Proceedings of the ASME 2020
International Symposium on Flexible Automation
ISFA2020
July 5-9, 2020, Chicago, IL, USA

ISFA2020-9659

HUMAN-ROBOT COLLABORATION: A PREDICTIVE COLLISION DETECTION APPROACH FOR OPERATION WITHIN DYNAMIC ENVIRONMENTS

Gabriel Streitmatter, Gloria Wiens[1]
University of Florida, Gainesville, FL

ABSTRACT

Robots and humans closely working together within dynamic environments must be able to continuously look ahead and identify potential collisions within their ever-changing environment. To enable the robot to act upon such situational awareness, its controller requires an iterative collision detection capability that will allow for computationally efficient Proactive Adaptive Collaboration Intelligence (PACI) to ensure safe interactions.

In this paper, an algorithm is developed to evaluate a robot's trajectory, evaluate the dynamic environment that the robot operates in, and predict collisions between the robot and dynamic obstacles in its environment. This algorithm takes as input the joint motion data of predefined robot execution plans and constructs a sweep of the robot's instantaneous poses throughout time. The sweep models the trajectory as a point cloud containing all locations occupied by the robot and the time at which they will be occupied. To reduce the computational burden, Coons patches are leveraged to approximate the robot's instantaneous poses. In parallel, the algorithm creates a similar sweep to model any human(s) and other obstacles being tracked in the operating environment. Overlaying temporal mapping of the sweeps reveals anticipated collisions that will occur if the robot-human do not proactively modify their motion. The algorithm is designed to feed into a segmentation and switching logic framework and provide real-time proactive-n-reactive behavior for different levels of human-robot interactions, while maintaining safety and production efficiency. To evaluate the predictive collision detection approach, multiple test cases are presented to quantify the computational speed and accuracy in predicting collisions.

Keywords: Collision Detection, Swept Volume Interference, Multiple Interference Detection, Coons Patches, Human-Robot Interaction

[1]Contact author: gwiens@ufl.edu

INTRODUCTION

The state of the art of manufacturing is ever evolving towards increased intelligent automation. Robots in particular are of supreme interest. As customer demands tend towards increased complexity and customization of products, the necessity of the speed and precision of robots coupled with the creativity and problem-solving capabilities of humans is ever prevalent. Teams of humans and robots are uniquely capable of meeting stringent requirements by engaging in human-robot collaboration (HRC) and combining their talents.

This paper is part of a collaborative effort to develop robust methodologies for operation of HRC manufacturing cells. The approach includes data sensing and analysis to provide a robot cognizance of its surrounding dynamic environment and an ability to predict how the environment will evolve in the immediate future. The information provided by this analysis is assumed to come in the form of generalized shapes of dynamic obstacles, along with their positions and orientations throughout time. This, in effect, gives the robot a situational awareness that is essential to intelligent dynamic response [1]. A Proactive Adaptive Collaborative Intelligence (PACI) module is currently under development to allow the robot to use this situational awareness to react to a dynamically evolving environment.

In previous works, robot trajectories were subdivided into a series of segments. The segments were highly customizable to specific demands on the robot during operation. With this approach, the robot was able to respond to obstacles according to localized features of the segment of the trajectory under execution rather than the generalized trajectory as a whole [2]. The authors research goal is to extend the use of this methodology by allowing the robot to look ahead in its trajectory with the introduction of a predictive collision detection technique to identify at risk segments of the trajectory. This enables the robot to look ahead and make optimal decisions about how to complete complex tasks.

When it comes to human safety during HRC, a collision detection method must be able to be applied iteratively and have a near real-time ability to identify collisions. Furthermore, such a method should be able to look far ahead in the robot's trajectory to identify future collisions with enough time for the robot to plan the most optimal response to avoid identified collisions. For this reason, the method proposed in this paper focuses on identifying potential collisions throughout the entire trajectory of the robot with a goal of minimizing computational time.

1. RELATED WORK

One of the simplest and fastest ways of performing a predictive collision detection is to employ "*Multiple Interference Detection"*. This is done by selecting a representative set of configurations (poses) throughout the robot's trajectory at which to check for interference. While this method is generally much faster than most other methods, it runs the risk of missing a collision due to an insufficient number of interference checks [3]. There are various techniques to help determine how often a trajectory should be checked for collision. One technique is to use the velocities of the obstacles and the distance between them to predict the earliest possible time at which a collision could happen. This prediction can be used to determine when to check for interference next [4].

One of the most common methods is to investigate "*Swept Volume Interference*". In this method, a volume is generated by sweeping an object along a trajectory and including in the volume all points occupied by the object at any given time. This can be done for multiple objects. If the volumes generated by any objects intersect and the times at which the objects pass through the intersection points in the volumes match, then a collision has been identified [3]. While this solution is attractive in that it doesn't suffer from the higher probability of missing collisions due to insufficient sampling that plagues "*Multiple Interference Detection*", it is computationally expensive and thus difficult to implement in real time or even near real time. For complex objects or sweeps that employ small step sizes, floating point error can prove to be inhibitive for even dedicated processors, let alone processors that are also controlling a robot [5]. For the purposes of predictive collision detection in dynamic environments in which the validity of a robot trajectory must be re-evaluated every time the environment changes, faster algorithms are required.

Attempts have been made to improve the computational efficiency of *Swept Volume Interference*. One proposed approach to reducing the computational cost of sweeping the volume is to work strictly with objects modeled as complex polyhedra. Since collision detection is much more efficient with convex polyhedra, algorithms have been developed to decompose concave meshes into a number of convex volumes. The convex volumes can then be used to generate a swept volume to use in a collision detection algorithm with a lower computational cost than if the concave volume were used [6]. This approach is utilized in [7] to represent a robot's swept volume during a trajectory so that collisions can be detected at a low computational expense.

Another approach compares sets of points that are contained within the respective swept volumes of multiple objects to identify collisions. A common method of constructing this set is to employ Point Membership Classification (PMC) which identifies weather a point falls within an object's boundary, on its boundary, or exterior to its boundary [8]. The points selected to represent each volume are minimized by only evaluating the boundary of the object and by decreasing the density of the points. This approach capitalizes on the tradeoff between computational efficiency, which can be maximized by minimizing the point density, and quality of the collision detection, which increases with the point density [9].

2. METHODS

The goal of this paper is to develop an algorithm that could look ahead at a robot's trajectory and predict potential collisions with dynamic objects. The proposed method seeks to combine the computational efficiency of "*Multiple Interference Detection"* with the guarantee of identifying all collisions offered by "*Swept Volume Interference*" methods. In other words, the algorithm provides continuous collision detection at a low computational cost. Without loss of generality, a serial manipulator with all revolute joints will be assumed whenever reference to a robot is made. For this paper, collision with a single, dynamic obstacle is used to evaluate and demonstrate the approach. However, it is a simple extension to add more dynamic and kinematically connected obstacles to the environment and check for collision.

2.1 Overview of the Approach

The proposed method is to represent the trajectory of the robot throughout time as an articulated swept surface. The algorithm developed only requires a few samples of the robot's position throughout time. These samples are connected sequentially with a series of Coons patches to form a continuous swept surface. Finally, the surface is offset in the two normal directions to approximate the robot's volume. This drastically reduces computation time with respect to a swept volume approach. Furthermore, the swept surface remedies the issue of missing collisions due to an insufficient frequency of sampling iterations. It allows a discrete approach to reduce the number of samples required to adequately check for collision by interpolating between the collision checks with Coons patches. This algorithm can be used to search the entire trajectory to identify all potential collisions in the robot's path, in online and real-time implementations.

In the formulation of this approach, it is assumed that a predefined joint trajectory, consisting of values for the robot's joint angles at a series of instances in time is given. The obstacle is also specified with kinematic information about its position, size, and orientation throughout time; determined from the manufacturing cell's sensor suite prediction algorithm [1].

To construct the swept surface of the robot, the approach is as follows. The joint and link locations at poses of the robot throughout its trajectory are determined with forward kinematics and are used to define boundary curves of discrete Coons patches

[10]. The discrete Coons patches describe the swept surface of the robot in the form of a point cloud containing discrete positions that are occupied by the robot and the time at which they are occupied. This is explained in greater detail in next section.

To construct the swept surface of the obstacle, the obstacle is iterated along its trajectory at discrete time steps. To account for uncertainty in the obstacle's predicted trajectory, the obstacle is superficially enlarged as time progresses. This creates a cone like point cloud describing discrete positions that are occupied and the time at which they are occupied. This concept is explained in greater detail in Section: Generation of the Obstacle Surface.

Once both surfaces, stored in the form of point clouds, have been developed, only one interference check is necessary. If the spatial and temporal data for both surfaces intersect, then a collision has been identified. Thus, the overall approach involves three general steps: generation of the robot's swept surface, generation of the obstacle's point cloud, and a collision check between the two. Discussion on how the final collision check is performed is presented in Section: Collision Check Between the Surfaces. In Section: Evaluation, a representative test cases that demonstrate and validate the approach are presented.

2.2 Generation of the Robot's Swept Surface

Two steps are required to generate the robot's swept surface. First, forward kinematics are used to determine the location of each joint at a few selected sets of joint angles throughout the entire robot's trajectory. Second, these points are used to develop boundary curves for the surface to be generated. The boundary curves are used to define discrete Coons patches, which yield the desired surface in the form of a point cloud.

To complete the forward kinematics, a coordinate system is established at each joint with the joint as its origin and the Z axis aligned with the axis of rotation. The origin of a fixed coordinate system is selected to be located at the base of the first link of the manipulator arm. The origin of the coordinate system attached to a joint, P_i, as seen from the coordinate system of the previous joint can be fully described by the pre-multiplication of a translation matrix and a rotation matrix to account for the rotation about the previous joint's axis of rotation. The formulation of the matrices in this paper utilize the Denavit-Hartenberg notation, specifically applied to the robot as done in [11]. These matrices for each joint and link pair are used to describe the location of the origin of each joint's coordinate system as seen from the previous joint's coordinate system.

It is important to note that since the link geometries are assumed to be constant, the translation matrices are constant. If cylindric or prismatic joints were present in the robot, the translation matrices would not be constant. Also, the links are assumed to be straight and aligned. If this were not the case, a simple extension would be to treat bends in the link as joints that don't rotate. While both of these assumptions can be addressed with simple extensions, for the robot of interest in this paper, this is not necessary. By multiplying together translation and rotation matrices for the current joint and all preceding joints, each joint location can be transformed from its position with respect to its coordinate system to its position in the fixed coordinate system.

A general formula for the determination of the joint locations with translation and rotation matrices is shown in Eq. (1), where T_i^{i+1} represents the location of coordinate system $i+1$ as seen from coordinate system i, R_i represents the rotation about the joint axis associated with the joint in coordinate system i, P_f^n represents the position of the joint of interest associated with coordinate system n as seen in the fixed coordinate system, and P^n represents the location of the coordinate system of the origin of the joint of interest n as seen in its own relative coordinate system. In Eq. (1), the location of the n^{th} joint as seen in the n^{th} coordinate system is transformed to the fixed coordinate system.

$$P_f^n = T_f^1 R_1 \left[\prod_{i=1}^{n-1} T_i^{i+1} R_{i+1} \right] P^n \qquad (1)$$

Each joint location as seen in a fixed coordinate system is now described as a function of the rotation angles supplied to the rotation matrices. Figure 1 depicts the identified joint locations at six different sets of joint angles along with a visualization of the actual robot in those configurations.

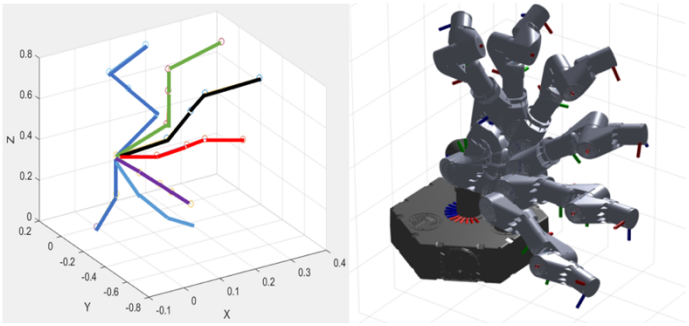

FIGURE 1: LOCATION OF THE JOINTS DETERMINED WITH FORWARD KINEMATICS

Now, the joint locations can be used to define boundary curves to serve as the edges of the swept surface. The total surface is subdivided into a series of smaller surfaces which represent the area through which each link of the robot travels from one instance in time to the next. Four boundary curves enclose each of the subdivided surfaces. The definition of these curves can be seen for one of the surfaces (Figure 2).

Let $P_{i,t}$ be the location of the joint attached to one end of a link of the robot at time t, and $P_{i,t+1}$ be the location of the same joint at a future time. Similarly, let $P_{i+1,t}$ be the location of the joint attached to the other end of a the same link of the robot at time t, and $P_{i+1,t+1}$ be the location of the same joint at a future time. The first boundary curve follows a line traced from $P_{i+1,t}$ to $P_{i+1,t+1}$. The second boundary curve travels along the link at the second instance in time from $P_{i+1,t+1}$ to $P_{i,t+1}$. The third

boundary curve follows a line traced from $P_{i,t+1}$ to $P_{i,t}$. The fourth boundary curve travels along the link at the first instance in time from $P_{i,t}$ to $P_{i+1,t}$.

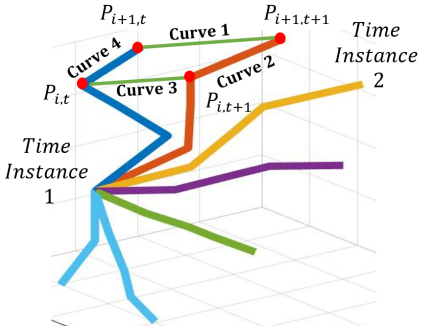

FIGURE 2: DEFINITIONS OF THE BOUNDARY CURVES FOR EACH SECTION OF THE ROBOT'S TRAJECTORY

A discrete Coons patch algorithm [10] is now implemented to define locations of intermediate points within the boundary curves. A patch is made for each dimension, X, Y, and Z. A time dimension patch is also created in which curves two and four are characterized by their known times, respectively, and curves one and three are defined as linear interpolations between the time at the first configuration and the time at the second. The four boundary curves for each patch can be described in an m x n x 4 matrix, where the four m x n layers represent each dimension (X, Y, Z) and time, respectively. In the matrix configuration for one m x n layer shown below in Eq. (2), $\boldsymbol{b}_{0,0}$ to $\boldsymbol{b}_{0,n}$ defines curve one, $\boldsymbol{b}_{0,n}$ to $\boldsymbol{b}_{m,n}$ defines curve two, $\boldsymbol{b}_{m,n}$ to $\boldsymbol{b}_{m,0}$ defines curve three, and $\boldsymbol{b}_{m,0}$ to $\boldsymbol{b}_{0,0}$ defines curve four.

$$[\boldsymbol{b}] = \begin{bmatrix} \boldsymbol{b}_{0,0} & \cdots & \boldsymbol{b}_{0,n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{b}_{m,0} & \cdots & \boldsymbol{b}_{m,n} \end{bmatrix} \tag{2}$$

The points that lie at each set of indices i and j in $\boldsymbol{b}_{i,j}$, where i and j range from 1 to m-1 and 1 to n-1 respectively, are calculated in the discrete equation of a Coons patch Eq. (3). The values for m and n can be selected to tune the resolution of the Coons patch. They define what will be referred to in this paper as the mesh. Influences of the mesh size on computational efficiency are explored in more depth in the evaluation section of this paper.

Applying Eq. (3) to every discretized point within the boundary curves defines a tightly packed point cloud representing X, Y, and Z positional data and time data for one link as it moves from one orientation to the next (Figure 3 – Left). Each link at each joint configuration can be stepped through. Patches connecting each robot configuration (pose) can be generated by Eq. (3), defining the spatial and temporal data of the robot as it sweeps through the trajectory (Figure 3 – Right) [10].

$$\begin{aligned} \boldsymbol{b}_{i,j} = {} & \left(1 - {}^{i}/_{m}\right)\boldsymbol{b}_{0,j} + {}^{i}/_{m}\,\boldsymbol{b}_{m,j} \\ & + \left(1 - {}^{j}/_{n}\right)\boldsymbol{b}_{i,0} + {}^{j}/_{n}\,\boldsymbol{b}_{i,n} \\ & - \left[1 - {}^{i}/_{m} \quad {}^{i}/_{m}\right]\begin{bmatrix} \boldsymbol{b}_{0,0} & \boldsymbol{b}_{0,n} \\ \boldsymbol{b}_{m,0} & \boldsymbol{b}_{m,n} \end{bmatrix}\begin{bmatrix} 1 - {}^{j}/_{n} \\ {}^{j}/_{n} \end{bmatrix} \end{aligned} \tag{3}$$

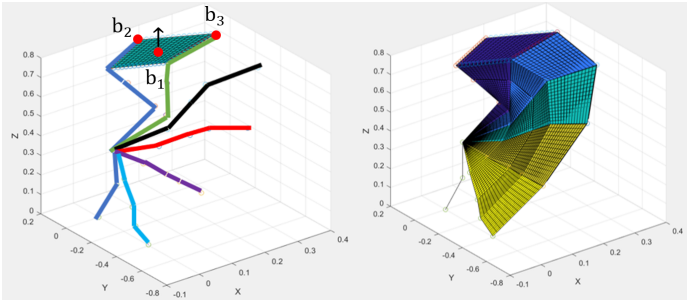

FIGURE 3: APPLICATION OF COONS PATCH TO DEFINE SURFACE SWEEP

This surface represents the area occupied by the centerline of the robot as it moves. To account for the thickness of the robot as it sweeps through its trajectory, the surface must be offset in both normal directions to bound the swept volume of the robot between two surfaces. The direction of the offset is determined by the normal vector of each Coons patch used. In each plane, three points are selected to define a unit normal vector: the center point of the patch, b_1, and two corner points on the patch, b_2, and b_3. With these three points, the unit normal vector, $\hat{n}$, is calculated Eq. (4). This is done for each surface that makes up the total swept surface.

$$\hat{n} = |(b_1 - b_2) \times (b_1 - b_3)| \tag{4}$$

The normal vector is then scaled to create a vector with a magnitude equal to half the width of the robot arm. The X, Y, and Z components of the scaled $\hat{n}$ vector are added to or subtracted from the X, Y, and Z position of each point within the patch to create each respective bounding surfaces. The two bounding surfaces allow the collision detection to take place at the approximated surface of the robot rather than at the centerline of the robot (Figure 4).

2.3 Generation of the Obstacle Surface

Once the robot trajectory has been described as a discrete surface in the form of a point cloud with temporal data, a similar methodology for obstacles is utilized to model the obstacle location over time in a computationally efficient way that can be compared with the swept surface.

The position, size, and orientation of the obstacle throughout its trajectory are assumed to be defined by the shape dimensions, the position of the centroid throughout time, and the angle by which the object is rotated about the trajectory. From the positional information, a velocity vector, v, can be approximated

in the X, Y, and Z direction by dividing the change in distance by the change in time between two positions. Similarly, an acceleration vector, a, can be approximated between each velocity vector. This kinematic information is used to derive an intrinsic coordinate system composed of a tangential (Eq. (5)), a normal (Eq. (6)) and a binormal unit vector (Eq. (7)) along the obstacle trajectory.

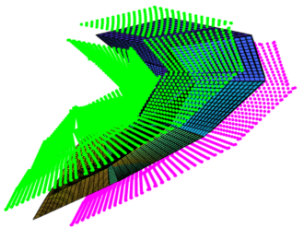

FIGURE 4: GENERATION OF OFFSET SURFACES TO ACCOUNT FOR ROBOT'S VOLUMETRIC SIZE

$$\widehat{e_t} = \frac{[v_x \quad v_y \quad v_z]}{\sqrt{{v_x}^2 + {v_y}^2 + {v_z}^2}} \tag{5}$$

$$\widehat{e_n} = \frac{[a_x \quad a_y \quad a_z]}{\sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}} \tag{6}$$

$$\widehat{e_b} = \widehat{e_t} \times \widehat{e_n} \tag{7}$$

In the intrinsic coordinate system, the obstacle, modeled as an ellipse, with a specified width and height, is generated with discrete points at the ellipse's perimeter. By using an intrinsic coordinate system, the pitch, θ, and the yaw, Ψ, are specified by the supplied predicted obstacle trajectory. The roll of angle ϕ, assumed to be supplied with the input data, is accounted for by multiplying the ellipse in the intrinsic coordinate system by a rotation matrix, R, to rotate it about the tangential axis, Eq. (8).

$$R = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

To account for uncertainty in the prediction of the object's location, the dimensions of the ellipse are set to increase proportionally with time. The rational is that the further out the prediction is made, the greater the uncertainty in the prediction. This results in a cone like shape for the swept ellipse. A scaling factor relating the time to the dimensions of the ellipse can be tuned to adjust the safety threshold of the robot.

Finally, after the ellipse has been generated in the intrinsic coordinate system, it is transformed to a fixed coordinate system via the intrinsic to fixed transformation matrix, T_f^I, formulated by aligning the intrinsic coordinate system, $\widehat{e_t}$, $\widehat{e_n}$, and $\widehat{e_b}$ with the fixed coordinate system and translating the origin of the intrinsic coordinate system to the specified location of the obstacle as seen in the fixed coordinate system x_i, y_i, and z_i. This fixed transformation matrix takes the form of Eq. (9).

$$T_f^I = \begin{bmatrix} \widehat{e_{nx}} & \widehat{e_{bx}} & \widehat{e_{tx}} & x_i \\ \widehat{e_{ny}} & \widehat{e_{by}} & \widehat{e_{ty}} & y_i \\ \widehat{e_{nz}} & \widehat{e_{bz}} & \widehat{e_{tz}} & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

Figure 5 shows a side view and a front view of a sample of an obstacle's swept volume, in which temporal data is displayed in the form of the plotted color.

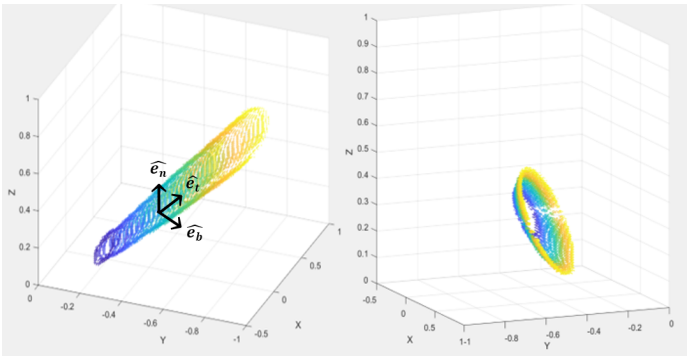

FIGURE 5: SWEPT VOLUME ALONG THE OBSTACLE'S TRAJECTORY WITH TIME DISPLAYED AS COLOR

2.4 Collision Check Between the Surfaces

Now that both the robot and the obstacle(s) have been described in the form of separate point clouds, each containing positional and temporal data, potential collisions can be detected. In order to compare the point clouds, they must be resolved to a standard 3D spatial grid. Similarly, a standard discretized time array is adopted.

Once a desired grid spacing and time step is selected, the value of each point in the point clouds is set to equal the nearest spatial grid coordinate (X, Y, Z) and standard time value. The robot's point cloud is then compared with the obstacle(s) point cloud(s), determining all common points where the positional (X, Y, Z) and time data are equal; each representing a collision. In Figure 6, the point clouds are plotted against each other. The color signifies the time at which these points are occupied.

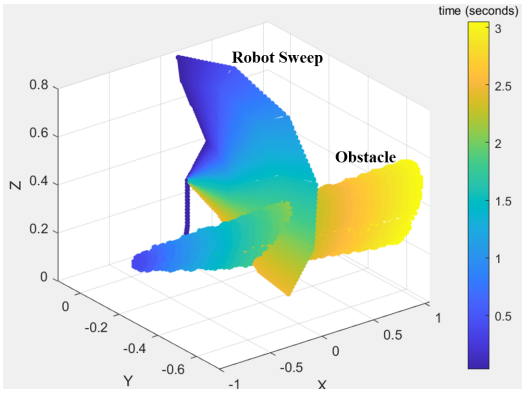

FIGURE 6: OVERLAID POINT CLOUDS: TIME AT WHICH A LOCATION IS OCCUPIED IS DISPLAYED AS COLOR (CASE A)

After comparing the standardized data, the collisions between the point clouds are identified and plotted against the swept surface (Figure 7). The red indicates a predicted collision location between the robot and the obstacle(s).

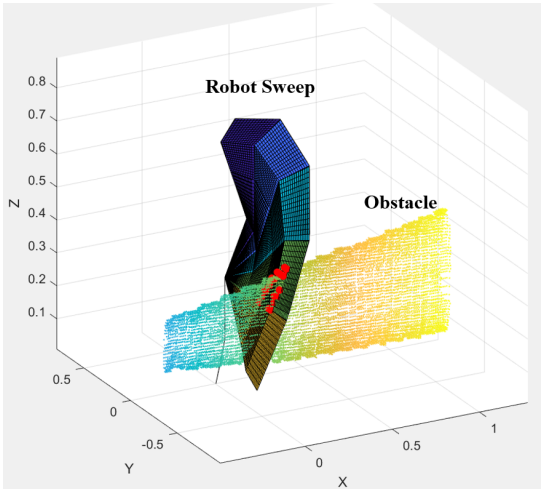

FIGURE 7: TEST CASE A: DETECTED COLLISION OF THE ROBOT AND THE OBSTACLE (DENOTED IN RED)

3. EVALUATION

Multiple test cases were developed to evaluate the speed and effectiveness of the algorithm. These test cases were implemented on an Intel® Core™ i7-7500 CPU processor.

Test Case A is shown in Figures 6 and 7. In this case, a collision was identified (shown in Figure 7 as red points). For the test shown, six poses of the robot throughout its trajectory were used to construct the sweep. In further evaluation, the term "resolution" is used to refer to the number of poses used to construct the sweep. Test Cases B and C are shown in Figure 8 and Figure 9, respectively. Test Case B demonstrates the situation in which the plot of the two swept volumes intersect; but the times at which each body is at the intersection are not the same. Therefore, there is no collision between the robot and obstacle. In Test Case C, a collision is identified, in that, both point clouds share the same temporal data value at their spatial intersection location.

3.1 Run time data for the Collision Detection Algorithm

The computational efficiency of the algorithm was evaluated at various resolutions of the sweep. At each resolution (number of robot poses used to generate the sweep), the algorithm was run ten times. The tests were timed, and their average times reported in Table I. The effectiveness of the algorithm in accurately predicting whether or not a collision will occur was determined by tracking the reported collisions at each resolution. It is assumed that higher resolution tests yield better predictions than lower resolution tests. Thus, as the resolution increases, it is expected that the algorithm will converge on a prediction. Table I reports for each test case at various resolutions whether or not a collision is identified with either a "yes" or a "no". Since all tests converged on a solution, the effectiveness of the algorithm is characterized by the lowest resolution at which the prediction converges (highlighted in Table I).

Plotting the run time versus the resolution for each test case reveals that the relation between the resolution and the run time is exponential (Figure 10). Starting at a resolution of around 40, the computational time begins to rapidly grow. Thus, for this implementation of the algorithm, the resolution must be carefully selected. As seen in Table I, the algorithm was able to successfully predict collisions at resolutions far below the resolution range at which the computational efficiency is dramatically affected.

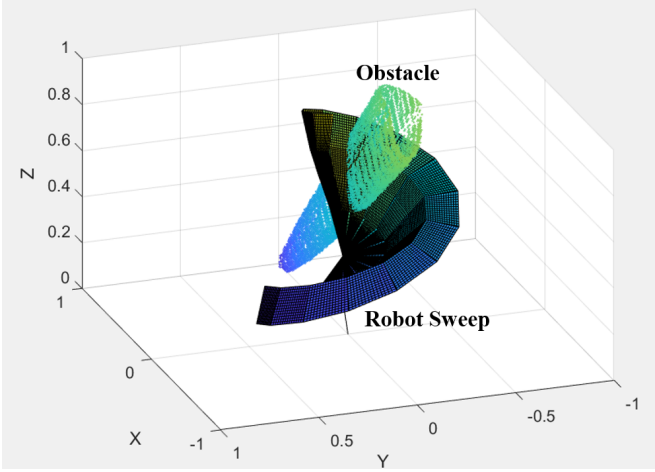

FIGURE 8: TEST CASE B: SPATIAL INTERSECTION OF SWEPT VOLUMES; BUT NOT IN TIME – NO COLLISION

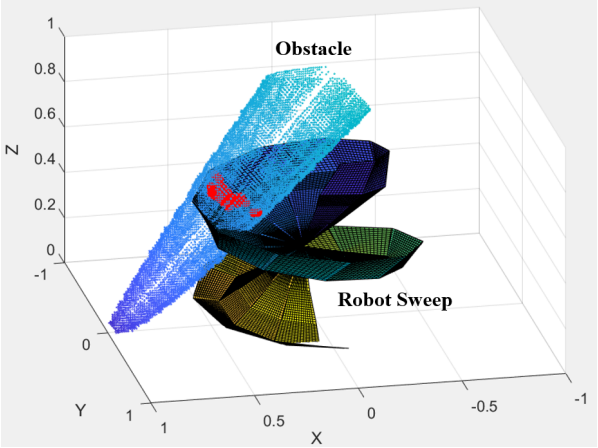

FIGURE 9: TEST CASE C: SPATIAL INTERSECTION OF BOTH SWEPT VOLUMES OCCUR AT SAME TIME – COLLISION

Greater computational efficiency can also be achieved by optimizing the mesh size of the Coons patches. The tests carried out previously were redone with a coarser mesh for the Coons patches. Rather than a (16x16) mesh, an (8x8) mesh was used in each patch. The modeled surfaces for the sweep with the (16x16) mesh and the (8x8) mesh are shown in Figure 11. An important

result was obtained with this mesh. With a coarser Coons patch mesh, the algorithm identified the collisions with the same accuracy as the finer mesh results in Table I, but at a much lower computational cost. This indicates that the mesh size should be selected in order to accurately model contours and finer detail, not simply to increase point density. Results from these tests can be seen in Table II and in Figure 10. The "Collision Detected" field was omitted as the data for Table II in this field exactly matched that of Table I. Since the algorithm was still able to demonstrate the same collision detection capabilities with a reduced Coons patch mesh, results in Table II are an indication of the computational efficiency at which this algorithm can detect collisions.

TABLE I
RUN TIME AND COLLISION DETECTION TEST RESULTS (MESH (16X16))

	Test Case A		Test Case B		Test Case C	
Resolution	Run Time (seconds)	Collision Detected	Run Time (seconds)	Collision Detected	Run Time (seconds)	Collision Detected
5	0.1297	yes	0.1266	yes*	0.1328	no**
10	0.1750	yes	0.1969	yes*	0.1813	yes
15	0.2313	yes	0.2297	no	0.2922	yes
20	0.3250	yes	0.3203	no	0.3234	yes
25	0.3719	yes	0.3344	no	0.3438	yes
30	0.4906	yes	0.4938	no	0.4625	yes
35	0.6234	yes	0.5734	no	0.5938	yes
40	0.7500	yes	0.7031	no	0.6328	yes
45	0.7891	yes	0.8078	no	0.7984	yes
50	0.9734	yes	1.0047	no	0.9766	yes
60	1.4063	yes	1.4500	no	1.4984	yes
70	1.9219	yes	2.0938	no	1.9703	yes
80	2.8344	yes	2.6953	no	2.7203	yes
90	3.7781	yes	3.8141	no	3.7781	yes
100	6.5359	yes	6.0875	no	6.0750	yes

* False positive, algorithm falsely identifies a collision
** Missed collision, algorithm fails to identify a collision

Since the coarser mesh had the same collision detection ability as the finer mesh, and since the results in Table I suggest that a resolution as low as 15 was able to accurately predict collision, much longer trajectories than those tested in this paper can be expected to be evaluated with only a proportional increase in computational time. This provides the robot with a greater ability to look ahead and respond to collisions that may occur.

The algorithm is designed to be run each time the robot creates a new plan. Once the algorithm has run once, the sweep of the robot along this trajectory can be used to re-evaluate collision with dynamic obstacles each time the environment is updated. This algorithm can serve as an auxiliary function to the main robot controller acting as a "watchdog" to predict collisions.

From the data collected on whether or not a collision was detected, the consequences of too coarse of a resolution are manifested. At low resolutions, Test Cases B and C both made an incorrect prediction as to whether or not there was a collision. Test Case A did not seem to have this problem. One explanation is that since the discrete Coons patches are essentially linear interpolations between the selected robot poses, greater resolution is required to avoid cutting off large sections of the curvature when the surface displays a high degree of curvature. To emphasize this point, note the geometric difference between Test Case C at a resolution of thirty (Figure 12 – Left), and ten (Figure 12 – Right). While both tests were able to predict collision, it is obvious that the curved nature of the surface caused the approximation to miss vital sections of the trajectory.

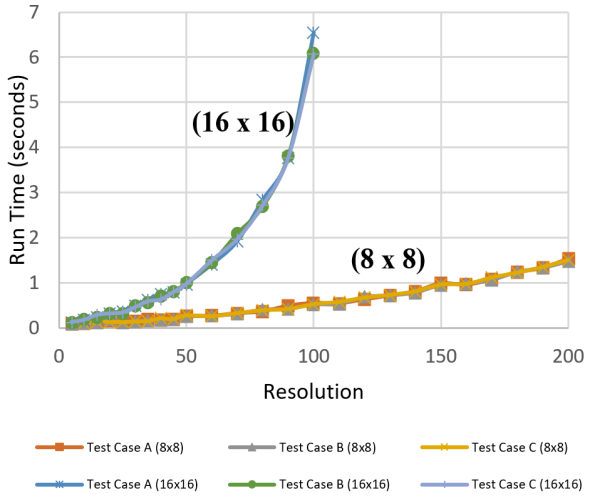

FIGURE 10: RUN TIME IN SECONDS VERSUS THE NUMBER OF ROBOT CONFIGURATIONS USED TO DEVELOP THE SWEEP FOR COONS PATCH MESH SIZES (8X8) AND (16X16)

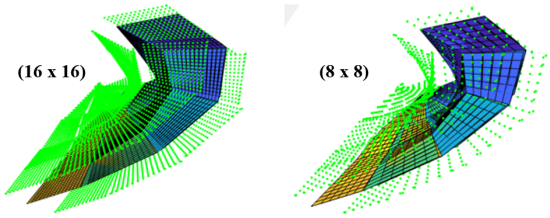

FIGURE 11: SURFACE CONSTRUCTED WITH TWO MESH SIZES

TABLE II
TEST RESULTS FOR COARSE COONS PATCH MESH (8X8)

Step Size	Test Case A (seconds)	Test Case B (seconds)	Test Case C (seconds)
5	0.0922	0.0938	0.1094
10	0.0969	0.1000	0.1016
15	0.1469	0.1203	0.1422
20	0.1672	0.1891	0.1437
25	0.1344	0.1328	0.1359
30	0.1484	0.1547	0.1516
35	0.1891	0.1688	0.1594
40	0.1797	0.1781	0.2203
45	0.1922	0.1969	0.2031
50	0.2641	0.2594	0.2844
60	0.2687	0.2734	0.2656
70	0.3281	0.3141	0.3344
80	0.3656	0.4078	0.3969
90	0.4922	0.4234	0.4250
100	0.4750	0.4750	0.4797

A topic for further study will be to develop an evaluation of the curvature of a trajectory to optimally set the required resolution for accurate modeling. To further optimize the algorithm settings, the Coons patch mesh can be selected to be as course as possible without leaving spacing between points large enough for the smallest dimension of the obstacles in the environment to pass through. In other words, the minimum obstacle dimension in the environment can be used as the maximum coarseness of the Coons patch mesh. Both of these evaluations can serve as prechecks completed before execution of the algorithm to optimally set parameters by which the algorithm should execute.

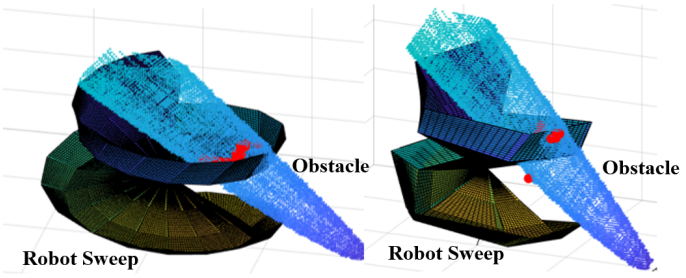

FIGURE 12: SENSITIVITY OF TEST CASE C TO INCREASED COARSENESS

CONCLUSION

This paper presents an algorithm that is able to detect collisions along lengthy robot trajectories in fractions of a second. The fundamental idea to this approach was to combine the strengths of "*Multiple Interference Detection"* and "*Swept Volume Interference"* by creating a continuous sweep of the robot motion from a limited number of sample configurations (poses) of the robot throughout the trajectory. The continuous surface is approximated by connecting each sample configuration with Coons patches. Thus, the algorithm is able to approach the collision detection robustness of a swept volume with the computational cost of taking just a few samples.

It is important to note the limitations of the work. It is assumed that the joints of the robot are precisely known. Error in the actual joint position of the robot is unaccounted for directly. It should also be noted that the compounding error of each joint will result in the maximum positional error at the end effector. One way to account for this error is to quantify the worst-case scenario positional error based on all the joints and add this error to the offsets of the surface to ensure the surface bounds the uncertain region of the robot's sweep.

In conclusion, the techniques implemented in this paper yield a fast, predictive collision detection algorithm that can be used to give the robot the power to look ahead and preemptively respond to dynamically changing environments. This work serves as a contribution to a work-in-progress to develop an effective integration of perception, cognition, and prediction data to provide real-time intelligent human-robot collaborative control in smart factories.

ACKNOWLEDGEMENTS

Funding was provided by University of Florida's Scholars Program and 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] Xiong, Q., Zhang, J., Wang, P., X. Gao, R., 2020, "Transferable Two-stream Convolutional Neural Network in Human-robot Collaboration", *48th SME North American Manufacturing Research Conference*, NAMRC 48.

[2] Nicora, M.L., Ambrosetti, R., Wiens, G.J., and Fassi, I., 2020, "Human-Robot Collaboration in Smart Manufacturing: Robot Reactive Behavior Intelligence", *ASME Manufacturing Science and Engineering Conference*, MSEC2020-8402.

[3] Jiménez, P., Thomas, F., and Torras, C., 2001, "3D Collision Detection: A Survey", *Computers & Graphics*, 25(2), pp. 1-7.

[4] Culley, R. K., and Kempf, K. G., 1986, "A Collision Detection Algorithm based on Velocity and Distance bounds", *IEEE Int. Conf. on Robotics and Automation*, San Francisco, CA, Vol. 2, pp. 1064-1066.

[5] Abrams, S. and Allen, P. K., 2000, "Computing Swept Volumes", *Journal of Visualization and Computer Animation*, 11(2), pp. 69-82.

[6] Mamou, K. and Ghorbel, F., 2009, "A Simple and Efficient Approach for 3D Mesh Approximate Convex Decomposition", *IEEE International Conference on Image Processing*, ICIP V9, pp. 3501–3504.

[7] Gaschler, A., Petrick, R. P. A., Kroger, T., Knoll, A., and Khatib, O., 2013, "Robot Task and Motion Planning With Sets of Convex Polyhedra", *RSS Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*, pp. 1-5.

[8] Erdim, H. Ilieş, and H. T., 2008 "Classifying Points for Sweeping Solids", *Computer-Aided Design,* 40(9), pp. 987–998.

[9] Ilies, H. T., 2009, "Continuous Collision and Interference Detection For 3D Geometric Models", *J. Comput. Inf. Sci. Eng.*, 9(2) pp. 1-7.

[10] Farin, G. and Hansford, D., 1999, "Discrete Coons Patches", *Comput. Aided. Gerom. Design,* 16, pp. 691-700.

[11] Crane, C. III, Duffy, J., 1998, *Kinematic Analysis of Robot Manipulators*, Cambridge University Press, New York, NY, Chapter 4, pp. 39-43.