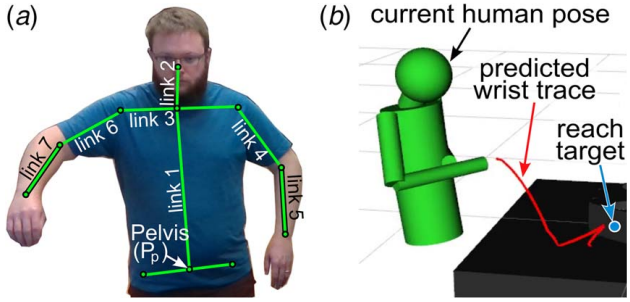

Fig. 2 (*a*) Human represented by pelvis position and link quaternions and (*b*) right wrist trace for a motion prediction

Nominal human link lengths are determined once a human is present in the field of view of depth cameras that monitor the HRC workcell, using the procedure in Ref. [35]. Once the nominal link lengths are determined, they are held constant until the human leaves the cameras' field of view.

3.1.2 Human Motion Predictor. STAP-PPF uses the human motion prediction GNN in Ref. [11] to predict a human's trajectory when reaching for a part or tool. The input to the predictor is the stacked vector of the reach target (e.g., location of part to grab) and current human pose in the format discussed in Sec. 3.1.1. The reach target is the desired location of the reaching wrist when the reach is complete. The input to the GNN is

$$\mathbf{z_h} = \left[P_{tgt}, \mathbf{h_s}\right]^T \tag{2}$$

where $\mathbf{h_s}$ is the human's pose at the start of the reach motion. The P_{tgt} is the Cartesian position of the human wrist at the end of the reaching motion, known as the "reach target" herein, defined in the world coordinate frame in meters. Examples of the P_{tgt} are the location of a part to pick up or the location of a workpiece to place a part on. The $\mathbf{h_s}$ has 31 elements as described in Sec. 3.1.1. Therefore, $\mathbf{z_h}$ has 34 elements. While $\mathbf{z_h}$ has mixed units, the GNN developed in Ref. [11] was tolerant to mixed unit input and generated mixed unit outputs. This work assumes the reach target is known based on objects the human will interact with in the nominal sequence. Location of objects can be predefined for a manufacturing sequence, or a computer vision algorithm can segment images and identify locations of parts in the workcell, such as that developed by Papadaki et al. [36].

The GNN in Ref. [11] is a sequence of convolution transpose layers with the scaled exponential linear unit (SELU) nonlinear activation function applied between layers [37]. Each convolution transpose layer uses a convolution kernel in combination with stride and padding to expand an input with relatively few elements into an output with more elements. The GNN takes the $\mathbf{z_h}$ vector as input and outputs a matrix having 10 rows and 31 columns. Each row in the output matrix is a phase step in the human motion prediction. Each column corresponds to an element of the human pose. Phase steps are evenly spaced on a scale of percent completion of the motion, but not time. The GNN simultaneously estimates the human poses when the human wrist is at the P_{tgt}, at the end of the prediction, and estimates the sequence of poses the human will take when traversing between the start pose and the estimated final pose. The start pose ($\mathbf{h_s}$) would be either the human's current pose, or the final pose from a preceding prediction, as described in the next subsection. The GNN outputs the most likely human pose sequence based on the 2800 recordings of human reaching motions used to train the GNN.

The method in Ref. [11] also developed a wrist speed versus reach distance quadratic regression model to convert the phase scale of poses to a time scale. The method predicts wrist speed, v_{est}, in meters per second as

$$v_{est} = -0.214d^2 + 0.659d - 0.0176 \tag{3}$$

where d is the distance between the starting wrist location and the reaching target. The steps in the phase scale become evenly spaced steps on a time scale ranging from zero to (d/v_{est}) seconds. Figure 2(*b*) shows the current human pose as cylinders and reaching target as a circle at the right side. The line is the anticipated trace of the wrist location as the human moves through the predicted sequence of poses to reach the target with the right wrist. The prediction is generated for the pose of the whole upper body, but for clarity only the wrist trace is shown. The human pose sequence generated by the GNN and then converted to a time scale by the velocity model had an L2 joint error of 7.6 cm and L2 link roll-pitch-yaw error of 0.301 radians relative to actual motion, averaged over all motions in the test set for validating the method. While the prediction error was larger than desired, the STAP-PPF planner creates a space buffer between the robot and the predicted human when the planner considers SSM effect due to robot/human proximity. This reduces the effect of human prediction error on the STAP-PPF generated robot path.

The human motion prediction GNN outputs a prediction of human motion given a single target the human is anticipated to reach for. When the human has multiple targets to choose from, i.e., there are multiple copies of the same part on the table, STAP-PPF can generate a human prediction and corresponding robot path for each target in offline planning. Online, when the human begins reaching for one of the parts, STAP-PPF can select one of the pre-planned robot paths according to which part the human reaches for, determined by an algorithm that estimates human sequence like [38]. Then STAP-PPF can adapt the robot path in real-time according to updated human predictions for the observed target.

3.1.3 Prediction for Sequences of Human–Robot Actions. The human motion predictor can be used in the STAP-PPF framework to predict motions for a sequence of human motions. Details of the human motions must be provided to the framework to know how to connect each human motion, shown in Table 1. The reach arm must be specified because the GNN uses separate networks for left and right arm motion predictions to improve accuracy. The "prior robot step" indicates a robot task number that must be completed before the human can proceed. A sequence of human motions can be planned sequentially until reaching a human motion that depends on the completion of a robot task. An example use case for this parameter is when the robot must place a part before the human can install something on the part. When generating a sequence of predicted human motions, the final step of one prediction becomes the input pose for the next prediction. The "start delay" parameter indicates a time delay between finishing one human motion and starting the next motion. In between predicting individual motions, the human pose would be held constant for an amount of time equal to the start delay before predicting the next motion. An example of when the start delay should be used is when a person tightens a screw with a screwdriver. In between bringing the screwdriver to the screw and removing the screwdriver from the screw, the person's hand will remain in the nearly the same position while tightening the screw, so the start delay would indicate the time the person is tightening the screw.

Table 1 Parameters for predicting a multi-step human motion sequence

Parameter	Description
Reach target	Position of part/tool to grab
Reach arm	Arm performing the reach (left or right)
Start delay	Seconds between the completion of previous predicted motion and start of this prediction
Prior robot step	Robot task that must be completed before starting this human motion

In addition to specifying the human sequence, the robot's sequence parameters may also require an index of a human step at which the robot must wait for human task completion before continuing. An example is when the human must move an assembly out of its fixture so the robot can place a part in the fixture.

3.2 Proactive Robot Path Planning. This subsection first presents the spatio-temporal human avoidance model developed as part of STAP in Ref. [12]. It determines intervals of time over which robot passage through Cartesian points is blocked by anticipated human poses. Then an NN approximation of the spatio-temporal human avoidance model is developed to improve computation time. Finally, the method for planning time-optimal robot paths considering the spatio-temporal human avoidance model is presented in this subsection.

3.2.1 Ground Truth Spatio-Temporal Avoidance Model. The original spatio-temporal avoidance model from our prior work will be summarized first [12]. It will generate the training data for the NN avoidance interval estimator. To generate the avoidance model, the time sequence of human poses is first fit to a discretized 3D grid. Cylinders are generated for each link of the human for the entire input human pose sequence based on the pose **h** at each time-step and the link lengths and radii. Those link cylinders are then fit into the discretized 3D grid. Figure 3 shows the steps to determine the human's 3D occupancy, showing the sensed pose at left, cylinders in the middle, and point cloud at the right.

Next, the ith time interval of occupancy, known herein as an avoidance interval, anticipated by the oth human for every 3D point in the robot's workspace (denoted $\mathcal{W}$) is found by

$$\mathcal{A}_{o_i}(x, y, z) = \begin{cases} \left[t_{s_{o_i}}, t_{f_{o_i}}\right] & t_{f_{o_i}} < t_{end} \\ \left[t_{s_{o_i}}, \infty\right) & t_{f_{o_i}} = t_{end} \end{cases} \tag{4}$$

The $t_{s_{o_i}}$ and $t_{f_{o_i}}$ are the start and end times, respectively, for the ith interval of continuous occupancy for the oth human to occupy point (x, y, z). A human could potentially occupy the same Cartesian point over multiple intervals. For example, a human may occupy a point while reaching for a part and then occupy the point a second time when retrieving the part. The t_{f_o} is the last time the oth human occupies point (x, y, z) and t_{end} is the final time-step of the predicted human sequence. Therefore, if the oth human occupies point (x, y, z) at the end of the input predicted sequence, then it must be assumed that the human occupies the point for the rest of

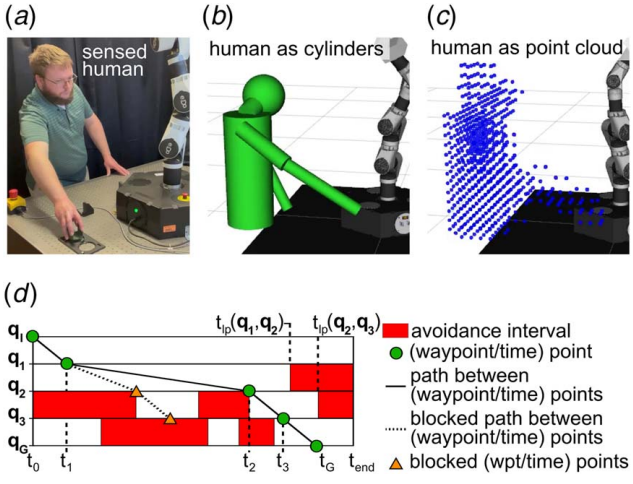

Fig. 3 (*a*) Actual human, (*b*) human generalized to cylinders, (*c*) human cylinders fit to 3D grid, and (*d*) robot path considering avoidance intervals

time. The intervals for each human are then combined

$$\mathcal{A}(x, y, z) = \bigcup_{o} \bigcup_{i} \mathcal{A}_{o_i}(x, y, z) \quad \forall\, i \in o, \quad \forall\, o \in \mathcal{W} \tag{5}$$

A time of last passage (t_{lp_o}) for point (x, y, z) due to the oth human is given by

$$t_{lp_o}(x, y, z) = \begin{cases} \infty & t_{f_o} < t_{end} \\ t_{s_o} & t_{f_o} = t_{end} \end{cases} \tag{6}$$

Then the time of last passage (t_{lp}) for point (x, y, z) due to all humans is

$$t_{lp}(x, y, z) = \min_{o \in (x,y,z)} t_{lp_o}(x, y, z) \tag{7}$$

While planning a robot path, a connection from an existing node, denoted $\mathbf{q_p}$, to a new node, denoted $\mathbf{q_c}$, inherits the avoidance intervals of Cartesian points the robot passes through between $\mathbf{q_p}$ and $\mathbf{q_c}$. The avoidance intervals for a connection are denoted $\mathcal{A}(\mathbf{q_p}, \mathbf{q_c})$. The connection also has a time of last passage, denoted $t_{lp}(\mathbf{q_p}, \mathbf{q_c})$, which is the minimum time of last passage from any point the robot passes through between $\mathbf{q_p}$ and $\mathbf{q_c}$.

Figure 3(*d*) illustrates the concept of avoidance intervals. The vertical axis is a sequence of robot configurations starting at $\mathbf{q_I}$ and ending at $\mathbf{q_G}$. The horizontal axis indicates the time each waypoint in the sequence is reached, relative to the sequence start time. The red rectangles indicate avoidance intervals when configurations are blocked by human occupancy. The green circles and solid black line indicate the (waypoint, timing) sequence connecting start to goal. The orange triangles and dotted lines indicate connections from $\mathbf{q_1}$ to $\mathbf{q_3}$ occurring too soon and not permitting connection to the goal due to avoidance intervals. The earliest timing for the connections from $\mathbf{q_1}$ to $\mathbf{q_3}$ is indicated by the green circles. In Fig. 3(*d*), red rectangles at the right side indicate avoidance intervals that persist at the end of the human prediction, which lead to the time of last passage for the $\mathbf{q_1}$ to $\mathbf{q_3}$ connections (top of the Fig. 3(*d*)).

3.2.2 Approximation of the Spatio-Temporal Avoidance Model. Now, the $\mathcal{A}(\mathbf{q_p}, \mathbf{q_c})$ computed by Eq. (5) for a given input human prediction ($\mathbf{h}$) can be used to train an NN to estimate avoidance intervals for a robot connection from $\mathbf{q_p}$ and $\mathbf{q_c}$. The network selected for this method estimates the probability of intersection, denoted P_{int}, between a robot along a path connection and a single human pose. To evaluate the intersection of the robot for a predicted human pose sequence, the network estimates probability of intersection for all time-steps in the predicted human sequence with batch processing. The input to the NN is the vector

$$\mathbf{z_{ai_t}} = \left[\mathbf{q_p}, \mathbf{q_c}, \mathbf{h_t}\right]^T \tag{8}$$

where $\mathbf{h_t}$ is the human pose at time-step t of the input predicted sequence. For a robot having n_{dof} degrees-of-freedom (DOF), $\mathbf{z_{ai_t}}$ has $2n_{dof}+31$ elements, since $\mathbf{h_t}$ has 31 elements.

Through trial and error, an NN consisting of a sequence of five fully connected layers having sizes 43×512, 512×512, 512×256, 256×256, and 256×1 going from input to output was found to work best for this application. Increasing the number of network layers or the size of network layers beyond this point reduced prediction error with diminishing returns and added computation time. The avoidance interval NN architecture is depicted in Fig. 4(*a*), with $\mathbf{z_{ai_t}}$ at the left and the P_{int} in the middle. Between each fully connected layer, the SELU non-linearity was applied to each layer output [37]. The sigmoid non-linearity function was applied to the output of the final fully connected layer to scale the output for probability of intersection between the robot connection and a single human pose. Figure 4(*a*) shows the layers as rectangles with number of neurons per layer and SELU activation between layers below and above the rectangles, respectively. The binary cross entropy (BCE) loss function determined the loss between the output probability and the true probability of

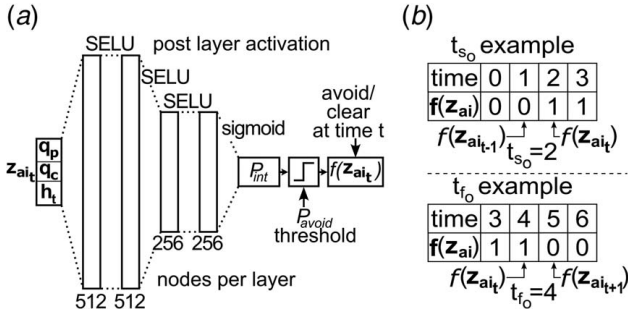

Fig. 4 (*a*) Architecture of the neural network that approximates avoidance intervals and (*b*) example of interval start/end times based on estimated avoid/clear flags

intersection (1 if avoid, 0 if clear). The BCE was used in training network parameters, as it is commonly used for classification networks like herein this section. The ADAM optimizer was used to backpropagate the BCE loss via gradient descent to adjust network parameters [39]. In NN training, a dropout rate of 0.5 was used between each SELU operation and the subsequent fully connected layer to improve the robustness of the network.

A robot connection is flagged to be avoided at time t if the P_{int} output by the network is greater than a user-set probability threshold, denoted P_{avoid} herein, depicted by the left 2 blocks in Fig. 4(*a*). Otherwise, the connection is clear at time t. To estimate the avoid/clear status for a robot connection and all time-steps of the predicted human motion, a batch of inputs is constructed from $\mathbf{z_{ai_t}}$ for all time-steps of the predicted human sequence. Then the output of the network is a vector having a probability of intersection (avoid) for each time-step, denoted $\mathbf{P_{int}}(\mathbf{z_{ai}})$. Next, the probabilities are thresholded to get the avoid/clear flag for each time-step, denoted $f(\mathbf{z_{ai_t}})$ at time t. Finally, avoidance intervals are determined based on the avoid/clear flags. Interval start times (t_{s_o}) occur where $f(\mathbf{z_{ai_{t-1}}})$ is flagged as clear (0) and $f(\mathbf{z_{ai_t}})$ is flagged as avoid (1), as depicted at the top of Fig. 4(*b*). Interval end times (t_{f_o}) occur where $f(\mathbf{z_{ai_t}})$ is flagged as avoid (1) and $f(\mathbf{z_{ai_{t+1}}})$ is flagged as clear (0), as depicted at the bottom of Fig. 4(*b*). If $f(\mathbf{z_{ai_t}})$ at the final time-step of the human prediction is flagged as avoid, then t_{lp} is the preceding t_{s_o} and infinity otherwise.

To generate data to train and test the avoidance interval NN, STAP-PPF planned robot paths in a simulated workcell using the ground-truth spatio-temporal avoidance model considering human motion predicted by the GNN for a variety of targets. While planning a path, when STAP-PPF considers robot travel from $\mathbf{q_p}$ to $\mathbf{q_c}$, the ground-truth spatio-temporal avoidance model generates and saves the avoidance intervals for the connection ($\mathcal{A}(\mathbf{q_p}, \mathbf{q_c})$) considering the input human prediction. Each set of avoidance intervals generated by a $\mathbf{q_p}$, $\mathbf{q_c}$ pair is used to generate a number of feature vectors ($\mathbf{z_{ai_t}}$) for input to the NN equal to the number of time-steps in the human motion prediction. Each of those $\mathbf{z_{ai_t}}$ would contain the $\mathbf{q_p}$, $\mathbf{q_c}$, and human pose for a time-step ($\mathbf{h_t}$). If the time-step of $\mathbf{h_t}$ was within an avoidance interval in $\mathcal{A}(\mathbf{q_p}, \mathbf{q_c})$ generated by the ground-truth spatio-temporal avoidance model, then $\mathbf{z_{ai_t}}$ would be labeled with $P_{int}(\mathbf{z_{ai_t}}) = 1$. Otherwise, $\mathbf{z_{ai_t}}$ was labeled with $P_{int}(\mathbf{z_{ai_t}}) = 0$. The results of training the avoidance interval NN are discussed in Sec. 5.1.

3.2.3 Planning Time-Optimal Robot Paths Considering Human Predictions

Safety-Aware Time-Based Cost Function. The robot path planner in STAP-PPF uses the same safety aware cost function included in STAP [12]. When evaluating a connection from $\mathbf{q_p}$ to $\mathbf{q_c}$, the cost function first determines the minimum time for the robot to travel from $\mathbf{q_p}$ to $\mathbf{q_c}$

$$t(\mathbf{q_p}, \mathbf{q_c}) = \max_{i \in [1, n_{dof}]} \left| \frac{\mathbf{q_c}[i] - \mathbf{q_p}[i]}{\bar{\dot{\mathbf{q}}}[i]} \right| \tag{9}$$

The $\overline{\dot{\mathbf{q}}}$ is the vector of maximum absolute joint velocities for the robot. Each node in the planner's node graph (denoted $\mathcal{G}$) stores the earliest time $\mathbf{q_p}$ can be reached via the most optimal path, denoted t_{arr_p}. The minimum time to reach node $\mathbf{q_c}$ through the connection from $\mathbf{q_p}$ to $\mathbf{q_c}$ is

$$t_c = t_p + t(\mathbf{q_p}, \mathbf{q_c}) \tag{10}$$

where t_p is the time the robot would leave $\mathbf{q_p}$ and t_c is the time the robot would arrive at $\mathbf{q_c}$ via this connection. The t_p is initially considered to be t_{arr_p}.

Next, the effect of the SSM speed limit enforced by the safety controller is applied to the connection timing, according to Ref. [13]. The safety controller enforces the speed limit

$$v_{\max}(\mathbf{q}, i, j, t_h) = \begin{cases} -a_s T_r - v_h + \sqrt{v_h^2 + (a_s T_r)^2 + 2a_s D_{ij}} & D_{ij} > \underline{D} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

The a_s and T_r are positive, non-zero parameters for maximum Cartesian deceleration of the robot and reaction time of the robot, respectively. The v_h is the velocity of the human in the direction of the robot and the $\underline{D}$ is the minimum distance between robot and human that permits robot motion. The vector from the jth point on the robot at configuration $\mathbf{q}$ to the ith point on the human at time t_h in the human prediction is denoted $\mathbf{V_{ij}}$. Herein, minimal sets of points (i and j) along the robot body and human upper body are used to sufficiently represent each entity. The distance D_{ij} is the Euclidean norm of vector $\mathbf{V_{ij}}$. The robot's speed in the direction of the human for the connection from $\mathbf{q_p}$ to $\mathbf{q_c}$ based on the nominal time from Eq. (9) is

$$v_{robot}(\mathbf{q}, i, j, t_h) = \max\left(\boldsymbol{J}_j(\mathbf{q}) \frac{\mathbf{q_c} - \mathbf{q_p}}{t(\mathbf{q_p}, \mathbf{q_c})} \cdot \frac{\mathbf{V_{ij}}}{D_{ij}}, 0\right) \tag{12}$$

The $\boldsymbol{J}_j(\mathbf{q})$ is the robot's translational Jacobian for point j on the body of the robot at configuration $\mathbf{q}$. The Jacobian is computed as $\left[\frac{\partial \boldsymbol{FK}_j(\mathbf{q})}{\partial \mathbf{q}}\right]$, where $\boldsymbol{FK}_j(\mathbf{q})$ is the position of point j determined from the robot forward kinematics. The SSM enabled safety controller would enforce a speed reduction to a fraction of planned speed, i.e., the v_{lim} velocity ratio

$$v_{lim}(\mathbf{q}, i, j, t_h) = \min\left(\frac{v_{\max}(\mathbf{q}, i, j, t_h)}{v_{robot}(\mathbf{q}, i, j, t_h)}, 1\right) \tag{13}$$

Equations (11)–(13) may cause robot speed to be reduced if the robot is approaching the human, but don't result in speed reduction if the robot is moving away from the human. Next, the planner updates $t(\mathbf{q_p}, \mathbf{q_c})$ by summing effects of SSM at configurations between $\mathbf{q_p}$ and $\mathbf{q_c}$, as follows:

$$t(\mathbf{q_p}, \mathbf{q_c}) = \sum_{\mathbf{q}=\mathbf{q_p}}^{\mathbf{q_c}} \max_{i,j} \frac{\|\mathbf{dq}\| t(\mathbf{q_p}, \mathbf{q_c})}{v_{lim}(\mathbf{q}, i, j, t_h) \|\mathbf{q_c} - \mathbf{q_p}\|} \tag{14}$$

The connection from $\mathbf{q_p}$ to $\mathbf{q_c}$ is interpolated to get intermediate configurations spaced $\mathbf{dq}$ apart. The time to complete each intermediate $\mathbf{dq}$ is summed to estimate the $t(\mathbf{q_p}, \mathbf{q_c})$ based on the speed reduction the SSM safety controller would enforce.

If v_{lim} for a $\mathbf{q}$ in the summation of Eq. (14) is computed to be less than a user set threshold, then a time window (Δt) into the future human motion prediction is considered to determine if the speed reduction is temporary

$$v_{lim}(\mathbf{q}, i, j, t_h) = \max_{t_s \in [t_h, t_h + \Delta t]} v_{lim}(\mathbf{q}, i, j, t_s) \tag{15}$$

The new $v_{lim}(\mathbf{q}, i, j, t_h)$ computed by Eq. (15) is used to recompute the summation element for $\mathbf{q}$ in Eq. (14). For implementation, the Δt was set to 3 s because most human reaching motions observed in Ref. [11] were less than 3 s long. This means that if Eq. (13) resulted in a low v_{lim} due to human proximity at time t_h, then Eq. (15) can consider enough time-steps to complete one predicted human motion. Smaller t_h could lead to STAP-PPF anticipating a low v_{lim} for some connections resulting in long path duration when in reality the human obstruction is temporary. Longer t_h leads to excessive computation time.

If the passage interval $[t_p, t_c]$ after consideration of avoidance intervals and SSM effects intersects an avoidance interval in $\mathcal{A}(\mathbf{q_p}, \mathbf{q_c})$, then t_p is updated to be the end of that avoidance interval. Then t_c is updated again according to Eqs. (10)–(14) to account for effect of SSM speed reduction and delay of t_c. Then $[t_p, t_c]$ must be checked for intersection with avoidance intervals again. After iteratively checking time interval intersections and SSM effects, if the resulting passage interval $[t_p, t_c]$ contains $t_{lp}(\mathbf{q_p}, \mathbf{q_c})$, then the connection from $\mathbf{q_p}$ to $\mathbf{q_c}$ is considered blocked indefinitely and cannot be a waypoint in the robot's path. The cost of connection from $\mathbf{q_p}$ to $\mathbf{q_c}$ used to determine a time-optimal robot path is t_c.

Spatio-Temporal Robot Path Planner. The STAP method from our prior work included an adaptation of optimal rapidly exploring random trees (RRT*) [12,40]. The adaptations in STAP permitted consideration of the time-based cost function, which lends a more complex implementation than the traditional shortest-path planning problem. The STAP planner is intended for use with serial manipulators having many DOF. Therefore, the STAP-PPF planner must utilize random sampling, such as in RRT*, to ensure exploration of the robot's workspace. The optimality of RRT* is achieved via a node rewiring step. In STAP-PPF, the cost of a connection from $\mathbf{q_p}$ to $\mathbf{q_c}$ in the planner's node graph ($\mathcal{G}$) is t_c (robot time of arrival at $\mathbf{q_c}$) resulting from Eq. (10). When a new node ($\mathbf{q_{new}}$) is added to $\mathcal{G}$ via the lowest cost connection from $\mathbf{q_{near}}$ to $\mathbf{q_{new}}$, the rewiring step evaluates costs of connections from $\mathbf{q_{new}}$ to each node ($\mathbf{q_{near}}$) in the neighborhood of $\mathbf{q_{new}}$. In considering a connection from $\mathbf{q_{new}}$ to $\mathbf{q_{near}}$, $\mathbf{q_{new}}$ is considered $\mathbf{q_p}$ and $\mathbf{q_{near}}$ is considered $\mathbf{q_c}$ for Eqs. (9)–(15). If a connection from $\mathbf{q_{new}}$ to $\mathbf{q_{near}}$ has lower cost than the current connection to $\mathbf{q_{near}}$, then rewiring assigns $\mathbf{q_{new}}$ as the parent to $\mathbf{q_{near}}$.

In STAP, when a rewiring event occurs, then the time to reach $\mathbf{q_{near}}$ is reduced. STAP must also consider if timing for connections for which $\mathbf{q_{near}}$ is the parent can be improved. The timing of a number of connections stemming from $\mathbf{q_{near}}$ is checked, up to a user set number of child connections, to see if timing has been improved by rewiring. It is computationally intractable to check for connection timing improvements all the way to the end of the branches of $\mathcal{G}$. When a new node is added to $\mathcal{G}$, all $\mathbf{q_{near}}$ in the neighborhood of $\mathbf{q_{new}}$ are also checked to see if any other node in the neighborhood of $\mathbf{q_{new}}$ could be a lower cost parent to $\mathbf{q_{near}}$. This ensures that as planning iterations progress, each node in $\mathcal{G}$ has an opportunity to improve its timing if possible while keeping computation tractable.

To improve the performance of STAP-PPF path planning, herein, two additional adaptations are made. First, a number of pre-samples of robot configurations are added to the planner's node graph ahead of planning. The pre-sample configurations position the end-effector near human points throughout the human motion prediction. An artificial potential field (APF) like approach is used to rapidly generate robot poses that place the end-effector near desired points. Selection of a pre-sample starts at a default robot pose ($\mathbf{q_0}$) and then iteratively changes the configuration according to

$$\mathbf{q_{n+1}} = \mathbf{q_n} - \alpha \boldsymbol{J_{ee}}(\mathbf{q_n})^T \left(h_i(t_h) - \boldsymbol{FK_{ee}}(\mathbf{q_n})\right) \tag{16}$$

The α is a user-set gain. The $\boldsymbol{J_{ee}}(\mathbf{q_n})$ is the robot's translational Jacobian for the tip of the end-effector at configuration $\mathbf{q_n}$ after iteration n. The $h_i(t_h)$ is ith human point (e.g., wrist) on the human at time-step t_h in the predicted human motion. The $\boldsymbol{FK_{ee}}(\mathbf{q_n})$ is the position of the end-effector determined by robot forward kinematics at configuration $\mathbf{q_n}$. The pre-samples do not need a specific

end-effector orientation since they will be seeds for future configuration samples during planning iterations. In generating one pre-sample, the configuration $\mathbf{q_{n+1}}$ is updated for 200 iterations and $\mathbf{q_{200}}$ is taken as the pre-sample. Pre-samples are generated until a user-set number of pre-samples (N_{ps}) is reached.

The second adaptation for STAP-PPF is batch sampling. When using a graphics processing unit (GPU) to accelerate NN inference, as is the case with the avoidance intervals NN, time required to transfer data between the GPU and CPU can be minimized by moving larger volumes of data per transfer. Therefore, large batches of input connection data for the avoidance intervals network are used. In consideration of a single connection to a new node ($\mathbf{q_{new}}$), a batch will include input vectors $\mathbf{z_{ai_t}}$ for all steps of the human sequence, which will be denoted as the set Z_c. Then to consider all connections between $\mathbf{q_{new}}$ and neighboring nodes ($\mathbf{q_{near}}$), a batch would be a larger set containing all Z_c for each node pair, denoted $Z_{c_{new}}$. To construct an even larger batch, within a single planner iteration a user-set number of random $\mathbf{q_{new}}$ are generated. Now a batch could be the set of $Z_{c_{new}}$ for each $\mathbf{q_{new}}$, denoted Z_{mult}.

After the avoidance interval network infers avoidance intervals and last pass times for all inputs in Z_{mult}, best parent nodes for each $\mathbf{q_{new}}$ can be determined using the STAP-PPF cost function. Additionally, rewiring from each $\mathbf{q_{new}}$ can seek to improve timing to reach $\mathbf{q_{near}}$ in the neighborhood of each $\mathbf{q_{new}}$. The time-optimal sequence of waypoints, or path, is that which minimizes the t_c for the connection in which $\mathbf{q_c}$ is the path goal. The planner in STAP-PPF is programmed as a MoveIt! motion planning plugin [41]. It uses MoveIt!'s collision checking functions to ensure the entire robot arm is free of collision with objects in the workcell along every connection in $\mathcal{G}$. The iterative parabolic time parameterization (IPTP) in MoveIt! is applied to the sequence of waypoints output by STAP-PPF to assign times to reach each waypoint and robot joint velocities while passing through each waypoint. This ensures the path that is commanded of the robot will have a smooth position profile and that robot velocities and accelerations along the path stay within the robot's limits. The two adaptations to STAP included in STAP-PPF made it 15 times faster than the original STAP method at generating robot paths. However, STAP-PPF still requires 5 s of computation time using six cores of an Intel i9 CPU and NVidia RTX-1070 GPU. Therefore, STAP-PPF needs a faster component, defined in the next subsection, for real-time updates to the trajectories generated by the robot path planner to account for deviations in human motion.

3.3 Real-Time Adaption of Robot Paths. STAP-PPF can generate real-time updates to trajectories generated by the STAP-PPF planner. Herein, real-time is defined as the frequency new data that are generated by depth cameras in the workcell sensor suite, which is 30 Hz. Since the STAP-PPF planner takes multiple seconds to compute a path, it can't generate an update in real-time. APF like approaches can adapt robot poses with relatively low computation time compared to sampling-based path planning. Therefore, STAP-PPF uses an APF like approach to warp the path generated by the planner according to updated human motion predictions. The human motion predictor in STAP-PPF can update as fast as 500 Hz, providing frequent input for warping the robot path. Figure 5 illustrates STAP-PPF real-time path warping. It shows the human's current pose as green cylinders and the pose predicted 5 s into the future as the red cylinders.

In the process of warping the nominal robot path, the path is first interpolated into a number of intermediate robot configurations between each path waypoint. The intermediate configurations have the same spacing as in Eq. (14). In Fig. 5, the nominal path and waypoints are the solid green line and red circles, respectively. The intermediate configurations are iterated through from the current pose to the goal node. The effect of the SSM safety controller is estimated at each configuration based on the updated human prediction, estimated by a repulsive torque that yields

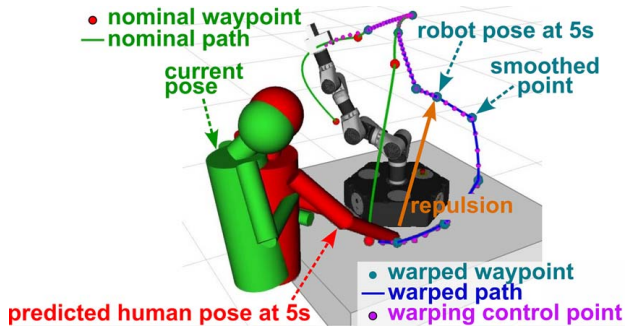

Fig. 5 Warping of the nominal robot path according to predicted human proximity and SSM effects

$$\Delta\mathbf{q}_{\mathbf{r}_{(\mathbf{k},n)}} = \beta_r \sum_{i,j} \left(1 - v_{lim}\left(\mathbf{q}_{\mathbf{k}_n}, i, j, t_h\right)\right) \boldsymbol{J}_j(\mathbf{q}_{\mathbf{k}_n})^T \frac{\mathbf{V_{ij}}}{D_{ij}} \tag{17}$$

The $\Delta\mathbf{q}_{\mathbf{r}_{(\mathbf{k},n)}}$ is the repulsive effect after warping iteration n due to the sum of SSM effects of all ith human points on all jth robot points at the kth configuration along the path. The β_r is the user-set repulsion gain. The $v_{lim}\left(\mathbf{q}_{\mathbf{k}_n}, i, j, t_h\right)$ is the speed reduction fraction computed according to Eq. (13), based on the location of the ith human point at time-step t_h of the updated human prediction relative to the jth robot point. When warping the path, the updated human motion prediction starts from the human's current pose, so t_h is zero for the current pose at $k = 0$. If a point along the nominal path is repelled by Eq. (17), tests showed it is beneficial to repel a number of proceeding points in the nominal path to generate a new path that is smooth. Therefore, points following the point repelled by Eq. (17) are repelled according to

$$\Delta\mathbf{q}_{\mathbf{s}_{(\mathbf{k},l,n)}} = \left(\frac{\max(N_{smooth} - (k - l), 0)}{N_{smooth}}\right) \Delta\mathbf{q}_{\mathbf{r}_{(\mathbf{l},n)}} \tag{18}$$

where l is the index of the last point on the path that was repelled by Eq. (17) and N_{smooth} is a user-set parameter. Hence, points experience diminishing repulsion as they become farther from the point originally repelled. Larger N_{smooth} prevents sharp changes in path direction, but may lead to a trajectory of longer duration.

An attractive torque also pulls points along the robot path toward the direct path. The attractive torque yields

$$\Delta\mathbf{q}_{\mathbf{a}_{(\mathbf{k},n)}} = \beta_a\left(\mathbf{q_d} - \mathbf{q}_{\mathbf{k}_n}\right) \tag{19}$$

where $\mathbf{q_d}$ is the robot configuration nearest $\mathbf{q}_{\mathbf{k}_n}$ along the straight line path between the current robot configuration and goal configuration and β_a is a gain. This attraction is necessary to pull a previously repelled robot configuration back toward an optimal configuration if the human prediction no longer requires that it be repelled. When this occurs, the human motion has deviated from prediction so the nominal path computed by the planner is likely no longer optimal. Therefore, the path is attracted to the straight-line path instead as the best estimate of the current optimal path.

After computing repulsive and attractive effects, the configuration at step k along the robot's path is updated according to

$$\mathbf{q}_{\mathbf{k}_{n+1}} = \mathbf{q}_{\mathbf{k}_n} + \Delta\mathbf{q}_{\mathbf{a}_{(\mathbf{k},n)}} - \Delta\mathbf{q}_{\mathbf{r}_{(\mathbf{k},n)}} - \Delta\mathbf{q}_{\mathbf{s}_{(\mathbf{k},l,n)}} \tag{20}$$

where subscript n indicates the iteration number. If MoveIt! collision and joint limit checking functions find that $\mathbf{q}_{\mathbf{k}_{n+1}}$ is outside the robot's joint limits or causes collision between any point on the robot and any object in the workcell, then $\mathbf{q}_{\mathbf{k}_{n+1}}$ is held at $\mathbf{q}_{\mathbf{k}_n}$ from the previous iteration [41]. During each path update, a user-set number of warp iterations were performed, denoted N_{warp}. At the end of each path update, configurations in the set of all $\mathbf{q_k}$ were selected to be waypoints for the resulting robot path if

they met the following criteria:

$$\left|(\mathbf{q_k} - \mathbf{q_{k-1}}) \cdot (\mathbf{q_{k-1}} - \mathbf{q_{k-2}})\right| < c_{tol} \tag{21}$$

This prevents selection of intermediate $\mathbf{q_k}$ along a nearly straight path which could be represented simply with two waypoints. The selected waypoints must also be a user-set distance (d_{start}) away from the current robot configuration and a user-set distance (d_{tol}) away from the previously selected waypoint. These restrictions prevent selection of a $\mathbf{q_k}$ very close to the current robot pose, which may require a significant speed reduction to pass through $\mathbf{q_k}$. The d_{tol} prevents selection of $\mathbf{q_k}$ so close together that the robot must significantly reduce speed to pass through all selected $\mathbf{q_k}$. After selecting the waypoints from the warped path, IPTP is applied to the resulting waypoint sequence to assign timing and robot velocities and accelerations to each waypoint so the updated robot trajectory has a smooth position profile and does not violate robot joint velocity and acceleration limits.

In Fig. 5, the intermediate configurations between waypoints after repulsion and attraction are shown as the smaller pink spheres. The new warped robot path is defined by the larger blue circles, which were selected from the pink circles using the mentioned criteria. Figure 5 delineates part of the warped path caused by smoothing after a repulsion. Equations (17)–(21) are performed during real-time robot path execution at a user-set rate (F_{warp}). Continuous warp updates cause intermediate robot path points to be repelled by human predictions until the predictions no longer cause anticipated robot speed reduction or until the robot passes the point of repulsion. Higher values of N_{warp} can generate robot paths with more waypoints and sharper turns.

4 Experiments

To validate the STAP-PPF, experiments were conducted in a real, live HRC workcell, shown in Fig. 6. The cell has a 6DOF Comau e.Do serial robot manipulator sitting on a table [42]. The table has a workspace that is shared between a human and the robot, indicated by the box labeled "Shared Workspace" at the bottom of Fig. 6. The workcell sensor suite contains two ZED2 depth cameras, which are circled in Fig. 6. The ZED skeleton tracking was used to localize human joints [43]. The skeleton fusion method in Ref. [35] combined the data from both cameras to obtain the human pose.

Experiments were conducted using the STAP-PPF planner, with and without the path warping feature, and with three baseline human-aware planners. The STAP-PPF planner without path warping is denoted "STAP-C" for "constant" in results. The STAP-PPF using path warping is denoted "STAP-W" for "warping" in results. Two baseline methods are based on the HAMP [7]. This planner utilizes a similar time-based cost function as STAP-PPF,

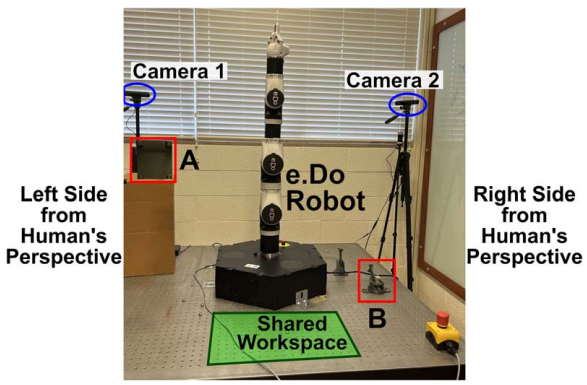

Fig. 6 Experimental HRC workcell for testing, showing robot, two depth cameras, shared workspace, and targets for scenarios A and B

but considers human occupancy as static. In one version of HAMP, denoted "HAMP-S" in results, the planner considers the current, static pose of humans, making it reactive to current human pose. During tests, HAMP-S replanned the robot's trajectory when the robot was either within 0.2 m of the human or the SSM safety controller reduced robot speed to 35% of planned speed due to robot/human proximity. When HAMP-S replanned, it only considered the SSM effect on robot speed due to the current human pose. The second version of HAMP, denoted "HAMP-P" in results, proactively pre-plans robot motion by considering HOVs. The HOVs estimate probability of human occupancy of each workcell voxel based on observation of human occupied workcell voxels over task iterations. The human motion prediction generated by the method of Sec. 3.1 was used to pre-train the HOV probabilities ahead of tests. During tests, HOV probabilities were updated based on real-time human pose. The third baseline method, denoted "STOMP" in results, uses the STOMP trajectory optimizer as described in Ref. [20] to pre-plan robot motion by warping the straight line robot path to minimize robot penetration into the human motion prediction. It also used the human motion prediction method from Sec. 3.1. Due to high computation time of STOMP, it could only pre-plan robot motion before starting a task. During the task, the robot stayed on the STOMP trajectory while the SSM safety controller slowed or stopped the robot due to human proximity.

In one set of tests, called "test case 1" herein, the human tried to replicate the predicted human motion as closely as possible. This allows comparison of the proactive planning capabilities of STAP-PPF and baseline methods. In another set of tests, called "test case 2" herein, the human started motion either 2 s ahead or behind the predicted motion. These tests will show how STAP-PPF can adapt the robot's path in real-time to accommodate human deviation. In these two sets of tests, the robot and human perform one of three scenarios.

In scenario A, the human gets a part off a shelf at the left of the workcell, indicated by the box labeled "A" in Fig. 6. Then, the human puts the part down in the shared workspace. Meanwhile, the robot moves a part from the right side of the cell to the left side. In scenario B, the human reaches to the right side of the cell to get a part from the table, indicated by the box labeled "B" in Fig. 6. Then the human places the part in the shared workspace, waits 5 s, and returns the part to the right side of the cell. The robot moves a part from the left side of the cell to the right side during scenario B. In scenario C, the human separates the right arm up and left arm down to about 45 deg from horizontal and then returns his/her arms to horizontal. This motion sequence repeats twice, taking 5 s per motion. Meanwhile, the robot moves a part from left to right across the cell. Scenarios A and B emulate more realistic activities in an HRC workcell. Scenario C is used to create a narrow time-varying window of passage around the human's arms. Figure 7 shows the start, end, and two intermediate time-steps of scenarios A, B, and C.

In a third set of tests, called "test case 3" herein, the human and robot collaborate to assemble two piston and connecting rod sets. Figure 8 shows the layout of the test cell for test case 3. It shows the location of the assembly pieces before assembly in the labeled green boxes. It also shows the location of the fixture where the pieces are assembled. The robot performs 26 motion segments and the human performs 21 motions. The person picks up a connecting rod, shown in Fig. 9(*a*), and brings it back near the assembly area while the robot brings a piston to the assembly fixture, shown in Figs. 9(*b*) and 9(*c*). When the piston is placed, the person puts the connecting rod in the assembly fixture with the right hand, picks up a pin with the left hand, and places the pin through the piston and connecting rod. Then the person takes the piston/rod assembly out of the fixture and places it on the table. The sequence repeats again for the second assembly. After the robot places the second piston in the fixture, the robot picks the crankshaft, as shown in Fig. 9(*d*), and places it in the shared workspace to complete the sequence.

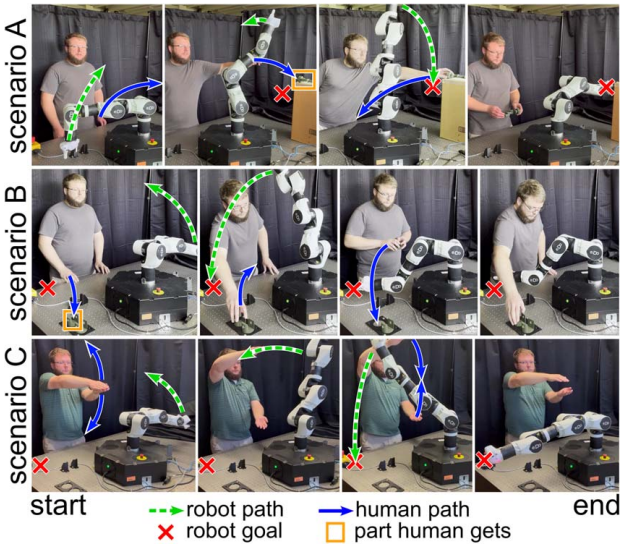

Fig. 7 Scenarios used for test cases 1 and 2

Test case 3 demonstrates prediction for a multi-step sequence of human motions and further shows the benefit of coordinating robot and anticipated human motion. Test case 3 is performed with STAP-PPF with path warping (STAP-W), HAMP-S, and HAMP-P. When using HAMP-S and HAMP-P, the robot re-planned its path as necessary. Experiments with test case 2 showed the STOMP based method is not suited for the complexity of test case 3. When using STAP-PPF, human motion for the multi-step human sequence is predicted first. Then STAP-PPF considers the mutli-step prediction in pre-planning each robot motion. For each robot motion, STAP-PPF estimates the robot path duration based on the prediction. Then the next robot motion in the sequence is planned considering the predicted human sequence starting from the estimated completion time of the prior robot motion. STAP-PPF and the HAMP methods were used for the eight non-restrictive robot motions in the sequence, as defined in Sec. 2.3. For the short, restrictive robot segments, such as going from a perch position over the piston to having the gripper around the piston, the motion was planned as linear in joint configuration space.

Throughout all tests, the safety controller in the robot control system enforced speed reduction according to robot–human proximity. The speed was reduced to a fraction of full speed according to Eqs. (11)–(13), but only considering the human's current pose, not the predicted pose. For the experimental workcell, the parameters for computing the speed limit were $\underline{D}$, T_r, a_s of 0.2 m, 0.15 s, and 0.1 m/s^2, respectively. The $\underline{D}$ was selected based on desired human comfort and sensor suite inaccuracy and T_r are a_s are based on specifications of the robot. Parameters of STAP-PPF were set at N_{ps} of 300 samples, d_{start} of 0.8 radians, c_{tol} of 0.99, d_{tol} of 0.3 radians, N_{smooth} of 2, β_a of 0.0001, β_r of 0.0067, N_{warp} of 2, selected through trial and error. Observations forming the basis of these selections showed that lower values of d_{start}, c_{tol}, or d_{tol} caused the real-time path warping of STAP-PPF to place too many waypoints in the robot's path. This caused the robot to reduce speed so as to pass through all waypoints. Higher values of those parameters diminished the capability of STAP-PPF warping. Smaller β_a and β_r led STAP-PPF to adjust the robot path too slowly to accommodate human deviations. Larger values led STAP-PPF to warp the robots path too severely, leading to excessively long robot paths. Smaller N_{smooth} led to sharper corners in the warped robot path while larger N_{smooth} minimized robot acceleration changes at the expense of excessive robot path duration. Higher values of N_{warp} allowed the path to be warped around the human prediction more accurately, but increasing N_{warp} beyond 2 did not result in better human avoidance. STAP-PPF generated less than 300 pre-samples given most human predictions. Hence, N_{ps} was set at 300 to utilize as many pre-samples as possible, but limited to prevent high computation time in case predictions permitted more pre-samples as they are dependent on the duration the human spends in the robot's workspace. The F_{warp} was selected to be 30 Hz to permit robot path updates as fast as the sensor suite cameras provided new data. The computer of the experimental cell, which processed all camera data, generated predictions, and planned robot paths, has an Intel i9 CPU and NVidia RTX-1070 GPU.

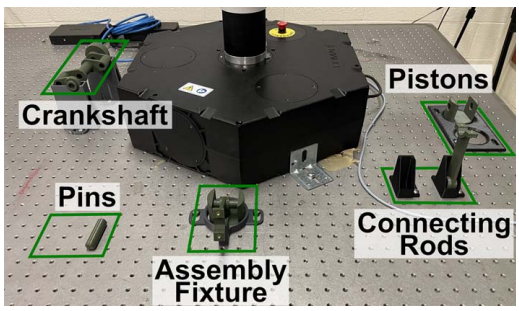

Fig. 8 Layout of the HRC workcell for test case 3

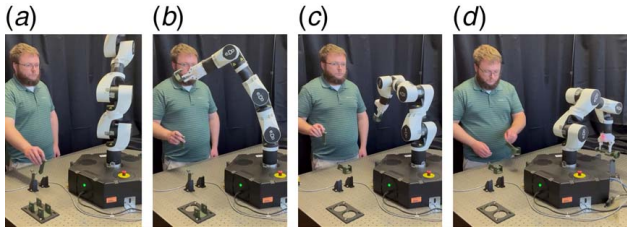

Fig. 9 (*a*) Person retrieves the connecting rod, (*b*) robot moves piston to the fixture, (*c*) robot lowers piston into fixture, and (*d*) robot moves crankshaft to shared workspace while human removes assembly from fixture

5 Results and Discussion

5.1 Avoidance Interval Neural Network Training. To train and validate the avoidance interval NN, first, a set of over 58 million samples was generated from simulations in which STAP-PPF used the ground-truth avoidance model to generate samples using the procedure at the end of Sec. 3.2.2.

Once the dataset of samples was amassed, it was divided into an 80%/20% split for training and testing, respectively. The NN was fit to the samples in the training set. In training, 100 epochs were performed. In each epoch, every training sample was evaluated by the network. Then the BCE loss was evaluated and backpropagated through the network layers to adjust network parameters, using the ADAM optimizer with a learning rate of 0.0005.

The samples from the test set were evaluated to show that the network inferred avoid/clear status with an accuracy of 95.5% with a probability threshold (P_{avoid}) of 0.5. Accuracy mean and standard deviation from five-fold cross-validation were 95.4% and 0.11%, respectively, showing consistency of network training with varying train/test data. The network accuracy was also evaluated at other P_{avoid} over the range [0.1, 0.9], shown in Fig. 10(*a*). The duration of robot trajectories while the human–robot team performed test scenario B was averaged over 10 samples at each of the P_{avoid} levels, shown in Fig. 10(*b*). These figures show a P_{avoid} of 0.5 generates the most accurate avoid/clear status and shortest duration robot trajectories. Larger P_{avoid} make STAP-PPF less conservative in avoiding human predictions, leading to less coordinated human–robot interaction (HRI). Smaller P_{avoid} make STAP-PPF

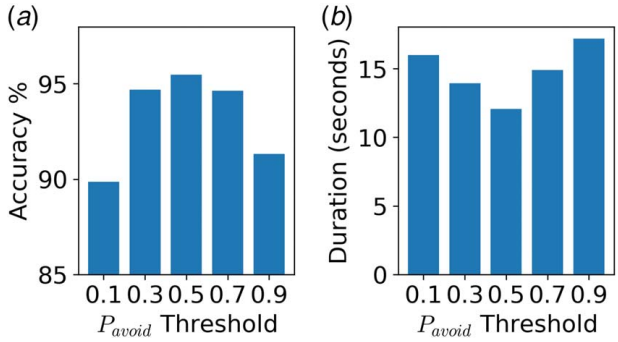

Fig. 10 (*a*) Accuracy of the avoidance interval neural network and (*b*) duration of robot trajectories using a range of probability thresholds

excessively conservative in avoiding humans, leading to longer path duration. Therefore, selection of P_{avoid} of 0.5 was substantiated.

The mentioned computer took less than 4 h to generate train/test samples and train the avoidance interval NN. Therefore, when the STAP-PPF is applied to a new workcell, less than 4 hours of preparation are needed for STAP-PPF to be operational.

5.2 STAP-PPF Benchmarking. To validate STAP-PPF, its performance was compared with the performance of the competing benchmarking methods while performing test cases 1 and 2. Test case 1 demonstrates each planner's ability to plan for the human performing the nominal human motion. Test case 2 demonstrates each planner's robustness to deviations in real-time human motion relative to the prediction. One metric for comparison is the duration of the robot's trajectory to go from the start pose to the goal pose for each test while the human performed the corresponding motions. Another metric discussed is the robot/human separation distance averaged over each test.

5.2.1 Robot Trajectory Durations. Figure 11 shows the trajectory durations averaged over 20 repetitions each of test case 1 (nominal human timing) and test case 2 (perturbed human timing) with each scenario. The larger bars in Fig. 11 and the upper numbers over the bars indicate the sample mean durations. The hatched bars correspond to STAP-W and STAP-C. The solid bars correspond to the other methods. The thin bars at the top of the large bars and the number in parenthesis over the bars indicate the sample standard deviation. The column for nominal human timing in Fig. 11 shows STAP-W and STAP-C generated robot trajectories that required less execution time than the other methods, all subject to the dynamic experimental scenarios. STAP-PPF path warping enabled the shortest robot trajectory durations for test scenarios B and C with nominal human timing. Even when the human attempted to closely follow nominal timing, slight deviation occurred in the live tests. Path warping allowed STAP-PPF to accommodate these deviations and reduce robot path duration, reducing duration by at least 8% relative to STAP-C and at least 14% relative to others.

When human timing was perturbed, Fig. 11 shows STAP-W and STAP-C still generate shorter robot trajectory durations than other methods. STAP-W generated the lowest robot trajectory durations for all three scenarios. Additionally, the STAP-W robot trajectory duration results were closest to those with nominal human timing. These observations indicate path warping in STAP-PPF mitigated the effect of human timing deviations. Figure 11 also shows that STAP-W and STAP-C robot trajectories resulted in smaller standard deviation in duration, meaning STAP-PPF results in more consistent robot trajectories.

Independent sample *t*-tests were used to generate 95% confidence intervals (CIs) for difference in population mean robot trajectory durations among pairs of methods. The *t*-tests show if the conclusions made on the collected samples also apply to the population or only to collected samples. The collected samples may be too few in number or have too great a standard deviation to show significant differences among methods considering the population of samples. In Fig. 12, the bars indicate the lower and upper limits of the CIs for difference in population mean trajectory durations in seconds. Hatched bars and solid bars correspond to the comparisons between STAP-W and STAP-C and comparison of STAP-PPF (STAP-W or STAP-C) to the benchmarking methods, respectively. The thin lines in the middle of each CI bar indicate the sample mean difference in trajectory durations. The dotted horizontal lines at zero are shown for reference.

Figure 12 shows STAP-W and STAP-C result in shorter population robot trajectory durations than the benchmarking methods at 95% confidence, based on the collected samples. This is indicated by all CIs being entirely below zero. The CIs permit the claim that STAP-PPF (STAP-W and STAP-C) generates robot trajectories of shorter duration relative to the benchmarking methods.

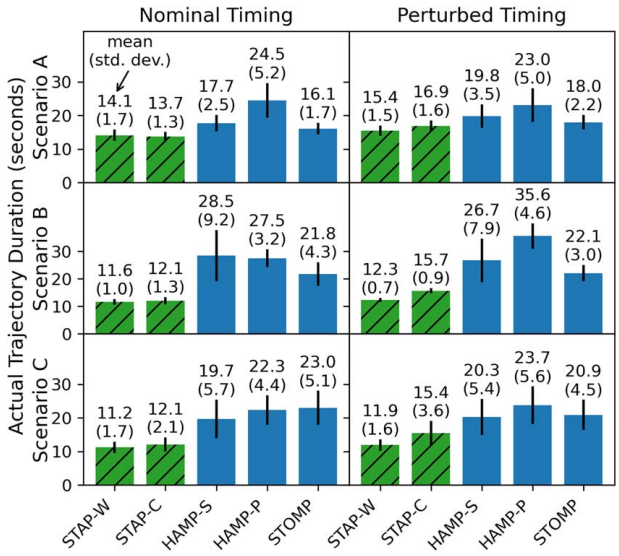

Fig. 11 Robot trajectory durations from test cases 1 and 2 with test scenarios A–C. Large bars indicate mean and small bars indicate standard deviation.

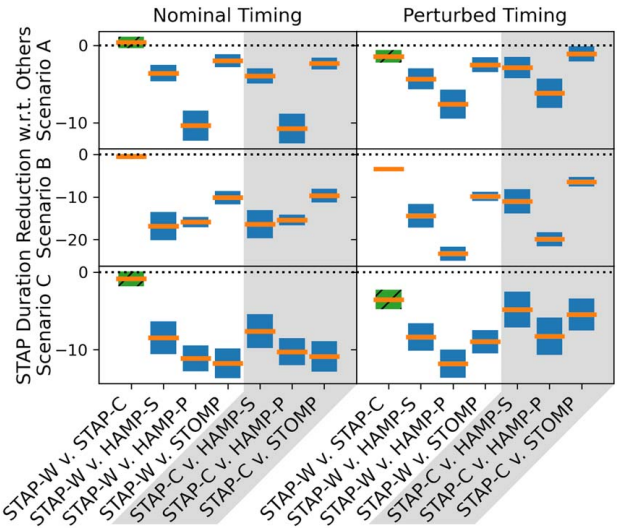

Fig. 12 Confidence intervals for STAP-PPF reduction in robot trajectory durations relative to other methods. Thin lines show difference in sample means and bars show upper and lower limits of confidence intervals.

Figure 12 also shows that the difference in robot trajectory durations with nominal human timing between STAP-W and STAP-C is insignificant at 95% confidence. This is expected since the human follows the prediction and STAP-W and STAP-C generate similar plans to avoid the nominal human prediction. However, when the real-time human timing is perturbed, STAP-W decreases robot trajectory durations at 95% significance relative to STAP-C.

The experiments also permit comparison of the error in trajectory duration estimates generated by each planning method. STAP-PPF robot trajectories took 30% more time to execute than estimated by the STAP-PPF planner. HAMP-S, HAMP-P, and STOMP generated robot trajectories that took 485%, 287%, and 248% more time to complete than each planner's estimate, respectively. These results show STAP-PPF estimates robot trajectory with much greater accuracy due to consideration of predicted human motion. The robot trajectory duration estimates generated by STAP-PPF could be used in task sequence optimization to select the next robot action that would minimize robot delay.

5.2.2 Robot–Human Separation Distance. The minimum robot–human separation distance was recorded at 30 Hz during experiments. It is the minimum distance between the robot shape and the real-time human pose generalized to cylinders. The separation distance metric is assumed to be directly proportional to the level of human comfort and safety in prior works [6,7,13,15,20,21,28,29]. Figure 13 shows the average separation distance for STAP-PPF and the benchmarking methods throughout all iterations of test cases 1 and 2. This figure shows STAP-W and STAP-C maintained at least 14% higher average separation between the robot and human than the benchmarking methods when the human closely followed nominal timing. When the human timing deviated significantly from nominal, STAP-W maintained at least 17% greater average robot/human separation distance than the other methods.

T-tests were also applied to the difference in robot/human sample separation distance between STAP-PPF and other methods, with CIs shown in Fig. 14. In Fig. 14, if a CI is entirely above zero meters, then the STAP-PPF method generates greater population mean robot/human separation distance than the comparison method with 95% confidence, based on the collected samples. The figure shows STAP-W maintained greater separation distance than other methods when the human timing was either nominal or perturbed.

5.2.3 Assembly Tests. The duration and robot-human separation distance were recorded for 20 iterations of test case 3, which was the multi-step assembly sequence, with the STAP-W (STAP-PPF), HAMP-S, and HAMP-P methods. The top left plot in Fig. 15 shows the sample mean and standard deviation of task durations with each method. The top right plot shows the 95% CIs for the difference in population mean task duration of STAP-W relative to the HAMP methods. These plots show STAP-W resulted in durations at least 17% shorter for the assembly task, relative to other methods, indicating other methods could not anticipate when a human would reach into the workspace.

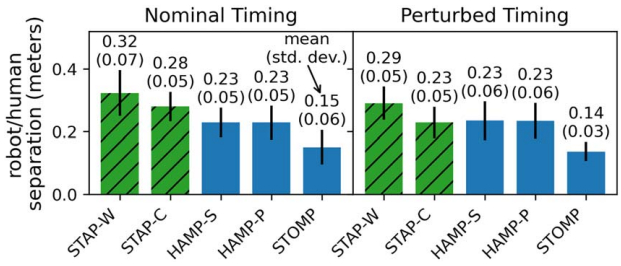

Fig. 13 Robot–human separation distances from test cases 1 and 2. Large bars indicate mean and small bars indicate standard deviation.

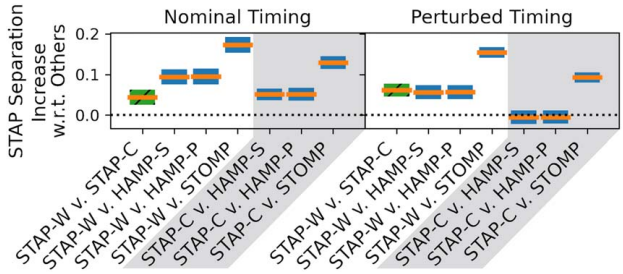

Fig. 14 Confidence intervals for STAP-PPF increase in robot/human separation distance relative to other methods. Thin lines show difference in sample means and bars show upper and lower limits of confidence intervals.

The bottom left plot in Fig. 15 shows the average robot/human separation distance throughout the assembly task for each method. The bottom right plot shows the 95% CIs for the difference in population mean separation distance of STAP-W relative to the HAMP methods. The sample mean separation differences were almost the same, within 4 cm, for the three methods and all greater than 49 cm. The CIs show separation distance was less with STAP-W relative to HAMP-P. However, since all three methods resulted in average separation of 49 cm, which is large relative to the workcell size, the CI showing difference between STAP-W and HAMP-P is not relevant. The separation distances with test cases 1 and 2 were smaller because they challenged the methods using less realistic motions. However, test case 3 is designed to demonstrate each planner's effectiveness in a realistic scenario.

5.3 Discussion. When conducting experiments, other observations became apparent that may have contributed to the differences in robot path durations and robot/human separation. It appeared that the human predictions for the three test scenarios were too complex for the STOMP based planner and HAMP-P planner because those planners generate a path directly through the predicted human poses. Inspection of the STOMP based method revealed complex human motions could prevent STOMP from finding an equilibrium robot path that avoided predictions. For example, a predicted human pose at one time-step repelled the robot up and then the predicted human pose at a later time-step repelled the robot downward, counteracting avoidance of the previous pose. Complexity of human motion for scenarios B and C led

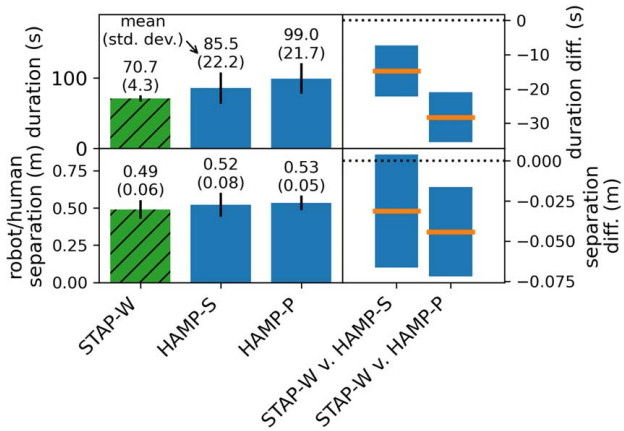

Fig. 15 Test case 3 task durations, robot-human separation distances, and 95% CIs for population mean of each. In the left plots, height of large bars indicates sample mean and the size of the thinner bars indicate standard deviation. In the right plots, thin lines show the sample means and bars indicate upper and lower CI limits.

HAMP-P to estimate high probability of occupancy for much of the workcell between the robot and human for the entire task. This led the cost function of HAMP-P to estimate that a direct trajectory through the human prediction was optimal since the human prediction couldn't be avoided. The HAMP-S method often reacted to a current human pose with a trajectory that deviated far from the human. If HAMP-S re-planned the robot path to avoid a real-time human pose and then the human backed away, HAMP-S would remain on the avoidance path rather than return to a more direct path. This led to a delay in robot trajectory completion with HAMP-S. Since STAP-PPF considered the timing of the human prediction in generating nominal robot trajectories, it was able to find paths around the human prediction that STOMP and HAMP-P could not find. STAP-PPF planned robot paths that accounted for times when the human approached the robot and times when the human backed away to find optimal paths, unlike HAMP-S.

For durations of robot path generated for scenario A compared to scenarios B and C, STAP-PPF yields greater improvements in reducing robot duration times as the human motion complexity within the task(s) increases. In other words, scenario A involved human motion of lower complexity compared to scenarios B and C. Scenarios B and C required the human to cross the robot's path multiple times while scenario A only had the human cross the robot's path once. The robot duration results showed that the reduction in path duration by using STAP-PPF with path warping relative to other methods was less with scenario A compared to scenarios B and C. Comparison of robot/human separation distance from test case 3 relative to test cases 1 and 2 also indicate that complexity of the HRC task may affect the utility of STAP-PPF for maintaining robot/human separation. The average robot/human separation for test case 3 was much higher than for test cases 1 and 2. This is because the sequence in test case 3 had more steps and took more time to complete. Additionally, not all actions in test case 3 required close robot/human interaction. Alternatively, if close robot/human interaction is not anticipated, then a different planning method with lower computation time could be used. Therefore, future research will develop a framework for selecting a path planner for each robot motion based on factors such as anticipated level of HRI and overlap of robot/human motions.

A limitation of STAP-PPF is computation time required to plan robot paths considering predicted human motion. Even though STAP-PPF computes 15 times faster than its predecessor, STAP, it still required 5 s of planning time to generate time-optimal paths. This necessitated the path warping feature of STAP-PPF, since the trajectory could not be regenerated at real-time speeds. In other words, STAP-PPF could update human predictions and warp the nominal path in real-time, considered to be at least the camera update rate of 30 Hz. However, time-optimality of the warped path could not be guaranteed. Computation time limits the amount of human deviation STAP-PPF can tolerate. Still, STAP-PPF can warp the robot path to accommodate deviations in human pose or timing in reaching for a target. But, if the human changes his/her sequence and reaches for a target not close to the anticipated target, then STAP-PPF path warping can't accommodate the large difference between real and anticipated human motion and the STAP-PPF planner can't re-plan before the human finishes the motion.

Ideally, STAP-PPF would regenerate the entire robot path at a real-time speed so the time-optimality of the robot path could be maintained. This would also allow STAP-PPF to handle a wider range of human variations. Future work can explore approximation of the SSM effect on robot speed with a neural network as well to reduce path planner computation time. Another improvement for future work would be consideration of the probability of robot/human intersection (P_{int}), output from the avoidance interval NN from Sec. 3.2.2, in determining cost of robot connections when planning. A challenge with including probability in STAP-PPF's planner is relating the probability of robot/human intersection to a time delay in the planner's cost function so STAP-PPF's estimate of path duration is not distorted.

6 Conclusions

This work presented the STAP-PPF framework for human motion prediction and robot path planning. STAP-PPF can be used in an HRC workcell to predict motions of humans reaching for objects, pre-plan nominal robot paths offline, re-plan upcoming robot motion segments as STAP-PPF updates human predictions, and warp robot paths in real-time to avoid updated human predictions. STAP-PPF pre-plans time-optimal robot paths considering the spatio-temporal constraints imposed by the human predictions and SSM enabled safety controller. Results show that STAP-PPF generated robot trajectories of at least 14% shorter duration relative to others, meaning less interrupted by real-time human activity. STAP-PPF also maintained at least 17% greater robot–human separation distance relative to others in tests designed to challenge the planners. Additionally, STAP-PPF estimated robot path execution times more accurately than other methods. Hence, STAP-PPF can be applied in an HRC workcell requiring close human–robot collaboration to mitigate production interruption and human discomfort. STAP-PPF robot execution time estimates can also be used to generate optimal robot sequences. Future work will investigate using STAP-PPF outputs in HRC sequence optimization.

Acknowledgment

Funding was provided by the NSF/NRI: INT: COLLAB:Manufacturing USA: Intelligent Human-Robot Collaboration for Smart Factory (Award ID #:1830383). Any opinions, findings, and conclusions or recommendations expressed are those of the researchers and do not necessarily reflect the views of the National Science Foundation.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

Nomenclature

$\underline{D}$ = min robot-human separation distance to permit robot motion
c_{tol} = dot product tolerance for connecting waypoints in the STAP-PPF warped path
n_{dof} = number of robot degrees-of-freedom
$\mathbf{h_s}$ = human pose at the start of a reach motion
$\mathbf{q_d}$ = robot configuration nearest $\mathbf{q_k}$ along the linear path in joint config. space
$\mathbf{q_{k_n}}$ = kth robot configuration in the nominal path after the nth iteration of path warping
$\mathbf{q_n}$ = robot configuration after the nth iteration of pre-sample adjustment
$\mathbf{q_{near}}$ = configuration in $\mathcal{G}$ in the neighborhood of $\mathbf{q_{new}}$
$\mathbf{q_{new}}$ = a new robot configuration sample
a_s, T_r = max abs. robot Cartesian deceleration and robot reaction time, respectively
$\mathcal{A}(x, y, z)$ = set of all avoidance intervals due to human occupancy of point (x, y, z)
$\mathcal{A}_{o_i}(x, y, z)$ = ith avoidance intervals due to the oth human occupying point (x, y, z)
$\mathcal{A}(\mathbf{q_p}, \mathbf{q_c})$ = set of all avoidance intervals for the robot connection from $\mathbf{q_p}$ to $\mathbf{q_c}$
$\mathbf{dq}$ = spacing of intermediate $\mathbf{q}$ between $\mathbf{q_p}$ and $\mathbf{q_c}$ for checking SSM effects

$\boldsymbol{FK_{ee}}(\mathbf{q})$ = position of the robot's end-effector found via forward kinematics at configuration $\mathbf{q}$

$\mathcal{G}$ = robot path planner's node graph

$\boldsymbol{J_j}(\boldsymbol{q})$, $\boldsymbol{J_{ee}}(\boldsymbol{q})$ = translational Jacobian of the jth point or end-effector of the robot, respectively, at cfg. $\mathbf{q}$

N_{smooth} = num. points to be smoothed by warping

P_p, q_x, $\mathbf{h}$ = human pelvis location, quaternion for link x, and pose vector for the upper body

P_{tgt} = human wrist target location for a reach

$\dot{\mathbf{q}}$ = vector of max speed for each robot joint

$\mathbf{q_p}$, $\mathbf{q_c}$ = robot configurations at the ends of a connection for travel from $\mathbf{q_p}$ (parent) to $\mathbf{q_c}$ (child)

Δt = time horizon for anticipating SSM effect due to future human poses

$t(\mathbf{q_p}, \mathbf{q_c})$ = estimated duration for robot travel from $\mathbf{q_p}$ to $\mathbf{q_c}$

t_{arr_p} = earliest time of robot arrival for $\mathbf{q_p}$

$t_{lp}(x, y, z)$ = last time robot passage is not blocked at point (x, y, z)

$t_{lp_o}(x, y, z)$ = last time robot passage is not blocked by the oth human occupying point (x, y, z)

$t_{lp}(\mathbf{q_p}, \mathbf{q_c})$ = last time robot passage is not blocked for the robot connection from $\mathbf{q_p}$ to $\mathbf{q_c}$

t_p, t_c = time when the robot should leave $\mathbf{q_p}$ and arrive at $\mathbf{q_c}$, respectively

$t_{s_{oi}}$, $t_{f_{oi}}$ = start and end time, respectively, of the ith avoidance interval for the oth human

v_{est}, d = estimated velocity of the human wrist for a reaching motion and distance between P_{tgt} and the initial wrist position, respectively

v_h = velocity of human toward the robot

$\mathbf{V_{ij}}$, D_{ij} = vector between the ith human point and jth robot point, and the distance between those points given by the Euclidean norm of $\mathbf{V_{ij}}$

$v_{lim}(\mathbf{q}, i, j, t_h)$ = robot speed reduction factor due to the velocity of the jth robot point at configuration $\mathbf{q}$ relative to the ith human point at time t_h of the human prediction

$v_{max}(\mathbf{q}, i, j, t_h)$ = maximum speed permitted by SSM considering the jth point on the robot at config. $\mathbf{q}$ and ith point on the human at time t_h

$v_{robot}(\mathbf{q}, i, j, t_h)$ = velocity of the jth robot point at config. $\mathbf{q}$ relative to the ith human point at time t_h of the human prediction

$\mathcal{W}$ = robot's workspace

$\mathbf{z_{ai_t}}$ = input vector for the avoidance interval NN considering step t of the human prediction

$\mathbf{z_h}$ = input vector for the human motion predictor

α = pre-sample adjustment gain

β_r, β_a = gains for repulsive and attractive warping

$\mathbf{\Delta q_{a_{(k,n)}}}$ = attraction for the kth config. along the robot's path after the nth iteration

$\mathbf{\Delta q_{r_{(k,n)}}}$ = repulsion due to robot and human proximity at the kth robot config. after the nth iteration

$\mathbf{\Delta q_{s_{(k,l,n)}}}$ = smoothing repulsion for the kth config. along the robot's path due to repulsion on the lth config. after the nth iteration

References

[1] Wang, L., Gao, R., Váncza, J., Krüger, J., Wang, X., Makris, S., and Chryssolouris, G., 2019, "Symbiotic Human-Robot Collaborative Assembly," CIRP Ann., **68**(2), pp. 701–726.

[2] Gao, Z., Wanyama, T., Singh, I., Gadhrri, A., and Schmidt, R., 2020, "From Industry 4.0 to Robotics 4.0 – A Conceptual Framework for Collaborative and Intelligent Robotic Systems," Proc. Manuf., **46**(1), pp. 591–599.

[3] Torn, I., and Vaneker, T., 2019, "Mass Personalization With Industry 4.0 by SMEs: A Concept for Collaborative Networks," Proc. Manuf., **28**(1), pp. 135–141.

[4] Ivanov, D., 2023, "The Industry 5.0 Framework: Viability-Based Integration of the Resilience, Sustainability, and Human-Centricity Perspectives," Int. J. Prod. Res., **61**(5), pp. 1683–1695.

[5] Mourtzis, D., Angelopoulos, J. D., and Panopoulos, N., 2022, "A Literature Review of the Challenges and Opportunities of the Transition From Industry 4.0 to Society 5.0," Energies, **15**(17), p. 6276.

[6] International Organization for Standardization, 2016, "ISO/TS 15066:2016 Robots and Robotic Devices – Collaborative Robots," Geneva, CH, Standard.

[7] Faroni, M., Beschi, M., and Pedrocchi, N., 2022, "Safety-Aware Time-Optimal Motion Planning With Uncertain Human State Estimation," IEEE Rob. Autom. Lett., **7**(4), pp. 12219–12226.

[8] Rubagotti, M., Tusseyeva, I., Baltabayeva, S., Summers, D., and Sandygulova, A., 2022, "Perceived Safety in Physical Human–Robot Interaction–A Survey," Rob. Auton. Syst., **151**(1), p. 104047.

[9] Pedersen, M. R., Nalpantidis, L., Andersen, R. S., Schou, C., Bøgh, S., Krüger, V., and Madsen, O., 2016, "Robot Skills for Manufacturing: From Concept to Industrial Deployment," Rob. Comput.-Integr. Manuf., **37**(1), pp. 282–291.

[10] Li, S., Zheng, P., Liu, S., Wang, Z., Wang, X. V., Zheng, L., and Wang, L., 2023, "Proactive Human–Robot Collaboration: Mutual-Cognitive, Predictable, and Self-Organising Perspectives," Rob. Comput.-Integr. Manuf., **81**(1), p. 102510.

[11] Flowers, J. T., and Wiens, G. J., 2023, "Prediction of Human Reaching Pose Sequences in Human Robot Collaboration," ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC), Boston, MA, Aug. 19–23.

[12] Flowers, J., Faroni, M., Wiens, G., and Pedrocchi, N., 2023, "Spatio-Temporal Avoidance of Predicted Occupancy in Human-Robot Collaboration," IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Busan, South Korea, Aug. 28–31.

[13] Faroni, M., Beschi, M., and Pedrocchi, N., 2019, "An MPC Framework for Online Motion Planning in Human-Robot Collaborative Tasks," IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, Sept. 10–13, pp. 1555–1558.

[14] Liu, H., and Wang, L., 2017, "Human Motion Prediction for Human-Robot Collaboration," J. Manuf. Syst., **44**(2), pp. 287–294.

[15] Wang, Y., Sheng, Y., Wang, J., and Zhang, W., 2018, "Optimal Collision-Free Robot Trajectory Generation Based on Time Series Prediction of Human Motion," IEEE Rob. Autom. Lett., **3**(1), pp. 226–233.

[16] Liu, R., and Liu, C., 2021, "Human Motion Prediction Using Adaptable Recurrent Neural Networks and Inverse Kinematics," IEEE Control Syst. Lett., **5**(5), pp. 1651–1656.

[17] Martinez, J., Black, M. J., and Romero, J., 2017, "On Human Motion Prediction Using Recurrent Neural Networks," IEEE Conference On Computer Vision and Pattern Recognition, Honolulu, HI, July 21–26.

[18] Li, C., Zhang, Z., Lee, W. S., and Lee, G. H., 2018, "Convolutional Sequence to Sequence Model for Human Dynamics," IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, June 18–22.

[19] Mao, W., Liu, M., Salzmann, M., and Li, H., 2019, "Learning Trajectory Dependencies for Human Motion Prediction," IEEE/CVF International Conference On Computer Vision, Seoul, South Korea, Oct. 27–Nov. 2.

[20] Mainprice, J., and Berenson, D., 2013, "Human-Robot Collaborative Manipulation Planning Using Early Prediction of Human Motion," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, Nov. 3–7, pp. 299–306.

[21] Kanazawa, A., Kinugawa, J., and Kosuge, K., 2019, "Adaptive Motion Planning for a Collaborative Robot Based on Prediction Uncertainty to Enhance Human Safety and Work Efficiency," IEEE Trans. Rob., **35**(4), pp. 817–832.

[22] Li, Q., Zhang, Z., You, Y., Mu, Y., and Feng, C., 2020, "Data Driven Models for Human Motion Prediction in Human-Robot Collaboration," IEEE Access, **8**(1), p. 227690.

[23] Callens, T., van der Have, T., Rossom, S. V., De Schutter, J., and Aertbeliën, E., 2020, "A Framework for Recognition and Prediction of Human Motions in Human-Robot Collaboration Using Probabilistic Motion Models," IEEE Rob. Autom. Lett., **5**(4), pp. 5151–5158.

[24] Pellegrinelli, S., Moro, F. L., Pedrocchi, N., Molinari Tosatti, L., and Tolio, T., 2016, "A Probabilistic Approach to Workspace Sharing for Human–Robot Cooperation in Assembly Tasks," CIRP Ann., **65**(1), pp. 57–60.

[25] Hayne, R., Luo, R., and Berenson, D., 2016, "Considering Avoidance and Consistency in Motion Planning for Human-Robot Manipulation in a Shared Workspace," IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, May 16–21, pp. 3948–3954.

[26] Phillips, M., and Likhachev, M., 2011, "SIPP: Safe Interval Path Planning for Dynamic Environments," IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9–13, pp. 5628–5635.

[27] Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S., 2011, "STOMP: Stochastic Trajectory Optimization for Motion Planning," IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9–13, pp. 4569–4574.

[28] Zanchettin, A. M., Ceriani, N. M., Rocco, P., Ding, H., and Matthias, B., 2016, "Safety in Human-Robot Collaborative Manufacturing Environments: Metrics and Control," IEEE Trans. Autom. Sci. Eng., **13**(2), pp. 882–893.

[29] Liu, C., and Tomizuka, M., 2016, "Algorithmic Safety Measures for Intelligent Industrial Co-Robots," IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, May 16–21, pp. 3095–3102.

[30] Zhao, X., and Pan, J., 2018, "Considering Human Behavior in Motion Planning for Smooth Human-Robot Collaboration in Close Proximity," IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Nanjing, China, Aug. 27–31, pp. 985–990.

[31] Sisbot, E. A., and Alami, R., 2012, "A Human-Aware Manipulation Planner," IEEE Trans. Rob., **28**(5), pp. 1045–1057.

[32] Holmes, P. D., Kousik, S., Zhang, B., Raz, D., Barbalata, C., Johnson-Roberson, M., and Vasudevan, R., 2020, "Reachable Sets for Safe, Real-Time Manipulator Trajectory Design," Robotics: Science and Systems, Corvalis, OR, July 12–16.

[33] Tonola, C., Faroni, M., Beschi, M., and Pedrocchi, N., 2023, "Anytime Informed Multi-Path Replanning Strategy for Complex Environments," IEEE Access, **11**(1), pp. 4105–4116.
[34] Nicora, M. L., Ambrosetti, R., Wiens, G. J., and Fassi, I., 2021, "Human–Robot Collaboration in Smart Manufacturing: Robot Reactive Behavior Intelligence," ASME J. Manuf. Sci. Eng., **143**(3), p. 031009.
[35] Flowers, J. T., and Wiens, G. J., 2022, "Comparison of Human Skeleton Trackers Paired With a Novel Skeleton Fusion Algorithm," ASME International Manufacturing Science and Engineering Conference (MSEC), West Lafayette, IN, June 27– July 1.
[36] Papadaki, A., and Pateraki, M., 2023, "6D Object Localization in Car-Assembly Industrial Environment," J. Imaging, **9**(3), pp. 72–94.
[37] Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S., 2017, "Self-Normalizing Neural Networks," Conference on Neural Information Processing Systems, Long Beach, CA, Dec. 4–9.
[38] Zhang, J., Wang, P., and Gao, R. X., 2021, "Hybrid Machine Learning for Human Action Recognition and Prediction in Assembly," Rob. Comput. Integr. Manuf., **72**(1), p. 102184.
[39] Kingma, D. P., and Ba, J., 2014, "Adam: A Method for Stochastic Optimization," International Conference on Learning Representations (ICLR), San Diego, CA, May 7–9.
[40] Karaman, S., and Frazzoli, E., 2011, "Sampling-Based Algorithms for Optimal Motion Planning," Int. J. Rob. Res., **30**(7), pp. 846–894.
[41] Coleman, D., Sucan, I. A., Chittat, S., and Correl, N., 2014, "Reducing the Barrier to Entry of Complex Robotic Software: A MoveIt! Case Study," J. Softw. Eng. Rob., **5**(1), pp. 3–16.
[42] e.Do Six Axes Technical Sheet, https://edo.cloud/wp-content/uploads/2021/06/edo-6-axes-technical-sheet.pdf, Accessed May 25, 2023.
[43] COZED Body Tracking, https://www.stereolabs.com/docs/body-tracking, Accessed May 25, 2023.