POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Decision making in HRC combining offline path planning and Machine Learning methods

Laurea Magistrale in Ingegneria Meccanica

Author: Bianca Grieco

Advisor: Prof. Andrea Maria Zanchettin

Co-advisor: Ing. Martina Pelosi, Prof. Paolo Rocco

Academic year: 2023-2024

1. Introduction

The fourth industrial revolution marks a significant transformation towards a digitalized and automated manufacturing environment. Collaborative robotics is an essential component of Industry 4.0, enhancing Human-Robot Collaboration (HRC) by promoting a synergistic relationship. Ensuring a safe environment in this context deserves thoughtful attention, as cobots are designed to work in close proximity to humans without causing any harm. Cobots must adhere to the *ISO/TS 15066* safety specification [1]. In this context, three main challenges can be identified [4]: Safe Interaction, Intuitive Interfaces and Design Methods. Safe Interaction includes safety standards as well as guidelines for implementing robot operational modes designed for human interaction. Intuitive Interfaces refer to user-friendly technologies that allow humans to interact with robots in an effortless way, such as with direct manipulation, gestures or voice commands. Design Methods cover the development of control laws and techniques to design robotic systems that can work effectively and safely alongside human workers, focusing on collaborative tasks, responsiveness and adaptability to human actions.

This thesis falls within the Design Methods addressing the challenges associated with HRC. It focuses on the development of a control logic allowing the robot to choose actions to be performed depending on the actual human position. The approach aims at empowering the robot with the capacity to autonomously identify the path that strikes an optimal balance between ensuring a safe distance from the human worker and reducing task completion time.
An offline approach followed by an online validation is proposed. A database of admissible paths is created using the Bidirectional Rapidly Exploring Random Trees (BiRRT) algorithm to enable the robotic arm to navigate from its starting position to the designated target while avoiding static obstacles. This algorithm ensures a comprehensive coverage of the workspace by concurrently expanding two trees from opposite ends and facilitates the establishment of connections at multiple points, optimizing the path-finding process. Subsequently, a Reinforcement Learning method, in particular the Q-Learning (QL), is developed to enable the robot to dynamically select on which of the previously computed paths to travel and when to transition from a path to another, depending

on the human worker presence. The training and testing phases of the QL algorithm utilize different sets of data from the Motion Capture (MoCap) Database [2], simulating human movements across different tasks. Upon the completion of the training phase, an optimal policy is derived and subsequently utilized during the testing phase.
The novelty of this research lies in computing in advance all possible admissible paths the robot might take, as proposed in [3], as well as defining an optimal policy for the robot's navigation, overcoming online path planning methods limitation which demand substantial computational resources as decisions are made in real time as the robot moves. In the proposed approach, when the robot is active and functioning, the algorithm no longer needs to perform complex calculations to decide its path. Instead, it simply observes the current condition, which corresponds to the actual robot position and the actual human position, and refers to the dataset of paths and to the optimal policy to quickly determine the best route to take. This drastically reduces the computational load during real-time operations, as the robot is effectively matching the current situation to a solution it already knows, rather than computing a new solution from scratch.
The optimal policy is then validated on the ABB GoFa™ CRB 15000 cobot in the MeRLIn Lab at Politecnico di Milano. In the validation phase, a real-time detection of the human position guides the manipulator in choosing optimal actions to navigate the trees using the established policy.

2. Offline paths generation

For the purpose of the thesis, multiple feasible paths connecting the starting configuration to the goal while avoiding static obstacles must be available. The key point is to enable the robotic arm to switch between paths at any moment to avoid excessive proximity with the human operator. Employing the BiRRT algorithm to compute these paths is extremely advantageous. By setting a high maximum number of iterations, the algorithm allows the manipulator to explore the configuration space by simultaneously expanding two trees with root nodes at the specified start and goal configurations, generating a comprehensive graph of collision-free paths across the workspace. Its random sampling strategy allows to efficiently cover large areas of the space, focusing on unexplored regions. Figure 1 illustrates the created paths network in the Matlab simulated environment.

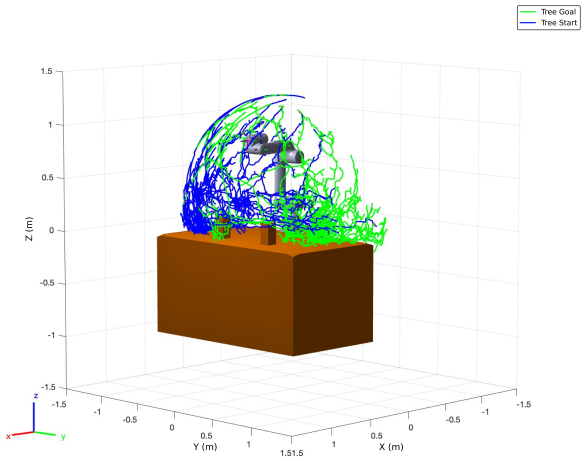

Figure 1: Result of the BiRRT algorithm

From this intricate graph of connections, it is necessary to identify only the feasible paths which effectively lead to the goal configuration, removing dead ends. Moreover, the selected paths are then simplified using the Ramer-Douglas-Peucker (RDP) algorithm to make them smoother in order to facilitate the robot's motion, as shown in Figure 2. The simplified paths include nodes which fall outside the RDP threshold and various branching and junction nodes where paths intersect or split. Furthermore, the parent-child relationships graph is derived from the trees structure, which provides a clear hierarchy of the nodes within each path.

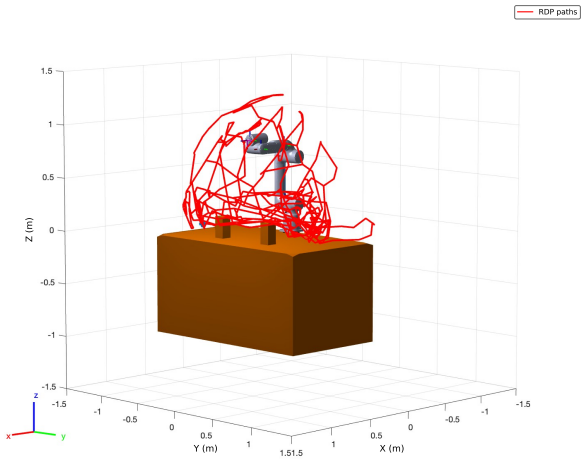

Figure 2: Result of the simplification process

3. Reinforcement learning application

The dataset of precomputed paths allows the robot to have multiple routes to choose from, especially when it reaches significant nodes in the graph. This multiplicity of routes is strategic for the subsequent learning phase, designed to offer the robot an array of navigational choices. For the training and testing phases of the QL algorithm, input data regarding human movements within the workspace are sourced from the MoCap Database. This ensures that the model accounts for a range of motion within the workspace, providing a more realistic representation of the human activity. The motion of the human worker in the workspace is depicted through a series of occupied voxels.
By learning the layout of the parent-child relationship graph and how humans move within the workspace, the robot uses the acquired knowledge optimally, becoming able to find the shortest and safer path.

3.1. Training of the QL algorithm

The parent-child relationship graph consists of *131* nodes and *201* connections between them. Moreover, the workspace is discretized in *31* voxels. In a Markov Decision Process (MDP) problem, states and actions are designed to comprehensively describe the environment to the extent necessary for decision-making purposes. Consequently, each state is defined as the combination of the node where the manipulator is located and the voxel occupied by the human's hand. This makes the number of states for the problem under analysis equal to *4.000*. The reward function quantifies the benefit or consequence of performing a specific action in a given state, serving as a critical feedback mechanism that guides the learning process. The reward function assigned to impossible actions within the current context is set to a significant penalty for attempting such actions. Instead, for feasible actions, the reward function is designed to achieve a balance between ensuring that the robotic manipulator consistently maintains a certain minimum distance from the voxel occupied by the human and minimizing the distance from the current configuration of the manipulator to the goal configuration. Algorithm 1 presents the mathematical formulation of this concept, where the parameter a is used to identify two zones of close proximity of the robot's end-effector to the voxel area.

Algorithm 1 Rewards definition

Require: $Dist_{Node\text{-}Voxel}$, $Dist_{Node\text{-}GoalNode}$, a
if $Dist_{Node\text{-}Voxel} = 0$ **then**
  $R_1 = -1.000$
else if $Dist_{Node\text{-}Voxel} \leq a$ **then**
  $R_1 = \ln\left(\frac{\text{Dist}_{\text{Node-Voxel}}}{2a}\right) \cdot \text{Dist}_{\text{Node-Voxel}} \cdot 50$
else
  $R_1 = \ln\left(\frac{\text{Dist}_{\text{Node-Voxel}}}{2a}\right) \cdot \text{Dist}_{\text{Node-Voxel}}$
end if
$R_2 = \mu \cdot \frac{1}{\text{Dist}_{\text{Node-GoalNode}}}$
$Reward = R_1 + R_2$
return $Reward$

The action selection process, reported in Algorithm 2, draws inspiration from the ϵ-greedy policy. Given the width of alternatives in terms of possible paths connecting the starting configuration to the goal, this approach allows to prioritize exploration over exploitation. Each episode of the training uses a trajectory selected randomly from the MoCap database to simulate the human's movement through the environment. The goal is to make the final model adaptable to any task by generalizing human movements during training.

Algorithm 2 Action selection process

Require: $Qtable$, $episode$, $episode_{tot}$, ϵ, $state$
$Qtable \leftarrow$ Inizialize $Qtable$ with zeros
if $episode > \frac{episode_{\text{tot}}}{2}$ **then**
  $Decay = -\log(0,1) / \left(\frac{episode_{tot}}{2}\right)$;
  $diff = episode - \left(\frac{episode_{tot}}{2}\right)$;
  $\epsilon_1 = \epsilon \cdot \exp[-Decay \cdot diff]$;
else
  $\epsilon_1 = \epsilon$;
end if
if $rand(1) > \epsilon_1$ **then**
  $NextAction \leftarrow max(Qtable(state, :))$
else
  $NextAction \leftarrow$ Choose a random action
end if
return $NextAction$

The outcome of the training phase is the Q-table. Various training sessions are performed to improve the convergence of the result, varying the number of episodes, and two different strategies are applied to obtain a fully populated matrix. The primary strategy makes the most of a fundamental characteristic of the MDP: the outcome of future states is determined only by the current state and the action taken, rather than by the history of events that preceded it. Thus, instead of starting every episode from a predetermined starting node, the algorithm is modified to introduce an element of randomness in the selection of the starting position. The second strategy involves using the knowledge of the parent-child relationship graph and, in particular, of the actions that are feasible and not feasible in each node. During the exploration phase, the next action is no longer chosen randomly from the entire range from *1* to *201*, which counts for both feasible and not feasible actions, for each node independently. Instead, for every node, the subsequent action is selected exclusively within the range of feasible actions at that specific node.
Even though the Q-table is fully populated, meaning that every state-action pair has been evaluated and assigned a specific value by the algorithm, the *7%* of the rows of the matrix display a maximum value equal to zero, indicating that all actions are considered unfeasible for those states. In such cases, the negative values indicate that, although some actions are permitted, they lead to undesirable outcomes because the robot node is either located within the same voxel occupied by the human or is extremely close to it. To address situations where no action is deemed optimal, a self-transition mechanism is proposed, allowing the robot to remain in the same node until the voxel in input changes, thereby altering the state and enabling the algorithm to discover a feasible action based on a new combination of previous node and new voxel occupied. This method ensures progress only halts when a final state is reached under safer conditions.

3.2. Testing of the QL algorithm

Once the optimal policy is defined, the algorithm is tested under new, previously non analyzed MoCap tasks. It successfully finds a solution by identifying a sequence of nodes that navigate around the voxels occupied for a specific task. The optimal Q-table serves as a decision-making guide, enabling the robot to select actions that lead to the desired outcomes. Figure 3 illustrates the manipulator's optimal path in response to a sequence of human hand-occupied voxels.

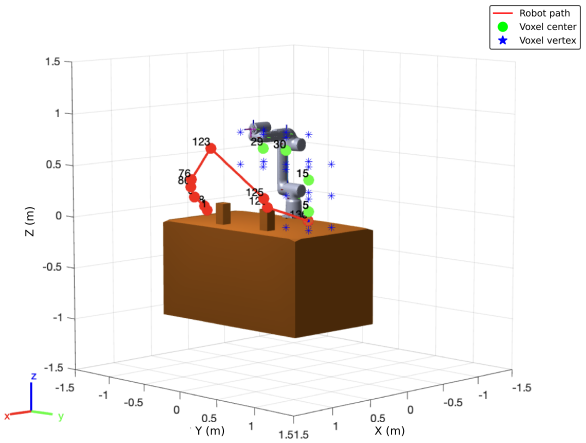

Figure 3: Manipulator's optimal path

4. Experimental validation

In the validation phase, the Kinect camera, in conjunction with an ArUco marker attached to the operator's wrist, is utilized for real-time detection of the human hand's position. Various tests have been conducted on the GoFa™ robotic arm to validate the effectiveness of the algorithm: one test without the human presence in the workspace ("No human" category), six tests with the human hand maintaining a static position in the workspace ("Static pose" category), sixteen tests with the human hand moving linearly along a single axis ("One axis" category) and six tests in which the human performs diverse tasks requiring hand movement in all three axes concurrently ("Random case" category). All the tests confirmed the absence of collisions between the operator's hand and the robot's end-effector. For a comprehensive validation of the algorithm, two distinct criteria are delineated: the time needed for the task completion and the maintenance of a minimum safety distance throughout the entire task execution.
Using the recorded robot poses and human positions over time, the data points can be visualized in Figure 4 for test *3* from "Random case" category.

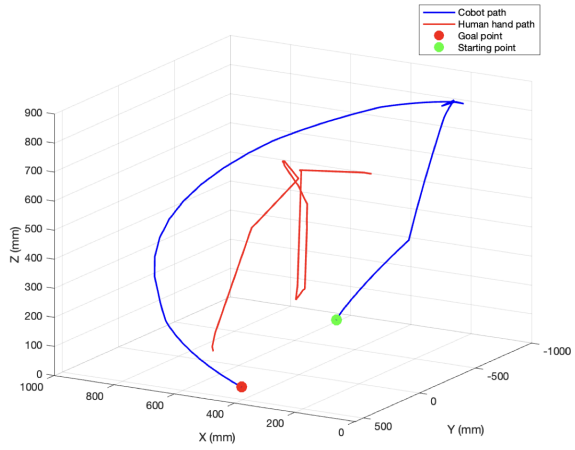

Figure 4: Cobot and human hand paths

The tests results are presented in Table 1.

Test	ID	T [s]	V [cm^3]	D_m [cm]
No human	-	18,48	0,00	0,00
Static pose	1	28,79	6,23	44,58
	2	18,48	7,93	52,88
	3	30,58	10,23	21,78
	4	39,48	17,28	36,17
	5	29,71	8,22	32,80
	6	35,95	5,89	34,36
One axis	1	50,52	145,60	30,84
	2	29,70	252,31	54,76
	3	29,14	191,24	37,81
	4	39,28	269,56	34,98
	5	35,81	160,09	16,30
	6	28,07	9,03	35,85
	7	29,00	196,64	4,34
	8	43,05	430,37	29,34
	9	42,58	686,46	44,57
	10	39,17	960,02	19,24
	11	25,13	23,61	41,39
	12	35,11	67,91	49,44
	13	35,83	118,81	47,45
	14	86,30	68,90	55,69
	15	35,81	134,47	56,83
	16	38,04	79,94	36,89
Random case	1	35,82	1147,55	31,08
	2	36,11	621,71	17,52
	3	35,85	416,96	18,49
	4	39,59	967,10	25,71
	5	22,06	49,54	36,29
	6	27,60	347,98	22,56

Table 1: Results of the validation process

The shortest time to complete the task is recorded at *18,48 s*, associated to the test where no human is detected and to the test with ID equal to *2* from the "Static pose" category, while the slowest completion time is *86,30 s*. The volume is computed by considering the sequence of point coordinates occupied by the human's hand over time as a point cloud volume. The occupancy volume varies significantly, ranging from *0* cm^3 to *1147,55* cm^3. As expected, this parameter is lower in the "Static pose" tests, where the human is confined to a single voxel, reaching its peak in the "Random case" test category as it reflects the dynamic nature of human movement within the workspace. The distance between the human hand and the robot's end-effector consistently exceeds the established safety threshold of *15 cm*, except during test *7* of the "One axis" category. This occurred because the robot was moving towards a node that was deemed safe based on its distance from the human, but the actual path came too close to the human. To prevent this, adding more nodes could help create safer paths. Additionally, the positions of the human hand were interpolated from successive Kinect frames, which might not accurately reflect real-time movements.

Figure 5 showcases the data collected from all the tests using a scatter plot. The x-axis reports the volume occupied by the human within the robot's operational space while the y-axis displays the task execution time. The correlation coefficient, calculated across all the experiments, between the task execution time and volume occupied by the human in the workspace, is equal to *0,16*. This low value suggests a weak relationship between the two variables. It is possible to conclude that the task execution time is not influenced by the nature of the test, i.e. whether the hand is stationary or in motion, but rather by the specific voxel occupied at the moment the robot is asked to make a decision. The identification number of the occupied voxel, which identifies its position in the workspace, affects the decision-making process. Moreover, the extensive range of the path network, with its numerous alternative routes, enables the robot to adjust its trajectory responsively to the human's changing location and the associated spatial occupation within the workspace.

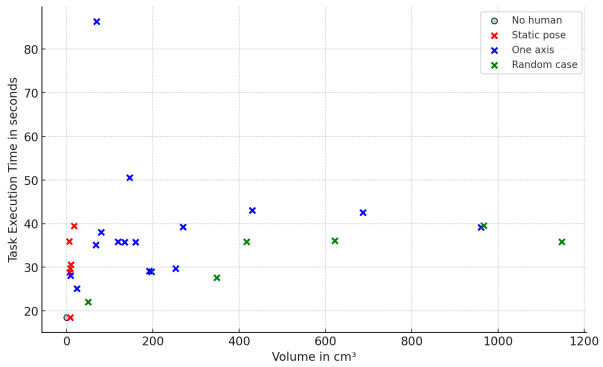

Figure 5: Data from all the test categories

The plot highlights the presence of an outlier. This point, with a a space occupation of *68,9* cm^3 and a task execution time of *86,30 s*, stands out due to its significantly longer completion time. Without this outlier, the impact of human presence on task execution time varies from no change to an increase of *173%*. Including the outlier, this increase goes up to *300%*, underlining the substantial effect human presence can have on operational efficiency. The presence of the outlier can be justified by analyzing test *14* in the "One axis" category. The robot's journey is paused at certain nodes when the next move is deemed too close to the human, based on the Q-table's safety guidelines. This careful navigation significantly contributes to the extended execution time.

5. Conclusions

In collaborative robotic environments, safety is a critical concern. It is essential for robots to employ techniques for human activities recognition as a fundamental part of their decision-making processes. The developed algorithm allows robots to detect and respond to the human presence and movements. This capability is crucial for preventing collisions and injuries, thereby enhancing safety in environments where humans and robots work closely together. Additionally, the proposed algorithm ensures an efficient execution of the task, demonstrating that safety measures can coexist with productivity in a shared workspace.

An intriguing aspect of this research is that motion planning and decision-making algorithms are entirely developed either offline or within a simulated environment to bypass complex real-time calculations. The experimental validation of the algorithm on the GoFa™ robotic arm confirmed its effectiveness, initially demonstrated during the offline testing phases, in navigating the workspace without coming into contact with the operator's hand. The algorithm enables real-time detection of human presence within the workspace, allowing the robot to alter its trajectory timely to maintain a safe distance from humans or to halt completely in the event of an imminent collision.

References

[1] International Organization for Standardization. *Robots and robotic devices – Collaborative robots, ISO/TS 15066:2016.*

[2] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. Documentation mocap database hdm05. Technical Report CG-2007-2, Universität Bonn, 2007.

[3] Stefania Pellegrinelli, Federico Lorenzo Moro, Nicola Pedrocchi, Lorenzo Molinari Tosatti, and Tullio Tolio. A probabilistic approach to workspace sharing for human–robot cooperation in assembly tasks. *CIRP Annals*, 65(1):57–60, 2016.

[4] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248–266, 2018.