ACTA IMEKO
ISSN: 2221-870X
September 2021, Volume 10, Number 3, 72 - 80

Human–robot collision predictor for flexible assembly

Imre Paniti[1,2], János Nacsa[1,2], Péter Kovács[1], Dávid Szűr[1]

[1] *ELKH SZTAKI, Centre of Excellence in Production Informatics and Control, Kende Street 13–17, 1111 Budapest, Hungary*
[2] *Széchenyi István Egyetem, Egyetem Square 1, 9026 Győr, Hungary*

ABSTRACT
The performance of human–robot collaboration can be improved in some assembly tasks when a robot emulates the effective coordination behaviours observed in human teams. However, this close collaboration could cause collisions, resulting in delays in the initial scheduling. Besides the commonly used acoustic or visual signals, vibrations from a mobile device can be used to communicate the intention of a collaborative robot (cobot). In this paper, the communication time of a virtual reality and depth camera-based system is presented in which vibration signals are used to alert the user of a probable collision with a UR5 cobot. Preliminary tests are carried out on human reaction time and network communication time measurements to achieve an initial picture of the collision predictor system's performance. Experimental tests are also presented in an assembly task with a three-finger gripper that functions as a flexible assembly device.

Section: RESEARCH PAPER

Keywords: Collaborative robot; human–robot collaboration; virtual reality; collision prediction

Citation: Imre Paniti, János Nacsa, Péter Kovács, Dávid Szűr, Human-robot collision predictor for flexible assembly, Acta IMEKO, vol. 10, no. 3, article 12, September 2021, identifier: IMEKO-ACTA-10 (2021)-03-12

Section Editor: Bálint Kiss, Budapest University of Technology and Economics, Hungary

Received February 15, 2021; **In final form** August 16, 2021; **Published** September 2021

Funding: This work was supported by the TKP2020-NKA-14 grant and by the H2020 project EPIC grant no. 739592.

Corresponding author: Imre Paniti, e-mail: imre.paniti@sztaki.hu

1. INTRODUCTION

According to the International Federation of Robotics 2019 report, the average robot density in the manufacturing industry has grown to a new global record of 113 units per 10,000 employees [1]. Although the automation of small- and medium-sized enterprises (SMEs) is supported within the European Union according to the European Commission's Digital Economy and Society Index Report 2019 [2], the share of large enterprises that use industrial or service robots is four times higher than that of SMEs, and the use of robots varies widely with company size.

One of the most commonly asked questions in the semi-robotised industry is how to make production more efficient, which is related to a study [3], where robots in an assembly operation could reduce the idle time of an operator by 85 %. Therefore, using collaborative robots (cobots) in a factory for assembly tasks could lead to greater efficiency, which means shorter production times. This statement can also be useful for the assembly of different products or product families, which requires a set of different fixtures or reconfigurable fixtures, such as those based on the parallel kinematic machine in [4] or the fixed but flexibly useable gripper presented in this article.

However, the problem is that despite well-defined task sequences, the changeover from one product to another in a collaborative operation could lead to human failures and, consequently, to collisions with the cobot due to the previous habitual sequence of actions.

By definition, a cobot has to operate with strict safety installations (protective stop execution when a certain force in a collision is reached), as outlined in ISO/TS 15066:2016 [5], ISO 10218-1:2011 [6] and ISO 10218-2:2011 [7], but these protective stops could cause a significant cumulative delay in production. This depends largely on how the robot program has been written, i.e. whether operations can be continued after a protective stop.

Review articles such as those of Hentout et al. [8] and Zacharaki et al. [9] present solutions for pre-collision approaches in the frame of human–robot interaction (HRI). Pre-collision control methods, referred to as 'prevention' methods, are techniques intended to ensure safety during HRI by monitoring either the human, the robot or both and modifying robot control parameters prior to incidence of collision or contact [9]. Pre-collision approaches can be distinguished between reactive control strategies, proprioceptive sensor-based strategies and exteroceptive sensor-based control [8]. However, these approaches are all manifested in robot control parameter modification rather than in operator warnings.

Both the above studies refer to the work of Carlos Morato et al. [10], who presented a similar solution by creating a framework using multiple Kinects to generate a 3D model with bounding spheres for human movements in real time. The proposed framework calculates human–robot interference in a 3D space with a physics-based simulation engine. The deficiency of the study is the pre-collision strategy for safe human–robot collaboration because this results in the complete stoppage of the robot. This is indeed a safe protocol, as it reduces the production break time, but it does not eliminate it completely.

The aim of this paper is to highlight the importance of a new pre-collision strategy that does not modify the trajectories but relies fully on the warning of the operator (using a non-safety-critical system), especially when flexible/reconfigurable fixtures are used.

Section 2 provides an overview of standards and definitions related to robotic and cobotic systems, especially in relation to protective separation distance, which is crucial for the proposed solution. Section 3 presents an experimental environment and use cases in which the proposed solution can be used. Section 4 describes the new pre-collision approach and its system elements in detail, together with some communication measurement results to demonstrate the feasibility of the solution. Finally, Section 5 presents a summary with conclusions.

2. STANDARDS AND DEFINITIONS FOR COBOT USE

In general, when using a robotic arm with a gripper the 2006/42/EC Machinery Directive [11] and the 2014/35/EU Low Voltage Directive [12], together with ISO/TS 15066:2016 [5] and 16 standards, have to be considered [13]. These are detailed in Table 1.

According to ISO 10218-1:2011 [6], a collaborative workspace is a space within the operating space where the robot system (including the workpiece) and a human can perform tasks concurrently during production operations, and a collaborative operation is a state in which a purposely designed robot system and an operator work within a collaborative workspace.

Table 1. Standards in manufacturing when using a robotic arm with a gripper.

Standard	Ref.
EN ISO 10218-1:2011	[6]
EN ISO 10218-2:2011	[7]
ISO/TR 20218-1:2018	[14]
EN ISO 13855:2010	[15]
EN ISO 13849-1:2015	[16]
EN ISO 13849-2:2012	[17]
EN ISO 12100:2010	[18]
EN ISO 13850:2015	[19]
EN IEC 60204-1:2018	[20]
EN IEC 62061:2005	[21]
EN ISO 11161:2007	[22]
EN ISO 13854:2017	[23]
EN ISO 13857:2019	[24]
EN ISO 14118:2017	[25]
EN IEC 62046:2018	[26]
EN ISO 13851:2019	[27]

According to ISO/TS 15066:2016 [5], collaborative operations may include one or more of the following methods:

- a safety-rated monitored stop,
- hand guiding,
- speed and separation monitoring,
- power and force limiting.

In power- and force-limiting operations, physical contact between the robot system (including the workpiece) and an operator can occur either intentionally or unintentionally. Power- and force-limited collaborative operations require robot systems specifically designed for this particular type of operation using built in measurement units. According to ISO/TS 15066 [5], risk reduction is achieved, either through inherently safe processes in the robot or through a safety-related control system, by keeping hazards associated with the robot system below threshold limit values, which are determined during the risk assessment.

If an operator wants to maintain a safe distance in a collaborative operation, ISO/TS 15066:2016 Robots and robotic devices - Collaborative robots (clause 5.5.4: Speed and separation monitoring) [5], EN ISO 13850:2015 [19], EN ISO 13855:2010 [15], EN IEC 60204-1:2018 [20] and EN IEC 62046:2018 [26] should be applied together with the following regulations and standards: Directive 2006/42/EC [11], EN ISO 10218-1:2011 [6] and EN ISO 10218-2:2011 [7]. In addition, EN ISO 12100:2010: Safety of machinery - General principles for design - Risk assessment and risk reduction [18] should be considered.

In speed and separation monitoring, the protective separation distance is the shortest permissible distance between any moving hazardous part of the robot system and any human in the collaborative workspace, and this value can be fixed or variable.

During automatic operations, the hazardous parts of the robot system should never get closer to the operator than the protective separation distance, which is calculated based on the concepts used to create the minimum distance formula in ISO 13855:2010 [15].

The protective separation distance S_p can be described by formula (1):

$$S_{\mathrm{p}}(t_0) = S_{\mathrm{h}} + S_{\mathrm{r}} + S_{\mathrm{s}} + C + Z_{\mathrm{d}} + Z_{\mathrm{r}}, \tag{1}$$

where

$S_{\mathrm{p}}(t_0)$ is the protective separation distance at time t_0 (present or current time);

S_{h} is the contribution to the protective separation distance attributable to the operator's change in location;

S_{r} is the contribution to the protective separation distance attributable to the robot system's reaction time;

S_{s} is the contribution to the protective separation distance due to the robot system's stopping distance;

C is the intrusion distance, as defined in ISO 13855, which is the distance that a part of the body can intrude into the sensing field before it is detected;

Z_{d} is the position uncertainty of the operator in the collaborative workspace as measured by the presence sensing device resulting from the sensing system measurement tolerance; and

Z_{r} is the position uncertainty of the robot system, resulting from the accuracy of the robot position measurement system [5].

Based on this, the authors propose to extend the protective separation distance (1) with an extra distance based on the communication time of a pre-collision system (S_{pc}) and with a contribution to the protective separation distance attributable to the robot operator's reaction time (S_{ort}) to avoid speed reductions or protective stops. This would result in a modified protective separation distance (S_{p}*):

$$S_{\mathrm{p}}^{*} = S_{\mathrm{p}} + S_{\mathrm{pc}} + S_{\mathbf{ort}} \,. \tag{2}$$

However, the proposed system in this paper is, as has already been mentioned, an extra non-safety certified solution. The purpose of the presented measurements in this paper is to determine the above-mentioned time parameters (communication time and reaction time) of the additional distances (S_{pc} and S_{ort}) in this specific environment.

3. EXPERIMENTAL ENVIRONMENT AND USE CASES

Robots are usually moved on prespecified trajectories that are defined in the robot's program, and, in most cases, a new task involves starting a new robot program. Another method is to move the high-level robot control from the robot to a computer, and the robot then continuously receives the required movements and other actions via a stream. In this case, the robot runs a general-purpose program or framework that interprets and executes the external instructions received. In this scenario, the framework is called URSZTAKI, developed by the SZTAKI Research Laboratory for Engineering and Management Intelligence. URSZTAKI has three kinds of instructions: (a) basic instructions that constitute the robot's programming language, (b) instructions for the robot add-ons (e.g. the gripper and force sensor) integrated into the robot language by the accessory suppliers and (c) frequently used, more complex task instructions (e.g. putting down or picking up an object when the table distance is unknown). The third type of instruction constitute the real features of URSZTAKI.

It should also be mentioned that the expansion of the UR robot's functions and language is possible with the help of so-called URCAPs (which is a platform where users, distributors and integrators can demonstrate accessories that run successfully in UR robot applications [28]), and currently, URSZTAKI can also be installed as an URCAP.

The experimental layout consists of a UR10 robot with a force sensor and a two-finger gripper. The environment was designed

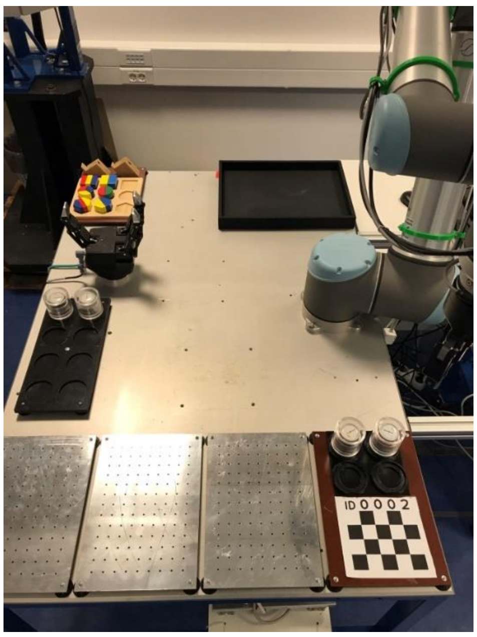
Figure 1. Demonstration environment.

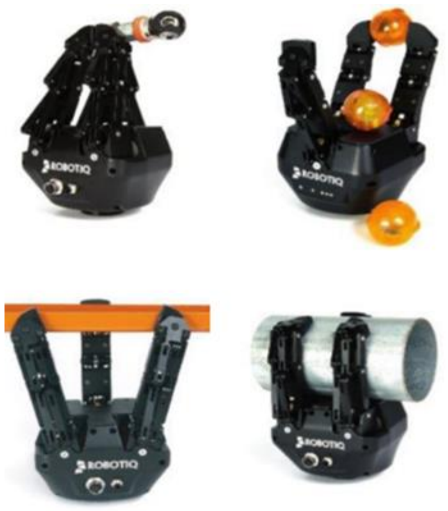
Figure 2. The four different modes of the three-finger RobotiQ gripper [29].

to support different assembly tasks, either fully robotised or collaborative. To equip partly or even fully different components, universal mounting technology was required instead of special fixtures. Another gripper (with three fingers) was used that allowed a wide variety of fixings. All three fingers could be moved independently of the selected adaptive gripper fixed to the robot worktable (Figure 3).

The three-finger gripper from RobotiQ [29] has four different modes for operating the fingers (Figure 2). In the 'pinch' mode on the top left side of Figure 1, the gripper acts as a two-finger model, and the fingers move closely together to be able to pick up small objects. The next mode is the 'scissor' mode in which the closing–opening ability of the gripper is used to pick up an object. In the third 'wide' mode, the fingers are fan-like, and they provide a firm wide grip for longer objects. For the leftmost 'normal' grip, the three fingers move in parallel and, depending on the relative position of the object, the fingertips also turn for greater precision in 'normal' and 'wide' mode. This is the encompassing grip.

From the software point of view, both grippers can be directly programmed from the robot's program code. Despite the fact that both grippers are from the same manufacturer, which could make the development easier, the commands of one of the grippers had to be modified to avoid conflict between the individual instructions.

A typical scenario is that the robotic arm picks up and transfers a part to the fixed gripper, which grabs it, and after that, another part is placed or pressed with the desired force by the robotic arm on the part fixed by the immobile gripper. There are some detailed tasks, such as the insertion of a spring into a housing, which have to be performed by the human operator.

In this environment, it is also possible for the robot to hold a screwdriver and fasten the assembled parts with screws at the set torque limit (Figure 3 and Figure 4).

The prototype was designed specifically for the previously shown push-button element. However, it can be easily redesigned for another part, or a universal piece can be made to support different types of product assembly.

Following the parallel movement of the fingers, a form-fitting shape is created that holds the part motionless while the required actions are carried out. Because the holder is connected to the fingertips, slippage is also prevented in cases where the pressing

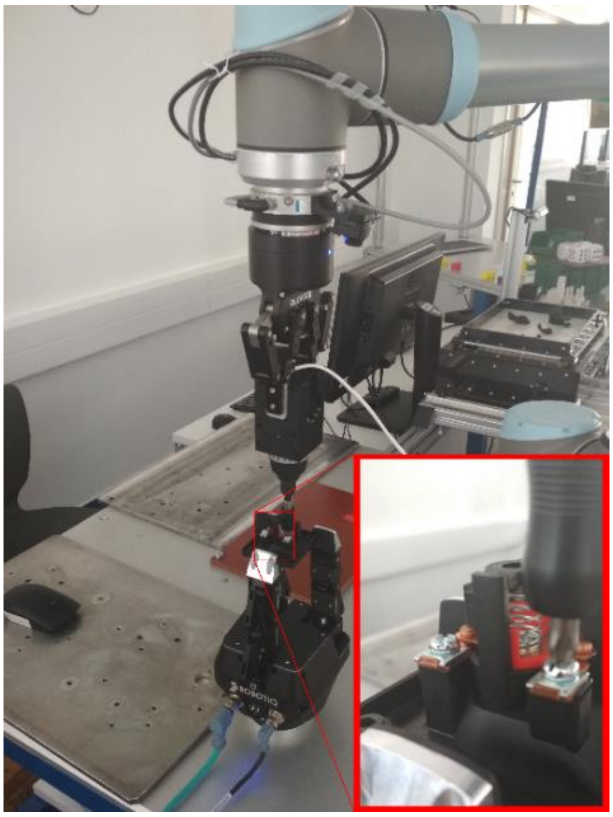
Figure 3. Illustration of the robotised screwdriving of a push-button element in which the spring has to be inserted manually.

force applied is too great or an inappropriate human movement occurs.

The proposed solution with the immobile three-finger gripper satisfies the requirements of a flexible fixture for certain parts. In this scenario, human–robot collision problems might occur if the human operator forgets the predefined assembly task sequence when beginning the assembly of a new product, reaches for an assembly part and the hand trajectory intersects that of the robot.

To demonstrate a flexible assembly with the three-finger gripper, an additional application was developed in which both grippers were used to perform the assembly task, requiring human intervention at certain points of the assembly process at

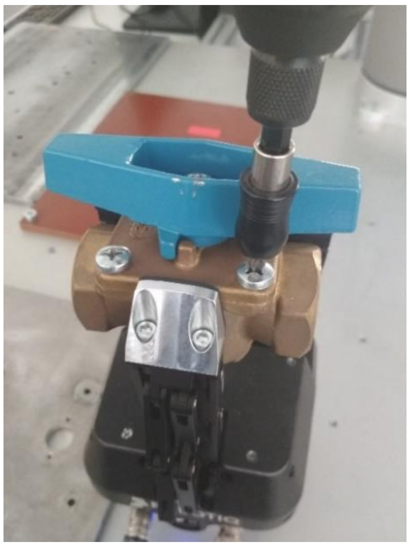
Figure 4. Illustration of the robotised screwdriving of a ball valve element.

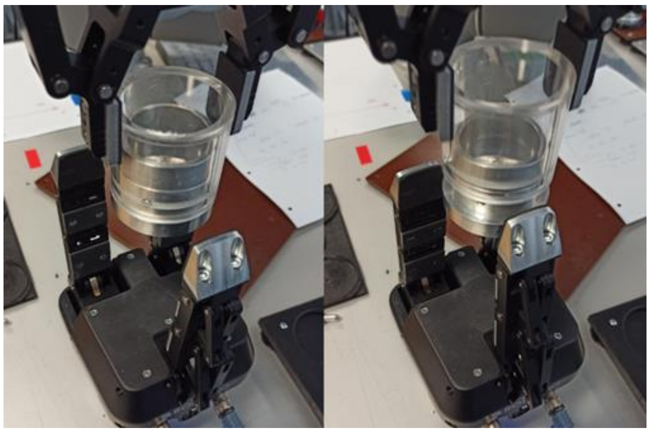
Figure 5. Illustration of the second step.

the same time. In the task, a didactic element, which had been packaged with a transparent plastic lid and a metal base, were pushed together at the beginning of the operation (presumably, this packaging material came from the supplier). The operation steps of the complete assembly were the following:

1. Pick up the packaging material with the robot arm and fix the base with the three-finger gripper.
2. Remove the lid from the base and put it down (Figure 5).
3. Pick up the didactic element and place it onto the metal base.
4. Put the plastic lid back on the base.
5. Fix the packed object, release the three-finger gripper and put it back in its starting position.

Inserting the didactic element is the bottleneck in the assembly process. Normally, the robot finds the hole with a small spiral movement using force sensing. Since the gap between the meter and the base is narrow, this operation is not always successful (see Figure 6), in which case, human intervention is possible or necessary to avoid any wastage.

In some instances, the next operation (putting the lid back) corrected the skewed didactic element, and it slipped into the base. However, the success of a process should not be based on coincidence, and this is when a collision predictor system can be very useful. An easy movement by the operator can prevent wastage, thereby reducing costs.

It is a simple operation sequence, but because of the positioning errors, human intervention may be required during two of the steps.

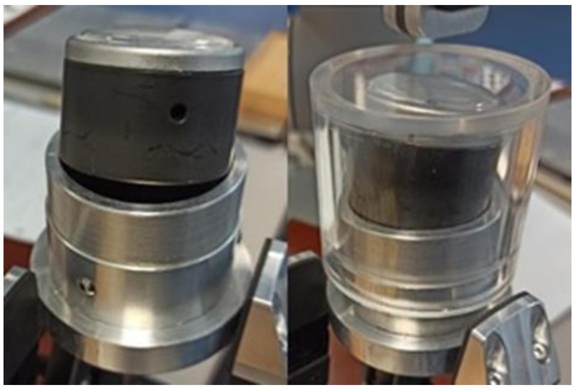
Figure 6. Illustration of the failed insertion of the didactic element.

4. PRE-COLLISION APPROACH AS A PREDICTOR

In order to avoid collisions with the robot, either the robot trajectory has to be modified in real time (which might cause additional production time, something companies want to avoid) or the human operator has to be warned with a pre-defined understandable signal so the human movement can be modified in time.

The warning signal can be given to the operator in several ways: visual, acoustic or tactile. In this paper, the latter has been developed as part of a PREdictor of Human–RObot COllision (PREHUROCO) framework. The subject of the prediction in this case is the predetermined movement of the robot, which can be recorded and will occur after a certain time, so a similar framework had to be created to that described in [10]. However, instead of a digital twin of the robot (real-time 3D visualisation of the robot), a pre-played robot model motion was used together with the 3D skeleton model of the operator. The virtual collisions of the two models were used as trigger signals to warn the operator before a real collision.

4.1. Requirement analysis

The following features were needed for the candidate software library, based on the requirement analysis of PREHUROCO:

1) Fully open source: the system must fulfil all the security requirements of a real manufacturing system; therefore, complete control of the source code is obligatory.
2) Modular: the system should be divided into various software components, so the candidate software library must support responsibility encapsulation.
3) Distributed: in a manufacturing system, many computers and Internet-of-Things (IoT) devices can be connected; therefore, the PREHUROCO software components must have the ability to run on different computers or IoT devices.
4) Cross platform: as the distribution requirement is for many computers and devices with different operating systems to be connected, the candidate framework should be cross platform.
5) Programming language variability: as the distribution and cross-platform requirements are for different devices and computer operating systems in manufacturing scenarios, the candidate software library should support different application programming interfaces (APIs).
6) Scalability: PREHUROCO software components should be developed independently of whether they run on the same computer or not. In terms of performance, the software components should be be easily put together in one machine or one application and easily distributed.
7) Rapid prototyping: the candidate framework should provide examples or even pre-made components that can be improved during PREHUROCO implementation because the proposed system should deal with
    - rigid-body simulation,
    - visualisation (including VR or AR),
    - real-time 3D scanning,
    - an X3D model format and
    - various communication protocols.

Table 2. Comparison of different frameworks in relation to the PREHUROCO requirements.

Requirement	Unity Engine	Unreal Engine	ApertusVR
Open source	Partially	Yes	Yes
Modular	Yes	Yes	Yes
Distributed	Partially	Corner Case	Yes
Cross platform	Yes	Partially	Partially
Prog. lang. variability	Partially	Corner Case	Yes
Scalability	Partially	Partially	Yes
Rapid prototyping	Yes	Yes	Yes

Unity Engine [30] and Unreal Engine [31] are well-known cross-platform game engines. ApertusVR [32] is a software and hardware vendor-free, open-source software library. It offers a no-vendor-lock-in approach for integrating VR technologies into industrial software systems.

The comparison of the candidate frameworks considering the requirements is summarised in Table 2.

Based on the PREHUROCO requirement analysis, the ApertusVR software library was chosen for implementing the system. With the help of this software library, a distributed software ecosystem was created via the Intranet/Internet, which was divided into two main parts, the core and plugins. The core system is responsible for the Internet/Intranet communication between the elements of the distributed software ecosystem, and it synchronises the information between them during the session. The plugin mechanism makes it possible to extend the capability of any solution created by the ApertusVR library. Plugins can access and manipulate the information within the core system.

4.2. Explanation of the PREHUROCO system

The system is distributed into five major responsibilities:

1) 3D scanning of the human operator,
2) streaming the joint angles of the robot,
3) collision detection between the human and the robot,
4) alerting the human to the possible collision and
5) visualising the whole scenario.

In the present study, these responsibilities were implemented with the help of the ApertusVR library, and each responsibility was encapsulated into six plugins [33]: the collision detection plugin, the visualisation plugin, the Kinect plugin, the WebSocket server plugin, the X3D loader plugin and the NodeJS plugin.

The seventh element was a WebSocket client, which was implemented in the form of an HTML site using the jQuery JavaScript library and the vibration API method [34] for mobile phones; but for more comfortable use, the WebSocket client could also run on a smart watch.

Figure 7 shows the realised system with the connections and applied protocols in an experimental set up with an UR5 robot.

Collision detection plugin [35]: this plugin was created based on the pre-made ApertusVR ‘bulletPhysics’ plugin. Previously, this plugin had been able to run rigid-body simulations, but collision events were not created during these simulations. The ApertusVR rigid-body abstraction was enhanced by the functionality of collision events.

Visualisation plugin [36]: this plugin was used as-is from the ApertusVR repository for visualisation purposes.

Kinect plugin [37]: this plugin was created based on the pre-made ApertusVR ‘Kinect’ plugin. Previously, this plugin had

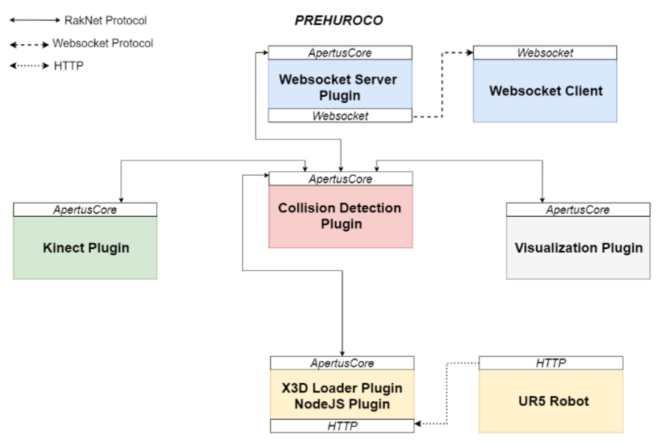

Figure 7. PREHUROCO system elements and connections with the applied protocols.

been able to create the skeleton of the tracked human or even its point cloud, but rigid bodies were not created. For collision detection, rigid bodies are mandatory; therefore, rigid bodies were created based on the geometries of the human skeletons.

Websocket server plugin [38]: this plugin was created based on the pre-made ApertusVR 'WebSocketServer' plugin. Previously, this plugin had been able to forward all events developed in the core. For collision detection, only the collision event of the rigid bodies is necessary. During the implementation of that plugin, a filter feature was added to forward only the desired event into the WebSocket connection.

X3D loader plugin [39]: this plugin was created based on the pre-made ApertusVR 'X3DLoader' plugin. Previously, this plugin had been able to parse the X3D format and create only the geometries of the robot. For collision detection, rigid bodies are mandatory; therefore, rigid bodies were created based on the parsed geometries.

NodeJS plugin [40]: this plugin was used as-is from the ApertusVR repository and allows a web server to be run to receive the joint angle of the UR5 robot via HTTP requests.

In the PREHUROCO system, these plugins are encapsulated in different applications. These different applications can be run on different computers to distribute the computational power and achieve real-time collision prediction. As the diagram in Figure 7 shows, these applications communicate through Internet/Intranet communication via different protocols.

The collision detection application has to be run on a high-performance computing (HPC) server to process the virtual collisions in real time.

The Kinect application can run on a dedicated computer for the Kinect device or on the same computer that calculates the virtual collisions.

The X3DLoader and the NodeJS plugins are integrated into one application and can run on the dedicated computer for the UR5 robot.

The WebSocket server application can also be run on a different computer to ensure security and locality requirements.

The joint positions are stored in a jsonlist file, which is generated by executing the whole robot program. During the execution, the joint positions are 'grabbed' and saved with a given frequency.

The speed of the simulation is equal to the speed of the robot movement, and the 'forecast' can be determined by the delay between the simulation starting time and the real robot execution start time.

4.3. Modified PREHUROCO system and measurements

During the validation process, the PREHUROCO system was reconfigured to eliminate any unnecessary delay in the system. The reconfiguration process was achieved by the ApertusVR configuration feature; thus, all the plugins were re-used without any modification. The previously distributed PREHUROCO system was therefore easily reconfigured to form a single application (Figure 8) and was able to run on a single computer.

The elimination of unnecessary network connections/delays was a crucial step in avoiding any latency in the system. Through this approach, the human–robot-collision calculation time and the human-operator reaction time were measured precisely. Timestamps were buffered before and after the collision events, the WebSocket message transmission/receipt and the human operator pressing the button on a Bluetooth keyboard.

The proposed framework was tested on two local network topologies. In the first case, the calculations were divided into a cloud-service-based computer (with four virtual CPUs, 8 GB RAM, running a Windows 10 operating system) and an HPC server (Ideum with Intel i7-8700, RTX 2080 8 GB GDDR6 NVIDIA graphics card, dual 250 GB NVMe M.2 SSD, 32 GB 2400 MHz DDR4 RAM, running a Windows 10 operating system), and the collision events were delivered to the WebSocket client with significant delay.

By running all ApertusVR plugins on the Ideum and sending only the collision events via a wireless LAN connection (2.4 GHz Wi-Fi) the user experience was quasi real-time.

Figure 9 shows a virtual collision test running on the Ideum (HPC server) with the skeleton model of a single operator (1), virtual UR5 robot movement simulation (2), a real robot (3), a Kinect sensor (4) and a mobile phone (5) with an android operating system running the WebSocket client to vibrate the device. The 3D scene was visualised with a top camera view, but arbitrary camera views are possible.

To avoid the execution of large JavaScript files locally on the android mobile phone, external calls to cdn.jsdelivr.net and code.jquery.com were used. The ping time to these services were measured with an android application (PingTools version 4.52), which gave 9 ms and 30 ms as the average from three measurements, respectively.

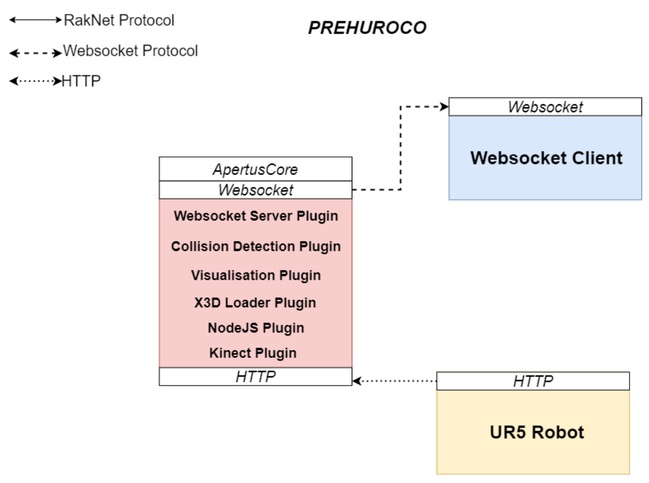

Figure 8. Reconfigured PREHUROCO system.

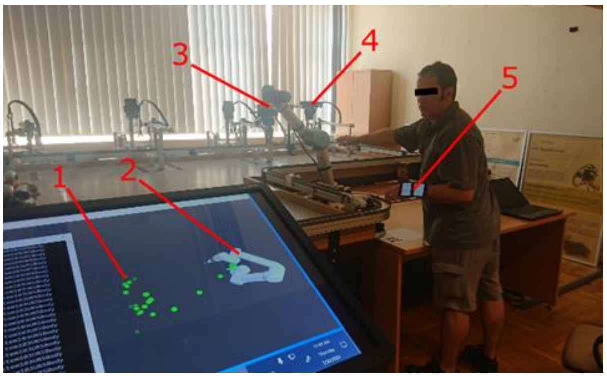

Figure 9. Virtual collision test.

The second network topology was used to measure the communication time of the system with five more people of different genders and ages (see Figure 10). The reaction time of each operator was measured using an android application (Reaction Test version 1.3), which vibrates at randomised short time intervals (a couple of seconds) and calculates the average of five measurements.

The average calculation time of the human–robot model collision until the HTTP-request send was 98 ms, the average time from the HTTP-request send to the keypress event was 1,355 ms and the average reaction time was 449 ms. Each virtual collision with keyboard pressing as confirmation was tested three times. According to a Bluetooth keyboard performance test, 'Microsoft delays in a non-interference test environment by approximately 40 to 200 milliseconds' [41], so the calculation time for the human–robot collision together with the network communication time would be less than 1 s using this PREHUROCO configuration.
However, by using RakNet instead of HTTP requests the performance of the system can be significantly improved. RakNet communication time measurements from 223 collision events showed that only 36.52 ms was needed on average. Furthermore, it is worth mentioning that with 5G communication an average two-way latency of 1.26 ± 0.01 ms would be possible, as noted in [42].

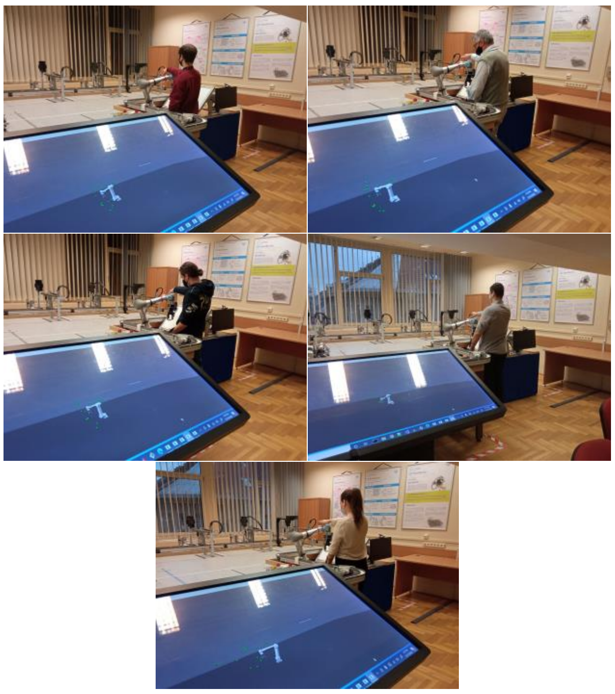
Figure 10. Virtual collision measurements with five additional candidates.

The Kinect plugin creates a simplified skeleton model from the human operator, which needs improvement. An anthropomorphic skeleton model or voxelisation could be a solution in the future.

It should be highlighted that the communication time increased by the human reaction time should not exceed the ΔT time between the pre-played simulated motion and the actual motion of the robot. A jsonlist file of the simulated UR5 robot movement is provided in [43].

5. CONCLUSION

In this paper, a commercially available gripper as a flexible fixture for assembly and a new pre-collision approach as a predictor for human–robot collaboration were presented. The proposed framework was realised with the help of a modular, distributed, open-source cross platform (ApertusVR) with different programming API support and scalability solutions.

Seven interconnected system modules were developed with the goal of monitoring the movement of the human operator in 3D space, calculating collisions with a virtual robot (with pre-played movements rather than the movement of a real robot) and alerting the human operator before a real collision could occur. Successful virtual collision tests with six candidates showed that the operator received the warning signal immediately (under 1 s) in the form of a mobile-device vibration to modify the planned movement.

In some cases, real-time path planning is required, especially in a changing environment, such as when the position of the workpiece to be gripped is variable (e.g. litter picking). In a collaborative environment, this is a serious security challenge that the whole system has to manage. The static parts of the environment can be checked regularly through collision detection, but the presence of the human means that 'simple' collision detection is not sufficient. This was the main reason for the current research and development presented in this paper.

ACKNOWLEDGEMENT

This research has been supported by the 'Thematic Excellence Program – National Challenges Subprogram – Establishment of the Center of Excellence for Autonomous Transport Systems at Széchenyi István University (TKP2020-NKA-14)' project and by the European Commission through the H2020 project EPIC (https://www.centre-epic.eu/) under grant no. 739592.

REFERENCES

[1] IFR Press Releases. Online [Accessed 16 August 2021] https://ifr.org/ifr-press-releases/news/robot-race-the-worlds-top-10-automated-countries
[2] European Commission, Digital Economy and Society Index Report 2019, Integration of Digital Technology. Online [Accessed 16 August 2021] https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=59979
[3] J. Shah, J. Wiken, B. Williams, C. Breazeal, Improved human-robot team performance using chaski, a human-inspired plan execution system, Proc. of the 6th Int. Conf. on Human-Robot Interaction, Lausanne Switzerland, 8-11 March 2011, pp. 29-36. DOI: 10.1145/1957656.1957668

[4] T. Gaspar, B. Ridge, R. Bevec, M. Bem, I. Kovač, A. Ude, Z. Gosar, Rapid hardware and software reconfiguration in a robotic workcell, Proc. of the 18th IEEE Int. Conf. on Advanced Robotics (ICAR), Hong Kong, China, 10-12 July 2017, pp. 229-236.
DOI: 10.1109/ICAR.2017.8023523

[5] ISO/TS 15066:2016, Robots and robotic devices - Collaborative robots. Online [Accessed 16 August 2021]
https://www.iso.org/standard/62996.html

[6] ISO 10218-1:2011, Robots and robotic devices - Safety requirements for industrial robots - Part 1: Robots. Online [Accessed 16 August 2021]
https://www.iso.org/standard/51330.html

[7] ISO 10218-2:2011, Robots and robotic devices - Safety requirements for industrial robots – Part 2: Robot systems and integration. Online [Accessed 16 August 2021]
https://www.iso.org/standard/41571.html

[8] A. Hentout, M. Aouache, A. Maoudj, I. Akli, Human–robot interaction in industrial collaborative robotics: a literature review of the decade 2008–2017, Advanced Robotics 33 (2019) pp. 764-799.
DOI: 10.1080/01691864.2019.1636714

[9] A. Zacharaki, I. Kostavelis, A. Gasteratos, I. Dokas, Safety bounds in human robot interaction: A survey, Safety Science 127 (2020) 104667.
DOI: 10.1016/j.ssci.2020.104667

[10] C. Morato, K. Kaipa, B. Zhao, S. K. Gupta, Safe Human Robot Interaction by Using Exteroceptive Sensing Based Human Modeling, Proc. of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 2A: 33rd Computers and Information in Engineering Conference. Portland, Oregon, USA. 4-7 August 2013, 10 pp.
DOI: 10.1115/DETC2013-13351

[11] European Commission, 2006/42/EC Machinery Directive. Online [Accessed 16 August 2021]
https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32006L0042

[12] European Commission, 2014/35/EU Low Voltage Directive. Online [Accessed 16 August 2021]
https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32014L0035

[13] COVR project database for directives and standards. Online [Accessed 16 August 2021]
https://www.safearoundrobots.com/toolkit/documentfinder

[14] ISO/TR 20218-1:2018 Robotics - Safety design for industrial robot systems - Part 1: End-effectors. Online [Accessed 16 August 2021]
https://www.iso.org/standard/69488.html

[15] EN ISO 13855:2010 Safety of machinery - Positioning of safeguards with respect to the approach speeds of parts of the human body. Online [Accessed 16 August 2021]
https://www.iso.org/standard/42845.html

[16] EN ISO 13849-1:2015 Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design. Online [Accessed 16 August 2021]
https://www.iso.org/standard/69883.html

[17] EN ISO 13849-2:2012 Safety of machinery - Safety-related parts of control systems - Part 2: Validation. Online [Accessed 16 August 2021]
https://www.iso.org/standard/53640.html

[18] EN ISO 12100:2010 Safety of machinery - General principles for design - Risk assessment and risk reduction. Online [Accessed 16 August 2021]
https://www.iso.org/standard/51528.html

[19] EN ISO 13850:2015 Safety of machinery - Emergency stop function - Principles for design. Online [Accessed 16 August 2021]
https://www.iso.org/standard/59970.html

[20] EN IEC 60204-1:2018 Safety of machinery - Electrical equipment of machines - Part 1: General requirements. Online [Accessed 16 August 2021]
https://standards.iteh.ai/catalog/standards/sist/e7d3ec34-16ab-476d-b979-1de5762a3ed7/sist-en-60204-1-2018

[21] EN IEC 62061:2005 Safety of machinery - Functional safety of safety-related electrical, electronic and programmable electronic control systems. Online [Accessed 16 August 2021]
https://standards.iteh.ai/catalog/standards/sist/4c933a51-d926-457b-b3da-4bfaef9908ac/sist-en-62061-2005

[22] EN ISO 11161:2007 Safety of machinery - Integrated manufacturing systems - Basic requirements. Online [Accessed 16 August 2021]
https://www.iso.org/standard/35996.html

[23] EN ISO 13854:2017 Safety of machinery - Minimum gaps to avoid crushing of parts of the human body. Online [Accessed 16 August 2021]
https://www.iso.org/standard/66459.html

[24] EN ISO 13857:2019 Safety of machinery - Safety distances to prevent hazard zones being reached by upper and lower limbs. Online [Accessed 16 August 2021]
https://www.iso.org/standard/69569.html

[25] EN ISO 14118:2017 Safety of machinery - Prevention of unexpected start-up. Online [Accessed 16 August 2021]
https://www.iso.org/standard/66460.html

[26] EN IEC 62046:2018 Safety of machinery - Application of protective equipment to detect the presence of persons. Online [Accessed 16 August 2021]
https://standards.iteh.ai/catalog/standards/sist/b62f0bb2-9011-413a-a717-caf55f66f289/sist-en-iec-62046-2018

[27] EN ISO 13851:2019 Safety of machinery - Two-hand control devices - Principles for design and selection. Online [Accessed 16 August 2021]
https://www.iso.org/standard/70295.html

[28] Universal Robots, URCap Software Platform of Universal Robots. Online [Accessed 16 August 2021]
https://www.universal-robots.com/

[29] RobotiQ website. Online [Accessed 16 August 2021]
www.robotiq.com

[30] Unity, Cross-platform game engine. Online [Accessed 16 August 2021]
https://unity.com

[31] Unreal Engine, Cross-platform game engine. Online [Accessed 16 August 2021]
https://www.unrealengine.com/en-US/

[32] ApertusVR Documentation, GitBook. Online [Accessed 16 August 2021]
https://apertus.gitbook.io/vr/

[33] PREHUROCO sample files on Github. Online [Accessed 16 August 2021]
https://github.com/MTASZTAKI/ApertusVR/tree/0.9.1/samples/collisionDetection

[34] Vibration API (Second Edition), W3C recommendation, 18 October 2016. Online [Accessed 16 August 2021]
https://www.w3.org/TR/vibration/

[35] Collision detection plugin, ApertusVR on Github. Online [Accessed 16 August 2021]
https://github.com/MTASZTAKI/ApertusVR/tree/0.9.1/plugins/physics/bulletPhysics

[36] Visualisation plugin, ApertusVR on Github. Online [Accessed 16 August 2021]
https://github.com/MTASZTAKI/ApertusVR/tree/0.9.1/plugins/render/ogreRender

[37] Kinect plugin, ApertusVR on Github. Online [Accessed 16 August 2021]
https://github.com/MTASZTAKI/ApertusVR/tree/0.9.1/plugins/track/body/kinect

[38] Websocket server plugin, ApertusVR on Github. Online [Accessed 16 August 2021]
https://github.com/MTASZTAKI/ApertusVR/tree/0.9.1/plugins/languageAPI/webSocketServer

[39] X3D loader plugin, ApertusVR on Github. Online [Accessed 16 August 2021] https://github.com/MTASZTAKI/ApertusVR/tree/0.9.1/plugins/languageAPI/jsAPI/nodeJsPlugin/js/plugins/x3dLoader

[40] NodeJS plugin, ApertusVR on Github. Online [Accessed 16 August 2021] https://github.com/MTASZTAKI/ApertusVR/tree/0.9.1/plugins/languageAPI/jsAPI/nodeJsPlugin

[41] Bluetooth keyboard performance test. Online [Accessed 16 August 2021] http://www.technical-direct.com/en/bluetooth-keyboard-performance-test/

[42] Interactivity test: examples from real 5G networks (part 3) . Online [Accessed 16 August 2021] https://www.rohde-schwarz.com/us/solutions/test-and-measurement/mobile-network-testing/stories-insights/article-interactivity-test-examples-from-real-5g-networks-part-3-_253380.html

[43] Jsonlist file of the simulated UR5 robot movement. Online [Accessed 16 August 2021] https://github.com/MTASZTAKI/ApertusVR/blob/89aefbc9b2a0e7524092b87d728ad539cfc0a856/plugins/languageAPI/jsAPI/nodeJsPlugin/js/plugins/httpSimulator/ur5.jsonlist