Human-robot collaboration while Sharing Production Activities in Dynamic Environment : SPADER system

Ramy Meziane, Martin J.-D. Otis, Hassan Ezzaidi
REPARTI Center, University of Quebec at Chicoutimi
Chicoutimi, Canada
Email : meziane.ramy@gmail.com

***Abstract*—Interactive robot doing collaborative work in hybrid work cell need adaptive trajectory planning strategy. Indeed, systems must be able to generate their own trajectories without colliding with dynamic obstacles like humans and assembly components moving inside the robot workspace. The aim of this paper is to improve collision-free motion planning in dynamic environment in order to insure human safety during collaborative tasks such as sharing production activities between human and robot. Our system proposes a trajectory generating method for an industrial manipulator in a shared workspace. A neural network using a supervised learning is applied to create the waypoints required for dynamic obstacles avoidance. These points are linked with a quintic polynomial function for smooth motion which is optimized using least-square to compute an optimal trajectory. Moreover, the evaluation of human motion forms has been taken into consideration in the proposed strategy. According to the results, the proposed approach is an effective solution for trajectories generation in a dynamic environment like a hybrid workspace.**

Keywords —Trajectory generation; neural network; dynamic environment; human-robot interaction; point-to-point; geometry obstacle deformation; smooth trajectory; hybrid cell.

1. Introduction

Since the introduction of the first interactive robots in industry, which was the collaborative robots (labelled as COBOT), the field of human robot interaction has made considerable progress. In its early version, those robots were used to increase muscle strength of the operator for moving heavy loads. Recently, robots and humans can share, in the same workspace, production activities or working time [1]. However, new needs in industry require more flexibility and reactivity supporting fast change in product characteristics. One solution consists in the adaptation of an industrial robot already installed in the production line for interaction and collaboration purpose such as kinetic learning [2], assembly task [1] and adaptive third hand [3]. Furthermore, sharing workspace shall allow reducing musculoskeletal disorders when the robot absorbs shock, vibration, heavy load or avoid inadequate posture of the operator [4]. One issue for the robot is to avoid human and assembly parts while sharing production activities. Indeed, such hybrid environment is complex and dynamic, which adds an additional difficulty in both trajectory generation and path planning algorithms. Different constraints have to be taken into account for trajectories planning in dynamic environment, which are the robot dynamic and static workspaces, time constraints and real-time implementation on an Industrial Programmable Logic Controller (PLC) or other development platform such as Robotic Operating System [5].

The conventional method of collision-free and smooth trajectory in dynamic environments are based on approaches as computing of configuration space (C-space) [6] that might be difficult to achieve because of the strong nonlinearities equations of the dynamics systems and algorithms free path planning [7]. This computation complexity in trajectory could be reducing by using neural network [8]. In this work, a new method for safe trajectories generation applied in sharing production activities is presented. With respect to the state of the art in collision avoidance algorithm, the main contribution of this paper is to propose a trajectory manager algorithm applied in dynamic environment, thus allowing humans to operate in shared activities with a collaborative robot. Following a review on the state of the art in motion planning algorithms and collision avoidance, we described the primary contribution of this paper. The contribution is the design of a new method labelled SPADER for “Sharing Production Activities in Dynamic Environment” used to safely sharing workspace. The SPADER simulations show encouraging results, which are then, discussed.

2. Related work

Motion planning could be divided into two main categories [9]: path planning and trajectory planning. Path planning considers navigation strategies such as the A* algorithm [10] or road-map [11]. In contrast, trajectory planning is related to speed, acceleration, and time constraints. The following section presents a brief review on both path planning and trajectory planning. Then, a discussion one some previous strategies for collision avoidance based on neural networks is explained regarding theirs advantage and inconvenient.

2.1 Path planning while sharing production activities

Several studies have been conducted on the safe path planning applied in dynamic environment. One of the most well-known approaches is the potential fields [12, 13]. This method, introduced by Khatib in 1985 consists in creating a potential field in the robot workspace with repulsive or attractive pressures on the surface of the obstacle and the target respectively. Therefore, the robot's motion is performed according to a potential gradient [14]. However, this method could have an indetermination when both repulsive and attractive force are equal or similar [15]. In [16], they propose a strategy by computing a smooth convex envelope around the obstacle to avoid. The robot motion follows this convex shape until the goal is reached. In order to move inside a grid which each subspace is a position (or a waypoint in the following), a set of constraint links is synthesized in which a path could be computed for the end-effector.

2.2 Smoothing the trajectories

A path is characterized by a sequence of subspaces defined in both operational space (Cartesian) or joint space (configuration space). Basically, robot motion is computed using desired joint angles or Cartesian position versus time constrained by its own dynamic (motor torque and current limits). A smooth motion should be generated using for example spline functions for robotic system, UAVs [17, 18] or manipulators [19, 20]. This motion equation is characterized by spatial and temporal constraints defined by *interpolation functions*, which can satisfy the transition between two positions. In [21], they proposes an approach based on Augmented Reality. The trajectories are generated by cubic-spline using waypoints. Liu and al. [22] applies a cubic splines in Cartesian space and an B-spline in joint space. The combination of two splines interpolation allows smooth motion. A path-smoothing algorithm using the modified quadratic polynomial is proposed in [23]. The trajectory is then produced via waypoints. However, the method is applied to fixed obstacles. Generally, the global methods used to search the possible free paths in the workplace are computationally expensive when the environment is complex and dynamic. One of the solutions is using a neural network in order to find a free path. The next section presents some approaches applied in dynamic environment based on a neural network.

2.3 Neural network for obstacle avoidance

In the field of path generation using neural networks two main categories could be found, either, motion planning and collision avoidance strategies [24, 25]. Several studies are proposed in the latter. For example, Silva and al. present an approach based on a modular neural network named MONODA (MOdular Network for Obstacle Detection and Avoidance), which enables to detect and avoid the obstacle in an unknown environment [26]. MONODA architecture consists in four modules containing three layers of neural network with back-propagation. The architecture is defined so that each module corresponds to a mobile robot direction: North, South, East and West. In [27], an adaptive controller based on neural network which allows dynamic obstacle avoidance is presented and evaluated. The dynamic equation parameters of the robot are estimated when the minimum distance between obstacle and robot's segments is not respected. Another example applied to UAVs for computing a trajectory based on dynamic learning [28]. The neural network model allows producing a collision free path by giving the best direction considering the current motion. However, this approach is applied in a static environment. In dynamic environment, we cannot entirely rely on a static map while navigating. Therefore, a solution is designed by Duguleana and al. in order to manage an unknown environment. They suggest to combine two multilayer neural networks with dynamic training [29]. Neural network weights are updated throughout the movement using Q-learning (iterative method) to generate the next robot configuration (articular joint). Nevertheless, the network weights are reset whenever the maximum number of iterations is reached which could present an issue in real-time application and high velocity motion. Indeed, in a complex environment, the time may be important before reaching the target position. Of course, the approaches presented previously provide conclusive results for a specific application and could not be applied to our system. As complexity is increased when dynamic interaction occurs between robot and human in sharing production activities application, it is necessary to consider dynamic environment, obstacle geometry deformation, robot kinematic and dynamic constraints while optimazing assembly time and production cost.

2.4 Contributions

In this paper, we present a method for motion planning using neural network used for sharing production activities while assembling product components. The assembling task is shared with a robot in order to reduce the risk of an increase of musculoskeletal disorder (MSD). Then, the robot should absorb heavy load (mass), vibrations and shock in the assembling process. It should be configured in order to avoid inadequate posture and give an ergonomic workplace. The robot motion is performed inside a set of subspaces of this hybrid workspace. Then, each subspace is linked with constraints in order to generate smooth motion and avoid dynamic obstacle (human moving its limbs and components). In our suggested approach, the neural network is applied to generate a waypoint in the intermediate subspaces needed for moving the robot end-effector in a collision free path. Moreover, a set of quintic polynomials is executed in order to guarantee the continuity of velocity and acceleration, and then to reduce jerk impact on robot joints. In the rest of the paper, we will describe our *Sharing Production Activities in Dynamic Environment with Robot* (SPADER).

3. Online trajectory computation

In order to understand our suggested SPADER system for hybrid workspace management, we present first some concepts about online trajectories computation. Theses sections have been inspired by research works presented in [30-32].

3.1 Online trajectory generation

Trajectory generation for robotic systems has been extensively studied; there are several reviews discussing different types of trajectories. Generally, four classes of trajectories could be differentiated and are explained in Table 1. However, the classes two and four are mainly applied for obstacle avoidance. Indeed, the waypoints computed by the algorithm are used in order to deviate from the initial path and thus avoid the obstacles.

N	Trajectories classes	Definition
1	Operational trajectory	The movement is performed between two points with a free path in Cartesian space
2	Via-points	The displacement is realized through waypoints with a free path
3	Operational trajectory with constraint	The movement is performed with geometric constraints, as like a straight line
4	Via-Points with constraints	The trajectory is performed through waypoints with geometric constraints between points.

Table 1: Trajectories class for movements generation

The trajectory generation is completed in two main spaces, either joint space or Cartesian space. Both spaces should be considered to avoid a collision between any robot frames (limbs) or the end-effector with a human limb:

- In the joint space, the robot follows a path between two configurations by determining the values in each joint. This gives all the distances between human and robot limbs.
- Trajectories generation in the operational space allows the control of the path of the end-effector by following a defined way as a straight line or any geometric (circle or ellipse) arc. However, it is necessary to produce a set of points where a path is computed, in order to achieve a movement between two positions.

The path should be characterized by a sequence of points defined in the Cartesian and joint space. Practically, the motion generation is a function able to compute a desired set of waypoints. Those waypoints are computed following a fifth order polynomial function presented in the next section.

3.2 Quintic polynomial

The polynomial function characterizing a motion in a straight line is a linear equation versus time. This function is continuous in pose but discontinuous in velocity; therefore, it causes jerks on the articular joint of the robot. Then, higher degrees are useful in order to find a continuous velocity between each waypoint, usually third or fifth order. The third order polynomial allows the velocity constraints. In contrast, the quintic polynomial ensures also both a continuous acceleration and velocity.

Generally, the coefficients number depend to the number of constraints [32, 33]. Therefore, in our goal, the robot's end effector should move between two positions respecting constraints on position, velocity and acceleration. To satisfy these required constraints, a fifth order polynomial is more suitable for our case since six coefficients of the polynomial could satisfy these six constraints which are position, velocity and acceleration between two points (for both initial and final conditions). A sixth order polynomial could be also used to respect these constraints. However, in each computation of the collision-free trajectory, one of the coefficient has a zero value. Therefore, we have six coefficients for a sixth order polynomial. Three examples of coefficient computation for the six-order polynomial is presented in Table 2. In these simulations, the trajectories are generated from different initial and final positions (x-axis). We can observe the coefficients zero in each simulation. Therefore, the trajectories are calculated only for six coefficients which mean a five-order polynomial is sufficient for our six coefficients.

The quintic polynomial versus time (t) form is defined as follow:

$$\begin{cases} \boldsymbol{u(t)} = \boldsymbol{a_0} + \boldsymbol{a_1 t} + \boldsymbol{a_2 t^2} + \boldsymbol{a_3 t^3} + \boldsymbol{a_4 t^4} + \boldsymbol{a_5 t^5} \\ \boldsymbol{\dot{u}(t)} = \boldsymbol{a_1} + \boldsymbol{2a_2 t} + \boldsymbol{3a_3 t^2} + \boldsymbol{4a_4 t^3} + \boldsymbol{5a_5 t^4} \\ \boldsymbol{\ddot{u}(t)} = \boldsymbol{2a_2} + \boldsymbol{6a_3 t} + \boldsymbol{12a_4 t^2} + \boldsymbol{20a_5 t^3} \end{cases}, \qquad \textbf{(1)}$$

where $\boldsymbol{u(t)}, \boldsymbol{\dot{u}(t)}, \boldsymbol{\ddot{u}(t)}$ are respectively pose, velocity and acceleration vectors at the moment (t); $[\boldsymbol{a_0} \ldots\ldots \boldsymbol{a_5}]$ are unknown coefficients which will be solved according to constraints. A configuration is a movement between two points of a trajectory, which is defined as a rotation and a translation. The boundary conditions of initial u_0 and final u_f configuration are:

$$\begin{cases} u(t_0) = u_0 \\ \dot{u}(t_0) = 0 \\ \ddot{u}(t_0) = 0 \end{cases} \quad \begin{cases} u(t_f) = u_f \\ \dot{u}(t_f) = 0 \\ \ddot{u}(t_f) = 0 \end{cases}, \qquad \textbf{(2)}$$

where $\boldsymbol{t_0}$ and $\boldsymbol{t_f}$ are respectively the initial time (zero) and final time. The constraints are presented in the equation system (referenced 3):

$$\begin{cases} u_0 = a_0 \\ u_f = a_0 + a_1 t_f + a_2 {t_f}^2 + a_3 {t_f}^3 + a_4 {t_f}^4 + a_5 {t_f}^5 \\ \dot{u_0} = a_1 \\ \dot{u}_f = a_1 + 2a_2 t_f + 3a_3 {t_f}^2 + 4a_4 {t_f}^3 + 5a_5 {t_f}^4 \\ \ddot{u_0} = 2a_2 \\ \ddot{u_f} = 2a_2 + 6a_3 t_f + 12a_4 {t_f}^2 + 20a_5 {t_f}^3 \end{cases}$$

The solution is presented in the following equation system (referenced 4):

$$\begin{cases} a_0 = u_0 \\ a_1 = \dot{u}_0 \\ a_2 = \dfrac{\ddot{u}_2}{2} \\ a_3 = \dfrac{20u_f - 20u_0 - \left(8\dot{u}_f + 12\dot{u}_0\right)t_f - (3\ddot{u}_0 - \ddot{u}_f){t_f}^2}{2{t_f}^3} \\ a_4 = \dfrac{30u_0 - 30u_f + \left(14\dot{u}_f + 16\dot{u}_0\right)t_f + (3\ddot{u}_0 - \ddot{u}_f){t_f}^2}{2{t_f}^4} \\ a_5 = \dfrac{-12u_0 + 12u_f - \left(6\dot{u}_f + 6\dot{u}_0\right)t_f - (\ddot{u}_0 - \ddot{u}_f){t_f}^2}{2{t_f}^5} \end{cases}$$

Simulations	**1**	**2**	**3**
a0	4.1186	5.0000	3.0000
a1	1.2500	1.1995	0.8615
a2	-0.1451	0.7464	4.1085
a3	-1.1555	-11.0854	-4.8006
a4	1.4983	14.0465	-0.7203
a5	0	0	0
a6	-0.3400	-4.0220	1.6567

Table 2: Results of coefficients computation for trajectories generation on x-axis

The quintic polynomial is a useful tool for computing waypoint (collision free trajectory) while ensuring continuity in both velocity and acceleration. However, it is also necessary to ensure the safety of the operator in the hybrid work cell for allowing robot collaboration in shared production activities. Therefore, the next section presents our suggested system.

4. Suggested SPADER system

The proposed method illustrated in Fig. 1 suggests waypoints required for obstacle avoidance based on neural network. Then, the algorithm computes an operational trajectory between those waypoints. Moreover, the strategy takes into account a dynamic environment of moving human limbs and a deformable geometry as an envelope around the human. We shall also consider any assembly parts (moving in the robot workspace) in the dynamic environment. In the following, the human is represented by a geometric envelope: positions of the human head and wrist. Indeed, wireless sensors, labelled as wireless body area network (WBAN) [34] located on human wrist (safety glove) and head (safety helmet [35, 36]) could be used to recover these positions without camera and evaluate the human envelope geometry. Those three points are linked to a human avatar in Robotic Operating System (ROS) which give an accurate upper limbs position.

Firstly, positions are recovered by a discretization of the robot workspace. Thereof, the workspace is represented in a square matrix, which contains the positions of initial end-effector and final target where some waypoints will be computed considering the obstacle geometry. After the discretization step, data are presented to the neural network input in order to define the waypoints of trajectory. The strategy is configured such as waypoints are generated when an obstacle increase the risk of collision.

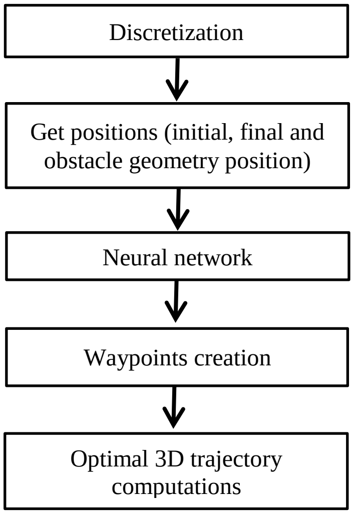

Discretization
Get positions (initial, final and obstacle geometry position)
Neural network
Waypoints creation
Optimal 3D trajectory computations

Figure 1 : The proposed method for collision avoidance

The next subsections present the hybrid cell for sharing production activities and the workspace discretization. In order to compute a set of waypoints, a neural network is developed. Finally, optimal 3D trajectory computation using quintic polynomial and least-square are presented.

4.1 Flexible manufacturing system

A flexible manufacturing system (FMS) illustrate in Fig. 2 is used where a human is installed in order to execute complex task. This hybrid workspace includes a Programmable Logic Controller (PLC), a robot, a conveyor, a distributor and a storage system [37]. The PLC is an Adventech APAX and is programmed in any usual structured language such as C++. Then, the neural network could be implemented in order to control the robot using TAG modification over IP. Robotic Operating System is also used for the libraries MoveIt! and RViz.

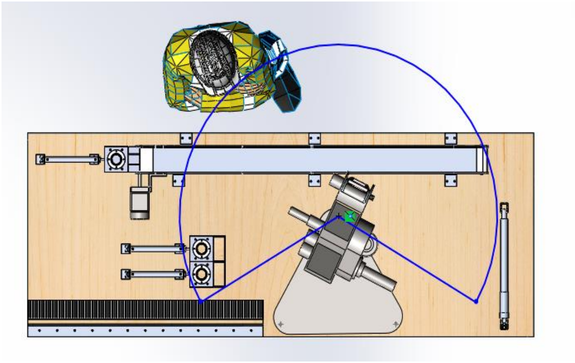


Figure 2: Representation of the Flexible manufacturing system

The recognition of human activities and localization of the operator's position were introduced in previous work [37] where inertial measurement unit (IMU) is combined with RSSI in order to compute human positions, intentions and activities. IMU could be disposed on the human wrist and shoulder in order to increase accuracy of human intentions recognition during the cooperation and to evaluate the geometry deformation of the human. Then, we derivate the minimal distance between human and robot limbs and are able to compute an estimated time to a potential collision as suggested in [38].

4.2 Workspace discretization

In our approaches, the human envelope geometry (acting as a shield) is evaluated without camera in a dynamic environment with point-to-point trajectory generations (collision-free paths). Moreover, since this algorithm is running in hard real-time, the online computation time should be minimum. Therefore, in order to respect these requirements, a discretization space is more suitable for our approaches [39-42]. A square of five by five is used to facilitate the neural network training and to cover the robot workspace for a known pick-and-place task. This workspace discretization is a trade-off between the resolution of the collision space, computation time and the complexity of the neural network architecture. Of course, the resolution could be increased. However, the resolution improvement has an impact on neural network structure. The latter is decomposed into 3D grid of square cell with five lines, five columns and five plans for a total of 125 subspaces in the robot workspace. Fig. 3 shows an example of a 3D matrix presented as inputs for the neural network. The different positions are represented by the value «0.5» and «-0.5» corresponding respectively to departure and arrival positions.

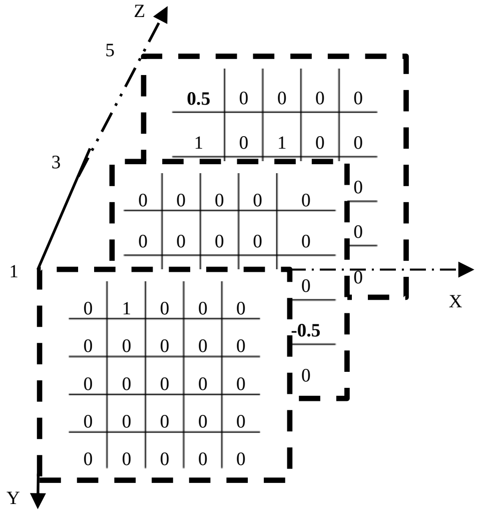


Figure 3: Workspace example discretized in 3D-Matrices

The obstacle is represented by the value «1». The remaining values «0» indicate the free space where the robot can move. Fig. 4 illustrates the output matrix corresponding to the matrix shown in Fig. 3. The value «1» represents the waypoint position, computed for avoiding a collision. However, all other elements are represented by the value «-1» in order to facilitate the neural network training. Many configurations are discretized in matrices to create a training database. Three configurations discretized in 3D grid are presented in Fig. **6**.

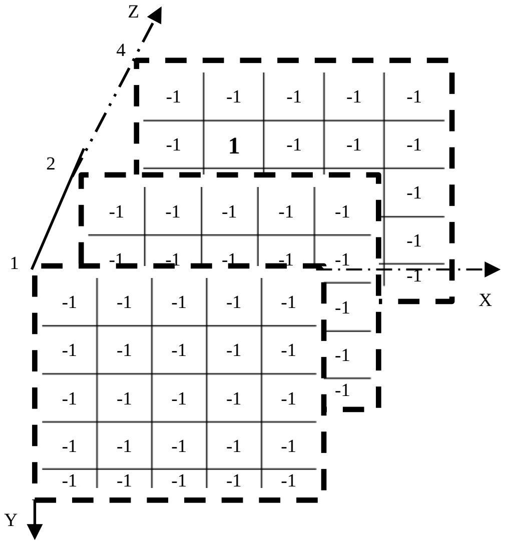


Figure 4: Output data example presented to the network for training process with a waypoint

In our approach, each grid contains 125 items with three red dots corresponding to position of wrists and head. The geometry of human form is considered in each sequence. However, the maximum length of arm coincides to the matrices limits, which are the positions (X, Y, Z) = (5, 5, 5). Then, the workspace discretization should be adapted to each configuration.

4.3 Neural Network model and training

For our system, a feedforward neural network with a back-propagation algorithm is used [43]. Three steps are defined to find the neural network model.

- First, 664 training data (input and output matrices) were used for training the neural network. The input matrices is represented in Fig. 3 and the output matrices in Fig. 4. The inputs correspond to configurations during the collaboration in assembly process with the different human movements as illustrated in Fig. **6**. In order to match the output with a collision-free path, many waypoint positions are tested. The later change randomly until to find a proper waypoints position for this configuration. For example, a first waypoint position is randomly proposed and then verified: If the collision still occurs, a new waypoints position is introduced until a valid solution is found. After verifying 664 different configurations which correspond to the robot positions during the assembly process, we finally get our database for the training. At the beginning of the training step, these data (matrices input and output) are grouped into two vectors, a vector which comprises the input data and another vector which includes the corresponding output data.
- In order to optimize the weights values in the training step, inputs and outputs data were randomly mixed inside the vectors of the 664 data. For example, according to our database, a first input and output data (matrices) should be in the first position of each input and output vector respectively. However, in our case

the first data is at the tenth position of the input and output vector. Thereafter these two vectors are presented to the network to begin the training.

- Secondly, the training step used a back-propagation algorithm that is based on the gradient descent. Then, if the output response is different than desired, this means that an error is computed. The latter is back-propagated to the hidden layers to minimize errors. During the training, the weights values are adjusted and refined continuously throughout the phase. After stopping the training phase, network performance is verified.
- Finally, several combinations of network structures were tested in order to determine the best training rate. Table 3 illustrates three best combinations.

Structures	200	150-50	150-70
R value	0.9405	0.94	0.96
MSE	0.0038	0.0034	0.0025

Table 3: Structures evaluation

The results show that the performance of each tested structure varies according the number of hidden layers and neurons. The best performance rate is R = 0.96 and MSE = 0.0025 with two hidden layers (with 150 cells and 70 cells) linking 125 inputs and outputs. The first hidden layer contains 150 cells and the second has 70 cells. For each neuron, the activation function is a sigmoid. However, the result of neural network combination and the training set correspond only 3D grid of five square cell. If the size of the grid change, a new evaluation of neural network structure should be considered.

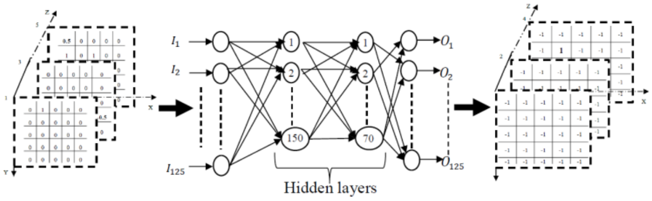


Figure 5: Neural network model

Fig. 5 illustrates a neural networks structure with two hidden layers. The input and output data are represented respectively by $[I_1, \dots\dots I_{125}]$ and $[O_1, \dots\dots O_{125}]$.

The neurons number in output level is usually predetermined by the nature of the problem. Therefore, the neurons number in the output layer is equal at 125 which correspond to the elements' number of the output matrix. After generating these waypoints, a quintic polynomial is applied in order to join them together.

4.4 Quintic polynomial with via-points and constrains

The goal is to move the robot's end effector between two positions respecting constraints on position, velocity, and acceleration. In order to achieve these conditions, a fifth order polynomial is more suitable for the respect of these constraints between each point. Therefore, the motion generation in operating space is carried out in two steps: (1) first, we define the polynomial coefficients between each position, and (2) we established a temporal evolution law to characterize the movement. The quintic polynomial is defined previously in equation (1) and the boundary conditions of initial and final configuration are presented in (2).

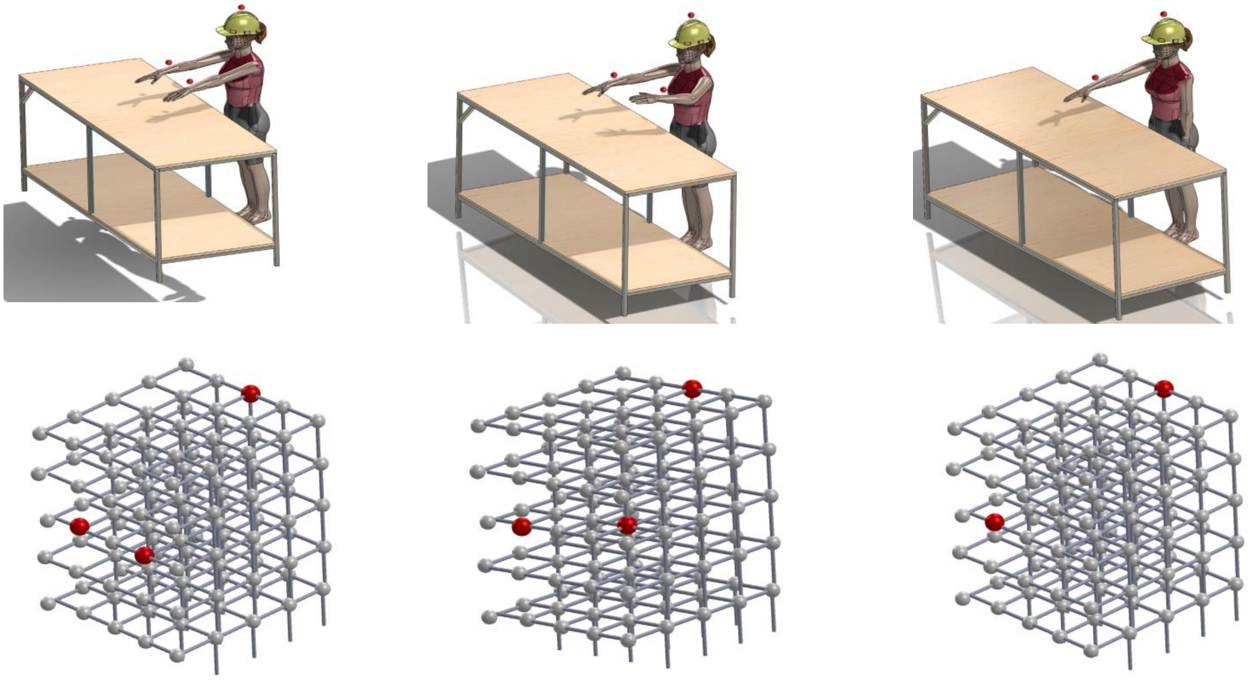


Figure 6: Human form discretized in flexible manufacturing system

However, the polynomial equation between each two position of the trajectory is presented in the system of equations (5):

$$\begin{cases} \boldsymbol{u_i} = \boldsymbol{a_{0ij}} + \boldsymbol{a_{1ij}t_i} + \boldsymbol{a_{2ij}t_i}^2 + \boldsymbol{a_{3ij}t_i}^3 + \boldsymbol{a_{4ij}t_i}^4 + \boldsymbol{a_{5ij}t_i}^5 \\ \boldsymbol{u_j} = \boldsymbol{a_{0ij}} + \boldsymbol{a_{1ij}t_j} + \boldsymbol{a_{2ij}t_j}^2 + \boldsymbol{a_{3ij}t_j}^3 + \boldsymbol{a_{4ij}t_j}^4 + \boldsymbol{a_{5ij}t_j}^5 \\ \dot{\boldsymbol{u}}_i = \boldsymbol{a_{1ij}} + 2\boldsymbol{a_{2ij}t_i} + 3\boldsymbol{a_{3ij}t_i}^2 + 4\boldsymbol{a_{4ij}t_i}^3 + 5\boldsymbol{a_{5ij}t_i}^4 \\ \dot{\boldsymbol{u}}_j = \boldsymbol{a_{1ij}} + 2\boldsymbol{a_{2ij}t_j} + 3\boldsymbol{a_{3ij}t_j}^2 + 4\boldsymbol{a_{4ij}t_j}^3 + 5\boldsymbol{a_{5ij}t_j}^4 \\ \ddot{\boldsymbol{u}}_i = 2\boldsymbol{a_{2ij}} + 6\boldsymbol{a_{3ij}t_i} + 12\boldsymbol{a_{4ij}t_i}^2 + 20\boldsymbol{a_{5ij}t_i}^3 \\ \ddot{\boldsymbol{u}}_j = 2\boldsymbol{a_{2ij}} + 6\boldsymbol{a_{3ij}t_j} + 12\boldsymbol{a_{4ij}t_j}^2 + 20\boldsymbol{a_{5ij}t_j}^3 \end{cases}$$

In order to solve this system of equations, we convert it in a matrix form:

$$\begin{pmatrix} 1 & t_i & t_i^2 & t_i^3 & t_i^4 & t_i^5 \\ 1 & t_j & t_j^2 & t_j^3 & t_j^4 & t_j^5 \\ 0 & 1 & 2t_i & 3t_i^2 & 4t_i^3 & 5t_i^4 \\ 0 & 1 & 2t_j & 3t_j^2 & 4t_j^3 & 5t_j^4 \\ 0 & 0 & 2 & 6t_i & 12t_i^2 & 20t_i^3 \\ 0 & 0 & 2 & 6t_j & 12t_j^2 & 20t_j^3 \end{pmatrix} \times \begin{pmatrix} a_{0ij} \\ a_{1ij} \\ a_{2ij} \\ a_{3ij} \\ a_{4ij} \\ a_{5ij} \end{pmatrix} = \begin{pmatrix} u_i \\ u_j \\ \dot{u}_i \\ \dot{u}_j \\ \ddot{u}_i \\ \ddot{u}_j \end{pmatrix},$$

where $[a_{0ij}, a_{1ij}, a_{2ij}, a_{3ij}, a_{4ij}, a_{5ij}]$ are the polynomial coefficients between each two positions, which will be solved. In order to have continuity in velocity and acceleration between waypoints, the following conditions are imposed for the transition between them:

$$\begin{cases} \dot{\boldsymbol{u}}_i(\boldsymbol{t_{if}}) = \dot{\boldsymbol{u}}_j(\boldsymbol{t_{j0}}) \\ \ddot{\boldsymbol{u}}_i(\boldsymbol{t_{if}}) = \ddot{\boldsymbol{u}}_j(\boldsymbol{t_{j0}}) \end{cases} \quad \textbf{(6)}$$

where $\boldsymbol{i}$ and $\boldsymbol{j}$ correspond to positions with $\{\boldsymbol{i}, \boldsymbol{j}\} \in [\boldsymbol{1}, \boldsymbol{b}]$; $\boldsymbol{b}$ is the number of maximum positions included waypoints, initial and final position; $\boldsymbol{t_{if}}$ and $\boldsymbol{t_{j0}}$ are respectively the final time in position $\boldsymbol{i}$ and initial time at the position $\boldsymbol{j}$ (next position).

4.5 3D trajectory optimization

After the computation of the fifth order polynomial coefficients, an optimal trajectory using the least-squares, which allowed reducing the length of the path between the initial and final position is computed. Fig. 7 and Fig. 8 illustrate an optimal and no-optimal path. The difference of position is presented in Fig. 9.

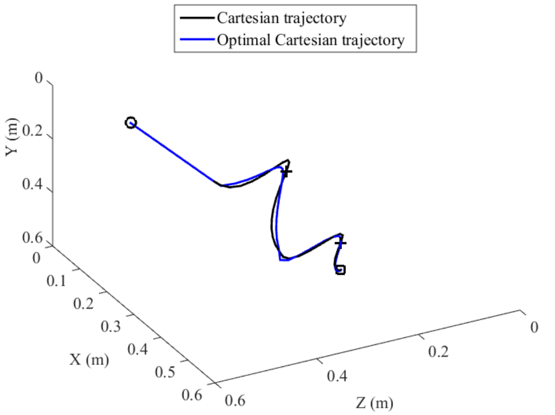


Figure 7: 3D trajectory generation with one waypoint for comparison between an optimal and no-optimal path

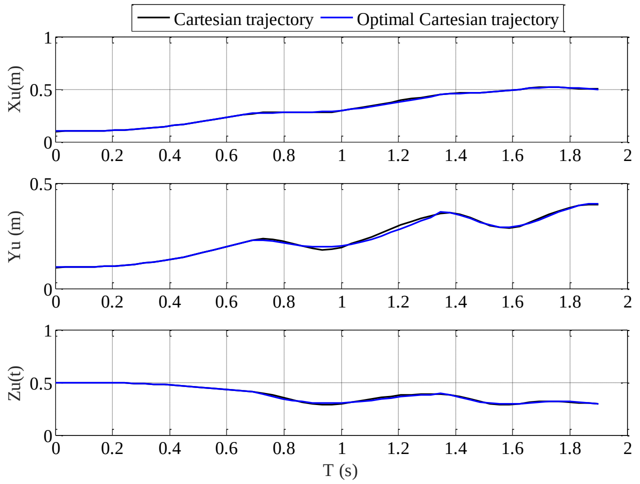


Figure 8: Trajectory evaluation during time with comparison optimal and no-optimal path

The model is represented as following:

$$\begin{pmatrix} \boldsymbol{1} & \boldsymbol{t_1} & \boldsymbol{t_1}^2 & \boldsymbol{t_1}^3 & \boldsymbol{t_1}^4 & \boldsymbol{t_1}^5 \\ \boldsymbol{1} & \boldsymbol{t_2} & \boldsymbol{t_2}^2 & \boldsymbol{t_2}^3 & \boldsymbol{t_2}^4 & \boldsymbol{t_2}^5 \\ \boldsymbol{1} & . & . & . & . & . \\ \boldsymbol{1} & . & . & . & . & . \\ \boldsymbol{1} & \boldsymbol{t_n} & \boldsymbol{t_n}^2 & \boldsymbol{t_n}^3 & \boldsymbol{t_n}^4 & \boldsymbol{t_n}^5 \end{pmatrix} \times \begin{pmatrix} \widehat{\boldsymbol{a_{0ij}}} \\ \widehat{\boldsymbol{a_{1ij}}} \\ \widehat{\boldsymbol{a_{2ij}}} \\ \widehat{\boldsymbol{a_{3ij}}} \\ \widehat{\boldsymbol{a_{4ij}}} \\ \widehat{\boldsymbol{a_{5ij}}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u_1} \\ \boldsymbol{u_2} \\ \vdots \\ \boldsymbol{u_n} \end{pmatrix}$$

The trajectory's coefficients are estimated between each two positions to achieve the best trajectory.

$$\widehat{\boldsymbol{\theta_{ij}}} = (\boldsymbol{H^T H})^{-1}\boldsymbol{H^T Y} \quad \textbf{(7)}$$

where $\widehat{\boldsymbol{\theta_{ij}}}$ is vector of the polynomial coefficients $[\widehat{\boldsymbol{a_{0ij}}} \dots \widehat{\boldsymbol{a_{5ij}}}]$; $\boldsymbol{Y}$ represent the vector of trajectory points computed from the quintic polynomial $[\boldsymbol{u_1} \dots \boldsymbol{u_n}]$; $\boldsymbol{H}$ is time matrix; $\boldsymbol{n}$ is the number of trajectory points.

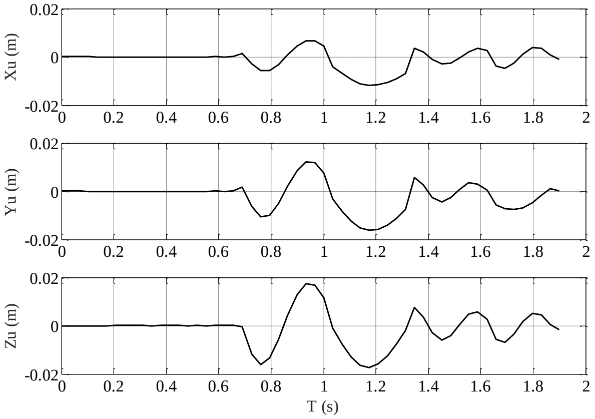


Figure 9: Graph of position error calculated form an optimal and no-optimal path

In conclusion, this strategy allows considering the dynamic environment, the evaluation of obstacle geometry deformation and the robot's constraints. Simulations on a hybrid workspace are discussed in the next section to evaluate the proposed strategy.

5. Simulation of the SPADER algorithm

In this section, simulations are used in order to verify and evaluate our proposed method. The study considered for shared production activities consist in semi-automated assembly in flexible manufacturing system. The robot was installed in order to reduce work-related musculoskeletal disorder (MSD). The simulation process has been divided in three steps. First, the discretized matrices are created such as presented in section 4.2. Second, the neural network training is performed. Finally, last step consists in an algorithm which generates a safe trajectory through the waypoints found by the neural network. Now that the algorithm is trained, we propose a solution for determining in real time the waypoint and trajectory in the next subsection.

5.1 Algorithm trajectory generator

The algorithm mentioned below (Fig. 11) is run as follows. First, the position of each parameter is recovered, either the initial, final and obstacle position.

Second, collisions detection is performed in each points of the trajectory using an interference-estimated time of arrival algorithm as suggested in [38]. This is determined by computing the distance between human avatar upper limbs and robot limbs as illustrated in Fig. 10. If the distance is smaller than the sum of radii and a threshold distance, a collision occurs. The distance is analyzed according the following function:

$$\boldsymbol{n_{md}} = \boldsymbol{n_{c2}} \times \boldsymbol{n_{c1}}$$

$$\boldsymbol{L_{c1} n_{c1}} + \boldsymbol{L_{md} n_{md}} - \boldsymbol{L_{c2} n_{c2}} = \boldsymbol{P_{m2}} - \boldsymbol{P_{m1}}$$

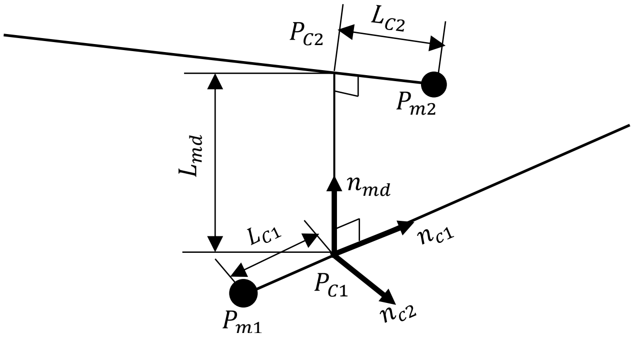


Figure 10: Minimum distance computation

where L_{c1} is the distance between wrist point P_{m1} and P_{c1} the intersection with common perpendicular vector $L_{md}n_{md}$. L_{c2} is the distance between robot wrist point P_{m2} and P_{c2} the intersection with common perpendicular vector $L_{md}n_{md}$. Depending of the result of this function, the neural network generates waypoints. In order to sort waypoints, the distance between the final position and each waypoint is computed and compared. The last intermediate point is the nearest of the final position. Until last waypoint created, the polynomial coefficients and 3D trajectory are computed with quintic polynomial between each two points. Then, the optimal trajectory is estimated.

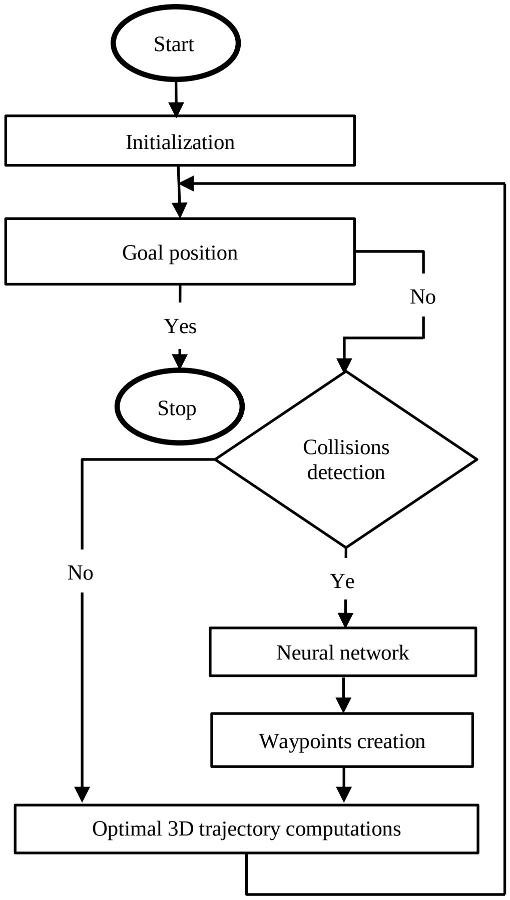


Figure 11 : Suggested motion planning algorithm in dynamic environment

6. Results and discussion

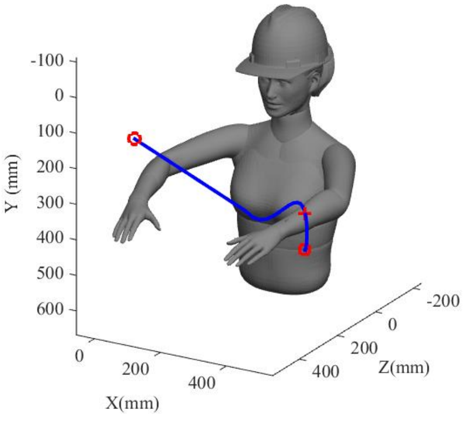
Several simulations were performed to validate the trajectory generation algorithm. According the simulation parameters, two cases have been achieved: (1) static simulations are presented first, (2) then dynamic simulation is demonstrated in order to show the behavior and the ability of the proposed method when a human arm is moving while a collision is detected.

6.1 Simulations parameters

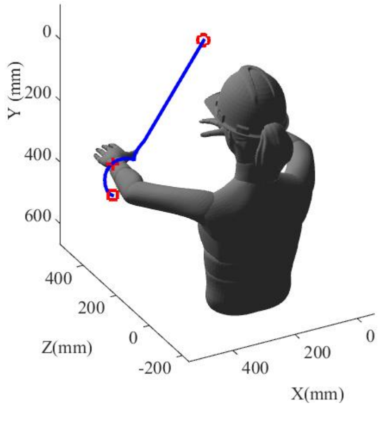
In each simulation, the operator avatar is represented according to the positions detected of two wrists and one head. Waypoints are represented by the sign (+) color red while round (o) and square (□) are respectively the initial and final (target) positions. According to ISO 10218-1 standard [44] for cooperative work, the maximum speed is fixed to 250 mm/s. The time between initial and final positions is given by the travelled distance by the end-effector and the maximal articular acceleration (motor torque) of the robot. For our setup, the travel time is computed by a basic equation which corresponds to distance covered per speed ($T = \frac{dist}{speed}$). Then, the maximum acceleration is computed by maximum speed per time. However, in our simulations, we consider a 3D grid which is formed by square cell of one meter square. In other terms, a square $(5\times5\times5)$ consist in $(0.5\,m\times0.5\,m\times0.5\,m)$. The conditions in each waypoint generated are defined by a maximum velocity and an acceleration in order to have an optimal operating from the manipulator.

6.2 Static simulation without human motion

In this section, three configurations are proposed. 3D velocity and 3D acceleration of each configuration are also illustrated. Fig. 12 presents one configuration with a departure position which corresponds to 100 millimeters (mm) in $\boldsymbol{x}$ and $\boldsymbol{y}$, 500 mm in $\boldsymbol{z}$. The final position is 500 mm in $\boldsymbol{x}$, 400 mm in $\boldsymbol{y}$ and 300 mm in $\boldsymbol{z}$. The algorithm has generated a waypoint in position 500 mm in $\boldsymbol{x}$, 300 in $\boldsymbol{y}$ and $\boldsymbol{z}$ axis in order to avoid the collision. Fig. 13 presents a second configuration with the same initial position as the first configuration, and a final position which correspond to 500 mm in $\boldsymbol{x}$, 400 mm in $\boldsymbol{y}$ and $\boldsymbol{z}$ axis. However, two waypoints are created for collision avoidance. Fig. 14 and Fig. 16 illustrate respectively the 3-axis graph of velocity and acceleration of both configuration. We see the velocities and accelerations are continuous in time. In the following example, we have fixed the maximum velocity and try to find the optimal time and acceleration. In each simulation, the algorithm generates waypoints needed to avoid collisions. However, the trajectory might don't follow a path through the waypoint, if the collision will not occur.

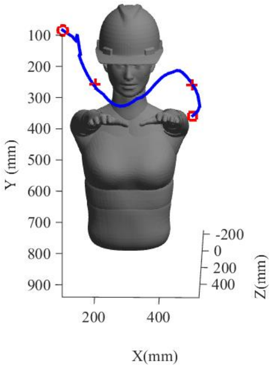


(a) Front view

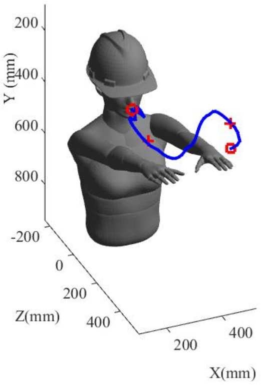


(b) Back view

Figure 12: Optimal 3D trajectory with one waypoint



(a) Front view



(b) Side view

Figure 13: Optimal 3D trajectory with two waypoints

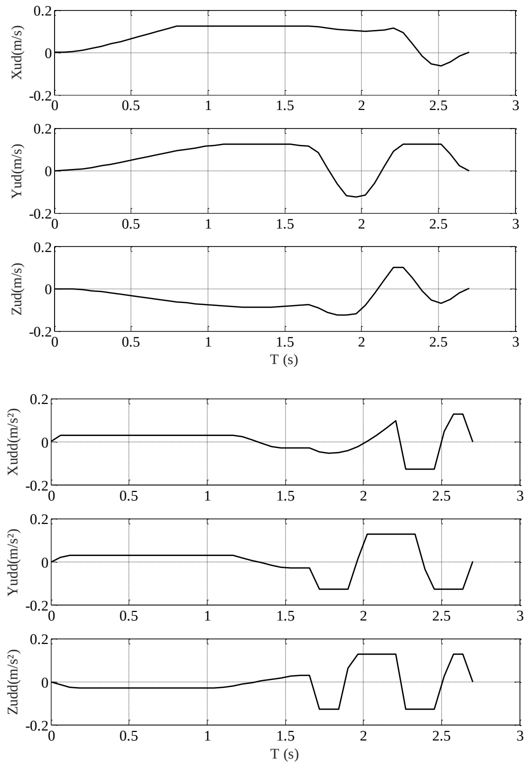
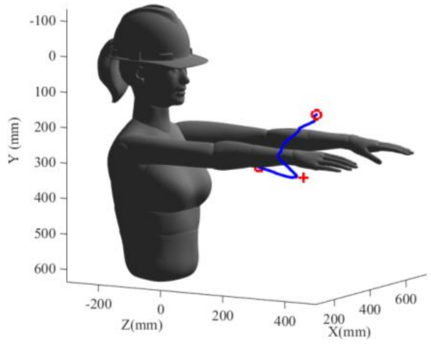
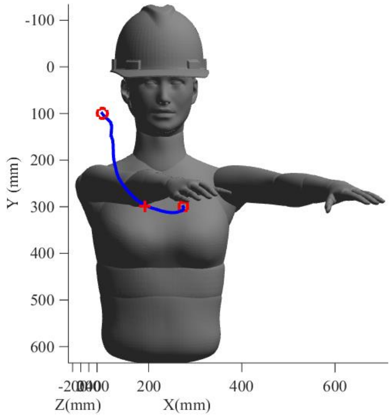


Figure 14: 3D speed and acceleration of first simulation

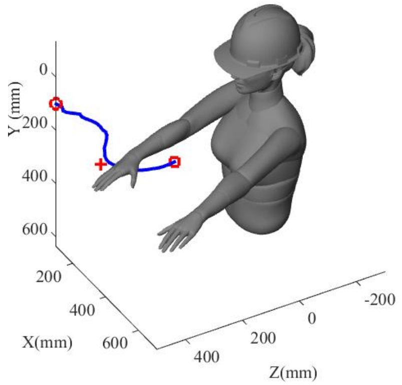
Indeed, after each move the minimum distance between human and robot limbs is computed according to collisions' detection function. If this distance is respected during the travel, a new trajectory is computed from the present trajectory point to final one. An example where the trajectory doesn't cross the waypoint is illustrated in Fig. 15.



(a) Side view



(b) Front view



(C) Side view

Figure 15: Optimal 3D trajectory with not reached waypoint

The next section presents a simulation with the human upper limb moving in the robot trajectory (dynamic simulation) which contains three human motions (three configurations).

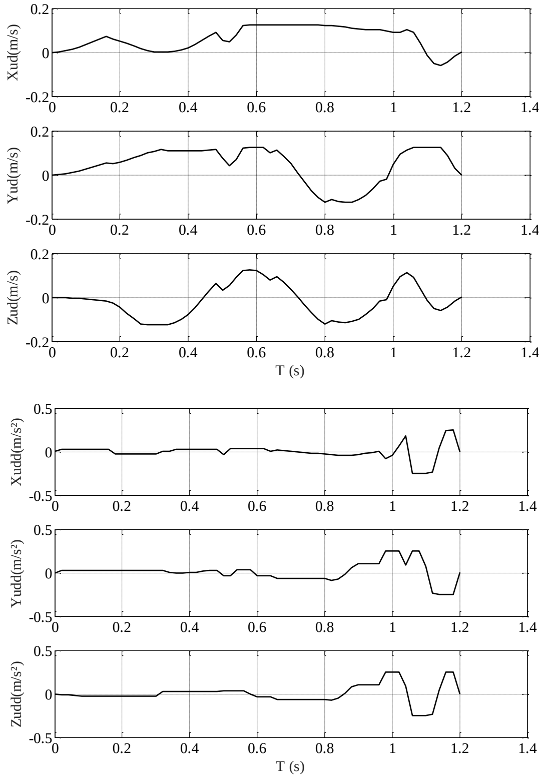


Figure 16: 3D speed and acceleration of second simulation

6.3 *Dynamic correction of the trajectory*

In this section, a dynamic simulation with moving obstacle is presented. Fig. 17 shows the simulation result including three configurations with an online adaptation of the trajectory. The result is presented in nine divided pictures in Fig. 17.

- A first configuration of human is located in the picture (a). The trajectory is computed from the initial position to the final one.
- Then the human arms move which leads to the second picture (b).
- When the minimum distance is reached, one waypoint is generated in order to avoid the collision with the operator left arm in (c).
- The real-time correction of the trajectory is applied and illustrated in (d). The continuity of the old path which corresponds to the initial human form is also presented in this picture.
- The operator moves again and human's geometric form have changed (e).
- The method results in a new smooth obstacle-avoidance trajectory. Another waypoint is generated when the minimum distance is not respected (f).
- In (g), it is shown the continuity of the oldest path.

Finally, the final position is reached and collisions have been avoided. In (h) and (i) illustrate the final simulation with different views. In conclusion, the proposed method allows the collision avoidance during sharing production activities in hybrid workspace like assembly task in cooperation. The only strategy drawback is the training data collection, which could be time consuming depending of industrial application. However, this training is only done at the installation of the robot and work cell or when the task is modified.

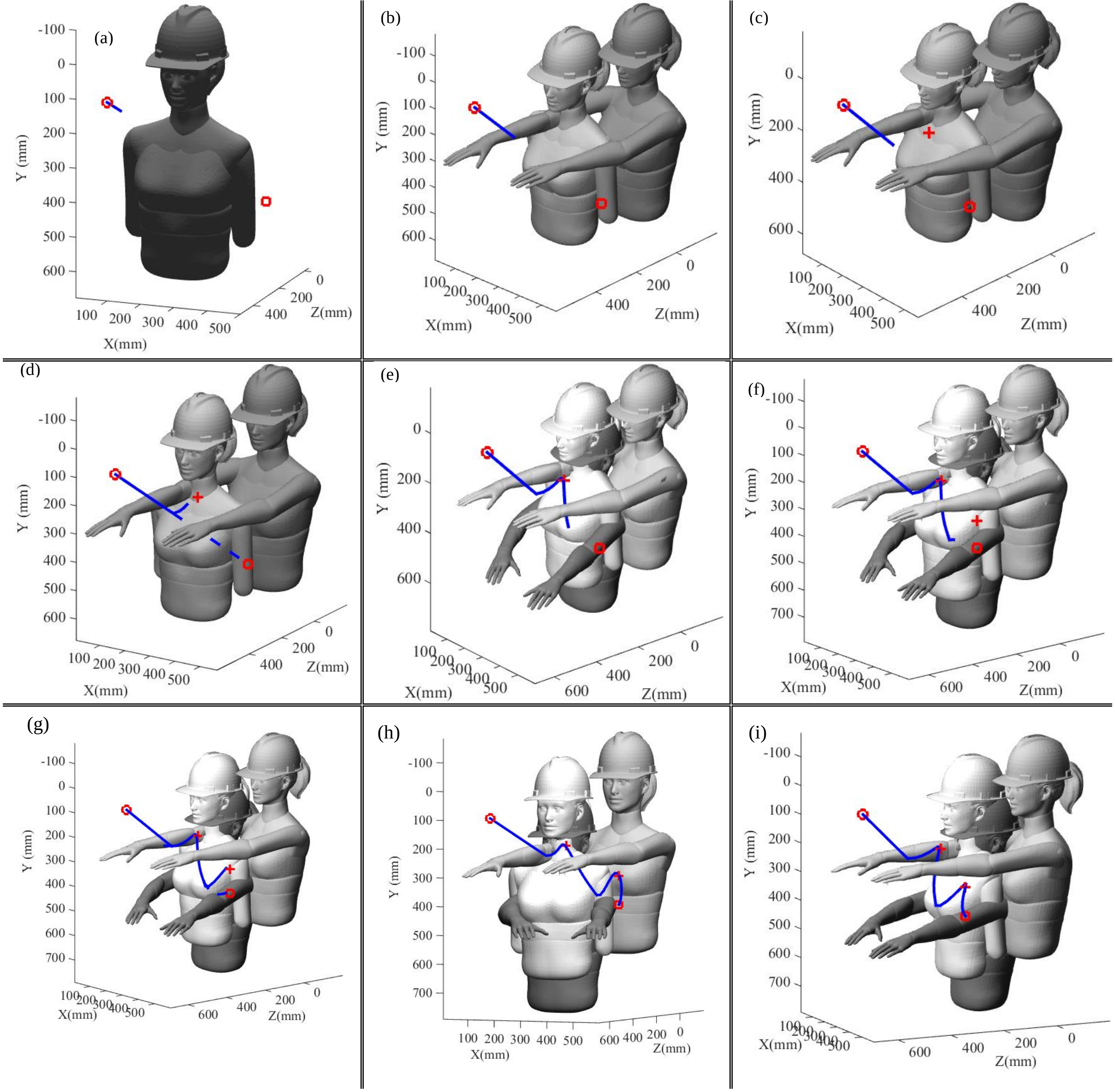


Figure 17: Dynamic simulation with optimal 3D trajectory correction

7. Conclusion

In this paper, a new approach for collisions avoidance during sharing production activities is presented. An assembly cooperation task on a flexible manufacturing system is proposed in order to simulate our strategy. The method consists in trajectories generating adapted for an industrial manipulator in order to produce a safe path in hybrid workspace. The obstacle geometrics, dynamic environment and smooth path are considered in our approach. The initial and the final positions are connected through waypoints in Cartesian space with quintic Polynomial function. A neural network back-propagation allows creating waypoints required for dynamic obstacles avoidance. Moreover, an optimization trajectory is applied to define the optimal path. According to the result, our strategy is a functional solution for trajectory generation in a dynamic environment like a hybrid work cell.

In future work, the proposed strategy should be integrated in safety interactive system which may lead to an intentional physical contact for kinesthetic teaching or third hand robotic applications.

Acknowledgments

This work is supported by the financial support of the Natural Sciences and Engineering Research Council *of Canada (*NSERC*), under the grant number 418235-2012. We would like to thanks Automation and 3D Multimodal Intelligent Interaction (LAIMI) Laboratory at University of Quebec at Chicoutimi (UQAC).*

REFERENCES

[1] J. Krüger, T. K. Lien, and A. Verl, "Cooperation of human and machines in assembly lines," *CIRP Annals - Manufacturing Technology,* vol. 58, no. 2, pp. 628-646, 2009.

[2] K. Kronander, and A. Billard, "Learning Compliant Manipulation through Kinesthetic and Tactile Human-Robot Interaction," *IEEE Transactions on Haptics,* vol. 7, no. 3, pp. 367-380, 2014.

[3] G. Maeda, M. Ewerton, R. Lioutikov, H. Ben Amor, J. Peters, and G. Neumann, "Learning interaction for collaborative tasks with probabilistic movement primitives," in 4th IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2014, pp. 527-534.

[4] G. Reinhart, R. Spillner, and Y. Shen, "Approaches of Applying Human-Robot-Interaction-Technologies to Assist Workers with Musculoskeletal Disorders in Production," *Intelligent Robotics and Applications*, Lecture Notes in Computer Science, pp. 74-84: Springer Berlin Heidelberg, 2012.

[5] B. L. Esperance, and K. Gupta, "Safety Hierarchy for Planning With Time Constraints in Unknown Dynamic Environments," *IEEE Transactions on Robotics,* vol. 30, no. 6, pp. 1398-1411, 2014.

[6] X. J. Wu, J. Tang, Q. Li, and K. H. Heng, "Development of a configuration space motion planner for robot in dynamic environment," *Robotics and Computer-Integrated Manufacturing,* vol. 25, no. 1, pp. 13-31, 2009.

[7] S. David, C. Hoam, K. Hyoun Jin, and S. Shankar, "Autonomous Exploration In Unknown Urban Environments For Unmanned Aerial Vehicles," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Guidance, Navigation, and Control and Co-located Conferences: American Institute of Aeronautics and Astronautics, 2005.

[8] A. Khoukhi, L. Baron, M. Balazinski, and K. Demirli, "A hierarchical neuro-fuzzy system to near optimal-time trajectory planning of redundant manipulators," *Engineering Applications of Artificial Intelligence,* vol. 21, no. 7, pp. 974-984, 2008.

[9] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Trajectory Planning in Robotics," *Mathematics in Computer Science,* vol. 6, no. 3, pp. 269-279, 2012.

[10] S. M. Persson, and I. Sharf, "Sampling-based A* algorithm for robot path-planning," *The International Journal of Robotics Research,* vol. 33, no. 13, pp. 1683-1708, 2014.

[11] S. H. Tang, W. Khaksar, N. B. Ismail , and M. K. A. Ariffin, "A Review on Robot Motion Planning Approaches," *Pertanika J Sci & Technol,* vol. 20, no. 1, pp. 15-29 2012.

[12] P. Chotiprayanakul , D. Liu, D. Wang, and G. Dissanayake, "Collision-Free Trajectory Planning for Manipulator Using Virtual Force based Approach," in International Conference on Engineering, Applied Sciences, and Technology (ICEAST), Bangkok, Thailand, 2007, pp. 4.

[13] M. Geravand, F. Flacco, and A. De Luca, "Human-robot physical interaction and collaboration using an industrial robot with a closed control architecture," in IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 2013, pp. 4000-4007.

[14] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in IEEE International Conference on Robotics and Automation, 1985, pp. 500 - 505.

[15] F. Flacco, T. Kroger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in IEEE International Conference on Robotics and Automation (ICRA), RiverCentre, Saint Paul, Minnesota - USA, 2012, pp. 338 - 345.

[16] Khansari-Zadeh, M. Seyed, and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots,* vol. 32, no. 4, pp. 433-454, 2012.

[17] A. Agudelo-Toro, and C. Velez, "Transition Management for the Smooth Flight of a Small Autonomous Helicopter," *Journal of Intelligent and Robotic Systems,* vol. 58, no. 1, pp. 69-94, 2010.

[18] Z. Yongguo, H. Xiang, F. Wei, and L. Shuanggao, "Trajectory Planning Algorithm Based on Quaternion for 6-DOF Aircraft Wing Automatic Position and Pose Adjustment Method," *Chinese Journal of Aeronautics,* vol. 23, no. 6, pp. 707-714, 2010.

[19] H.-I. Lin, "A Fast and Unified Method to Find a Minimum-Jerk Robot Joint Trajectory Using Particle Swarm Optimization," *Journal of Intelligent & Robotic Systems,* vol. 75, no. 3-4, pp. 379-392, 2014.

[20] L. Biagiotti, and C. Melchiorri, "Online trajectory planning and filtering for robotic applications via B-spline smoothing filters," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 5668-5673.

[21] H. C. Fang, S. K. Ong, and A. Y. C. Nee, "Interactive robot trajectory planning and simulation using Augmented Reality," *Robotics and Computer-Integrated Manufacturing,* vol. 28, no. 2, pp. 227-237, 2012.

[22] H. Liu, X. Lai, and W. Wu, "Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints," *Robotics and Computer-Integrated Manufacturing,* vol. 29, no. 2, pp. 309-317, 2013.

[23] S. R. Chang, and U. Y. Huh, "A collision-free G2 continuous path-smoothing algorithm using quadratic polynomial interpolation," *International Journal of Advanced Robotic Systems,* vol. 11, pp. 1-10, 2014.

[24] A. Abe, "Trajectory planning for flexible Cartesian robot manipulator by using artificial neural network: numerical simulation and experimental verification," *Robotica,* vol. 29, no. 05, pp. 797-804, 2011.

[25] S. H. Tang, C. K. Ang, M. K. A. Bin Mohd Ariffin, and S. B. Mashohor, "Predicting the Motion of a Robot Manipulator with Unknown Trajectories Based on an Artificial Neural Network," *International Journal of Advanced Robotic Systems*, pp. 1, 2014.

[26] C. Silva, M. Crisostomo, and B. Ribeiro, “MONODA: a neural modular architecture for obstacle avoidance without knowledge of the environment,” in Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, 2000, pp. 334-339 vol.6.

[27] B. Daachi, T. Madani, and A. Benallegue, “Adaptive neural controller for redundant robot manipulators and collision avoidance with mobile obstacles,” *Neurocomputing,* vol. 79, no. 0, pp. 50-60, 3/1/, 2012.

[28] M. S. Kumar, and S. Rajasekaran, “A neural network based path planning algorithm for extinguishing forest fires,” *International Journal of Computer Science Issues,* vol. 9, no. 2, pp. 563-568, 2012.

[29] M. Duguleana, F. G. Barbuceanu, A. Teirelbar, and G. Mogan, “Obstacle avoidance of redundant manipulators using neural networks based reinforcement learning,” *Robotics and Computer-Integrated Manufacturing,* vol. 28, no. 2, pp. 132-146, 2012.

[30] M. Haddad, T. Chettibi, W. Khalil, and H. Lehtihet, "Trajectory Generation," *Modeling, Performance Analysis and Control of Robot Manipulators*, pp. 189-239: ISTE, 2010.

[31] W. Khalil, and E. Dombre, "Chapter 13 - Trajectory generation," *Modeling, Identification and Control of Robots*, W. Khalil and E. Dombre, eds., pp. 313-345, Oxford: Butterworth-Heinemann, 2004.

[32] J. J. Craig, *Introduction to Robotics: Mechanics and Control*: Pearson/Prentice Hall, 2005.

[33] L. Chun-Shin, C. Po-Rong, and J. Y. S. Luh, “Formulation and optimization of cubic polynomial joint trajectories for industrial robots,” *IEEE Transactions on Automatic Control,* vol. 28, no. 12, pp. 1066-1074, 1983.

[34] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester, “A survey on wireless body area networks,” *Wireless Networks,* vol. 17, no. 1, pp. 1-18, 2011.

[35] P. Li, R. Meziane, M. J.-D Otis, H. Ezzaidi, and P. Cardou, “A Smart Safety Helmet using IMU and EEG sensors for worker fatigue detection,” in IEEE International Symposium on RObotic and SEnsors Environments (ROSE), Timisoara - Romania, 2014.

[36] E. Barkallah, M. J.-D Otis, S. Ngomo, and M. Heraud, “Measuring Operator’s Pain: Toward Evaluating Musculoskeletal Disorder at Work,” in IEEE International Conference on Systems, Man, and Cybernetics, 2015.

[37] R. Meziane, P. Li, M. J.-D Otis, H. Ezzaidi, and P. Cardou, “Safer Hybrid Workspace Using Human-Robot Interaction While Sharing Production Activities,” in IEEE International Symposium on RObotic and SEnsors Environments (ROSE), Timisoara - Romania, 2014.

[38] M. J. D. Otis, T.-L. Nguyen-Dang, D. Laurendeau, and C. Gosselin, “Interference estimated time of arrival on a 6-DOF cable-driven haptic foot platform,” in IEEE International Conference on Robotics and Automation ICRA '09 2009, pp. 1067-1072.

[39] D. Henrich, C. Wurll, and H. Worn, “Online path planning with optimal C-space discretization,” in IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications., 1998, pp. 1479-1484

[40] C. Wurll, and D. Henrich, “Point-to-point and multi-goal path planning for industrial robots,” *Journal of Robotic Systems,* vol. 18, no. 8, pp. 445-461, 2001.

[41] F. Rubio, C. Llopis-Albert, F. Valero, and J. L. Suñer, “Industrial robot efficient trajectory generation without collision through the evolution of the optimal trajectory,” *Robotics and Autonomous Systems,* vol. 86, pp. 106-112, 2016.

[42] H. Wada, A. Kanazawa, K. Konada, Y. Wakabayashi, M. Kamioka, S. Kondo, J. Kinugawa, and K. Kosuge, “Dynamic collision avoidance method for co-worker robot using time augmented configuration-space,” in 2016 IEEE International Conference on Mechatronics and Automation, 2016, pp. 2564-2569.

[43] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*: Addison-Wesley, 2005.

[44] Association Française de Normalisation, "NF EN ISO 10218-1: Norme européenne : Norme française : Robots pour environnements industriels," AFNOR, 2006.