Proceedings of the ASME 2021 16th International Manufacturing Science and Engineering Conference
MSEC2021
June 21-25, 2021, Virtual, Online

MSEC2021-63670

COLLABORATIVE ROBOT RISK OF PASSAGE AMONG DYNAMIC OBSTACLES

Jared T. Flowers
University of Florida
Gainesville, FL

Gloria J. Wiens[1]
University of Florida
Gainesville, FL

ABSTRACT

Industry 4.0 projects ubiquitous collaborative robots in smart factories of the future, particularly in assembly and material handling. To ensure efficient and safe human-robot collaborative interactions, this paper presents a novel algorithm for estimating Risk of Passage (ROP) a robot incurs by passing between dynamic obstacles (humans, moving equipment, etc.). This paper posits that robot trajectory durations will be shorter and safer if the robot can react proactively to predicted collision between a robot and human worker before it occurs, compared to reacting when it is imminent. I.e., if the risk that obstacles may prohibit robot passage at a future time in the robot's trajectory is greater than a user defined risk limit, then an Obstacle Pair Volume (OPV), encompassing the obstacles at that time, is added to the planning scene. Results found from simulation show that an ROP algorithm can be trained in ~120 workcell cycles. Further, it is demonstrated that when a trained ROP algorithm introduces an OPV, trajectory durations are shorter compared to those avoiding obstacles without the introduction of an OPV. The use of ROP estimation with addition of OPV allows workcells to operate proactively smoother with shorter cycle times in the presence of unforeseen obstacles.

Keywords: human-robot collaboration, adaptive control, augmented intelligence, machine learning, smart factory, industry 4.0

NOMENCLATURE

$\arg\min_{x \in S} f(x)$	returns the minimizer x of the function f. The minimizer x belongs to the set S.
c_{Ois}	obstacle keypoint i; cartesian coordinates for the actual location of the centroid of obstacle i face s, $i \in \{1,2\}$, $s \in \{n, f\}$, n for nearest to origin or f for farthest from origin
$\bar{E}_{12s}$	vector between c_{O1s} and c_{O2s}, $s \in \{n, f\}$
k	count of consecutive workcell cycles executing the current task
l_{max}	maximum length of the OPV ($\|\bar{E}_1\|$ or $\|\bar{E}_2\|$) to consider
m	count of workcell cycles when the OPV prevented passage for the current task
$\min(a, b)$	returns the minimum value between a and b
$\max(a, b)$	returns the maximum value between a and b
mp_x	midpoint of vector x
n	number of robot joints
${}^{F}P_{ijk}$	vertex of the Obstacle Pair Volume. $i \in \{1,2\}$ for obstacle 1 or obstacle 2. $j \in \{n, f\}$ for near or far. $k \in \{1,2\}$ indicating which of the two points per obstacle face is considered, and F indicates relative to the fixed reference frame.
q_s	robot joint variables: angles (revolute joints) or joint offsets (prismatic joints) at configuration indicated by $s \in \{entry, exit, max\}$
$\dot{q}_{max}$	robot maximum joint velocities
r_i	radius of obstacle i, $i \in \{1,2\}$
r_{max}	maximum obstacle radius to consider
r_{robot}	minimum clearance between obstacles for robot passage
$sat_{a,b}(x)$	function that saturates x such that $a \leq x \leq b$.
$t_{delay_{max}}$	maximum duration to consider for calculating and executing a re-planned trajectory due to the OPV prohibiting passage
$t_{horizon}$	maximum duration into the future to consider when checking for collapse of the OPV
t_{replan}	duration of the robot backing up and replanning
t_{exec}	duration of the robot executing the new plan
$v_{c_{Ois}}$	current velocity of the point c_{Ois} in the fixed coordinate frame
$\|\bar{X}\|$	the Euclidean norm of vector $\bar{X}$

[1] Corresponding author: Gloria J. Wiens, University of Florida, gwiens@ufl.edu

v_s	robot end effector velocity at the time the robot enters or exits the OPV. $s \in \{entry, exit\}$
v_{OPV}	robot end effector velocity, either v_{entry} or v_{exit}, that had the smallest magnitude
δ_i	unknown weight for the i^{th} factor in the Risk of Passage, $i \in \{1, \dots, 8\}$
$\hat{\delta}_i$	estimate of the i^{th} factor in the Risk of Passage. $i \in \{1, \dots, 8\}$. $\hat{\delta}$ is a vector of all $\hat{\delta}_i$
Γ	positive definite gains matrix to control the rate of adaptation of $\hat{\delta}$
φ and $\hat{\varphi}$	actual and estimated risk of passage

1. INTRODUCTION

Industry 4.0 projects a rise in the use of collaborative robots in manufacturing settings leading to ubiquitous co-robots in smart factories of the future, particularly in applications such as assembly and material handling. One of the main challenges in Human-Robot Collaboration (HRC) is ensuring safe interaction with humans in real-time while also ensuring maximum task efficiency. To ensure safe interaction, the robot manipulator must be able to avoid collisions either by stopping motion or executing an alternative, collision-free trajectory. In manufacturing, stopping robot motion to wait for a possible collision situation to clear can add significant time to the workcell cycle. Therefore, an alternate trajectory, which may require real-time replanning, is needed to minimize cycle time delays. However, replanning may require selecting trajectories with rapid changes in robot velocity when an unforeseen collision becomes imminent. As found by Nicora et al. [1], short reaction distances generally resulted in increased execution times due to sharp turns the robot must make to avoid the collision, constrained by actuation acceleration limits. Hence, augmented intelligence (seamless integration of sensing, cognition and prediction into the robot controller) is critical for real-time awareness, response and communication inside a heterogeneous manufacturing cell (robots, humans, equipment).

Path planning in a 3D space has been a persistent challenge in robotics. Much research has been conducted in developing path planning algorithms to improve performance with robot manipulators [2]. However, as algorithms become more complex, computational demand increases. Another challenge in path planning has been crowded environments where obstacles are many and dynamic [3]. This challenge can easily be visualized with the 2D example of pedestrians walking along a path, shown in Fig. 1. If pedestrian A is approaching two other pedestrians, pedestrian B and pedestrian C, then the three pedestrians' paths might converge at some time in the future, t_f. Pedestrian A will observe the motions of pedestrians B and C and try to infer what future paths they will take. Depending on the inferred paths, pedestrian A might make a trajectory change to avoid the gap between pedestrians B and C at time t_f.

Many replanning algorithms avoid the complications of dynamic obstacles by assuming that obstacles, which are

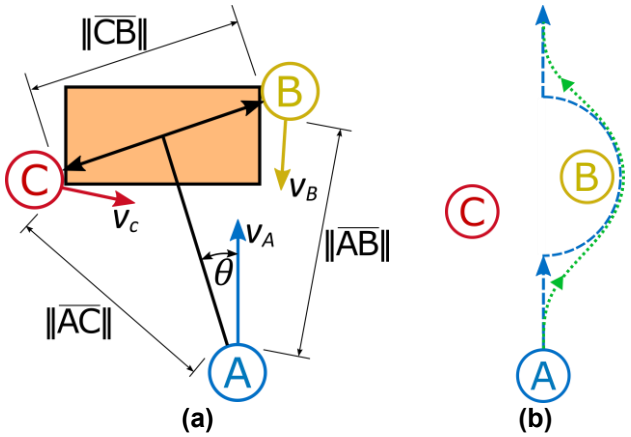

FIGURE 1. a) PEDESTRIANS A, B, AND C APPROACHING WITH THE OBSTACLE PAIR AREA BETWEEN B AND C, **b)** TWO POSSIBLE TRAJECTORIES FOR PEDESTRIAN A

dynamic between initial and final time, are static at each time step [4]. Another simplification is accounting only for the tip of the end effector when considering collisions [5]. These assumptions could be hazardous when a robot is working amongst a human or working in synchronization with the human.

A recent area of research is predicting obstacle trajectories. Mainprice and Berenson utilized a Gaussian Mixture Model (GMM) to extract human motion and compute likelihood of workspace occupancy from data collected from a depth camera [6]. Liu and Wang utilized the notion that human actions in a workcell cycle are a linear sequence that is pre-defined for a manufacturing task and created a Hidden Markov Model to predict human motions [7]. Zhang et al. utilized recurrent neural networks to recognize human tasks and predict future human motion trajectories, including time data [8]. In an offline approach to workspace sharing for human-robot cooperation, Pelligrinelli et al. [9] generated Human Occupancy Volumes that yield an indicator of the likelihood a volume will be occupied by a human worker.

This paper provides a method which leverages the predicted and real-time motions of humans and other obstacles throughout the workcell cycle to call for a trajectory re-plan before the robot must stop due to a pair of obstacles close enough in proximity to prohibit robot passage. This paper posits that if the collision situations between multiple dynamic obstacles can be predicted before the collision occurs, then a trajectory re-planned to avoid the predicted collision will be smoother and have less execution time compared to a trajectory that was re-planned at the time of impending collision. This concept is illustrated with the pedestrian example in Fig. 1b, where the trajectory of pedestrian A shown by the dashed line would be slower than that shown by the dotted line due to the sharp turn and corresponding severe velocity reduction in the dashed line once collision was imminent. The method, herein, captures not only predicted robot collisions with single dynamic obstacles, but also predicted collision risks of robot passage between pairs of obstacles that

would have been missed if obstacles were considered individually.

The risk estimation ties into a broader robot control structure by allowing the robot to execute trajectories with less interruptions. Figure 2 shows the industrial robot control loop, showing how the proposed method takes feedback from the sensors and predictions and provides an output to the trajectory planner. During offline trajectory planning, the proposed method ensures the planned trajectories will avoid high risk volumes of the workcell between obstacles where it is predicted that passage may not be possible. During the robot's real-time execution of its tasks, the proposed method continuously monitors for predicted and real-time occurring dynamic obstacles and detects associated high-risk volumes between obstacles prone to blocking robot passage, quantified by a threshold based on updated sensor and prediction data. If such a volume is detected along the robot's trajectory, an obstacle pair volume (OPV) is generated, added to the planning scene, and the robot's trajectory is re-planned online to avoid in real-time the volume of high risk and potential task interruptions. The remainder of the paper is organized into the following sections: estimation of the Risk of Passage (ROP) through the volume between two dynamic obstacles, development of actual risk and risk estimation error, OPV generation, and evaluation by simulation.

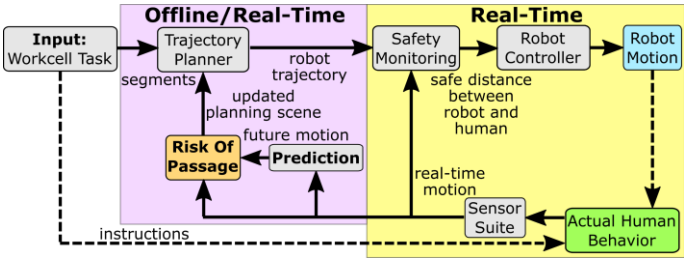

FIGURE 2. ROBOT SYSTEM CONTROL BLOCK DIAGRAM

2. ROP ALGORITHM, ADAPTATION AND CONTROL

In order to apply the estimated risk of passage algorithm to any robot and any number and type of obstacle, the algorithm parameters must be able to adapt accordingly. Additionally, an actual measure of risk can be derived from the times the robot waits for a re-plan and the time the robot executes a new trajectory in the event of a collision. Therefore, this paper presents a method of parameter adaptation to minimize the error between the estimated and actual risk of passage determination.

2.1 Assumptions

For this paper, it is assumed that motion of each obstacle in the robot's workspace is defined by a set of keypoints. As input to the controller's ROP, sensor data and the 'Prediction' block's prediction/cognition algorithms [8] are assumed to provide both the real-time and predicted location, velocity, and uncertainty of those keypoints, respectively. The predicted locations and velocities of the obstacle keypoints are assumed valid for the entirety of some planning horizon. The actual locations of the obstacle keypoints are also assumed to be updated at least as frequently as the robot system's real-time execution frequency. Additionally, the approximate shape and size of each obstacle is assumed to be provided by an algorithm that interprets data from sensors such as depth cameras. Furthermore, it is assumed that the robot's nominal trajectory has been pre-planned for a known task. The parameter adaptation method presented in this paper utilizes linear parameterizations so it must be assumed that the factors contributing to risk are linear in the unknown parameters.

2.2 Definition of the Obstacle Pair Volume (OPV)

The obstacle pair volume (OPV) is defined as the volume between a pair of obstacles in the robot's workspace. If the workspace contains more than two obstacles, then an OPV is defined for each unique set of two obstacles. Two keypoints on each obstacle will be determined that ensure the OPV completely encompasses both obstacles. If the obstacles are generalized to cylinders, then the center of each face of the cylinders will be used as keypoints for a total of four keypoints. Those four keypoints, along with the width and height of the generalized obstacles, can be used to define the shape of the OPV. For the purposes of calculating the robot risk of passage presented in the next section, only four keypoints of the obstacles will be considered for defining a 2D OPV. Figure 3 shows a 2D OPV defined between two human forearms. The human is modeled as a skeleton with the human's shape generalized to cylinders, with the forearms labeled as obstacle 1 and 2. The keypoints of the forearms are shown as (red) circles. The OPV, which is currently only defined as planes, is formed between the forearm keypoints, shown as a faceted (green) surface. The actual volume of the OPV, as opposed to the planes, is generated if the risk is too high.

2.3 Probability of Robot Passage

An approximation of the risk associated with passage through a volume between a dynamic obstacle pair can have

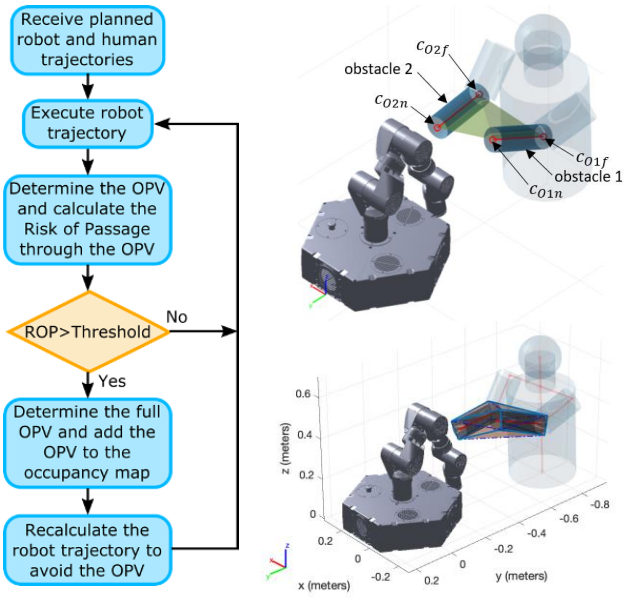

FIGURE 3. ROP FLOW DIAGRAM AND SCENE WITH A HUMAN AND ROBOT

many components. Compared to the 2D illustration of pedestrians presented in the introduction, the considerations become more complex when approximating the risk of passage of a robot manipulator through a volume between dynamic 3D obstacles. The obstacles for a robot manipulator in a work cell could be humans or their appendages, such as arms or hands, or the obstacles could be links of other manipulators, automated equipment, ground vehicles, etc. An expression for the predicted risk level will now be developed to quantify the estimated risk of delay or entrapment due to passing through an OPV. The components of the estimated risk, shown in Table 1, will have values between zero and one with zero corresponding to no risk and one corresponding to highest risk. A measure of the actual risk level will also be developed as delay or entrapment events occur so the estimation can be adapted to improve accuracy.

TABLE 1: FACTORS CONTRIBUTING TO ROBOT RISK OF PASSAGE THROUGH AN OPV.

Grouping	Factor	Contribution	
Backtracking capability	f_1	Reaction speed, or reaction time, of the robot required for the robot to move out of the volume between obstacles.	
	f_2	Approach angle between the robot manipulator and the volume between obstacles.	
Closure of passage window	f_3	Time to collapse the near (f_3) and far (f_4) gap (width) of the OPV if the obstacles are dynamic and approaching. Can robot make it through the space before the space is too small?	
	f_4		
Dimensions of obstacles, distance to backtrack out of collision situation	f_5	Length of obstacle 1.	Together, these two factors are the contribution due to the length of the OPV. The longer the OPV is, the more time required to retract parallel to the approach direction of the OPV if robot passage becomes blocked in the middle of the OPV.
	f_6	Length of obstacle 2.	
	f_7	Size of obstacle 1.	The larger the radius of an obstacle, or height of an obstacle, the longer it would take the robot to retract perpendicular to approach direction of the OPV should passage between obstacles become blocked.
	f_8	Size of obstacle 2.	

Figure 3 illustrates the proposed sequence between determining the initial OPV, determining Risk of Passage through the OPV, determining the full OPV, and re-planning the robot trajectory around the full OPV. The predicted risk level, which is updated continuously at the robot systems real-time frequency, will be defined as $\hat{\varphi}$ with the following function:

$$\hat{\varphi} = f\left(J, c_{O1n}, c_{O1f}, c_{O2n}, c_{O2f}\right), \hat{\varphi} \in \mathbb{R} \tag{1}$$

where J is the robot Jacobian, and c_{O1n}, c_{O1f}, c_{O2n}, and c_{O2f} are the obstacle keypoints at a given time step as shown in Fig. 3. Points c_{O1n}, c_{O1f}, c_{O2n}, and c_{O2f} would be the joints of a skeleton model in the case of human obstacles. Subscript *n* and *f* will be used to denote the points or edges that are nearest to the robot base or farthest from the robot base, respectively. In the case of a human with arms extended towards the robot, c_{O1n} and c_{O2n} would be the wrist joints and c_{O1f} and c_{O2f} would be the elbow joints as shown in Fig. 3. The position of those four points can be determined from real-time data or come from prediction algorithms such as recurrent neural networks [8]. Without loss of generality, the following theoretical derivations assume an articulated robot manipulator with all revolute joints.

The robot geometry at the point in time when the robot would enter the OPV should affect the risk of passage. First, to determine this risk factor, the robot's ability to navigate away from the OPV should be evaluated. Considering the robot geometry required to reach into the OPV, if the joint variables can be adjusted by a small amount to cause a displacement of the manipulator large enough to avoid the OPV, then this factor of risk is low. However, if a large change in joint variables is necessary to move the manipulator arm enough to avoid the OPV, then this factor of risk is high because the manipulator is less likely to avoid the OPV before colliding. Next, a predictive collision detection method is used to determine the predicted time a collision would start and end, which will be referred to as the predicted collision interval [10]. The geometry of the manipulator is swept through the trajectory of robot poses over time and a 3D spatial map is generated using Coon's patches. A line for collision checking is created between the two OPV points closest to the robot base and the two OPV points farthest from the robot base and each line is discretized to the grid space of the Coon's patches.

The robot Jacobian is utilized to determine the end effector velocity if all joints were actuated at their maximum velocity simultaneously for each robot instantaneous pose at the times the robot enters and exits the OPV. To normalize the risk factor due to robot geometry, the resulting end effector cartesian velocity will be divided by the maximum end effector velocity of the robot. This will be the norm of the end effector cartesian velocity if the robot were fully extended and all the joints were actuated with the maximum joint velocities. In determining the robot Jacobian, forward kinematics determined equations for the position of the end effector in the fixed cartesian coordinate system. Then the robot Jacobian was determined by taking the partial derivatives of the x, y, and z positions with respect to each variable robot joint parameter. The translational Jacobian for the end effector of a manipulator with n total, revolute and prismatic, joints is given by:

$$J = \begin{bmatrix} \frac{dx}{dq_1} & \cdots & \frac{dx}{dq_n} \\ \frac{dy}{dq_1} & \cdots & \frac{dy}{dq_n} \\ \frac{dz}{dq_1} & \cdots & \frac{dz}{dq_n} \end{bmatrix} \tag{2}$$

The maximum end effector velocities at the time the robot enters and exits the OPV are given by:

$$v_s = J(q_s)\dot{q}_{max}, \qquad s = \{entry, exit, max\} \tag{3}$$

where q_{entry} is the vector of joint variable values at the point when the robot enters the OPV, q_{exit} is the vector of joint

variable values at the point when the robot exits the OPV, and q_{max} is the vector of joint variable values that can maximize the end effector linear velocity. The $\dot{q}_{max}$ is a column vector of the robot's maximum joint velocities. When considering the velocity of the end effector, the worst-case scenario for getting out of a risky situation would be to consider the velocity v_{entry} or v_{exit} which has the smallest norm, defined as v_{OPV}:

$$v_{OPV} \triangleq \underset{v=v_{entry},v_{exit}}{\arg\min} \ (\|v\|) \tag{4}$$

Now the risk due to robot geometry at the time the robot is in the OPV is given by:

$$f_1 = \delta_1 \left(1 - \frac{\max\left(\left\| v_{OPV_{x,y}} \right\|, \left\| v_{OPV_z} \right\| \right)}{\max\left(\left\| v_{max_{x,y}} \right\|, \left\| v_{\max_z} \right\| \right)} \right). \tag{5}$$

Another factor that contributes to the risk of passage is the orientation of the obstacle pair volume relative to the manipulator. Two obstacles could be oriented such that if the manipulator were in between the obstacles and had to re-plan its trajectory, the manipulator would have great difficulty retracting from the obstacle pair volume. On the other hand, two obstacles could be oriented such that if the manipulator needed to get out of the OPV, a small adjustment in the joint variables could manipulate the robot out of the OPV. Before defining this factor of risk, some vector and point definitions must be introduced. Vectors between the obstacles' near and far keypoints are:

$$\bar{E}_{12n} \triangleq c_{O2n} - c_{O1n} \text{ and } \bar{E}_{12f} \triangleq c_{O2f} - c_{O1f}. \tag{6}$$

In Fig. 3, the points c_{O1n} and c_{O2n} are nearest to the robot base and c_{O1f} and c_{O2f} are farthest from the robot base. The midpoints of Eq. (6) vectors are defined as mp_{E12n} being the midpoint of $\bar{E}_{12n}$ and mp_{E12f} being the midpoint of $\bar{E}_{12f}$. Now the risk factor due to the orientation of the OPV relative to the manipulator can be expressed as:

$$f_2 = \delta_2 \left(1 - \frac{\left| \left(mp_{E12f} - mp_{E12n} \right) \cdot \left(mp_{E12n} - 0 \right) \right|}{\left\| mp_{E12f} - mp_{E12n} \right\| \left\| mp_{E12n} - 0 \right\|} \right) \tag{7}$$

The right side of Eq. (7), subtracted from one is the cosine of the angle between the vectors formed by $mp_{E12f} - mp_{E12n}$ and $mp_{E12n} - 0$. Point 0 is the location of the robot's base. When these two vectors are perpendicular as opposed to parallel, then the robot requires a side approach to the obstacles instead of a head on approach in order to back out from between the obstacles.

Another more obvious contribution to the risk of passing between obstacles in the future is the time until the volume between the obstacles is too small to allow passage, or in other words, the rate of collapse of the OPV. An OPV might be sufficiently large so there is no risk of passage at one time step, but at that time step the velocities of the points c_{O1n}, c_{O1f}, c_{O2n}, and c_{O2f} might indicate that at the next time step the OPV will be small enough that the risk of passage is too high for passage. This will be based on the velocities of the centroids at each end of the obstacles, $v_{c_{O1n}}$, $v_{c_{O2n}}$, $v_{c_{O1f}}$, $v_{c_{O1f}}$, as expressed by the following:

$$v_s \triangleq \left(\bar{v}_{c_{O1s}} - \bar{v}_{c_{O2s}} \right) \cdot \frac{\bar{E}_{12_s}}{\left\| \bar{E}_{12_s} \right\|}, \qquad s = \{n, f\} \tag{8}$$

The relative velocities are projected onto the direction of the vectors between the near and far centroids of the obstacles ($\bar{E}_{12_n}$ and $\bar{E}_{12_f}$) so that only the velocity in the direction that reduces the volume ($+v_s$) is considered. The contributions to risk due to the time until the OPV collapses enough at the near and far end of the OPV to prevent passage can be expressed by the following:

$$t_{collapse_s} \triangleq \frac{\left\| \bar{E}_{12_s} \right\| - r_1 - r_2 - r_{robot}}{v_s}, \qquad s \in \{n, f\} \tag{9}$$

$$f_{3,4} = \delta_{3,4} \begin{cases} 0 & if \ t_{collapse_s} < 0 \ or \ v_s = 0 \\ 1 - \dfrac{t_{entry} - t_{collapse_s}}{t_{horizon}} & if \ t_{collapse_s} < t_{entry} \\ 1 - \dfrac{t_{collapse_s} - t_{exit}}{t_{horizon}} & if \ t_{exit} < t_{collapse_s} \\ 1 & if \ t_{entry} \le t_{collapse_s} \le t_{exit} \\ & \forall \ t_{exit} > t_{entry}, \qquad s \in \{n, k\} \end{cases} \tag{10}$$

where $s = n$ for f_3 and $s = f$ for f_4. The t_{entry} is the time when the robot first enters the OPV, t_{exit} is the time when the robot exits the OPV, and $t_{horizon}$ is the maximum time horizon for consideration. The radii r_1, r_2, and r_{robot} are the radius of obstacle 1, the radius of obstacle 2, and the minimum distance for passage through the OPV required by the robot, respectively. Both ends of the OPV are considered because the obstacles could possibly be oriented such that the distance between obstacles along the near edge is not less than the distance along the far edge or vice versa. The contribution to risk is between zero and one for both cases: a) if the OPV collapses before the robot enters the OPV and b) if the OPV collapses after the robot exits the OPV. The contribution to risk is one if the OPV collapses while the robot is in the OPV. This factor of risk is zero if the OPV near or far keypoints are diverging or if the time of OPV collapse is predicted farther in the future than the planning horizon.

The length of each obstacle would determine the volume of the OPV and the amount of time or joint actuation required to move the robot out of the OPV in the event of a collision situation. Therefore, the risk contributions due to the length of the obstacles 1 and 2 ($\|\bar{E}_1\|$ or $\|\bar{E}_2\|$, respectively) are:

$$f_5 = \delta_5 sat_{(0,1)} \left(\frac{\|\bar{E}_1\|}{l_{max}} \right) \tag{11}$$

$$f_6 = \delta_6 sat_{(0,1)} \left(\frac{\|\bar{E}_2\|}{l_{max}} \right) \tag{12}$$

where l_{max} is a maximum length of an obstacle ($\|\bar{E}_1\|$ or $\|\bar{E}_2\|$) to consider, which is a parameter set by the user. The user would likely set l_{max} less than or equal to the maximum reach of the robot. The radius of each obstacle also determines the volume of the OPV, defining the following risk components:

$$f_7 = \delta_7 sat_{(0,1)} \left(\frac{r_1}{r_{max}} \right) \tag{13}$$

$$f_8 = \delta_8 sat_{(0,1)}\left(\frac{r_2}{r_{max}}\right) \quad (14)$$

where r_{max} is a maximum radius to consider for an obstacle. This will likely be less than or equal to the reach of the robot. A limit was placed on the obstacle radius because if each obstacle of the pair were larger in radius than the reach of the robot, then half of the robot's work area would be occupied and unreachable. A robot manipulator will more easily get trapped between obstacles of larger radius and can more easily remove itself from the volume between obstacles of smaller radius.

The estimated risk of passage, $\hat{\varphi}$, is now defined as the product of a regression vector Y and a vector $\hat{\delta}$ containing estimates of the unknown coefficients in Eq. (5) through (14), defined as

$$\hat{\varphi} = Y\hat{\delta} = [Y_1 \quad \dots \quad Y_8][\delta_1 \quad \dots \quad \delta_8]^T, \quad (15)$$

$$Y_i = \frac{f_i}{\delta_i}, i = \{1, \dots, 8\}. \quad (16)$$

The actual risk can only be updated once robot passage through the OPV becomes prohibited due to OPV size at that time. The actual risk is a function of the likelihood of a collision occurring, the time the manipulator trajectory planner is generating a new trajectory, as well as the additional time the manipulator takes to move around the obstacle pair volume. This can be expressed as

$$\varphi(k) = \frac{m}{k}\min\left(\frac{t_{replan} + t_{exec}}{t_{delay_{max}}}, 1\right) \quad (17)$$

where m is the number of times a re-plan was needed to navigate around the obstacles and k is the total number of simulated workcell cycles. The duration t_{replan} is the time the robot system spends backing out of the OPV and then computing an updated trajectory after the OPV prohibits robot passage. The time t_{exec} is the duration the manipulator takes to execute the replanned trajectory that goes around the OPV. The time $t_{delay_{max}}$ is a maximum time threshold for the delay. The rationale behind the delay threshold is that the robot could take an infinitely long time to generate and execute a re-plan, but the estimated risk should not be infinite and should saturate at one. Therefore, dividing the trajectory re-planning and execution time by the threshold and taking the minimum of that normalized value and one ensures $\varphi(k)$ will be bounded between zero and one. Since Eq. (17) consists of counts of cycles and time, $\varphi(k)$ can never be less than zero.

The risk estimation error and parameter estimation error will be given by:

$$\tilde{\varphi}(k) = \varphi - \hat{\varphi}(k) = Y\tilde{\delta}(k) \; where \; \tilde{\delta}(k) = \delta - \hat{\delta}(k). \quad (18)$$

A parameter update law to adjust the parameters in $\hat{\delta}(k)$ must be developed to minimize $\tilde{\delta}(k)$ as follows:

$$\hat{\delta}(k+1) = \max\left(0, \hat{\delta}(k) + \Gamma Y^T \tilde{\varphi}(k)\right). \quad (19)$$

The value of $\hat{\delta}(k+1)$ has a lower bound of zero because none of the factors contributing to risk should be negative, which would reduce the effect of other factors. The matrix Γ is a positive definite gains matrix that the user selects. The rate of learning for $\hat{\delta}$ is inversely proportional to the minimum eigenvalue of Γ, but larger values for the elements of Γ could cause greater steady state error in $\hat{\delta}$.

2.4 Parameter Estimation Error Stability

The stability analysis begins with the following simple, positive definite, Lyapunov like candidate function which was selected to show that $\tilde{\delta}(k)$ will be minimized:

$$V_l(k) = \tilde{\delta}(k)^T \Gamma^{-1} \tilde{\delta}(k). \quad (20)$$

The delta of the Lyapunov like function between iterations is:

$$\Delta V_l(k+1) = V_l(k+1) - V_l(k) \quad (21)$$

It can be shown that $\Delta V_l(k+1)$ after substitutions becomes:

$$\Delta V_l(k+1) = \left(\hat{\delta}(k+1) - \delta\right)^T Y^T \Gamma^{-1} Y \Gamma Y^T \tilde{\varphi}(k+1) \quad (22)$$

$\Delta V_l(k+1)$ will always be negative or zero because the elements of Y and Γ are always positive and the sign of the elements of $\hat{\delta}(k+1) - \delta$ and the sign of the elements of $\tilde{\varphi}(k+1)$ will always oppose each other, except when the estimation error is exactly zero. Therefore, as the number of robot cycles approaches infinity, the parameter estimation error will diminish.

3. RESULTS AND DISCUSSION

The goal of the Risk of Passage (ROP) algorithm presented in this paper is to estimate the risk of manipulator passage between dynamic obstacles encountered in manufacturing HRC. The algorithm predicts the likelihood obstacles will prevent the manipulators passage between them at a future time. If the estimated risk is greater than a user selected threshold, then an Obstacle Pair Volume (OPV) is created, blocking unsafe passage of any part of the robot arm between the moving obstacles. This ensures real-time updates by a trajectory planner and safe robotic interaction with humans within manufacturing workcells. If the manipulator can avoid the spaces between obstacles which may prohibit passage in the future, then the increase in the manipulators time to reach the goal pose will be less than if the OPV were not created and the robot's trajectory must be replanned after getting too close to the obstacles. In this section, implementation of the ROP-based robot system is presented in simulation, demonstrating learning of ROP parameters and creation of OPVs for a variety of likely scenarios.

3.1 Simulation Design

To test the ROP estimation algorithm, many scenarios were developed so the algorithm can be evaluated at many points in the state space defined by the parameters of Y in Eq. (5) through (14) and (16). Each scenario includes a trajectory for the robot as well as trajectories, dimensions, and rotations for two cylinders, which are the dynamic obstacles. The two cylinders are intended to represent the forearms of a human. For many scenarios, the cylinders have a length of 0.2 meters and radius of

0.04 meters, approximately the same dimensions as a forearm. The scenarios vary the parameters of the cylinders according to the list in Table 2. Those scenarios also have an associated robot trajectory, which is listed in Table 3. Most of the scenarios position the cylinders in a way that prohibits the robot from executing its original trajectory, which would pass between the cylinders. Scenarios 20, 21, 42, and 47 from Table 2 allow robot passage, demonstrating that the OPV does not need to be created when the ROP is low enough. The parameters adjusted for each scenario can be seen in Fig. 4.

The robot's trajectories from the starting joint configuration to the end joint configuration are planned before the obstacles have been added to the collision checking planning scene. Therefore, the obstacles represent unforeseen human movement in the robot's workcell. The obstacles are introduced according to one of the obstacle scenarios once the robot begins its trajectory. Initially, the risk threshold for introducing the OPV is set relatively high so the OPV is not introduced before the ROP parameters, from Eq. (15), are learned and at steady state. This allows the system to use every scenario provided to the simulation to learn the correct weights to make the ROP estimation as accurate as possible. When the OPV was not created, the robot proceeded along its original trajectory and then stopped once the robot was within 5 cm of either obstacle. Then the robot backed away from the obstacles and replanned its trajectory to navigate around the OPV and reach the goal pose. The robot needed to back away because when the robot was within 5 cm of an obstacle the trajectory planner did not have enough clearance to generate a valid collision-free trajectory.

TABLE 2. OBSTACLE TRAJECTORIES FOR SIMULATION

Obstacle Scenario	Description
1-21	Vary the pitch angle of cylinder 1 from 0 to 180 degrees with robot trajectory 1
22-32	Vary the approach duration, velocity and direction in the x and y directions of the cylinders with robot trajectory 1
33-34	Vary the yaw angle of the cylinders between -90 and 90 degrees with robot trajectory 1
35-42	Vary approach durations and velocities of the cylinders in the y direction with robot trajectory 2
43-47	Vary approach durations and velocities of the cylinders in the y direction with robot trajectory 3
48-52	Vary the radius and length of the cylinders with robot trajectory 1
53-57	Vary the radius and length of the cylinders with robot trajectory 2

TABLE 3. ROBOT TRAJECTORIES FOR LEARNING

Trajectory		Joint Angles (°)
1	start	[90,0, 0,0,0,0]
	goal	[90,97, 12,0, −6,0]
2	start	[34,74, 0,0,0,0]
	goal	[115,74, 0,0,0,0]
3	start	[34,74, 0,0,0,0]
	goal	[126,46, 46, −90, −17,0]

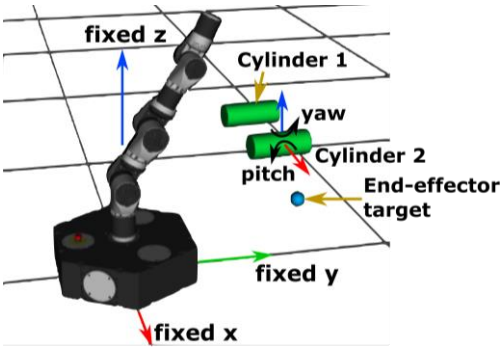

FIGURE 4. ROBOT AND OBSTACLE DEFINITIONS

Once the parameters of the ROP algorithm had been learned and were at steady state, the risk threshold for introducing the OPV was set relatively low to cause the OPV to be introduced for most scenarios. When the OPV is allowed, as soon as the obstacles are introduced, the system calculates the ROP at the beginning of each scenario and adds the OPV at the size necessary for when the robot will be near the obstacles. The ROP algorithm can calculate risk due to the robot trajectory and obstacles at any time in the trajectory, but due to limitations with simulation, only the beginning of the scenarios is used to calculate risk or add the OPV. Many scenarios were designed so that the cylinders are far apart at the start of a scenario and then approach as time progresses. The robot starts at a position far from the obstacles and then approaches the obstacles. Therefore, the OPV necessary for avoiding the obstacles needs to be sized according to the position of the cylinders at the future time when the robot would be near the cylinders. To ensure enough scenarios were completed for learning the weights, the 57 obstacle scenarios in Table 2 along with their corresponding robot trajectory were executed at least four times, for a total of at least 228 cycles. Additionally, to ensure the ordering of the obstacle scenarios does not create a bias in the learned weights, the obstacle scenarios are executed in a random order.

The ROP algorithm was evaluated on an Ubuntu 18.04 Virtual Machine running on a computer with four 2.3 Ghz Intel i7 CPU cores and using Robot Operating System, MoveIt! libraries, and Python 2.7 [11]. The robot used for simulation in both systems was a Comau e.Do robot [12], which can be seen in Fig. 4. ROS enables simulations that provide results that are very similar to results obtained when using a physical robot and collecting data from sensors. The MoveIt! libraries utilize the Flexible Collision Library for collision checking, which considers the entire manipulator geometry when checking for collisions with the obstacles [13]. Additionally, MoveIt! utilizes the Open Motion Planning Library (OMPL) [14], whose planners check for collisions between any point along the robot and obstacles when considering locations for nodes in joint space. Due to the constraints of ROS with the MoveIt! libraries, the OPV was added as a rectangular prism surrounding the two obstacles.

To evaluate the effectiveness of the ROP algorithm, the Bi-Rapidly Exploring Random Trees (BiRRT), Probabilistic Road Map (PRM), and Sparse Roadmap Spanners (SPARS) planners in OMPL will be considered. The BiRRT planner was selected for this experiment because Nicola et al. [1] found that it was the fastest RRT planner in OMPL and since it comes from the RRT family it has randomness built into its node selection. The BiRRT algorithm generates two trees in joint space, one which begins at the start node and the other that starts from the goal node [15]. The two trees are connected after expansion to generate a path from start to goal. Each node of the trees is generated by randomly generating a point in joint space, finding its nearest neighbor node, and then making a step from the nearest node in the direction of the random point. For testing, the BiRRT algorithm was configured to use an initial temperature of 100, temperature change factor of 0.1, frountier node ratio of 0.1, and cost threshold of infinity. The PRM algorithm was selected for testing because it is widely used in robot path planning and should provide trajectories that are less random than an RRT planner [16]. The PRM algorithm also randomly fills joint space with nodes [17]. Nodes are then selected and connected to form the most optimal path from start to goal from the generated nodes. PRM is more deterministic than an RRT method since PRM's determination of new nodes does not depend on previously generated nodes. For testing, PRM was set to find a maximum of 10 nearest neighbors per node. The SPARS planner was selected for testing because it is one of the more recently developed planners in OMPL and in theory should be less random than BiRRT and have faster trajectory generation time than PRM due to sparse node selection. The SPARS algorithm is an extension of sampling-based planners such as PRM [18]. SPARS generates nodes in C-space in the same way was PRM, except SPARS does not add all nodes to the graph, creating a sparse collection of nodes in C-space. SPARS includes adaptation to determine the number of nodes to keep. SPARS utilizes the PRM* algorithm to construct a graph of the nodes, but is different in that it uses "visibility regions" to ensure path lengths are less than or equal to that of paths generated by the standard PRM* algorithm.

The initial ROP parameter weights were $\hat{\delta}(0) = [.3, .1, .1, .1, .1, .1, .1]^T$, selected to ensure the parameters did not start at the steady state values to demonstrate adaptation. The learning gain was $\Gamma = diag([.2, .1, .1, .1, .2, .2, 1, 1])$, selected by trial and error to find the parameters that would provide the fastest parameter convergence, most stable parameter estimates, and least risk estimation error. Additionally, the maximum time considered for a re-planned trajectory was 45 seconds. No randomness was applied to the motion of the cylinders, so the m divided by k part of Eq. (17) is always one.

3.2 Simulation Results

The results obtained from simulating the robot and obstacle scenarios are shown in Table 4. The results indicate that when using a random planner, such as RRT, the randomness of the planned robot joint trajectories can have a major impact on the accuracy of the learned ROP parameters. Random planners build a tree by randomly placing tree nodes in the joint parameter space. A trajectory from start to goal is generated by finding the most optimal connection of the nodes in the tree. The number of allowed planning iterations was limited to ensure a solution for safe real-time robot reactive response to dynamic obstacles (e.g., human). Therefore, during a few cycles the random planners selected trajectories that were significantly less optimal than most other robot cycles, as indicated by the spikes in actual trajectory times in Fig. 5. Those instances had very large planning times, so the risk estimation reached an unusually large value for those cycles, shown by the spikes in in Fig. 6. This caused a slight increase in ROP parameter weights after those cycles due to continuous adaptation of the weights as new data is acquired. Then the parameters would require a few cycles to return to correct values again, as shown in Fig. 7.

The results indicate that the ROP algorithm is able to adapt its parameters because the parameters appear to reach steady state values, as shown after approximately 120 robot cycles of

TABLE 4. SIMULATED EXPERIMENTAL RESULTS

Trajectory Duration Statistics (seconds unless noted otherwise)			
Planner	BiRRT	PRM	SPARS
Nominal (no obstacle)-Average	6.92	6.92	7.01
Nominal (no obs.)-Std. Deviation	0.611	0.607	0.910
Actual w/o OPV-Average	26.87	41.71	49.93
Actual w/o OPV-Std. Deviation	9.72	11.78	17.36
Actual w/o OPV-% of Nominal	389%	603%	713%
Actual w/ OPV-Average	25.62	39.00	47.03
Actual w/ OPV-Std. Deviation	9.99	7.69	16.84
Actual w/ OPV-% of Nominal	373%	561%	656%
Reduction in Avg. Cycle Time[1]	**1.25**	**2.71**	**2.90**
Percent Improvement[2]	**4.65%**	**6.50%**	**5.81%**
Steady state ROP estimate average	0.285	0.342	0.442
Steady state ROP estimate std. dev.	0.062	0.056	0.081

1. Calculated as "Actual w/o OPV-Average"-"Actual w/ OPV-Average"
2. Calculated as "Reduction in Avg. Cycle Time"/"Actual w/o OPV-Average"

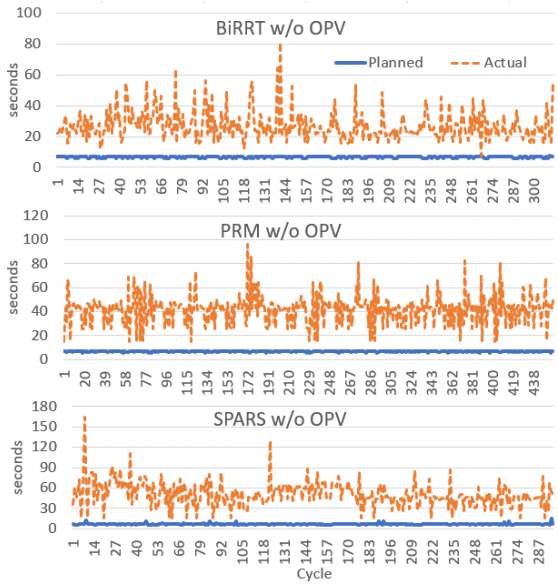

FIGURE 5. TRAJECTORY EXECUTATION TIMES

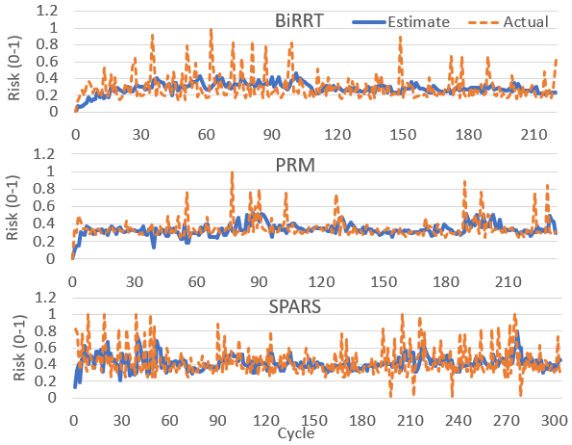

FIGURE 6. RISK ESTIMATES AND ACTUALS

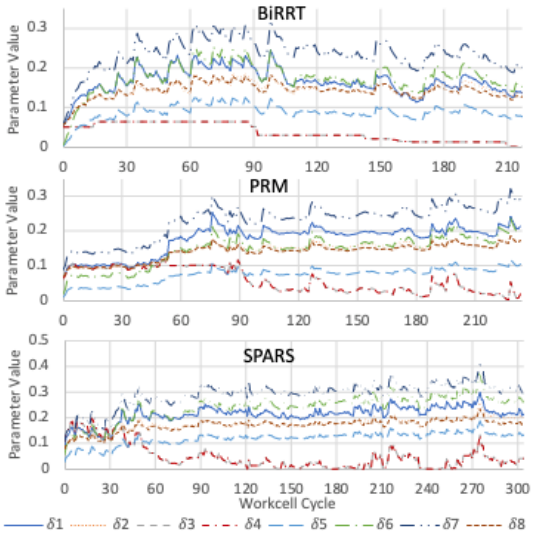

FIGURE 7. RISK OF PASSAGE PARAMETER ESTIMATES

learning in Fig. 7, and the estimated risk appears to be the average of the actual risk at steady state, as shown in Fig. 6. The success of adaptation would have been much clearer if the risk estimation were the same as the actual risk for every robot cycle at steady state. However, this is not possible due to randomness in trajectory durations. It is also possible that randomness comes from the hardware and software used. The computer operating system was not a real-time operating system, so CPU cores could not be dedicated to running only ROS threads.

The results show that when an obstacle prevented the robot from executing the original trajectory, the average trajectory duration was lower when the OPV was created compared to when the OPV was not created. This was observed with all three planners tested, shown in bold in Table 4. The nominal trajectory times were determined by the trajectory planners when no obstacles were present and serve as a baseline trajectory time. When the OPV was used, the average trajectory durations were 1.25 seconds shorter with the BiRRT planner, 2.71 seconds shorter with PRM, and 2.90 seconds shorter with SPARS compared to cycles times without introducing the OPV. Those reductions in cycle times corresponded to percentages of cycle time saved of 4.65% for BiRRT, 6.50% for PRM, and 5.81% for SPARS when compared to the cycle times without introducing the OPV. Therefore, the results indicate the ROP algorithm becomes more beneficial as the computation time of the planner increases. The observed reductions in trajectory durations are partially attributed to ROP avoiding the necessity of the robot backing away from the obstacles. Other factors, such as selection of planner, clearly influenced the improvement in cycle time because the reduction in average cycle time was not constant for all three planners. Results show ROP proactively smooths reactions, reducing cycle time and increasing productivity in a workcell with dynamic, unforeseen obstacles.

Another observation is that the PRM planner produced the lowest standard deviation in the steady state ROP parameters estimates. That was 0.056 for PRM compared to 0.062 for the BiRRT planner and 0.081 for the SPARS planner. The PRM planner also showed a smaller standard deviation in cycle time when using the OPV compared to the BiRRT and SPARS planners. The PRM planner utilizes a less random method of selecting nodes for the planning tree compared to the BiRRT planner. Therefore, the ROP algorithm may be more beneficial when used with a planner with more constrained randomness. The trajectory times for each individual scenario were also inspected to see if any scenarios led to unusually high trajectory durations. The results showed that outliers caused the average trajectory duration for some scenarios to be relatively large, but most cycles for those same scenarios had close to average trajectory durations. Therefore, the peaks in trajectory duration cannot be attributed to individual scenarios.

The results also show that the SPARS algorithm had the highest averages and standard deviations of trajectory durations when the robot must avoid the obstacles, with and without the OPV. The SPARS planner also produced ROP parameter estimates with higher standard deviation than the other planners. This result seems contrary to the motivation for using the SPARS algorithm, per [18]. It was the planner closest to the state-of-the-art and the planner that should have the lowest computation time due to node sparsity, but it did not perform as well as the BiRRT or PRM planners in terms of planning time and producing stable steady state results. Therefore, the SPARS algorithm does not appear to be a good candidate for use with the ROP algorithm.

4. CONCLUSION

In conclusion, this paper demonstrates a method for estimating the risk of passage through volumes between predicted obstacle pairs. This allows the robot system to respond to multiple, unforeseen, dynamic obstacles before collisions become imminent. This is advantageous because it allows for the creation of faster trajectories that avoid collision situations, as opposed to waiting to replan trajectories until collision is imminent. The results showed that when the ROP algorithm estimates risk and generates the OPV, robot trajectories take less time on average. Thus, ROP is effective in ensuring safety in

HRC while respecting productivity. However, randomness in the robot trajectory planners, which effected the actual measure of risk used to train the ROP algorithm, made the results appear less stable than desired. Alternate parameter adaptation and machine learning techniques will be explored towards mitigating outlier effects and providing faster convergence of parameter estimates and better theoretical stability. Future work will also include introducing obstacles of other geometries besides cylinders and validating the effectiveness of ROP, with implementation in a more comprehensive, physical industrial robot control system.

ACKNOWLEDGEMENTS

Funding was provided by the NSF/NRI: INT: COLLAB: Manufacturing USA: Intelligent Human-Robot Collaboration for Smart Factory (Award I.D. #: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.

REFERENCES

[1] Nicora, M.L., Ambrosetti, R., Wiens, G.J., and Fassi, I. "Human-Robot Collaboration in Smart Manufacturing: Robot Reactive Behavior Intelligence." *ASME Journal of Manuf. Sci. Eng*. Vol. 143 No. 3 (2021): pp. 31-40. DOI:10.1115/1.4048950.

[2] Yang, L., Qi, J., Song, D., Xiao, J., Han, J., and Xia, Y. "Survey of Robot 3D Path Planning Algorithms." *Journal of Control Science and Engineering* (2016): pp. 1–22. DOI:10.1155/2016/7426913.

[3] Chao, C., Trautman, P., and Iba, S. "Dynamic Channel: A Planning Framework for Crowd Navigation." *IEEE Int. Conf. on Robotics and Automation*: pp. 5551–557. Montreal, QC, Canada, 2019. DOI:10.1109/ICRA.2019.8794192.

[4] Holmes, P.D., Kousik, S., Zhang, B., Raz, D., Barbalata, C., Johnson-Roberson, M., and Vasudevan, R. "Reachable Sets for Safe, Real-Time Manipulator Trajectory Design." *Robotics: Science and Systems XVI* Vol. 16 (2020): pp. 100-113. DOI:10.15607/RSS.2020.XVI.100.

[5] Du, S., Shang, W., Cong, S., Zhang, C., and Liu, K. "Moving Obstacle Avoidance of a 5-DOF Robot Manipulator by Using Repulsive Vector." *IEEE Int. Conf. on Robotics and Biomimetics*: pp. 688–693. Macau, China, December 5-8, 2017. DOI:10.1109/ROBIO.2017.8324497.

[6] Mainprice, J., and Berenson, D. "Human-Robot Collaborative Manipulation Planning Using Early Prediction of Human Motion." *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*: pp. 299–306. Tokyo, Japan, 2013. DOI:10.1109/IROS.2013.6696368.

[7] Liu, H., and Wang, L. "Human Motion Prediction for Human-Robot Collaboration." *Journal of Manufacturing Systems* Vol. 44 (2017): pp. 287–94. DOI:10.1016/J.JMSY.2017.04.009.

[8] Zhang, J., Liu, H., Chang, Q., Wang, L., Gao, R.X. "Recurrent Neural Network for Motion Trajectory Prediction in Human-Robot Collaborative Assembly." *Cirp Annals-Manufacturing Technology* Vol. 69 No. 1 (2020): pp. 9–12. DOI:10.1016/j.cirp.2018.04.066.

[9] Pellegrinelli, S., Moro, F.L., Pedrocchi, N., Tosatti, L.M., and Tolio, T.A.M. "A Probabilistic Approach to Workspace Sharing for Human–Robot Cooperation in Assembly Tasks." *Cirp Annals-Manufacturing Technology* Vol. 65 No. 1 (2016): pp. 57–60. DOI:10.1016/J.CIRP.2016.04.035.

[10] Streitmatter, G., and Wiens, G.J. "Human-Robot Collaboration: A Predictive Collision Detection Approach for Operation Within Dynamic Environments." *ASME International Symposium on Flexible Automation*. Online. July 8–9, 2020. DOI:10.1115/ISFA2020-9659.

[11] Coleman, D., Şucan, I.A., Chitta, S., Correll, N. "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study." *Journal of Software Engineering for Robotics* Vol. 5 No. 1 (2014): pp. 3–16. DOI:10.6092/JOSER_2014_05_01_p3.

[12] *e.Do Technical Sheet.* Comau S.p.A. (2017).

[13] Pan, J., Chitta, S., and Manocha, D. "FCL: A General-Purpose Library for Collision and Proximity Queries." *IEEE International Conference on Robotics and Automation*: pp. 3859–3866. Saint Paul, MN, 2012. DOI:10.1177/0278364911406761.

[14] Şucan, I.A., Moll, M., Kavraki, L.E. "The Open Motion Planning Library." *IEEE Robotics & Automation Magazine* Vol. 19 No. 4 (2012): pp. 72–82. DOI:10.1109/MRA.2012.2205651. https://ompl.kavrakilab.org.

[15] LaValle, S.M., Kuffner, J.J. "Randomized Kinodynamic Planning." *The International Journal of Robotics Research* Vol. 20 No. 5 (2001): pp. 378–400. DOI:10.1177/02783640122067453.

[16] Karaman, S., and Frazzoli, E. "Sampling-Based Algorithms for Optimal Motion Planning." *The International Journal of Robotics Research* Vol. 30 No. 7 (2011): pp. 846–94. DOI:10.1177/0278364911406761.

[17] Kavraki, L.E., Svestka, P., Latombe, J.C., and Overmars, M.H. "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces." *IEEE Transactions on Robotics and Automation* Vol. 12 No. 4 (1996): pp. 566-580. DOI:10.1109/70.508439.

[18] Dobson, A., and Bekris, K.E. "Sparse Roadmap Spanners for Asymptotically Near-Optimal Motion Planning." *The International Journal of Robotics Research* Vol. 33 No. 1 (2014): pp. 18–47. DOI:10.1177/0278364913498292.

