Proceedings in MANUFACTURING SYSTEMS Proceedings in Manufacturing Systems, Volume 18, Issue 1, 2023, 33-38 ISSN 2067-9238

EXPLORING COLLABORATION OF HUMANS WITH INDUSTRIAL ROBOTS USING ROS-BASED SIMULATION

George-Christopher VOSNIAKOS[1),*], **Efthymios STATHAS**[2]

[1)] Prof., Manufacturing Technology Laboratory, School of Mechanical Engineering, National Technical University of Athens, Athens, Greece.
[2)] MSc graduate student, School of Mechanical Engineering, National Technical University of Athens, Athens, Greece

Abstract: *A typical 6 joint industrial robot was programmed in ROS^{TM} (Robot Operating System) environment in order to explore different scenarios of cooperation with humans. Human presence and motion are detected by simulated laser sensors. Algorithms are developed in Python for collaboration and security in the robot workspace leading to robot stopping, slowing down, or executing a certain trajectory when the human agent intervenes in its workspace whilst performing tasks such as pick-and-place or screwing. The robot is modelled in URDF format. The MoveIt!^{TM} platform is used to visualize the robot and its movements in space. The data collected by the laser sensors in the Gazebo^{tm} software environment is visualized in real time in Rviz^{TM} as the trajectory followed by the human within the workplace. The work demonstrates feasibility of human collaboration with industrial robots using ROS^{TM} as a "middleware" based alternative to collaborative robots provided that reliable sensors are involved.*

Key words: *industrial robot, human-robot collaboration, simulation, sensors, visualisation.*

1. INTRODUCTION

In human-robot collaboration in Manufacturing, safety challenges arise concerning issues such as: risk management, integration of ergonomic solutions, optimization of resource use, etc. [1].

Safety for conventional robot systems has traditionally involved isolating them from the human operator to prevent dangerous interactions. However, the isolation of the industrial robot is by definition impossible for cases that require cooperation and interaction with humans in the same workplace [2]. These cases are handled by deploying specially designed collaborative robots (cobots). The main differences between cobots and common robots are: (a) Ability to safely interact with humans based on special design: joint torque sensors, compliance, soft link material etc. (b) Flexibility of programming and ability to learn tasks easily [3].

In some cases, due to the abundance of classical robots installed in industry, but also the still low payload of collaborative robots, it could be possible to use classical industrial robots cooperating with humans by adding appropriate sensors and software [4].

This paper presents the use of ROS to investigate ways of cooperation between classical industrial robots and humans. Various scenarios are considered without risking human safety since they take place in a virtual world. At the same time, however, due to the use of real robot control middleware, and indeed with the integration of further sensors, the scenarios and responses recorded are realistic and ready for implementation in the physical world. A similar approach exploiting virtual worlds has been taken in the past [5, 6], but without the presence of middleware, which reduces its effectiveness.

Section 2 provides a brief overview of the basic concepts of a collaborative system. Section 3 describes the robot and software tools used to implement the scenarios. Sections 4 and 5 describe the scenarios and their function. In Section 5, conclusions are formulated and next steps of the research are presented.

2. HUMAN-ROBOT COLLABORATION DESIGN

The main approaches to safe collaboration fall into four directions [2] (Fig. 1):

(a) "Safety-based operation monitoring" – the robot stops when the operator enters the collaboration space and resumes when the operator is released.

(b) "Manual guidance" – the robot movements are controlled by humans with a special device.

(c) “Speed and separation tracking” – the operator and the robot may move simultaneously in the workspace. (d) "Force limitation" – contact forces between operator and moving robot are technically limited to a safe level.

Safety standards in the field of Human-Robot-Interaction (HRI) present requirements for collaborative industrial robot systems and the work environment complementing the guidance provided in ISO 10218-1 and ISO 10218-2. Overall, there are numerous different standards and regulations. ISO 15066 offers some rules regarding the design of production lines in order to achieve safety as well as human-robot performance in the industrial space [1].

[*] Corresponding author: Heroon Polytehniou 9, Athens 15773, Greece
Tel.: +30 210 7721457
E-mail addresses: *vosniak@central.ntua.gr* (G.-C. Vosniakos).

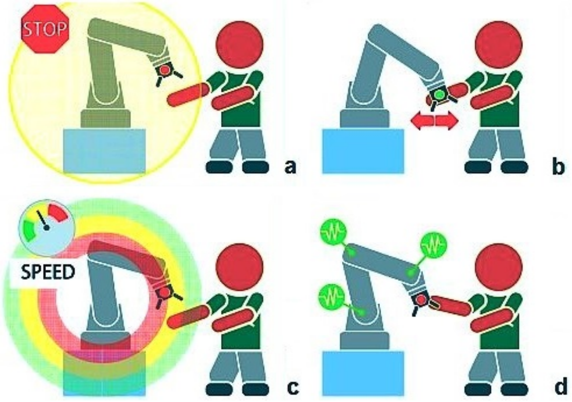


Fig. 1. Human-Robot collaboration approaches: *a* – safety rated monitored stop; *b* – hand guiding (c) Speed and separation monitoring; *d* – power and force limiting [7].

For example, minimizing the probability of the arm colliding with the human can be achieved by:

(a) defining trajectories in such a way that the human is not easily trapped between the robot and objects in the environment

(b) reducing the speed of moving parts,

(c) reducing the robot's workspace,

(d) setting limits on the robot's speed during collaboration,

(e) minimizing the torque of the arm by software,

(f) using touch sensors to control contact,

(g) appropriately designing the final action element so that to provide security.

Complementarily, optimizing collaboration ergonomics can be achieved by minimizing human fatigue due to repetitive or demanding tasks, long distances, heavy objects, etc. [2].

3. IMPLEMENTATION ENVIRONMENT

3.1. The robot and its digital model

The robot used in this work is the Stäubli^{TM} RX90L with a CS7 control unit. It is an industrial-type robot with a relatively small payload (3.5 kg), high repeatability (25 μm) and maximum speed: 12.6 m/s, used in assembly applications as well as in "pick and place" cases. The robot consists of six rotary joints, and its characteristics are given in [5]. The end effector is the fully electric Robotiq^{TM} 2F-85 gripper which has two fingers with two joints each and a maximum opening of 85mm. The robot is programmed in V+ language in dumb terminal type programs.

The robot links were modelled in Solidworks^{TM} compatible format. The appropriate mates were defined and the assembly was created based on the manufacturer's dimensions as well as the limits of the joint rotation angles. Appropriate angle mates were set and the configuration with the joints at zero angle was termed 'home'. Finally, the 'Solidworks to URDF Exporter' plugin was used to create a ROS compatible model of the robot.

3.2. ROS^{TM} environment

Robot Operating System (ROS^{TM}) is an open source robotics suite with a network architecture where processing is performed at nodes that may receive sensor data, control, status, and other messages with low latency [8]. It provides not only standard operating system services (such as hardware abstraction, contention management, process management), but also high-level functions (asynchronous and synchronous communication, central database, robot configuration system, etc.). ROS-Industrial (ROS-I^{TM}) includes interfaces for common industrial actuators, grippers and sensors. It also provides software libraries for automatic 2D / 3D sensor calibration, path planning, applications such as Scan-N-Plan, developer tools such as the Qt Creator^{TM} ROS^{TM} Plugin, and tutorials. [8]. The main structure of the ROS-I environment is given in Fig. 2. In this case, the ROS Melodic version was installed on Ubuntu^{TM} 18.04LTS operating system using Virtual Machine. The Atom program was used to write the code.

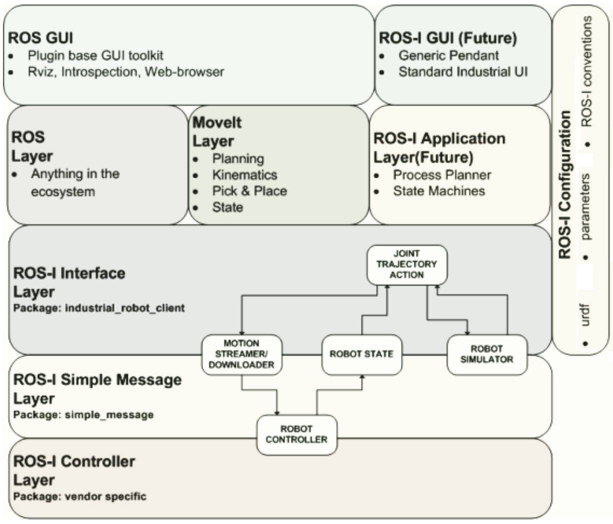


Fig. 2. ROS-I^{TM} architecture (modified from wiki.ros.org).

ROS^{TM} manages all information using Nodes. A node can correspond to a sensor, an engine, an algorithm, etc. Each node that starts its operation notifies the Master, which is a registration service so that the nodes can communicate with each other and exchange data. It also includes the Parameter Server. Data is exchanged with a Topic asynchronously, whilst with a Service synchronously. A Topic is a data transmission medium based on the subscribe/publish system. A Message is a collection of elements in a specific format and contains information about a Topic. For example, a node representing a robot servo motor 'publishes' its status to a topic with a message containing an integer number for the motor's position, a decimal number for its temperature, and a decimal number for its speed. Services are methods of synchronous communication between two nodes.

Another interesting contribution of ROS^{TM} is URDF (Unified Robot Description Format), an XML format used to describe a robot statically or dynamically while its physical properties can be added. Various tools are used for creation, analysis or control, for example representation of the robot is made with the Gazebo simulation software.

3.3. ROS^{TM} add-ins

MoveIt!^{TM} is the most widespread open source software for manipulating robotic devices incorporating the latest developments in path planning, 3D perception, kinematics, control and navigation. MoveIt!™ provides the basic functionality for manipulation in ROS™. The main node of MoveIt!™ is called move_group. It is built in a way that does not consume many resources and manages many threads at the same time, such as the robot's kinematics, the trajectory algorithm, etc. The user can access the core of move_group with C++ language, with Python language or in from the graphical interface (GUI).

The first step is the moveit_setup_assistant that allows importing new robots in a Semantic Robot Description Format (SRDF) file and creating a package to interact, visualize and simulate the robot in the workplace [9]. The Allowed Collision Matrix (ACM) refers to the collisions between the links and the endpoint determined based on a number of random samples. Virtual joints are used to indicate where the robot is in the world, e.g. can be related to its position on the ground. Planning groups, are subsystems of a robot such as a tool in the final action element, a subset of the kinematic chain, etc. Passive joints are not considered in trajectory calculation and control [9].

Rviz™ allows the user to create work environments and perform collision-free movements with objects in space. It has a number of functions such as Motion Planning, where trajectory planning is done with declaration of start-end points (collision free). In addition, there is the ability to read data from real sensors and map them to the environment ensuring realism and interactivity [9]

Gazebo™ is open source 3D simulation software. It integrates the ODE physics engine, OpenGL rendering as well as support for sensor simulation and motor control. Gazebo™, offers realistic environment rendering with high quality lighting, shadows and colors. It can also simulate environmental scanning sensors, such as laser sensors, cameras, trackers, and transfer the data for visualization in Rviz™ [10].

In this work, the gpu based Hokuyo™ laser scanning sensor from the Gazebo™ library was used. It allows human positions to be captured in the Rviz™ environment in order to be considered in the trajectory planning algorithm. The user can change the position and orientation of the sensor, the maximum recording distance as well as the range and step of the scanning angle.

To model the human, the "actor" animation was used, which allows adding animations. The skin describes the external appearance of the human in a COLLADA (.dae) file. The animation concerns the movement performed by the person (eg walking, running) based on his kinematic chain consisting of ten links and ten joints. [11] The planning of human movement is done by interpolating positions and orientations at corresponding moments in time.

4. SCENARIOS FOR STANDALONE ROBOT

Two scenarios were created initially for familiarization with the programming tools before implementing collaborative scenarios, but also for building the robot's representation and control infrastructure to be used in collaborative scenarios next.

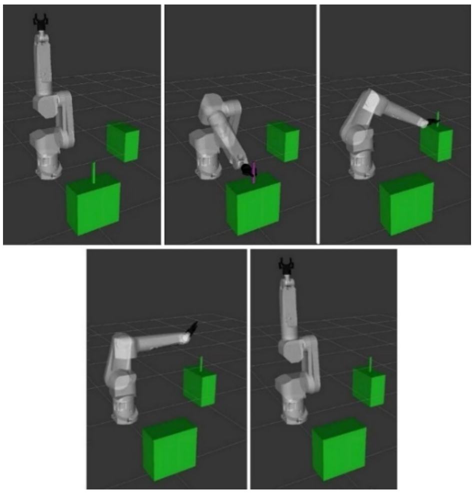


Fig. 3. Snapshots of the pick-&-place scenario implementation.

First, a common pick&place task was chosen. The movement of the robot to the object is done with the go_to_object function, the opening of the gripper and the capture of the object is done with the attach_box function, then the movement to the drop point is done with the go_to_2ndtable function while the creation and removal of the scene objects are done with the add_box and remove_box functions. The relevant Python code consists of 120 lines. The necessary PID controllers of the 6 joints (p: 100.0, i: 0.01, d: 10.0) and the gripper fingers (p: 20.0, i: 0.1, d: 0.0) were created to perform movements in 34 lines of Python code. Fig. 3 shows the arm movement programmed in the Rviz™ environment.

Afterwards, execution of a trajectory of the end point of action following a 3rd degree polynomial curve was programmed with kinematic control of three joints q1, q2, q3, considering the joints of the end effector stationary (q4=q5=q6=0). The analytical inverse kinematic solution for this case is trivial and is given in standard robotics notation as follows:

$$q_1 = tan^{-1}\frac{p_y}{p_x}, \tag{1}$$

$$q_2 = tan^{-1}\frac{p_y c3(l3+l4+lE)+p_y l2-(p_z-l1)(l3+l4+lE)s3s1}{s1(p_z-l1)(c3(l3+l4+lE)+l2)+p_y s3(l3+l4+lE)}, \tag{2}$$

$$q_3 = cos^{-1}\frac{{p_x}^2+{p_y}^2+(p_z-l1)^2-(l3+l4+lE)^2-l2^2}{2l2(l3+l4+lE)}. \tag{3}$$

Two Python files were created. The first (kinematics.py) 102 lines contains the calculations of the angles / kinematics of the robot. The second file, 170 lines long, contains all the necessary nodes-topics to start the simulation as well as the algorithm for interpolating points along the polynomial trajectory. Fig. 4 shows snapshots of the robot's movement along y-axis.

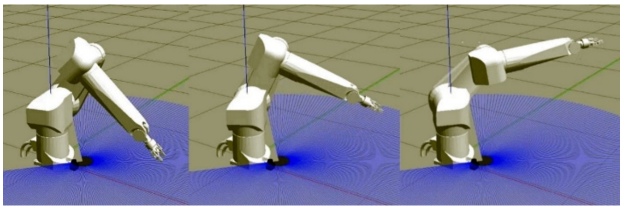


Fig. 4. Robot moving under full kinematic control along y-axis.

5. HUMAN-ROBOT INTERACTION SCENARIOS

Five scenarios were created involving interaction between human and robot. Four of them are relatively simple, whilst the fifth synthesizes the previous four into a more complex collaboration scenario.

5.1. Robot halt upon human entry in workspace

When the human enters the robot's workspace then it stops its movement. In particular, the laser sensor returns in a table the distance of the base of the robot from the link of the human closest to it (e.g. right arm or left leg). The sensor is defined in gazebo and transfers its data via a topic to Rviz^{TM}. When the human enters the workspace, the robot stops and the human's steps are visualized in the Rviz^{TM} software in real time, Fig. 5. The program takes up 135 lines of Python.

5.2. Robot speed reduction as human approaches.

Three different speed values were assigned to the robotic arm. As the human approaches the robot (relative distance between them from 1.7 m to 3 m), its speed decreases from 0.02 m/sec to 0.005 m/sec, until it stops when the distance from the human's feet to the base of the robot is small (under 1.7 m), see Fig. 6. The program takes up 147 lines of Python code.

5.3. Robot motion when human is at specific distance

In this scenario, a trajectory is executed only when the distance of the human's center of mass from the end effector of the robot is less than 1 m, Fig. 7. The program occupies 142 lines of Python.

5.4. Keeping distance between human and robot

In this scenario, as the human enters the workspace, the robot performs a motion in the reverse to the human's direction if the distance of the robot's end effector from the human's center of mass is between 1.2 and 3.5 m, Fig. 8. Conversely, the robot can advance towards the human if the latter moves away from it, to keep their distance. The program takes up 98 lines of Python.

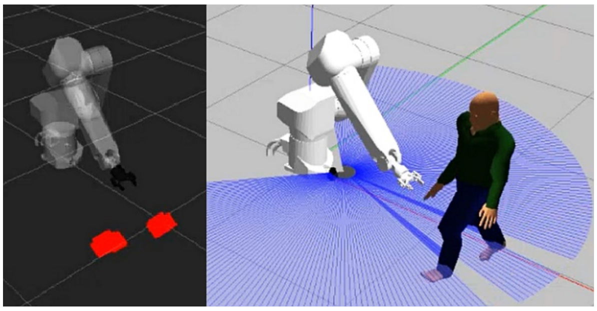


Fig. 5. Robot halts upon human entering its workspace.

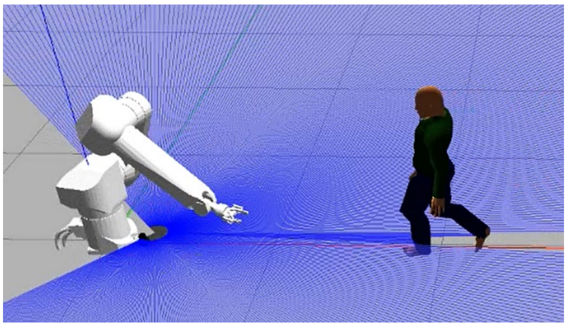
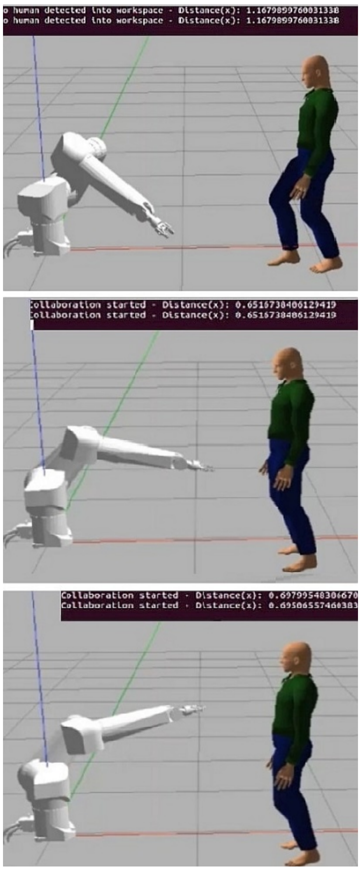


Fig. 6. Robot speed is reduced as human enters the workspace.



```
o human detected into workspace - Distance(x): 1.167989976003133B
o human detected into workspace - Distance(x): 1.167989976003133B
```

```
ollaboration started - Distance(x): 0.6516738406129419
ollaboration started - Distance(x): 0.6516738406129419
```

```
Collaboration started - Distance(x): 0.697995483066670
Collaboration started - Distance(x): 0.695065574660383
```

Fig. 7. Robot halts upon human entering its workspace.

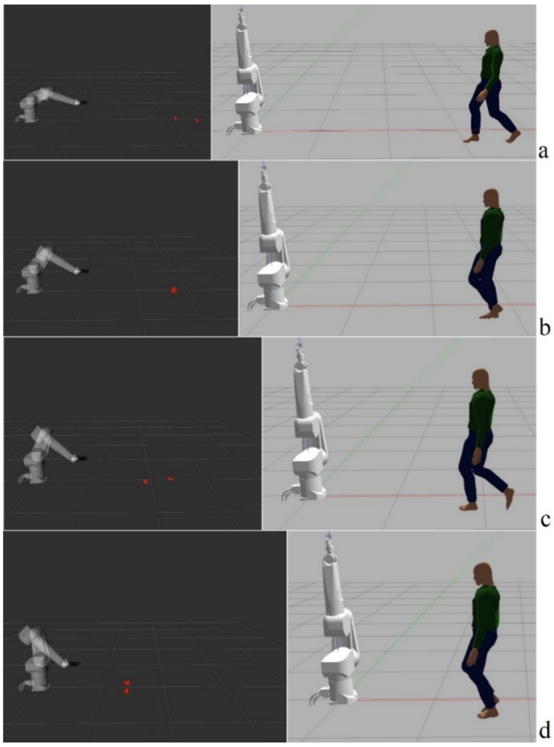


Fig. 8. Keeping human-robot (H-R) distance: *a* – start of human motion; *b* – as human approaches, robot starts retreating; *c* – intermediate human motion as robot continues retreating; *d* – minimum H-R distance and maximum robot retreat.

5.5. Complex collaborative task

The simpler individual scripts developed earlier were synthesized in this scenario where human-robot collaboration was implemented to screw a nut with the help of a wrench placed on a table, see Fig. 9.

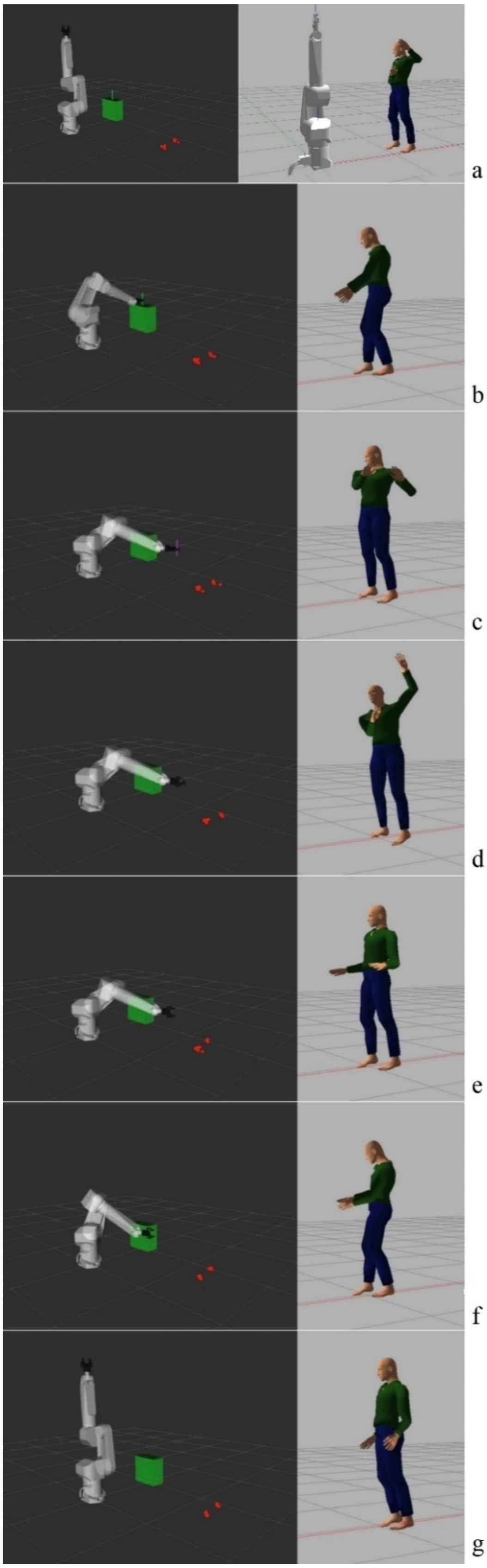


Fig. 9. Consecutive snapshots of collaborative screw mounting.

The human signals to start the cooperation, Fig. 9,*a*. The human approaches and reaches the desired position to hold the screw Fig. 9,*b*. The robot initially picks up a key that is on the table with the help of the gripper Fig. 9,*c*. He then takes it to the position where the nut is and screws it in by turning the sixth joint, Fig. 9,*d*. Then the gripper opens, Fig. 9,e, the robot moves backwards Fig. 9,*f* and returns to the home position, Fig. 9,*g*. The program takes up 203 lines of Python.

6. DISCUSSION AND CONCLUSIONS

This work dealt with human-robot collaboration in the ROS^{TM} environment. This includes various cooperative movements making up corresponding scenarios. Collision avoidance techniques included the integration of a laser sensor for human recognition in the workspace as well as the extraction of position and orientation data directly from the Gazebo^{TM} software.

The ROS^{TM} robotics platform offers great potential and flexibility in the development of new human-robot collaboration scenarios, with a wide range of devices that can be used (e.g. laser sensors, cameras). The programming languages used allow the user to develop suitable algorithms in order for the robot to understand human gestures and movements.

In a real environment, signals received from laser sensors and cameras are transmitted over a network to the ROS^{TM} environment. Many well-known robot manufacturers have implemented ready-made packages for the interface. For example, in the physical implementation of the robot halting scenario when the human enters its workspace, the laser sensor transmits data from the environment in real time and the Rviz^{TM} software captures it as in the case of the simulation. The avatar used in the gazebo software is replaced by the human entering the workspace. In practice, the behaviour of the real robot will be the same as that of the simulation, without any intervention in the algorithm.

In an industrial environment where people can be in different places and make various movements, holding objects etc., their identification with the laser distance sensor is not sufficient, thus it can be replaced by a camera and neural networks that can be used to recognize human movements and gestures. This is the future research direction of this work.

REFERENCES

[1] L. Gualtieri, E. Rauch, R. Vidoni, and D. T. Matt, *Safety, Ergonomics and Efficiency in Human-Robot Collaborative Assembly: Design Guidelines and Requirements*, Proc. CIRP 91 (2020), pp. 367–372.

[2] P. Chemweno, L. Pintelon, and W. Decre, *Orienting safety assurance with outcomes of hazard analysis and risk assessment: A review of the ISO 15066 standard for collaborative robot systems*, Saf. Sci., 129 May (2020), p. 104832.

[3] R. R. Galin and R. V. Meshcheryakov, *Human-Robot Interaction Efficiency and Human-Robot Collaboration*, Stud. Syst. Decis. Control, 272 January (2020) pp. 55–63.

[4] C. Maragkos, G.-C. Vosniakos and E. Matsas,. *Virtual reality assisted robot programming for human collaboration*. Proc. Manuf. 38 (2019) pp.1697-1704.

[5] E. Matsas, G. C. Vosniakos, and D. Batras, *Prototyping proactive and adaptive techniques for human-robot*

collaboration in manufacturing using virtual reality, Robot. Comp. Integr. Manuf. 50 (2018), pp. 168-180.

[6] G.C. Vosniakos, J. Deville and E. Matsas, *On immersive virtual environments for assessing human-driven assembly of large mechanical parts*, Proc. Manuf. 11 (2017), pp.1263-1270.

[7] V. Villani, F. Pini, F. Leali, and C. Secchi, *Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications*, Mechatr. 55, (2018), pp. 248–266.

[8] A. Ademovic, *An Introduction to Robot Operating System: The Ultimate Robot Application Framework,* available at: https://www.toptal.com/robotics/introduction-to-robot-operating-system, accessed: 2023-08-31.

[9] MoveIt!, *Official webpage*. available at: https://moveit.ros.org/, accessed: 2023-08-31.

[10] W. Qian, Z. Xia, J. Xiong, Y. Gan, Y. Guo, S. Weng, H. Deng, Y. Hu and J. Zhang, *Manipulation task simulation using ROS and Gazebo*, 2014 IEEE Int. Conf. Robotics and Biomimetics (ROBIO 2014), pp. 2594-2598. Bali, Indonesia, December 2014, IEEE-Xplore.

[11] L. He, P. Glogowski, K. Lemmerz, B. Kuhlenkötter, and W. Zhang, *Method to Integrate Human Simulation into Gazebo for Human-robot Collaboration*, IOP Conf. Ser. Mater. Sci. Eng. 825 1 (2020).